Copyright Information

Copyright © 2009 Internetwork Expert, Inc. All rights reserved.

The following publication, CCIE R&S Lab Workbook Volume I Version 5.0, was developed by Internetwork Expert, Inc. All rights reserved. No part of this publication may be reproduced or distributed in any form or by any means without the prior written permission of Internetwork Expert, Inc.

Cisco®, Cisco® Systems, CCIE, and Cisco Certified Internetwork Expert, are registered trademarks of Cisco® Systems, Inc. and/or its affiliates in the U.S. and certain countries.

All other products and company names are the trademarks, registered trademarks, and service marks of the respective owners. Throughout this manual, Internetwork Expert, Inc. has used its best efforts to distinguish proprietary trademarks from descriptive names by following the capitalization styles used by the manufacturer.

Disclaimer

The following publication, CCIE R&S Lab Workbook Volume I Version 5.0, is designed to assist candidates in the preparation for Cisco Systems' CCIE Routing & Switching Lab Exam. While every effort has been made to ensure that all material is as complete and accurate as possible, the enclosed material is presented on an "as is" basis. Neither the authors nor Internetwork Expert, Inc. assume any liability or responsibility to any person or entity with respect to loss or damages incurred from the information contained in this workbook.

This workbook was developed by Internetwork Expert, Inc. and is an original work of the aforementioned authors. Any similarities between material presented in this workbook and actual CCIE lab material is completely coincidental.

Table of Contents

Quality of Service 1
10.1 Hold-Queue and Tx-Ring..1
10.2 Weighted Fair Queuing (WFQ)..1
10.3 Legacy RTP Reserved Queue...1
10.4 Legacy RTP Prioritization 1
10.5 Legacy Custom Queueing 2
10.6 Legacy Custom Queueing with Prioritization 2
10.7 Legacy Priority Queueing 2
10.8 Legacy Random Early Detection 2
10.9 Legacy Flow-Based Random Early Detection 3
10.10 Selective Packet Discard..3
10.11 Payload Compression on Serial Links 3
10.12 Generic TCP/UDP Header Compression 3
10.13 MLP Link Fragmentation and Interleaving 3
10.14 Legacy Generic Traffic Shaping 4
10.15 Legacy CAR for Admission Control 4
10.16 Oversubscription with Legacy CAR and WFQ 4
10.17 Legacy CAR for Rate Limiting 4
10.18 Legacy CAR Access-Lists 4
10.19 Legacy GTS for Frame Relay...5
10.20 Legacy Frame Relay Traffic Shaping 5
10.21 Legacy Adaptive FRTS...5
10.22 Legacy FRTS with Per-VC WFQ 5
10.23 Legacy FRTS with Per-VC PQ 6
10.24 Legacy FRTS with Per-VC CQ 6
10.25 Legacy FRTS with Per-VC Fragmentation 6
10.26 Legacy FRTS with Per-VC IP RTP Priority.....................................6
10.27 Frame-Relay RTP/TCP Header Compression................................6
10.28 Frame-Relay Broadcast Queue..7
10.29 Frame-Relay DE Marking 7
10.30 Legacy FRTS PVC Interface Priority Queue 7
10.31 Frame-Relay Priority to DLCI Mapping...7
10.32 Frame-Relay Traffic Policing & Congestion Mgmt 8
10.33 MQC Classification and Marking 9
10.34 MQC Bandwidth Reservations and CBWFQ 9
10.35 MQC Bandwidth Percent 9
10.36 MQC LLQ and Remaining Bandwidth Reservations.....................10
10.37 MQC WRED 10
10.38 MQC Dynamic Flows and WRED 10
10.39 MQC WRED with ECN 10
10.40 MQC Class-Based Generic Traffic Shaping 10
10.41 MQC Class-Based GTS and CBWFQ 11
10.42 MQC Single-Rate Three-Color Policer 11

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.43 MQC Hierarchical Policers 11
10.44 MQC Two-Rate Three-Color Policer 12
10.45 MQC Peak Shaping 12
10.46 MQC Percent-Based Policing 12
10.47 MQC Header Compression 12
10.48 Using Class-Based GTS for FRTS 13
10.49 MQC Based Frame-Relay DE-Marking 13
10.50 Using MQC CBWFQ with Legacy FRTS 13
10.51 MQC Compatible FRF.12 Fragmentation 13
10.51 MQC Based Frame-Relay Traffic Shaping 14
10.53 Voice Adaptive Traffic Shaping 14
10.53 Voice Adaptive Fragmentation 14
10.54 MLPPP LFI over Frame Relay 15
10.56 QoS Pre-Classify 15
10.57 RSVP and WFQ 16
10.58 RSVP and SBM 16
10.59 RSVP and CBWFQ 16
10.60 RSVP and LLQ 16
10.61 RSVP and Per-VC WFQ 17
10.62 Catalyst QoS Port-Based Classification 17
10.63 Catalyst QoS Marking Pass-Through 17
10.64 Catalyst QoS ACL Based Classification & Marking 18
10.65 Catalyst 3550 Per-Port Per-VLAN Classification 18
10.66 Catalyst 3560 Per-VLAN Classification 18
10.67 Catalyst QoS Port-Based Policing and Marking 19
10.68 Catalyst 3560 Per-Port Per-VLAN Policing 19
10.69 Catalyst 3550 Per-Port Per-VLAN Policing 19
10.70 Catalyst QoS Aggregate Policers 20
10.71 Catalyst 3560 Ingress Queueing 20
10.72 Catalyst 3560 Ingress Queue Tuning 20
10.73 Catalyst 3550 Egress Queueing 21
10.74 Catalyst 3550 Regular Queues Tuning 21
10.75 Catalyst 3550 Gigabit Interface Queues Tuning 21
10.76 Catalyst 3550 Egress Policing 22
10.77 Catalyst 3560 SRR Shared Mode 22
10.78 Catalyst 3560 SRR Shaped Mode 22
10.79 Catalyst 3560 Egress Queues Tuning 23
10.80 Catalyst QoS DSCP Mutation 23
10.81 Advanced HTTP Classification with NBAR 23

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Quality of Service Solutions.. 25
10.1 Hold-Queue and Tx-Ring..25
10.2 Weighted Fair Queuing (WFQ)..28
10.3 Legacy RTP Reserved Queue..38
10.4 Legacy RTP Prioritization..44
10.5 Legacy Custom Queueing..47
10.6 Legacy Custom Queueing with Prioritization..............................55
10.7 Legacy Priority Queueing..58
10.8 Legacy Random Early Detection..64
10.9 Legacy Flow-Based Random Early Detection.............................71
10.10 Selective Packet Discard..73
10.11 Payload Compression on Serial Links......................................77
10.12 Generic TCP/UDP Header Compression...................................82
10.13 MLP Link Fragmentation and Interleaving.................................86
10.14 Legacy Generic Traffic Shaping..93
10.15 Legacy CAR for Admission Control...102
10.16 Oversubscription with Legacy CAR and WFQ...........................108
10.17 Legacy CAR for Rate Limiting..118
10.18 Legacy CAR Access-Lists..122
10.19 Legacy GTS for Frame Relay...126
10.20 Legacy Frame Relay Traffic Shaping.......................................130
10.21 Legacy Adaptive FRTS..134
10.22 Legacy FRTS with Per-VC WFQ...137
10.23 Legacy FRTS with Per-VC PQ...139
10.24 Legacy FRTS with Per-VC CQ...142
10.25 Legacy FRTS with Per-VC Fragmentation................................145
10.26 Legacy FRTS with Per-VC IP RTP Priority...............................152
10.27 Frame-Relay RTP/TCP Header Compression...........................157
10.28 Frame-Relay Broadcast Queue...159
10.29 Frame-Relay DE Marking...160
10.30 Legacy FRTS PVC Interface Priority Queue.............................162
10.31 Frame-Relay Priority to DLCI Mapping....................................164
10.32 Frame-Relay Traffic Policing & Congestion Mgmt.....................167
10.33 MQC Classification and Marking...176
10.34 MQC Bandwidth Reservations and CBWFQ.............................181
10.35 MQC Bandwidth Percent...192
10.36 MQC LLQ and Remaining Bandwidth Reservations...................196
10.37 MQC WRED...203
10.38 MQC Dynamic Flows and WRED..206
10.39 MQC WRED with ECN..209
10.40 MQC Class-Based Generic Traffic Shaping..............................212
10.41 MQC Class-Based GTS and CBWFQ......................................216
10.42 MQC Single-Rate Three-Color Policer.....................................224
10.43 MQC Hierarchical Policers...231
10.44 MQC Two-Rate Three-Color Policer..238

Copyright © 2009 Internetwork Expert

10.45 MQC Peak Shaping 245
10.46 MQC Percent-Based Policing 251
10.47 MQC Header Compression 253
10.48 Using Class-Based GTS for FRTS 256
10.49 MQC Based Frame-Relay DE-Marking 260
10.50 Using MQC CBWFQ with Legacy FRTS 262
10.51 MQC Compatible FRF.12 Fragmentation 266
10.52 MQC Based Frame-Relay Traffic Shaping 268
10.53 Voice Adaptive Traffic Shaping 272
10.54 Voice Adaptive Fragmentation 277
10.55 MLPPP LFI over Frame Relay 280
10.56 QoS Pre-Classify 287
10.57 RSVP and WFQ 291
10.58 RSVP and SBM 305
10.59 RSVP and CBWFQ 308
10.60 RSVP and LLQ 316
10.61 RSVP and Per-VC WFQ 321
10.62 Catalyst QoS Port-Based Classification 326
10.63 Catalyst QoS Marking Pass-Through 340
10.64 Catalyst QoS ACL Based Classification & Marking 345
10.65 Catalyst 3550 Per-Port Per-VLAN Classification 355
10.66 Catalyst 3560 Per-VLAN Classification 359
10.67 Catalyst QoS Port-Based Policing and Marking 364
10.68 Catalyst 3560 Per-Port Per-VLAN Policing 374
10.68 Catalyst 3560 Per-Port Per-VLAN Policing 374
10.69 Catalyst 3550 Per-Port Per-VLAN Policing 378
10.70 Catalyst QoS Aggregate Policers 383
10.71 Catalyst 3560 Ingress Queueing 390
10.72 Catalyst 3560 Ingress Queue Tuning 394
10.73 Catalyst 3550 Egress Queueing 397
10.74 Catalyst 3550 Regular Queues Tuning 403
10.75 Catalyst 3550 Gigabit Interface Queues Tuning 405
10.76 Catalyst 3550 Egress Policing 410
10.77 Catalyst 3560 SRR Shared Mode 413
10.78 Catalyst 3560 SRR Shaped Mode 421
10.79 Catalyst 3560 Egress Queues Tuning 426
10.80 Catalyst QoS DSCP Mutation 431
10.81 Advanced HTTP Classification with NBAR 434

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Quality of Service

Note

Load the *QoS* initial configurations prior to starting.

10.1 Hold-Queue and Tx-Ring

- Configure R1’s connection to VLAN 146 to have an input software queue length of 10 packets, and an output software queue length of 30 packets.
- Set the output hardware queue size to 15 packets.

10.2 Weighted Fair Queuing (WFQ)

- Configure the point-to-point Serial link between R4 and R5 with an interface clock rate and interface bandwidth of 128Kbps.
- Set the output hold-queue size to 256.
- Configure WFQ on the links with a length of 16 for the congestive discard threshold, 128 for the number of conversations, and 8 for RSVP reservable queues.
- Set the tx-ring size to the minimal value to engage the WFQ as fast as possible.
- Normalize the packet flows for the link by ajdusting the MTU so that each IP packet takes no more than 10ms to be sent.

10.3 Legacy RTP Reserved Queue

- Reset R4’s Serial link to R5 to the default fair-queue values.
- Configure a single command on R4’s interface so that 100% of the link bandwidth is reserved for RTP traffic in the UDP port range 16384 – 32767.

10.4 Legacy RTP Prioritization

- Reset R4’s Serial link to R5 to the default fair-queue values, and remove the RTP reservation.
- Configure legacy RTP prioritization for UDP ports in the range 16383 – 32767 up to 128Kbps.

Copyright © 2009 Internetwork Expert

10.5 Legacy Custom Queueing

- Remove R4's WFQ configuration on the Serial link connecting to R5.
- Configure legacy custom queueing on this link so that traffic is round-robin scheduled based on the following requirements:
    - RTP VoIP traffic should be guaranteed 30% of the link bandwidth.
    - File transfers from web servers in VLAN 146 should be guaranteed 60% of the link bandwidth.
    - The remaining 10% should be allocated to ICMP, but this traffic should not exceed 10 packets in the queue at any given time.
    - RIP routing packets should use the system priority queue.
- Clock the link at 128Kbps, and ensure that IP packets larger than 156 bytes are fragmented.
- Assume packet sizes of 60 bytes for RTP, 156 bytes for TCP, and 100 bytes for ICMP.
- Ensure to account for the layer 2 HDLC encapsulation overhead.

10.6 Legacy Custom Queueing with Prioritization

- Modify R4's custom queueing configuration so that the RTP VoIP traffic is prioritized after the RIP packets.

10.7 Legacy Priority Queueing

- Remove R4's custom queueing configuration and replace it with legacy priority queueing per the following requirements:
    - All RIP packets going out the Serial link to R5 should be sent first.
    - If there are no RIP packets, RTP VoIP should be sent next.
    - If there are no RIP or VoIP packets, web traffic should be sent next.
    - As a last resort ICMP traffic should be sent.
- Set the queue-sizes for the high, medium, normal, and low queues to 5, 40, 60, and 80 packets respectively.

10.8 Legacy Random Early Detection

- Configure R4's point-to-point Serial interface connecting to R5 to avoid tail drop behavior by randomly dropping packets before the output queue overflows.
- Set the hold-queue size to 10 packets, and the weighted averaging constant to 4.
- Ensure that routing updates are never randomly dropped.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.9 Legacy Flow-Based Random Early Detection

- Enable flow-based RED on the point-to-point Serial interface of R4.
- Set the FIFO queue depth to 10 packets.
- Set the average flow depth scale factor to 2.
- Set the maximum number of flows to 16.

10.10 Selective Packet Discard

- Enable Selective Packet Discard on R1 in aggressive mode.
- Increase R1's input queue size on its link to VLAN 146 to twice the default.
- Increase the amount of the memory headroom for IGP packets to 150 buffers.
- The headroom for BGP packets should be set to 120 packets.
- Start dropping low-priority packets randomly when the input queue is 50% full.

10.11 Payload Compression on Serial Links

- Configure PPP on the link between R1 and R3 using Predictor compression.
- Enable Stacker compression on the HDLC link between R2 and R3.
- Enable FRF.9 compression on the Frame Relay PVC between R2 and R5.

10.12 Generic TCP/UDP Header Compression

- Optimize the bandwidth usage between R4 and R5 by reducing TCP and RTP header sizes.
- Allow for a maximum of 16 concurrent RTP and TCP sessions.
- R5 should optimize traffic only if it detects already optimized traffic from R4.

10.13 MLP Link Fragmentation and Interleaving

- Enable PPP encapsulation on the serial link between R4 and R5.
- Configure PPP Multilink with interleaving on this connection.
- Ensure the maximum serialization delay is 10ms.
- Assume the bandwidth of the link is 128Kbps.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.14 Legacy Generic Traffic Shaping

- Configure shaping on R6 to limit the rate of packets going towards R4's Loopback0 via the connection to VLAN 146 to 128Kbps.
- Use a shaping interval of 10ms and disable the extended burst functionality.
- Limit the shaper's queue size to 1024 packets.

10.15 Legacy CAR for Admission Control

- Configure R4 to mark traffic received from R1's IP address 155.X.146.1 as it enters R4's connection to VLAN 146 using legacy committed access rate.
- Traffic up to 256Kbps should be marked with an IP precedence of 1.
- Exceeding traffic should be marked with an IP precedence of 0.
- Use an average traffic burst size of 4000 bytes.

10.16 Oversubscription with Legacy CAR and WFQ

- Configure WFQ on R4's point-to-point link to R5, and clock the link at 128Kbps.
- Configure a CAR policy on R4 to achieve the following:
    - 64Kbps that R4 receives from R1 and R6 each should be guaranteed on the link out to R5.
    - Traffic from R1 and R6 should be allowed up to 128Kbps each total.
    - Traffic above 128Kbps should be dropped.
    - Use the averaging time interval of 200ms.

10.17 Legacy CAR for Rate Limiting

- Configure R6 to drop traffic received in its link to VLAN 146 in excess of 256Kbps.
- Use a committed burst of 384Kbps and an excess burst of 768Kbps.

10.18 Legacy CAR Access-Lists

- Configure R1 to rate limit packets going out its link to VLAN 146 towards the MAC address of R4 to 128Kbps.
- Limit the rate of packets leaving R6 towards SW1 having IP precedence values of one, two, and four to 256Kbps.

10.19 Legacy GTS for Frame Relay

- R3's Frame Relay service provider has agreed to transport up to 128Kbps of incoming traffic, but all traffic that exceeds 96Kbps is marked as Discard Eligible.
- Configure R3 with GTS to slow down to the guaranteed rate if the service provider signals congestion with BECN frames.
- Use 30ms for the traffic-shaping interval.

10.20 Legacy Frame Relay Traffic Shaping

- R2, R3, and R5's physical connection rates to the Frame Relay network are 128Kbps, 384Kbps, and 1536Kbps respectively.
- R3's contracted rate is 256Kbps, but the service provider allows occasional packet bursts up to the physical line capacity without discarding them; the provider does not allow sustained oversubscription by R3.
- Configure Legacy Frame Relay Traffic Shaping on R3 and R5 to meet the following requirements:
  - Ensure that R5 does not overwhelm R2's or R3's physical connections.
  - Ensure that R3 adheres to its service contract.
  - Use a Tc value of 10ms to support a VoIP deployment planned in the near future.

10.21 Legacy Adaptive FRTS

- The service provider used by R3 has agreed to allow link oversubscription up to 384Kbps, but only guarantees delivery up to 256Kbps.
- Configure R3 to send as fast as possible on this circuit, but to slow down its sending rate to the guaranteed rate when service provider network notifies it of congestion or the physical interface is congested.
- Ensure that R5 can inform R3 about congestion in the case that R5 is receiving a unidirectional traffic feed from R3.

10.22 Legacy FRTS with Per-VC WFQ

- Configure WFQ in R3's map-class for the PVC towards R5.
- Set the number of flow queues to 32, the queue size to 512, and start discarding packets when any flow reaches a length of 16 packets.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.23 Legacy FRTS with Per-VC PQ

- Configure R5 to prioritize HTTP traffic sent to R3 over the Frame Relay link, but only if there are no RIP routing updates to send.
- ICMP traffic should only be sent if there is no other traffic queued on the link.

10.24 Legacy FRTS with Per-VC CQ

- Configure R5 to share the bandwidth for the VC connecting R5 to R2 in the following proportions:
    - 60% of bandwidth for HTTP traffic
    - 20% of bandwidth for telnet traffic
    - 10% of bandwidth for ICMP traffic
    - RIP should use the system queue
- Limit each queue size to 15 packets.
- Ensure all protocols use the same average packet size of 80 bytes.

10.25 Legacy FRTS with Per-VC Fragmentation

- Remove any artificial IP MTU settings from the Frame Relay interfaces of R3 and R5 along with any per-VC queueing strategies.
- The connection between R3 and R5 is used to send data and voice traffic.
- Configure Frame Relay fragmentation so that voice packets are never delayed for more than 10ms when serialized.
- Assume the speed of R3’s connection is 384Kbps
- Ensure R2 is not affected by this configuration.

10.26 Legacy FRTS with Per-VC IP RTP Priority

- Modify R3 and R5’s FRTS configuration to prioritize RTP VoIP traffic on the PVC between them.
- Allocate 128Kbps of bandwidth to voice traffic in the port range of 16384 – 32767.

10.27 Frame-Relay RTP/TCP Header Compression

- Configure R3 and R5 to compress RTP and TCP headers on the PVC between them.
- Compression should not be enabled on any other PVCs.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.28 Frame-Relay Broadcast Queue

- Modify R5's Frame Relay broadcast queue per the following requirements:
    - Limit the broadcast queue rate a maximum of 128Kbps
    - Limit the queue depth to 4 packets per spoke
    - Limit the total number of broadcast packets to 10 per second

10.29 Frame-Relay DE Marking

- Configure R3 to mark all IP packets larger than 60 bytes as Discard Eligible as they are sent out to the Frame Relay cloud.

10.30 Legacy FRTS PVC Interface Priority Queue

- Configure PIPQ on R5 so that frames going out to R3 are always sent first, followed by frames going to R2.
- All other PVCs assigned to R5's interface should be serviced after R3 and R2.

10.31 Frame-Relay Priority to DLCI Mapping

- Configure back-to-back Frame Relay on the Serial link between R4 and R5 using DLCIs 100 and 200.
- Configure a static mapping to R4 and R5's IP addresses using DLCI 100 on both sides.
- Create a legacy priority-list on both R4 and R5 that maps RIP updates and packets less than 65 bytes to the high queue.
- Map packets from the high queue to the DLCI 100 on both sides.
- Map all other queues to DLCI 200 on both sides.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.32 Frame-Relay Traffic Policing & Congestion Mgmt

- Enable Frame Relay switching on R3.
- Configure R3's links to R1 and R2 as Frame Relay DCE interfaces, and clock the links at 128Kbps.
- Configure R3 to terminate DLCI 133 to R1 using the IP address 155.X.13.3/24.
- Configure R3 to terminate DLCI 233 to R2 using the IP address 155.X.23.3/24.
- Configure R3 to switch DLCI 132 from R1 to R2 as DLCI 231.
- Configure two point-to-point subinterfaces on R1's link to R3.
- Configure R1 to use the IP address 155.X.13.1/24 and DLCI 133 for the first subinterface, and IP address 155.X.12.1/24 and DLCI 132 for the second subinterface.
- Configure two point-to-point subinterfaces on R2's link to R3.
- Configure R2 to use the IP address 155.X.23.2/24 and DLCI 233 for the first subinterface, and IP address 155.X.12.2/24 and DLCI 231 for the second subinterface.
- Ensure that R3 has IP connectivity to R1 and R2, and R1 and R2 have IP connectivity to each other over these circuits.
- Create two map-classes on R3 for the switched DLCIs with the following policies:
  - The class for R1 should be configured for an input CIR of 64Kbps, an input B_c and B_e of 8000 bits, and an input T_c of 125ms
  - The class for R2 should be configured for an input CIR zero, an input B_c of zero, an input B_e of 16000 bits, and an input T_c of 125ms
- Enable Frame Relay traffic-policing on R3's links to R1 and R2 and apply the above classes to the respective DLCIs.
- Configure generic traffic-shaping on R1 and R2 to average 128Kbps with a B_c of 16000 bits.
- R1 should reduce its sending rate to the guaranteed rate in the case that BECNs are received.
- Enable congestion management on R3's switched PVCs and set the thresholds for DE dropping to 5% and FECN/BECN marking to 3%.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Note

Reset all devices to the *QoS* initial configurations before proceeding.

10.33 MQC Classification and Marking

- Configure an outbound MQC policy on R4's Serial link to R5 per the following requirements:
    - WWW traffic from servers on VLAN 146 should be marked with an IP Precedence of 2
    - VoIP packets with UDP ports in the destination range of 16384 - 32767 and a Layer 3 packet size of 60 bytes should be marked with DSCP EF
    - ICMP packets larger than 1000 bytes should be dropped
    - All other packets that came from R4's connection to VLAN 146 with an IP precedence of 0 should be remarked with an IP precedence of 1
- Do not use an access-list to classify ICMP packets.

10.34 MQC Bandwidth Reservations and CBWFQ

- Modify R4's MQC policy to meet the following requirements:
    - Set the total size of the MQC queue to 512 buffers
    - The traffic flows for the web servers in VLAN 146 and the IP Precedence 0 packets from VLAN 146 should be guaranteed 32Kbps each
    - Limit the sizes of the FIFO queues for the HTTP and IP Precedence 0 traffic classes to 16 and 24 packets respectively
    - All other unmatched traffic in the policy should run WFQ
    - Dynamic flows in this WFQ should start dropping when they reach 32 packets in length

10.35 MQC Bandwidth Percent

- Modify R4's previous user-defined reservations to use a percentage reservation of 25% of the link bandwidth each, as compared to 32Kbps.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.36 MQC LLQ and Remaining Bandwidth Reservations

- Modify R4's policy to R5 as follows:
    - Configure Low Latency Queueing to allow for exactly one VoIP call to be prioritized
    - Additional VoIP calls should be allowed only if the link is not congested, but they should not be prioritized
    - Reserve 33% of the remaining bandwidth to the HTTP and SCAVENGER classes each
- Assume that VoIP calls are using the G.729 codec, which generates 50 packets per second with a Layer 3 size of 60 bytes.

10.37 MQC WRED

- Modify R4's policy so that the HTTP traffic class uses random detection for dropping as opposed to tail drop.
- Start dropping packets randomly when the average queue size is between 4 and 16.
- Drop 1 of every 4 packets when the average queue size reaches the maximum threshold.

10.38 MQC Dynamic Flows and WRED

- Modify the CBWFQ configuration on R4 to activate random drops for unclassified traffic's dynamic flows.
- Change the minimum and maximum RED thresholds for traffic with an IP precedence of one to 1 and 40 respectively.
- As the queue depth grows close to the maximum threshold, the probability of packet discard should be 25%.

10.39 MQC WRED with ECN

- Modify the HTTP traffic class on R4 so that explicit congestion notification for TCP is used for RED dropping.

10.40 MQC Class-Based Generic Traffic Shaping

- Configure MQC shaping on R6 to limit the sending rate on its link to VLAN 146 to 512Kbps.
- The link to VLAN 67 should be limited to 384Kbps.
- Use a burst interval of 20ms.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.41 MQC Class-Based GTS and CBWFQ

- Change the queue for the shaper applied to R6's connection to VLAN 146 as follows.
    - Provide 32Kbps of priority treatment for small packets with a layer three size of 60 bytes (voice packets) using a 400 byte burst value
    - Guarantee the HTTP traffic 256Kbps of the shaper's bandwidth
    - Unclassified traffic should receive fair-queue treatment

Note

Reset all devices to the *QoS* initial configurations before proceeding.

10.42 MQC Single-Rate Three-Color Policer

- Configure R4 to meter incoming HTTP traffic on its link to VLAN 146 as follows.
    - If the traffic rate is less than 128Kbps, mark the packets with an IP precedence of one
    - If the traffic exceeds 128Kbps, mark the packets with an IP precedence of zero
    - Drop violating traffic
- Ensure the burst size is large enough to accommodate normal and excess burst durations of 200ms and 300ms at a rate of 128Kbps.

10.43 MQC Hierarchical Policers

- Configure R4 to limit the aggregate rate of HTTP traffic entering the connection to VLAN 146 to 128Kbps.
- Transmit conforming packets, mark exceeding packets with an IP precedence of zero, and drop violating packets.
- For HTTP traffic flows from R1 and R6, limit the rate to 64Kbps for each flow.
- For these second-level policers, set the conform action to set the IP precedence to 1 and transmit, the exceed action to set the IP precedence to 0 and transmit, and the violate action to set the IP precedence to 0 and transmit.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.44 MQC Two-Rate Three-Color Policer

- Modify the first-level policy-map applied directly to R4's connection to VLAN 146 to remove the aggregate policer of 128Kbps.
- Modify the second-level policy-map to replace the single-rate policers with two-rate policers.
- Use the CIR value of 64Kbps and PIR value of 128Kbps.
- Use the values of CIR*400ms and PIR*200ms for normal and excess burst sizes.
- Set the conform action to set IP precedence 1 and transmit, the exceed action to set IP precedence of 0 and transmit, and the violate action to drop.

10.45 MQC Peak Shaping

- Configure R1 and R6 to shape HTTP traffic to a peak rate of 128Kbps as it is sent out to VLAN 146.
- Use B_c and B_e bursts based on a 10ms interval.

10.46 MQC Percent-Based Policing

- Hard code R1's FastEthernet link to a speed of 10Mbps.
- Limit the traffic entering this link to 10% of this rate with a burst value of 125ms.

10.47 MQC Header Compression

- Enable MQC based RTP header compression on the serial link between R1 and R3.
- Assume that RTP packets use the port range 16384-32767.
- Allocate enough priority bandwidth to allow 2 G.729 compressed calls over this link, for a total of 24Kbps.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.48 Using Class-Based GTS for FRTS

- Configure R3 and R5 to shape traffic going out to the Frame Relay network using MQC based GTS.
- R3's policy should match PVC 305, and shape to an average rate of 256Kbps.
- R5's policy should match PVC 503 with a peak shaping rate of 128Kbps, and match PVC 502 with a peak shaping rate of 256Kbps.
- R3 should use a Bc and Be of 12800 and 6400 for the shaper to R5.
- R5 should use a Bc and Be of 6400 for the shaper to R3, and a Bc and Be of 12800 for the shaper to R2.

10.49 MQC Based Frame-Relay DE-Marking

- Modify R5's shaper so that the Frame Relay Discard Eligible bit is set on all frames going out DLCI 502.

10.50 Using MQC CBWFQ with Legacy FRTS

- Configure legacy FRTS on R2's connection to DLCI 205 as follows:
  - Shape to the average rate of 512Kbps
  - Use an interval of 10ms
  - Adaptively shape down to 256Kbps if BECNs are received
- Configure an MQC policy inside this legacy shaper as follows:
  - Guarantee 64Kbps of PVC bandwidth to HTTP flows
  - Other unmatched flows should be guaranteed 192Kbps

10.51 MQC Compatible FRF.12 Fragmentation

- Configure R2, R3, and R5 to fragment packets to 480 bytes as they are sent out to the Frame Relay network.
- Enable fragmentation on the main Frame-Relay interfaces of R3 and R4.
- Enable fragmentation under the map-class for DLCI 205 on R2.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.51 MQC Based Frame-Relay Traffic Shaping

- Modify the traffic-shaping configuration on R5 and remove the classes matching the Frame-Relay DLCIs.
- Configure MQC shaping to apply per-VC as follows, but do not enable legacy FRTS on the interface
  - Shape peak with CIR values 128000 and 256000 for DLCI 503 and DLCI 502
  - Select a B_c value based on a T_c of 50ms, and a B_e value so that PIR=2*CIR
  - Set the Frame-Relay DE bit for traffic going out DLCI 502
- Replace the interface-based fragmentation with per-VC fragmentation.

10.53 Voice Adaptive Traffic Shaping

- Allocate 64Kbps of priority bandwidth to voice traffic on the PVCs connecting R2, R3 and R5. Classify voice traffic based on a DSCP value of EF
- Since the provider oversubscribes both PVCs on R5's interface, voice quality may degrade when sending traffic at rates over the CIR
- However, it is undesirable to turn off oversubscription and lose extra bandwidth
- Provide a solution to throttle down the sending rate on R3 and R5 down to the CIR only when the IOS detects traffic in the LLQ queue
- Leave the shaping time interval the same at 50ms for simplicity

10.53 Voice Adaptive Fragmentation

- Configure R5 to activate fragmentation only if voice packets are present in the LLQ queue of the respective PVCs

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.54 MLPPP LFI over Frame Relay

- Configure the link between R4 and R5 for Frame-Relay encapsulation. Use any method of your choice
- The company plans to interwork the PVC between R4 and R5 with an ATM connection and run VoIP traffic over both segments
- Suggest a solution that provides a fragmentation and interleaving scheme common for both ATM and Frame-Relay technologies and implement it over the Frame-Relay link between R4 and R5
- Assume the Frame-Relay PVC CIR of 256Kbps and interface rate of 512Kbps
- Provide enough priority bandwidth to accommodate two G.729 calls and guarantee 5% of PVC bandwidth to voice signaling traffic
- Allocate the remaining bandwidth to unallocated traffic
- Classify voice bearer and voice signaling traffic based on DSCP values of EF and CS3 respectively
- Choose the shaping interval optimal for VoIP traffic

10.56 QoS Pre-Classify

- Create a GRE tunnel between R1 and R6 over VLAN 146.
- Route traffic between the Loopback0 subnets of R1 and R6 across the tunnel using static routes.
- Limit the rate of traffic leaving the VLAN 146 interface of R6 to 256Kbps.
- Configure R6 to guarantee 64Kbps of priority bandwidth to IP traffic between the Loopback0 subnets of R1 and R6 marked with DSCP EF on the VLAN146 interface.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Note

Reset all devices to the *QoS* initial configurations before proceeding.

10.57 RSVP and WFQ

- Shutdown R5's Frame Relay connection.
- Configure the link bandwidth between R4 and R5 as 128Kbps.
- Enable RSVP on the connections to VLAN 146 on R1, R4, and R6.
- Allow RSVP to use the maximum possible bandwidth on the link between R4 and R5, but do not allow it to reserve more than 64Kbps per flow.

10.58 RSVP and SBM

- Configure R1 as the DSBM candidate to coordinate the bandwidth usage on the segment shared by R1, R4, and R6.
- Allocate just 1024Kbps to RSVP flows on this segment
- Configure R1 to inform other routers on the segment that they may not send above 50Kbyte/sec of non-RSVP traffic into the segment. Limit the burst size to 5Kbyte.

10.59 RSVP and CBWFQ

- Configure CBWFQ on the Serial link of R5 to ensure that HTTP server replies have a guarantee of all interface bandwidth not reserved to RSVP flows.
- Allow for 64 dynamic flow queues with CBWFQ.

10.60 RSVP and LLQ

- Configure R5 to classify RSVP flows with a PQ-profile that ignores peak-rate, and classifies all flows below 2000Kbyte/s with the burst sizes below or equal to 2000 bytes as eligible for LLQ.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.61 RSVP and Per-VC WFQ

- Configure the link between R4 and R5 for Frame-Relay encapsulation using DLCI 100.
- Enable legacy Frame-Relay traffic shaping on R5 and shape DLCI 100 down to 96Kbps, with the burst size corresponding to time-interval of 10ms.
- Allow RSVP to use up to 96Kbps of interface bandwidth with 64Kbps per-flow.
- Enable WFQ as the VC queue and configure per-VC WFQ as the RSVP resource provider.

10.62 Catalyst QoS Port-Based Classification

- Enable MLS QoS in all switches.
- Configure CoS to DSCP maps on SW1 and SW2 so that CoS 3 maps to DSCP 26 and CoS 5 maps to DSCP EF.
- Ensure CoS 4 maps to DSCP 44 on SW1.
- Configure IP Precedence to DSCP map in SW4 to map IP Precedence values 3 and 5 into DSCP 26 and DSCP EF.
- Trust DSCP values in packets coming from R1's interface connected to SW1.
- Trust IP Precedence on SW4 port connected to R4 and set the default CoS to 2.
- Override CoS to a value of four on the port connected to R5's VLAN58 interface.
- Use 802.1p bits in the Ethernet headers for classification of packets coming from R6; packets on native VLAN should use a CoS value of one.
- Use interface-level commands only to accomplish the task.

10.63 Catalyst QoS Marking Pass-Through

- Ensure SW2 classifies packets coming from R6's VLANs 67 and VLAN 146 based on CoS values, but does not change the DSCP value in IP packets.
- Additionally ensure that IP-precedence based classification on the SW4's connection to R4 does not change the CoS value in Ethernet headers.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.64 Catalyst QoS ACL Based Classification & Marking

- Create a policy-map on SW2 that performs the following on traffic received from R6.
    - Set a DSCP value of EF on IPX packets (EtherType value 0x8137)
    - Set a DSCP value of CS3 on ICMP packets
    - Set an IP Precedence of 2 on TELNET packets
    - Trust CoS under the default class
- Configure SW4 and R4 to change the port connected to R4 into a trunk.
- Configure SW4 to trust IP DSCP values in packets coming from R4, and set the CoS to a value of 2.

10.65 Catalyst 3550 Per-Port Per-VLAN Classification

- Configure SW4 so that ICMP traffic coming in on the port connected to R4 has the DSCP value set to CS3.
- For IPX traffic with EtherType 0x8137 set the CoS field to 2.
- Ensure your classification only affects traffic on VLAN 146 but not any VLANs trunked to R4 in the future.

10.66 Catalyst 3560 Per-VLAN Classification

- Enable VLAN-based QoS on the trunk ports of SW1 and on the port connected to R1.
- Configure SW1 to mark all TCP traffic on VLAN 146 with a DSCP value of EF.
- IPX SNAP-encapsulated traffic should be marked with a CoS value of 4.

10.67 Catalyst QoS Port-Based Policing and Marking

- Configure SW4's port connected to R4 as follows:
  - Limit the input rate of ICMP packets to 64Kbps. Meter using a burst size value of 16Kbytes. Trust DSCP values in conforming packets and re-mark exceeding traffic down to CS1.
  - Irrespective of the DSCP value, use a CoS value of two for ICMP packets entering the switch.
  - Mark traffic from WWW servers with DSCP CS2. If the traffic exceeds 256 Kbps, re-mark it down to CS1. Assume the burst size of 32000 bytes.
  - Meter the rate of non-IP traffic at 128Kbps using the minimum burst size. Set a CoS value of 0 for conforming traffic, and re-mark exceeding packets with a CoS value of 1.
  - Assume that incoming ICMP packets are pre-colored either with DSCP AF31 or DSCP CS3.
- Configure SW1 to limit the total rate of traffic coming from R1 to 128Kbps. Use a large enough burst size to accommodate 10ms of traffic generated by R1 transmitting at the full 100Mbps interface rate.

10.68 Catalyst 3560 Per-Port Per-VLAN Policing

- Remove any previous policing configuration on VLAN146 in SW2.
- Configure SW2 to limit the rate of IP traffic entering VLAN146 on every trunk port to 128Kbps.
- Mark all conforming packets with AF21 and exceeding with CS1.
- Use the burst size of twice the minimal size.
- Restrict the amount of IP traffic entering from R6 on VLAN146 to 256Kbps, and drop any exceeding packets. Use the burst size of 32000 bytes.

10.69 Catalyst 3550 Per-Port Per-VLAN Policing

- Remove any previously attached service-policy on the port connected to the VLAN 146 interface of SW4
- Limit the rate of IP traffic from R4 on VLAN 146 to 128Kbps. Mark conforming packets as AF11 and exceeding as CS1
- Limit the rate of non-IP traffic from R4 on VLAN 146 to 256Kbps and drop the exceeding packets. Mark conforming packets with a CoS value of 3
- The configuration should not affect any future VLANs trunked to R4.

Copyright © 2009 Internetwork Expert

10.70 Catalyst QoS Aggregate Policers

- Limit the aggregate rate of ICMP and IPX traffic coming from R1 on SW1 to 256Kbps, and drop exceeding packets.
- Trunk a new VLAN 47 to the port of R4 connected to SW4.
- Limit the aggregate rate of IP traffic to 128Kbps in both VLAN 146 and the new VLAN 47. Mark conforming packets with DSCP AF11 and remark packets to CS1 if they exceed.

Note

Reset all devices to the *QoS* initial configurations before proceeding.

10.71 Catalyst 3560 Ingress Queueing

- Enable MLS QoS on SW1 and trust IP precedence on the port connected to R1.
- Configure ingress queue 1 as priority queue with 15% of ring bandwidth.
- Map all packets marked with CoS values of 5 to the priority queue.
- Ensure all other packets map to the non-priority queue.
- The bandwidth remaining after the priority queue should be shared between ingress queues in proportions 1:3 for priority and non-priority queues respectively.

10.72 Catalyst 3560 Ingress Queue Tuning

- Configure drop thresholds on SW1 for the non-priority ingress queue to 80% and 100%.
- Ensure that routing and network control traffic mapped to queue 2 is less likely to be dropped than regular traffic.
- Ensure that the queue 2 buffer space is 3 times larger than the priority queue’s size.

10.73 Catalyst 3550 Egress Queueing

- Network traffic consists of the following traffic classes:
    - VoIP bearer packets marked with DSCP EF
    - Interactive video traffic marked with DSCP AF41
    - VoIP signaling traffic marked with DSCP CS3
    - Interactive Citrix traffic marked with DSCP AF21
    - Best-effort traffic marked with DSCP 0
- Implement the following traffic policy on SW4's connection to R4:
    - Provide priority treatment for VoIP bearer packets
    - Provide a separate queue for video traffic and guarantee it 30% of the remaining bandwidth
    - Provide a separate queue for interactive traffic and VoIP signaling with 20% of the bandwidth
    - The remaining bandwidth should be used by best-effort traffic
- Ensure no other traffic but voice can use the priority queue

10.74 Catalyst 3550 Regular Queues Tuning

- Configure SW4 to set the global min-reserve levels 7 and 8 to 170 (max) and 10 (min) buffers respectively.
- Under the interface connected to R4 map global level 7 to queue 1 and global level 8 to queue 4.

10.75 Catalyst 3550 Gigabit Interface Queues Tuning

- Configure the first GigbitEthernet interface on SW1 as follows:
    - Set thresholds to 50 and 100 for all four queues
    - Map DSCP values 46, 48, and 56 to threshold 2
    - Set WRR queue weights to 20, 20, 20 and 40 for queues 1 through 4
- Enable WRED on the second GigabitEthernet interface of SW1 and configure as following:
    - Set WRED minimum thresholds to 50 and 80 for all queues
    - Map DSCP values 46, 48 and 56 to WRED threshold 2
    - Set WRR queue weights to 20, 20, 20 and 40 for queues 1 through 4

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.76 Catalyst 3550 Egress Policing

- Enable the expedite queue on SW4's connection to R4.
- Ensure only CoS 5 maps to this queue.
- Limit the rate of the expedite queue to 128Kbps with the burst size of 8000 bytes.
- Limit the rate of traffic marked with CS0 to 384Kbps using a burst size of 32000 bytes for metering.

10.77 Catalyst 3560 SRR Shared Mode

- Configure SW1 to set the speed of the port connected to R1 to 10 Mbps, and limit the bandwidth to 20%.
- Preserve the default DSCP to Queue ID mapping.
- Set all SRR shaped weights to zero.
- Set the SRR shared weights to 1, 10, 20, and 30 for queues 1 through 4.
- Enable expedite queue on the interface.

10.78 Catalyst 3560 SRR Shaped Mode

- Change the shaped weight for queue 4 to 1Mbps.
- Set the shared weights to 1, 10, 20, and 1 for queues 1 through 4 respectively.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.79 Catalyst 3560 Egress Queues Tuning

- Configure SW1 to modify buffer sharing for queues on the interface connected to R1 only.
- Ensure that queue 2 has 70% of interface buffer pools and remaining queues share buffers in equal proportions.
- Reserve only 10% of allocated buffers for queue 2; the remaining queues should reserve all allocated buffers.
- Ensure that traffic marked with AF11, AF21, AF41, and CS1 maps to the second drop threshold in their respective queues.
- Set the first drop threshold to 200% and the second drop threshold to 150% for all queues.
- Ensure that the switch drops voice packets marked with DSCP EF only if the respective queue is exhausted.
- Set the maximum threshold to 250% for all queues

10.80 Catalyst QoS DSCP Mutation

- Configure SW4 so that R1 sees packets marked at R4 with DSCP values CS0, AF31, and CS5 as DSCP values CS1, CS3, and EF.

10.81 Advanced HTTP Classification with NBAR

- Configure R6’s VLAN146 interface so that HTTP transfers of files with extensions “.bin”, “.text” and “.taxt” are limited to 256Kbps.
- Use only one line to match the URL strings.
- Use the method most friendly for TCP connections.
- Do not apply this limit to the transfers of any other file types.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Quality of Service Solutions

10.1 Hold-Queue and Tx-Ring

- Configure R1's connection to VLAN 146 to have an input software queue length of 10 packets, and an output software queue length of 30 packets.
- Set the output hardware queue size to 15 packets.

Configuration

```
R1:
interface FastEthernet0/0
 tx-ring-limit 15
 hold-queue 10 in
 hold-queue 30 out
```

Verification

```
Rack1R1#show interfaces fastEthernet 0/0 | inclue queue
  Input queue: 0/10/0/0 (size/max/drops/flushes); Total output drops: 0
  Output queue: 0/30 (size/max)

Rack1R1#show controllers fastEthernet 0/0 | include tx
 rx ring entries=64, tx ring entries=128
 txring=0x7DC5800, txr shadow=0x84C0A6D4, tx_head=12, tx_tail=12, tx_count=0
 tx_one_col_err=0, tx_more_col_err=0, tx_no_enp=0, tx_deferred_err=0
 tx_underrun_err=0, tx_late_collision_err=0, tx_loss_carrier_err=6
 tx_exc_collision_err=0, tx_buff_err=0, fatal_tx_err=0
 tx_limited=0(15)
```

Note

Network interfaces on a router work asynchronously. This means the interface driver responds to hardware interrupts, notifies the switching process, and queues incoming packets if the central processor is not yet ready to start working on them. The same applies to outgoing packets. The IOS prepares the packet, puts it into a buffer, and notifies the interface driver of the new packet.

On the input direction if the processor is not too busy and the packet rate is not too high, the system will never actually use the input queue. Similarly, if the outgoing interface is not congested, the interface driver will work fast enough to stop the outgoing queue from growing.

To fully understand an end-to-end QoS model it's important to understand the key differences and similarities between the input and output queues. Both queue types consume memory chunks from buffers pools – interface buffer pools and public shared buffer pools. The buffer management scheme can be quite complicated, especially with particle buffer pools, but the idea remains the same. Queues consume I/O memory available via system buffers, which is a configurable parameter.

For input queueing there is just one queue per interface, which is always First-In-First-Out (FIFO), and has a size of 75 packets by default. The Interface driver assigns incoming packets directly into the incoming FIFO queue if the central processor has no time to switch them quickly (e.g. if sudden packet burst occurs). For more information about input queue management see the section dedicated to SPD in this document.

For output queueing there are two output queues by default. This value can be increased for interfaces that support Virtual Circuits (VCs), such as Frame Relay or ATM, when per-VC queuing is enabled. The first output queue is the "software queue", and defines the per-interface logical queuing strategy. This software queue can be a simple FIFO (hold-queue), or where more advanced "fancy queueing" methods are defined, such as Weighted Fair Queueing (WFQ) or Class-Based Weighted Fair Queueing (CBWFQ). Note that the FIFO hold-queue is the default and simplest queuing strategy on high-speed interfaces.

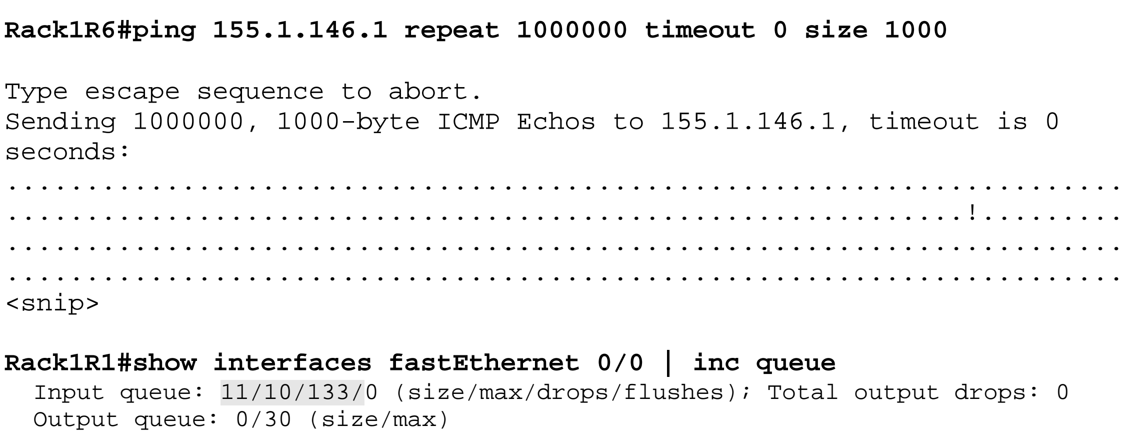
The second output queue sits right after the software queue, and is the hardware queue or "transmit ring" (TX-ring). The interface driver actually works with this memory space directly when looking for packets to send. This queue is typically smaller than the software queue, and is always FIFO.

Note that the software output queue only starts to fill up when the tx-ring is full. This is due to the fact that traffic does not need to be software queued unless the physical hardware queue is busy sending a packet. Therefore the tx-ring length defines how fast the system switches to a logical queuing strategy for interface scheduling. Triggering complicated software queues such as CBWFQ too often may degrade the overall router performance, such as when CBWFQ needs to make a full sorting run across all queues. This is why tuning the tx-ring and output queue size can be a complicated practice, especially when working with highly delay and jitter sensitive traffic such as voice.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Note that for the software output queue is most commonly referred to as just the “output queue”, while the hardware output queue is referred to as just the “transmit ring” (tx-ring). For the sake of differentiating the two these naming conventions are adhered to throughout the scope of this document.

To test the above configuration simulate a packet storm towards R1 and see how the input queue starts to fill up.



```
Rack1R6#ping 155.1.146.1 repeat 1000000 timeout 0 size 1000

Type escape sequence to abort.
Sending 1000000, 1000-byte ICMP Echos to 155.1.146.1, timeout is 0
seconds:
......................................................................
............................................................!.........
......................................................................
......................................................................
<snip>

Rack1R1#show interfaces fastEthernet 0/0 | inc queue
  Input queue: 11/10/133/0 (size/max/drops/flushes); Total output drops: 0
  Output queue: 0/30 (size/max)
```

10.2 Weighted Fair Queuing (WFQ)

- Configure the point-to-point Serial link between R4 and R5 with an interface clock rate and interface bandwidth of 128Kbps.
- Set the output hold-queue size to 256.
- Configure WFQ on the links with a length of 16 for the congestive discard threshold, 128 for the number of conversations, and 8 for RSVP reservable queues.
- Set the tx-ring size to the minimal value to engage the WFQ as fast as possible.
- Normalize the packet flows for the link by ajdusting the MTU so that each IP packet takes no more than 10ms to be sent.

Configuration

```
R4:
interface Serial0/1
 clock rate 128000
 bandwidth 128
 tx-ring-limit 1
 fair-queue 16 128 8
 hold-queue 256 out
 ip mtu 156

R5:
interface Serial0/1
clock rate 128000
 bandwidth 128
 tx-ring-limit 1
 fair-queue 16 128 8
 hold-queue 256 out
 ip mtu 156
```

Verification

Note

WFQ uses an intelligent congestion management solution that provides “fair” sharing of the interface bandwidth between multiple traffic *flows*. A traffic “flow” (or *conversation*) is a unidirectional sequence of packets, defined based on the protocol type, the source/destination IP addresses, the source/destination ports numbers (when available), and partially on the IPv4 ToS byte value. For example, an HTTP file transfer between two hosts represents one packet flow, while ICMP packets sent from one host to another represents a second.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

The term "fair" on WFQ refers to the *max-min* fairness. The WFQ calculation procedure is as follows.

First, divide the interface bandwidth equally between all flows. For example if there are 10 flows, and 100Kbps of bandwidth, each flow gets 10Kbps. If a flow demand is less than the "equal" share, e.g. a flow only needs 5Kbps instead of 10Kbps, allocate the unneeded bandwidth equally among the remaining flows. If there are flows that demand more than the equal share, e.g. the equal share is 10Kbps, but two flows demand 25Kbps and 20Kbps respectively, they will each get the equal, maximum, possible shares (e.g 19Kbps and 19Kbps) but only if there are flows which demand less than the equal share.

In this context *max-min* means that all flows first get the bare minimum, but the procedure tries to maximize each flow's share if possible. The concept of basic WFQ is very important to understand as many other congestion management techniques utilize it, such as Round Robin scheduling. To reiterate, the key fact in WFQ is that the *max-min* scheduling allows the sharing of a flow's unclaimed bandwidth between other flows.

An individual flow in the queue may be more or less demanding than other flows. The "demanding" flow generates the higher bit-rate, either due to larger packet sizes or a higher packet per second rate. For example compare a bulk FTP file transfer against telnet sessions or VoIP RTP streams. Within the context of flow-based sharing, it is also important to understand that a single host may generate *multiple* flows, such as with with P2P file-sharing applications or download "accelerators", thus this particular host can obtain an "unfair" share of bandwidth compared to other hosts. This, unfortunately, is a limitation of a classification scheme that based on simple flows, such as WFQ, which has no notion of "flow mask".

To enhance its scheduling logic, WFQ may assign a *weight* value to a flow. The weight affects the minimum guaranteed share of bandwidth available to a flow. If there are *N* flows with weights w1, w2 ... w*N*, then flow *K* will get the minimum share of bandwidth where s*K*=(w1+w2+… w*K*+… w*N*)/w*K* – inversely proportional to its weight. The shares are relative, in the sense that the scheduler divides the available bandwidth in proportions: s1:s2:…s*N*.

IOS implementation of WFQ assigns weights automatically based on the IP Precedence (IPP) value in the packet's IP header. The formula is Weight=32384/(IPP+1), where IPP is the IP Precedence of the flow.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

To understand the effect of the weight settings, imagine four flows with different IP Precedence values of zero, one, one and three.

Step 1:
Using the above formula for weight, we obtain:

Weight(1) = 32384/(0+1) = 32384
Weight(2) = 32384/(1+1) = 16192
Weight(3) = 32384/(1+1) = 16192
Weight(4) = 32384/(3+1) = 8096

Step 2:

Compute the sum of all weights:
Sum(Weight(i),1…4) = 32384+16192*2+8096 = 72864

Step 3:
Compute the shares:

Share(1) = 72864/Weight(1) = 72864/32384 = 2.25
Share(2) = 72864/Weight(2) = 72864/16192 = 4.5
Share(3) = 72864/Weight(3) = 72864/16192 = 4.5
Share(4) = 72864/Weight(4) = 72864/8096 = 9

The proportion is 2.25:4.5:4.5:9 = 1:2:2:4 – this is based on the “shifted” ip precedences: (IPP(i) + 1)

Note that the numerator “32384” is arbitrary to those computations, meaning that you can use any value. What is important however is the proportion of “shifted” IP precedences.

For bandwidth sharing to be correct, all flows must be adaptive. Adaptive means that a flow must respond to packet drops by slowing its sending rate. Non-adaptive, aggressive flows may defeat the bandwidth sharing logic of WFQ by claiming all available *buffer space* and starving other flows. This is directly linked to the fact that all flows use shared buffer space when implementing WFQ (more on this later).

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Specifically WFQ implements its fair sharing logic as follows. First, the scheduler creates a group of flow queues for the interface, e.g. 256 “conversations”, based on a manual setting or an automatic calculation derived from the interface bandwidth. When a new packet arrives for output scheduling, the scheduler applies a special *hash* function to the packet’s source/destination IP/Port values to yield the flow queue number. This is why the number of queues is always a power of 2, because the hash the output value is somewhere between 0 and 2^N. This procedure also means that multiple flows may share the same queue, called “hash collision”, when the number of flows is large.

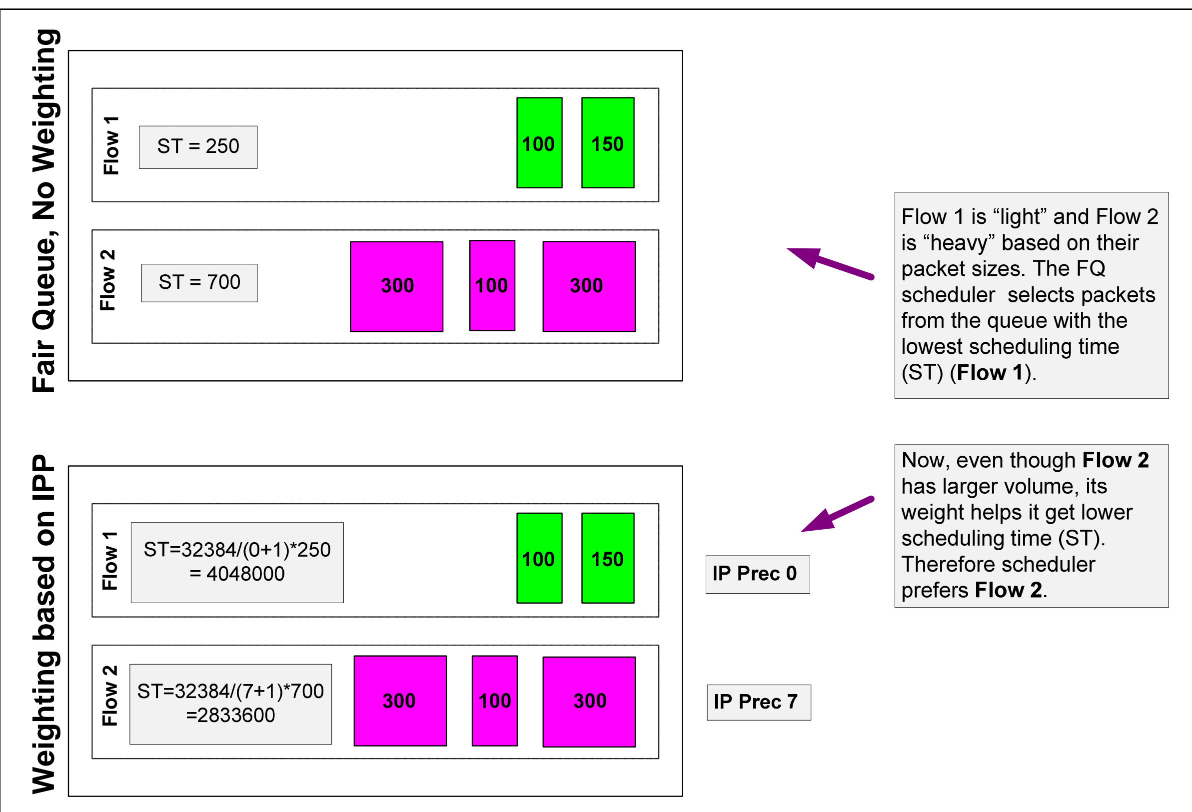
Each flow queue has a special *virtual scheduling time* assigned – the amount of “time” that would take to serialize the packets in the queue across the output interface. This “virtual time” is actually the total size of all packets stored in the flow queue scaled by the flow computational weight.

Now imagine a new packet of size *packet_size* and IP precedence *packet_ip_precedence* arrives to its respective flow queue (hash queue):

weight = 32384/(packet_ip_precedence +1)
*virtual_scheduling_time = queue_tail_time + packet_size*weight*
queue_tail_time = virtual_scheduling_time

The “*queue_tail_time”* variable stores the previous virtual scheduling time for the flow. Note that the weight is inversely proportional to IP precedence, and thus more important packets have smaller virtual scheduling time value (WFQ thinks that it can serialize them “faster”). It is more appropriate to call this computational weight the “scaling factor” to avoid confusion with the classic meaning of weight.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com



Every time WFQ needs to take a packet out of a queues, it sorts all queues based on their virtual scheduling time (sorting is time-consuming). The scheduler selects the flow queue with *minimum* virtual scheduling time, sends a packet out and then decreases the respective *queue_tail_time* variable by packet's size. In essence, this procedure tries to pull the short (smaller packets) flows traffic first.

However, the following question remains open: if a new flow queue has just started, how should we initialize its *queue_tail_time* variable? Should we do that based on just the first packets size? That would result in every new flow having better chances of being scheduled then the older ones, with larger *queue_tail_times*! This is why WFQ stores a special variable called *round_number* shared by all queues. The *round_number* is the *virtual_scheduling_time* of the last packet sent down the physical line. When a flow queue is first initialized, its *queue_tail_time* is set equal to the latest *round_number,* thus making it “comparable” with the existing flow queues, and not letting it monopolize the physical line. The idea is to penalize the new flow by the amount of time other flows have already waited so far.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

The next important point is the congestive discard threshold (CDT), which is a unique congestion avoidance scheme available with WFQ. First you configure the total buffer space allocated to WFQ using the **hold-queue *<N>* out** command. The WFQ shares this buffer space between all flow queues. Any queue may grow up to the maximum free space, but as soon as its size reaches the CDT, WFQ drops a packet from a flow queue with the *maximum* virtual scheduling time (this may be some other queue, not the one that crossed the packet threshold). This way, WFQ penalizes the most aggressive flow and triggers a mechanism similar to random early detection's (RED) prevention of tail-drop. The fair-queue settings command is **fair-queue *<CDT> <N Flow Queues> <N Reservable Queues>.***

The number of reservable conversations (queues) is the number of flow-queues available to RSVP reservations (if any). Those flows have a very small weight value, and thus are preferred above any other flows. In addition to reserved flow queues, there are special "Link Queues". The number of queues is fixed to 8, and they are numbered right after the maximum dynamic queue (e.g. if there are 32 dynamic queues, "Link Queues" start at number 33). WFQ uses those queues to service routing protocol traffic and Layer 2 keepalives – everything that is critical to router operations and management. Each queue has a weight 1024, which is lower than any dynamic queue can get, so control plane traffic has priority over regular data traffic.

Finally, note that the interface bandwidth setting does not influence the WFQ algorithm directly. It only prompts the WFQ command to pick up optimal parameters matching the interface bandwidth. The bandwidth, however, *is* used for admission control with RSVP flows.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

For this particular task, verification can be performed as follows.

```
Rack1R4#show queueing interface serial 0/1
Interface Serial0/1 queueing strategy: fair
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 2044
  Queueing strategy: weighted fair
  Output queue: 3/256/16/2044 (size/max total/threshold/drops)
     Conversations  2/3/32 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 96 kilobits/sec

Rack1R4#show queueing fair
Current fair queue configuration:

  Interface           Discard    Dynamic  Reserved  Link    Priority
                      threshold  queues   queues    queues  queues
  Serial0/0           64         256      0         8       1
  Serial0/1           64         32       1         8       1
```

Changing the **ip mtu** for the interface in this task causes fragmentation at layer 3, and normalizes the packet sizes for further testing. With an additional 4 bytes of overhead for the layer 2 HDLC encapsulation, every frame sent out the link has a maximum size of 160 bytes (156 of IP plus 4 bytes of HDLC) . With a physical clocking rate of 128000 bits per second, this means a 160 byte packet will take a maximum time of 10ms to serialize. Adjusting serialization delay through fragmentation is covered in more detail later in this document.

To verify how WFQ shares the interface bandwidth the network is modified as follows. The HTTP service is enabled on R1 and R6 with the root directory to “flash:”. R5’s Frame Relay interface is shutdown to ensure traffic from VLAN 146 takes the path across the Serial link between R4 and R5. R6 is configured to mark traffic leaving the VLAN146 interface with an IP precedence of 1, and R1 is configured to mark traffic leaving VLAN146 with an IP precedence of 3. A policy-map is configured on R5 to match incoming IP precedence 1 and IP precedence 3 packets, and it is applied inbound to interface Serial0/1 to meter incoming traffic. Finally, the IOS images stored in flash memory on R1 and R6 are downloaded by SW2 and SW4’s to “null:” using HTTP. The resulting configuration is as follows.

```
R1:
ip http server
ip http path flash:
!
policy-map MARK
 class class-default
  set ip precedence 3
!
interface FastEthernet 0/0
 service-policy output MARK

R5:
class-map match-all IP_PREC3
  match ip precedence 3
 class-map match-all IP_PREC1
  match ip precedence 1
!
 policy-map METER
  class IP_PREC1
  class IP_PREC3
!
interface Serial 0/1
 service-policy METER input
 load-interval 30
 clock rate 128000
!
interface Serial 0/0
 shutdown

R6:
ip http server
ip http path flash:
!
policy-map MARK
 class class-default
  set ip precedence 1
!
interface FastEthernet 0/0.146
 service-policy output MARK
```

Rack1SW2#copy http://admin:cisco@155.1.146.6/c2600-adventerprisek9-mz.124-10.bin null:

Rack1SW4#copy http://admin:cisco@155.1.146.1/c2600-adventerprisek9-mz.124-10.bin null:

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

The configuration effectively generates two TCP flows across R4's connection to R5. Since TCP is adaptive, the flows should eventually balance and start sharing bandwidth using their weights: 1+1 and 3+1, which is 2:4. This means file transfers from R1 to SW4 should have twice as much bandwidth as the file transfer from R6 to SW2.

```
Rack1R5#show policy-map int serial 0/1
 Serial0/1

  Service-policy input: METER

    Class-map: IP_PREC1 (match-all)
      91618 packets, 17299811 bytes
      30 second offered rate 41000 bps
      Match: ip precedence 1

    Class-map: IP_PREC3 (match-all)
      330231 packets, 75523458 bytes
      30 second offered rate 83000 bps
      Match: ip precedence 3

    Class-map: class-default (match-any)
      58353 packets, 8839968 bytes
      30 second offered rate 0 bps, drop rate 0 bps
      Match: any
```

Check the flow queues on R4 to see their WFQ parameters. Note that the average packet length is the same, and the inverse weight proportion is 1/8096:1/16192 = 2:1 (the guaranteed shares of bandwidth for IP Precedence 1 and IP Precedence 0 traffic).

```
Rack1R4#show queueing interface serial 0/1
<snip>
  (depth/weight/total drops/no-buffer drops/interleaves) 7/16192/0/0/0
  Conversation 20, linktype: ip, length: 580
  source: 155.1.146.6, destination: 155.1.58.8, id: 0x6927, ttl: 254,
  TOS: 32 prot: 6, source port 80, destination port 11009

  (depth/weight/total drops/no-buffer drops/interleaves) 6/8096/0/0/0
  Conversation 26, linktype: ip, length: 580
  source: 155.1.146.1, destination: 155.1.108.10, id: 0x7BCF, ttl: 254,
  TOS: 96 prot: 6, source port 80, destination port 11001
```

As mentioned above, non-adaptive flows, such as UDP/ICMP traffic floods, may oversubscribe the shared WFQ buffer space. This is due to their “greedy” behavior - a single flow tries to monopolize the whole buffers space available to all queues, causing excessive packet drops, and dose not respond to congestive discards.

To see how this behavior can manifest itself originate two ICMP packet floods from R6 and R1 towards R5 using a packet size of 100 and a timeout of 0. The timeout value of zero means that the devices do not wait for an echo-reply before sending its next echo. Additionally configure R5 so that it does not respond to the echos so the return traffic doesn’t affect the flow.

```
R5:
access-list 100 deny icmp any host 155.1.45.5
access-list 100 permit ip any any
!
interface Serial 0/1
 ip access-group 100 in
 no ip unreachables

Rack1R1#ping 155.1.45.5 repeat 100000000 timeout 0
Rack1R6#ping 155.1.45.5 repeat 100000000 timeout 0

Rack1R4#show queueing interface serial 0/1
<snip>
  (depth/weight/total drops/no-buffer drops/interleaves) 43/8096/7289/0/0
  Conversation 30, linktype: ip, length: 104
  source: 155.1.146.1, destination: 155.1.45.5, id: 0x05E1, ttl: 254, prot: 1

  (depth/weight/total drops/no-buffer drops/interleaves) 21/16192/25458/0/0
  Conversation 3, linktype: ip, length: 104
  source: 155.1.146.6, destination: 155.1.45.5, id: 0xC38A, ttl: 254, prot: 1

Rack1R5#show policy-map interface serial 0/1
 Serial0/1

  Service-policy input: METER

    Class-map: IP_PREC1 (match-all)
      104422 packets, 22233791 bytes
      30 second offered rate 47000 bps
      Match: ip precedence 1

    Class-map: IP_PREC3 (match-all)
      353680 packets, 85227338 bytes
      30 second offered rate 68000 bps
      Match: ip precedence 3
<snip>
```

Note the huge number of drops due to the queue getting full. Also, note that offered rates on R5 are *not* in a 1:2 proportion, even though the weights are set to share the bandwidth in 1:2 ratio and the packet lengths are the same.

10.3 Legacy RTP Reserved Queue

- Reset R4's Serial link to R5 to the default fair-queue values.
- Configure a single command on R4's interface so that 100% of the link bandwidth is reserved for RTP traffic in the UDP port range 16384 – 32767.

Configuration

```
R4:
interface Serial 0/1
 no hold-queue out
 fair-queue
 max-reserved-bandwidth 100
 ip rtp reserve 16384 16383 128
```

Verification

Note

Real Time Protocol (RTP) is the UDP based protocol used for audio and video transmissions, such as VoIP. **`ip rtp reserve`** was the first feature aimed at making WFQ aware of voice bearer traffic (RTP packets). By default, WFQ already prefers voice packets due to their small size and (usually) high IP Precedence value of 5. However as competing flows with the same weight appear in the WFQ, voice traffic is no longer given preferrential treatment.

The first solution for this problem was to enable automatic mapping of voice bearer traffic (RTP packets with a UDP port range 16384-32767) to a special flow queue. WFQ uses a Resource Reservation Protocol (RSVP) classifier and special RSVP conversation to queue the matching packets. The WFQ scheduler will treat this queue as a special "RSVP" reserved queue with a very low weight. Thus, only RSVP flows may compete with the special flow for the voice bearer packets. More details on RSVP and WFQ interaction are covered later.

The command format is **`ip rtp reserve`** ***<Starting UDP Port> <Port Range> <Bandwidth>***

For example **`ip rtp reserve 16384 16383 128`** means to reserve a special RSVP conversation for UDP traffic flows destined to ports in range 16384 – 32767 (16384+16383) – the standard RTP port range used by Cisco devices. The conversation number for IP RTP Reserve is equal to number_of_dynamic_WFQ_flows + 8 + 1, e.g. the first flow after the "Link Queues".

Copyright © 2009 Internetwork Expert

Configuring IP RTP Reserve reduces the “available interface bandwidth” counter by the amount of bandwidth reserved. This counter is needed for the purpose of admission control with RSVP flows, and user-defined classes in CBWFQ. By default, the amount of “available” bandwidth is 75% of total interface bandwidth. The idea is to “underestimate” the bandwidth, allowing some space for routing updates, Layer 2 keepalives, and any other data such as physical layer bit stuffing. This implies that the *<bandwidth>* value cannot be more than *<max-reserved-bandwidth>*<interface bandwidth>* - that is, the interface “available” bandwidth. In this task the **`max-reserved-bandwidth 100`** command is used to allow 128Kbps to be reserved.

The WFQ weight assigned to the special reserved queue is a fixed value of 128, and does not depend on the *<bandwidth>* setting. This value is relatively low to be able to preempt any data traffic flow with an IP precedence in the range of 0 through 7, since the best weight value for IP Precedence 7 traffic is 32384/(7+1) = 4048.

However, a special internal policer, with the rate equal to the configured bandwidth, applies to the matched traffic, and all exceeding packets will have a WFQ weight of 32384. This means that traffic in the RTP port range that exceeds the configured bandwidth is treated as if it had an IP Precedence of 0, which equates to best-effort forwarding. This is a special measure to make sure that the reserved conversation does not consume all of the interface bandwidth by using its low weight. Note that exceeding packets are not dropped, just handled by the WFQ scheduler as if they have less important weight.

```
Rack1R4#show queueing fair
Current fair queue configuration:

  Interface           Discard    Dynamic  Reserved  Link    Priority
                      threshold  queues   queues    queues  queues
  Serial0/0           64         256      0         8       1
  Serial0/1           64         32       1         8       1
```

Copyright © 2009 Internetwork Expert

To verify IP RTP reserve in action we will configure R6 to source G.729-like packet streams to R5 through the IP SLA feature, and measure jitter and round-trip time (RTT). R5 will be configured to respond to these probes. At the same time R1 will be configured as an HTTP server and allow other routers to download its IOS image, which will cause additional congestion in the output queue of R4. R6 will mark outgoing packets with an IP Precedence value of 1, and R1 will mark outgoing packets with an IP precedence value of 7 (the maximum priority and minimum WFQ weight). Note that the Frame-Relay interface of R5 is shutdown to make sure R1 prefers to reach R5 through R4, and vice versa.

```
R1:
ip http server
ip http path flash:
!
policy-map MARK
 class class-default
  set ip precedence 7
!
interface FastEthernet 0/0
 service-policy output MARK

R5:
interface Serial 0/0
 shutdown
!
rtr responder

R6:
policy-map MARK
 class class-default
  set ip precedence 1
!
interface FastEthernet 0/0.146
 service-policy output MARK
!
ip sla monitor 1
 type jitter dest-ipaddr 155.1.45.5 dest-port 16384 codec g729a
 timeout 1000
 frequency 1
!
ip sla monitor schedule 1 life forever start-time now
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Disable RTP reserve on R4 and start transferring the IOS image file from R1 to SW4.

```
R4:
interface Serial 0/1
 no ip rtp reserve
```

Rack1SW4#copy http://admin:cisco@155.1.146.1/c2600-adventerprisek9-mz.124-10.bin null:

Note the WWW file transfer has a minimum possible weight for a dynamic conversation (4048) based on its high IP precedence value of 7.

```
Rack1R4#show queueing interface serial 0/1
<snip>
  (depth/weight/total drops/no-buffer drops/interleaves) 19/4048/170/0/0
  Conversation 7, linktype: ip, length: 160
  source: 155.1.146.1, destination: 155.1.108.10, id: 0x3E96, ttl: 254, prot: 6

  (depth/weight/total drops/no-buffer drops/interleaves) 1/16192/22/0/0
  Conversation 2, linktype: ip, length: 64
  source: 155.1.146.6, destination: 155.1.45.5, id: 0x000F, ttl: 254,
  TOS: 32 prot: 17, source port 59144, destination port 16384
```

Note the RTT values for IP SLA probes and low mean opinion score (MOS) indicates poor voice quality.

```
Rack1R6#show ip sla monitor statistics 1
Round trip time (RTT)      Index 1
        Latest RTT: 458 ms
Latest operation start time: 07:34:06.306 UTC Mon Aug 4 2008
Latest operation return code: OK
RTT Values
        Number Of RTT: 978
        RTT Min/Avg/Max: 26/468/701 ms
Latency one-way time milliseconds
        Number of one-way Samples: 0
        Source to Destination one way Min/Avg/Max: 0/0/0 ms
        Destination to Source one way Min/Avg/Max: 0/0/0 ms
Jitter time milliseconds
        Number of Jitter Samples: 955
        Source to Destination Jitter Min/Avg/Max: 1/5/24 ms
        Destination to Source Jitter Min/Avg/Max: 1/1/11 ms
Packet Loss Values
        Loss Source to Destination: 22          Loss Destination to Source: 0
        Out Of Sequence: 0  Tail Drop: 0  Packet Late Arrival: 0
Voice Score Values
        Calculated Planning Impairment Factor (ICPIF): 32
MOS score: 3.15
Number of successes: 81
Number of failures: 0
Operation time to live: Forever
```

With the current design the voice packets with IP precedence 1 cannot compete with the larger TCP packets that have an IP precedence of 7. This is due to the large mismatch in their weights. Normally VoIP packets would have an IP precedence of 5, which would help somewhat, but IP precedence 7 would still overtake it as a whole. Note the weight values and the average packet sizes.

Now re-enable the RTP reserve feature and compare the weights for both flows once more.

```
R4:
interface Serial 0/1
 ip rtp reserve 16384 16383 128

Rack1R4#show queueing interface serial 0/1
<snip>
  (depth/weight/total drops/no-buffer drops/interleaves) 1/128/0/0/0
  Conversation 41, linktype: ip, length: 64
  source: 155.1.146.6, destination: 155.1.45.5, id: 0x0046, ttl: 254,
  TOS: 32 prot: 17, source port 58177, destination port 16384

  (depth/weight/total drops/no-buffer drops/interleaves) 19/4048/170/0/0
  Conversation 7, linktype: ip, length: 160
  source: 155.1.146.1, destination: 155.1.108.10, id: 0x4497, ttl: 254, prot: 6
```

The weight value assigned to the VoIP traffic flow is now 128 due to the IP RTP Reserve, which is much better than the weight assigned to data transfer. Also, note the number of the conversation for this reserved flow is 41, which is 32+8+1, meaning that we have 32 dynamic flow queues on the interface.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Now check the IP SLA statistics to confirm the increase in voice quality. Note the improved MOS (Mean Opinion Score) and the decreased RTT value.

```
Rack1R6#show ip sla monitor statistics 1
Round trip time (RTT)      Index 1
        Latest RTT: 18 ms
Latest operation start time: 07:35:33.300 UTC Mon Aug 4 2008
Latest operation return code: OK
RTT Values
        Number Of RTT: 1000
        RTT Min/Avg/Max: 12/18/34 ms
Latency one-way time milliseconds
        Number of one-way Samples: 0
        Source to Destination one way Min/Avg/Max: 0/0/0 ms
        Destination to Source one way Min/Avg/Max: 0/0/0 ms
Jitter time milliseconds
        Number of Jitter Samples: 999
        Source to Destination Jitter Min/Avg/Max: 1/4/9 ms
        Destination to Source Jitter Min/Avg/Max: 1/1/12 ms
Packet Loss Values
        Loss Source to Destination: 0           Loss Destination to Source: 0
        Out Of Sequence: 0  Tail Drop: 0  Packet Late Arrival: 0
Voice Score Values
        Calculated Planning Impairment Factor (ICPIF): 11
MOS score: 4.06
Number of successes: 85
Number of failures: 0
Operation time to live: Forever
```

Copyright © 2009 Internetwork Expert

10.4 Legacy RTP Prioritization

- Reset R4's Serial link to R5 to the default fair-queue values, and remove the RTP reservation.
- Configure legacy RTP prioritization for UDP ports in the range 16383 – 32767 up to 128Kbps.

Configuration

```
R4:
interface Serial 0/1
 bandwidth 128
 no hold-queue out
 fair-queue
 max-reserved-bandwidth 100
 no ip rtp reserve
 ip rtp priority 16384 16383 128
```

Verification

✐ Note

The **`ip rtp priority`** feature was the next natural progression beyond the **`ip rtp reserve`** feature, as a strict priority queue replaced the RSVP conversation used to schedule the VoIP packets. The IP RTP Priority feature differs from the IP RTP Reserve in that the priority queue has a WFQ weight of zero, meaning that the WFQ always services it first. Packets leaving the priority queue are rate-limited using the configurable bandwidth setting, and use a burst size of 1 second (e.g. for the bandwidth 128Kbps the burst is 128Kbps*1 second = 16 Kbyte). Furthermore, the IP RTP Priority feature performs an additional check to ensure that only *even* UDP port numbers are matched, which increases the VoIP traffic classification quality.

The biggest advantage of the feature is that the voice conversation no longer has to compete with *any* flow, because of its prioritized nature. This is the reason that rate limiting is necessary, to prevent the other queues from being starved.

The command syntax is the same as with IP RTP Reserve, and reads **`ip rtp priority`** ***<Starting UDP Port> <Port Range> <Bandwidth>***. Here *<Bandwidth> means* the policer rate. As with RTP reserve, the *<Bandwidth>* cannot exceed *<max-reserved-bandwidth>*<interface bandwidth>*, which is why the **`max-reserved-bandwidth`** is changed to 100% in this particular task. Note that the IP RTP priority drops exceeding packets, while IP RTP Reserve simply changes their scheduling weight.

The flow queue number assigned to the priority queue is set right after the special “Link Queue” numbers (used to queue routing updates, L2 keepalives, CDP etc). If there are 2^N dynamic flow queues, the priority queue number is 2^{N+8}. For example, if there are 32 dynamic queues, the priority queue number is 40.

```
Rack1R4#show queueing fair
Current fair queue configuration:

  Interface           Discard    Dynamic  Reserved  Link    Priority
                      threshold  queues   queues    queues  queues
  Serial0/0           64         256      0         8       1
  Serial0/1           64         32       1         8       1
```

The verifications of this feature are the same as with RTP Reserve. R6 sources G.729-like packet streams to R5 through the IP SLA feature, and measure jitter and round-trip time (RTT). At the same time R1 will be configured as an HTTP server and allow other routers to download its IOS image, which will cause additional congestion in the output queue of R4. R6 will mark outgoing packets with an IP Precedence value of 1, and R1 will mark outgoing packets with an IP precedence value of 7 (the maximum priority and minimum WFQ weight). Note that the Frame-Relay interface of R5 is shutdown to make sure R1 prefers to reach R5 through R4, and vice versa.

```
R1:
ip http server
ip http path flash:
!
policy-map MARK
 class class-default
  set ip precedence 7
!
interface FastEthernet 0/0
 service-policy output MARK

R5:
interface Serial 0/0
 shutdown
!
rtr responder

R6:
policy-map MARK
 class class-default
  set ip precedence 1
!
interface FastEthernet 0/0.146
 service-policy output MARK
!
ip sla monitor 1
 type jitter dest-ipaddr 155.1.45.5 dest-port 16384 codec g729a
 timeout 1000
 frequency 1
!
ip sla monitor schedule 1 life forever start-time now
```

Start transferring the large file (IOS image) from R1 to SW4.

```
Rack1SW4#copy http://admin:cisco@155.1.146.1/c2600-adventerprisek9-
mz.124-10.bin null:
```

Observe the queue of R4's Serial interface.

```
Rack1R4#show queue serial 0/1
<snip>

  (depth/weight/total drops/no-buffer drops/interleaves) 1/0/0/0/0
  Conversation 40, linktype: ip, length: 64
  source: 155.1.146.6, destination: 155.1.45.5, id: 0x01A0, ttl: 254,
  TOS: 32 prot: 17, source port 57143, destination port 16384

  (depth/weight/total drops/no-buffer drops/interleaves) 32/4048/0/0/0
  Conversation 7, linktype: ip, length: 160
  source: 155.1.146.1, destination: 155.1.108.10, id: 0x9280, ttl: 254, prot: 6

Rack1R4#show interfaces serial 0/1
Serial0/1 is up, line protocol is up
  Hardware is PowerQUICC Serial
  Internet address is 155.1.45.4/24
<snip>
  30 second input rate 38000 bits/sec, 84 packets/sec
  30 second output rate 121000 bits/sec, 139 packets/sec
```

Note the special "conversation 40", which has no weight value assigned. This is the IP RTP priority queue. The other flow is the HTTP traffic. Now verify the probe statistics on R6.

```
Rack1R6#show ip sla monitor statistics 1
Round trip time (RTT)      Index 1
        Latest RTT: 18 ms
Latest operation start time: 08:15:49.305 UTC Mon Aug 4 2008
Latest operation return code: OK
RTT Values
        Number Of RTT: 1000
        RTT Min/Avg/Max: 13/18/33 ms
Latency one-way time milliseconds
        Number of one-way Samples: 0
        Source to Destination one way Min/Avg/Max: 0/0/0 ms
        Destination to Source one way Min/Avg/Max: 0/0/0 ms
Jitter time milliseconds
        Number of Jitter Samples: 999
        Source to Destination Jitter Min/Avg/Max: 1/5/10 ms
        Destination to Source Jitter Min/Avg/Max: 1/1/15 ms
Packet Loss Values
        Loss Source to Destination: 0           Loss Destination to Source: 0
        Out Of Sequence: 0  Tail Drop: 0  Packet Late Arrival: 0
Voice Score Values
        Calculated Planning Impairment Factor (ICPIF): 11
MOS score: 4.06
Number of successes: 29
Number of failures: 0
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.5 Legacy Custom Queueing

- Remove R4's WFQ configuration on the Serial link connecting to R5.
- Configure legacy custom queueing on this link so that traffic is round-robin scheduled based on the following requirements:
    - RTP VoIP traffic should be guaranteed 30% of the link bandwidth.
    - File transfers from web servers in VLAN 146 should be guaranteed 60% of the link bandwidth.
    - The remaining 10% should be allocated to ICMP, but this traffic should not exceed 10 packets in the queue at any given time.
    - RIP routing packets should use the system priority queue.
- Clock the link at 128Kbps, and ensure that IP packets larger than 156 bytes are fragmented.
- Assume packet sizes of 60 bytes for RTP, 156 bytes for TCP, and 100 bytes for ICMP.
- Ensure to account for the layer 2 HDLC encapsulation overhead.

Configuration

```
R4:
access-list 100 permit tcp 155.1.146.0 0.0.0.255 eq www any
access-list 101 permit icmp any any
!
queue-list 1 protocol ip 0 udp 520
queue-list 1 protocol ip 1 lt 65
queue-list 1 protocol ip 2 list 100
queue-list 1 protocol ip 3 list 101
queue-list 1 default 3
queue-list 1 queue 3 limit 10
queue-list 1 queue 1 byte-count 320
queue-list 1 queue 2 byte-count 640
queue-list 1 queue 3 byte-count 104
!
interface Serial 0/1
 ip mtu 156

R5:
interface Serial 0/1
 clock rate 128000
 ip mtu 156
```

Verification

Note

Legacy custom queueing is similar to weighted fair queueing in that it tries to share the bandwidth between packet flows in a max-min fashion. This means that each flow gets a guaranteed minimum share that is directly proportional to its weight, but any flow can claim the unused interface bandwidth if other flows are not using their guaranteed rate.

Unlike WFQ there are no dynamic flows, but instead there are 16 static queues that must be manually defined. The custom queue assigns a byte counter to every configured queue, with a 1500-byte default value, and services the queues in round-robin fashion. Every time a queue is serviced the de-queued packet decrements the byte count by the packet size until the counter drops to zero, at which time the next queue is serviced.

Custom Queuing also supports an additional system queue, number 0. The system queue is the priority queue, and is always serviced first, before all other regular queues. By default the system queue is used for layer 2 keepalives, but not for routing update packets (e.g. RIP, OSPF, or EIGRP). Therefore routing updates should be mapped manually to queue 0, unless the interface is running Frame Relay encapsulation, which uses a special broadcast queue to send multicast routing updates. All unclassified packets are assigned to queue 1 by default unless the default queue number is changed.

The limitation of round robin scheduling is that it cannot de-queue less than one packet each time a queue is serviced. Furthermore, since different queues can have different average packet sizes, e.g. voice packets of 60 bytes and TCP packets of 1500 bytes, this can lead to an undesirable bandwidth distribution ratio. For example if a queue’s byte counter is 100 bytes, but a packet of 1500 bytes is in the queue, the packet is still sent since the counter is not zero. In order to make the distribution “fair”, every queue’s byte counter traditionally had to be proportional to the queue’s average packet size.

In IOS 12.1 the behavior of the Custom Queue changed to fix this distribution problem by implementing a so-called Deficit Round Robin. The new deficit based algorithm tracks the amount of excessive bytes consumed by every queue, and takes that into account in the next scheduling rounds. In essence a queue can “borrow” credit in order to send a packet whose size exceeds the byte count left, but the debt is paid back the next time the queue is serviced. Based on this change it is no longer necessary to set the byte counts proportional to average packets sizes. However, as a general rule setting the byte counts higher will increase the scheduling delay, but will make the traffic distribution smoother.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

In this example the traffic *is* classified based on the packet size, and the byte counts are computed based on this. Specifically in this task there are three traffic classes, RTP VoIP packets with a layer 3 size of 60 bytes (G.729 samples with 20 bytes per packet – 50pps), TCP packets with a layer 3 size of 156 bytes (due to the MTU adjustment forcing fragmentation), and ICMP packets with a layer 3 size of 100 bytes. Note that the custom queue byte counter does take Layer 2 overhead into consideration, so an additional 4 bytes is accounted for in the HDLC protocol overhead. The result is that our counters are based on frame sizes of 64 bytes, 160 bytes, and 104 bytes respectively. The desired distribution ratio of 30:60:10 is then calculated as follows.

Since the byte counters must be proportional to the packet size, the following should remain true: C1=a*64, C2=b*160; C3=c*104 where “a”, “b” and “c” are the multipliers we are looking for and C(i) is the byte-counter for queue “i”. From the proportion C1:C2:C3=30:60:10 we get a=30/64=0.47, b=60/160=0.38, and c=10/104=0.096. Now the problem is that those multiples are fractional, and we can’t hold a fractional number of packets in a queue.

To get an optimal whole number, we must first normalize the fractional numbers by dividing by the smallest one. In our case this is c=0.096. Therefore, now a1=a/c=4.89; b1=b/c=3.95; c1=1. It’s now possible to round up to the next integer and use this as the number of packets that should be sent every turn, but what if we had numbers like 3.5 or 4.2? Rounding them up we get integers far larger than the initial values, which results in a different packet distribution than desired. To avoid this we could scale (multiply) all numbers by some integer value (e.g. by 2 or 3) to obtain a value closer to a whole number.

In our case we could simply multiply by 100 to get exact whole numbers, but this would yield excessively high byte counts, and would result in considerable delays between round-robin scheduler runs. In addition to this when calculating the final byte counts we need to make sure the queue depths allow for large counters in case we still want to use them. For this example we simply round up to get multipliers a=5, b=4, and c=1. The resulting byte counters are C1=a*64=320, C2=b*160=640, C3=c*104=104. Now let’s calculate the weights each queue obtains with this configuration.

W1=C1/(C1+C2+C3)=320/(640+320+104)=320/1064=0.3
W2=C2/(C1+C2+C3)=640/1064=0.6
W3=C3/(C1+C2+C3)=104/1064=0.1

This is what we want – the proportions of 30:60:10. Also, the maximum byte counter is 640 bytes, so the maximum possible delay would be based on 4x160 byte packets, resulting in 40ms of maximum delay (not too VoIP friendly).

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

When traffic is mapped to a queue number, the system evaluates all statements sequentially, and the first one matched by the packet determines the queue where it goes. This is similar to how the modular quality of service policy-map matches classes in a top-down fashion. In this example packets with a size less than 65 bytes are matched in queue 1, per the statement **`queue-list 1 protocol ip 1 lt 65`**. This means that ICMP packets smaller than 65 bytes will be classified using the first statement, instead of the third. Also note that the size criteria with legacy QoS commands are based on the full layer 2 packet length, which means the layer 2 header, the layer 3 IP header, plus the packet payload. In this particular case the HDLC header is 4 bytes, and thus voice packets have a full layer 2 size of 64 bytes. In addition when you specify a port number to be matched for CQ classification, such as the UDP port 520 for RIP, it matches both source and destination ports.

To verify this configuration R6 is set to source a stream of G.729 packets to R5, R5 is set to respond, and the jitter and RTT are measured. The G.729 stream consumes slightly more than 24Kbps of bandwidth. At the same time R1 is configured as an HTTP server, and its IOS image is transferred to SW2. Finally note that the Frame Relay interface of R5 is disabled to force these traffic flows to route over the interface configured with the custom queueing parameters.

```
R1:
ip http server
ip http path flash:

R5:
interface Serial 0/0
 shutdown
!
rtr responder

R6:
ip sla monitor 1
 type jitter dest-ipaddr 155.1.45.5 dest-port 16384 codec g729a
 timeout 1000
 frequency 1
!
ip sla monitor schedule 1 life forever start-time now
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Configure R5 to meter the incoming traffic rate with an MQC policy. The three traffic classes below match the packets generated by the SLA probe, the WWW traffic, and the ICMP packets. When applied to the Serial link we can view the traffic utilization on a per-class basis. Also note that the MQC matches the layer 3 packet size (e.g. 60 bytes for G.729) before any layer 2 overhead.

```
R5:
access-list 100 permit tcp 155.1.146.0 0.0.0.255 eq www any
access-list 101 permit icmp any any
!
class-map match-all ICMP
 match access-group 101
!
class-map match-all VOICE
 match packet length min 60 max 60
!
class-map match-all WWW
 match access-group 100
!
policy-map METER
 class VOICE
 class WWW
 class ICMP
!
interface Serial 0/1
 service-policy input METER
 load-interval 30
```

Start transferring R1’s IOS to SW2, and send a barrage of ICMP packets from SW1 to R5.

```
Rack1SW2#copy http://admin:cisco@155.1.146.1/c2600-adventerprisek9-
mz.124-10.bin null:

Rack1SW1#ping 155.1.45.5 size 100 repeat 100000000 timeout 0

Type escape sequence to abort.
Sending 100000, 100-byte ICMP Echos to 155.1.45.5, timeout is 1
seconds:
................
<snip>
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Now verify the custom queue settings and statistics on R4.

```
Rack1R4#show queueing custom
Current custom queue configuration:

List   Queue  Args
1      3      default
1      1      protocol ip          lt 65
1      2      protocol ip          list 100
1      3      protocol ip          list 101
1      0      protocol ip          udp port 520
1      1      byte-count 320
1      2      byte-count 640
1      3      byte-count 104 limit 10
```

Below, the current queue utilization are checked. Note that almost all queues have a non-zero depth, meaning packets are currently queued up, and the queue for ICMP packets has the maximum amount of packet drops. Also note that the voice packets (queue 1) are queued as well, but the drop count is very low.

```
Rack1R4#show interfaces serial 0/1
Serial0/1 is up, line protocol is up
<snip>
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops:
1332299
  Queueing strategy: custom-list 1
  Output queues: (queue #: size/max/drops)
      0: 0/20/0 1: 8/20/55 2: 11/20/24497 3: 10/10/1307747 4: 0/20/0
     5: 0/20/0 6: 0/20/0 7: 0/20/0 8: 0/20/0 9: 0/20/0
     10: 0/20/0 11: 0/20/0 12: 0/20/0 13: 0/20/0 14: 0/20/0
     15: 0/20/0 16: 0/20/0
  5 minute input rate 49000 bits/sec, 95 packets/sec
  5 minute output rate 121000 bits/sec, 143 packets/sec
<snip>
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

> The current queue contents can be viewed with the command **show queue *<interface> <slot/port> <qNumber>***. For example, the second queue's contents are displayed below.

```
Rack1R4#show queue serial 0/1 2
Output queue for Serial0/1 is 5/20

Packet 1, linktype: ip, length: 160, flags: 0x88
  source: 155.1.146.1, destination: 155.1.58.8, id: 0x60A7, ttl: 254,
prot: 6
    data: 0x9577 0x93FD 0xB6FB 0xA7DC 0x7E54 0xBD08 0x74B7
          0xB79C 0x3EF4 0x27E1 0x75B2 0xC3B7 0x37F3 0xDC17

Packet 2, linktype: ip, length: 160, flags: 0x88
  source: 155.1.146.1, destination: 155.1.58.8, id: 0x60A8, ttl: 254,
  TOS: 0 prot: 6, source port 80, destination port 11015
    data: 0x0050 0x2B07 0x21FC 0xF1E0 0xA32B 0x3C56 0x5010
          0x0F62 0x7866 0x0000 0x3DBE 0xBE6D 0xE70D 0x3E27

Packet 3, linktype: ip, length: 160, flags: 0x88
  source: 155.1.146.1, destination: 155.1.58.8, id: 0x60A8, ttl: 254,
prot: 6
    data: 0x39FD 0xBEC1 0xC76A 0xF5F7 0x37FC 0xB731 0x4E09
          0xDEB5 0x761C 0xE6F6 0x7390 0xB3B3 0x791C 0x400A
<snip>
```

> Check the SLA probe statistic on R6, and note the high RTT. This is because the voice packets are not in a priority queue, but are serviced as regular classes.

```
Rack1R6#show ip sla monitor statistics 1
Round trip time (RTT)   Index 1
      Latest RTT: 85 ms
Latest operation start time: 05:28:54.827 UTC Tue Aug 5 2008
Latest operation return code: OK
RTT Values
      Number Of RTT: 1000
      RTT Min/Avg/Max: 12/85/169 ms
Latency one-way time milliseconds
      Number of one-way Samples: 0
      Source to Destination one way Min/Avg/Max: 0/0/0 ms
      Destination to Source one way Min/Avg/Max: 0/0/0 ms
Jitter time milliseconds
      Number of Jitter Samples: 999
      Source to Destination Jitter Min/Avg/Max: 2/26/148 ms
      Destination to Source Jitter Min/Avg/Max: 1/2/19 ms
Packet Loss Values
      Loss Source to Destination: 0       Loss Destination to Source: 0
      Out Of Sequence: 0      Tail Drop: 0      Packet Late Arrival: 0
Voice Score Values
      Calculated Planning Impairment Factor (ICPIF): 11
<snip>
```

Look at R5's MQC policy statistics to see the resulting bandwidth allocation. Note that the voice class's bandwidth is close to 24Kbps, and does not exceed its allocated share of 38Kbps (0.3*128K). Therefore the other classes can claim the remaining bandwidth. This explains why WWW traffic has more bandwidth than its guaranteed share of 76Kbps (0.6*128K).

Note the ICMP packet flood gets 13Kbps of bandwidth, and does not seriously impair the other traffic flows like it did with WFQ configured. This is because each class has its separate FIFO queue, and traffic flows do not share the same buffer space (plus there are no congestive discard procedures). Furthermore, the aggressive ICMP traffic behavior does not seriously affect the other queues since the scheduler drops the exceeding traffic.

In total, all classes draw approximately 120Kbps of L3 bandwidth, which is naturally less than the configured 128Kbps line rate.

```
Rack1R5#show policy-map interface serial 0/1
 Serial0/1

  Service-policy input: METER

    Class-map: VOICE (match-all)
      878252 packets, 56208128 bytes
      1 minute offered rate 23000 bps
      Match: packet length min 60 max 60

    Class-map: WWW (match-all)
      211383 packets, 72688125 bytes
      1 minute offered rate 84000 bps
      Match: access-group 100

    Class-map: ICMP (match-all)
      68628 packets, 7137268 bytes
      1 minute offered rate 13000 bps
      Match: access-group 101

    Class-map: class-default (match-any)
      2573 packets, 630376 bytes
      1 minute offered rate 0 bps, drop rate 0 bps
      Match: any
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.6 Legacy Custom Queueing with Prioritization

- Modify R4's custom queueing configuration so that the RTP VoIP traffic is prioritized after the RIP packets.

Configuration

```
R4:
queue-list 1 lowest-custom 2
```

Verification

In this example queue 1 is next in line for prioritization after the system queue 0. The **lowest-custom** option tells the round robin scheduler where to start, meaning that all queues up to this number will use priority scheduling. Queue 0 is the most important, queue 1 is the next in importance, etc.

Note that even though this configuration allows for priority queueing and custom queueing at the same time, it is not recommended, because there is no way to limit the priority queues. Just like with the legacy priority queue, this means that other traffic classes can be effectively starved of bandwidth if there are consistently packets in the priority queues. In a practical design, voice traffic should use either the IP RTP Priority feature, or the MQC low-latency queue (LLQ), due to their built-in policer functions.

The above configuration can be verified as follows.

```
Rack1R4#clear counters Serial 0/1
Clear "show interface" counters on this interface [confirm]
Rack1R4#conf t
Rack1R4(config)#queue-list 1 lowest-custom 2

Rack1R4#show queueing custom
Current custom queue configuration:

List   Queue  Args
1      2      lowest custom queue
1      3      default
1      1      protocol ip          lt 65
1      2      protocol ip          list 100
1      3      protocol ip          list 101
1      0      protocol ip          udp port 520
1      1      byte-count 320
1      2      byte-count 640
1      3      byte-count 104 limit 10
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
Rack1R4#show interfaces serial 0/1
Serial0/1 is up, line protocol is up
<snip>
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops:
1053009
  Queueing strategy: custom-list 1
  Output queues: (queue #: size/max/drops)
     0: 0/20/0 1: 0/20/55 2: 7/20/20301 3: 7/10/1032653 4: 0/20/0
     5: 0/20/0 6: 0/20/0 7: 0/20/0 8: 0/20/0 9: 0/20/0
     10: 0/20/0 11: 0/20/0 12: 0/20/0 13: 0/20/0 14: 0/20/0
     15: 0/20/0 16: 0/20/0
  5 minute input rate 49000 bits/sec, 95 packets/sec
  5 minute output rate 121000 bits/sec, 144 packets/sec
<snip>
```

Now verify the SLA statistics for the RTP packet probe on R6. Note that RTT has dropped significantly since the packets now go through the priority queue.

```
Rack1R6#show ip sla monitor statistics 1
Round trip time (RTT)   Index 1
      Latest RTT: 19 ms
Latest operation start time: 05:44:00.485 UTC Tue Aug 5 2008
Latest operation return code: OK
RTT Values
      Number Of RTT: 1000
      RTT Min/Avg/Max: 12/19/35 ms
Latency one-way time milliseconds
      Number of one-way Samples: 0
      Source to Destination one way Min/Avg/Max: 0/0/0 ms
      Destination to Source one way Min/Avg/Max: 0/0/0 ms
Jitter time milliseconds
      Number of Jitter Samples: 999
      Source to Destination Jitter Min/Avg/Max: 1/4/10 ms
      Destination to Source Jitter Min/Avg/Max: 1/2/18 ms
Packet Loss Values
      Loss Source to Destination: 0       Loss Destination to Source: 0
      Out Of Sequence: 0      Tail Drop: 0      Packet Late Arrival: 0
Voice Score Values
      Calculated Planning Impairment Factor (ICPIF): 11
MOS score: 4.06
Number of successes: 91
Number of failures: 0
Operation time to live: Forever
```

Copyright © 2009 Internetwork Expert

Check bandwidth usage on R5. Note that bandwidth allocation did not change significantly since the voice traffic rate remained the same.

```
Rack1R5#show policy-map interface serial 0/1
 Serial0/1

  Service-policy input: METER

    Class-map: VOICE (match-all)
      916126 packets, 58632064 bytes
      1 minute offered rate 25000 bps
      Match: packet length min 60 max 60

    Class-map: WWW (match-all)
      277613 packets, 81443501 bytes
      1 minute offered rate 82000 bps
      Match: access-group 100

    Class-map: ICMP (match-all)
      82087 packets, 8537004 bytes
      1 minute offered rate 13000 bps
      Match: access-group 101

    Class-map: class-default (match-any)
      2881 packets, 675328 bytes
      1 minute offered rate 0 bps, drop rate 0 bps
      Match: any
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.7 Legacy Priority Queueing

- Remove R4's custom queueing configuration and replace it with legacy priority queueing per the following requirements:
  - All RIP packets going out the Serial link to R5 should be sent first.
  - If there are no RIP packets, RTP VoIP should be sent next.
  - If there are no RIP or VoIP packets, web traffic should be sent next.
  - As a last resort ICMP traffic should be sent.
- Set the queue-sizes for the high, medium, normal, and low queues to 5, 40, 60, and 80 packets respectively.

Configuration

```
R4:
access-list 102 permit udp any any range 16384 32767
access-list 103 permit icmp any any
!
priority-list 1 protocol ip high udp rip
priority-list 1 protocol http normal
priority-list 1 protocol ip medium list 102
priority-list 1 protocol ip low list 103
priority-list 1 queue-limit 5 40 60 80
!
interface Serial 0/1
 no custom-queue-list 1
 priority-group 1
```

Verification

✐ Note

Unlike WFQ or legacy Custom Queuing (CQ), legacy Priority Queueing (PQ) does not provide a max-min sharing of interface bandwidth. Instead, it simply sends the most important traffic flow completely before it moves to the next important flow, and so on, from greatest to least important class. The PQ supports four static queues, unlike 16 in CQ, and a large number of dynamic queues in WFQ, and classification is based on static configuration, similar to Custom Queueing. Every queue has a limit which is based on number of packets, not bandwidth, with the lower priority queues having more packet space than the higher priority queues. The default values can be found by applying the **`priority-list`** command to an interface and issuing the **`show interface`** command.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

The biggest design flaw with Priority Queueing is that the lower queues may starve if the higher queues are constantly busy. This is why legacy Priority Queueing is not very helpful for real VoIP deployments, because the VoIP traffic streams are continuous, e.g. they keep their queue busy almost consistently, and there is no way the PQ can restrict them. However, PQ is suitable for use in networks where sporadic traffic bursts of different priorities exist. For example, you may want to give OLTP database traffic the most priority and allow it to preempt other traffic types, but the traffic flow is only periodic.

Just like the legacy Custom Queue the port numbers specified in the **priority-list** command match the source or destination port, and the packet length match checks against the full layer 2 packet size.

To verify this configuration R6 is set to source a stream of G.729 packets to R5, R5 is set to respond, and the jitter and RTT are measured. The G.729 stream consumes slightly more than 24Kbps of layer 3 bandwidth. At the same time R1 is configured as an HTTP server, and its IOS image is transferred to SW2. Finally note that the Frame Relay interface of R5 is disabled to force these traffic flows to route over the interface configured with the priority qeueing parameters.

```
R1:
ip http server
ip http path flash:

R5:
interface Serial 0/0
 shutdown
!
rtr responder

R6:
ip sla monitor 1
 type jitter dest-ipaddr 155.1.45.5 dest-port 16384 codec g729a
 timeout 1000
 frequency 1
!
ip sla monitor schedule 1 life forever start-time now
```

Start transferring R1's IOS image to SW2, and ping R5's serial interface from SW1. Note that the ping packets cannot make it through R4 while the file is transferring from R1 to SW2. This is because the higher queues completely block the low priority queue.

```
Rack1SW2#copy http://admin:cisco@155.1.146.1/c2600-adventerprisek9-mz.124-10.bin null:
Loading http://admin:cisco@155.1.146.1/c2600-adventerprisek9-mz.124-10.bin !!!!!!

Rack1SW1#ping 155.1.45.5 repeat 100000 size 100

Type escape sequence to abort.
Sending 100000, 100-byte ICMP Echos to 155.1.45.5, timeout is 2 seconds:
....................
```

Verify the priority queue statistics on the serial interface of R4. Note the high queue depth for the low queue, which is used for ICMP packets. Also note that the system uses the "high" queue for RIP updates and layer 2 keepalives; this is why we see packet matches here.

```
Rack1R4#show interfaces serial 0/1
Serial0/1 is up, line protocol is up
  <snip>
  Queueing strategy: priority-list 1
  Output queue (queue priority: size/max/drops):
     high: 0/5/0, medium: 0/40/0, normal: 6/60/0, low: 43/80/0
  5 minute input rate 40000 bits/sec, 92 packets/sec
  5 minute output rate 123000 bits/sec, 71 packets/sec
     96338 packets input, 5550390 bytes, 0 no buffer
<snip>

Rack1R4#show queueing interface serial 0/1
Interface Serial0/1 queueing strategy: priority

Output queue utilization (queue/count)
      high/149 medium/53360 normal/23090 low/5105
```

To view the current queue contents issue the **show queue *<interface>* *<slot/port>* *<qNumber>***. Note the current queue sizes and the maximum queue depth fields.

```
Rack1R4#show queue serial 0/1 2
Output queue for Serial0/1 is 32/60

Packet 1, linktype: ip, length: 160, flags: 0x88
  source: 155.1.146.1, destination: 155.1.58.8, id: 0x4727, ttl: 254,
prot: 6
    data: 0x75B6 0x856D 0x986F 0x0278 0x4B11 0xE94C 0x69B0
          0x8E0B 0xE9D9 0xA556 0x9BBA 0xF9F3 0xC018 0xE443

Packet 2, linktype: ip, length: 36, flags: 0x88
  source: 155.1.146.1, destination: 155.1.58.8, id: 0x4727, ttl: 254,
prot: 6
    data: 0xE64E 0x415B 0xD71C 0x1BFA 0xEDC2 0xAFFB 0x02AA
          0x1C36 0xABCD 0xABCD 0xABCD 0xABCD 0xABCD 0xABCD
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
Packet 3, linktype: ip, length: 160, flags: 0x88
  source: 155.1.146.1, destination: 155.1.58.8, id: 0x4728, ttl: 254,
  TOS: 0 prot: 6, source port 80, destination port 11010
    data: 0x0050 0x2B02 0xEAE5 0x5240 0x2235 0x7AA8 0x5010
          0x0F62 0x2D5F 0x0000 0x99E6 0x2C18 0x1FF6 0x3127

Packet 4, linktype: ip, length: 160, flags: 0x88
  source: 155.1.146.1, destination: 155.1.58.8, id: 0x4728, ttl: 254,
prot: 6
    data: 0x16E0 0xEB4B 0x91C7 0xA734 0xF13C 0x350A 0xE020
          0x97E3 0x00EF 0x589F 0x8CF7 0x8D03 0xFFB6 0xF0D8
0xA97C 0x0000 0x0000 0x03B4 0x0000 0xABCD 0xABCD
<snip>

Rack1R4#show queue serial 0/1 3
Output queue for Serial0/1 is 80/80

Packet 1, linktype: ip, length: 104, flags: 0x88
  source: 155.1.67.7, destination: 155.1.45.5, id: 0xD54F, ttl: 253,
prot: 1
    data: 0x0800 0x5F60 0x0012 0x0000 0x0000 0x0000 0x02AA
          0x1C2E 0xABCD 0xABCD 0xABCD 0xABCD 0xABCD 0xABCD

Packet 2, linktype: ip, length: 104, flags: 0x88
  source: 155.1.67.7, destination: 155.1.45.5, id: 0xD550, ttl: 253,
prot: 1
    data: 0x0800 0x5F5F 0x0012 0x0001 0x0000 0x0000 0x02AA
          0x1C2E 0xABCD 0xABCD 0xABCD 0xABCD 0xABCD 0xABCD

Packet 3, linktype: ip, length: 104, flags: 0x88
  source: 155.1.67.7, destination: 155.1.45.5, id: 0xD551, ttl: 253,
prot: 1
    data: 0x0800 0x5F5E 0x0012 0x0002 0x0000 0x0000 0x02AA
          0x1C2E 0xABCD 0xABCD 0xABCD 0xABCD 0xABCD 0xABCD

Packet 4, linktype: ip, length: 104, flags: 0x88
  source: 155.1.67.7, destination: 155.1.45.5, id: 0xD552, ttl: 253,
prot: 1
    data: 0x0800 0x5F5D 0x0012 0x0003 0x0000 0x0000 0x02AA
          0x1C2E 0xABCD 0xABCD 0xABCD 0xABCD 0xABCD 0xABCD
<snip>
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Note that the file transfer does not affect voice packets.

```
Rack1R6#show ip sla monitor statistics 1
Round trip time (RTT)   Index 1
      Latest RTT: 18 ms
Latest operation start time: 05:07:10.702 UTC Tue Aug 5 2008
Latest operation return code: OK
RTT Values
      Number Of RTT: 1000
      RTT Min/Avg/Max: 12/18/43 ms
Latency one-way time milliseconds
      Number of one-way Samples: 0
      Source to Destination one way Min/Avg/Max: 0/0/0 ms
      Destination to Source one way Min/Avg/Max: 0/0/0 ms
Jitter time milliseconds
      Number of Jitter Samples: 999
      Source to Destination Jitter Min/Avg/Max: 1/5/26 ms
      Destination to Source Jitter Min/Avg/Max: 1/1/4 ms
Packet Loss Values
      Loss Source to Destination: 0       Loss Destination to Source: 0
      Out Of Sequence: 0      Tail Drop: 0      Packet Late Arrival: 0
Voice Score Values
      Calculated Planning Impairment Factor (ICPIF): 11
MOS score: 4.06
Number of successes: 161
Number of failures: 0
Operation time to live: Forever
```

Now stop the file transfer from R1 to SW2 and try pinging R5 from SW1 again.

```
Rack1SW1#ping 155.1.45.5 repeat 100000 size 100

Type escape sequence to abort.
Sending 100000, 100-byte ICMP Echos to 155.1.45.5, timeout is 2
seconds:
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Note the queue depth for the "low" queue is zero now compared to the larger number before.

```
Rack1R4#show interfaces serial 0/1
Serial0/1 is up, line protocol is up
<snip>
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops:
320
  Queueing strategy: priority-list 1
  Output queue (queue priority: size/max/drops):
     high: 0/5/0, medium: 0/40/0, normal: 0/60/0, low: 0/80/320
<snip>
```

The voice stream quality basically remains the same:

```
Rack1R6#show ip sla monitor statistics 1
Round trip time (RTT)   Index 1
      Latest RTT: 12 ms
Latest operation start time: 05:05:25.421 UTC Tue Aug 5 2008
Latest operation return code: OK
RTT Values
      Number Of RTT: 1000
      RTT Min/Avg/Max: 11/12/31 ms
Latency one-way time milliseconds
      Number of one-way Samples: 0
      Source to Destination one way Min/Avg/Max: 0/0/0 ms
      Destination to Source one way Min/Avg/Max: 0/0/0 ms
Jitter time milliseconds
      Number of Jitter Samples: 999
      Source to Destination Jitter Min/Avg/Max: 1/1/19 ms
      Destination to Source Jitter Min/Avg/Max: 1/1/7 ms
Packet Loss Values
      Loss Source to Destination: 0       Loss Destination to Source: 0
      Out Of Sequence: 0      Tail Drop: 0      Packet Late Arrival: 0
Voice Score Values
      Calculated Planning Impairment Factor (ICPIF): 11
MOS score: 4.06
Number of successes: 156
Number of failures: 0
Operation time to live: Forever
```

10.8 Legacy Random Early Detection

- Configure R4's point-to-point Serial interface connecting to R5 to avoid tail drop behavior by randomly dropping packets before the output queue overflows.
- Set the hold-queue size to 10 packets, and the weighted averaging constant to 4.
- Ensure that routing updates are never randomly dropped.

Configuration

```
R4:
interface Serial0/1
 random-detect
 random-detect exponential-weighting-constant 4
 random-detect precedence 6 11 12
 hold-queue 10 out
```

Verification

Note

Legacy QoS configuration supports WRED (Weighted Random Early Detection) only on main interfaces. Since WRED is essentially a FIFO queue with a modified drop strategy, you can tune its size using the **`hold-queue`** command. WRED parameters include the following:

1) Queue size averaging factor
2) Maximum and Minimum thresholds (per IP Precedence or per DSCP value)
3) Mark (drop) probability

Recall that IP Routing updates are sent from the router with an IP precedence of 6. In this example, the minimum threshold for IP precedence 6 is 11 packets, meaning that random dropping does not occur until this threshold is breached. However, since the tail drop threshold (hold-queue) is set to 10 packets, random dropping never occurs for these packet flows. A detailed explanation of the usage of the WRED feature is as follows.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

A First-In First-Out (FIFO) queueing strategy is the simplest to implement, but it has several flaws critical to QoS implementations. One of them is the notion of *tail drop* used by FIFO queues. The *tail* of the output queue is where packets enter, while a packet at the *head* of the output queue is next in line for the transmit ring. Tail drop simply means that if the output queue is full, the system drops every new packet trying to enter the queue's tail, while packets already in the queue keep waiting to get to the head.

The tail drop behavior inherent to FIFO queues has undesirable effects on TCP flows, which are bursty in nature. A FIFO queue can quickly grow over its limit due to a traffic burst, even if the average sending rate is not very high. When this happens, packets are denied admission to the queue (tail drop). TCP interprets this packet loss as a signal of link congestion, and slows down its sending rate through a feature known as *slow start*. For more information on TCP congestion management techniques see RFC 2001, *TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms*.

Since tail drop usually affects many flows at one time, multiple flows will throttle down their sending rate (by going into slow start) simultaneously. Soon afterwards, these flows will synchronously increase their sending rates, resulting in another packet burst, and more FIFO tail drops. This situation is known as "global TCP synchronization", and results in an interface bandwidth utilization graph resembling a "saw tooth", with periods of high utilization followed by periods of low utilization, and an average utilization rate significantly lower than interface available bandwidth.

To avoid such behavior the Random Early Detection (RED) feature can be used to selectively drop traffic before tail dropping is necessary. The idea of RED is to recognize potential interface congestion before the interface's FIFO queue is full, and start signaling individual TCP flows to slow down the sending rate. RED is known as a *congestion avoidance* technique, because through selective dropping it tries to prevent the output queue from filling up, which reduces the need for trail dropping. The key points about the RED technique are as follows.

First and foremost, the interface queueing method is still FIFO, but packets may be dropped before the queue is absolutely full. Next, instead of tracking the current size, or depth, of the output queue, which may vary quickly depending on packet bursts, an average depth is calculated to smooth out the queue depth every time a new packet arrives. This AverageQueueSize is calculated as:

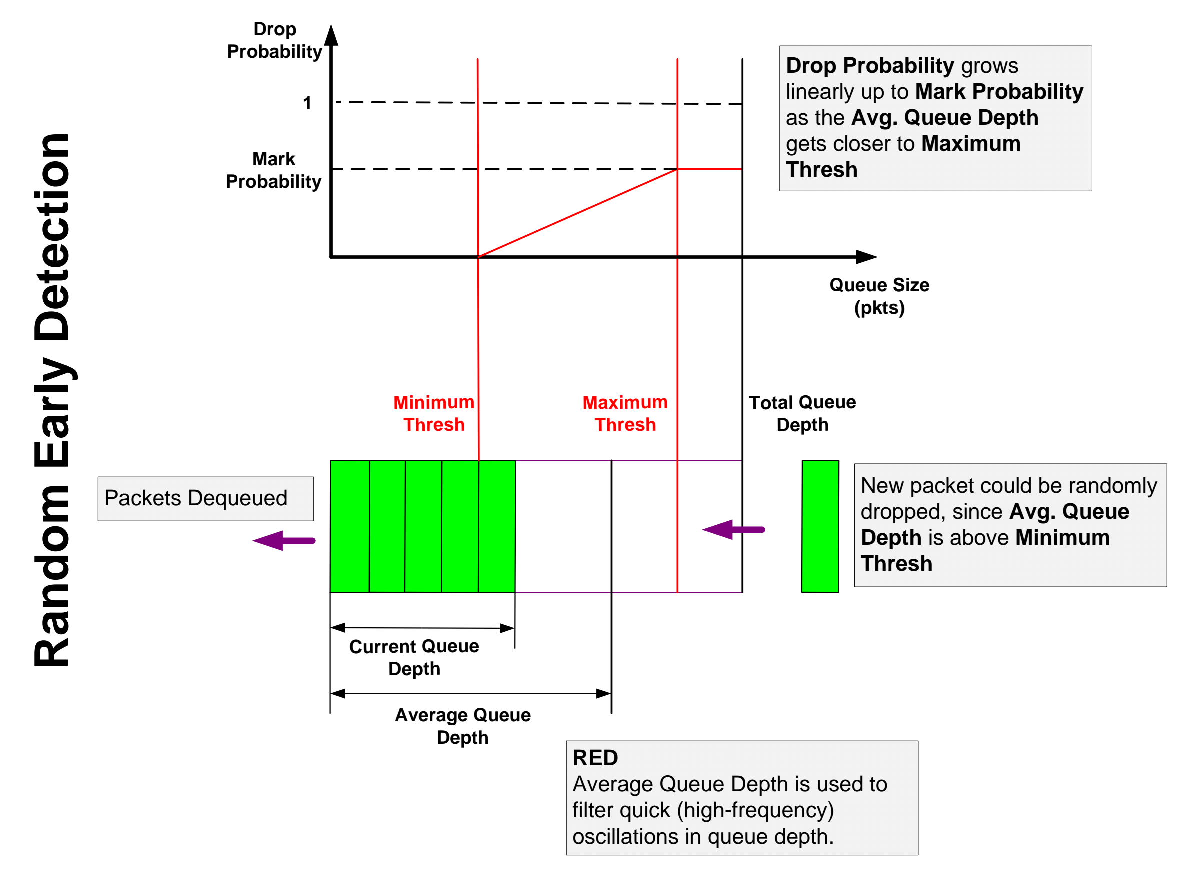
$$AvgQueueSize = OldQueueSize*(1-1/2^N) + NewQueueSize*1/(2^N)$$

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

This value slowly tracks the current queue size, lagging behind by some time interval, and dampens quick oscillations in the tracked value. The algorithm computes this average value every time a packet arrives or leaves the queue. Here *N* is the so-called *Exponential Weighting Constant*. The higher the value of *N*, the less sensitive the average queue size will be to the interface queue size changes.

To determine if and when a packet should be dropped from the queue, two thresholds are defined for the average queue depth, the *Minimum* and *Maximum* thresholds. Using these thresholds, the queue drop strategy is defined as follows.

If a new packet arrives and the *AvgQueueSize* computed is less than the *Minimum* value, the packet is accepted into the queue. If a new packet arrives and the *AvgQueueSize* is between *Minimum* and *Maximum*, then the packet is randomly dropped with the probability of *(AvgQueueSize-Minimum)/(Maximum-Minimum)*MarkProbability,* where *MarkProbability* is the last constant defined for RED (e.g. 1/10, 1/5, etc). If a new packet arrives and the *AvgQueueSize* is above *Maximum* but less than the total queue size, the packet is dropped with the probability of *MarkProbability*. Finally if a new packet arrives and the queue is 100% full, tail drop occurs.



Overall the effect is that the queue drops packets randomly before the queue size grows up to its limit. This allows for avoiding massive packet drops in multiple flows, resolving the TCP synchronization issue.

Cisco's implementation of the feature is known as Weighted Random Early Detection (WRED), or Weighted RED, which allows for defining different *Maximum* and *Minimum* thresholds as well as *MarkProbability* per IP Precedence of DSCP value. The result of WRED with the default values is that packets with a higher weighting (e.g. VoIP or routing updates) are less likely to be dropped as compared to lower weighted packets (e.g. FTP flows). All packets still share the same FIFO queue, however WRED starts dropping some packets earlier than others. As for the optimal values for WRED, they are subject to empirical verification. Cisco IOS automatically calculates the values based on the queue depth and WRED mode (IP Precedence or DSCP). The *Maximum* threshold value is automatically taken from the queue-depth of the link, meaning that if this threshold is breached packets are tail dropped. For most implementations the default values are sufficient.

RED configuration can be verified as follows. Note that you can see noticeable effects from RED only when you run multiple TCP flows on a relatively high-speed interface. For lower-speed WAN interfaces, its usually better to use WFQ.

```
Rack1R4#show interfaces serial 0/1
Serial0/1 is up, line protocol is up
  Hardware is QUICC Serial
  Internet address is 155.1.45.4/24
  MTU 1500 bytes, BW 128 Kbit, DLY 20000 usec,
     reliability 255/255, txload 247/255, rxload 35/255
  Encapsulation HDLC, loopback not set
  Keepalive set (10 sec)
  Last input 00:00:00, output 00:00:00, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops:
58
  Queueing strategy: random early detection(RED)
<snip>
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
Rack1R4#show queueing interface serial 0/1
Interface Serial0/1 queueing strategy: random early detection (WRED)
    Random-detect not active on the dialer
    Exp-weight-constant: 4 (1/16)
    Mean queue depth: 5

  class          Random drop      Tail drop    Minimum Maximum  Mark
                  pkts/bytes       pkts/bytes    thresh  thresh  prob
      0            41/22058          0/0            5       10  1/10
      1             0/0              0/0            5       10  1/10
      2             0/0              0/0            6       10  1/10
      3             0/0              0/0            6       10  1/10
      4             0/0              0/0            7       10  1/10
      5             0/0              0/0            7       10  1/10
      6             0/0              0/0           11       12  1/10
      7             0/0              0/0            8       10  1/10
   rsvp             0/0              0/0            9       10  1/10
```

For further verification configure R1 and R6 as HTTP servers and transfer their IOS images to SW2 and SW4. In addition to this, shutdown R5's Frame-Relay interface to ensure the traffic takes the path across the serial link between R4 and R5. You may also need to increase the MTU value on R4's serial interface to make the IOS TCP stack work properly on slow connections.

```
R1 & R6:
ip http server
ip http path flash:

R4:
interface Serial 0/1
 no ip mtu
 load-interval 30

R5:
interface Serial 0/0
 shutdown


Rack1SW2#copy http://admin:cisco@155.1.146.6/c2600-adventerprisek9-
mz.124-10.bin null:

Rack1SW4#copy http://admin:cisco@155.1.146.1/c2600-adventerprisek9-
mz.124-10.bin null:
```

Check the statistics and queue usage on R4:

```
Rack1R4#show interfaces serial 0/1
Serial0/1 is up, line protocol is up
  Hardware is QUICC Serial
  Internet address is 155.1.45.4/24
  MTU 1500 bytes, BW 128 Kbit, DLY 20000 usec,
     reliability 255/255, txload 247/255, rxload 37/255
  Encapsulation HDLC, loopback not set
  Keepalive set (10 sec)
  Last input 00:00:05, output 00:00:00, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops:
96
  Queueing strategy: random early detection(RED)
  30 second input rate 19000 bits/sec, 53 packets/sec
  30 second output rate 124000 bits/sec, 27 packets/sec
  <snip>

Rack1R4#show queueing interface serial 0/1
Interface Serial0/1 queueing strategy: random early detection (WRED)
    Random-detect not active on the dialer
    Exp-weight-constant: 4 (1/16)
    Mean queue depth: 5

  class          Random drop      Tail drop    Minimum Maximum   Mark
                  pkts/bytes       pkts/bytes    thresh  thresh   prob
      0            97/53749           0/0            5       10  1/10
      1              0/0              0/0            5       10  1/10
      2              0/0              0/0            6       10  1/10
      3              0/0              0/0            6       10  1/10
      4              0/0              0/0            7       10  1/10
      5              0/0              0/0            7       10  1/10
      6              0/0              0/0           11       12  1/10
      7              0/0              0/0            8       10  1/10
   rsvp              0/0              0/0            9       10  1/10
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
Rack1R4#show queue serial 0/1
Output queue for Serial0/1 is 5/0

Packet 1, linktype: ip, length: 580, flags: 0x88
  source: 155.1.146.1, destination: 155.1.108.10, id: 0xAA2C, ttl: 254,
  TOS: 0 prot: 6, source port 80, destination port 11000
    data: 0x0050 0x2AF8 0x32F8 0xD825 0xE64C 0xB783 0x5010
          0x0F62 0x70AA 0x0000 0xFCC7 0x5856 0x6DD5 0xCBAF

Packet 2, linktype: ip, length: 580, flags: 0x88
  source: 155.1.146.1, destination: 155.1.108.10, id: 0xAA2D, ttl: 254,
  TOS: 0 prot: 6, source port 80, destination port 11000
    data: 0x0050 0x2AF8 0x32F8 0xDA3D 0xE64C 0xB783 0x5010
          0x0F62 0xB6AF 0x0000 0xE39F 0xBE57 0xFB4E 0x7FF1

Packet 3, linktype: ip, length: 580, flags: 0x88
  source: 155.1.146.1, destination: 155.1.108.10, id: 0xAA2E, ttl: 254,
  TOS: 0 prot: 6, source port 80, destination port 11000
    data: 0x0050 0x2AF8 0x32F8 0xDC55 0xE64C 0xB783 0x5010
          0x0F62 0xA4AB 0x0000 0xBD11 0x63D9 0x3FE1 0x7CF3

Packet 4, linktype: ip, length: 580, flags: 0x88
  source: 155.1.146.1, destination: 155.1.108.10, id: 0xAA2F, ttl: 254,
  TOS: 0 prot: 6, source port 80, destination port 11000
    data: 0x0050 0x2AF8 0x32F8 0xDE6D 0xE64C 0xB783 0x5010
          0x0F62 0x2B2F 0x0000 0x3A70 0x0EE4 0x7B30 0x63F9

Packet 5, linktype: ip, length: 580, flags: 0x88
  source: 155.1.146.1, destination: 155.1.108.10, id: 0xAA30, ttl: 254,
  TOS: 0 prot: 6, source port 80, destination port 11000
    data: 0x0050 0x2AF8 0x32F8 0xE085 0xE64C 0xB783 0x5010
          0x0F62 0xA50E 0x0000 0xF18D 0x8535 0xFCF6 0x7784
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.9 Legacy Flow-Based Random Early Detection

- Enable flow-based RED on the point-to-point Serial interface of R4.
- Set the FIFO queue depth to 10 packets.
- Set the average flow depth scale factor to 2.
- Set the maximum number of flows to 16.

Configuration

```
R4:
interface Serial0/1
 random-detect
 random-detect flow
 random-detect flow count 16
 random-detect flow average-depth-factor 2
 hold-queue 10 out
```

Verification

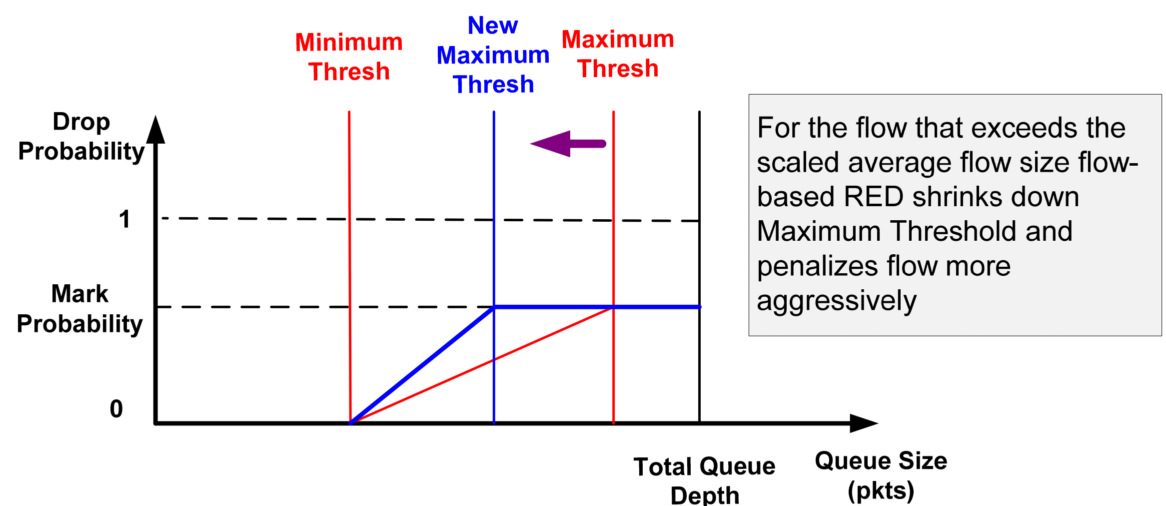
Note

Classic RED uses a FIFO queue, and does not distinguish between traffic flows (e.g. TCP/UDP sessions). This may lead to some undesirable effects, such as unevenly dropping packets in one flow while ignoring the others. This situation is more likely to happen when multiple flows with different behaviors occupy the same interface queue. RED packet drops could penalize an interactive section such as Remote Desktop Protocol (RDP) more when the interface queue is congested by an aggressive UDP flow, resulting in poor RDP session performance.

To resolve this issue, Cisco extended RED to support the concept of traffic flows. Using a special hash-function, flow-based RED classifies data streams into traffic flows, and tries to apply packet drops evenly across the flows. To accomplish this the interface's FIFO queue size *Q* is divided by the number of currently active flows *N* (flows that have queued packets in the interface queue). This yields the average queue size per flow, *Avg=Q/N*. For every flow, the flow depth *Depth* is compared with the scaled average queue size per flow as *Depth <= Avg*Scale*, where *Scale* is the scaling factor configured. If this assertion holds true, packets from the flow are not dropped. On the other hand if the flow size is greater than the scaled average queue size per flow, RED starts dropping packets from the flow more aggressively.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Specifically if the RED scheduler detects that a flows size is over the scaled average size, it shrinks down the maximum threshold for this flow using the formula *NewMaxThreshold = MinThreshold + (MaxThreshold-MinThreshold)/2*.



This procedure ensures that the aggressive flow is penalized more aggressively, and does not consume the queue space from other flows. For the flow-based RED it is possible to tune the maximum number of flows per interface queue and the scaling factor. This configuration can be verified as follows.

```
Rack1R4#show queueing random-detect
Current random-detect configuration:
  Serial0/1
    Queueing strategy: random early detection (WRED)
    Random-detect not active on the dialer
    Exp-weight-constant: 4 (1/16)
    Mean queue depth: 3
    Max flow count: 16       Average depth factor: 2
    Flows (active/max active/max): 0/0/16

  class          Random drop      Tail drop    Minimum Maximum  Mark
                  pkts/bytes       pkts/bytes    thresh  thresh  prob
      0              0/0             0/0            5      10   1/10
      1              0/0             0/0            5      10   1/10
      2              0/0             0/0            6      10   1/10
      3              0/0             0/0            6      10   1/10
      4              0/0             0/0            7      10   1/10
      5              0/0             0/0            7      10   1/10
      6              0/0             0/0           11      12   1/10
      7              0/0             0/0            8      10   1/10
   rsvp              0/0             0/0            9      10   1/10
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.10 Selective Packet Discard

- Enable Selective Packet Discard on R1 in aggressive mode.
- Increase R1's input queue size on its link to VLAN 146 to twice the default.
- Increase the amount of the memory headroom for IGP packets to 150 buffers.
- The headroom for BGP packets should be set to 120 packets.
- Start dropping low-priority packets randomly when the input queue is 50% full.

Configuration

```
R1:
spd extended-headroom 150
spd headroom 120
ip spd mode aggressive
ip spd queue max-threshold 150
ip spd queue min-threshold 75
!
interface FastEthernet 0/0
 hold-queue 150 in
```

Verification

Note

Selective Packet Discard is the queue management technique for interface input queueing. The SPD commands are hidden in the IOS parser, but you can see them in the running configuration once you enter them. By default SPD is enabled in Normal mode. The following is the list of SPD commands:

```
spd enable
spd headroom <N>
spd extended-headroom <N>
ip spd mode aggressive
ip spd queue max-threshold <N>
ip spd queue min-threshold <N>
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Every physical interface has an input FIFO queue. The router uses this queue to buffer packets going to the route processor. Usually these packets include control plane packets, such as Layer 2 Keepalives (e.g. HDLC/PPP keepalives), IGP packets (OSPF, ISIS, etc.) and BGP packets. The routing protocol packets are classified based on their default IP precedence of 6 or higher. In addition to control plane packets, the input queue holds other packets destined to the route processor, such as packets with an expired TTL, wrong header length, wrong checksum, or non-existent local UDP port numbers. The latter packets are malformed, in the sense that they require the router to generate an ICMP error message in response. Lastly the input queue holds packets that are to be process-switched, which is uncommon on modern CEF-based systems.

SPD input queueing is desirable for a number of reasons. The first is for control plane security. It's possible to block the router's input queue with a high rate of malformed packets, which effectively blocks legitimate routing traffic. The result is a control plane DoS against the router. The next reason is for layer 2 keepalive, IGP, and BGP traffic separation.

Large BGP tables generate considerably large updates. These updates could potentially block the input queue for some time, preventing the router from receiving keepalive packets or IGP updates. This may result in IGP adjacency flapping or layer 2 link loss reports. The third reason is due to issues with process switching.

If for some reason CEF is disabled, the IP INPUT process can result in regular IP traffic blocking the input queues, causing a loss of the control plane. SPD prevents this through input drops.

So how does SPD work? First, the input queue consists of two parts. One part is the regular hold queue, which is visible through **`show interface`** command, and the other part is the priority queue, which stores routing updates and keepalives. The processor serves the priority queue first until it is empty, and then switches to the regular hold-queue. Additionally, the priority queue consists of two parts, the SPD Headroom and the SPD Extended Headroom. The Extended Headroom queue is emptied before the SPD Headroom in a priority manner. Specifically input packets are queued as follows.

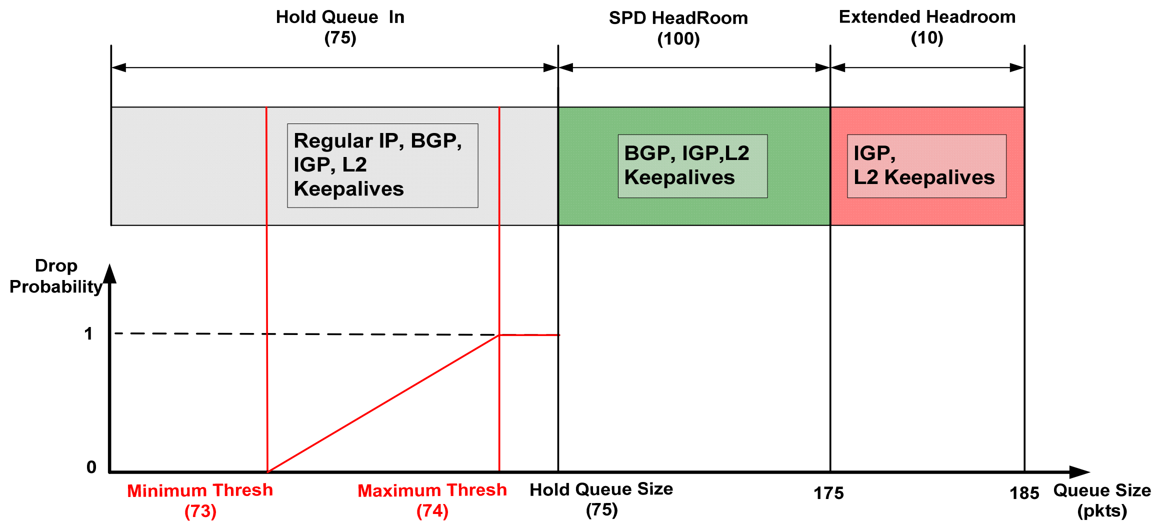
Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Layer 2 keepalives and IGP packets go to the SPD Extended Headroom. If there is no space available in the SPD Extended Headroom, packets go to the SPD Headroom. As a last resort, if both the Extended Headroom and Headroom are full, these packets go to the regular Hold Queue. BGP updates go directly to SPD Headroom. If the SPD Headroom is full, BGP packets hit the Hold Queue. All other IP packets (malformed or process-switched) go to the Hold Queue. The result is that L2 Keepalive/IGP packets are serviced first, BGP next, and other packets last.

Although the Hold Queue is FIFO, it uses the RED drop procedure. Two thresholds (*Minimum* and *Maximum*) set for hold queue define the random drop behavior. If the current hold queue length is less than the *Minimum Threshold*, packets are never dropped. If the queue length grows beyond *Minimum*, but is less than *Maximum*, every new packet is randomly dropped with the probability proportional to queue depth:

Prob = (*QueueDepth* – *MinimumThresh*)/(*MaximumThresh*-*MinimumThresh*)

If the queue depth is above *Maximum Threshold*, SPD drops every new incoming packet. See the figure below for an illustration using the SPD default values:



Note the important fact that SPD thresholds are global for all queues. SPD computes *Min* and *Max* thresholds based on the lowest hold-queue size in the system. Therefore if you set the hold queue size lower on some interfaces, you will affect all other interface drop thresholds.

Finally, SPD has two modes of operation, normal and aggressive. They differ in their treatment of malformed packets (packets that require the router to generate ICMP responses). When SPD is set for normal mode (the default) it treats malformed packets as it would all regular IP packets. It places them in the hold queue, subject to random drop. However, in aggressive mode, the malformed packets are dropped as soon as the hold queue grows above the minimum threshold. Effectively, SPD Aggressive mode replaces the random drop for malformed packets with an unconditional drop. SPD configuration can be verified as follows.

```
Rack1R1#show interface fastEthernet 0/0
FastEthernet0/0 is up, line protocol is up
  Hardware is AmdFE, address is 000d.659b.9140 (bia 000d.659b.9140)
  Internet address is 155.1.146.1/24
  MTU 1500 bytes, BW 100000 Kbit, DLY 100 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  Keepalive set (10 sec)
  Full-duplex, 100Mb/s, 100BaseTX/FX
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input 00:00:00, output 00:00:00, output hang never
  Last clearing of "show interface" counters never
  Input queue: 3/150/0/0 (size/max/drops/flushes); Total output drops:
0
  Queueing strategy: fifo
  Output queue: 0/40 (size/max)
<snip>

Rack1R1#show ip spd
Current mode: normal.
Queue min/max thresholds: 75/150, Headroom: 120, Extended Headroom: 150
IP normal queue: 2, priority queue: 0.
SPD special drop mode: aggressively drop bad packets
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.11 Payload Compression on Serial Links

- Configure PPP on the link between R1 and R3 using Predictor compression.
- Enable Stacker compression on the HDLC link between R2 and R3.
- Enable FRF.9 compression on the Frame Relay PVC between R2 and R5.

Configuration

```
R1:
interface Serial0/1
 encapsulation ppp
 compress predictor

R2:
interface Serial0/1
 compress stac
!
interface Serial0/0
 frame-relay map ip 155.1.0.5 205 broadcast IETF payload-compression
FRF9 stac one-way-negotiation

R3:
interface Serial1/2
 encapsulation ppp
 compress predictor
!
interface Serial 1/3
 encapsulation ppp
 compress stac

R5:
interface Serial0/0
 frame-relay map ip 155.1.0.2 502 broadcast IETF payload-compression
FRF9 stac one-way-negotiation
```

Copyright © 2009 Internetwork Expert

Verification

Note

Payload compression algorithms try to increase the available bandwidth on a link by removing redundant information present in data packets. Payload compression does not modify packet headers, unlike TCP or RTP header compression, but instead only applies to the packet's contents. Cisco IOS supports a variety of compression algorithms on serial links.

Some of these algorithms are proprietary (e.g. Frame-Relay packet-by-packet compression) and some are standard (e.g. FRF.9). Most of the compression procedures use the Lempel-Ziv algorithm and its descendants (e.g. LZS – Lempel-Ziv-Stac). LZ "Stacker" compression removes redundancy in the payload by replacing repetitive occurrences of the same strings with an index linking back to a dictionary of the most common strings. This procedure is memory efficient, yet consumes many CPU cycles. This is why compression is only allowed on slow serial links (HDLC, PPP, Frame-Relay, up to 2Mbps) and often routers use hardware accelerators to offload compression from their central processors. The "Predictor" compression algorithm uses a minimum of CPU cycles on the router, but needs more memory compared to the others. Instead of looking for repetitive substrings, it tries to predict every upcoming character in the flow, using the prediction tables common to both sending and receiving parties. The algorithm is very fast, but less efficient compared to the LZ family.

Some encapsulations support only a single compression procedure, such as Cisco's HDLC sole support for the Cisco proprietary "Stacker" implementation of LZS. The only encapsulation supporting the "Predictor" algorithm is PPP, and it is less effective than the LZ group of algorithms. Frame Relay supports per-VC compression through the **`frame-relay map`** command, either with the Cisco proprietary packet compression based on the LZS algorithm, or the standard FRF.9 compression. Note that using FRF.9 automatically changes the Frame Relay encapsulation to IETF from the default of "Cisco".

A key question that now arises is how compression interacts with other QoS features. Do traffic shapers and policers take compressed or uncompressed packet sizes into account? With Legacy QoS features (e.g. CQ, GTS, CAR) the byte counters and packet rates are based on uncompressed packet sizes. However, after the compression stage, the system informs the QoS feature about the actual packet size being sent on the line, and credits the respective queue.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

With MQC QoS features such as LLQ or CBWFQ, packets are compressed before computing the current burst or queue weight. Thus you can specify parameters based on compressed packet sizes, taking into account the compression effects.

Finally keep in mind that compression is only effective on redundant data streams, such as clear text. Sending already compressed payloads, such as PDFs, MP3s, GZIPs, or encrypted information will not yield any performance benefits.

To verify this configuration generate ICMP packets with highly redundant data payloads and view the compression statistics.

```
Rack1R1#show interfaces serial 0/1
Serial0/1 is up, line protocol is up
  Hardware is PowerQUICC Serial
  Internet address is 155.1.13.1/24
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation PPP, LCP Open
  Open: IPCP, CCP, CDPCP, loopback not set
  Keepalive set (10 sec)
  Last input 00:00:07, output 00:00:00, output hang never
  Last clearing of "show interface" counters 00:00:50
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: weighted fair
<snip>
```

Note that PPP negotiated the CCP phase (compression control protocol).

```
Rack1R1#show compress
 Serial0/1
         Software compression enabled
         uncompressed bytes xmt/rcv 0/0
         compressed bytes   xmt/rcv 0/0
         Compressed bytes sent:         0 bytes   0 Kbits/sec
         Compressed bytes recv:         0 bytes   0 Kbits/sec
         1  min avg ratio xmt/rcv 0.000/0.000
         5  min avg ratio xmt/rcv 0.000/0.000
         10 min avg ratio xmt/rcv 0.000/0.000
         no bufs xmt 0 no bufs rcv 0
         resyncs 0

Rack1R1#ping 155.1.23.3 size 1000 repeat 100 data AAAA

Type escape sequence to abort.
Sending 100, 1000-byte ICMP Echos to 155.1.23.3, timeout is 2 seconds:
Packet has data pattern 0xAAAA
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Success rate is 100 percent (100/100), round-trip min/avg/max = 52/55/112 ms
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
Rack1R1#show compress
 Serial0/1
         Software compression enabled
         uncompressed bytes xmt/rcv 100526/100526
         compressed bytes   xmt/rcv 14461/14472
         Compressed bytes sent:     14461 bytes   1 Kbits/sec  ratio: 6.951
         Compressed bytes recv:     14472 bytes   1 Kbits/sec  ratio: 6.946
         1  min avg ratio xmt/rcv 5.466/5.442
         5  min avg ratio xmt/rcv 5.466/5.442
         10 min avg ratio xmt/rcv 5.466/5.442
         no bufs xmt 0 no bufs rcv 0
         resyncs 0
```

Note the average RTT of 55ms. The link speed is 128Kbps, so a 1000 byte uncompressed packet would take approximately 125ms to serialize. Now verify the “Stacker” compression in sending packets from R2 to R3 and R5.

```
Rack1R2#ping 155.1.0.5 size 1000 repeat 100 data ABAB

Type escape sequence to abort.
Sending 100, 1000-byte ICMP Echos to 155.1.0.5, timeout is 2 seconds:
Packet has data pattern 0xABAB
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Success rate is 100 percent (100/100), round-trip min/avg/max = 44/49/176 ms

Rack1R2#ping 155.1.23.3 size 1000 repeat 100 data ABAB

Type escape sequence to abort.
Sending 100, 1000-byte ICMP Echos to 155.1.23.3, timeout is 2 seconds:
Packet has data pattern 0xABAB
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Success rate is 100 percent (100/100), round-trip min/avg/max = 24/27/36 ms

Rack1R2#show compress
 Serial0/0 - DLCI: 205
         Software compression enabled
         uncompressed bytes xmt/rcv 101048/101744
         compressed bytes   xmt/rcv 6303/6591
         Compressed bytes sent:      6303 bytes   1 Kbits/sec  ratio: 16.031
         Compressed bytes recv:      6591 bytes   2 Kbits/sec  ratio: 15.436
         1  min avg ratio xmt/rcv 1.044/0.753
         5  min avg ratio xmt/rcv 1.044/0.753
         10 min avg ratio xmt/rcv 1.044/0.753
         no bufs xmt 0 no bufs rcv 0
         resyncs 0
         Additional Stac Stats:
         Transmit bytes:  Uncompressed =        0 Compressed =       5582
         Received bytes:  Compressed =       5863 Uncompressed =        0
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
Serial0/1
         Software compression enabled
         uncompressed bytes xmt/rcv 100996/102076
         compressed bytes   xmt/rcv 7553/7994
         Compressed bytes sent:      7553 bytes   2 Kbits/sec  ratio: 13.371
         Compressed bytes recv:      7994 bytes   2 Kbits/sec  ratio: 12.769
         1  min avg ratio xmt/rcv 12.566/12.017
         5  min avg ratio xmt/rcv 12.566/12.017
         10 min avg ratio xmt/rcv 12.566/12.017
         no bufs xmt 0 no bufs rcv 0
         resyncs 0
         Additional Stac Stats:
         Transmit bytes:  Uncompressed =       60 Compressed =       7553
         Received bytes:  Compressed =       8422 Uncompressed =        0
```

```
Rack1R2#show frame-relay map
Serial0/0 (up): ip 155.1.0.1 dlci 205(0xCD,0x30D0), static,
              CISCO, status defined, active
Serial0/0 (up): ip 155.1.0.3 dlci 205(0xCD,0x30D0), static,
              CISCO, status defined, active
Serial0/0 (up): ip 155.1.0.4 dlci 205(0xCD,0x30D0), static,
              CISCO, status defined, active
Serial0/0 (up): ip 155.1.0.5 dlci 205(0xCD,0x30D0), static,
              broadcast,
              IETF, status defined, active
              Payload Compression FRF9
```

Note that just one PVC on the Frame-Relay interface is affected by compression.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.12 Generic TCP/UDP Header Compression

- Optimize the bandwidth usage between R4 and R5 by reducing TCP and RTP header sizes.
- Allow for a maximum of 16 concurrent RTP and TCP sessions.
- R5 should optimize traffic only if it detects already optimized traffic from R4.

Configuration

```
R4:
interface Serial0/1
 ip tcp header-compression
 ip tcp compression-connections 32
 ip rtp header-compression
 ip rtp compression-connections 32

R5:
interface Serial0/1
 ip tcp header-compression passive
 ip tcp compression-connections 32
 ip rtp header-compression passive
 ip rtp compression-connections 32
```

Verification

Note

The idea behind header compression is that interactive applications and voice packets usually have a very small payload size compared to the IP/TCP/UDP/RTP header size. For example a telnet packet has 1 or 2 bytes of payload per key press, and VoIP has a 20 byte payload for a G.729 packet. On slow links (up to 768Kbps) the serialization delay for these packets can be significantly reduced if the header information is somehow reduced, and only the differences between sequential headers are sent across the link.

Header compression works between two directly connected routers, on a hop-by-hop basis. For each TCP or RTP data stream special contexts are created on each side containing the header information. The actual TCP or RTP packet is sent with the original header stripped, and carries just a small header of around 4 to 5 bytes depending on the protocol and its options (e.g. preservation of checksums). The receiving side selects the local context based on the small header, and then recreates the original header using the stored information plus the difference vector received. Periodically the router sends the full headers to refresh the contexts on both sides.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Cisco IOS allows for using different formats of header compression, as well as limiting the number of local compression contexts (flows). Additionally it is possible to configure the router for passive compression, which means that it only starts packet compression when it receives compressed information. Note that since each VoIP/TCP connection is usually bi-directional, it requires 2 contexts on every side (sending and receiving).

TCP header compression can be verified using telnet streams as follows.

```
Rack1R4#show ip rtp header-compression
RTP/UDP/IP header compression statistics:
  Interface Serial0/1 (compression on, Cisco, RTP)
    Rcvd:    0 total, 0 compressed, 0 errors, 0 status msgs
             0 dropped, 0 buffer copies, 0 buffer failures
    Sent:    0 total, 0 compressed, 0 status msgs, 0 not predicted
             0 bytes saved, 0 bytes sent
    Connect: 32 rx slots, 32 tx slots,
             0 misses, 0 collisions, 0 negative cache hits, 32 free contexts

Rack1R4#show ip tcp header-compression
TCP/IP header compression statistics:
  Interface Serial0/1 (compression on, VJ)
    Rcvd:    0 total, 0 compressed, 0 errors, 0 status msgs
             0 dropped, 0 buffer copies, 0 buffer failures
    Sent:    0 total, 0 compressed, 0 status msgs, 0 not predicted
             0 bytes saved, 0 bytes sent
    Connect: 32 rx slots, 32 tx slots,
             0 misses, 0 collisions, 0 negative cache hits, 32 free contexts
```

Open TCP connections from SW2 and SW4 through R4's Serial connection to R5.

```
Rack1SW2#telnet 155.1.45.4
Trying 155.1.45.4 ... Open


User Access Verification

Password: cisco
Rack1R4>en
Password: cisco

Rack1SW4#telnet 155.1.146.6
Trying 155.1.146.6 ... Open


User Access Verification

Password: cisco
Rack1R6>en
Password: cisco
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Check the connection contexts.

```
Rack1R4#show ip tcp header-compression serial 0/1 detail
TCP/IP header compression statistics:
Configured:
 Max Header 168 Bytes, Max Time 5 Secs, Max Period 256 Packets, Feedback On
Negotiated:
 Max Header 168 Bytes, Max Time 5 Secs, Max Period 256 Packets, Feedback On
TX contexts:

RX contexts:

  Connection 0:
   IP source: 155.1.108.10, IP destination: 155.1.146.6
   TCP source: 11003, TCP destination: 23
   Last packet   received is VJcompressed-tcp and last sequence received is 0

  Connection 1:
   IP source: 155.1.58.8, IP destination: 155.1.45.4
   TCP source: 11008, TCP destination: 23
   Last packet   received is VJcompressed-tcp and last sequence received is 0

Rack1R5#show ip tcp header-compression serial 0/1 detail
TCP/IP header compression statistics:
Configured:
 Max Header 168 Bytes, Max Time 5 Secs, Max Period 256 Packets, Feedback On
Negotiated:
 Max Header 168 Bytes, Max Time 5 Secs, Max Period 256 Packets, Feedback On
TX contexts:

RX contexts:

  Connection 0:
   IP source: 155.1.146.6, IP destination: 155.1.108.10
   TCP source: 23, TCP destination: 11003
   Last packet   received is VJcompressed-tcp and last sequence received is 0

  Connection 1:
   IP source: 155.1.45.4, IP destination: 155.1.58.8
   TCP source: 23, TCP destination: 11008
   Last packet   received is VJcompressed-tcp and last sequence received is 0
Rack1R5#
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Check the compression statistics.

```
Rack1R4#show ip tcp header-compression
    TCP/IP header compression statistics:
      Interface Serial0/1 (compression on, VJ)
        Rcvd:    253 total, 238 compressed, 0 errors, 0 status msgs
                 0 dropped, 0 buffer copies, 0 buffer failures
        Sent:    182 total, 168 compressed, 0 status msgs, 0 not predicted
                 5690 bytes saved, 3910 bytes sent
                 2.45 efficiency improvement factor
        Connect: 32 rx slots, 32 tx slots,
                 14 misses, 0 collisions, 0 negative cache hits, 32 free
    contexts
                 92% hit ratio, five minute miss rate 0 misses/sec, 0 max

Rack1R5#show ip tcp header-compression
TCP/IP header compression statistics:
  Interface Serial0/1 (compression on, VJ, passive)
    Rcvd:    182 total, 168 compressed, 0 errors, 0 status msgs
             0 dropped, 0 buffer copies, 0 buffer failures
    Sent:    253 total, 238 compressed, 0 status msgs, 0 not predicted
             8312 bytes saved, 2117 bytes sent
             4.92 efficiency improvement factor
    Connect: 32 rx slots, 32 tx slots,
             15 misses, 2 collisions, 0 negative cache hits, 32 free contexts
             94% hit ratio, five minute miss rate 0 misses/sec, 0 max
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.13 MLP Link Fragmentation and Interleaving

- Enable PPP encapsulation on the serial link between R4 and R5.
- Configure PPP Multilink with interleaving on this connection.
- Ensure the maximum serialization delay is 10ms.
- Assume the bandwidth of the link is 128Kbps.

Configuration

```
R4:
interface Virtual-Template1
 bandwidth 128
 ip address 155.1.45.4 255.255.255.0
 fair-queue
 ppp multilink fragment delay 10
 ppp multilink interleave
!
multilink virtual-template 1
!
interface Serial0/1
 bandwidth 128
 no ip address
 encapsulation ppp
 load-interval 30
 ppp multilink

R5:
interface Virtual-Template1
 bandwidth 128
 ip address 155.1.45.5 255.255.255.0
 fair-queue
 ppp multilink fragment delay 10
 ppp multilink interleave
!
multilink virtual-template 1
!
interface Serial0/1
 bandwidth 128
 no ip address
 encapsulation ppp
 load-interval 30
 ppp multilink
 clock rate 128000
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Verification

Note

The PPP multilink feature allows the binding of multiple physical interfaces into a single logical interface. Multilink PPP is usually used to load balance at layer 2, as opposed to layer 3. For example if two T1 links are provisioned between two routers, layer 3 load balancing occurs by inserting parallel routes in the routing table, or though some method such as EIGRP variance or BGP multipath. With PPP Multilink these T1's can be treated as one logical layer 3 link, removing the need for additional layer 3 control plane overhead. The multilink procedure shrinks outgoing packets into pieces, assigns them sequence numbers, and forwards them down the parallel physical links. Based on the physical interface bandwidth some links forward more fragments than others. Therefore this behavior can be used as a QoS solution in addition to a load balancing solution.

Consider the following problem: a company uses a slow link (less than T1 rate) to transfer data and VoIP traffic. VoIP packets are small (around 60 bytes) and the general goal of the network is to provide as little delay as possible when sending these packets across the link (e.g. 10ms). This is certainly possible, as it does not take that much time to serialize 60 bytes across the WAN link. The problem arises, however, when large data packets (e.g. 1500 bytes) and voice packets share the same link, as the large data packets may block the transmit ring of the line for a considerable length of time delaying the transmission of voice packets. This may result is delays much higher than 10ms for small voice packets. The solution to this problem consists of two steps.

First, split the large packets into smaller pieces so that each piece does not take more than 10ms to serialize across the slow link. Secondly, interleave the fragments of the large packets with small voice packets, so a large "train" of data fragments will not block the line before voice packets.

Different implementations of this idea exist, such as solving the problem at layer 3 versus layer 2. A layer 3 solution would be to reduce the IP MTU on the serial link, as seen in previous examples, which causes the router to fragment large packets at the IP layer. The interface software queue (e.g. WFQ) then interleaves the large fragments with the small voice packets. This problem with this solution is that excessive fragmentation of IP packets occurs, which is highly undesirable due to performance and security reasons. An alternate solution is to solve the problem with layer 2 fragmentation.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

With layer 2 fragmentation, fragments only exist on the point-to-point layer 2 link between two routers in the transit path, such a PPP link or Frame Relay PVC, as opposed to layer 3 fragmentation which occurs end-to-end. Multilink PPP with Link Fragmentation and Interleaving (MLP LFI) is one of these layer 2 schemes, and is accomplished as follows.

First, run PPP multilink on a **single** physical link (no real bundling) to activate fragmentation. Secondly, do not fragment IP packets smaller than a defined size (e.g. 60 bytes) but send them between fragments of large packets. The problem with this solution is that only works across one physical link. The sending of non--fragmented packets interleaved with PPP fragments across multiple links results in the receiver getting packets out of order, and losing sequence number synchronization. These out of order fragments are simply dropped, since there is no reliable re-sending procedure in MLP. The following requirements are necessary for MLP LFI to work effectively.

First, a software interface queue is needed that prioritizes voice (small) packets. This could be either simple WFQ, or some more advanced solution, like LLQ or IP RTP Priority. Next, an interleaving queue is needed that sits between the software queue (e.g. WFQ) and physical link queue (FIFO). This queue has two parts – high priority and normal priority. Small, non-fragmented packets use the high priority part, while large packet fragments go to the normal part. Using this queue the scheduler ensures that a group of data fragments never blocks any voice packet.

The interleaving queue is not visible when using MLP LFI, but it does exist. The fragment size for MLP is calculated based on two factors, the logical MLP interface bandwidth represented by the **`bandwidth`** command, and the fragment delay in milliseconds represented by the **`ppp multilink fragment delay`** command. Effectively the *fragment_size* = *bandwidth** *delay*.

It is important to remember that fragment size should be based upon the physical rate of the link (e.g. clock rate) not the logical bandwidth (e.g. PVC CIR). In the case of MLP Interleaving it is irrelevant, since PPP does not support logical circuits, but it's important when fragmenting packets over Frame Relay with PPP.

To solve the problem of multilink fragmentation and interleaving over multiple physical links at the same time, a standards based solution called Multiclass Multilink PPP is available. It allows MLP to keep track of traffic "classes" running across the MLP bundle. Each class has its own sequence numbering, independent of other classes. This permits encapsulating voice packets using fragmentation headers, while the receiving side can still recover the proper sequence numbers with interleaved packets.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

For this particular task there are multiple ways to configure MLP LFI. In this case we use a virtual-template interface to group the physical link, however we could also use either a Multilink interface or a Dialer Profile. Note that this solution also uses WFQ via the **`fair-queue`** command to provide more expedited treatment to small voice packets, and enable interleaving with the secondary queue.

To verify the MLP LFI configuration, first shut down the Frame-Relay interface of R5 to ensure all packets from VLAN 146 take the path across the Serial link between R4 and R5. Next, configure R6 to send SLA Jitter probes to R5 using G.729 sized packets, and configure R5 to respond to those probes. Lastly, configure R1 as an HTTP server and start transferring the IOS image from R1 to SW2 across the Serial link between R4 and R5.

```
R1:
ip http server
ip http path flash:

R5:
rtr responder
!
interface Serial 0/0
 shutdown

R6:
ip sla monitor 1
 type jitter dest-ipaddr 155.1.45.5 dest-port 16384 codec g729a
 timeout 1000
 frequency 1
ip sla monitor schedule 1 life forever start-time now


Rack1SW2#copy http://admin:cisco@155.1.146.1/c2600-adventerprisek9-
mz.124-10.bin null:
Loading http://admin:cisco@155.1.146.1/c2600-adventerprisek9-mz.124-
10.bin !!!!!!
```

Verify the status of the multilink bundle.

```
Rack1R4#show ppp multilink

Virtual-Access2, bundle name is Rack1R5
  Endpoint discriminator is Rack1R5
  Bundle up for 00:13:44, total bandwidth 128, load 241/255
  Receive buffer limit 12192 bytes, frag timeout 1000 ms
  Interleaving enabled
    0/0 fragments/bytes in reassembly list
    0 lost fragments, 0 reordered
    0/0 discarded fragments/bytes, 0 lost received
    0x10F51 received sequence, 0x18912 sent sequence
  Member links: 1 (max not set, min not set)
    Se0/1, since 00:13:44, 160 weight, 152 frag size
No inactive multilink interfaces

Rack1R4#sh interface virtual-access 2
Virtual-Access2 is up, line protocol is up
  Hardware is Virtual Access interface
  Internet address is 155.1.45.4/24
  MTU 1500 bytes, BW 128 Kbit, DLY 100000 usec,
     reliability 255/255, txload 155/255, rxload 77/255
  Encapsulation PPP, LCP Open, multilink Open
  Open: IPCP
  MLP Bundle vaccess, cloned from Virtual-Template1
  Vaccess status 0x40, loopback not set
  Keepalive set (10 sec)
  DTR is pulsed for 5 seconds on reset
  Last input 00:00:06, output never, output hang never
  Last clearing of "show interface" counters 00:05:57
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: weighted fair
  Output queue: 6/1000/64/0/8445 (size/max
total/threshold/drops/interleaves)
     Conversations  2/3/256 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 96 kilobits/sec
  5 minute input rate 39000 bits/sec, 61 packets/sec
  5 minute output rate 78000 bits/sec, 51 packets/sec
     20645 packets input, 1154494 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     16389 packets output, 3549759 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions
```

Note the size of the WFQ queue and the number of interleaves. Two conversations currently occupy the WFQ.

Now note the member (physical link) queue status. The queue is FIFO, and the original queueing method is not effective.

```
Rack1R4#show interface serial 0/1
Serial0/1 is up, line protocol is up
  Hardware is PowerQUICC Serial
  MTU 1500 bytes, BW 128 Kbit, DLY 20000 usec,
     reliability 255/255, txload 241/255, rxload 83/255
  Encapsulation PPP, LCP Open, multilink Open
  Link is a member of Multilink bundle Virtual-Access2, loopback not
set
  Keepalive set (10 sec)
  Last input 00:00:00, output 00:00:00, output hang never
  Last clearing of "show interface" counters 00:29:58
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 3
  Queueing strategy: weighted fair  [suspended, using FIFO]
  FIFO output queue 0/10, 3 drops
  30 second input rate 42000 bits/sec, 87 packets/sec
  30 second output rate 121000 bits/sec, 126 packets/sec
     99239 packets input, 6112709 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     9 input errors, 0 CRC, 9 frame, 0 overrun, 0 ignored, 0 abort
     143778 packets output, 17191470 bytes, 0 underruns
     0 output errors, 0 collisions, 16 interface resets
     0 output buffer failures, 0 output buffers swapped out
     22 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
```

Look at the WFQ contents on R4. Note the packet length for the voice conversation is 62 bytes (60 bytes plus 2 for PPP encapsulation) and the HTTP conversation is 578 bytes (576 bytes plus 2 for PPP). Also note the number of “Interleaves” next to the HTTP conversation.

```
Rack1R4#show queueing interface virtual-access 2
Interface Virtual-Access2 queueing strategy: fair
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: weighted fair
  Output queue: 7/1000/64/0/9079 (size/max total/threshold/drops/interleaves)
     Conversations  2/3/256 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 96 kilobits/sec

  (depth/weight/total drops/no-buffer drops/interleaves) 1/32384/0/0/0
  Conversation 79, linktype: ip, length: 62
  source: 155.1.146.6, destination: 155.1.45.5, id: 0x01A1, ttl: 254,
  TOS: 0 prot: 17, source port 54312, destination port 16384

  (depth/weight/total drops/no-buffer drops/interleaves) 6/32384/0/0/9010
  Conversation 137, linktype: ip, length: 578
  source: 155.1.146.1, destination: 155.1.58.8, id: 0x7397, ttl: 254,
  TOS: 0 prot: 6, source port 80, destination port 11004
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Now verify the IP SLA statistics at R6. Note that a large packet would take approximately 576*8/128000=36ms to serialize (plus the delay to reach R4 from R6 and additional timing overhead) without MLP LFI. At the same time, the average measured RTT is just 23ms, thanks to fragmentation and interleaving on R4.

```
Rack1R6#show ip sla monitor statistics 1
Round trip time (RTT)   Index 1
        Latest RTT: 23 ms
Latest operation start time: 05:59:38.896 UTC Thu Aug 14 2008
Latest operation return code: OK
RTT Values
        Number Of RTT: 1000
        RTT Min/Avg/Max: 14/23/50 ms
Latency one-way time milliseconds
        Number of one-way Samples: 0
        Source to Destination one way Min/Avg/Max: 0/0/0 ms
        Destination to Source one way Min/Avg/Max: 0/0/0 ms
Jitter time milliseconds
        Number of Jitter Samples: 999
        Source to Destination Jitter Min/Avg/Max: 1/7/35 ms
        Destination to Source Jitter Min/Avg/Max: 1/1/6 ms
Packet Loss Values
        Loss Source to Destination: 0           Loss Destination to Source: 0
        Out Of Sequence: 0      Tail Drop: 0    Packet Late Arrival: 0
Voice Score Values
        Calculated Planning Impairment Factor (ICPIF): 11
MOS score: 4.06
Number of successes: 16
Number of failures: 1
Operation time to live: Forever
```

Copyright © 2009 Internetwork Expert

10.14 Legacy Generic Traffic Shaping

- Configure shaping on R6 to limit the rate of packets going towards R4's Loopback0 via the connection to VLAN 146 to 128Kbps.
- Use a shaping interval of 10ms and disable the extended burst functionality.
- Limit the shaper's queue size to 1024 packets.

Configuration

```
R6:
access-list 104 permit ip any 150.1.4.0 0.0.0.255
!
interface FastEthernet 0/0.146
 traffic-shape group 104 128000 1280 0 1024
```

Verification

Note

Legacy Generic Traffic Shaping (GTS) was the first feature to limit the outbound traffic rate of an interface introduced into IOS. With GTS all traffic leaving an interface/subinterface can be shaped, or a subset of traffic can be classified using an access-list. The syntax is as follows.

```
traffic-shape rate <CIR> <Bc> <Be> <QueueLimit>
traffic-shape group <ACL> <CIR> <Bc> <Be> <QueueLimit>
```

<QueueLimit> sets the maximum size of the WFQ buffer space. WFQ is the built-in traffic-shaping queue scheduler. Additionally the system can inspect multiple traffic-shape statements on the interface in the order configured to classify the outgoing traffic in a top-down fashion. The additional arguments of the traffic-shape statement are explored in detail below.

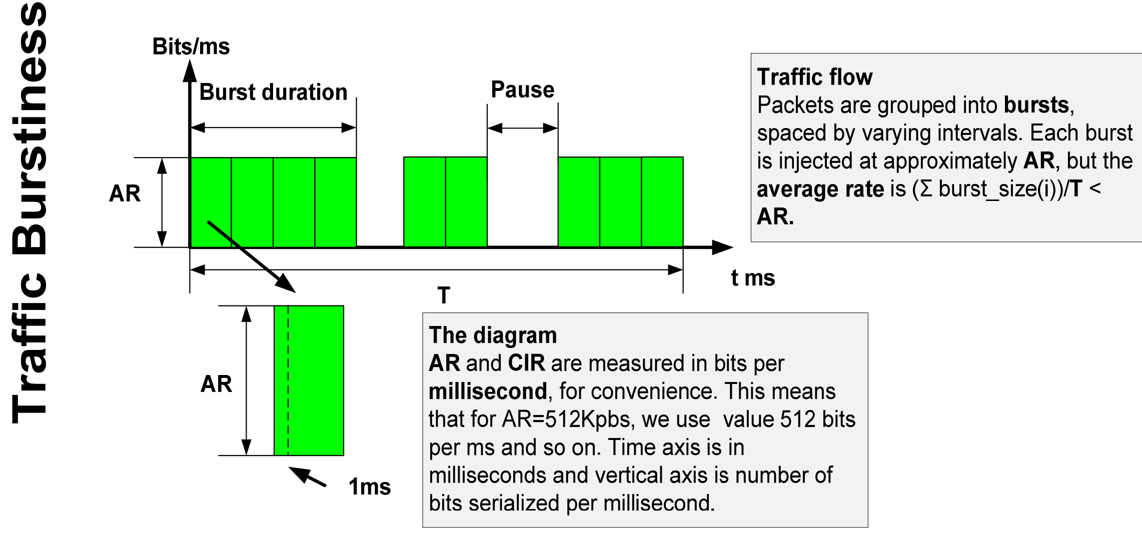
The purpose of traffic shaping is to "format" an outbound packet flow so that it conforms to a traffic contract. The formatting process slows down the average bitrate and the packet flow structure, resulting in a traffic flow consisting of uniformly spaced traffic bursts. Service provider traffic contracts are usually verified using ingress policers, such as Frame Relay Traffic Policing. The burst values used in outbound customer edge shaping typically should match the inbound metering function used on the service provider edge. This input metering function is explored further in the sections relating to policing with CAR.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

An example when traffic shaping is needed is the case that a customer's interface's physical rate, the Access Rate (AR), is higher than the contracted rate guaranteed by the service provider. For example a customer buys a Frame Relay circuit provisioned at 10Mbps, but the physical link to the provider is DS3 (45Mbps). Since the customer's interface always serializes packets outbound at 45Mbps, and the service provider performs traffic policing/admission control inbound, shaping is needed on the customer side to slow the average output rate down to 10Mbps.

Another case is when a remote site's connection speed is lower than the local site's. This means that the local side may overwhelm the remote side's connection by sending packets faster than the other side can accept. An example of this is in a WAN hub-and-spoke design, where the total sending rates of the spokes combined exceed the rate at which the hub can receive. By shaping the spoke sites down, the hub's site is not oversubscribed.

In both situations mentioned above, the physical AR is greater than the desired output rate. To slow the rate down, the first task of the shaper is to meter the traffic coming into the output queue, and decide whether it exceeds the target average rate. The concept of metering is based on the fact that traffic leaves an interface in a serial manner (bit by bit, packet by packet), and that packets are usually grouped in bursts, separated by periods of interface silence, as illustrated in the figure below.



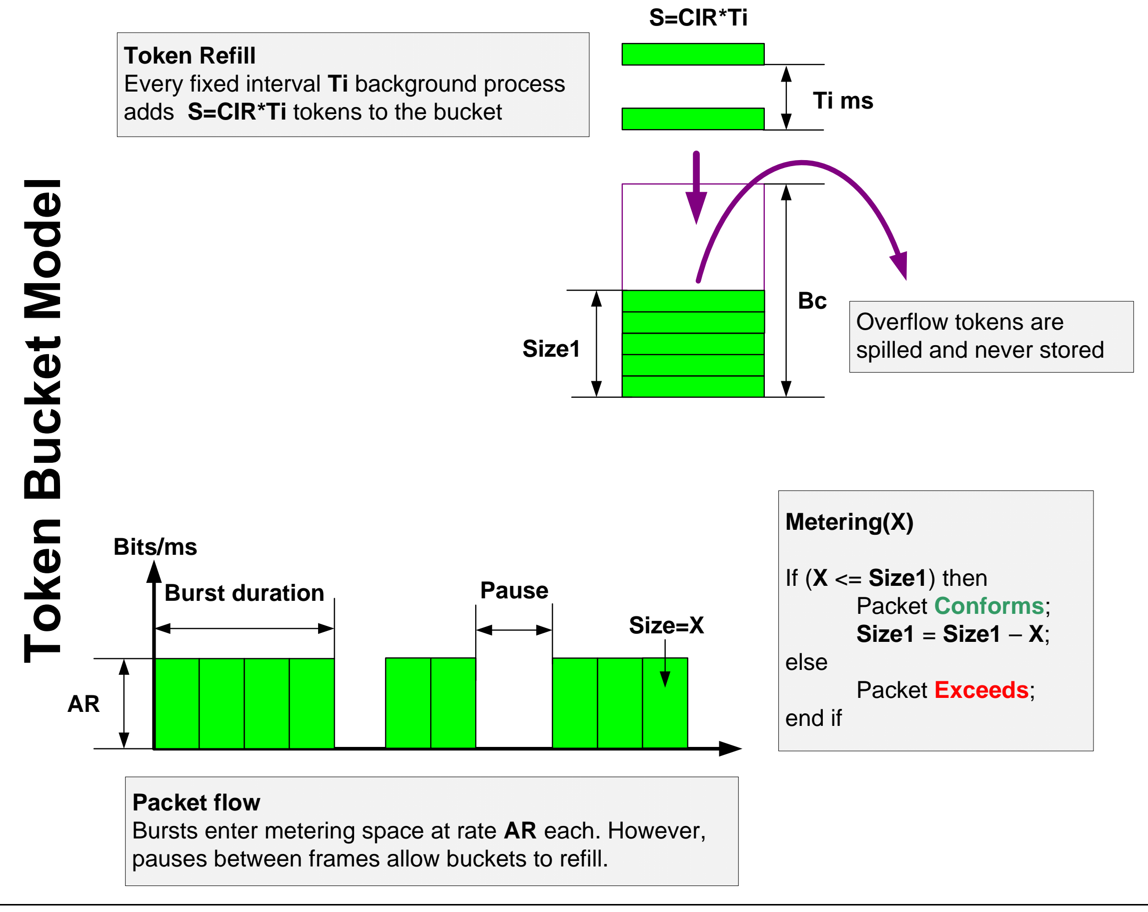
While the router sends each burst at AR speed, the spacing between bursts makes the average rate less than the AR. The goal of metering is to mark those bursts that exceed (do not conform to) the desired average rate, called the Committed Information Rate (CIR).

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Note that for the purpose of metering, the shaper's "CIR" value does not relate to the service provider's "CIR" value, where the latter actually means the "guaranteed" rate of the circuit based on the SLA. Instead, "CIR" in the context of shaping simply means the target average rate.

The metering function of traffic shaping uses what is known as a *token bucket* model to determine if traffic conforms to, or exceeds, the average rate. Every time a packet tries to be dequeued to the transmit ring, the metering function compares the size of the packet trying to leave to the amount of tokens, or credit, in the token bucket. If the size of the packet is less than or equal to the amount of credit, the packet conforms and is sent. If the size of packet is greater than the amount of credit in the token bucket, the packet exceeds, and is delayed.

The size of the token bucket is calculated by taking the desired average rate (CIR) in bits per second, and breaking it down into a smaller value of burst in bits per interval in milliseconds. These values are expressed as B_c (Burst Committed) bits, and T_c (Time Committed) milliseconds. The size of the token bucket is B_c bits, and the system refills the token bucket at a rate of B_c bits per T_c milliseconds. The key point here is that B_c bits per T_c interval is the same value as CIR bits per second, but is simply expressed in smaller units.



When a packet conforms to the average rate per interval it is dequeued to the transmit ring, and the number of tokens in the bucket equal to the size of the packet sent in bits is deducted. If packet exceeds because there are not enough tokens in the bucket, the shaping process delays the packets and holds it in the internal shaping queue. By this logic even though traffic is always sent at AR, the periods of delay incurred by non-conforming traffic in the shaping queue result in the overall average rate (CIR) being lower than the AR. This method is known the *leaky bucket* algorithm.

Leaky Bucket Model

Queue Depth

Tc

Input AR

Output AR

Bc

Leaky Bucket Process
The process queues packets coming at rate **input AR**. Every **Tc** interval scheduler runs and dequeues up to **Bc** amount of bytes and sends them at output **AR**.

Since the shaper sends packets in bursts every T_c milliseconds, the outgoing traffic "shape" becomes uniform, meaning that packet bursts of almost the same size are separated by T_c intervals. The size of T_c is configured indirectly by configuring the CIR and the B_c values based on the formula $B_c = CIR * T_c/1000$. Note that most documentation for traffic shaping denotes this as $B_c = CIR*T_c$, which is true as long as T_c is expressed in *seconds*, not *milliseconds*, which it is not by default.

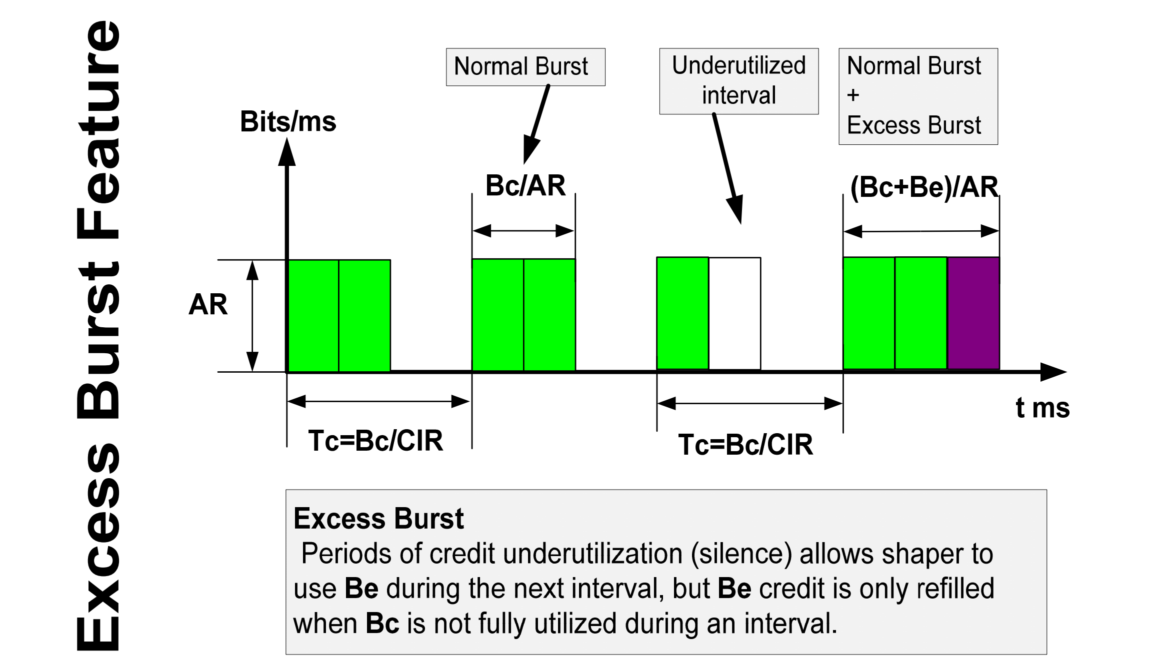
The minimal value for T_c is platform and version dependent, and is generally 10ms. This limits the scheduler "precision", but also puts a limit on the CPU utilization. Most importantly though, the value of T_c sets the minimum delay between packet bursts to 10ms. This generally limits the use of traffic shaping to WAN connections, since using it across a LAN connection may introduce unacceptably high delay. This point is explored in more detail within the context of priority queueing and traffic shaping simultaneously.

Copyright © 2009 Internetwork Expert

One possible problem with the above calculation for B_c is the case that the packet trying to be dequeued is larger than B_c, which means that there would never be enough credit in the token bucket to send it. For example if a packet's size is 1500 bytes, but the B_c is only 1000 bytes. To deal with this situation the shaper calculates a deficit counter (e.g. 1000-1500=-500) and adds this counter to the accumulated credit in the next round. In effect this reduces the amount of traffic to send the next time around. With this method although the scheduler sends a packet every time the accumulated credit is above zero, the accumulated credit will eventually drop to zero and the scheduler will have to wait 2 T_c intervals to accumulate credit and send the next packet. This may result in an average sending rate per T_c to be greater than or less than CIR, but the average rate over a longer period of time still never exceeds CIR. To avoid this problem altogether ensure that B_c is greater than the average packet size, which will achieve a smoother packet distribution. This is not always possible though, since there are cases when CIR value is too low. In the latter case layer 2 fragmentation can be introduced, which is covered in more detail in later tasks.

The next problem case that the scheduler can run into is when it has no traffic to send during a time interval (e.g. a pause in the packet stream), but it has more than B_c bytes to send in the next time interval. Based on the leaky token bucket algorithm, no more than B_c bytes can be sent per T_c interval, even if in previous intervals it did not send enough traffic. The result of this is that the shaper achieves *less* than the desired average rate. To resolve this problem traffic shaping uses what is known as a *dual leaky token bucket*, with the first token bucket represented as Committed Burst (B_c) and the second token bucket as Excess Burst (B_e).

The Excess Burst bucket is only filled in the case that the full B_c bucket was not emptied in the previous interval. The extra credit left over from the B_c bucket is then moved to the B_e bucket before B_c is refilled. For example if the B_c size is 10 bits, but only 8 bits were sent in the current interval, a credit of 2 bits can be moved to the B_e bucket if space is available. During the next interval the scheduler can now de-queue up to B_c+B_e bits. If B_c capacity is again not used completely, the left over credit is moved to B_e, up to its maximum size.



Like B_c, B_e has some finite size defined which controls how much credit can be stored. The size of the B_e bucket is constrained by the Access Rate of the physical link, since the packets are always serialized at this rate. Therefore the maximum B_e value (*maxB_e*) is equal to (AR-CIR)*T_c/1000 which implies that if the shaper sends B_c+maxB_e per T_c, it is sending at the Access Rate. The B_e value can be set lower than maxB_e, but should never exceed maxB_e. Note that since B_e is only populated due to a lack of B_c being used, the average sending rate over time still never exceeds the CIR. By default, with legacy GTS, if you do not specify a B_e value it is equal to B_c.

GTS configuration can be verified as follows.

```
Rack1R6#show traffic-shape

Interface   Fa0/0.146
       Access Target    Byte   Sustain   Excess    Interval  Increment Adapt
VC     List   Rate      Limit  bits/int  bits/int  (ms)      (bytes)   Active
-      104    128000    160    1280      0         10        160       -
```

In the above output the *Target Rate* is CIR in bps, *Byte Limit* is B_c+B_e in bytes – the maximum amount of data that could be sent during an interval - *Sustain bits/int* is B_c in bits, *Excess Bits/int* is B_e in bits, *Interval* is T_c in ms, and *Increment* is the B_c in bytes – the number of bytes that could be sent every regular interval.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

To see the traffic-shaper in action, configure R6 to send IP SLA jitter probes to R4's Loopback0 interface, and configure R4 to respond to the probes. Additionally configure R4 to meter the incoming traffic destined for the Loopback0 interface via the MQC.

```
R6:
ip sla monitor 1
 type jitter dest-ipaddr 150.1.4.4 dest-port 16384 codec g729a
 timeout 1000
 frequency 1
!
ip sla monitor schedule 1 life forever start-time now

R4:
rtr responder
!
access-group 120 permit ip any 150.1.4.0 0.0.0.255
!
class-map match-all TRAFFIC_TO_LO0
 match access-group 120
!
policy-map METER
 class TRAFFIC_TO_LO0
!
interface FastEthernet 0/1
 service-policy input METER
```

Now source ICMP ping packets from SW1 and SW3 to R4's Loopback0 IP address.

```
Rack1SW1#ping 150.1.4.4 repeat 1000000 size 160 timeout 1

Type escape sequence to abort.
Sending 1000000, 160-byte ICMP Echos to 150.1.4.4, timeout is 1
seconds:
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Rack1SW3#ping 150.1.4.4 repeat 1000000 size 160 timeout 1

Type escape sequence to abort.
Sending 1000000, 160-byte ICMP Echos to 150.1.4.4, timeout is 1
seconds:
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!
```

Now view the shaper queue with the **show traffic-shape queue** command. Note the weights assigned to each flow – they are the same since all flows use an IP precedence of 0. Also note the queue size, which is now 1024, the buffer limit we set. The system determines the maximum number of flow queues (conversations) automatically based on the CIR value provided for the shaper. This limits the traffic-shape command's usage for high-speed shapers, since automatic WFQ does not scale well over 2Mbps sending rates.

```
Rack1R6#show traffic-shape queue fastEthernet 0/0.146
Traffic queued in shaping queue on FastEthernet0/0.146
 Traffic shape group: 104
  Queueing strategy: weighted fair
  Queueing Stats: 3/1024/64/0 (size/max total/threshold/drops)
     Conversations  3/3/16 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 128 kilobits/sec

  (depth/weight/total drops/no-buffer drops/interleaves) 1/32384/0/0/0
  Conversation 9, linktype: ip, length: 78
  source: 155.1.146.6, destination: 150.1.4.4, id: 0x02EC, ttl: 255,
  TOS: 0 prot: 17, source port 53779, destination port 16384

  (depth/weight/total drops/no-buffer drops/interleaves) 1/32384/0/0/0
  Conversation 6, linktype: ip, length: 178
  source: 155.1.67.7, destination: 150.1.4.4, id: 0xEEB4, ttl: 254,
prot: 1

  (depth/weight/total drops/no-buffer drops/interleaves) 1/32384/0/0/0
  Conversation 4, linktype: ip, length: 178
  source: 155.1.79.9, destination: 150.1.4.4, id: 0xFD33, ttl: 253,
prot: 1

Rack1R6#show traffic-shape statistics
                Acc. Queue Packets   Bytes     Packets   Bytes     Shaping
I/F             List Depth                     Delayed   Delayed   Active
Fa0/0.146         104 2     1530365   216942630 1496672   212005056 yes
```

Note the packet sizes in the above output. Conversation 9 is 78 bytes long, which is our 60-byte "voice" packets plus the 18 byte overhead for an Ethernet header with a VLAN tag. The ICMP packets are 160 bytes plus the 18 bytes for Ethernet. This implies that WFQ uses the L2 packet sizes to calculate scheduling times and implement queue disciplines.

Now verify the input rate at R4.

```
Rack1R4#show policy-map interface fastEthernet 0/1
 FastEthernet0/1

  Service-policy input: METER

    Class-map: TRAFFIC_TO_LO0 (match-all)
      915439 packets, 125028746 bytes
      30 second offered rate 123000 bps
      Match: access-group 120

    Class-map: class-default (match-any)
      1768 packets, 522792 bytes
      30 second offered rate 0 bps, drop rate 0 bps
      Match: any.
```

The input rate is slightly less than the configured target rate of 128Kbps due to the layer 2 overhead taken into consideration by WFQ, but not by the MQC, and a slightly uneven packet distribution in the shaped packet stream.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.15 Legacy CAR for Admission Control

- Configure R4 to mark traffic received from R1's IP address 155.X.146.1 as it enters R4's connection to VLAN 146 using legacy committed access rate.
- Traffic up to 256Kbps should be marked with an IP precedence of 1.
- Exceeding traffic should be marked with an IP precedence of 0.
- Use an average traffic burst size of 4000 bytes.

Configuration

```
R4:
!
access-list 111 permit ip host 155.1.146.1 any
!
interface FastEthernet0/1
 rate-limit input access-group 111 256000 4000 4000 conform-action set-
prec-transmit 1 exceed-action set-prec-transmit 0
```

Verification

Note

The Legacy Committed Access Rate (CAR) feature, or rate-limiting, was designed for two purposes - admission control (e.g. packet remarking), and traffic limiting (e.g. policing) at the network edge. Unlike traffic shaping, rate limiting can be configured as both an input and output feature, and it does not buffer (delay) exceeding traffic bursts. Traffic classification for CAR can be based on all traffic on the interface, a standard or extended IP access-list, a MAC access-list (**`access-list rate-limit [0-99]`**), or an IP Precedence access-list (**`access-list rate-limit [100-199]`**).

Multiple rate-limit statements can be configured sequentially on the same interface, using the **`continue`** keyword, allowing for a hierarchical structure, such as the following:

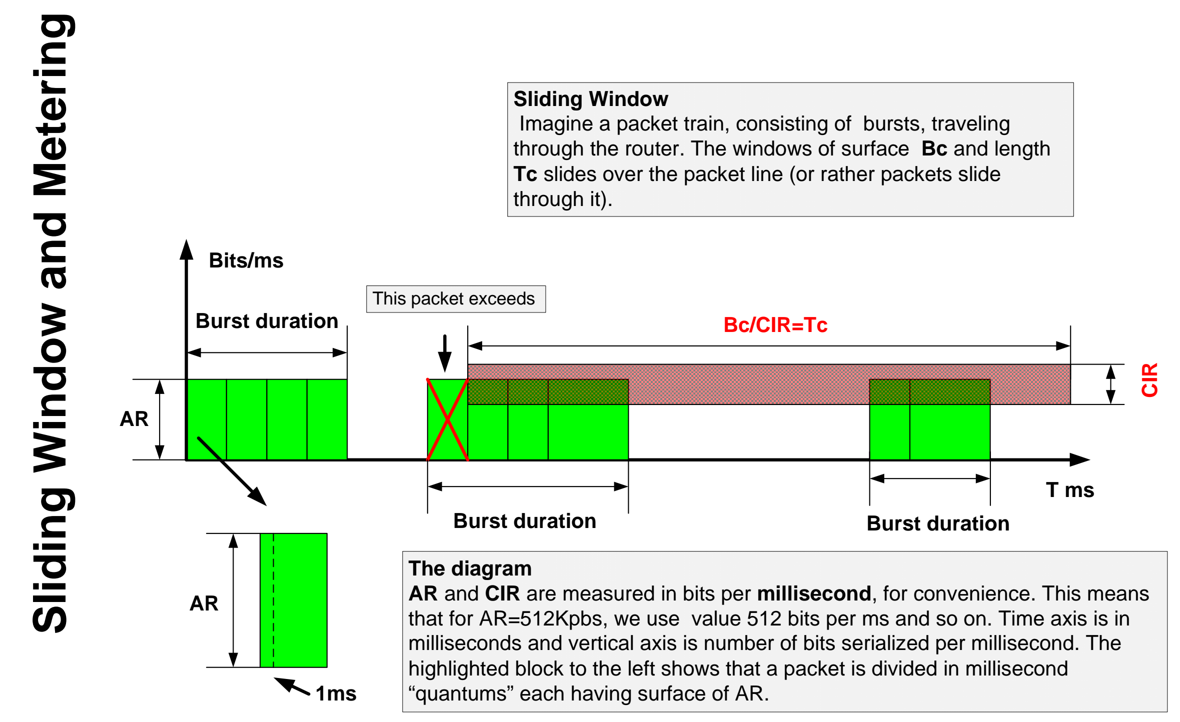
```
rate-limit input access-group 100 128000 8000 8000 conform-action transmit
exceed action continue
rate-limit input 256000 8000 8000 conform-action transmit exceed action drop
```

Configuration statements are evaluated sequentially until the first match is found and the specified action is executed. The **`continue`** keyword instructs the router to follow to the next statement in list, which makes the order significant.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Admission control through CAR is similar to how traffic-shaping's algorithm works, but with some key differences. Consider the situation in which a customer's edge device connects to the service provider via an interface with a physical Access Rate (AR) that is higher than the contracted rate (the provider's CIR). In order to enforce the contract, the provider needs to meter the input traffic's bitrate, and mark (dropping is also a form of marking) the packets based on the measured rate. Like in shaping the idea of metering is based on the fact that network traffic enters the interface in a serial manner (bit by bit, packet by packet), and that packets are usually grouped in bursts, separated by "islands" of networking interface silence. While the router receives each burst at AR, the spacing between bursts makes average input rate less than AR. The goal of metering is marking those bursts that conform or exceed the contracted average rate (CIR). For example, if two shortly separated bursts enter the router at AR speed, the average rate measured over those two bursts would be close to AR and much higher than CIR, and thus the second burst would most likely be exceeding.

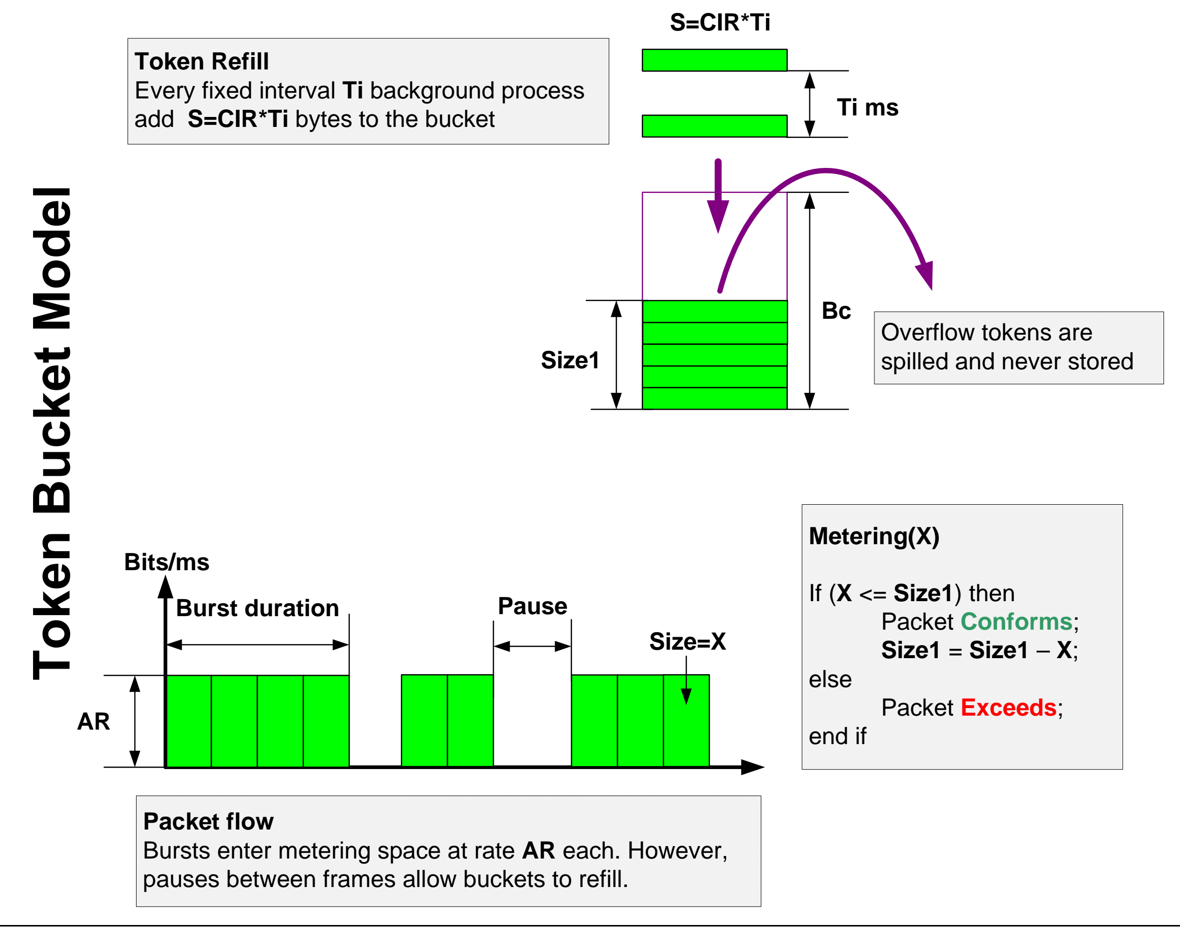
To measure the average speed, metering uses a sliding "averaging time interval" (T_c), not to be confused with traffic shaping's T_c. The process considers a new incoming packet as conforming if the amount of traffic already received during the current T_c, plus the size of the new packet, is less than or equal to Committed Burst (B_c). The interval T_c is sliding in the sense that it moves across the packet line as packets enter the router. The larger the T_c value, the greater amount of averaging that is performed over the input packet rate.



Copyright © 2009 Internetwork Expert

Note that like in traffic shaping, T_c, B_c and CIR values are not independent, since B_c=CIR*T_c. CIR is enforced by the fact that during any interval T_c, the amount of conforming traffic is no more than the B_c, and thus the average CIR the provider enforces is B_c per T_c. The difference between shaping and policing however, is that shaping produces packet bursts uniformly separated by interval T_c, each burst of size B_c, while CAR uses a sliding T_c window to measure the current average traffic rate and mark every new packet accordingly.

The metering implementation uses the same token bucket model as shaping, however the process does not delay exceeding packets, but instead simply remarks or drops them.



The token bucket for CAR works as follows. In the background the rate-limiting process adds CIR*T_i bytes to the token bucket ever T_i ms. For traffic shaping T_i = T_c, but for policing T_i is a constant value of about 1/8000 of a second. This is why you can only change the CAR rate in steps of 8000bps. Like shaping, the depth of the bucket is still defined as B_c. When a new packet of size *X* arrives, CAR consults the token bucket to see if it can borrow *X* credits. If there are enough tokens, the packet conforms, the conforming action is executed, and *X* tokens are subtracted from the bucket. If there are not enough tokens in the bucket, the process considers the packet as exceeding, and the exceed action is performed.

The key factor for correct inbound admission control with CAR depends on the burstiness of the traffic. If the source formats traffic in burst sizes that do not match the configured CAR burst, metering may be incorrect (e.g too much traffic is marked as exceeding or conforming). This is why customers connected to service providers should shape their traffic according to traffic contract parameters (CIR, B_c, etc) that match the policing settings on the service provider side.

Finally, similar to GTS, CAR computes the bandwidth used based on full layer 2 packet sizes, including the layer 2 encapsulation overhead.

To test this configuration set R1 to shape to uniform traffic bursts of 4000 bytes every 125ms, and generate packets using an unconstrained ping command with the packet size of 1000 bytes. Based on the below B_c value the traffic shaper will send 4x1000 byte packets every 125ms. In this case, the shaper burst size matches the CAR burst size:

```
R1:
interface FastEthernet0/0
 traffic-shape rate 256000 32000 0 1000

Rack1R1#ping 150.1.4.4 size 1000 timeout 0 repeat 1000000

Type escape sequence to abort.
Sending 1000000, 1000-byte ICMP Echos to 150.1.4.4, timeout is 0
seconds:
......................................................................
......................................................................
```

```
Rack1R4#show interfaces fastEthernet 0/1 rate-limit
FastEthernet0/1
  Input
    matches: access-group 111
      params:  256000 bps, 4000 limit, 4000 extended limit
      conformed 3841 packets, 3885702 bytes; action: set-prec-transmit 1
      exceeded 24 packets, 24336 bytes; action: set-prec-transmit 0
      last packet: 20ms ago, current burst: 3928 bytes
      last cleared 00:02:21 ago, conformed 219000 bps, exceeded 1000 bps
```

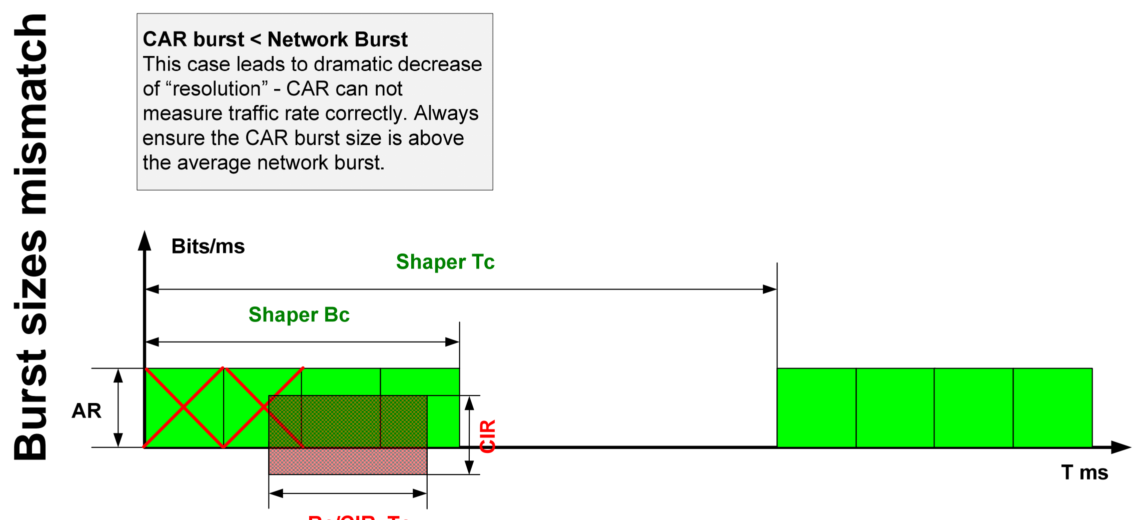
Some packets exceed due to small variations of inter-packet gaps, however since the burst values match on each side CAR isn't very sensate to the small variations.

Now let's see what happens when the CAR burst is set to a value that is *smaller* than the average network burst. To accomplish this change R1's shaper to emit bursts of twice the previous size (8000 bytes=8 packets of 1000 bytes) at the same rate (256Kbps) resulting in a T_c of 250ms. Leave the CAR configuration the same (CIR=256Kbps, B_c=4000bytes, T_c=125ms) on R4 and view the output.

```
R1:
interface FastEthernet0/0
 traffic-shape rate 256000 64000 0 1000

Rack1R1#ping 150.1.4.4 size 1000 timeout 0 repeat 1000000

Type escape sequence to abort.
Sending 1000000, 1000-byte ICMP Echos to 150.1.4.4, timeout is 0
seconds:
......................................................................
......................................................................
<snip>

Rack1R4#show interfaces fastEthernet 0/1 rate-limit
FastEthernet0/1
  Input
    matches: access-group 111
      params:  256000 bps, 4000 limit, 4000 extended limit
      conformed 1169 packets, 1180182 bytes; action: set-prec-transmit 1
      exceeded 817 packets, 828438 bytes; action: set-prec-transmit 0
      last packet: 232ms ago, current burst: 3928 bytes
      last cleared 00:01:18 ago, conformed 120000 bps, exceeded 84000 bps
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Although the shaper is sending at the same average rate, its burst size is twice as big as the compared to the CAR burst. This results in incorrect measurements, as CAR can not average across the whole bursts. The following figure illustrates this concept.



The small averaging interval does not allow for a correct rate measurement. It actually makes the CAR algorithm think traffic is being sent faster than the configured average rate. This example demonstrates the importance of picking the correct CAR burst size, which should be greater than the average traffic burst size. Remember though that this only applies to bursty flows. The case of uniform packet density is explored in the tasks relating to rate limiting with CAR.

10.16 Oversubscription with Legacy CAR and WFQ

- Configure WFQ on R4's point-to-point link to R5, and clock the link at 128Kbps.
- Configure a CAR policy on R4 to achieve the following:
  - 64Kbps that R4 receives from R1 and R6 each should be guaranteed on the link out to R5.
  - Traffic from R1 and R6 should be allowed up to 128Kbps each total.
  - Traffic above 128Kbps should be dropped.
  - Use the averaging time interval of 200ms.

Configuration

```
R4:
access-list 111 permit ip host 155.1.146.1 any
access-list 116 permit ip host 155.1.146.6 any
!
interface FastEthernet0/1
 rate-limit input access-group 111 64000 3200 3200 conform-action set-
prec-transmit 1 exceed-action continue
 rate-limit input access-group 111 128000 3200 3200 conform-action set-
prec-transmit 0 exceed-action drop
!
 rate-limit input access-group 116 64000 3200 3200 conform-action set-
prec-transmit 1 exceed-action continue
 rate-limit input access-group 116 128000 3200 3200 conform-action set-
prec-transmit 0 exceed-action drop
!
interface Serial 0/1
 fair-queue
 clock rate 128000

R5:
interface Serial 0/1
 clock rate 128000
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Verification

Note

Oversubscription is one of the prominent features of packet switched networks. Statistical multiplexing allows multiple users to share the same resources in dynamic proportions. The idea behind oversubscription is that not all users may need the same resources at the same time, and thus it's possible to grant some users the ability to temporary use more resources than the network can guarantee. This is in clear contrast with time-division multiplexing (TDM) networks, where users cannot exceed the guaranteed share of resources, such as a group of TDM slots.

In packet networks, service providers may sell more bandwidth than they can actually provide. For example a 10Mbps link could be shared between 10 users, with the agreement that each user can pull down up to 10Mbps if nobody else uses the link, but users are only guaranteed 1Mbps of link bandwidth in the case of heavy congestion. The actual implementation of this "downstream" sharing may vary, but most often it utilizes some sort of "fair-queuing" (slow links) or "round-robin" (high-speed links) scheduling, which implements the concept of max-min fairness for resource sharing.

The same idea can apply to traffic sent from users to the network ("upstream" oversubscription). A provider may allow customers to send more traffic than it can actually guarantee to deliver. If the network is not congested, it delivers all oversubscribed traffic. However if there is resource competition, the network delivers only the guaranteed part of the traffic. This brings in two important concepts – Committed Information Rate (CIR) and Peak Information Rate (PIR). (Note that CIR is similar to the concept used in traffic shaping, but it serves a different purpose here). CIR is the guaranteed rate, and PIR is the maximum acceptable rate. Often, the term EIR (Excess Information Rate) is also used, which is defined as EIR=PIR-CIR. To implement this "upstream" oversubscription, the provider uses some sort of traffic metering procedure, for example CAR or policing to mark the incoming traffic. Theoretically, a policer may sit at the ingress edge of the service provider's network, and meter traffic received from users, comparing it with the contracted CIR and PIR values. Note that the PIR value may be optional and the network may just meter against a CIR value. All the traffic marked above CIR is not guaranteed delivery across the network, so it could be dropped somewhere in case of congestion. However, in a carefully planned network, the sum of all incoming CIRs should not exceed the network capacity, and thus guarantee delivery for in-profile (conforming) traffic. As an extreme case, it is even possible for the provider to sell traffic with a CIR of zero, allowing users to send at any possible rate up to AR, but not guaranteeing any delivery.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

In this task the oversubscribed resource is the link between R4 and R5. R4 allows its customers (R1 and R6) to send up to a maximum of 128Kbps, which ensures that if one user is not sending, the other can use 100% of the link bandwidth. However, if both users are sending traffic simultaneously, we want them to share the link in equal proportions. To accomplish this, we utilize WFQ on the serial link.

Traffic received from the customers at a rate of 64Kbps or below is marked with an IP precedence of 1. If the traffic rate exceeds 64Kbps, the packets are marked with an IP precedence of 0. Since WFQ shares bandwidth proportional to IP precedence, any customer's "conforming" traffic will preempt the "exceeding" traffic of the other customer. Additionally a second nested rate-limit statement drops traffic from each customer if the rate exceeds 128Kbps.

The burst size is calculated based on a maximum transmit rate of 128Kbps and averaging interval of 200ms, where B_c = 128000/8 *200/1000 = 3200 bytes. Note that CAR takes the burst value in bytes, not bits, so 128000bps is divided by 8 to convert to bytes, and 200ms is divided by 1000 to convert to seconds.

Next consider the case where a customer's traffic causes congestion in the provider's network due to oversubscription. The service provider must have some way to signal the customer to slow down their sending rate. In legacy layer 2 networks, such as Frame Relay or ATM, there exists explicit layer 2 notifications (e.g. FECNs, BECNs) that allow the provider to directly tell customer equipment to slow down the sending rate to CIR. If no explicit congestion mechanisms exist, the provider could rely on upper-level protocol mechanics to respond to implicit congestion notification (e.g. packet delays or drops). An example of such upper layer protocols is TCP, which dynamically responds to packet loss by slowing down the sending rate. However, some upper level protocols do not respond to any implicit congestion notification (e.g. UDP packet floods). Such protocols pose serious problems to oversubscribed networks if no explicit Layer 2 signaling exists.

Based on their congestion behavior, traffic flows are commonly divided into three types: adaptive, aggressive, and sensitive. Adaptive flows, such as TCP, respond to implicit or explicit (e.g. TCP ECN) congestion notification by slowing down the sending rate. Aggressive flows, such as ICMP packet floods, do not respond to packet drops, and may monopolize network resources. Sensitive flows, such as voice traffic, are sensitive to congestion and/or packet drops. The provider should treat them in a special manner, such as using priority queuing.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

In general, the provider should not let aggressive flows enter their network at the peak rate. The provider needs to limit such flows at the edge by using an admission procedure. Sensitive flows should also not be oversubscribed. Only the adaptive flows are good candidates for oversubscription, due to their positive congestion response behavior. However, since TCP carries the majority of Internet traffic, oversubscription has become common in modern packet networks.

To verify this particular scenario enable the HTTP server service on R1 and R6, and configure R5 to meter the input traffic from R1 and R6 on its serial link to R4. Set the IP MTU to 576 on the serial interfaces of R4 and R5 to avoid problems with TCP traffic retransmissions and reordering due to fragmentation. Lastly shutdown the Frame Relay interface of R5.

```
R1, R6:
ip http server
ip http path flash:

R4:
interface Serial 0/1
 ip mtu 576

R5:
access-list 111 permit ip host 155.1.146.1 any
access-list 116 permit ip host 155.1.146.6 any
!
class-map match-all R1_PREC0
 match access-group 111
 match ip precedence 0
!
class-map match-all R1_PREC1
 match access-group 111
 match ip precedence 1
!
class-map match-all R6_PREC0
 match access-group 116
 match ip precedence 0
!
class-map match-all R6_PREC1
 match access-group 116
 match ip precedence 1
!
no policy-map METER
policy-map METER
 class R1_PREC0
 class R1_PREC1
 class R6_PREC0
 class R6_PREC1
!
interface Serial 0/1
 service-policy input METER
 ip mtu 576
!
interface Serial 0/0
 shutdown
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Set SW2 to download an IOS image from R1.

```
Rack1SW2#copy http://admin:cisco@155.1.146.1/c2600-adventerprisek9-mz.124-10.bin null:
Loading http://admin:cisco@155.1.146.1/c2600-adventerprisek9-mz.124-10.bin !!!!!!
```

Check the admission statistics on R4. Note that 64Kbps are conforming and almost the same amount is exceeding. R1 effectively uses the total serial link bandwidth.

```
Rack1R4#show interfaces fastEthernet 0/1 rate-limit
FastEthernet0/1
  Input
    matches: access-group 111
      params:  64000 bps, 3200 limit, 3200 extended limit
      conformed 48 packets, 28320 bytes; action: set-prec-transmit 1
      exceeded 46 packets, 27140 bytes; action: continue
      last packet: 24ms ago, current burst: 2874 bytes
      last cleared 00:00:03 ago, conformed 64000 bps, exceeded 61000 bps
    matches: access-group 111
      params:  128000 bps, 3200 limit, 3200 extended limit
      conformed 46 packets, 27140 bytes; action: set-prec-transmit 0
      exceeded 0 packets, 0 bytes; action: drop
      last packet: 28ms ago, current burst: 0 bytes
      last cleared 00:00:03 ago, conformed 61000 bps, exceeded 0 bps
    matches: access-group 116
      params:  64000 bps, 3200 limit, 3200 extended limit
      conformed 1 packets, 226 bytes; action: set-prec-transmit 1
      exceeded 0 packets, 0 bytes; action: continue
      last packet: 2545ms ago, current burst: 0 bytes
      last cleared 00:00:03 ago, conformed 0 bps, exceeded 0 bps
    matches: access-group 116
      params:  128000 bps, 3200 limit, 3200 extended limit
      conformed 0 packets, 0 bytes; action: set-prec-transmit 0
      exceeded 0 packets, 0 bytes; action: drop
      last packet: 235787ms ago, current burst: 0 bytes
      last cleared 00:00:06 ago, conformed 0 bps, exceeded 0 bps
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Check the metering results on R5. The traffic flow is split almost equally between precedence 0 (exceeding) and precedence 1 (conformed).

```
Rack1R5#show policy-map interface serial 0/1
 Serial0/1

  Service-policy input: METER

    Class-map: R1_PREC0 (match-all)
      310 packets, 179800 bytes
      30 second offered rate 58000 bps
      Match: access-group 111
      Match: ip precedence 0

    Class-map: R1_PREC1 (match-all)
      316 packets, 183280 bytes
      30 second offered rate 62000 bps
      Match: access-group 111
      Match: ip precedence 1

    Class-map: R6_PREC0 (match-all)
      0 packets, 0 bytes
      30 second offered rate 0 bps
      Match: access-group 116
      Match: ip precedence 0

    Class-map: R6_PREC1 (match-all)
      0 packets, 0 bytes
      30 second offered rate 3000 bps
      Match: access-group 116
      Match: ip precedence 1

    Class-map: class-default (match-any)
      3 packets, 1248 bytes
      30 second offered rate 0 bps, drop rate 0 bps
      Match: any
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Check the WFQ queue contents on R4. There should be one flow with alternating weights, IP Precedence 1 and IP precedence 0 packets hitting the same flow sequentially. Note that weight 16912 corresponds to an IP precedence of 1 and the weight 32384 corresponds to an IP precedence of 0.

```
Rack1R4#show queue serial 0/1
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 128
  Queueing strategy: weighted fair
  Output queue: 5/1000/64/128 (size/max total/threshold/drops)
     Conversations  1/3/256 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 1158 kilobits/sec

  (depth/weight/total drops/no-buffer drops/interleaves) 5/16192/0/0/0
  Conversation 137, linktype: ip, length: 580
  source: 155.1.146.1, destination: 155.1.58.8, id: 0x5CDA, ttl: 254,
  TOS: 32 prot: 6, source port 80, destination port 11004

Rack1R4#show queue serial 0/1
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 128
  Queueing strategy: weighted fair
  Output queue: 5/1000/64/128 (size/max total/threshold/drops)
     Conversations  1/3/256 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 1158 kilobits/sec

  (depth/weight/total drops/no-buffer drops/interleaves) 5/32384/0/0/0
  Conversation 137, linktype: ip, length: 580
  source: 155.1.146.1, destination: 155.1.58.8, id: 0x5D77, ttl: 254,
  TOS: 0 prot: 6, source port 80, destination port 11004
```

Now download R6's IOS image from SW4.

```
Rack1SW4#copy http://admin:cisco@155.1.146.6/c2600-adventerprisek9-
mz.124-10.bin null:
Loading http://admin:cisco@155.1.146.6/c2600-adventerprisek9-mz.124-
10.bin !!!!!!
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Check the admission control on R4. Since TCP is an adaptive protocol, it slowed down both sending rates to 64Kbps because of packet drops in the path. Now both flows are conforming, and no excess traffic is admitted to the network due to TCP's adaptive behavior.

```
Rack1R4#show interfaces fastEthernet 0/1 rate-limit
FastEthernet0/1
  Input
    matches: access-group 111
      params:  64000 bps, 3200 limit, 3200 extended limit
      conformed 403 packets, 236618 bytes; action: set-prec-transmit 1
      exceeded 0 packets, 0 bytes; action: continue
      last packet: 12ms ago, current burst: 834 bytes
      last cleared 00:00:29 ago, conformed 63000 bps, exceeded 0 bps
    matches: access-group 111
      params:  128000 bps, 3200 limit, 3200 extended limit
      conformed 0 packets, 0 bytes; action: set-prec-transmit 0
      exceeded 0 packets, 0 bytes; action: drop
      last packet: 83549ms ago, current burst: 0 bytes
      last cleared 00:00:29 ago, conformed 0 bps, exceeded 0 bps
    matches: access-group 116
      params:  64000 bps, 3200 limit, 3200 extended limit
      conformed 403 packets, 236678 bytes; action: set-prec-transmit 1
      exceeded 0 packets, 0 bytes; action: continue
      last packet: 60ms ago, current burst: 870 bytes
      last cleared 00:00:29 ago, conformed 63000 bps, exceeded 0 bps
    matches: access-group 116
      params:  128000 bps, 3200 limit, 3200 extended limit
      conformed 0 packets, 0 bytes; action: set-prec-transmit 0
      exceeded 0 packets, 0 bytes; action: drop
      last packet: 85100ms ago, current burst: 0 bytes
      last cleared 00:00:32 ago, conformed 0 bps, exceeded 0 bps
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Check the statistics on R5. We no longer see IP Precedence 0 packets, as sources don't send any traffic above the guaranteed rate.

```
Rack1R5#show policy-map interface serial 0/1
 Serial0/1

  Service-policy input: METER

    Class-map: R1_PREC0 (match-all)
      67184 packets, 9741533 bytes
      30 second offered rate 0 bps
      Match: access-group 111
      Match: ip precedence 0

    Class-map: R1_PREC1 (match-all)
      102271 packets, 15322824 bytes
      30 second offered rate 62000 bps
      Match: access-group 111
      Match: ip precedence 1

    Class-map: R6_PREC0 (match-all)
      3 packets, 1740 bytes
      30 second offered rate 0 bps
      Match: access-group 116
      Match: ip precedence 0

    Class-map: R6_PREC1 (match-all)
      1875 packets, 1086387 bytes
      30 second offered rate 62000 bps
      Match: access-group 116
      Match: ip precedence 1

    Class-map: class-default (match-any)
      535 packets, 93304 bytes
      30 second offered rate 0 bps, drop rate 0 bps
      Match: any
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Check the WFQ queue on R4. Now we see just two TCP flows with the same weights, corresponding to an IP Precedence value of 1.

```
Rack1R4#show queue serial 0/1
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops:
128
  Queueing strategy: weighted fair
  Output queue: 11/1000/64/128 (size/max total/threshold/drops)
     Conversations  2/3/256 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 1158 kilobits/sec

  (depth/weight/total drops/no-buffer drops/interleaves) 6/16192/0/0/0
  Conversation 194, linktype: ip, length: 580
  source: 155.1.146.6, destination: 155.1.108.10, id: 0xE2ED, ttl: 254,
  TOS: 32 prot: 6, source port 80, destination port 11004

  (depth/weight/total drops/no-buffer drops/interleaves) 5/16192/0/0/0
  Conversation 137, linktype: ip, length: 580
  source: 155.1.146.1, destination: 155.1.58.8, id: 0x4E8E, ttl: 254,
  TOS: 32 prot: 6, source port 80, destination port 11004
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.17 Legacy CAR for Rate Limiting

- Configure R6 to drop traffic received in its link to VLAN 146 in excess of 256Kbps.
- Use a committed burst of 384Kbps and an excess burst of 768Kbps.

Configuration

```
R6:
interface FastEthernet 0/0.146
 rate-limit input 256000 48000 96000 conform-action transmit exceed-
action drop
```

Verification

✐ Note

By using CAR with the exceed action set to “drop” it’s possible to limit the rate of both incoming or outgoing traffic. The syntax remains the same, but finding the optimal B_c value becomes a challenge. Also, note that the B_e value with CAR is not used to define the Peak Information Rate (PIR), but rather to implement a pseudo-random “exceed” action. This feature simulates random early detection’s drop behavior for TCP flows and prevents flow synchronization.

One serious obstacle exists here where policing with CAR though – the metering function assumes burstiness of incoming traffic. Usually a customer will pre-shape outbound traffic to match the inbound metering on the service provider side. As we’ve seen in the previous tasks, a mismatch in average network burst size and policer burst size can lead to incorrect metering. Based on this observation, how can we accurately estimate the burst size for network traffic if the flow is not already shaped?

Some protocols demonstrate inherent bursty behavior, for example TCP sends traffic in bursts up to the receiver’s window size. The receiver’s window size for TCP flows tends to be around *Traffic_Rate*RTT*, where *RTT* is the round-trip time for a particular connection. Therefore for a traffic flow made of a single TCP connection, the burst size could be found by multiplying the CIR by the maximum RTT. However this assumption only holds true for single flow, and only when the RTT is large enough to introduce a significant pause between bursts.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

In a practical situation, you cannot expect a flow to be truly bursty. At best it would be reasonable to assume the packet distribution in the flow to be close to uniform. In such situations, when no burst structure exists in a flow, a policer (CAR) would act as a “random” marker. Suppose that a flow sends packets of size S every T_s units of time. Then for any policer with parameters CIR and B_c, approximately $S/(CIR*T_s)$ percent of incoming packets will be marked as exceeding, in the case that $S/T_s > CIR$. Note that for uniform flows this percent of exceeding packets does not depend on committed burst size (B_c). However, the marking becomes more accurate with the B_c size growing, due to the large statistical sample. At the same time, a large B_c will inject lengthy packet trains in the network, possibly overfilling queues.

Uniform Traffic flow
The simplest model for traffic flow with average packet size **S** and inter-packet delay of **Ts**. Could be adequate for traffic mix, however S and Ts values may vary with time, as traffic flows are non-stationary.

Another consideration for burst size arises from TCP adaptive behavior. When a packet drop is detected by the TCP stack (via either a timeout or dup-ack) the sending side considers this as a signal of congestion. Different TCP implementations (e.g. Reno, Tahoe, New Reno, Vegas etc.) deal differently with packet loss, but in general, they all reduce the sending rate by some amount. However in real congestion, the sender first expects an increase in the RTT before packet losses, so TCP may adjust more smoothly. Even worse, multiple packet drops in a row may affect *multiple* flows (not just one) and signal them all to slow down. During the recovery phase, all flows will try to increase their sending rate *synchronously*, resulting in a sudden burst and another intense packet drop.

In this manner, multiple flows may "synchronize" and end up with a lower effective bandwidth utilization, due to their mutual influence. Based on those observations, we can see that TCP does not work ideally well with rate-limiting based on CAR, and we can never expect the average sending rate to get close to the configured CIR. However, simply increasing the B_c would make packet drops less "dense", and will allow the TCP sender to recover more quickly.

In practical situations there is no "optimal" value formula, instead the best approach is to find the optimal rate-limit B_c value empirically, starting with some reasonable value. Cisco recommends starting with B_c=CIR*1.5, and testing network performance with a growing window if necessary.

Another CAR enhancement, specifically targeted to rate-limit adaptive TCP flows, is called CAR Excess Burst or B_e. Note that CAR's B_e is totally unrelated to traffic-shaping's B_e, or classic traffic policing's B_e (e.g. as it is used with frame-relay traffic policing). To start, if the B_e value is equal to B_c, then the excess burst is disabled. If $B_e > B_c$ then for every new incoming packet the *probability* of a packet being marked as exceeding is proportional to $(Burst-B_c)/(B_e-B_c)$ where *Burst* is the amount of traffic accounted for in the last T_c units of time.

The actual implementation details of CAR's B_e are not important, but the resulting "random" behavior allows packets to be accepted even if the amount of traffic accepted last T_c seconds is greater than B_c. The result is that CAR marks fewer packets as exceeding (dropped) in a row, and distributes drops more evenly across the flow. This feature is specifically useful to avoid TCP flows "global synchronization", and resembles the Random Early Detection congestion avoidance behavior. Cisco recommends setting B_e to 2*B_c, if the rate-limited flow is based on TCP, but the optimal values can only be found empirically. Note that enabling B_e will increase the amount of packets admitted in your network over B_c during a single T_c, and may result in the measured traffic rate being slightly over CIR.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

To verify the effect of using CAR for policing, first disable B_e functionality by letting $B_c=B_e$. Next enable the HTTP server service on R1 and R4, and start downloading IOS images from those two routers to SW1 and SW3.

```
R1,R4:
ip http server
ip http path flash:

Rack1SW2#copy http://admin:cisco@155.1.146.4/c2600-adventerprisek9-mz.124-10.bin null:
Loading http://admin:cisco@155.1.146.4/c2600-adventerprisek9-mz.124-10.bin !!!!!!

Rack1SW3#copy http://admin:cisco@155.1.146.1/c2600-adventerprisek9-mz.124-10.bin null:
Loading http://admin:cisco@155.1.146.1/c2600-adventerprisek9-mz.124-10.bin !!!!!!
```

Check the rate-limiting statistics on R6.

```
Rack1R6#show interfaces fastEthernet 0/0.146 rate-limit
FastEthernet0/0.146
  Input
    matches: all traffic
      params:  256000 bps, 48000 limit, 48000 extended limit
      conformed 29025 packets, 15419612 bytes; action: transmit
      exceeded 9047 packets, 5333030 bytes; action: drop
      last packet: 0ms ago, current burst: 32213 bytes
      last cleared 00:08:04 ago, conformed 254000 bps, exceeded 87000 bps
```

Note the high number of exceeded packets, almost 23%. That usually means the TCP flows sending rate behaves in a “saw tooth” manner, accelerating over CIR and then rapidly falling down. Now set $B_e=2*B_c=96000$, download the same files again, and check R6’s statistics.

```
Rack1R6#show interfaces fastEthernet 0/0.146 rate-limit
FastEthernet0/0.146
  Input
    matches: all traffic
      params:  256000 bps, 48000 limit, 96000 extended limit
      conformed 2718 packets, 1501580 bytes; action: transmit
      exceeded 573 packets, 315637 bytes; action: drop
      last packet: 220ms ago, current burst: 66622 bytes
      last cleared 00:00:46 ago, conformed 256000 bps, exceeded 53000 bps
```

Note the amount of exceeding packets is now 17% - over a 6% decrease compared with the case above. In addition, the exceeded packet rate is less than it was before, thanks to the fact that the flows are no longer tightly synchronized.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.18 Legacy CAR Access-Lists

- Configure R1 to rate limit packets going out its link to VLAN 146 towards the MAC address of R4 to 128Kbps.
- Limit the rate of packets leaving R6 towards SW1 having IP precedence values of one, two, and four to 256Kbps.

Configuration

```
R1:
access-list rate-limit 100 000d.2846.8f21
!
interface FastEthernet 0/0
 rate-limit output access-group rate-limit 100 128000 8000 8000
conform-action transmit exceed-action drop

R6:
access-list rate-limit 0 mask 16
!
interface FastEthernet 0/0.67
 rate-limit output access-group rate-limit 0 128000 8000 8000 conform-
action transmit exceed-action drop
```

Verification

Note

CAR supports two types of special ACLs for rate limiting, MAC-based, and IP precedence based. Each ACL can contain just one line, but IP precedence based ACLs support matching multiple values at the same time by using a special mask format. The idea behind these ACLs is to create high-performance rate-limiting configurations based on just a single-line access-list.

For the MAC based access-list either issue the **`show ip arp`** command on R1 or **`show interface`** command on R4 to find R4's MAC address for the link. For the IP precedence access-list the mask is specified in hex, matching against the precedence values in binary as follows.

[p7][p6][p5][p4][p3][p2][p1][p0]

A binary 1 means to check for the precedence value, where 0 means to ignore (the opposite of a normal access-list wildcard). For example, 0x88 (10001000), mean IP precedence values 7 and 3. For our task precedence values 1, 2, and 4 are represented as 0x16 (00010110).

This configuration can be verified as follows.

```
Rack1R1#ping 155.1.146.4 repeat 1000 size 100 timeout 0

Type escape sequence to abort.
Sending 1000, 100-byte ICMP Echos to 155.1.146.4, timeout is 0 seconds:
..............................................................!.....
....!...!...!.......................................................
....................................................................
....................................................................
<snip>
Success rate is 0 percent (4/1000), round-trip min/avg/max = 1/4/8 ms

Rack1R1#show interfaces fastEthernet 0/0 rate-limit
FastEthernet0/0
  Output
    matches: access-group rate-limit 100
      params:  128000 bps, 8000 limit, 8000 extended limit
      conformed 164 packets, 18696 bytes; action: transmit
      exceeded 836 packets, 95304 bytes; action: drop
      last packet: 3265ms ago, current burst: 7942 bytes
      last cleared 00:09:05 ago, conformed 0 bps, exceeded 1000 bps
```

Note the number of matched packets. Now send the same amount of packets to R6 from R1 and note that the amount of matched packets did not change.

```
Rack1R1#ping 155.1.146.6 repeat 1000 size 100 timeout 0

Type escape sequence to abort.
Sending 1000, 100-byte ICMP Echos to 155.1.146.6, timeout is 0 seconds:
.........................................!..........................
...................................................................!
<snip>
Success rate is 2 percent (25/1000), round-trip min/avg/max = 1/2/8 ms

Rack1R1#show interfaces fastEthernet 0/0 rate-limit
FastEthernet0/0
  Output
    matches: access-group rate-limit 100
      params:  128000 bps, 8000 limit, 8000 extended limit
      conformed 164 packets, 18696 bytes; action: transmit
      exceeded 836 packets, 95304 bytes; action: drop
      last packet: 118875ms ago, current burst: 7942 bytes
      last cleared 00:11:00 ago, conformed 0 bps, exceeded 1000 bps
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Source a flood of packets from R6 to SW1 with the default IP precedence of 0.

```
Rack1R6#ping 155.1.67.7 repeat 1000 timeout 0

Type escape sequence to abort.
Sending 1000, 100-byte ICMP Echos to 155.1.67.7, timeout is 0 seconds:
!.!......!.!.!...!.!.!.!.!....!.!.!......!......!.!......!.!....!.!...
..!!.!...!.!.!...!.......!.!......!.!.!..!.!.!.!....!.!......!..!.....
<snip>
Success rate is 25 percent (251/1000), round-trip min/avg/max = 1/2/5 ms

Rack1R6#show interface fastEthernet 0/0.67 rate-limit
FastEthernet0/0.67
  Output
    matches: access-group rate-limit 0
      params:  128000 bps, 8000 limit, 8000 extended limit
      conformed 0 packets, 0 bytes; action: transmit
      exceeded 0 packets, 0 bytes; action: drop
      last packet: 43193693ms ago, current burst: 0 bytes
      last cleared 00:07:15 ago, conformed 0 bps, exceeded 0 bps
```

Note that the show command demonstrates no matches for the rate-limit. Now send packets marked with IP precedence 2 (TOS byte 64).

```
Rack1R6#ping
Protocol [ip]:
Target IP address: 155.1.67.7
Repeat count [5]: 1000
Datagram size [100]:
Timeout in seconds [2]: 0
Extended commands [n]: y
Source address or interface:
Type of service [0]: 64
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 1000, 100-byte ICMP Echos to 155.1.67.7, timeout is 0 seconds:
.!.!.....!..!...!.!.!.!.!.!.!......!....!.........!.!.........!.!.!.!
..!.!.!.....!...!.......!.!.!.......................................
<snip>
Success rate is 4 percent (44/1000), round-trip min/avg/max = 1/2/4 ms

Rack1R6#show interfaces fastEthernet 0/0.67 rate-limit
FastEthernet0/0.67
  Output
    matches: access-group rate-limit 0
      params:  128000 bps, 8000 limit, 8000 extended limit
      conformed 181 packets, 21358 bytes; action: transmit
      exceeded 819 packets, 96642 bytes; action: drop
      last packet: 11599ms ago, current burst: 7976 bytes
      last cleared 00:11:05 ago, conformed 0 bps, exceeded 1000 bps
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Now generate packets with IP precedence 4 (TOS 128).

```
Rack1R6#ping
Protocol [ip]:
Target IP address: 155.1.67.7
Repeat count [5]: 1000
Datagram size [100]:
Timeout in seconds [2]: 0
Extended commands [n]: y
Source address or interface:
Type of service [0]: 128
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 1000, 100-byte ICMP Echos to 155.1.67.7, timeout is 0 seconds:
..!..!.........!.!.......!.!.!.!.!......!.!.....!..!.!......!.!......
...!.!.!......!.!.!......................................!.............
<snip>
Success rate is 3 percent (38/1000), round-trip min/avg/max = 1/2/4 ms

Rack1R6#show interfaces fastEthernet 0/0.67 rate-limit
FastEthernet0/0.67
  Output
    matches: access-group rate-limit 0
      params:  128000 bps, 8000 limit, 8000 extended limit
      conformed 362 packets, 42716 bytes; action: transmit
      exceeded 1638 packets, 193284 bytes; action: drop
      last packet: 6242ms ago, current burst: 7976 bytes
      last cleared 00:15:39 ago, conformed 0 bps, exceeded 1000 bps
```

10.19 Legacy GTS for Frame Relay

- R3's Frame Relay service provider has agreed to transport up to 128Kbps of incoming traffic, but all traffic that exceeds 96Kbps is marked as Discard Eligible.
- Configure R3 with GTS to slow down to the guaranteed rate if the service provider signals congestion with BECN frames.
- Use 30ms for the traffic-shaping interval.

Configuration

```
R3:
interface Serial 1/0
 traffic-shape rate 128000 3840 0
 traffic-shape adaptive 96000
```

Verification

Note

Frame Relay service providers often practice oversubscription – the selling of more bandwidth than can actually be provided for by the network. This feature is typically allowed by a statistically multiplexed packet network architecture. The general idea is that all users are unlikely to use the maximum bandwidth at the same time, and thus most of the time resources are underutilized. By marking the oversubscribed traffic as low priority at admission, the Frame Relay network can signal QoS mechanics inside the cloud to start dropping oversubscribed traffic first, before the legitimate guaranteed traffic. In this manner the network can offer guarantees and oversubscriptions at the same time.

Frame Relay networks support this design through a combination of markings and explicit congestion notifications. Typically the provider marks frames sent at a rate above the contracted CIR with the Discard Eligible (DE) bit set as they enter the provider network. In case of network congestion the Frame Relay switches drop the DE-marked packets first before other frames. Additionally when congestion occurs, the Frame Relay network can signal the Customer Edge (CE) devices to slow down their sending rates through the Backward Explicit Congestion Notification (BECN), and the Forward Explicit Congestion Notification (FECN) bits in the Frame Relay header. On a congested link, the Frame Relay switch marks all frames going towards the destination with the FECN bit, and frames traveling back towards the sender with the BECN bit. The CE devices can then respond to these notifications through the support of *Adaptive* Frame Relay Traffic Shaping. Specifically this process happens on the provider side as follows.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

First, traffic received from the customer edge is subject to admission control through Frame Relay Traffic Policing (FRTP). The Service Provider Edge (PE) equipment meters the input traffic bursts against the configured policer. If the measured traffic rate is within the guaranteed rate (CIR), the PE admits the burst. If the measured rate is higher than CIR, but less than the Peak Information Rate (PIR), which is calculated as the CIR plus the Excessive Information Rate (EIR), the PE admits the burst, but marks the packet with the DE bit. This oversubscribed traffic is allowed in, but has no guarantee of delivery. Finally if the measured burst rate is above PIR, the burst is dropped.

Note that EIR is strictly a service provider side parameter, and it has nothing to do with excessive bursting at the customer side (defined by the customer's B_e shaping value). Excessive bursting is occasional, and only happens when the sending router has accumulated spare credits. Although the provider classifies excessive burst or traffic sent over the contract rate and marks it as DE, the client equipment does not send excessive burst constantly. Sending over the CIR, or sending at maximum allowed PIR=CIR+EIR rate, implies that the customer has their average rate (the shaper's CIR) set above the guaranteed rate.

As previously mentioned when congestion occurs inside the service provider network the switches start discarding DE-marked frames first, and start sending FECN towards the traffic destination and BECN towards the traffic source. The Legacy GTS shaper can respond to the BECN through the implementation of the *minCIR*. minCIR defines the rate which the shaper shrinks down its current B_c towards, while leaving the T_c value the same, when a BECN is received. The effect is that the normal CIR slows down to configured minCIR, or until BECNs stop being received. This feature is configured with GTS via the command **`traffic-shape adaptive <minCIR>`**. FECN adaptation is supported through a feature known as BECN reflection.

FECN adapt is useful in situations when the traffic flow over the link is unidirectional, such as a video broadcast server. Due to the absence of a return traffic flow, the Frame Relay network cannot piggyback BECNs in packets returning to the source, but the destination CE device still received packets marked with FECNs. When the receiver CE enabled FECN adapt, it generates its own dummy test frame with the BECN set back towards the source DLCI, effectively "reflecting" a BECN back. In this manner the receiver informs the unidirectional sender about congestion in the network. This feature is configured with GTS via the command **`traffic-shape fecn-adapt`**.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Although GTS can be applied to a Frame Relay encapsulated interface, as seen in this task, this configuration is technically not *Frame Relay Traffic Shaping* (FRTS). The main reason why is that GTS does not allow the attaching of a shaper to a particular PVC, only applying it to an interface or subinterface. Thus all active PVCs on an interface are shaped the same. True FRTS supports the attaching of different policies to different PVCs through a function known as *Per-VC Queueing*, which is explored in depth in later sections.

Another limitation of GTS applied to a Frame Relay interface is that with adaptive shaping, the reception of a BECN frame on any active PVC results in the shaper slowing down its sending rate for all VCs. Furthermore, the queueing technique for GTS is limited to WFQ, and does not allow using more advanced queueing techniques with Frame-Relay VCs.

In this particular task the service provider instructs the customer that the guaranteed rate (minCIR) is 96Kbps, and the desired average rate (CIR) is 128Kbps. Using the desired T_c value of 30ms we obtain the B_c value of B_c = CIR * T_c/1000 = 128000 * 30/1000 = 3840 bits. We don't set the B_e value since any traffic above 128Kbps is simply dropped by the provider.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

To verify the traffic shaping settings send a barrage of ping packets from SW1 to R5 and verify the resulting statistics. Note that the *VC* field is NULL, since FRTS Per-VC Queueing is not supported through GTS.

```
Rack1SW1#ping 155.1.0.5 repeat 1000000 timeout 0

Type escape sequence to abort.
Sending 1000000, 100-byte ICMP Echos to 155.1.0.5, timeout is 0
seconds:
..............................................................
..............................................................
..............................................................

Rack1R3#show traffic-shape

Interface   Se1/0
       Access Target    Byte   Sustain   Excess    Interval  Increment Adapt
VC     List   Rate      Limit  bits/int  bits/int  (ms)      (bytes)   Active
-             128000    480    3840      0         30        480       BECN


Rack1R3#show traffic-shape statistics
                  Acc. Queue Packets   Bytes     Packets   Bytes     Shaping
I/F               List Depth                     Delayed   Delayed   Active
Se1/0                   43    2517      906392    1891      684636    yes


Rack1R3#show traffic-shape queue serial 1/0
Traffic queued in shaping queue on Serial1/0
  Queueing strategy: weighted fair
  Queueing Stats: 21/1000/64/40994 (size/max total/threshold/drops)
     Conversations  1/3/16 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 128 kilobits/sec

  (depth/weight/total drops/no-buffer drops/interleaves) 21/32384/11009/0/0
  Conversation 10, linktype: ip, length: 104
  source: 155.1.37.7, destination: 155.1.0.5, id: 0x22B4, ttl: 254, prot: 1
```

Note the high number of packet drops due to buffer overflows. Also note that the packet length is 104, meaning that a 100 byte IP packet has a 4 byte Frame-Relay header.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.20 Legacy Frame Relay Traffic Shaping

- R2, R3, and R5's physical connection rates to the Frame Relay network are 128Kbps, 384Kbps, and 1536Kbps respectively.
- R3's contracted rate is 256Kbps, but the service provider allows occasional packet bursts up to the physical line capacity without discarding them; the provider does not allow sustained oversubscription by R3.
- Configure Legacy Frame Relay Traffic Shaping on R3 and R5 to meet the following requirements:
    - Ensure that R5 does not overwhelm R2's or R3's physical connections.
    - Ensure that R3 adheres to its service contract.
    - Use a Tc value of 10ms to support a VoIP deployment planned in the near future.

Configuration

```
R3:
map-class frame-relay TO_R5
 frame-relay cir 256000
 frame-relay bc 2560
 frame-relay be 1280
!
interface Serial 1/0
 frame-relay traffic-shaping
 frame-relay interface-dlci 305
  class TO_R5

R5:
map-class frame-relay TO_R2
 frame-relay cir 128000
 frame-relay bc 1280
 frame-relay be 0
!
map-class frame-relay TO_R3
 frame-relay cir 256000
 frame-relay bc 2560
 frame-relay be 0
!
interface Serial 0/0
 frame-relay traffic-shaping
 frame-relay interface-dlci 503
  class TO_R3
 frame-relay interface-dlci 502
  class TO_R2
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Verification

✎ **Note**

Legacy Frame Relay Traffic Shaping (FRTS) was intended as a replacement for legacy GTS through the support of Per-VC Queuing. When the command **frame-relay traffic-shaping** is applied on the main Serial interface, a separate queue is created per Virtual Circuit. Note that once enabled, a CIR of 56Kbps and a T_c of 125ms applies to all active PVCs, and should be adjusted accordingly.

Shaping parameters for FRTS are configured using **map-class frame-relay** command, which is then attached to a particular PVC with the **class <*name*>** command or to the interface with the **frame-relay class <*name*>** command. When applied to the VC level the map-class applies just to that VC, but when applied to the interface level the map-class applies to all VCs on the interface, or if it is the main interface, all VCs at the main interface and on any subinterfaces.

Legacy FRTS implements three levels of queueing, the per-VC queues, which are the logical queues of the per-VC shapers, the main interface FIFO queue, where packets go after they are dequeued from the per-VC queue, and the physical interface's transmit ring.

More advanced queueing strategies such as custom queueing, priority queuing, or CBWFQ, along with additional QoS features such as fragmentation and interleaving, and various forms of adaptive shaping, can be called from inside the frame-relay map-class. You cannot change the main interface's queuing strategy, but you can tune the queue size using **hold-queue out** command as well as adjust the transmit ring's size.

In this particular scenario R2 does not require any additional configuration, as its physical circuit speed already matches the provider's guarantee. On R3 the connection speed is higher than the contracted CIR, so shaping is needed. On R5 two classes are used to set individual CIRs to match the respective remote sites' connection speeds. Note that the values calculated for B_c and B_e use the same logic as GTS, where B_c = CIR * T_c/1000, and maxB_e = (AR – CIR) * T_c/1000. The Tc value is indirectly set to 10ms by configuring B_c to provide the minimum amount of latency possible for VoIP packets.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

This configuration can be verified as follows.

```
Rack1R5#show traffic-shape

Interface   Se0/0
       Access Target    Byte   Sustain   Excess    Interval  Increment Adapt
VC     List   Rate      Limit  bits/int  bits/int  (ms)      (bytes)   Active
502           128000    160    1280      0         10        160       -
503           256000    320    2560      0         10        320       -
504           56000     875    7000      0         125       875       -
513           56000     875    7000      0         125       875       -
501           56000     875    7000      0         125       875       -

Rack1R3#show traffic-shape

Interface   Se1/0
       Access Target    Byte   Sustain   Excess    Interval  Increment Adapt
VC     List   Rate      Limit  bits/int  bits/int  (ms)      (bytes)   Active
301           56000     875    7000      0         125       875       -
302           56000     875    7000      0         125       875       -
304           56000     875    7000      0         125       875       -
305           256000    480    2560      1280      10        320       -
```

Note that all PVCs, even non-configured PVCs, have shaping applied with the default CIR/T_c values 56Kbps/125ms. It is possible to verify the shaping settings per VC, as well as see the default queuing strategy for the per-VC shapers.

```
Rack1R3#show frame-relay pvc 305

PVC Statistics for interface Serial1/0 (Frame Relay DTE)

DLCI = 305, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial1/0

  input pkts 89754         output pkts 1033755      in bytes 10568524
  out bytes 120994552      dropped pkts 0           in pkts dropped 0
  out pkts dropped 0                out bytes dropped 0
  in FECN pkts 0           in BECN pkts 0           out FECN pkts 0
  out BECN pkts 0          in DE pkts 0             out DE pkts 0
  out bcast pkts 2558      out bcast bytes 908868
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
  pvc create time 06:34:36, last time pvc status changed 06:34:04
  cir 256000    bc 2560      be 1280      byte limit 480    interval 10
  mincir 128000    byte increment 320   Adaptive Shaping none
  pkts 496       bytes 176576    pkts delayed 0         bytes delayed 0
  shaping inactive
  traffic shaping drops 0
  Queueing strategy: fifo
  Output queue 0/40, 0 drop, 0 dequeued
```

If you want to change the per-VC FIFO queue depth configure it under the map-class settings using the **frame-relay holdq** command. In addition you can change the interface's FIFO queue depth using the **hold-queue** command. This can be verified as follows.

```
R3:
map-class frame-relay TO_R5
 frame-relay holdq 100
!
interface Serial 1/0
 hold-queue 50 out
```

```
Rack1R3#show interfaces serial 1/0
Serial1/0 is up, line protocol is up
<snip>
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops:
943611
  Queueing strategy: fifo
  Output queue: 0/50 (size/max)
  5 minute input rate 1000 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     92400 packets input, 10673708 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     92697 packets output, 10350543 bytes, 0 underruns
     0 output errors, 0 collisions, 2 interface resets
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
```

```
Rack1R3#show frame-relay pvc 305

PVC Statistics for interface Serial1/0 (Frame Relay DTE)

DLCI = 305, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial1/0

<snip>
  5 minute output rate 0 bits/sec, 0 packets/sec
  pvc create time 06:49:24, last time pvc status changed 06:48:52
  cir 256000    bc 2560      be 1280      byte limit 480    interval 10
  mincir 128000    byte increment 320   Adaptive Shaping none
  pkts 592       bytes 210752    pkts delayed 0         bytes delayed 0
  shaping inactive
  traffic shaping drops 0
  Queueing strategy: fifo
  Output queue 0/100, 0 drop, 0 dequeued
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.21 Legacy Adaptive FRTS

- The service provider used by R3 has agreed to allow link oversubscription up to 384Kbps, but only guarantees delivery up to 256Kbps.
- Configure R3 to send as fast as possible on this circuit, but to slow down its sending rate to the guaranteed rate when service provider network notifies it of congestion or the physical interface is congested.
- Ensure that R5 can inform R3 about congestion in the case that R5 is receiving a unidirectional traffic feed from R3.

Configuration

```
R3:
map-class frame-relay TO_R5
 frame-relay cir 384000
 frame-relay mincir 256000
 frame-relay bc 3840
 frame-relay be 0
 frame-relay adaptive-shaping becn
 frame-relay adaptive-shaping interface-congestion

R5:
map-class frame-relay TO_R3
 frame-relay fecn-adapt
```

Verification

> **Note**
>
> Legacy FRTS supports four types of adaptive shaping, BECN adapt, FECN reflection, queue depth monitoring, and a proprietary method called ForeSight. Like GTS on Frame Relay interfaces, BECN adapt allows the router to slow its sending rate down to minCIR when BECNs are received from the provider cloud. FECN reflection allows the receiving router to generate a test frame back to the sender with the BECN field set when a FECN is received from the provider network. Like previously discussed this feature is needed for unidirectional traffic flows, such as IPTV. Queue depth monitoring allows the sending router to slow its sending rate if the main interface's logical queue depth exceeds a certain threshold. Since the interface level queue with FRTS is FIFO, and is shared by all PVCs, once the interface queue reaches a configured threshold, all PVCs set to adapt to interface congestion slow their sending rate down to the configured minCIR. Queue depth monitoring is configured with the command **frame-relay adaptive-shaping interface-congestion <*depth*>** under the frame-relay map-class, as seen in the above example.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

In this particular scenario, the service provider allows R3 to send traffic above the guaranteed CIR. Note that in this case *CIR* refers to the guaranteed rate of the service provider contract, and not the average rate the router is shaping to with the **`frame-relay cir`** command. The traffic rate above the provider's CIR is called the Excess Information Rate (EIR), and the sum CIR+EIR is called the Peak Information Rate (PIR). In oversubscription scenarios it is common to call the provider's guaranteed rate "minCIR" and configure the **`frame-relay cir`** parameter to the value of PIR. This is where much confusion arises in Frame Relay Traffic Shaping, as the **`frame-relay cir`** command would be more appropriately named the *frame-relay average-rate*, as the router's CIR does not relate in any way to the provider's guaranteed CIR.

The router's parameters for this task are therefore as follows:

CIR = 384Kbps
minCIR = 256Kbps
T_c=10ms (for future voice traffic deployments per the previous task)
B_c = CIR * T_c/1000 = 384000 * 10/1000 = 3840 bits
B_e=0 (no Excess Bursting since we are already sending at the maximum allowed interface speed).

This configuration can be verified as follows.

```
Rack1R3#show frame-relay pvc 305

PVC Statistics for interface Serial1/0 (Frame Relay DTE)

DLCI = 305, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial1/0

<snip>
  Shaping adapts to BECN
  pvc create time 08:23:18, last time pvc status changed 08:22:46
  cir 384000    bc 3840      be 0         byte limit 480    interval 10
  mincir 256000    byte increment 480   Adaptive Shaping BECN and IF_CONG
  pkts 1202      bytes 427912    pkts delayed 0         bytes delayed 0
  shaping inactive
  traffic shaping drops 0
  Queueing strategy: fifo
  Output queue 0/100, 0 drop, 0 dequeued
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Send a barrage of ICMP packets from SW1 towards R5 and see if R3's output queue fills up and adaptive shaping kicks in.

```
Rack1SW1#ping 155.1.0.5 repeat 1000000 timeout 0

Type escape sequence to abort.
Sending 1000000, 100-byte ICMP Echos to 155.1.0.5, timeout is 0
seconds:
......................................................................
......................................................................

Rack1R3#show interfaces serial 1/0
Serial1/0 is up, line protocol is up
<snip>
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops:
1055360
  Queueing strategy: fifo
  Output queue: 50/50 (size/max)
  5 minute input rate 30000 bits/sec, 35 packets/sec
  5 minute output rate 29000 bits/sec, 35 packets/sec
<snip>

Rack1R3#show traffic-shape queue
Traffic queued in shaping queue on Serial1/0 dlci 305
  Queueing strategy: fcfs
  Queueing Stats: 96/100/71477 (size/max total/drops)
Packet 1, linktype: ip, length: 104, flags: 0x10000088
  source: 155.1.37.7, destination: 155.1.0.5, id: 0xA44E, ttl: 254, prot: 1
    data: 0x0800 0xDE91 0x001D 0x6E55 0x0000 0x0000 0x0223
          0x2F23 0xABCD 0xABCD 0xABCD 0xABCD 0xABCD 0xABCD

Packet 2, linktype: ip, length: 104, flags: 0x10000088
  source: 155.1.37.7, destination: 155.1.0.5, id: 0xA44F, ttl: 254, prot: 1
    data: 0x0800 0xDE90 0x001D 0x6E56 0x0000 0x0000 0x0223
          0x2F23 0xABCD 0xABCD 0xABCD 0xABCD 0xABCD 0xABCD

Packet 3, linktype: ip, length: 104, flags: 0x10000088
  source: 155.1.37.7, destination: 155.1.0.5, id: 0xA450, ttl: 254, prot: 1
    data: 0x0800 0xDE8F 0x001D 0x6E57 0x0000 0x0000 0x0223
          0x2F23 0xABCD 0xABCD 0xABCD 0xABCD 0xABCD 0xABCD

Packet 4, linktype: ip, length: 104, flags: 0x10000088
  source: 155.1.37.7, destination: 155.1.0.5, id: 0xA451, ttl: 254, prot: 1
    data: 0x0800 0xDE8E 0x001D 0x6E58 0x0000 0x0000 0x0223
          0x2F23 0xABCD 0xABCD 0xABCD 0xABCD 0xABCD 0xABCD
<snip>
```

The above output *Queueing stragety: fcfs* means First Come First Served, or in other words, FIFO.

10.22 Legacy FRTS with Per-VC WFQ

- Configure WFQ in R3's map-class for the PVC towards R5.
- Set the number of flow queues to 32, the queue size to 512, and start discarding packets when any flow reaches a length of 16 packets.

Configuration

```
R3:
map-class frame-relay TO_R5
 frame-relay fair-queue 16 32 0 512
```

Verification

Note

Legacy FRTS supports two levels of queuing, Per-VC, and per-interface. While the interface queue is fixed to a simple FIFO queue, it is possible to tune the Per-VC queues. Per-VC queuing allows moving congestion management to the VC level, thus preventing congestion at one PVC from affecting other VCs. Note that the VC queue is implemented using the shaper, thus the minimum time quantum for the queue scheduler is based on the minimum T_c for the platform.

The settings for Per-VC WFQ are the same as for legacy interface level WFQ, but the size of the queue is set with the **frame-relay fair-queue** command, not the **frame-relay holdq**. With Per-VC WFQ it's also possible to allocate RSVP flows, and activate other WFQ features such as IP RTP Priority. The syntax is:

frame-relay fair-queue <*CDT*> <*NumFlows*> <*RSVP Flows*> <*Max Buffers*>

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

This configuration can be verified as follows.

```
Rack1R3#show traffic-shape queue
Traffic queued in shaping queue on Serial1/0 dlci 305
  Queueing strategy: weighted fair
  Queueing Stats: 0/512/16/0 (size/max total/threshold/drops)
     Conversations  0/0/32 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 256 kilobits/sec
```

WFQ computes the available bandwidth based on the minCIR setting in the map-class. Note that if this value is not defined it defaults to half of the **`frame-relay cir`**. In any case the queue only activates when the PVC exceeds it's configured traffic rate, e.g. when the shaper starts delaying traffic.

```
Rack1R3#show frame-relay pvc 305

PVC Statistics for interface Serial1/0 (Frame Relay DTE)

DLCI = 305, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial1/0

<snip>
  cir 384000    bc 3840      be 0         byte limit 480    interval 10
  mincir 256000    byte increment 480   Adaptive Shaping BECN and IF_CONG
  pkts 148       bytes 52688     pkts delayed 0         bytes delayed 0
  shaping inactive
  traffic shaping drops 0
  Queueing strategy: weighted fair
  Current fair queue configuration:
   Discard     Dynamic      Reserved
   threshold   queue count  queue count
    16          32           0
  Output queue size 0/max total 512/drops 0
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.23 Legacy FRTS with Per-VC PQ

- Configure R5 to prioritize HTTP traffic sent to R3 over the Frame Relay link, but only if there are no RIP routing updates to send.
- ICMP traffic should only be sent if there is no other traffic queued on the link.

Configuration

```
R5:
access-list 100 permit icmp any any
!
priority-list 1 protocol ip high tcp www
priority-list 1 protocol ip low list 100
!
map-class frame-relay TO_R3
 frame-relay priority-group 1
```

Verification

> **Note**
>
> Legacy FTRS allows the attaching of different priority-lists on a per-VC basis, thereby allowing different per-VC priority queues. Create the priority-list as normal, then attach it to the PVC by issuing the **`frame-relay priority-group <ListNumber>`** command under the frame-relay map-class.
>
> Note that even though this task states that RIP packets should be dequeued before HTTP, no additional configuration is needed to do this since RIP updates will *not* be queued at the per-VC queue level. The reason is that by default Frame-Relay *broadcasts* (packets that require replication over multiple VC) are queued to special interface-level priority queue, called the frame-relay broadcast queue. This queue is separate from the interface-level FIFO queue shared by all VCs, and is discussed in more detail later.
>
> The operation of the priority queue can be verified as follows.

```
Rack1R5#show traffic-shape queue
Traffic queued in shaping queue on Serial0/0 dlci 501
  Queueing strategy: fcfs
Traffic queued in shaping queue on Serial0/0 dlci 513
  Queueing strategy: fcfs
Traffic queued in shaping queue on Serial0/0 dlci 504
  Queueing strategy: fcfs
Traffic queued in shaping queue on Serial0/0 dlci 503
  Queueing strategy: priority-group 1
Traffic queued in shaping queue on Serial0/0 dlci 502
  Queueing strategy: fcfs
```

```
Rack1R5#show frame-relay pvc 503

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 503, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

  input pkts 122699        output pkts 130586       in bytes 18493120
  out bytes 24033430       dropped pkts 0           in pkts dropped 0
  out pkts dropped 0                out bytes dropped 0
  in FECN pkts 0           in BECN pkts 0           out FECN pkts 0
  out BECN pkts 0          in DE pkts 0             out DE pkts 0
  out bcast pkts 36099     out bcast bytes 13978192
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 6000 bits/sec, 2 packets/sec
  pvc create time 3d03h, last time pvc status changed 1d10h
  cir 256000    bc 2560      be 0         byte limit 320    interval 10
  mincir 128000    byte increment 320   Adaptive Shaping none
  pkts 9035      bytes 1399132   pkts delayed 5275      bytes delayed 626532
  shaping inactive
  traffic shaping drops 0
  Queueing strategy: priority-list 1

List   Queue  Args
1      low    protocol ip          list 100

List   Queue  Args
1      high   protocol ip          tcp port www
  Output queue: high 0/20/0, medium 0/40/0, normal 0/60/0, low 0/80/0
```

To verify the priority queue temporarily set the shaping rate to 64Kbps for the map-class for the DLCI switched to R3. Next, enable the HTTP server service on R5 and start transferring IOS images from R5 to R3, and ping R3 from SW2. Set the IP MTU on R5's Frame-Relay interface to 80 bytes to allow for fast serialization of large HTTP packets.

```
R5:
ip http server
ip http path flash:
!
map-class frame-relay TO_R3
 frame-relay cir 64000
!
interface Serial 0/0
 ip mtu 80
```

```
Rack1R3#copy http://admin:cisco@155.1.0.5/c2600-adventerprisek9-mz.124-10.bin null:
Loading http://***********@155.1.0.5/c2600-adventerprisek9-mz.124-10.bin !!!!

Rack1SW2#ping 155.1.0.3 repeat 1000 size 100

Type escape sequence to abort.
Sending 1000, 100-byte ICMP Echos to 155.1.0.3, timeout is 2 seconds:
!!..!
<snip>
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Verify the queuing statistics on the PVC connecting R5 to R3. Note the queue usage counters for each queue.

```
Rack1R5#show frame-relay pvc 503

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 503, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

  input pkts 1991          output pkts 1952         in bytes 93680
  out bytes 167452         dropped pkts 23          in pkts dropped 0
  out pkts dropped 23               out bytes dropped 1932
  late-dropped out pkts 23          late-dropped out bytes 1932
  in FECN pkts 0           in BECN pkts 0           out FECN pkts 0
  out BECN pkts 0          in DE pkts 0             out DE pkts 0
  out bcast pkts 85        out bcast bytes 11060
  5 minute input rate 3000 bits/sec, 5 packets/sec
  5 minute output rate 3000 bits/sec, 6 packets/sec
  pvc create time 00:02:35, last time pvc status changed 00:02:22
  cir 64000     bc 2560      be 0         byte limit 320    interval 40
  mincir 32000     byte increment 320   Adaptive Shaping none
  pkts 1931      bytes 163288    pkts delayed 1669      bytes delayed 139588
  shaping inactive
  traffic shaping drops 0
  Queueing strategy: priority-list 1

List   Queue  Args

List   Queue  Args
1      high   protocol ip          tcp port www
1      low    protocol ip          list 100
  Output queue: high 14/20/23, medium 0/40/0, normal 0/60/0, low 2/80/0
```

Check the contents of the traffic-shaping queue.

```
Rack1R5#show traffic-shape queue serial 0/0 dlci 503
Traffic queued in shaping queue on Serial0/0 dlci 503
  Queueing strategy: priority-group 1
  Queueing Stats: high 9/20/116 (queue/size/max total/drops)
Packet 1, linktype: ip, length: 84, flags: 0x10000008
  source: 155.1.0.5, destination: 155.1.0.3, id: 0xCBC1, ttl: 255,
  TOS: 0 prot: 6, source port 80, destination port 24427
    data: 0x0050 0x5F6B 0x0CC9 0xF509 0x3D35 0x677E 0x5010
          0x1020 0x1C35 0x0000 0x6EFC 0x74A2 0xBA31 0x45A4

Packet 2, linktype: ip, length: 84, flags: 0x10000008
  source: 155.1.0.5, destination: 155.1.0.3, id: 0xCBC2, ttl: 255,
  TOS: 0 prot: 6, source port 80, destination port 24427
    data: 0x0050 0x5F6B 0x0CC9 0xF531 0x3D35 0x677E 0x5010
          0x1020 0x3B78 0x0000 0x409E 0xC9D5 0x9324 0x34CA
<snip>
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.24 Legacy FRTS with Per-VC CQ

- Configure R5 to share the bandwidth for the VC connecting R5 to R2 in the following proportions:
    - 60% of bandwidth for HTTP traffic
    - 20% of bandwidth for telnet traffic
    - 10% of bandwidth for ICMP traffic
    - RIP should use the system queue
- Limit each queue size to 15 packets.
- Ensure all protocols use the same average packet size of 80 bytes.

Configuration

```
R5:
access-list 100 permit icmp any any
!
queue-list 1 queue 0 limit 15
queue-list 1 protocol ip 1 tcp www
queue-list 1 protocol ip 2 tcp telnet
queue-list 1 protocol ip 3 list 100
!
queue-list 1 queue 0 limit 15
queue-list 1 queue 1 byte-count 600 limit 15
queue-list 1 queue 2 byte-count 200 limit 15
queue-list 1 queue 3 byte-count 100 limit 15
!
map-class frame-relay TO_R2
 frame-relay custom-queue-list 1
!
interface Serial 0/0
 ip mtu 80
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Verification

> ✎ **Note**
>
> Similar to legacy priority queueing, legacy custom queueing can be applied with FRTS on a per-VC basis. To apply this create the custom queue as normal, then attach it to the map-class with the **`frame-relay queue-list <ListNumber>`** command.
>
> As discussed in the previous task, RIP packets will not use the per-VC queue if they are broadcasts or multicasts, and instead will go to the broadcast queue. To force RIP packets to go into the per-VC queue the multicast updates can be converted to unicast updates by using the **`neighbor`** command under the RIP process.
>
> To verify this configuration, temporarily configure RIP unicasts across the Frame Relay interface, enable the HTTP server service on R5, and start a file transfer from R5 down to R2. Temporality shape R2's PVC on R5 down to 64Kbps to match the actual physical line rate.

```
R5:
router rip
 passive-interface Serial 0/0
 neighbor 155.1.0.2
 neighbor 155.1.0.3
 neighbor 155.1.0.4
 neighbor 155.1.0.1
!
map-class TO_R2
 frame-relay cir 64000
!
ip http server
ip http path flash:

Rack1R2#copy http://admin:cisco@155.1.0.5/c2600-adventerprisek9-mz.124-
10.bin null:
Loading http://***********@155.1.0.5/c2600-adventerprisek9-mz.124-
10.bin !!!!

Rack1SW2#ping 155.1.0.2 repeat 10000 size 80

Type escape sequence to abort.
Sending 1000, 100-byte ICMP Echos to 155.1.0.2, timeout is 2 seconds:
!!
<snip>
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Verify the custom queue statistics on VC 502 as follows.

```
Rack1R5#show frame-relay pvc 502

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 502, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

<snip>
  pvc create time 06:51:37, last time pvc status changed 06:51:25
  cir 128000    bc 1280      be 0         byte limit 160    interval 10
  mincir 64000     byte increment 160   Adaptive Shaping none
  pkts 197       bytes 15432     pkts delayed 145       bytes delayed 11132
  shaping active
  traffic shaping drops 0
  Queueing strategy: custom-list 1

List   Queue  Args

List   Queue  Args
1      0      protocol ip          udp port rip
1      1      protocol ip          tcp port www
1      2      protocol ip          tcp port telnet
1      3      protocol ip          list 100
1      0      limit 15
1      1      byte-count 600 limit 15
1      2      byte-count 200 limit 15
1      3      byte-count 100 limit 15
  Output queues: (queue #: size/max/drops/dequeued)
     0: 0/15/0/12 1: 2/15/47/148 2: 0/15/0/0 3: 0/15/0/6 4: 0/20/0/0
     5: 0/20/0/0 6: 0/20/0/0 7: 0/20/0/0 8: 0/20/0/0 9: 0/20/0/0
     10: 0/20/0/0 11: 0/20/0/0 12: 0/20/0/0 13: 0/20/0/0 14: 0/20/0/0
     15: 0/20/0/0 16: 0/20/0/0
```

Look at the queue usage counters and note the system queue counter used to queue unicast RIP updates is going up, since RIP no longer uses the interface's broadcast queue. Below are the actual contents of the VC's custom-queue. It shows the HTTP flow between R5 and R2.

```
Rack1R5#show traffic-shape queue serial 0/0 dlci 502
Traffic queued in shaping queue on Serial0/0 dlci 502
  Queueing strategy: custom-queue list 1
  Queueing Stats: 1/9/15/9 queue #/size/max total/drops)
Packet 1, linktype: ip, length: 84, flags: 0x10000008
  source: 155.1.0.5, destination: 155.1.0.2, id: 0x64CB, ttl: 255,
  TOS: 0 prot: 6, source port 80, destination port 23603
    data: 0x0050 0x5C33 0x1FEC 0xBCE0 0xA7F4 0x5B03 0x5010
          0x1020 0xA5B5 0x0000 0x417B 0x2E1D 0x5158 0xCFE2

Packet 2, linktype: ip, length: 84, flags: 0x10000008
  source: 155.1.0.5, destination: 155.1.0.2, id: 0x64CC, ttl: 255,
  TOS: 0 prot: 6, source port 80, destination port 23603
    data: 0x0050 0x5C33 0x1FEC 0xBD08 0xA7F4 0x5B03 0x5010
          0x1020 0x9CE6 0x0000 0xD8DD 0x5019 0x5B01 0xD548
<snip>
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.25 Legacy FRTS with Per-VC Fragmentation

- Remove any artificial IP MTU settings from the Frame Relay interfaces of R3 and R5 along with any per-VC queueing strategies.
- The connection between R3 and R5 is used to send data and voice traffic.
- Configure Frame Relay fragmentation so that voice packets are never delayed for more than 10ms when serialized.
- Assume the speed of R3's connection is 384Kbps
- Ensure R2 is not affected by this configuration.

Configuration

```
R3:
map-class frame-relay TO_R5
 frame-relay fragment 480
!
interface Serial 1/0
 no ip mtu

R5:
map-class frame-relay TO_R3
 frame-relay fragment 480
!
interface Serial 0/0
 no ip mtu
```

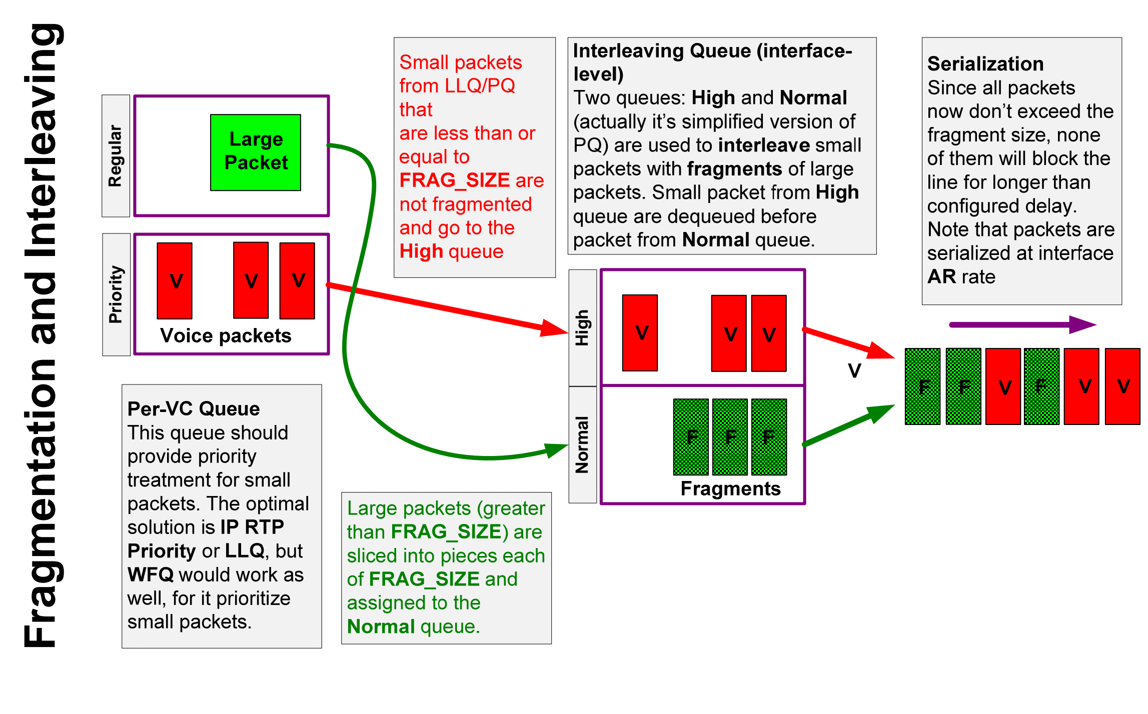
Verification

Note

Fragmentation is a technique used to avoid a "line blocking" effect due to the transmit ring of slow links. "Slow" links are defined as any media that takes more than 10ms to serialize an average data packet of 1500 bytes. On these links small voice packets, such as packets with a size around 60 bytes, may have to wait a considerable amount of time if the system serializes them after a large data packet. This "line blocking" effect can occur even if prioritization is configured in the software queue to send VoIP as fast as possible, and can effectively cancel out any QoS calculations.

For example assume that a priority VoIP packet of 60 bytes is trying to be sent out a 512Kbps Serial link. As the VoIP packet arrives in the output queue, it is moved from the tail of the queue to the head of the queue due to priority QoS configuration. At this exact instant, a 1500 byte data packet is admitted to the transmit ring, which has a queue depth of exactly 1 packet. Based on the data packet size and the 512Kbps access rate of the link, the VoIP packet now has to wait almost 24ms (1500 bytes * 8 bits/byte * 1000 ms/sec * 1 sec/512000 bits) to be admitted to the TxR, even though it is prioritized. If the TxR depth is more than one packet, the delay in the output queue can get even worse for the VoIP packet. Fragmentation techniques are designed to avoid this effect by breaking these larger packets into smaller pieces that require no more than 10ms to serialize.

Fragmentation on its own however, solves only part of the problem. We still need to ensure that fragmentation occurs for large packets, that VoIP is prioritized, and that the VoIP packets can be inserted (interleaved) between the now fragmented data packets. Frame Relay End-to-End Fragmentation (FRF.12) coupled with priority queueing can be used to solve this problem, as illustrated below.



FRF.12 utilizes two levels of queueing, the per-VC queue and the interface level queue. When you turn on fragmentation for a map-class, WFQ becomes the VC queuing strategy automatically. As we will see below, the WFQ strategy alone is *not* enough to enable interleaving. Next, the system converts the interface level FIFO queue into a special dual-FIFO queue. This dual-FIFO is actually the legacy PQ with two active queues – High and Normal. The system uses this special priority queue for interleaving as follows.

First, the shaper delays packets when congestion above CIR occurs. If there is a priority queue enabled on the VC (IP RTP Priority or LLQ), packets from this queue are serviced first. If the packet size is less than or equal to the configured fragment size, the system places the packet into the *high* portion of the dual-FIFO. In other words packets only get into the dual-FIFO interleaving priority queue if they were already prioritized and they don't need to be fragmented.

Next, large data packets with a size greater than the fragment size are sliced into smaller pieces, each of size *FRAG_SIZE* or smaller. These fragments are then queued to the *normal* interface queue of the dual-FIFO. Fragments are never queued to the *high* queue.

Once packets are at the interface level queue, the high queue is emptied completely before the normal queue is serviced. The result of this is that a large group of fragments can never block a small voice packet on the line. Note that the system always fragments large packets greater than *FRAG_SIZE*, even control plane packets such as routing updates, irrespective of whether there is congestion on the PVC or not. For this reason control plane problems can occur if the fragment size on both ends of the link is not the same. Also from a design prospective it does not make sense to enable fragmentation just on a single PVC, as other PVCs containing large packets may block the line. If you enable fragmentation, ensure it's enabled for all PVCs on the interface.

The computation of the fragment size must be based on the interface's physical access rate, not the PVC's CIR value, and should result in the fragment taking 10ms at most to be serialized. If the B_c value in your FRTS calculations already results in a T_c of 10ms (e.g. B_c = CIR * 10/1000), then the *FRAG_SIZE* will simply be the B_c value in bytes, as seen in the *Byte Increment* field in the **show traffic-shape** output. Fragmentation is usually unneeded on links with a speed of T1 and higher, as they already serialize an average 1500 byte packet in less than 10ms.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

FRF.12 can be verified as follows.

```
Rack1R3#clear counters
Rack1R3#clear frame-relay counter

Rack1R3#show frame-relay pvc 305

PVC Statistics for interface Serial1/0 (Frame Relay DTE)

DLCI = 305, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial1/0

<snip>
  fragment type end-to-end fragment size 480
<snip>
  frags 56         bytes 19028     frags delayed 56        bytes delayed 19028
  shaping active
  traffic shaping drops 0
  Queueing strategy: weighted fair
  Current fair queue configuration:
   Discard     Dynamic      Reserved
   threshold   queue count  queue count
    64          16           0
  Output queue size 3/max total 600/drops 0

Rack1R3#show frame-relay fragment
interface                dlci frag-type  size in-frag    out-frag   dropped-frag
Se1/0                    305  end-to-end 480  12         82         0
```

Check the interface queue of the Frame Relay interface of R3 and note that it is now dual-FIFO.

```
Rack1R3#show interfaces serial 1/0
Serial1/0 is up, line protocol is up
  Hardware is CD2430 in sync mode
  Internet address is 155.1.0.3/24
  MTU 1500 bytes, BW 128 Kbit, DLY 20000 usec,
     reliability 255/255, txload 43/255, rxload 19/255
  Encapsulation FRAME-RELAY, loopback not set
  Keepalive set (10 sec)
  LMI enq sent  185, LMI stat recvd 185, LMI upd recvd 0, DTE LMI up
  LMI enq recvd 0, LMI stat sent  0, LMI upd sent  0
  LMI DLCI 1023  LMI type is CISCO  frame relay DTE
  FR SVC disabled, LAPF state down
  Broadcast queue 0/64, broadcasts sent/dropped 176/0, interface
broadcasts 176
  Last input 00:00:00, output 00:00:00, output hang never
  Last clearing of "show interface" counters 00:30:42
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: dual fifo
  Output queue: high size/max/dropped 0/256/0
  Output queue: 0/128 (size/max)
<snip>
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
Rack1R3#show queueing interface serial 1/0
Interface Serial1/0 queueing strategy: priority

Output queue utilization (queue/count)
      high/192 medium/0 normal/32238 low/0
```

Check the fragmentation statistics on R5 and confirm that packets between R2 and R5 are not fragmented.

```
Rack1R5#clear counters
Rack1R5#clear frame-relay counter

Rack1R5#show frame-relay fragment
interface                dlci frag-type  size in-frag    out-frag   dropped-frag
Se0/0                    503  end-to-end 480  0          320        0

Rack1R5#show frame-relay pvc 503

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 503, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

<snip>
  frags 83         bytes 10612     frags delayed 24        bytes delayed 4476
  shaping inactive
  traffic shaping drops 0
  Queueing strategy: weighted fair
  Current fair queue configuration:
   Discard     Dynamic      Reserved
   threshold   queue count  queue count
    64          16           0
  Output queue size 0/max total 600/drops 0

Rack1R5#show frame-relay pvc 502

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 502, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

<snip>
  pvc create time 09:33:23, last time pvc status changed 09:33:10
  cir 128000    bc 1280      be 0         byte limit 160    interval 10
  mincir 64000     byte increment 160   Adaptive Shaping none
  pkts 27        bytes 8752      pkts delayed 13        bytes delayed 1248
  shaping active
  traffic shaping drops 0
  Queueing strategy: custom-list 1

List   Queue  Args
1      0      protocol ip          udp port rip

List   Queue  Args
1      1      protocol ip          tcp port www
1      2      protocol ip          tcp port telnet
1      3      protocol ip          list 100
1      0      limit 15
1      1      byte-count 600 limit 15
1      2      byte-count 200 limit 15
1      3      byte-count 100 limit 15
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
  Output queues: (queue #: size/max/drops/dequeued)
     0: 0/15/0/22 1: 0/15/0/0 2: 0/15/0/0 3: 0/15/0/0 4: 0/20/0/0
     5: 0/20/0/0 6: 0/20/0/0 7: 0/20/0/0 8: 0/20/0/0 9: 0/20/0/0
     10: 0/20/0/0 11: 0/20/0/0 12: 0/20/0/0 13: 0/20/0/0 14: 0/20/0/0
     15: 0/20/0/0 16: 0/20/0/0
```

Now let’s try to see if the interface-level dual-FIFO works. Configure R5 to download an IOS image from R3, and set R3’s PVC speed to 64Kbps to match the interface rate. Additionally, source “small” ping packets from SW1 across R3 to R5.

```
R3:
map-class TO_R5
 frame-relay cir 64000
!
ip http server
ip http path flash:

Rack1R5#copy http://admin:cisco@155.1.0.3/c2600-adventerprisek9-mz.124-
10.bin null:
Loading http://***********@155.1.0.3/c2600-adventerprisek9-mz.124-
10.bin !!!!

Rack1SW1#ping 155.1.0.5 repeat 10000 size 80

Type escape sequence to abort.
Sending 1000, 100-byte ICMP Echos to 155.1.0.2, timeout is 2 seconds:
!!
```

Check the interleaving and shaping queue statistics. Note that two flows exist, corresponding to ping packets (80 bytes) and file transfers (large packets, sourced from port 80). Note the “High” queue is not used, and “Interleaves” counters are zero. That means **interleaving is not working**, since we didn’t enable prioritization yet, and have just basic WFQ enabled per-queue.

```
Rack1R3#show queueing interface serial 1/0
Interface Serial1/0 queueing strategy: priority

Output queue utilization (queue/count)
      high/0 medium/0 normal/138 low/0
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
Rack1R3#show traffic-shape queue serial 1/0 dlci 305
Traffic queued in shaping queue on Serial1/0 dlci 305
  Queueing strategy: weighted fair
  Queueing Stats: 3/600/64/0 (size/max total/threshold/drops)
     Conversations  2/5/16 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 32 kilobits/sec

  (depth/weight/total drops/no-buffer drops/interleaves) 1/32384/0/0/0
  Conversation 10, linktype: ip, length: 84
  source: 155.1.37.7, destination: 155.1.0.5, id: 0x5499, ttl: 254, prot: 1

  (depth/weight/total drops/no-buffer drops/interleaves) 2/32384/0/0/0
  Conversation 6, linktype: ip, length: 738
  source: 155.1.0.3, destination: 155.1.0.5, id: 0x264F, ttl: 255,
  TOS: 0 prot: 6, source port 80, destination port 25437
```

To confirm that interleaving is not working, enable debugging for the interface level priority queue (dual-FIFO). Note the packets of size 84 – those are the ping packets of size 80 plus the Frame-Relay 4 byte header. The other packets – 486 bytes and 86 bytes are fragments of the 1500 byte data packets. Each 1500-byte packet is split into 3x480 and 1x60 byte packets at L3, and then a 6 byte header is added. The additional 2 bytes are the FRF.12 header information for fragmented packets. Note that non-fragmented packets do not carry this additional 2 bytes of FRF.12 overhead.

```
Rack1R3#debug priority
Priority output queueing debugging is on
PQ: Serial1/0 output (Pk size/Q 84/2)
PQ: Serial1/0 output (Pk size/Q 486/2)
PQ: Serial1/0 output (Pk size/Q 486/2)
PQ: Serial1/0 output (Pk size/Q 486/2)
PQ: Serial1/0 output (Pk size/Q 486/2)
PQ: Serial1/0 output (Pk size/Q 486/2)
PQ: Serial1/0 output (Pk size/Q 486/2)
PQ: Serial1/0 output (Pk size/Q 486/2)
PQ: Serial1/0 output (Pk size/Q 486/2)
PQ: Serial1/0 output (Pk size/Q 68/2)
PQ: Serial1/0 output (Pk size/Q 84/2)
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.26 Legacy FRTS with Per-VC IP RTP Priority

- Modify R3 and R5's FRTS configuration to prioritize RTP VoIP traffic on the PVC between them.
- Allocate 128Kbps of bandwidth to voice traffic in the port range of 16384 – 32767.

Configuration

```
R3:
map-class frame-relay TO_R5
 frame-relay ip rtp priority 16384 16383 128

R5:
map-class frame-relay TO_R3
 frame-relay ip rtp priority 16384 16383 128
```

Verification

Note

Configuration of Frame Relay IP RTP Priority is the same as classic IP RTP Priority, with the exception that it applies per-VC, activating a priority flow inside the shaper's queue. The <*Bandwidth*> value in the syntax defines the policer rate applied to priority packets with the burst size of 1 second. This ensures that non-priority packets are not starved of bandwidth like they are with the legacy priority-list.

Once the per-VC Priority Queue is enabled, interleaving should now be active due to the previous FRF.12 fragmentation configuration. Note that Frame-Relay IP RTP priority only activates if FRF.12 fragmentation is active on the same PVC. This configuration can be verified as follows.

```
Rack1R3#show frame-relay pvc 305

PVC Statistics for interface Serial1/0 (Frame Relay DTE)

DLCI = 305, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial1/0

<snip>
  fragment type end-to-end fragment size 480
  cir 384000    bc   3840      be 0         limit 480    interval 10
  mincir 256000    byte increment 480   BECN response yes IF_CONG yes
  frags 103       bytes 36668     frags delayed 0         bytes delayed 0
  shaping inactive
  traffic shaping drops 0
  ip rtp priority parameters 16384 32767 128000
  Queueing strategy: weighted fair
  Current fair queue configuration:
<snip>
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Now disable FRF.12 and see how IP RTP Priority goes away at the same time. This is due to the fact that the interface queue does not provide priority service without FRF.12 interleaving.

```
Rack1R3#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Rack1R3(config)#map-class frame-relay TO_R5
Rack1R3(config-map-class)#no frame-relay fragment 480
Rack1R3(config-map-class)#^Z

Rack1R3#show frame-relay pvc 305

PVC Statistics for interface Serial1/0 (Frame Relay DTE)

DLCI = 305, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial1/0

  input pkts 243           output pkts 126          in bytes 75848
  out bytes 42656          dropped pkts 0           in pkts dropped 0
  out pkts dropped 0                out bytes dropped 0
  in FECN pkts 0           in BECN pkts 0           out FECN pkts 0
  out BECN pkts 0          in DE pkts 0             out DE pkts 0
  out bcast pkts 124       out bcast bytes 42264
  5 minute input rate 1000 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
  Shaping adapts to BECN
  pvc create time 00:19:07, last time pvc status changed 00:18:35
  cir 384000    bc 3840      be 0         byte limit 480    interval 10
  mincir 256000    byte increment 480   Adaptive Shaping BECN and IF_CONG
  pkts 112       bytes 39872     pkts delayed 0         bytes delayed 0
  shaping inactive
  traffic shaping drops 0
  Queueing strategy: weighted fair
  Current fair queue configuration:
   Discard     Dynamic      Reserved
   threshold   queue count  queue count
    64          16           0
  Output queue size 0/max total 600/drops 0
```

Re-enable FRF.12, but now set the fragment size to 80 bytes for testing. At the same time set the CIR to 64Kbps, and the RTP Priority bandwidth to 32Kbps, and enable the HTTP server for verification purposes.

```
R3:
map-class frame-relay TO_R5
 frame-relay mincir 64000
 frame-relay cir 64000
 frame-relay ip rtp priority 16384 16383 32
 frame-relay fragment 80
!
ip http server
ip http path flash:
```

Configure R6 to start an SLA jitter probe towards R5 using G.729 sized packets and a destination port of 16384. Shutdown the VLAN 146 interface on R6 to ensure that packets are taking the path to R5 via R3.

```
R5:
rtr responder

R6:
no ip domain lookup
ip sla monitor 1
 type jitter dest-ipaddr 155.1.0.5 dest-port 16384 codec g729a
 timeout 1000
 frequency 1
ip sla monitor schedule 1 life forever start-time now
!
interface FastEthernet 0/0.146
 shutdown
```

Start transferring a large file (IOS image) from R3 to R5, and ping R5 from SW1.

```
Rack1R5#copy http://admin:cisco@155.1.0.3/c2600-adventerprisek9-mz.124-
10.bin null:
Loading http://***********@155.1.0.3/c2600-adventerprisek9-mz.124-
10.bin !!!!

Rack1SW1#ping 155.1.0.5 repeat 10000 size 80

Type escape sequence to abort.
Sending 1000, 100-byte ICMP Echos to 155.1.0.2, timeout is 2 seconds:
!!
```

Check the interleaving and shaping queue statistics. Note that the HTTP flow is now being interleaved, and the UDP flow has a WFQ weight of 0 for priority.

```
Rack1R3#show traffic-shape queue serial 1/0 dlci 305
Traffic queued in shaping queue on Serial1/0 dlci 305
  Queueing strategy: weighted fair
  Queueing Stats: 3/600/64/0 (size/max total/threshold/drops)
     Conversations  2/3/16 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 32 kilobits/sec

  (depth/weight/total drops/no-buffer drops/interleaves) 2/0/0/0/0
  Conversation 24, linktype: ip, length: 64
  source: 155.1.67.6, destination: 155.1.0.5, id: 0x0003, ttl: 253,
  TOS: 0 prot: 17, source port 56329, destination port 16384

  (depth/weight/total drops/no-buffer drops/interleaves) 1/32384/0/0/1197
  Conversation 5, linktype: ip, length: 1504
  source: 155.1.0.3, destination: 155.1.0.5, id: 0x3720, ttl: 255,
  TOS: 0 prot: 6, source port 80, destination port 44243
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
Rack1R3#show frame-relay pvc 305

PVC Statistics for interface Serial1/0 (Frame Relay DTE)

DLCI = 305, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial1/0

<snip>
  5 minute input rate 8000 bits/sec, 18 packets/sec
  5 minute output rate 58000 bits/sec, 15 packets/sec
  Shaping adapts to BECN
  pvc create time 00:52:45, last time pvc status changed 00:52:12
  fragment type end-to-end fragment size 80
  cir 64000     bc   3840      be 0         limit 480    interval 60
  mincir 64000     byte increment 480   BECN response yes IF_CONG yes
  frags 61037     bytes 5362484   frags delayed 60736     bytes delayed 5264932
  shaping active
  traffic shaping drops 0
  ip rtp priority parameters 16384 32767 32000
  Queueing strategy: weighted fair
  Current fair queue configuration:
   Discard     Dynamic      Reserved
   threshold   queue count  queue count
    64          16           0
  Output queue size 0/max total 600/drops 0

Rack1R3#show queueing interface serial 1/0
Interface Serial1/0 queueing strategy: priority

Output queue utilization (queue/count)
      high/1981 medium/0 normal/61972 low/0
```

Now that priority queueing is enabled on the VC, the high queue of the dual-FIFO is being used.

Check the SLA statistics on R6. Note the high RTT, which is inevitable with a link speed of 64Kbps, but that it is lower than the ICMP flow.

```
Rack1R6#show ip sla monitor statistics 1
Round trip time (RTT)   Index 1
      Latest RTT: 435 ms
Latest operation start time: 22:46:39.181 UTC Fri Aug 8 2008
Latest operation return code: OK
RTT Values
      Number Of RTT: 1000
      RTT Min/Avg/Max: 284/435/738 ms
Latency one-way time milliseconds
      Number of one-way Samples: 23
      Source to Destination one way Min/Avg/Max: 602/655/711 ms
      Destination to Source one way Min/Avg/Max: 4/25/52 ms
Jitter time milliseconds
      Number of Jitter Samples: 999
      Source to Destination Jitter Min/Avg/Max: 1/14/89 ms
      Destination to Source Jitter Min/Avg/Max: 1/3/253 ms
Packet Loss Values
      Loss Source to Destination: 0       Loss Destination to Source: 0
      Out Of Sequence: 0      Tail Drop: 0      Packet Late Arrival: 0
Voice Score Values
      Calculated Planning Impairment Factor (ICPIF): 21
MOS score: 3.68
Number of successes: 25
Number of failures: 0
Operation time to live: Forever
```

Note that ICMP packets of the same size have a larger RTT, even though the system schedules packets by using per-VC WFQ.

```
Rack1R6#ping 155.1.0.5 repeat 50 size 60

Type escape sequence to abort.
Sending 50, 60-byte ICMP Echos to 155.1.0.5, timeout is 2 seconds:
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Success rate is 100 percent (50/50), round-trip min/avg/max = 377/711/1017 ms
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.27 Frame-Relay RTP/TCP Header Compression

- Configure R3 and R5 to compress RTP and TCP headers on the PVC between them.
- Compression should not be enabled on any other PVCs.

Configuration

```
R3:
interface Serial 1/0
 frame-relay map ip 155.1.0.5 305 broadcast compress

R5:
interface Serial 0/0
 frame-relay ip rtp header-compression
 frame-relay ip tcp header-compression
 frame-relay map ip 155.1.0.1 501 broadcast nocompress
 frame-relay map ip 155.1.0.2 502 broadcast nocompress
 frame-relay map ip 155.1.0.4 504 broadcast nocompress
 frame-relay map ip 155.1.0.3 503 broadcast
```

Verification

Note

Frame-Relay IP RTP/TCP header compression is the same feature found on regular serial interfaces, however it allows applying compression on a per-VC basis. In the above example the feature is enabled two ways, on a per-VC basis with the **`frame-relay map`** statement, and on a per interface basis with the **`frame-relay ip rtp header-compression`**. In the latter case the other PVC (501, 502, and 504) have compression *disabled*, by adding the **`nocompress`** keyword to the end of their **`frame-relay map`**.

This feature can be verified as follows.

```
Rack1R3#show frame-relay pvc 305

Rack1R5#show  ip tcp header-compression
TCP/IP header compression statistics:
 DLCI 503     Link/Destination info: ip 155.1.0.3
  Interface Serial0/0 DLCI 503 (compression on, VJ)
    Rcvd:    0 total, 0 compressed, 0 errors, 0 status msgs
             0 dropped, 0 buffer copies, 0 buffer failures
    Sent:    0 total, 0 compressed, 0 status msgs, 0 not predicted
             0 bytes saved, 0 bytes sent
    Connect: 256 rx slots, 256 tx slots,
             0 misses, 0 collisions, 0 negative cache hits, 256 free contexts

Rack1R5#show ip rtp header-compression
RTP/UDP/IP header compression statistics:
 DLCI 503     Link/Destination info: ip 155.1.0.3
  Interface Serial0/0 DLCI 503 (compression on, Cisco, RTP)
    Rcvd:    0 total, 0 compressed, 0 errors, 0 status msgs
             0 dropped, 0 buffer copies, 0 buffer failures
    Sent:    0 total, 0 compressed, 0 status msgs, 0 not predicted
             0 bytes saved, 0 bytes sent
    Connect: 256 rx slots, 256 tx slots,
             0 misses, 0 collisions, 0 negative cache hits, 256 free contexts

Rack1R5#show frame-relay ip tcp header-compression
TCP/IP header compression statistics:
 DLCI 305     Link/Destination info: ip 155.1.0.5
  Interface Serial0/0 DLCI 305 (compression on, VJ)
    Rcvd:    0 total, 0 compressed, 0 errors, 0 status msgs
             0 dropped, 0 buffer copies, 0 buffer failures
    Sent:    0 total, 0 compressed, 0 status msgs, 0 not predicted
             0 bytes saved, 0 bytes sent
    Connect: 256 rx slots, 256 tx slots,
             0 misses, 0 collisions, 0 negative cache hits, 256 free contexts

Rack1R3#show frame-relay ip rtp header-compression
 DLCI 305     Link/Destination info: ip 155.1.0.5
  Interface Serial1/0 DLCI 305 (compression on, Cisco, RTP)
    Rcvd:    0 total, 0 compressed, 0 errors, 0 status msgs
             0 dropped, 0 buffer copies, 0 buffer failures
    Sent:    0 total, 0 compressed, 0 status msgs, 0 not predicted
             0 bytes saved, 0 bytes sent
    Connect: 256 rx slots, 256 tx slots,
             0 misses, 0 collisions, 0 negative cache hits, 256 free context
```

10.28 Frame-Relay Broadcast Queue

- Modify R5's Frame Relay broadcast queue per the following requirements:
  - Limit the broadcast queue rate a maximum of 128Kbps
  - Limit the queue depth to 4 packets per spoke
  - Limit the total number of broadcast packets to 10 per second

Configuration

```
R5:
interface Serial 0/0
 frame-relay broadcast-queue 16 16000 10
```

Verification

> **Note**
>
> The Frame Relay broadcast queue is a special interface-level priority queue used to store and send pseudo-broadcast packets. Within the context of Frame Relay interfaces, a pseudo-broadcast is a layer 3 broadcast or multicast packet that must be replicated to many PVCs at the same time at layer 2. Since the broadcast queue include multicast packets, routing updates are queued here by default, which is why it is given preferential treatment automatically.
>
> The **frame-relay broadcast-queue** command can be used to rate limit the depth, bit per second, or packet per second rate of broadcasts sent out the link, and can be verified as follows.

```
Rack1R5#show interfaces serial 0/0
Serial0/0 is up, line protocol is up
  Hardware is PowerQUICC Serial
  Internet address is 155.1.0.5/24
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation FRAME-RELAY, loopback not set
  Keepalive set (10 sec)
  LMI enq sent  2848, LMI stat recvd 2849, LMI upd recvd 0, DTE LMI up
  LMI enq recvd 0, LMI stat sent  0, LMI upd sent  0
  LMI DLCI 1023  LMI type is CISCO  frame relay DTE
  FR SVC disabled, LAPF state down
  Broadcast queue 2/16, broadcasts sent/dropped 20438/0, interface broadcasts
6102
<snip>
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.29 Frame-Relay DE Marking

- Configure R3 to mark all IP packets larger than 60 bytes as Discard Eligible as they are sent out to the Frame Relay cloud.

Configuration

```
R3:
frame-relay de-list 1 protocol ip gt 64
!
interface Serial 1/0
 frame-relay de-group 1 305
```

Verification

Note

The Discard Eligible (DE) bit is used in Frame Relay networks to determine which packets should be dropped first in the case of congestion. Like any marking, the DE bit has no meaning on its own without a policy configured in the service provider's network to actually implement the dropping. Usually the policy involves configuring a queue depth threshold where DE-marked packets are discarded, while regular packets are preserved.

Normally the DE bit is set on the PE device during input metering, but as seen in the above example it is possible to set the bit on the CE device outbound. This legacy configuration is rarely used however, as it only supports process switched packets, and does not support other switching paths such as fast switching or CEF switching, like the MQC-based configuration does. Remember that with legacy QoS configuration packet sizes include the full layer 2 packet length, so 64 bytes in the above configuration means any IP packet with a size of 60 bytes, plus 4 bytes of "Cisco" Frame Relay encapsulation, or greater.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

To verify this configuration check the DE counter as follows. In this particular case the counter is increasing as local routing packets are larger than 60 bytes and are process switched.

```
Rack1R3#show frame-relay pvc 305 | inc DLCI|PVC|DE
PVC Statistics for interface Serial1/0 (Frame Relay DTE)
DLCI = 305, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE =
Serial1/0
  out BECN pkts 0          in DE pkts 0             out DE pkts 90

Rack1R5#show frame-relay pvc 503 | inc DLCI|PVC|DE
PVC Statistics for interface Serial0/0 (Frame Relay DTE)
DLCI = 503, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE =
Serial0/0
  out BECN pkts 0          in DE pkts 95            out DE pkts 0
```

CEF switched transit packets are not be marked with a DE bit.

```
Rack1SW1#ping 155.1.0.5 size 1000 repeat 10000 timeout 0

Type escape sequence to abort.
Sending 10000, 1000-byte ICMP Echos to 155.1.0.5, timeout is 0 seconds:
......................................................................
<snip>

Rack1R3#show frame-relay pvc 305 | inc DLCI|PVC|DE
PVC Statistics for interface Serial1/0 (Frame Relay DTE)
DLCI = 305, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE =
Serial1/0
  out BECN pkts 0          in DE pkts 0             out DE pkts 93
```

With CEF disabled the ICMP flow is marked as DE.

```
Rack1R3(config)#interface Serial1/0
Rack1R3(config-if)#no ip route-cache
Rack1R3(config-if)#end

Rack1R3#show frame-relay pvc 305 | inc DLCI|PVC|DE
PVC Statistics for interface Serial1/0 (Frame Relay DTE)
DLCI = 305, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE =
Serial1/0
  out BECN pkts 0          in DE pkts 0             out DE pkts 861

Rack1R3#show frame-relay pvc 305 | inc DLCI|PVC|DE
PVC Statistics for interface Serial1/0 (Frame Relay DTE)
DLCI = 305, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE =
Serial1/0
  out BECN pkts 0          in DE pkts 0             out DE pkts 1411
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.30 Legacy FRTS PVC Interface Priority Queue

- Configure PIPQ on R5 so that frames going out to R3 are always sent first, followed by frames going to R2.
- All other PVCs assigned to R5's interface should be serviced after R3 and R2.

Configuration

```
R5:
map-class frame-relay TO_R2
 frame-relay interface-queue priority normal
!
map-class frame-relay TO_R3
 frame-relay interface-queue priority medium
!
map-class frame-relay DEFAULT
 frame-relay interface-queue priority low
!
interface Serial 0/0
 frame-relay interface-queue priority
 frame-relay class DEFAULT
```

Verification

Note

PVC Interface Priority Queueing (PIPQ) enables the legacy Priority Queue at the VC level itself. The interface queue defined in the map-class maps the PVC to the respective high, medium, normal, or low queue. Note that broadcast packets (e.g. routing updates) still go to the interface-level broadcast queue, while Layer 2 keepalives (i.e. LMI keepalives) use the high priority queue.

Enabling PIPQ with per-VC fragmentation and IP RTP Priority breaks interleaving, as the system replaces the dual-FIFO interface queue with a fully functional Priority Queue. The system no longer splits the PVC traffic between interleaving queues, but rather all packets (small voice and large packet fragments) use the same interface-level queue. Therefore if you plan to use PIPQ for VoIP deployments ensure that voice traffic is the only traffic using the priority PVC, or that the connection speed is high enough to eliminate the problem of serialization delay.

PIPQ can be verified as follows.

```
Rack1R5#show interfaces serial 0/0
Serial0/0 is up, line protocol is up
  Hardware is PowerQUICC Serial
  Internet address is 155.1.0.5/24
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
     reliability 255/255, txload 4/255, rxload 4/255
  Encapsulation FRAME-RELAY, loopback not set
  Keepalive set (10 sec)
  LMI enq sent  211, LMI stat recvd 206, LMI upd recvd 0, DTE LMI up
  LMI enq recvd 0, LMI stat sent  0, LMI upd sent  0
  LMI DLCI 1023  LMI type is CISCO  frame relay DTE
  FR SVC disabled, LAPF state down
  Broadcast queue 0/16, broadcasts sent/dropped 1776/0, interface broadcasts
456
  Last input 00:00:00, output 00:00:00, output hang never
  Last clearing of "show interface" counters 00:35:04
  Input queue: 4/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: DLCI priority
  Output queue (queue priority: size/max/drops):
     high: 0/20/0, medium: 0/40/0, normal: 0/60/0, low: 0/80/0
  5 minute input rate 26000 bits/sec, 50 packets/sec
  5 minute output rate 27000 bits/sec, 52 packets/sec
     93649 packets input, 6201446 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     1 input errors, 0 CRC, 1 frame, 0 overrun, 0 ignored, 0 abort
     94712 packets output, 6481205 bytes, 0 underruns
     0 output errors, 0 collisions, 1 interface resets
     0 output buffer failures, 0 output buffers swapped out
     2 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up

Rack1R5#show frame-relay pvc 503 | include priority
  priority medium

Rack1R5#show frame-relay pvc 501 | include priority
  priority low

Rack1R5#show queueing interface serial 0/0
Interface Serial0/0 queueing strategy: priority

Output queue utilization (queue/count)
      high/228 medium/102656 normal/486 low/974
```

Note the PVC priority levels. By default the priority level is normal. Also note the high queue counter getting incremented, as this queue is used for LMI keepalive messages.

10.31 Frame-Relay Priority to DLCI Mapping

- Configure back-to-back Frame Relay on the Serial link between R4 and R5 using DLCIs 100 and 200.
- Configure a static mapping to R4 and R5's IP addresses using DLCI 100 on both sides.
- Create a legacy priority-list on both R4 and R5 that maps RIP updates and packets less than 65 bytes to the high queue.
- Map packets from the high queue to the DLCI 100 on both sides.
- Map all other queues to DLCI 200 on both sides.

Configuration

```
R4:
priority-list 1 protocol ip high lt 64
priority-list 1 protocol ip high udp 520
!
interface Serial0/1
 ip address 155.1.45.4 255.255.255.0
 encapsulation frame-relay
 no keepalive
 priority-group 1
 frame-relay priority-dlci-group 1 100 200 200 200
 frame-relay map ip 155.1.45.5 100 broadcast
 frame-relay interface-dlci 100
 frame-relay interface-dlci 200

R5:
priority-list 1 protocol ip high lt 64
priority-list 1 protocol ip high udp 520
!
interface Serial0/1
 ip address 155.1.45.5 255.255.255.0
 encapsulation frame-relay
 no keepalive
 priority-group 1
 frame-relay priority-dlci-group 1 100 200 200 200
 frame-relay map ip 155.1.45.4 100 broadcast
 frame-relay interface-dlci 100
 frame-relay interface-dlci 200
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Verification

Note

Although not widely used, the DLCI-based Frame Relay priority feature is used to forward traffic out different VCs based on the traffic's priority level. The idea of priority to DLCI mapping is to use different "parallel" DLCIs to carry different types of traffic. This strategy however, requires that the service provider cloud assigns different SLAs to the different PVCs.

With this configuration only one **`frame-relay map`** statement is needed to a PVC in the priority group. The priority-list entries and queue to DLCI mappings actually determine which PVC is used when encapsulation occurs.

The format of the command syntax is **`frame-relay priority-dlci-group`** ***<N> <High> <Med> <Norm> <Low>***, where *N* is the list number, and *High, Med, Norm,* and *Low* are the queue levels that the circuits are mapped to. In this particular case the result is that traffic matched by list number 1 in the high queue goes to VC 100, while all other traffic goes to VC 200.

To verify this clear the counters and generate a traffic flow.

```
Rack1R5#clear counters serial 0/1
Rack1R5#clear frame-relay counters

Rack1R5#show queueing interface serial 0/1
Interface Serial0/1 queueing strategy: priority

Output queue utilization (queue/count)
      high/94 medium/0 normal/12 low/0

Rack1R5#show frame-relay pvc 200

PVC Statistics for interface Serial0/1 (Frame Relay DTE)

DLCI = 200, DLCI USAGE = LOCAL, PVC STATUS = STATIC, INTERFACE = Serial0/1

  input pkts 10            output pkts 12           in bytes 2040
  out bytes 2692           dropped pkts 0           in pkts dropped 0
  out pkts dropped 0                out bytes dropped 0
  in FECN pkts 0           in BECN pkts 0           out FECN pkts 0
  out BECN pkts 0          in DE pkts 0             out DE pkts 0
  out bcast pkts 2         out bcast bytes 652
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
  pvc create time 01:06:26, last time pvc status changed 01:06:26
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
Rack1R5#ping 155.1.45.4 size 200 repeat 10

Type escape sequence to abort.
Sending 10, 200-byte ICMP Echos to 155.1.45.4, timeout is 2 seconds:
!!!!!!!!!!
Success rate is 100 percent (10/10), round-trip min/avg/max = 52/55/57 ms

Rack1R5#show queueing interface serial 0/1
Interface Serial0/1 queueing strategy: priority

Output queue utilization (queue/count)
      high/98 medium/0 normal/22 low/0

Rack1R5#show frame-relay pvc 200

PVC Statistics for interface Serial0/1 (Frame Relay DTE)

DLCI = 200, DLCI USAGE = LOCAL, PVC STATUS = STATIC, INTERFACE =
Serial0/1

  input pkts 20            output pkts 22           in bytes 4080
<snip>
```

Note that packets assigned to the normal queue use DLCI 200 for encapsulation, even though we map the IP address of the other router to DLCI 100. Now generate traffic for the high priority queue.

```
Rack1R5#clear counters serial 0/1
Clear "show interface" counters on this interface [confirm]

Rack1R5#show queueing interface serial 0/1
Interface Serial0/1 queueing strategy: priority

Output queue utilization (queue/count)
      high/0 medium/0 normal/0 low/0

Rack1R5#ping 155.1.45.4 repeat 100 size 40

Type escape sequence to abort.
Sending 100, 40-byte ICMP Echos to 155.1.45.4, timeout is 2 seconds:
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
<snip>

Rack1R5#show queueing interface serial 0/1
Interface Serial0/1 queueing strategy: priority

Output queue utilization (queue/count)
      high/102 medium/0 normal/0 low/0
```

Note the number of packets classified to the high queue. The additional matches are due to the RIP routing updates using the same queue.

Copyright © 2009 Internetwork Expert

10.32 Frame-Relay Traffic Policing & Congestion Mgmt

- Enable Frame Relay switching on R3.
- Configure R3's links to R1 and R2 as Frame Relay DCE interfaces, and clock the links at 128Kbps.
- Configure R3 to terminate DLCI 133 to R1 using the IP address 155.X.13.3/24.
- Configure R3 to terminate DLCI 233 to R2 using the IP address 155.X.23.3/24.
- Configure R3 to switch DLCI 132 from R1 to R2 as DLCI 231.
- Configure two point-to-point subinterfaces on R1's link to R3.
- Configure R1 to use the IP address 155.X.13.1/24 and DLCI 133 for the first subinterface, and IP address 155.X.12.1/24 and DLCI 132 for the second subinterface.
- Configure two point-to-point subinterfaces on R2's link to R3.
- Configure R2 to use the IP address 155.X.23.2/24 and DLCI 233 for the first subinterface, and IP address 155.X.12.2/24 and DLCI 231 for the second subinterface.
- Ensure that R3 has IP connectivity to R1 and R2, and R1 and R2 have IP connectivity to each other over these circuits.
- Create two map-classes on R3 for the switched DLCIs with the following policies:
  - The class for R1 should be configured for an input CIR of 64Kbps, an input B_c and B_e of 8000 bits, and an input T_c of 125ms
  - The class for R2 should be configured for an input CIR zero, an input B_c of zero, an input B_e of 16000 bits, and an input T_c of 125ms
- Enable Frame Relay traffic-policing on R3's links to R1 and R2 and apply the above classes to the respective DLCIs.
- Configure generic traffic-shaping on R1 and R2 to average 128Kbps with a B_c of 16000 bits.
- R1 should reduce its sending rate to the guaranteed rate in the case that BECNs are received.
- Enable congestion management on R3's switched PVCs and set the thresholds for DE dropping to 5% and FECN/BECN marking to 3%.

Configuration

```
R1:
interface Serial0/1
 encapsulation frame-relay
!
interface Serial0/1.1 point-to-point
 ip address 155.1.12.1 255.255.255.0
 frame-relay interface-dlci 132
 traffic-shape rate 128000 16000 0
 traffic-shape adaptive 64000
!
interface Serial0/1.2 point-to-point
 ip address 155.1.13.1 255.255.255.0
 frame-relay interface-dlci 133

R2:
interface Serial0/1
 encapsulation frame-relay
!
interface Serial0/1.1 point-to-point
 ip address 155.1.12.2 255.255.255.0
 frame-relay interface-dlci 231
 traffic-shape rate 128000 16000 0
!
interface Serial0/1.2 point-to-point
 ip address 155.1.23.2 255.255.255.0
 frame-relay interface-dlci 233

R3:
frame-relay switching
!
! Note the “in” keyword only works for policing
! It’s possible to set “out” keyword to use different values
! for traffic shaping on the same VC
!
map-class frame-relay FROM_R1
 frame-relay cir in 64000
 frame-relay bc in 8000
 frame-relay be in 8000
!
! Zero CIR contract. Thus “cir in” is set to 0, as well
! as “bc in”. Note that now the “Tc” variable comes into play,
! as it is used to define the measurement interval.
!
map-class frame-relay FROM_R2
 frame-relay cir in 0
 frame-relay bc in 0
 frame-relay be in 16000
 frame-relay tc 125
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
!
! Apply map-classes to switched VCs
! and enable traffic policing
!
interface Serial1/2
 ip address 155.1.13.3 255.255.255.0
 encapsulation frame-relay
 clock rate 128000
 frame-relay interface-dlci 132 switched
  class FROM_R1
 frame-relay interface-dlci 133
 frame-relay intf-type dce
 frame-relay policing
!
! Congestion management thresholds
!
 frame-relay congestion-management
  threshold ecn 3
  threshold de 5
!
! Enable policing on the interface for switched DLCIs
! Apply map-classes to apply contract values
!
interface Serial1/3
 ip address 155.1.23.3 255.255.255.0
 encapsulation frame-relay
 clock rate 128000
 frame-relay interface-dlci 231 switched
   class FROM_R2
 frame-relay interface-dlci 233
 frame-relay intf-type dce
 frame-relay policing
!
! Congestion management thresholds
!
 frame-relay congestion-management
  threshold ecn 3
  threshold de 5
!
! Configure PVC switching
!
connect R1_TO_R2 Serial1/2 132 Serial1/3 231
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Verification

Note

This task summarizes many of the legacy Frame Relay QoS concepts covered so far, including output shaping, input metering, oversubscription, DE marking, and adaptive shaping with BECN/FECN. In this case R3 is the service provider, and sells traffic contracts in terms of the PIR (Peak Information Rate) and CIR (Committed Information Rate), and defines the metering interval T_c. R3 guarantees CIR to the customer at any time, but the customer may send up to the PIR without any guarantee. The measurement interval is an essential part of the traffic contract, as it defines the “averaging” window to meter the traffic rate at the service providers side. In order to conform to the traffic contract, the customers (R1 and R2) must perform output *shaping* of their packet stream in order to conform to the provider’s input *policing* used to meter the input burst rates.

Frame Relay Traffic Policing (FRTP) uses a *two-rate* policing process based on the token bucket model. The following parameters are used by the policer:

Bc=CIR/Tc
Be=(PIR-CIR)/Tc=EIR/Tc

Note that the definition of the B_e in this context is different than with shaping or with CAR. In FRTP B_e defines the EIR (Excess Information Rate) rate – the amount of exceeding burst size that could be accepted from the customer equipment. The B_e value in traffic shaping, on the other hand, defines the amount of excess data that could be sent if the previous shaping interval was underutilized. CAR uses B_e to simulate a random marking/dropping behavior. FRTP uses the classic policing procedure adapted to utilize Frame-Relay DE marking as follows.

If the sum of the incoming packet size plus the size of packets received during the last T_c are less than or equal to B_c, the packet *conforms* to CIR. If the incoming packet size plus the size of packets received during the last T_c is greater than B_c but less than B_c+B_e, the packet *exceeds*, but still under PIR. Packets above CIR but under PIR are marked with the DE bit automatically. If the incoming packet is already marked as DE, it automatically exceeds. If the incoming packet size plus the size of packets received during the last T_c is above B_c+B_e, the packet *violates* the contract, and the policer drops it.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

As discussed previously a correct policer behavior depends on the customer shaping traffic properly. The customer accomplishes this by using the CIR, PIR and T_c values matching the provider settings. Note that legacy FRTS as well as GTS have no notion of PIR value, and therefore the customer side is commonly configured with Customer CIR=Provider PIR, and Customer minCIR = Provider CIR. This ensures that in case of congestion signaled by the service provider cloud, the customer throttles down its Frame Relay sending rate to the guaranteed rate.

Note that FRTP is configured with the same syntax as FRTS, but the options "in" and "out" are used to define if policing or shaping is occurring (policing is in, shaping is out). This allows the traffic on the same switched PVC to be policed (admitted) ingress and shaped egress. This feature is useful for network-to-network (NNI) connections, where the PVC is switched between different provider clouds, and it requires ingress contract verification by using policing as well as egress contract enforcement by using shaping.

Cisco IOS implements congestion management on DCE interfaces for traffic crossing switched PVCs only. The system measures the interface queue depth, and if the queue depth crosses the DE threshold (expressed in percent), all DE packets leaving all switched PVCs on the interface are discarded, while conforming packets are kept until the queue is 100% full. If the queue depth crosses a configurable ECN threshold, all Frame Relay packets leaving the interface across switched PVCs are marked with FECNs, and all incoming packets are marked with BECNs.

Verification for FRTP is as follows.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
Rack1R3#show frame-relay pvc 132

PVC Statistics for interface Serial1/2 (Frame Relay DCE)

DLCI = 132, DLCI USAGE = SWITCHED, PVC STATUS = ACTIVE, INTERFACE = Serial1/2

  input pkts 3030          output pkts 2938         in bytes 1836582
  out bytes 1688472        dropped pkts 9           in pkts dropped 9
  out pkts dropped 0                out bytes dropped 0
  in FECN pkts 0           in BECN pkts 0           out FECN pkts 0
  out BECN pkts 170        in DE pkts 0             out DE pkts 2257
  out bcast pkts 0         out bcast bytes 0
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
  switched pkts 3030
  Detailed packet drop counters:
  no out intf 0            out intf down 0          no out PVC 0
  in PVC down 0            out PVC down 0           pkt too big 0
  shaping Q full 0         pkt above DE 0           policing drop 9
  pvc create time 05:28:47, last time pvc status changed 05:25:23
  policing enabled, 538 pkts marked DE
  policing Bc  8000        policing Be  8000        policing Tc  125 (msec)
  in Bc pkts   2086        in Be pkts   538         in xs pkts   9
  in Bc bytes  1058508     in Be bytes  590152      in xs bytes  5086
```

Note that the policing rate is not specified explicitly, just B_c, B_e, and T_c values are provided. The packets counted as “xs” (excess) have violated the policy and the system dropped them.

```
Rack1R3#show frame-relay pvc 231

PVC Statistics for interface Serial1/3 (Frame Relay DCE)

DLCI = 231, DLCI USAGE = SWITCHED, PVC STATUS = ACTIVE, INTERFACE = Serial1/3

  input pkts 2939          output pkts 2979         in bytes 1688968
  out bytes 1784910        dropped pkts 59          in pkts dropped 7
  out pkts dropped 52               out bytes dropped 52208
  late-dropped out pkts 52          late-dropped out bytes 52208
  in FECN pkts 0           in BECN pkts 0           out FECN pkts 172
  out BECN pkts 0          in DE pkts 0             out DE pkts 538
  out bcast pkts 0         out bcast bytes 0
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
  switched pkts 2939
  Detailed packet drop counters:
  no out intf 0            out intf down 0          no out PVC 0
  in PVC down 0            out PVC down 0           pkt too big 0
  shaping Q full 0         pkt above DE 52          policing drop 7
  pvc create time 05:28:06, last time pvc status changed 05:25:47
  policing enabled, 2258 pkts marked DE
  policing Bc  0           policing Be  16000       policing Tc  125 (msec)
  in Bc pkts   0           in Be pkts   2258        in xs pkts   7
  in Bc bytes  0           in Be bytes  1377514     in xs bytes  3346
```

This PVC has zero CIR, so all input packets are marked as DE.

Copyright © 2009 Internetwork Expert

Now let's see how oversubscription works with Frame-Relay. Configure both R1 and R3 as HTTP servers, and start downloading IOS images from both routers to R2. Both flows will oversubscribe the connection to R2 on R3, and we should see backpressure of BECN packets on R1.

```
R1:
ip http server
ip http path flash:
!
! Adjust MTU to minimize serialization delay to 10ms on 128K links
! Note that IP MTU of 156 corresponds to 160 bytes of L2 frame
!
interface Serial 0/1.1
 ip mtu 156

R3:
ip http server
ip http path flash:
!
! Adjust MTU to minimize serialization delay to 10ms on 128K links
! Note that IP MTU of 156 corresponds to 160 bytes of L2 frame
!
interface Serial 1/3.1
 ip mtu 156
```

```
Rack1R2#copy http://admin:cisco@155.1.23.3/c2600-adventerprisek9-
mz.124-10.bin null:

Rack1R2#copy http://admin:cisco@155.1.12.2/c2600-adventerprisek9-
mz.124-10.bin null:
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Clear the Frame Relay counters using the command **`clear frame-relay counters`** and check the PVC statistic counters for the DLCI between R1 and R3. Note the large amount of BECN packets sent back to R1 and that 3800 input packets have been marked as DE.

```
Rack1R3#show frame-relay pvc 132

PVC Statistics for interface Serial1/2 (Frame Relay DCE)

DLCI = 132, DLCI USAGE = SWITCHED, PVC STATUS = ACTIVE, INTERFACE = Serial1/2

  input pkts 45041         output pkts 69551        in bytes 7083975
  out bytes 4490548        dropped pkts 118         in pkts dropped 0
  out pkts dropped 118              out bytes dropped 6321
  late-dropped out pkts 118         late-dropped out bytes 6321
  in FECN pkts 0           in BECN pkts 0           out FECN pkts 346
  out BECN pkts 69661      in DE pkts 0             out DE pkts 69669
  out bcast pkts 0         out bcast bytes 0
  5 minute input rate 50000 bits/sec, 40 packets/sec
  5 minute output rate 24000 bits/sec, 66 packets/sec
  switched pkts 45041
  Detailed packet drop counters:
  no out intf 0            out intf down 0          no out PVC 0
  in PVC down 0            out PVC down 0           pkt too big 0
  shaping Q full 0         pkt above DE 118         policing drop 0
  pvc create time 06:13:33, last time pvc status changed 06:10:07
  policing enabled, 3800 pkts marked DE
  policing Bc  8000        policing Be  8000        policing Tc  125 (msec)
  in Bc pkts   46733       in Be pkts   3800        in xs pkts   9
  in Bc bytes  8251771     in Be bytes  1086818     in xs bytes  5086
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

The DLCI between R2 and R3 demonstrates a large number of dropped packets, which is equal to the number of outgoing DE packets (Frame-Relay counters have been reset during the transfer). Note the large amount of FECN marked frames leaving R3 towards R2.

```
Rack1R3#show frame-relay pvc 231

PVC Statistics for interface Serial1/3 (Frame Relay DCE)

DLCI = 231, DLCI USAGE = SWITCHED, PVC STATUS = ACTIVE, INTERFACE = Serial1/3

  input pkts 70245         output pkts 42081        in bytes 4522679
  out bytes 6635281        dropped pkts 3275        in pkts dropped 0
  out pkts dropped 3275             out bytes dropped 498050
  late-dropped out pkts 3275        late-dropped out bytes 498050
  in FECN pkts 0           in BECN pkts 0           out FECN pkts 41442
  out BECN pkts 108        in DE pkts 0             out DE pkts 3275
  out bcast pkts 0         out bcast bytes 0
  5 minute input rate 24000 bits/sec, 67 packets/sec
  5 minute output rate 45000 bits/sec, 35 packets/sec
  switched pkts 70245
  Detailed packet drop counters:
  no out intf 0            out intf down 0          no out PVC 0
  in PVC down 0            out PVC down 0           pkt too big 0
  shaping Q full 0         pkt above DE 3275        policing drop 0
  pvc create time 06:12:49, last time pvc status changed 06:10:30
  policing enabled, 75455 pkts marked DE
  policing Bc  0           policing Be  16000       policing Tc  125 (msec)
  in Bc pkts   308         in Be pkts   75455       in xs pkts   7
  in Bc bytes  144682      in Be bytes  6502239     in xs bytes  3346
```

At the same time the DLCI for the direct connection between R2 and R3 demonstrates a higher transmission rate and no FECN/DE marked frames. The latter is because the DLCI is not switched, and thus is not subject to congestion management. The higher sending rate is because of two facts, the lower RTT, and the shaper at R1.

```
Rack1R3#show frame-relay pvc 233

PVC Statistics for interface Serial1/3 (Frame Relay DCE)

DLCI = 233, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial1/3

  input pkts 229750        output pkts 121198       in bytes 10182318
  out bytes 19352278       dropped pkts 0           in pkts dropped 0
  out pkts dropped 0                out bytes dropped 0
  in FECN pkts 0           in BECN pkts 0           out FECN pkts 0
  out BECN pkts 0          in DE pkts 0             out DE pkts 0
  out bcast pkts 917       out bcast bytes 132560
  5 minute input rate 43000 bits/sec, 122 packets/sec
  5 minute output rate 78000 bits/sec, 61 packets/sec
  pvc create time 06:08:19, last time pvc status changed 06:08:19
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.33 MQC Classification and Marking

- Configure an outbound MQC policy on R4's Serial link to R5 per the following requirements:
  - WWW traffic from servers on VLAN 146 should be marked with an IP Precedence of 2
  - VoIP packets with UDP ports in the destination range of 16384 - 32767 and a Layer 3 packet size of 60 bytes should be marked with DSCP EF
  - ICMP packets larger than 1000 bytes should be dropped
  - All other packets that came from R4's connection to VLAN 146 with an IP precedence of 0 should be remarked with an IP precedence of 1
- Do not use an access-list to classify ICMP packets.

Configuration

```
R4:
ip access-list extended HTTP
 permit tcp 155.1.146.0 0.0.0.255 eq www any
!
ip access-list extended VOICE
 permit udp any any range 16384 32767
!
class-map HTTP
 match access-group name HTTP
!
class-map match-all LARGE_ICMP
 match protocol icmp
 match packet length min 1001
!
class-map match-all VOICE
 match access-group name VOICE
 match packet length min 60 max 60
!
class-map match-all SCAVENGER
 match input-interface FastEthernet0/1
 match ip precedence 0
!
policy-map SERIAL_LINK
 class VOICE
  set ip dscp ef
 class HTTP
  set ip precedence 2
 class LARGE_ICMP
  drop
 class SCAVENGER
  set ip precedence 1
!
interface Serial 0/1
 service-policy output SERIAL_LINK
```

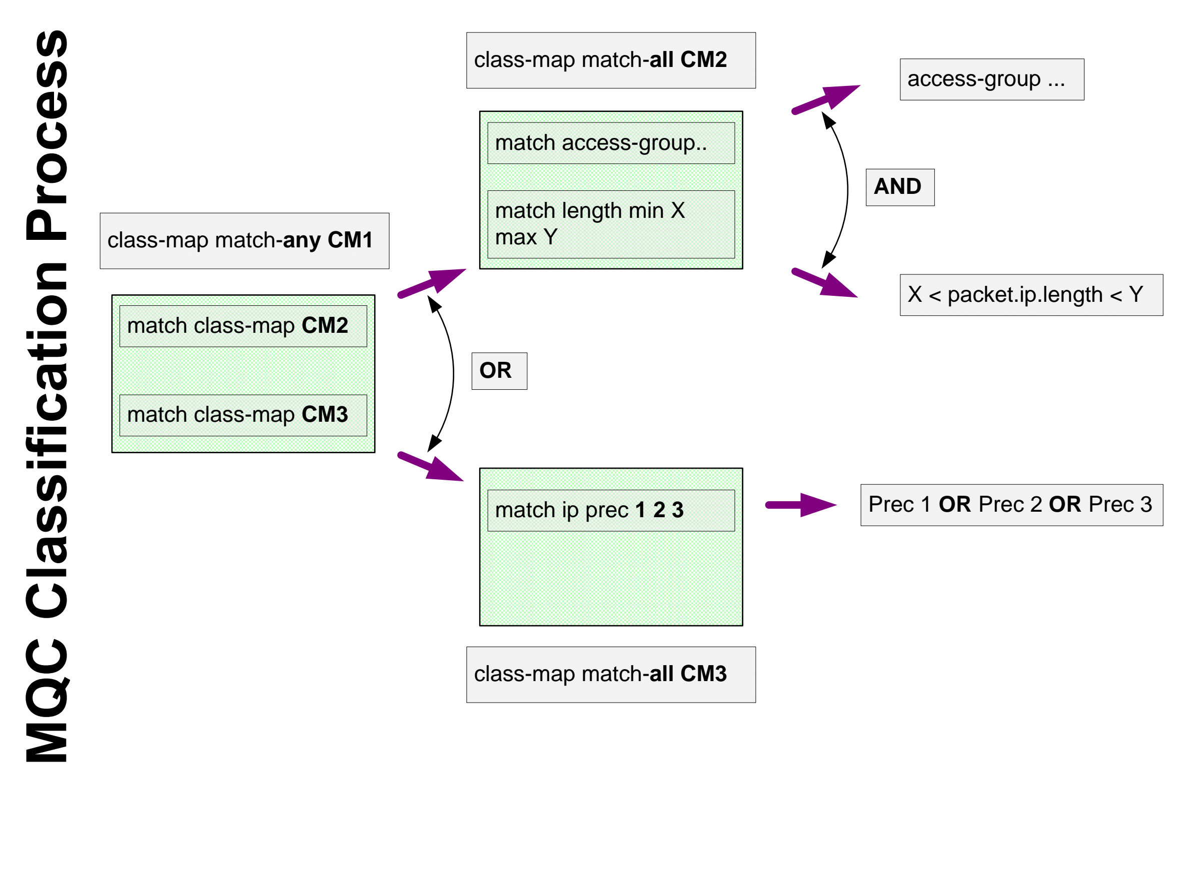
Copyright © 2009 Internetwork Expert

Verification

Note

The Modular Quality of Service Command Line Interface (MQC), also known as Class-Based Weighted Fair Queueing (CBWFQ), unifies all IOS QoS features under a single interface. MQC allows the implementation of a full suite of QoS tools, including classification, congestion management, traffic metering, marking, traffic shaping, and link efficiency. The main advantage of using the MQC over the legacy methods is that multiple QoS features can be applied to the same interface in the same direction. For example, with legacy QoS, you can't apply custom queueing and priority queueing at the same time, but with MQC you can.

Classification in MQC uses case sensitive *class-maps* (not to be confused with a frame-relay map-class) to group criteria. Each class-map performs a logical AND (match-all) or a logical OR (match-any) on its criteria. In other words in a match-all class-map, all matches must be TRUE in order for the class to be TRUE. Class-maps can be nested inside other maps to build complicated classification "AND-OR" logic gates. If multiple match criteria appear on the same line (e.g. match ip dscp, or match ip precedence), they are treated as a logical OR match.



Different IOS versions and platforms support different matches in the class-map, but as a general rule the following classification criteria are supported:

- Named and numbered access-lists - allows matching of IP addresses, TCP/UDP ports, IP protocol numbers, etc.
- Layer 3 packet length
- Layer 2 addresses - source/destination MAC address, Frame-Relay DLCI, etc.
- Packet marking - Layer 2 CoS, Layer 3 DSCP/IP precedence, Frame Relay DE, ATM CLP, MPLS EXP, etc.
- Network-Based Application Recognition (NBAR)
- Inverse logical matching (logical NOT)

When you apply a logical NOT to a nested class-map or multiple criteria in a single line, *De Morgan's law* applies, where NOT (X AND Y) = (NOT X) OR (NOT Y), while NOT (X OR Y) = (NOT X) AND (NOT Y).

Once classification is configured in a class-map, actions are defined for the different classes in a case sensitive *policy-map*. A policy map is an *ordered* list of class-maps with their corresponding actions, similar to how a route-map works. The router matches packets entering/leaving the interface against all class-map entries in the respective input/output policy-map on the interface in a top-down fashion. This means that the first match in a class-map is used for classification, which implies that the order of the classes called in the policy-map is significant. The policy-map actions include marking, shaping, policing, assigning queue weight, compressing, etc. Any unclassified traffic in a policy-map falls into the *class-default* category, is covered in depth, along with the policy-map actions, in following sections.

☠ Pitfall

Correct traffic flow classification within the class-map, and the correct order of operations in the policy-map, is key in the implementation of an MQC policy. In this task the question asks to classify traffic flows from web servers in VLAN 146, which means that they will be using *source port* 80 in their responses to clients. Additionally the SCAVENGER class-map, which matches IP Precedence 0 traffic from VLAN 146, may overlap other traffic classes, such as the HTTP class, which makes it important that SCAVENGER is called last in the policy-map to match any un-classified traffic up to that point.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Note

To verify this configuration first shutdown R5's Frame Relay link. Next, enable the HTTP server service on R1 and start transferring an IOS image from R1 to SW2, start an IP SLA jitter operation on R6 to source "voice-like" packets with the G.729 codec (60 bytes each), and finally send a large number of ICMP packets from R6 to R5, each larger than 1000 bytes.

```
R1:
ip http server
ip http path flash:

R5:
rtr responder

R6:
ip sla monitor 1
 type jitter dest-ipaddr 155.1.45.5 dest-port 16384 codec g729a
 timeout 1000
 frequency 1
!
ip sla monitor schedule 1 life forever start-time now

Rack1SW2#copy http://admin:cisco@155.1.146.1/c2600-adventerprisek9-
mz.124-10.bin null:

Rack1R6#ping 155.1.45.5 repeat 100 size 1004 timeout 0

Type escape sequence to abort.
Sending 100, 1004-byte ICMP Echos to 155.1.45.5, timeout is 0 seconds:
......................................................................
..............................
Success rate is 0 percent (0/100)
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Check the statistics to see the policy-map matches.

```
Rack1R4#show policy-map interface serial 0/1
 Serial0/1

  Service-policy output: SERIAL_LINK

    Class-map: VOICE (match-all)
      2006 packets, 128384 bytes
      30 second offered rate 20000 bps, drop rate 0 bps
      Match: access-group name VOICE
      Match: packet length min 60 max 60
      QoS Set
        dscp ef
          Packets marked 2006

    Class-map: HTTP (match-all)
      899 packets, 521420 bytes
      30 second offered rate 75000 bps, drop rate 0 bps
      Match: access-group name HTTP
      QoS Set
        precedence 2
          Packets marked 899

    Class-map: LARGE_ICMP (match-all)
      100 packets, 100800 bytes
      30 second offered rate 24000 bps, drop rate 24000 bps
      Match: protocol icmp
      Match: packet length min 1001
      drop

    Class-map: SCAVENGER (match-all)
      2 packets, 168 bytes
      30 second offered rate 0 bps, drop rate 0 bps
      Match: input-interface FastEthernet0/1
      Match: ip precedence 0
      QoS Set
        precedence 1
          Packets marked 2

    Class-map: class-default (match-any)
      8 packets, 1568 bytes
      30 second offered rate 0 bps, drop rate 0 bps
      Match: any
```

Note that all MQC configurations use the same unified syntax for configuration and verification.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.34 MQC Bandwidth Reservations and CBWFQ

- Modify R4's MQC policy to meet the following requirements:
  - Set the total size of the MQC queue to 512 buffers
  - The traffic flows for the web servers in VLAN 146 and the IP Precedence 0 packets from VLAN 146 should be guaranteed 32Kbps each
  - Limit the sizes of the FIFO queues for the HTTP and IP Precedence 0 traffic classes to 16 and 24 packets respectively
  - All other unmatched traffic in the policy should run WFQ
  - Dynamic flows in this WFQ should start dropping when they reach 32 packets in length

Configuration

```
R4:
interface Serial 0/1
 bandwidth 128
 max-reserved-bandwidth 100
 no fair-queue
 hold-queue 512 out
!
policy-map SERIAL_LINK
 class VOICE
 class HTTP
  bandwidth 32
  queue-limit 16
 class SCAVENGER
  bandwidth 32
  queue-limit 24
 class class-default
  fair-queue
  queue-limit 32
```

Verification

Note

Class-Based Weighted Fair Queueing (CBWFQ) is the MQC equivalent of a combination of the legacy interface level **`fair-queue`** command and the custom-queue. CBWFQ is used to reserve a *minimum* amount of bandwidth in the output queue for a particular traffic flow in the case of congestion. The **`bandwidth`** statement used in the policy-map for the class is used to compute the scheduling weight of the traffic flow, relative to the configured interface bandwidth.

The logic of CBWFQ is that during congestion, a class with a bandwidth reservation of *ClassBandwidth* will have at least a *ClassBandwidth/InterfaceBandwidth* share of the total interface bandwidth. For this reason it is key that the interface level **`bandwidth`** statement is configured correctly, in respect to the access rate or guaranteed rate of the link in question whenever an MQC policy is applied. The **`bandwidth`** statement under the policy-map does not have a built-in policer, which means that if congestion is not occurring, the traffic flow matched by the class can use as much bandwidth as it wants. Based on this fact the CBWFQ reservation only becomes active once congestion occurs. Configuration of this feature is much more straightforward than the legacy custom-queue, as the bandwidth reserved is not based on a relative ratio like it is in the legacy version.

The feature is deemed Class-Based Weighted Fair Queueing, because the WFQ flow is manually defined through the MQC class-map. The weighting value for the flow is then defined through the **`bandwidth`** value under the respective policy-map. Once the policy-map is applied to the interface the entire software queue changes to CBWFQ. The CLI will not allow you to assign a service-policy with CBWFQ weights unless the interface is using FIFO queuing, which implies that CBWFQ is not compatible with any of the legacy queuing methods, such as custom queueing or priority queueing. These must be disabled explicitly before applying the service-policy.

By default, the sum of all configured bandwidth reservations cannot exceed 75% of the interface bandwidth value, and can be modified like the legacy RTP reservations with the **`max-reserved-bandwidth`** interface command. The system subtracts the bandwidth reserved for a class from the available interface bandwidth for the purpose of admission control. The default limits are designed to leave some best-effort bandwidth to the unclassified traffic, such as routing updates and layer 2 keepalives, which fall into the class-default of the policy-map.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Each class configured with a **`bandwidth`** statement under the policy-map has its own dedicated FIFO queue in the interface's CBWFQ conversation pool. The depth of each FIFO queue can be changed on a per-class basis with the **`queue-limit`** command. The overall WFQ settings, such as the CDT and the total queue size, can be set using the **`queue-limit`** under class-default, and the **`hold-queue out`** command at the interface level.

Recall that with legacy WFQ, flows are classified automatically. The scheduler shares the bandwidth in proportion to the flow's IP precedence value, normalized against the sum of other flows' precedences. Specifically if there are *N* active flows, and flow *i* has IP Precedence IPP(*i*), then any flow *k* is guaranteed a share as:

Share(k) = (IPP(k)+1)/(IPP(1)+IPP(2)…+IPP(N)+N)*100%

Note that each precedence value is shifted by 1 to ensure non-zero values. Since WFQ implements max-min sharing, any flow may claim unused interface bandwidth. Finally, the implementation uses a computation "weight" value based on the formula:

Weight(i) = 32384/(IPP(i)+1) [Formula 1]

This allows for the flow with higher IP precedence to have lower computation weight, and larger share of bandwidth. Remember that computation weights are inversely proportional to the actual weights you use to compute the share of the bandwidth, as the bandwidth is shared in the proportion [1/Weight(1):1/(Weight(2)… :1/Weight(N)]. For example, WFQ assigns a weight of 4048 to a flow with an IP precedence of 7, which is the maximum IPP value. This is an important fact that will be referenced again later.

With the new MQC format, CBWFQ retains the above computations for *unclassified* traffic only. Traffic flows that are explicitly matched in a class-map and have a bandwidth reservation configured are treated differently. To start, this class uses a separate flow queue with its own limits. Secondly, this flow is assigned a computation weight based on the following formula:

Weight(i) = Const*InterfaceBandwidth/Bandwith(i) [Formula 2]

Here *Bandwidth(i)* is the value configured under the respective class using the **`bandwidth`** keyword, and *InterfaceBandwidth* is the value configured under the main interface using the **`bandwidth`** keyword. The value of the constant *Const* depends on the number of flow queues active in CBWFQ, and varies from 1 to 64. The implications of the above formula's calculations are as follows.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Almost any manually configured class has a weight value significantly lower than any automatically classified flow. Recall that IP Precedence 7, the best value, has a best possible weight of 4048 for a dynamic conversation. However for a manually defined class, even in the worst case, *Const* equals 64 (e.g. with 32 flow-queues), the ratio of *interfaceBandwidth/Bandwidth* should be less than 0.02 to be comparable to 4048. This means that a class should have less than a 2% reservation of the interface's bandwidth, which is rare. Furthermore since any user-defined class has its own separate FIFO queue, we can conclude that CBWFQ isolates it from congestive discard drops performed across dynamic WFQ queues. Instead, each class queue is FIFO with a tail-drop discipline by default.

The next question is how CBWFQ shares bandwidth between the user-defined classes. Looking at the weight calculation we can assume that each class has a relative bandwidth share of

Share(i) = Bandwidth(i)/InterfaceBandwidth.

Recall that by default, the sum of all *Bandwidth(i)* reservation is no more than 75% of *InterfaceBandwidth*, due to the **`max-reserved-bandwidth`** defaults. So what happens with the remaining 25%? This is where class-default comes in.

As soon as the **`bandwidth`** keyword is specified under any user-defined class, the interface queue turns into CBWFQ. This means any unmatched flows that fall back into class-default are scheduled using dynamic WFQ weights. This means that automatic classification occurs, along with precedence-based weight assignment and sharing of the single buffer space of WFQ. This behavior is default, even if you did not configure **`fair-queue`** under the class-default.

In addition to dynamic flow queues, WFQ and CBWFQ both have the concept of Link Queues. The system uses these queues for critical control plane traffic, such as routing updates and layer 2 keepalives. Each Link Queue has computation weight of 1024, which is better than any dynamic queue's weight. By default, CBWFQ matches all control plane traffic to class-default, so it is reasonable to assume that some bandwidth (25% by default) is saved for those flows. This safeguard measure prevents user-defined classes from oversubscribing the interface bandwidth, and does not let the other important queues starve.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

If you want to disable fair-queue for unclassified packets, an explicit **bandwidth** value for the class-default can be configured, which turns it into a single FIFO queue. For example, the following code snippet disables fair-queue for unclassified traffic, and gives it a static weight as relative to 96Kbps:

```
policy-map TEST
 class class-default
   bandwidth 96
```

Moreover if you do not define any classes other than class-default, and class-default has a bandwidth value defined, the entire interface queue essentially becomes a FIFO queue.

The following tables illustrate the weight values that are used for CBWFQ. Here N is the constant that defines the number of dynamic WFQ queues Their number is always a power of 2, and equals 2^N. Normally the IOS automatically calculates this based on the interface's bandwidth value, but it can be manually specified with the **fair-queue** command in class-default.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Conversation Numbers	CBWFQ Weight	Description
Below 2^N	32384/(IPP+1)	Used for automatic classification of dynamic WFQ Queues. Configurable via **fair-queue** command under "class-default".
2^N…2^N+7	1024	Link Queues. There are 8 conversations used to queue routing updates (marked as *PAK_PRIORITY* internally) and Layer 2 keepalives.
2^N+8	0	Priority queue which maps directly to legacy WFQ IP RTP Priority. Typically used for VoIP bearer traffic. CBWFQ polices this flow using a configurable token bucket procedure to ensure it does not starve other queues.
Above 2^N+8	Const*IntfBW/ClassBW OR RSVP flow weight	These conversations are used for user-defined traffic classes. Each class has its own FIFO queue and configurable queue depth. In addition, RSVP flows use the same range of conversation numbers.

The "N" constant	Number of dynamic Flows	CBWFQ constant "Const"
4	16	64
5	32	64
6	64	57
7	128	30
8	256	16
9	512	8
10	1024	4
11	2048	2
12	4096	1

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

In summary, the key point about CBWFQ is that it uses the same scheduling logic as the legacy WFQ, but user-configurable classes have a special low-weight, making them more important than any dynamic conversation. Thus, user-defined classes always get the guaranteed proportion of bandwidth, while flows using class-default may starve in the case of congestion.

For verification of this task shut down R5's Frame Relay link to force the traffic from VLAN 146 to the hosts behind R4 to take the path across the serial link. R6 should send IP SLA jitter probes to R5 across the serial link; this flow simulates voice traffic and has no explicit bandwidth set under the class. Next, configure R1 as an HTTP server, and set SW2 to transfer the IOS image from R1, oversubscribing the serial link. Lastly, generate a flow of ICMP packets from R6 to R5, simulating the SCAVENGER class traffic.

```
R1:
ip http server
ip http path flash:

R4:
interface Serial 0/1
 load-interval 30

R5:
rtr responder
!
interface Serial 0/0
 shutdown

R6:
ip sla monitor 1
 type jitter dest-ipaddr 155.1.45.5 dest-port 16384 codec g729a
 timeout 1000
 frequency 1
!
ip sla monitor schedule 1 life forever start-time now
```

```
Rack1SW2#copy http://admin:cisco@155.1.146.1/c2600-adventerprisek9-
mz.124-10.bin null:

Rack1R6#ping 155.1.45.5 repeat 100000 size 150

Type escape sequence to abort.
Sending 100000, 150-byte ICMP Echos to 155.1.45.5, timeout is 2
seconds:
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Check the policy-map matches and observe the measured traffic rates. Note that the load-interval is set to 30 seconds to ensure more adaptive response to changes in traffic patterns.

```
Rack1R4#show policy-map interface serial 0/1
 Serial0/1
  Service-policy output: SERIAL_LINK

    Class-map: VOICE (match-all)
      173240 packets, 11087360 bytes
      30 second offered rate 23000 bps, drop rate 0 bps
      Match: access-group name VOICE
      Match: packet length min 60 max 60
      QoS Set
        dscp ef
          Packets marked 173241

    Class-map: HTTP (match-all)
      45477 packets, 26371423 bytes
      30 second offered rate 104000 bps, drop rate 0 bps
      Match: access-group name HTTP
      QoS Set
        precedence 2
          Packets marked 45477
      Queueing
        Output Queue: Conversation 41
        Bandwidth 32 (kbps)Max Threshold 16 (packets)
        (pkts matched/bytes matched) 6020/3491600
        (depth/total drops/no-buffer drops) 6/0/0

    Class-map: LARGE_ICMP (match-all)
      0 packets, 0 bytes
      30 second offered rate 0 bps, drop rate 0 bps
      Match: protocol icmp
      Match: packet length min 1001
      drop

    Class-map: SCAVENGER (match-all)
      38429 packets, 5905886 bytes
      30 second offered rate 15000 bps, drop rate 0 bps
      Match: input-interface FastEthernet0/1
      Match: ip precedence 0
      QoS Set
        precedence 1
          Packets marked 104852
      Queueing
        Output Queue: Conversation 42
        Bandwidth 32 (kbps)Max Threshold 24 (packets)
        (pkts matched/bytes matched) 2286/351274
        (depth/total drops/no-buffer drops) 0/0/0

    Class-map: class-default (match-any)
      908 packets, 205544 bytes
      30 second offered rate 0 bps, drop rate 18000 bps
      Match: any
      Queueing
        Flow Based Fair Queueing
        Maximum Number of Hashed Queues 32
        (total queued/total drops/no-buffer drops) 64/9916/0
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Note that the conversation numbers for the user-defined classes are 41 and 42, which means that there are 32 dynamic queues plus 8 link queues. The IOS picked the number of dynamic queues automatically based on interface bandwidth of 128Kbps.

Next, the class for VOICE traffic demonstrates a measured bitrate close to 24Kbps. However, we also see the high drop rate under class-default, along with its queue stats. Since the VOICE class has no weight of its own, CBWFQ assigns it to a dynamic queue.

Two other user-configured classes (HTTP and SCAVENGER) have equal bandwidth weights configured, but the SCAVENGER class is not consuming all of its allocated bandwidth. This is why the HTTP class uses the unclaimed resources. Finally, since both user configurable classes have better weights than the WFQ dynamic queues, the VOICE traffic cannot get enough bandwidth. Thus, CBWFQ drops most of the VOICE class traffic.

Now let's see the CBWFQ queue contents:

```
Rack1R4#show queueing interface serial 0/1
Interface Serial0/1 queueing strategy: fair
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 11322
  Queueing strategy: Class-based queueing
  Output queue: 12/512/32/0 (size/max total/threshold/drops)
     Conversations  3/4/32 (active/max active/max total)
     Reserved Conversations 3/3 (allocated/max allocated)
     Available Bandwidth 40 kilobits/sec

  (depth/weight/total drops/no-buffer drops/interleaves) 6/256/0/0/0
  Conversation 41, linktype: ip, length: 580
  source: 155.1.146.1, destination: 155.1.58.8, id: 0x74CD, ttl: 254,
  TOS: 64 prot: 6, source port 80, destination port 11003

  (depth/weight/total drops/no-buffer drops/interleaves) 1/256/0/0/0
  Conversation 42, linktype: ip, length: 154
  source: 155.1.146.6, destination: 155.1.45.5, id: 0x874A, ttl: 254, prot: 1

  (depth/weight/total drops/no-buffer drops/interleaves) 5/5397/601/0/0
  Conversation 5, linktype: ip, length: 64
  source: 155.1.146.6, destination: 155.1.45.5, id: 0x02B8, ttl: 254,
  TOS: 184 prot: 17, source port 50700, destination port 16384
```

There are three flows active in the queue. The total queue length is 512, which is what we set the hold-queue to. A total of 12 packets are currently queued up. Note that packet sizes include layer 2 overhead, which is 4 bytes for HDLC.

The first displayed flow corresponds to the HTTP traffic, and has a maximum accumulated queue depth compared to the other flows. Its weight is 256, which corresponds to 25% of the interface bandwidth, or 32Kbps (note that CBWFQ does not take the IP Precedence of the HTTP traffic into account). The ICMP traffic has the same weight, while the flow corresponding to VOICE class has conversation number 5. This means it uses a dynamic queue for scheduling, and its weight of 5397 is actually 32384/(5+1) – the automatic weight corresponding to IP Precedence 5 traffic. We can calculate the guaranteed bandwidth shares based on observed weights using the following computations.

First, the bandwidth is shared in values inversely proportional to computation weights:

1/256:1/256:1/5397.

Normalize the proportion by dividing by the smallest number:

5397/256:5397/256:1 = 21:21:1.

Now the actual bandwidth shares are (roughly):

[21/(21+21+1), 21/(21+21+1), 1/(21+21+1)] * 100% = 49% 49% 2%.

Therefore the net effect is that the available bandwidth is almost equally divided between the two user-defined classes, and the dynamic conversation gets a much smaller share. To confirm this, verify the IP SLA statistics on R6, and note that conversation statistics for the VOICE traffic exhibits the largest number of packet drops.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
Rack1R6#show ip sla monitor statistics 1
Round trip time (RTT)   Index 1
        Latest RTT: 6 ms
Latest operation start time: 04:09:04.672 UTC Fri Aug 15 2008
Latest operation return code: OK
RTT Values
        Number Of RTT: 29
        RTT Min/Avg/Max: 37/228/434 ms
Latency one-way time milliseconds
        Number of one-way Samples: 0
        Source to Destination one way Min/Avg/Max: 0/0/0 ms
        Destination to Source one way Min/Avg/Max: 0/0/0 ms
Jitter time milliseconds
        Number of Jitter Samples: 27
        Source to Destination Jitter Min/Avg/Max: 5/13/17 ms
        Destination to Source Jitter Min/Avg/Max: 1/1/3 ms
Packet Loss Values
        Loss Source to Destination: 655         Loss Destination to Source: 64
        Out Of Sequence: 0      Tail Drop: 252  Packet Late Arrival: 64
Voice Score Values
        Calculated Planning Impairment Factor (ICPIF): 40
MOS score: 2.75
Number of successes: 136
Number of failures: 18
Operation time to live: Forever
```

Even though the traffic is marked with IP Precedence 5, CBWFQ penalizes the flow as compared to the other user-defined classes. The user-defined flows always get what they ask for, and are always preferred over the dynamic conversations.

10.35 MQC Bandwidth Percent

- Modify R4's previous user-defined reservations to use a percentage reservation of 25% of the link bandwidth each, as compared to 32Kbps.

Configuration

```
R4:
policy-map SERIAL_LINK
 class HTTP
  no bandwidth
  bandwidth percent 25
 class SCAVENGER
  no bandwidth
  bandwidth percent 25
```

Verification

Note

Recall that CBWFQ does not use the absolute **`bandwidth`** value in a class to directly compute the scheduling weight, but instead it is computed relative to the interface's bandwidth value as *Weight(i) = Const*InterfaceBandwidth/Bandwith(i)*. Since this value is in reality a relative reservation in the first place, the calculation can be simplified by ignoring the interface level bandwidth, and expressing the reservation as a percentage. In this manner with the **`bandwidth percent`** command the weight calculation changes to:

Weight(i) = Const*100/Percent(i)

This implementation is useful in designs where a common template of reservation is applied to multiple links of differing bandwidth values. For example a standard policy-map could be defined to reserve 20% of link bandwidth for HTTP flows, and then applied to both 100Mbps FastEthernet and 45Mbps DS3. The resulting CBWFQ weights would be the same, but the actual bandwidth value differs based on the underlying physical link bandwidth.

Like the Kbps reservation through the **`bandwidth`** command, the sum of all configured **`bandwidth percent`** values in all classes of a policy-map cannot exceed the **`max-reserved-bandwidth`** configured on the respective interface. In addition, it is not possible to mix the **`bandwidth`** and **`bandwidth percent`** commands in the same policy-map; the units must be the same among classes.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Verification of this configuration is the same as the previous task. Shut down R5's Frame Relay link to force the traffic from VLAN 146 to the hosts behind R4 to take the path across the serial link. R6 should send IP SLA jitter probes to R5 across the serial link; this flow simulates voice traffic and has no explicit bandwidth set under the class. Next, configure R1 as an HTTP server, and set SW2 to transfer the IOS image from R1, oversubscribing the serial link. Lastly, generate a flow of ICMP packets from R6 to R5, simulating the SCAVENGER class traffic.

```
R1:
ip http server
ip http path flash:

R4:
interface Serial 0/1
 load-interval 30

R5:
rtr responder
!
interface Serial 0/0
 shutdown

R6:
ip sla monitor 1
 type jitter dest-ipaddr 155.1.45.5 dest-port 16384 codec g729a
 timeout 1000
 frequency 1
!
ip sla monitor schedule 1 life forever start-time now

Rack1SW2#copy http://admin:cisco@155.1.146.1/c2600-adventerprisek9-
mz.124-10.bin null:

Rack1R6#ping 155.1.45.5 repeat 100000 size 150

Type escape sequence to abort.
Sending 100000, 150-byte ICMP Echos to 155.1.45.5, timeout is 2
seconds:
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Check the policy-map statistics as follows.

```
Rack1R4#show policy-map interface serial 0/1
 Serial0/1

  Service-policy output: SERIAL_LINK

    Class-map: VOICE (match-all)
      183205 packets, 11725120 bytes
      30 second offered rate 23000 bps, drop rate 0 bps
      Match: access-group name VOICE
      Match: packet length min 60 max 60
      QoS Set
        dscp ef
          Packets marked 183205

    Class-map: HTTP (match-all)
      50438 packets, 29248267 bytes
      30 second offered rate 105000 bps, drop rate 0 bps
      Match: access-group name HTTP
      QoS Set
        precedence 2
          Packets marked 50438
      Queueing
        Output Queue: Conversation 41
        Bandwidth 25 (%)
        Bandwidth 32 (kbps)Max Threshold 64 (packets)
        (pkts matched/bytes matched) 361/208844
        (depth/total drops/no-buffer drops) 2/0/0

    Class-map: LARGE_ICMP (match-all)
      0 packets, 0 bytes
      30 second offered rate 0 bps, drop rate 0 bps
      Match: protocol icmp
      Match: packet length min 1001
      drop

    Class-map: SCAVENGER (match-all)
      41195 packets, 6331150 bytes
      30 second offered rate 16000 bps, drop rate 0 bps
      Match: input-interface FastEthernet0/1
      Match: ip precedence 0
      QoS Set
        precedence 1
          Packets marked 107619
      Queueing
        Output Queue: Conversation 42
        Bandwidth 25 (%)
        Bandwidth 32 (kbps)Max Threshold 64 (packets)
        (pkts matched/bytes matched) 196/30114
        (depth/total drops/no-buffer drops) 1/0/0

    Class-map: class-default (match-any)
      965 packets, 218264 bytes
      30 second offered rate 0 bps, drop rate 17000 bps
      Match: any
      Queueing
        Flow Based Fair Queueing
        Maximum Number of Hashed Queues 32
        (total queued/total drops/no-buffer drops) 64/18239/0
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

This output displays the percent values along with inferred absolute bandwidth reservation, where bandwidth = percent*interface_bandwidth. Other than this the bandwidth shares remain the same. Next look at the queue content and note the CBWFQ weights.

```
Rack1R4#show queueing interface serial 0/1
Interface Serial0/1 queueing strategy: fair
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 28259
  Queueing strategy: Class-based queueing
  Output queue: 70/512/32/28259 (size/max total/threshold/drops)
     Conversations  3/6/32 (active/max active/max total)
     Reserved Conversations 3/3 (allocated/max allocated)
     Available Bandwidth 40 kilobits/sec

  (depth/weight/total drops/no-buffer drops/interleaves) 1/256/0/0/0
  Conversation 42, linktype: ip, length: 154
  source: 155.1.146.6, destination: 155.1.45.5, id: 0x9D03, ttl: 254, prot: 1

  (depth/weight/total drops/no-buffer drops/interleaves) 5/256/0/0/0
  Conversation 41, linktype: ip, length: 580
  source: 155.1.146.1, destination: 155.1.58.8, id: 0x98D6, ttl: 254,
  TOS: 64 prot: 6, source port 80, destination port 11003

  (depth/weight/total drops/no-buffer drops/interleaves) 64/5397/686/0/0
  Conversation 27, linktype: ip, length: 64
  source: 155.1.146.6, destination: 155.1.45.5, id: 0x02ED, ttl: 254,
  TOS: 184 prot: 17, source port 51424, destination port 16384
```

Just as before, flows 41 and 42 are assigned weights of 256, while the VOICE flow with a weight of 5397 is heavily penalized as compared to the user-defined classes.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.36 MQC LLQ and Remaining Bandwidth Reservations

- Modify R4's policy to R5 as follows:
    - Configure Low Latency Queueing to allow for exactly one VoIP call to be prioritized
    - Additional VoIP calls should be allowed only if the link is not congested, but they should not be prioritized
    - Reserve 33% of the remaining bandwidth to the HTTP and SCAVENGER classes each
- Assume that VoIP calls are using the G.729 codec, which generates 50 packets per second with a Layer 3 size of 60 bytes.

Configuration

```
R4:
policy-map SERIAL_LINK
 class VOICE
  priority 27
 class HTTP
  no bandwidth
  bandwidth remaining percent 33
 class SCAVENGER
  no bandwidth
  bandwidth remaining percent 33
```

Verification

Note

The Low Latency Queuing (LLQ) feature is the MQC equivalent of the legacy IP RTP Priority, however classification is not limited to just RTP packets. Like IP RTP Priority, LLQ uses the special conversation number 2^N+8, where *N* defines the number of dynamic conversations. For example, if there are 32 (2^5) dynamic queues, then the LLQ conversation is number 40. This conversation has a weight value of 0, which means that it is always serviced first. In order to prevent starvation of other queues, the packets de-queued from the LLQ conversation are metered using a simple token bucket, with a configurable rate and burst size. Packets that exceed the token bucket are dropped (policed) during times of congestion, while if there is no congestion, exceeding traffic is not dropped, but it is simply not prioritized. For these reasons it is better to use the MQC LLQ feature in practical designs versus the legacy priority-queue or IP RTP Priority, because the legacy priority-queue does not have a policer, IP RTP Priority can not classify other traffic flows, and neither of them can be used in conjunction with other QoS features, such as a bandwidth reservation.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Multiple classes inside a single policy-map can use the priority keyword, but only one single priority queue exists. This design of multiple priority classes is used to ensure that one priority flow does not starve another priority flow. For example assume that VoIP and Video traffic are grouped together in one class-map, which has a **`priority 128`** value defined under a policy-map. If Video traffic is using 128Kbps, VoIP traffic can potentially be dropped (policed), or at least not be guaranteed priority. This is because both traffic flows must contend for the same 128Kbps rate. However if separate VoIP and Video class-maps are defined, each with a **`priority 64`** rate configured under the same policy-map, each flow is guaranteed priority up to 64Kbps. Although this still totals 128Kbps of priority, one class can't completely starve the other.

Note that the LLQ policer takes the layer 2 packet length into account, so in this case the HDLC overhead of 7 bytes is added to the 60 bytes of layer 3 VoIP payload, which results in a value close to 27Kbps (67 bytes/packet * 50 packets/second * 8 bits/byte = 26800bps) being reserved. The burst value for the token bucket can be configured manually, or automatically calculated by the IOS itself. A burst size of 1 or 1.5 seconds should be enough, since this is probably the maximum duration of a single spoken word in VoIP. See the *Further Learning* section following the verification for more info about computing LLQ sizes for VoIP calls.

Once the LLQ priority reservation is configured, the CBWFQ algorithm subtracts the reserved bandwidth of the priority queue from the interface's available bandwidth. Like in IP RTP Priority, the available bandwidth value defaults to 75% of the interface bandwidth, and can be modified with the **`max-reserved-bandwidth`** interface level command. The remaining bandwidth can be used to create a relative bandwidth reservation for other classes in the CBWFQ. By this logic, instead of configuring absolute bandwidth values - either numerically or in interface bandwidth percents - for other reservations, you can just specify relative shares of the bandwidth that remains after the priority queue finished its run. This method is seen in the above configuration through the **`bandwidth remaining percent`** command.

For example in this particular task the link between R4 and R5 was configured with a interface **`bandwidth`** value of 128Kbps, and a **`max-reserved-bandwidth`** of 100, meaning that 100% of 128Kbps is reservable. With the 27Kbps reservation for VoIP in the LLQ, 101Kbps is the remaining bandwidth. Since the HTTP and SCAVENGER classes reserve 33% of the remaining bandwidth, they are each allocated a minimum of 33Kbps ((128-27) * .33 = 33).

To verify this configuration generate three different traffic flows, like in the previous examples. First, shutdown the Frame Relay interface of R5 to force the traffic from VLAN 146 to hosts behind R5 to take path across the serial link. Configure R6 to send IP SLA jitter probes to R5; this flow simulates the voice traffic in the LLQ. Next, configure R1 as an HTTP server, and set SW2 to transfer the IOS image from R1, causing the serial link to be congested. Lastly, generate a flow of ICMP packets from R6 to R5, simulating the SCAVENGER class traffic.

```
R1:
ip http server
ip http path flash:

R4:
interface Serial 0/1
 load-interval 30

R5:
rtr responder
!
interface Serial 0/0
 shutdown

R6:
ip sla monitor 1
 type jitter dest-ipaddr 155.1.45.5 dest-port 16384 codec g729a
 timeout 1000
 frequency 1
!
ip sla monitor schedule 1 life forever start-time now

Rack1SW2#copy http://admin:cisco@155.1.146.1/c2600-adventerprisek9-
mz.124-10.bin null:

Rack1R6#ping 155.1.45.5 repeat 100000 size 150

Type escape sequence to abort.
Sending 100000, 150-byte ICMP Echos to 155.1.45.5, timeout is 2
seconds:
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Check the statistics for policy map matches. Note that the priority queue for VOICE traffic uses conversation 40, and its 30 seconds offered rate is 25Kbps. Since the voice packets use the priority queue now, the remaining classes have less bandwidth to share. Therefore the bandwidth measured for the HTTP class is only 78Kbps, compared to the higher value available seen in prior configurations without the LLQ. Also, note that SCAVENGER class does not use all its guaranteed bandwidth, so the HTTP class can use the remaining part.

```
Rack1R4#show policy-map interface serial 0/1
 Serial0/1

  Service-policy output: SERIAL_LINK

    Class-map: VOICE (match-all)
      26349 packets, 1686336 bytes
      30 second offered rate 25000 bps, drop rate 0 bps
      Match: access-group name VOICE
      Match: packet length min 60 max 60
      QoS Set
        dscp ef
          Packets marked 26349
      Queueing
        Strict Priority
        Output Queue: Conversation 40
        Bandwidth 27 (kbps) Burst 3400 (Bytes)
        (pkts matched/bytes matched) 26274/1681536
        (total drops/bytes drops) 0/0

    Class-map: HTTP (match-all)
      9491 packets, 5504156 bytes
      30 second offered rate 78000 bps, drop rate 0 bps
      Match: access-group name HTTP
      QoS Set
        precedence 2
          Packets marked 9491
      Queueing
        Output Queue: Conversation 41
        Bandwidth remaining 33 (%)Max Threshold 64 (packets)
        (pkts matched/bytes matched) 3092/1793048
        (depth/total drops/no-buffer drops) 6/0/0

    Class-map: LARGE_ICMP (match-all)
      0 packets, 0 bytes
      30 second offered rate 0 bps, drop rate 0 bps
      Match: protocol icmp
      Match: packet length min 1001
      drop
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
    Class-map: SCAVENGER (match-all)
      8739 packets, 1343986 bytes
      30 second offered rate 19000 bps, drop rate 0 bps
      Match: input-interface FastEthernet0/1
      Match: ip precedence 0
      QoS Set
        precedence 1
          Packets marked 75162
      Queueing
        Output Queue: Conversation 42
        Bandwidth remaining 33 (%)Max Threshold 64 (packets)
        (pkts matched/bytes matched) 2843/437262
        (depth/total drops/no-buffer drops) 1/0/0

    Class-map: class-default (match-any)
      139 packets, 31440 bytes
      30 second offered rate 0 bps, drop rate 0 bps
      Match: any
      Queueing
        Flow Based Fair Queueing
        Maximum Number of Hashed Queues 32
        (total queued/total drops/no-buffer drops) 0/0/0
```

View the CBWFQ contents, and note that the flow corresponding to the VOICE class has weight value of zero, and CBWFQ no longer penalizes it by excessive packet drops. Also, note the weight of two other remaining flows (HTTP and SCAVENGER classes). The value is “194”, which is actually 64/0.33. Thus both flows have equal chances of claiming the bandwidth remaining after the priority queue.

```
Rack1R4#show queueing interface serial 0/1
Interface Serial0/1 queueing strategy: fair
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 29854
  Queueing strategy: Class-based queueing
  Output queue: 8/512/32/29854 (size/max total/threshold/drops)
     Conversations  3/6/32 (active/max active/max total)
     Reserved Conversations 3/3 (allocated/max allocated)
     Available Bandwidth 101 kilobits/sec

  (depth/weight/total drops/no-buffer drops/interleaves) 1/0/0/0/0
  Conversation 40, linktype: ip, length: 64
  source: 155.1.146.6, destination: 155.1.45.5, id: 0x02A1, ttl: 254,
  TOS: 184 prot: 17, source port 51032, destination port 16384

  (depth/weight/total drops/no-buffer drops/interleaves) 1/194/0/0/0
  Conversation 42, linktype: ip, length: 154
  source: 155.1.146.6, destination: 155.1.45.5, id: 0xA6B0, ttl: 254, prot: 1

  (depth/weight/total drops/no-buffer drops/interleaves) 6/194/0/0/0
  Conversation 41, linktype: ip, length: 580
  source: 155.1.146.1, destination: 155.1.58.8, id: 0xA432, ttl: 254,
  TOS: 64 prot: 6, source port 80, destination port 11003
```

Copyright © 2009 Internetwork Expert

Further Learning

Computing voice bandwidth is usually required for scenarios where you provision an LLQ based on the number of calls and VoIP codec used. To accomplish this you need to account for codec payload rate, the Layer 3 overhead (RTP and UDP headers), and the Layer 2 overhead (Frame-Relay, Ethernet, HDLC etc. headers). Accounting for Layer 2 overhead is important since the LLQ policer takes this overhead in account when enforcing the maximum rate.

For this example let's consider two codecs for bandwidth computation, G.729 and G.711. By default both codecs generate 50 VoIP packets per second, however the codec framing rate is 10ms (100 packets per second). Thus each VoIP packet carries two frames with VoIP samples. The frame sizes are 10 bytes and 80 bytes for G.729 and G.711 respectively. Therefore G.729 generates (10*2)*50*8=8000bps and G.711 generates (80*2)*50*8=64000bps of payload rate.

The RTP header size is 12 bytes, and the UDP header size is 8 bytes. A typical IP header (with no options) is 20 bytes. Therefore, the Layer 3 overhead is 40 bytes, if we don't use header compression.

The following are the formats WAN frames commonly used to transport voice (with or without FRF.12/MLP fragmentation – voice packets are never fragmented).

Cisco FR Encapsulation 7 bytes ovehead	IETF FR Encapsulation 7 bytes overhead	PPP over Frame-Relay 9 bytes overhead	HDLC 7 bytes overhead
Flag (1 byte)	Flag (1 byte)	Flag (1 byte)	Flag (1 byte)
Address Field (2 bytes)	Address Field (2 bytes)	Address Field (2 bytes)	Address Field (1 byte)
EtherType (2 bytes)	Control (1 byte)	Control (1 byte)	Control (1 byte)
	NLPID (1 byte)	NLPID (1 byte) = PPP	Protocol (2 bytes)
		PPP ID (2 bytes)	
IP+UDP+RTP+ Voice Frames	**IP+UDP+RTP+ Voice Frames**	**IP+UDP+RTP+ Voice Frames**	**IP+UDP+RTP+ Voice Frames**
FCS (2 bytes)	FCS (2 bytes)	FCS (2 bytes)	FCS (2 bytes)

As we can see both Cisco and IETF Frame-Relay encapsulations add 7 bytes of Layer 2 overhead to VoIP packets. The same holds true for HDLC encapsulation (which is not very common but added here for sake of completeness). PPP over Frame-Relay adds 9 bytes of overhead – the maximum overhead of all encapsulation types.

Using the information above, you can compute bandwidth usage for uncompressed voice traffic flow across any WAN connection. For example, let's compute the bandwidth consumption for 3 G.729 calls across Frame-Relay link with FRF.12 fragmentation. First off, FRF.12 does not fragment voice packets if set up properly, as the fragment size is greater than or equal to the VoIP packet. Thus we assume 7 bytes of Layer 2 overhead for any of the Frame Relay encapsulation types. Next, the size of the payload + Layer 3 overhead is 20 bytes + 40 bytes = 60 bytes. Based on the 50pps rate we end up with the bandwidth value of (20+40+7)*50*8=26800bps. If you want to use the G.711 codec, then replace the 20 bytes payload with 160 bytes. The result is (160+40+7)*50*8=82800bps.

Another thing to consider is IP/RTP/UDP header compression. Cisco's implementation reduces the total overhead of 40 bytes (12+8+20) down to 2 bytes (no UDP checksum). This limits the Layer 3 overhead to just 2 bytes. Based on this reduction from 40 to 2 bytes we could compute the bandwidth usage for a G.729 call over MLPoFR with UDP header compression as (20+2+9)*50*8=12400bps. The same computations for compressed G.729 over Frame Relay with or without FRF.12 yields (20+2+7)*50*8=11600bps.

For VoIP over Ethernet the Layer 2 overhead for is typically 18 bytes – 14 bytes for the Ethernet header and 4 bytes for FCS (32 bits). If the frame carries a VLAN tag, add another 4 bytes, for 22 bytes of total overhead. Note that you typically see the G.711 codec used over LAN links.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.37 MQC WRED

- Modify R4's policy so that the HTTP traffic class uses random detection for dropping as opposed to tail drop.
- Start dropping packets randomly when the average queue size is between 4 and 16.
- Drop 1 of every 4 packets when the average queue size reaches the maximum threshold.

Configuration

```
R4:
policy-map SERIAL_LINK
 class HTTP
  random-detect
  random-detect precedence 2 4 16 4
```

Verification

Note

CBWFQ supports three different drop policies, classic tail-drop, which is the default for user-defined classes, Congestive Discard for WFQ, and Random Early Detection. When you apply **`random-detect`** command under a user-defined class, it automatically removes **`queue-limit`** command and enforces RED as the drop policy. When using RED with CBWFQ, each flow is considered an individual FIFO queue. This is similar to flow-based WRED, though the big improvement is the ability to use random drop per flow, not per whole queue.

Similar to legacy per-interface WRED, you can tune settings per IP precedence value or per DSCP. In our case HTTP traffic uses IP precedence of 2, therefore we tune the RED settings for IP precedence 2.

To verify this configuration generate three traffic flows similar to before. The goal is to see random drops occur under the HTTP traffic class. To do this first shutdown the Frame Relay interface of R5 to force the traffic from VLAN146 to the hosts behind R5 take path across the serial link. R6 should generate IP SLA jitter probes to R5 across the serial link from R4 to R5. This flow simulates voice traffic. Next, configure R1 as an HTTP server and set SW2 to transfer the IOS image from R1, oversubscribing the serial link. Lastly, generate a flow of ICMP packets from R6 to R5, simulating the SCAVENGER class traffic.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
R1:
ip http server
ip http path flash:

R4:
interface Serial 0/1
 load-interval 30

R5:
rtr responder
!
interface Serial 0/0
 shutdown

R6:
ip sla monitor 1
 type jitter dest-ipaddr 155.1.45.5 dest-port 16384 codec g729a
 timeout 1000
 frequency 1
!
ip sla monitor schedule 1 life forever start-time now
```

```
Rack1SW2#copy http://admin:cisco@155.1.146.1/c2600-adventerprisek9-
mz.124-17.bin null:

Rack1R6#ping 155.1.45.5 repeat 100000 size 150

Type escape sequence to abort.
Sending 100000, 150-byte ICMP Echos to 155.1.45.5, timeout is 2
seconds:
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

Check the statistics for the service policy. Note the long detailed output for the HTTP class. It demonstrates the number of random drops occurred in the queue for IP traffic marked with IP precedence of 2.

```
Rack1R4#show policy-map interface serial 0/1
 Serial0/1

  Service-policy output: SERIAL_LINK

    Class-map: VOICE (match-all)
      255190 packets, 16332160 bytes
      30 second offered rate 23000 bps, drop rate 0 bps
      Match: access-group name VOICE
      Match: packet length min 60 max 60
      QoS Set
        dscp ef
          Packets marked 255190
      Queueing
        Strict Priority
        Output Queue: Conversation 40
        Bandwidth 26 (kbps) Burst 3200 (Bytes)
        (pkts matched/bytes matched) 67601/4326464
        (total drops/bytes drops) 0/0
```

```
    Class-map: HTTP (match-all)
      76985 packets, 44639911 bytes
      30 second offered rate 77000 bps, drop rate 2000 bps
      Match: access-group name HTTP
      QoS Set
        precedence 2
          Packets marked 76985
      Queueing
        Output Queue: Conversation 41
        Bandwidth remaining 33 (%)
        (pkts matched/bytes matched) 24482/14195104
        (depth/total drops/no-buffer drops) 0/16/0
         exponential weight: 9
         mean queue depth: 0

  class    Transmitted       Random drop       Tail drop    Minimum Maximum  Mark
           pkts/bytes        pkts/bytes        pkts/bytes    thresh  thresh  prob
      0        0/0               0/0               0/0          20      40  1/10
      1        0/0               0/0               0/0          22      40  1/10
      2      713/408916         16/9280            0/0           4      16  1/4
      3        0/0               0/0               0/0          26      40  1/10
      4        0/0               0/0               0/0          28      40  1/10
      5        0/0               0/0               0/0          30      40  1/10
      6        0/0               0/0               0/0          32      40  1/10
      7        0/0               0/0               0/0          34      40  1/10
   rsvp        0/0               0/0               0/0          36      40  1/10

<snip>
```

Now look at the CBWFQ queue contents. Pay special attention to the flow corresponding to the HTTP file transfer. Note that the HTTP flow has the fields random/tail drops, while the ICMP flow does not.

```
Rack1R4#show queueing interface serial 0/1
Interface Serial0/1 queueing strategy: fair
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 29928
  Queueing strategy: Class-based queueing
  Output queue: 3/512/32/29928 (size/max total/threshold/drops)
     Conversations  3/6/32 (active/max active/max total)
     Reserved Conversations 3/3 (allocated/max allocated)
     Available Bandwidth 78 kilobits/sec

  (depth/weight/total drops/no-buffer drops/interleaves) 1/0/0/0/0
  Conversation 40, linktype: ip, length: 64
  source: 155.1.146.6, destination: 155.1.45.5, id: 0x03C0, ttl: 254,
  TOS: 184 prot: 17, source port 57336, destination port 16384

  (depth/weight/total drops/no-buffer drops/interleaves) 1/194/0/0/0
  Conversation 42, linktype: ip, length: 154
  source: 155.1.146.6, destination: 155.1.45.5, id: 0xFA71, ttl: 254, prot: 1

  (depth/weight/total drops/no-buffer drops/random/tail/interleaves)
1/194/74/0/0/0/0
  Conversation 41, linktype: ip, length: 580
  source: 155.1.146.1, destination: 155.1.58.8, id: 0xF9C0, ttl: 254,
  TOS: 64 prot: 6, source port 80, destination port 11003
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.38 MQC Dynamic Flows and WRED

- Modify the CBWFQ configuration on R4 to activate random drops for unclassified traffic's dynamic flows.
- Change the minimum and maximum RED thresholds for traffic with an IP precedence of one to 1 and 40 respectively.
- As the queue depth grows close to the maximum threshold, the probability of packet discard should be 25%.

Configuration

```
R4:
policy-map SERIAL_LINK
 class class-default
  fair-queue
  random-detect
  random-detect precedence 1   1     40    4
```

Verification

✎ **Note**

There are two ways to enable WRED within class-default. The first is to configure a **bandwidth** reservation statement - turning the class's queue into a FIFO queue - and then enabling RED, and the second is to enable RED with WFQ. The second case activates RED dropping to replace Congestive Discard Threshold-based drops for dynamic flows. Think of this as a closer equivalent to flow-based RED, with automatic flow classification. All other WRED parameters remain the same as in the legacy case (e.g. averaging exponential factor, thresholds, etc).

Note that CBWFQ classifies the special *link queues* under class-default, which are therefore subject to WRED drop as soon as it's enabled. However, the default thresholds are high enough for the link queues to prevent the drops of critical control plane traffic.

To verify this configuration generate three traffic flows similar to before. The goal is to see random drops occur under the HTTP traffic class. To do this first shutdown the Frame Relay interface of R5 to force the traffic from VLAN146 to the hosts behind R5 take path across the serial link. R6 should generate IP SLA jitter probes to R5 across the serial link from R4 to R5. This flow simulates voice traffic. Next, configure R1 as an HTTP server and set SW2 to transfer the IOS image from R1, oversubscribing the serial link. Lastly, generate an ICMP traffic flow from SW1 to R5, marking it with IP precedence of 1 to feed packets into class-default.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
R1:
ip http server
ip http path flash:

R4:
interface Serial 0/1
 load-interval 30

R5:
rtr responder
!
interface Serial 0/0
 shutdown

R6:
ip sla monitor 1
 type jitter dest-ipaddr 155.1.45.5 dest-port 16384 codec g729a
 timeout 1000
 frequency 1
!
ip sla monitor schedule 1 life forever start-time now
```

```
Rack1SW2#copy http://admin:cisco@155.1.146.1/c2600-adventerprisek9-
mz.124-10.bin null:
```

```
Rack1SW1#ping
Protocol [ip]:
Target IP address: 155.1.45.5
Repeat count [5]: 100000000
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface:
Type of service [0]: 32
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 100000000, 100-byte ICMP Echos to 155.1.45.5, timeout is 2
seconds:
....
```

Verify the WRED statistics for class-default in your policy map. Note that no packets have been dropped due to the high maximum threshold (drop probability is inversely proportional to (max. thresh – min. thresh)). Also note matches for IP precedence 6 traffic, which corresponds to RIP updates that R4 sends to R5.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
Rack1R4#show policy-map interface serial 0/1
 Serial0/1

  Service-policy output: SERIAL_LINK

<snip>

    Class-map: class-default (match-any)
      2354 packets, 452176 bytes
      30 second offered rate 2000 bps, drop rate 0 bps
      Match: any
      Queueing
        Flow Based Fair Queueing
        Maximum Number of Hashed Queues 32
        (total queued/total drops/no-buffer drops) 1/0/0
         exponential weight: 9

  class    Transmitted      Random drop      Tail drop    Minimum Maximum  Mark
           pkts/bytes       pkts/bytes       pkts/bytes    thresh  thresh  prob
      0      72/6400           0/0              0/0           20      40  1/10
      1     657/68328          0/0              0/0            1      40  1/4
      2       0/0              0/0              0/0           24      40  1/10
      3       0/0              0/0              0/0           26      40  1/10
      4       0/0              0/0              0/0           28      40  1/10
      5       0/0              0/0              0/0           30      40  1/10
      6      53/22048          0/0              0/0           32      40  1/10
      7       0/0              0/0              0/0           34      40  1/10
   rsvp       0/0              0/0              0/0           36      40  1/10
```

Look at the CBWFQ queue contents, and note that now two flows have WRED enabled. One is the HTTP flow (user-configured) and another is the dynamic flow for ICMP traffic.

```
Rack1R4#show queueing interface serial 0/1
Interface Serial0/1 queueing strategy: fair
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 30580
  Queueing strategy: Class-based queueing
  Output queue: 4/512/32/30580 (size/max total/threshold/drops)
     Conversations  2/5/32 (active/max active/max total)
     Reserved Conversations 3/3 (allocated/max allocated)
     Available Bandwidth 78 kilobits/sec

  (depth/weight/total drops/no-buffer drops/interleaves) 1/0/0/0/0
  Conversation 40, linktype: ip, length: 64
  source: 155.1.146.6, destination: 155.1.45.5, id: 0x017D, ttl: 254,
  TOS: 184 prot: 17, source port 51901, destination port 16384

  (depth/weight/total drops/no-buffer drops/random/tail/interleaves)
2/194/274/0/0/0/0
  Conversation 41, linktype: ip, length: 396
  source: 155.1.146.1, destination: 155.1.58.8, id: 0xBCE1, ttl: 254,
  TOS: 64 prot: 6, source port 80, destination port 11004

  (depth/weight/total drops/no-buffer drops/random/tail/interleaves)
3/16192/0/0/0/0/0
  Conversation 21, linktype: ip, length: 104
  source: 155.1.67.7, destination: 155.1.45.5, id: 0x0BE8, ttl: 253, prot: 1
```

10.39 MQC WRED with ECN

- Modify the HTTP traffic class on R4 so that explicit congestion notification for TCP is used for RED dropping.

Configuration

```
R4:
policy-map SERIAL_LINK
 class HTTP
  random-detect ecn
  random-detect precedence 2   4     16    4
```

Verification

Note

TCP Explicit Congestion Notification (ECN), similar to how BECN and FECN work in Frame Relay, is used to signal the forthcoming of network congestion for TCP flows. Originally TCP detected network congestion based on packet loss, timeouts, and duplicate acknowledgements. This was usually the result of full queues and unconditional packet drops. TCP ECN allows the network to signal the receiver of the flow that the network is close to dropping packets. It's then up to the TCP receiver to decide how to react to this notification, which is usually that it signals the sender to slow down sending rate. The overall effect of TCP ECN is better performance, as compared to simple packet drops and slow start, as it allows the sender to respond fast than slow start would, and results in less time spent on the recovery from a packet loss.

TCP ECN works together with RED by changing the exceed action from random drop to ECN marking. Instead of randomly dropping a packet when the average queue depth grows above the minimum threshold, RED marks packet with the special ECN flag. This marking uses the two least-significant bits of the TOS bytes in the IP header. The Diff-Serv QoS model uses upper six bits of the TOS byte for DSCP/IP Precedence values. The lower two bits are known as ECN Capable Transport (ECT) and Congestion Experienced (CE). If a endpoint (sender or receiver) is ECN capable, it will set the ECT bit in packet headers to signal that it is capable to responding to congestion notifications. If RED is ECN enabled, and it's going to drop a packet randomly, the following actions apply.

First, ECN RED checks to see that either the ECT or CE bits are set. If both of the bits are zero, the sending or receiving endpoint is not ECN capable, and the packet is randomly dropped. If either of the bits are set (ECT or CE) the RED procedure does not drop the packet, but instead sets the other bit and transmits the packet with both bits set. If both ECT and CE bits are set, the packet is simply transmitted. If the queue is full, RED drops the packets irrespective of their markings.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

To verify this configuration, enable the TCP ECN feature on both R1 and R5. Enable the HTTP server service on R1, and configure R5 to transfer a file from R1. Shutdown the Frame Relay interface of R5 to ensure that packets take path across the serial link between R4 and R5.

```
R1:
ip http server
ip http path flash:
!
ip tcp ecn

R5:
ip tcp ecn
!
interface Serial 0/0
 shutdown
```

```
Rack1R5#copy http://admin:cisco@155.1.146.1/c2600-adventerprisek9-
mz.124-10.bin null:
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Check the statistics for the service policy. Note the long detailed output for the HTTP class. It demonstrates the number of random drops and ECN marks occurred in the queue for IP traffic marked with IP precedence of 2.

```
Rack1R4#show policy-map interface serial 0/1
 Serial0/1

  Service-policy output: SERIAL_LINK

<snip>

    Class-map: HTTP (match-all)
      13530 packets, 7740858 bytes
      5 minute offered rate 98000 bps, drop rate 0 bps
      Match: access-group name HTTP
      QoS Set
        precedence 2
          Packets marked 13530
      Queueing
        Output Queue: Conversation 41
        Bandwidth remaining 33 (%)
        (pkts matched/bytes matched) 11508/6580550
        (depth/total drops/no-buffer drops) 3/79/0
         exponential weight: 9
         explicit congestion notification
         mean queue depth: 4

  class    Transmitted      Random drop      Tail drop    Minimum Maximum  Mark
           pkts/bytes       pkts/bytes       pkts/bytes    thresh  thresh  prob
      0       0/0              0/0              0/0           20      40  1/10
      1       0/0              0/0              0/0           22      40  1/10
      2   10031/5714738       79/44285          0/0            4      16  1/4
      3       0/0              0/0              0/0           26      40  1/10
      4       0/0              0/0              0/0           28      40  1/10
      5       0/0              0/0              0/0           30      40  1/10
      6       0/0              0/0              0/0           32      40  1/10
      7       0/0              0/0              0/0           34      40  1/10
   rsvp       0/0              0/0              0/0           36      40  1/10


  class     ECN Mark
            pkts/bytes
      0        0/0
      1        0/0
      2      673/390340
      3        0/0
      4        0/0
      5        0/0
      6        0/0
      7        0/0
   rsvp        0/0
```

10.40 MQC Class-Based Generic Traffic Shaping

- Configure MQC shaping on R6 to limit the sending rate on its link to VLAN 146 to 512Kbps.
- The link to VLAN 67 should be limited to 384Kbps.
- Use a burst interval of 20ms.

Configuration

```
R6:
policy-map SHAPE_VLAN146
 class class-default
   shape average 384000 7680
!
policy-map SHAPE_VLAN67
 class class-default
   shape average 512000 10240
!
interface FastEthernet 0/0.146
  service-policy output SHAPE_VLAN146
!
interface FastEthernet 0/0.67
  service-policy output SHAPE_VLAN67
```

Verification

Note

MQC based traffic-shaping allows Generic Traffic Shaping to be applied to any traffic class. This makes traffic-shaping's configuration modular and unified, without special variants for encapsulations like Frame Relay. Class-based GTS is similar to legacy GTS, but allows shaping based on class-maps, as opposed to access-list classification with legacy GTS, thus allowing any MQC classification options, such as NBAR, to define shaped traffic. Additionally MQC based shaping allows the shaping queue to be tuned, while legacy GTS supports just simple WFQ with no option to tune it.

All shaping parameters and the metering model remains the same as with the legacy GTS approach. However, class-based shaping unifies the previously separated GTS and FRTS syntax, as we'll see in further tasks. Note one subtle difference – FRTS by default assumes B_e=0, while GTS by default assumes B_e=B_c.

Like legacy GTS, MQC based GTS limits the average rate based on the full packet size, including layer 2 overhead. Though this is not important for many applications, it may become of concern on low-speed interfaces with small layer 3 packet payloads.

With the command **shape average** under the policy-map class configuration, the CIR, B_c, B_e, and buffer limit are defined. The buffer limit is the maximum size of the default WFQ queue. Since this task did not mention anything about B_e, the default value is used. B_c can be calculated the same way as with legacy GTS, where B_c=CIR*T_c/1000. In our case, T_c is 20ms, therefore the B_c values are 7680 and 10240 bits.

In this scenario we used class-based shaping to limit the sub-interfaces sending rate. This is a common use of GTS, and the effect is that each sub-interface now uses its own software queue, where as by default, all sub-interfaces share the software queue of their main interface. This also allows the use of separate QoS policies per sub-interface, due to the ability to tune shaper's queue (see the next task for more information on this).

To verify this configuration generate three traffic flows across R6 to see the class-based GTS in action. Below this is accomplished with R6 sending IP SLA jitter probes to R4, R6 acting as an HTTP server for R4, and an ICMP flow being sent from SW1 to R4.

```
R4:
rtr responder

R6:
ip http server
ip http path flash:
!
! Configure SLA probe for jitter with 60 byte packets
!
ip sla monitor 1
 type jitter dest-ipaddr 155.1.146.4 dest-port 16384 codec g729a
 timeout 1000
 frequency 1
!
ip sla monitor schedule 1 life forever start-time now

Rack1R4#copy http://admin:cisco@155.1.146.6/c2600-adventerprisek9-
mz.124-10.bin null:

Rack1SW1#ping 155.1.146.4 repeat 100000

Type escape sequence to abort.
Sending 100000, 150-byte ICMP Echos to 155.1.146.4, timeout is 2
seconds:
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

Use the following show command to verify your configuration and check the basic traffic-shaping statistics. Note the time interval and B_c/B_e values. B_c is "Sustain bits/int" and B_e is "Excess bits/int". The "Byte Limit" equals to B_c+B_e in bytes – the maximum amount of data that the shaper can send during the interval.

```
Rack1R6#show policy-map interface fastEthernet 0/0.146
 FastEthernet0/0.146

  Service-policy output: SHAPE_VLAN146

    Class-map: class-default (match-any)
      65844 packets, 28003483 bytes
      5 minute offered rate 334000 bps, drop rate 0 bps
      Match: any
      Traffic Shaping
           Target/Average   Byte   Sustain   Excess    Interval  Increment
             Rate           Limit  bits/int  bits/int  (ms)      (bytes)
           384000/384000    1920   7680      7680      20        960

        Adapt  Queue     Packets   Bytes     Packets   Bytes     Shaping
        Active Depth                         Delayed   Delayed   Active
        -      6         65838     27998907  56930     27085058  yes
```

Next view the shaping queue contents. Note that the queue appears to be WFQ, while in reality it is CBWFQ (you can tune it). The IOS computed the WFQ parameters based on the configured shaping rate (384Kbps), and ended up with 32 flow queues and a CDT value of 64. IOS takes the queue depth as the buffer limit configured in **shape** command.

One substantial difference from per-interface CBWFQ is that shaper queue has no notion of Link Queues – control plane traffic uses the same dynamic flows and WFQ computes weights based on its IP precedence values.

Note that all three traffic flows have equal weights in the CBWFQ, but different average packet sizes.

```
Rack1R6#show traffic-shape queue
<snip>

Traffic queued in shaping queue on FastEthernet0/0.146
 Traffic shape class: class-default
  Queueing strategy: weighted fair
  Queueing Stats: 5/1000/64/0 (size/max total/threshold/drops)
     Conversations  3/4/32 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 384 kilobits/sec

  (depth/weight/total drops/no-buffer drops/interleaves) 1/32384/0/0/0
  Conversation 18, linktype: ip, length: 78
  source: 155.1.146.6, destination: 155.1.146.4, id: 0x0151, ttl: 255,
  TOS: 0 prot: 17, source port 50359, destination port 16384

  (depth/weight/total drops/no-buffer drops/interleaves) 1/32384/0/0/0
  Conversation 25, linktype: ip, length: 118
  source: 155.1.67.7, destination: 155.1.146.4, id: 0x8C4B, ttl: 254, prot: 1

  (depth/weight/total drops/no-buffer drops/interleaves) 3/32384/0/0/0
  Conversation 2, linktype: ip, length: 1518
  source: 155.1.146.6, destination: 155.1.146.4, id: 0x8F7E, ttl: 255,
  TOS: 0 prot: 6, source port 80, destination port 55310
```

Based on this we can see that MQC based GTS performs the same as legacy GTS, but has the powerful classification options available via the MQC.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.41 MQC Class-Based GTS and CBWFQ

- Change the queue for the shaper applied to R6's connection to VLAN 146 as follows.
    - Provide 32Kbps of priority treatment for small packets with a layer three size of 60 bytes (voice packets) using a 400 byte burst value
    - Guarantee the HTTP traffic 256Kbps of the shaper's bandwidth
    - Unclassified traffic should receive fair-queue treatment

Configuration

```
R6:
!
! Define the class for voice traffic
!
class-map VOICE
 match packet length min 60 max 60
!
! Access-list to match HTTP traffic
!
ip access-list extended HTTP
 permit tcp any eq 80 any
!
! Class-map to match HTTP traffic
!
class-map HTTP
 match access-group name HTTP
!
! This policy map is the child policy
! to the shaping policy-map
!
policy-map CBWFQ
 class VOICE
   priority 32 4000
 class HTTP
   bandwidth 256
 class class-default
   fair-queue
!
! Nest the above-configured service-policy
!
policy-map SHAPE_VLAN146
  class class-default
    shape average 384000 7680
    service-policy CBWFQ
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Verification

Note

The ability to configure the shaper's queue is one of the most powerful features of class-based shaping. Previously, GTS was able to use only WFQ, and the only parameter you could change was the WFQ buffer limit.

Using MQC syntax you can "nest" another service policy inside a class configured for shaping. This nested service policy defines "sub-classes" and their respective parameters, including CBWFQ settings. The most prominent feature is the ability to configure an LLQ conversation, allowing optimal voice traffic handling when limiting traffic rate on sub-interfaces.

When you nest a CBWFQ configuration inside a shaper, the amount of bandwidth you can allocate to nested classes equals to the shaper average rate (384Kbps in this scenario). The default 75% rule does not work for available bandwidth here. In addition, the link queues are no longer in play, since the queue does not apply to a physical interface. Therefore, you may want to define a separate class for control plane traffic and guarantee it some bandwidth.

As mentioned in the previous task, the common use of shaping is limiting sub-interfaces' sending rates. By nesting a CBWFQ policy, you can define flexible, per sub-interface queuing policies, comparable to the features you have for VC-based technologies such as ATM and Frame-Relay.

Note the priority queue bandwidth of 32Kbps. We get this bandwidth from the assumption of VoIP packet sizes of 60 bytes (G.729), plus 18 bytes overhead for an Ethernet header with 4 bytes of VLAN tag at 50 packets per second = (60+18)*50*8=31200bps. Remember that priority bandwidth must take layer 2 overhead in account.

To verify the configuration, configure three traffic sources. R6 should generate IP SLA jitter probes and R4 should respond. Set this probe to time out in 5 seconds, and use the ToS byte 224 (IP Precedence 7) for signaling. This is to provide that the IP SLA control connection has a weight high enough not to starve in shaper queue. In addition to that, R6 should act as a web-server and R4 should transfer a large file from it. Lastly, source an ICMP packet stream from SW1 to R4.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
R4:
rtr responder

R6:
ip http server
ip http path flash:
!
! Configure SLA probe for jitter with 60 byte packets
! Note the ToS byte corresponding to IP Pecedence of 7
!
ip sla monitor 1
 type jitter dest-ipaddr 155.1.146.4 dest-port 16384 codec g729a
 frequency 1
 timeout 1000
 tos 224
!
ip sla monitor schedule 1 life forever start-time now
```

Rack1R4#copy http://admin:cisco@155.1.146.6/c2600-adventerprisek9-mz.124-10.bin null:

Rack1SW1#ping 155.1.146.4 repeat 100000

```
Type escape sequence to abort.
Sending 100000, 150-byte ICMP Echos to 155.1.146.4, timeout is 2
seconds:
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

Check the statistics for policy map matches. Note that the voice packet rate is far below its average of 32Kbps. This is because the system classifies the IP SLA control connection using dynamic queues, based on IP Precedence of seven. However, even this is not enough to guarantee timely communication, as we will see later.

```
Rack1R6#show policy-map interface fastEthernet 0/0.146
 FastEthernet0/0.146

  Service-policy output: SHAPE_VLAN146

    Class-map: class-default (match-any)
      269715 packets, 88958085 bytes
      30 second offered rate 384000 bps, drop rate 0 bps
      Match: any
      Traffic Shaping
           Target/Average   Byte   Sustain   Excess    Interval  Increment
             Rate           Limit  bits/int  bits/int  (ms)      (bytes)
           384000/384000    1920   7680      7680      20        960

        Adapt  Queue     Packets   Bytes     Packets   Bytes     Shaping
        Active Depth                         Delayed   Delayed   Active
        -      7         269209    88909805  141137    75436903  yes

      Service-policy : CBWFQ

        Class-map: VOICE (match-all)
          7018 packets, 547404 bytes
          30 second offered rate 3000 bps, drop rate 0 bps
          Match: packet length min 60 max 60
          Queueing
            Strict Priority
            Output Queue: Conversation 40
            Bandwidth 32 (kbps) Burst 4000 (Bytes)
            (pkts matched/bytes matched) 4001/312078
            (total drops/bytes drops) 486/37908

        Class-map: HTTP (match-all)
          13966 packets, 20215902 bytes
          30 second offered rate 380000 bps, drop rate 0 bps
          Match: access-group name HTTP
          Queueing
            Output Queue: Conversation 41
            Bandwidth 256 (kbps)Max Threshold 64 (packets)
            (pkts matched/bytes matched) 13916/20175173
        (depth/total drops/no-buffer drops) 3/0/0

        Class-map: class-default (match-any)
          2727 packets, 327338 bytes
          30 second offered rate 1000 bps, drop rate 0 bps
          Match: any
          Queueing
            Flow Based Fair Queueing
            Maximum Number of Hashed Queues 32
        (total queued/total drops/no-buffer drops) 4/0/0
```

Copyright © 2009 Internetwork Expert

Look at the traffic-shaper queue contents. Note the HTTP flows (source port 80) and its weight of 96. The next flow with weight 4048 corresponds to the IP SLA control channel. It has a low chance of competing with the greedy HTTP flow, and thus the scheduler constantly delays it. The next flow corresponds to RIP routing updates. Note that CBWFQ treats them as dynamic flows, not as the link conversation. The scheduler calculates the weight for this flow based strictly on the IP Precedence of 6 – 4626. The last flow corresponds to the stream of ICMP packets from SW1 to R4. This flow hardly has any chances to get there.

Note that ICMP packets have the size 118 in the queue. This is because their original layer 3 size was 100 bytes, and 18 bytes was added as Ethernet header overhead – 14 bytes for frame-header and 4 bytes for VLAN tag.

```
Rack1R6#show traffic-shape queue
<snip>

Traffic queued in shaping queue on FastEthernet0/0.146
 Traffic shape class: class-default
  Queueing strategy: weighted fair
  Queueing Stats: 8/1000/64/308 (size/max total/threshold/drops)
     Conversations  4/6/32 (active/max active/max total)
     Reserved Conversations 1/1 (allocated/max allocated)
     Available Bandwidth 102 kilobits/sec

  (depth/weight/total drops/no-buffer drops/interleaves) 3/96/0/0/0
  Conversation 41, linktype: ip, length: 1266
  source: 155.1.146.6, destination: 155.1.146.4, id: 0x915E, ttl: 255,
  TOS: 0 prot: 6, source port 80, destination port 48053

  (depth/weight/total drops/no-buffer drops/interleaves) 3/4048/0/0/0
  Conversation 29, linktype: ip, length: 98
  source: 155.1.146.6, destination: 155.1.146.4, id: 0x0000, ttl: 255,
  TOS: 224 prot: 17, source port 57870, destination port 1967

  (depth/weight/total drops/no-buffer drops/interleaves) 1/4626/0/0/0
  Conversation 2, linktype: ip, length: 230
  source: 155.1.146.6, destination: 224.0.0.9, id: 0x0000, ttl: 2,
  TOS: 192 prot: 17, source port 520, destination port 520

  (depth/weight/total drops/no-buffer drops/interleaves) 1/32384/0/0/0
  Conversation 25, linktype: ip, length: 118
  source: 155.1.67.7, destination: 155.1.146.4, id: 0x8741, ttl: 254, prot: 1
```

Now check the IP SLA probe statistics. Note the high number of probe failures. To improve this, we are going to give a special flow to the IP SLA control signaling. From the queue contents dumped above, you may note that it uses UDP destination port of 1967.

```
Rack1R6#show ip sla monitor statistics 1
Round trip time (RTT)   Index 1
        Latest RTT: 18 ms
Latest operation start time: 07:41:16.514 UTC Fri Aug 15 2008
Latest operation return code: OK
RTT Values
        Number Of RTT: 873
        RTT Min/Avg/Max: 2/18/94 ms
Latency one-way time milliseconds
        Number of one-way Samples: 0
        Source to Destination one way Min/Avg/Max: 0/0/0 ms
        Destination to Source one way Min/Avg/Max: 0/0/0 ms
Jitter time milliseconds
        Number of Jitter Samples: 745
        Source to Destination Jitter Min/Avg/Max: 1/14/83 ms
        Destination to Source Jitter Min/Avg/Max: 1/1/2 ms
Packet Loss Values
        Loss Source to Destination: 127         Loss Destination to Source: 0
        Out Of Sequence: 0      Tail Drop: 0    Packet Late Arrival: 0
Voice Score Values
        Calculated Planning Impairment Factor (ICPIF): 38
MOS score: 2.85
Number of successes: 81
Number of failures: 31
Operation time to live: Forever
```

Configure a special class to take care of the IP SLA signaling. Assign this class a bandwidth value of 24Kbps in the CBWFQ policy map.

```
R6:
ip access-list extended SLA_SIGNALING
 permit udp any any eq 1967
!
class-map SLA_SIGNALING
 match access-gorup name SLA_SIGNALING
!
policy-map CBWFQ
  class SLA_SIGNALING
   bandwidth 24
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Look at the policy-map statistics now. Note the offered rate for the Voice packets (the load-interval has been set to 30 seconds for this output), and that HTTP traffic claimed almost all the bandwidth available after the VOICE traffic.

```
Rack1R6#show policy-map interface fastEthernet 0/0.146
 FastEthernet0/0.146

  Service-policy output: SHAPE_VLAN146

    Class-map: class-default (match-any)
      10148 packets, 6778660 bytes
      30 second offered rate 382000 bps, drop rate 0 bps
      Match: any
      Traffic Shaping
           Target/Average   Byte   Sustain   Excess    Interval  Increment
             Rate           Limit  bits/int  bits/int  (ms)      (bytes)
           384000/384000    1920   7680      7680      20        960

        Adapt  Queue     Packets   Bytes     Packets   Bytes     Shaping
        Active Depth                         Delayed   Delayed   Active
        -      7         10151     6779014   10144     6776812   yes

      Service-policy : CBWFQ

        Class-map: VOICE (match-all)
          5716 packets, 445848 bytes
          30 second offered rate 30000 bps, drop rate 0 bps
          Match: packet length min 60 max 60
          Queueing
            Strict Priority
            Output Queue: Conversation 40
            Bandwidth 32 (kbps) Burst 4000 (Bytes)
            (pkts matched/bytes matched) 5712/445536
            (total drops/bytes drops) 0/0

        Class-map: HTTP (match-all)
          4350 packets, 6321484 bytes
          30 second offered rate 352000 bps, drop rate 0 bps
          Match: access-group name HTTP
          Queueing
            Output Queue: Conversation 41
            Bandwidth 256 (kbps)Max Threshold 64 (packets)
            (pkts matched/bytes matched) 4348/6319712
        (depth/total drops/no-buffer drops) 3/0/0

        Class-map: SLA_CONTROL (match-all)
          7 packets, 686 bytes
          30 second offered rate 0 bps, drop rate 0 bps
          Match: access-group name SLA_CONTROL
          Queueing
            Output Queue: Conversation 42
            Bandwidth 24 (kbps)Max Threshold 64 (packets)
            (pkts matched/bytes matched) 7/686
        (depth/total drops/no-buffer drops) 0/0/0

        Class-map: class-default (match-any)
          75 packets, 10642 bytes
          30 second offered rate 0 bps, drop rate 0 bps
          Match: any
          Queueing
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
            Flow Based Fair Queueing
            Maximum Number of Hashed Queues 32
         (total queued/total drops/no-buffer drops) 2/0/0
```

Now check the IP SLA statistics. Note that the number of failures is zero now, and the average RTT is within the range of shaper's Tc interval (20ms).

```
Rack1R6#show ip sla monitor statistics 1
Round trip time (RTT)   Index 1
        Latest RTT: 22 ms
Latest operation start time: 00:47:16.013 UTC Tue Aug 19 2008
Latest operation return code: OK
RTT Values
        Number Of RTT: 1000
        RTT Min/Avg/Max: 7/22/41 ms
Latency one-way time milliseconds
        Number of one-way Samples: 0
        Source to Destination one way Min/Avg/Max: 0/0/0 ms
        Destination to Source one way Min/Avg/Max: 0/0/0 ms
Jitter time milliseconds
        Number of Jitter Samples: 999
        Source to Destination Jitter Min/Avg/Max: 1/11/30 ms
        Destination to Source Jitter Min/Avg/Max: 1/1/2 ms
Packet Loss Values
        Loss Source to Destination: 0           Loss Destination to Source: 0
        Out Of Sequence: 0      Tail Drop: 0    Packet Late Arrival: 0
Voice Score Values
        Calculated Planning Impairment Factor (ICPIF): 11
MOS score: 4.06
Number of successes: 9
Number of failures: 0
Operation time to live: Forever
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.42 MQC Single-Rate Three-Color Policer

- Configure R4 to meter incoming HTTP traffic on its link to VLAN 146 as follows.
  - If the traffic rate is less than 128Kbps, mark the packets with an IP precedence of one
  - If the traffic exceeds 128Kbps, mark the packets with an IP precedence of zero
  - Drop violating traffic
- Ensure the burst size is large enough to accommodate normal and excess burst durations of 200ms and 300ms at a rate of 128Kbps.

Configuration

```
R4:
ip access-list extended HTTP
 permit tcp any eq 80 any
!
class-map HTTP
 match access-group name HTTP
!
policy-map POLICE_VLAN146
 class HTTP
   police 128000 3200 4800
    conform-action set-prec-transmit 1
    exceed-action set-prec-transmit 0
    violate-action drop
!
interface FastEthernet 0/1
  service-policy input POLICE_VLAN146
```

Copyright © 2009 Internetwork Expert

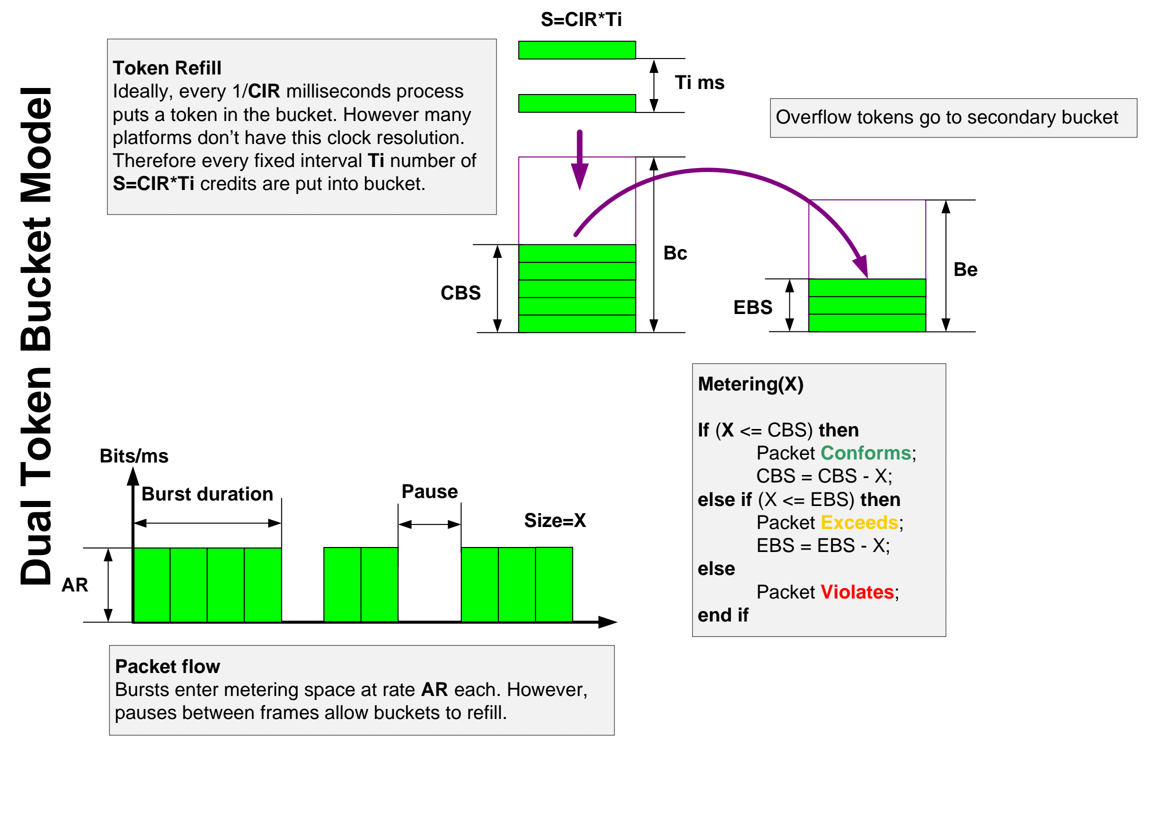
Verification

Note

A Single Rate Three Color Marker (Policer) or “srTCM” is the RFC-based implementation of the metering process. Compared to legacy CAR the following has changed.

First, the concept of Excess Burst (B_e) is no longer tied to a pseudorandom marking behavior. Rather, B_e is used as extra credit for periods of inactivity. Secondly, the policing process uses a Dual Token Bucket model, but the implementation is different. “Three color” in this case means that incoming traffic is metered against CIR using the configured burst sizes, and the result is a marking using either of three colors: Green (Conform), Yellow (Exceed), Red (Violate).

As usual, the token bucket exhibits the concept of a sliding metering window over the packet flow. However, this time process uses a secondary, Excess Burst bucket.



The Excess Burst bucket accumulates extra credit, spilled over from the Normal Bucket when it overfills. This only happens during long periods of pause that exceeds the averaging interval T_c. With just a single bucket, the process does not account for “extended” periods of silence. The Excess Burst bucket allows accumulating credits and using them when incoming burst exceeds the configured B_c value. Thus, B_e in this usage serves the similar purpose that the B_e functionality has with traffic-shaping. In fact, these two are complementary in the sense that srTCM’s B_e properly “recognizes” the excessive bursting of traffic shaping.

In a classic token bucket model, a special timer fires every T_i interval, triggering the token refresh procedure. This method, however, limits the “resolution” of metering. At the same time, the amount of CPU operations needed to meter a packet is small. This is because the procedure uses only addition and subtraction operations, which consume little CPU cycles.

Cisco’s implementation of the token bucket used for srTCM uses a different algorithm. This algorithm has “unlimited” precision, but consumes more CPU cycles per-packet. The process keeps track of three variables:

CBS – current normal burst size
EBS – current excess burst size
T0 – last packet arrival time.

Now imagine that a new packet of size “S” arrives at time “T”.

Step 1:
The process computes accumulated credit:

Cr = CIR*(T-T0)

then adds this values to CBS:

```
if ((CBS + Cr) <= Bc) then
      CBS = CBS + Cr;
else
      CBS = Bc;
end if
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Step 2:
The process computes a new EBS size if there are enough credits:

```
If (CBS + Cr > Bc) then
      EBS = CBS + Cr – Bc;
end

if (EBS > Be) then
      EBS = Be;
```

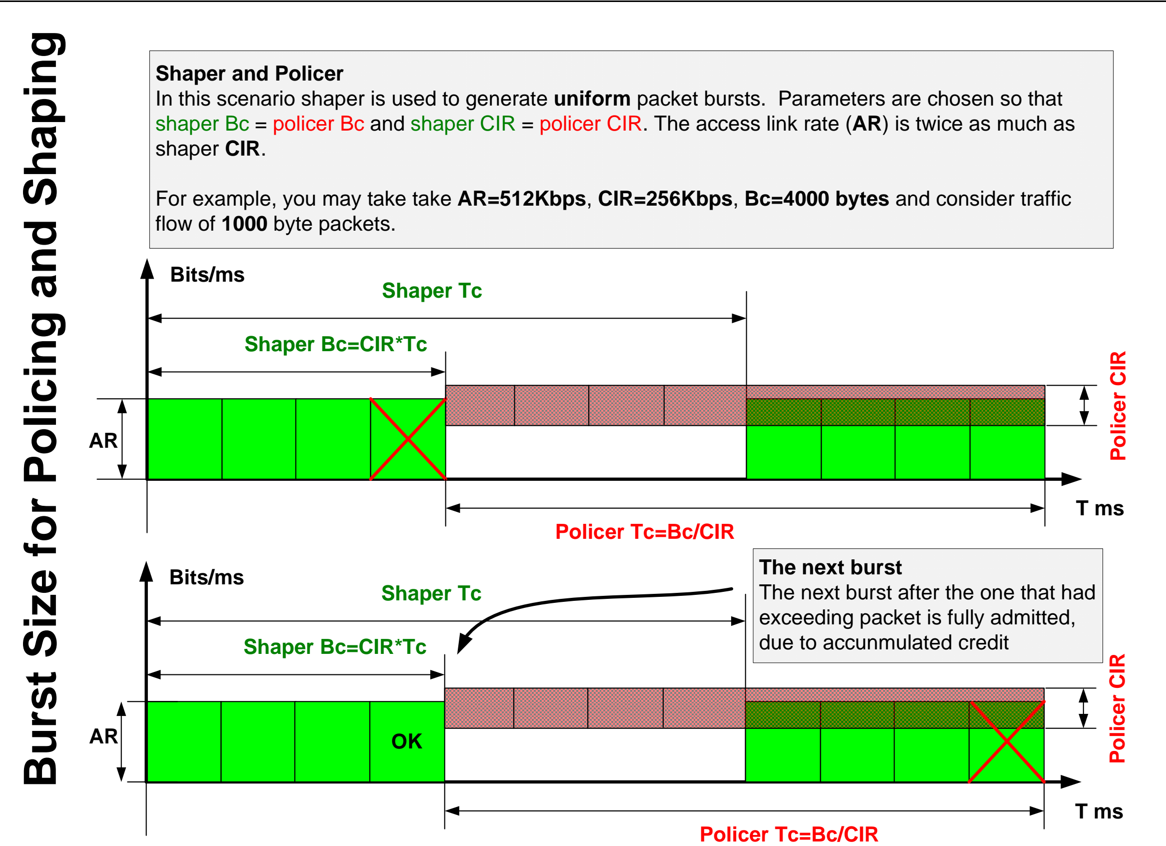
Step 3:
The process compares packet size to accumulated credits and performs marking:

```
if ( S < CBS ) then
    Packet Conforms;
    CBS = CBS – S;
else if (S < EBS) then
    Packet Exceeds;
    EBS = EBS – S;
else
    Packet Violates;
end if
```

This implementation does not use any hardware timer, and thus the "precision" is not limited. However, additional multiplication operations may become CPU consuming on intense packet flows.

For an optimal burst size selection, if you are using srTCM to perform traffic admission at the network edge, ensure your policing burst sizes match at least one packet size more than the burst size configured at the customer side. This is needed because now the router meters traffic with extra precision, and the effect illustrated on the next figure (shaper Bc = policer Bc) may occur.

Copyright © 2009 Internetwork Expert



Note that the policer may mark the beginning of the second burst as exceeding. This is because the policer B_c exactly equals to shaper B_c. In this situation when the sliding window hits the beginning of the next burst it has no capacity to accommodate another packet. The packet marked as exceeding does not count for the sliding window content, and thus the next burst will have all four packets admitted. For this ideal simulation, the policer B_c should be larger than traffic burst size by at least one packet. This brings us to the formula:

Policer B_c = Shaper Burst Size + Average Packet Size

In addition to that, it makes sense to configure burst size proportional to the average packet size as well, to ensure more precise matching. However, this is not necessary, especially with larger bursts.

Note that the above effect was insignificant with CAR, due to the “lower” resolution feature. Next, if you are using policing to limit the traffic rate, pick up the burst sizes to reduce the negative impact on TCP traffic. This has been discussed in the section dedicated to rate limiting with CAR.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

In this particular example the burst sizes are based on the time intervals as follows:

Bc = 128000*0,2/8=3200 bytes
Be = 128000*0,3/8=4800 bytes

To test the policer simulate an HTTP traffic flow from R1 to R4. At the same time, pre-shape the traffic rate from R1 to R4 using GTS. Set the shaper's burst size to 12800 bits (1600 bytes), which is half of the policer's burst.

```
R1:
ip http server
ip http path flash:
!
interface FastEthernet 0/0
  traffic-shape rate 128000 12800 0

R4:
interface FastEthernet 0/1
  load-interval 30
```

Rack1R4#copy http://admin:cisco@155.1.146.1/c2600-adventerprisek9-mz.124-10.bin null:

Check the policy-map statistics.

```
Rack1R4#show policy-map interface fastEthernet 0/1
 FastEthernet0/1

  Service-policy input: POLICE_VLAN146

    Class-map: HTTP (match-all)
      1245 packets, 1786990 bytes
      30 second offered rate 127000 bps, drop rate 0 bps
      Match: access-group name HTTP
      police:
          cir 128000 bps, bc 3200 bytes, be 4800 bytes
        conformed 1241 packets, 1780934 bytes; actions:
          set-prec-transmit 1
        exceeded 4 packets, 6056 bytes; actions:
          set-prec-transmit 0
        violated 0 packets, 0 bytes; actions:
          drop
        conformed 127000 bps, exceed 0 bps, violate 0 bps

    Class-map: class-default (match-any)
      124 packets, 11184 bytes
      30 second offered rate 0 bps, drop rate 0 bps
      Match: any
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Note the small number of exceeding packets. Now change the shaper burst size to be equal to the burst size of the policer (25600 bits) and repeat the metering on the same file transfer.

```
R1:
interface FastEthernet 0/0
  traffic-shape rate 128000 25600 0

Rack1R4#show policy-map interface fastEthernet 0/1
 FastEthernet0/1

  Service-policy input: POLICE_VLAN146

    Class-map: HTTP (match-all)
      2510 packets, 2949429 bytes
      30 second offered rate 127000 bps, drop rate 0 bps
      Match: access-group name HTTP
      police:
          cir 128000 bps, bc 3200 bytes, be 4800 bytes
        conformed 2254 packets, 2575997 bytes; actions:
          set-prec-transmit 1
        exceeded 256 packets, 373432 bytes; actions:
          set-prec-transmit 0
        violated 0 packets, 0 bytes; actions:
          drop
        conformed 111000 bps, exceed 16000 bps, violate 0 bps

    Class-map: class-default (match-any)
      108 packets, 16776 bytes
      30 second offered rate 0 bps, drop rate 0 bps
      Match: any
```

The amount of packets incorrectly marked as exceeding is around 10%, even though shaper burst equals the policer burst. This is the demonstration of the effect described above.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.43 MQC Hierarchical Policers

- Configure R4 to limit the aggregate rate of HTTP traffic entering the connection to VLAN 146 to 128Kbps.
- Transmit conforming packets, mark exceeding packets with an IP precedence of zero, and drop violating packets.
- For HTTP traffic flows from R1 and R6, limit the rate to 64Kbps for each flow.
- For these second-level policers, set the conform action to set the IP precedence to 1 and transmit, the exceed action to set the IP precedence to 0 and transmit, and the violate action to set the IP precedence to 0 and transmit.

Configuration

```
R4:
!
! All HTTP traffic
!
ip access-list extended HTTP
 permit tcp any eq 80 any
!
class-map HTTP
 match access-group name HTTP

!
! Traffic from R1 and R6 respectively
!
ip access-list extended FROM_R1
 permit ip host 155.1.146.1 any
!
ip access-list extended FROM_R6
 permit ip host 155.1.146.6 any
!
!
!
class-map FROM_R1
 match access-group name FROM_R1
!
class-map FROM_R6
 match access-group name FROM_R6

!
! Nested policers (subrate)
!
policy-map SUBRATE_POLICER
 class FROM_R1
  police 64000 3200 4800
   conform-action set-prec-transmit 1
   exceed-action set-prec-transmit 0
   violate-action set-prec-transmit 0
 class FROM_R6
  police 64000 3200 4800
```

```
   conform-action set-prec-transmit 1
   exceed-action set-prec-transmit 0
   violate-action set-prec-transmit 0

!
! Policer configuration using MQC syntax.
!
policy-map POLICE_VLAN146
 class HTTP
   police 128000 3200 4800
    conform-action transmit
    exceed-action set-prec-transmit 0
    violate-action drop
   service-policy SUBRATE_POLICER
!
interface FastEthernet 0/1
  service-policy input POLICE_VLAN146
```

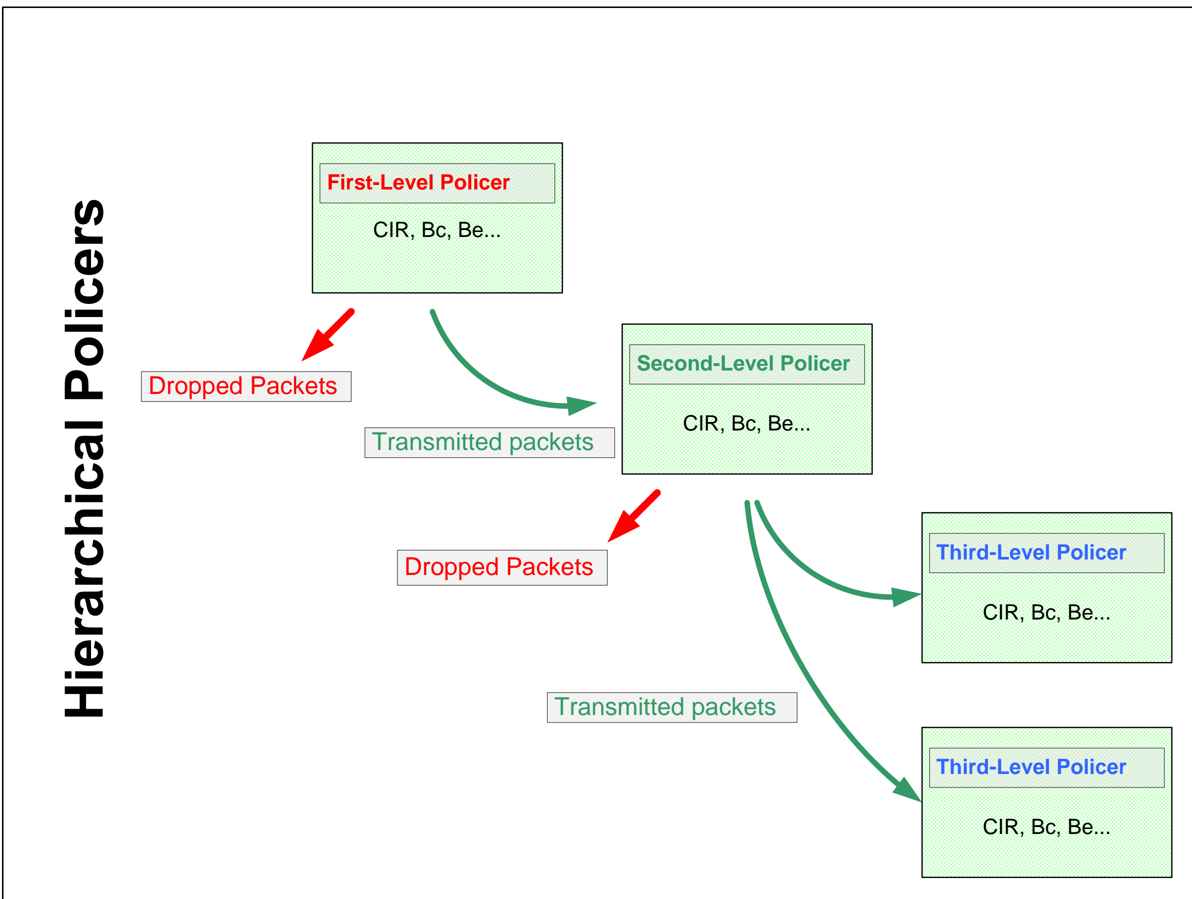
Verification

Note

MQC allows the nesting of up to three levels of policing. The syntax uses nested service-policies, where the parent policy defines the aggregate rate, and the child policies define the sub-rates for each class. The effect is similar to the cascaded rate-limit statements used in legacy CAR, however, note the following differences.

First, with MQC hierarchical policing, the upper level policers are applied first before the nested policers. With CAR, you usually configure the “aggregate” statement in the end of the other “sub-rate” statements. Secondly, every policer could be a single rate or two-rate policer. Furthermore, the MQC allows for a more natural and intuitive grouping of classes and policing statements.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com



Start verification by generating a single traffic flow from R1 down to SW2 across R4. To accomplish this, you need to shut down the Frame-Relay interface of R5. The general idea is that link between R4 and R5 is set to 128Kbps and the traffic flows will eventually oversubscribe it. The task configuration ensures that aggregate traffic across the serial link never exceeds 128Kbps, but allows every flow to send traffic above the guaranteed 64Kbps. Generate an HTTP traffic flow from R1 to SW2 and pre-shape it down to 128Kbps on R6. Ensure the traffic shaper burst smaller than the configured policer burst by at least one average packet size to avoid the mis-marking behavior as seen in the previous task.

Accessed by p_saffari@yahoo.com from 94.182.214.117 at 04:54:22 Oct 21, 2009

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
R1:
ip http server
ip http path flash:
!
! Bc = 12800 bits = 1600 bytes
!
interface FastEthernet 0/0
  traffic-shape rate 128000 12800 0

R4:
interface FastEthernet 0/1
  load-interval 30

R5:
interface Serial 0/0
 shutdown
```

Rack1SW2#copy http://admin:cisco@155.1.146.1/c2600-adventerprisek9-mz.124-10.bin null:

Note that traffic from R1 to SW2 exceeds the nested policer rate and thus the second-level policer marks almost half of the traffic with IP Precedence 0.

```
Rack1R4#show policy-map interface fastEthernet 0/1
 FastEthernet0/1

  Service-policy input: POLICE_VLAN146

    Class-map: HTTP (match-all)
      2428 packets, 1432520 bytes
      30 second offered rate 126000 bps, drop rate 0 bps
      Match: access-group name HTTP
      police:
          cir 128000 bps, bc 3200 bytes, be 4800 bytes
        conformed 2428 packets, 1432520 bytes; actions:
          transmit
        exceeded 0 packets, 0 bytes; actions:
          set-prec-transmit 0
        violated 0 packets, 0 bytes; actions:
          drop
        conformed 123000 bps, exceed 0 bps, violate 0 bps

      Service-policy : SUBRATE_POLICER

        Class-map: FROM_R1 (match-all)
          2428 packets, 1432520 bytes
          30 second offered rate 126000 bps, drop rate 0 bps
          Match: access-group name FROM_R1
          police:
              cir 64000 bps, bc 3200 bytes, be 4800 bytes
            conformed 1231 packets, 726290 bytes; actions:
              set-prec-transmit 1
            exceeded 0 packets, 0 bytes; actions:
              set-prec-transmit 0
            violated 1197 packets, 706230 bytes; actions:
              set-prec-transmit 0
            conformed 63000 bps, exceed 0 bps, violate 62000 bps
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
        Class-map: FROM_R6 (match-all)
          0 packets, 0 bytes
          30 second offered rate 0 bps, drop rate 0 bps
          Match: access-group name FROM_R6
          police:
              cir 64000 bps, bc 3200 bytes, be 4800 bytes
            conformed 0 packets, 0 bytes; actions:
              set-prec-transmit 1
            exceeded 0 packets, 0 bytes; actions:
              set-prec-transmit 0
            violated 0 packets, 0 bytes; actions:
              set-prec-transmit 0
            conformed 0 bps, exceed 0 bps, violate 0 bps

        Class-map: class-default (match-any)
          0 packets, 0 bytes
          30 second offered rate 0 bps, drop rate 0 bps
          Match: any

    Class-map: class-default (match-any)
      10 packets, 860 bytes
      30 second offered rate 0 bps, drop rate 0 bps
      Match: any
```

Now add another traffic flow, from R6 down to SW4 and across the serial link between R4 and R5.

```
R6:
ip http server
ip http path flash:
!
interface FastEthernet 0/0.146
  traffic-shape rate 128000 12800 0

Rack1SW4#copy http://admin:cisco@155.1.146.6/c2600-adventerprisek9-mz.124-
10.bin null:
```

Verify the policer statistics once again. Note that now both second-level policers meter input rate close to 64Kbps with no violating traffic.

```
Rack1R4#show policy-map interface fastEthernet 0/1
 FastEthernet0/1

  Service-policy input: POLICE_VLAN146

    Class-map: HTTP (match-all)
      11514 packets, 6792149 bytes
      30 second offered rate 126000 bps, drop rate 0 bps
      Match: access-group name HTTP
      police:
          cir 128000 bps, bc 3200 bytes, be 4800 bytes
        conformed 11513 packets, 6791559 bytes; actions:
          transmit
        exceeded 2 packets, 1180 bytes; actions:
          set-prec-transmit 0
        violated 0 packets, 0 bytes; actions:
          drop
        conformed 126000 bps, exceed 0 bps, violate 0 bps
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
  Service-policy : SUBRATE_POLICER

    Class-map: FROM_R1 (match-all)
      10295 packets, 6074050 bytes
      30 second offered rate 63000 bps, drop rate 0 bps
      Match: access-group name FROM_R1
      police:
          cir 64000 bps, bc 3200 bytes, be 4800 bytes
        conformed 5817 packets, 3432030 bytes; actions:
          set-prec-transmit 1
        exceeded 0 packets, 0 bytes; actions:
          set-prec-transmit 0
        violated 4479 packets, 2642610 bytes; actions:
          set-prec-transmit 0
        conformed 63000 bps, exceed 0 bps, violate 0 bps

    Class-map: FROM_R6 (match-all)
      1219 packets, 718099 bytes
      30 second offered rate 63000 bps, drop rate 0 bps
      Match: access-group name FROM_R6
      police:
          cir 64000 bps, bc 3200 bytes, be 4800 bytes
        conformed 1216 packets, 716329 bytes; actions:
          set-prec-transmit 1
        exceeded 3 packets, 1770 bytes; actions:
          set-prec-transmit 0
        violated 0 packets, 0 bytes; actions:
          set-prec-transmit 0
        conformed 63000 bps, exceed 0 bps, violate 0 bps

    Class-map: class-default (match-any)
      0 packets, 0 bytes
      30 second offered rate 0 bps, drop rate 0 bps
      Match: any

Class-map: class-default (match-any)
  68 packets, 8728 bytes
  30 second offered rate 0 bps, drop rate 0 bps
  Match: any
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

This is due to the fact the Serial interface runs WFQ, and shares bandwidth equally between two flows.

```
Rack1R4#show interfaces serial 0/1
Serial0/1 is up, line protocol is up
  Hardware is PowerQUICC Serial
  Internet address is 155.1.45.4/24
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
     reliability 255/255, txload 20/255, rxload 2/255
  Encapsulation HDLC, loopback not set
  Keepalive set (10 sec)
  Last input 00:00:01, output 00:00:00, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: weighted fair
  Output queue: 12/1000/64/0 (size/max total/threshold/drops)
     Conversations  2/3/256 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 1158 kilobits/sec
  5 minute input rate 18000 bits/sec, 53 packets/sec
  5 minute output rate 124000 bits/sec, 27 packets/sec
     37685 packets input, 2352426 bytes, 0 no buffer
     Received 785 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     18965 packets output, 10363861 bytes, 0 underruns
     0 output errors, 0 collisions, 3 interface resets
     0 unknown protocol drops
     0 output buffer failures, 0 output buffers swapped out
     6 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
```

Check the serial interface queue contents.

```
Rack1R4#show queueing interface serial 0/1
Interface Serial0/1 queueing strategy: fair
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: weighted fair
  Output queue: 12/1000/64/0 (size/max total/threshold/drops)
     Conversations  2/3/256 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 1158 kilobits/sec

  (depth/weight/total drops/no-buffer drops/interleaves) 6/16192/0/0/0
  Conversation 134, linktype: ip, length: 580
  source: 155.1.146.1, destination: 155.1.58.8, id: 0xEB41, ttl: 254,
  TOS: 32 prot: 6, source port 80, destination port 11001

  (depth/weight/total drops/no-buffer drops/interleaves) 6/16192/0/0/0
  Conversation 192, linktype: ip, length: 580
  source: 155.1.146.6, destination: 155.1.108.10, id: 0x70CA, ttl: 254,
  TOS: 32 prot: 6, source port 80, destination port 11002
```

Note that both flow queues have the same weights and same queue depth. This is why both flows eventually end up with the same rate.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.44 MQC Two-Rate Three-Color Policer

- Modify the first-level policy-map applied directly to R4's connection to VLAN 146 to remove the aggregate policer of 128Kbps.
- Modify the second-level policy-map to replace the single-rate policers with two-rate policers.
- Use the CIR value of 64Kbps and PIR value of 128Kbps.
- Use the values of CIR*400ms and PIR*200ms for normal and excess burst sizes.
- Set the conform action to set IP precedence 1 and transmit, the exceed action to set IP precedence of 0 and transmit, and the violate action to drop.

Configuration

```
R4:
policy-map SUBRATE_POLICER
 class FROM_R1
  police cir 64000 bc 3200 pir 128000 be 6400
   conform-action set-prec-transmit 1
   exceed-action set-prec-transmit 0
   violate-action drop
 class FROM_R6
  police cir 64000 bc 3200 pir 128000 be 6400
   conform-action set-prec-transmit 1
   exceed-action set-prec-transmit 0
   violate-action drop
!
policy-map POLICE_VLAN146
 class HTTP
   no police
   service-policy SUBRATE_POLICER
!
interface FastEthernet 0/1
  service-policy input POLICE_VLAN146
```

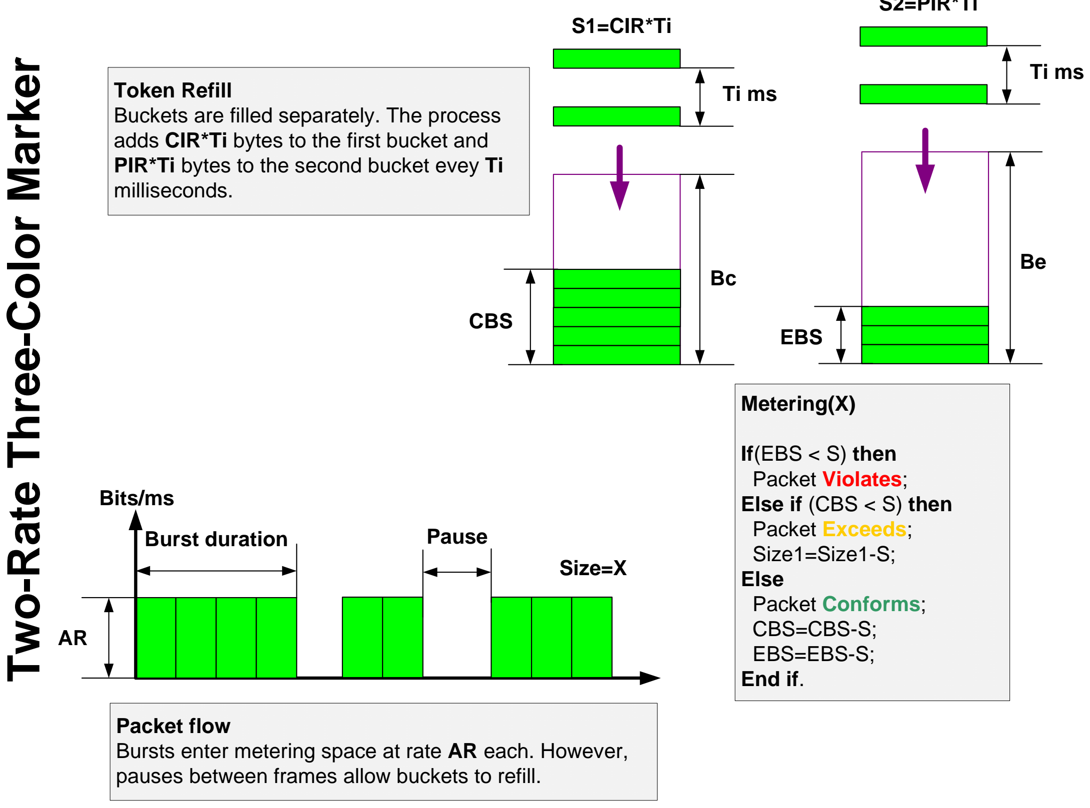
Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Verification

Note

The two-rate policer implements an RFC-based procedure to meter traffic rate against two pre-configured rates, average and peak. The RFC names this procedure as a two-rate three-color marker, or “trTCM”. Three colors correspond to “Conform”, “Exceed” and “Violate” actions of the policer.

Four configurable parameters apply to the two-rate policer: CIR, PIR, B_c, B_e. This time B_c bounds the average bursts, and B_e bounds the peak burst. Two-rate policer runs two token buckets at the same time. Each bucket has its own fill rate, as illustrated below.



The process refills each bucket separately, at its own rate. The first bucket fills at rate CIR, while the second bucket fills at rate PIR. As usual, a special timer fires every T_i interval and triggers the token refresh procedure. This, however, limits the “resolution” of metering procedure.

The alternate implementation, used by Cisco with trTCM, is similar to the implementation of srTCM. The algorithm uses the following variables:

CBS – current normal burst size (initially set to B_c)
EBS – current excess burst size (initially set to B_e)
T0 – last packet arrival time.

Now imagine that a new packet of size "S" arrives at time "T".

Step 1:
The process computes the accumulated credit and new burst sizes:

```
Cr1 = CIR*(T-T0);
Cr2 = PIR*(T-T0);
```

then adds these values to CBS and EBS:

```
if ((CBS + Cr1) <= Bc) then
      CBS = CBS + Cr1;
else
      CBS = Bc;
end if

if ((EBS + Cr2) <= Be) then
      EBS = EBS + Cr2;
else
      EBS = Be;
end if
```

Step 2:
The process compares the packet size to accumulated credits:

```
if ( S > EBS ) then
    Packet Violates;
else if (S > CBS) then
    Packet Exceeds;
    EBS = EBS – S;
else
    Packet Conforms;
    EBS = EBS – S;
    CBS = CBS – S;
end if.
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Effectively the two-rate policer runs two sliding windows over the packet stream, metering against two bitrates using two separate averaging intervals. This procedure is common on Service Provider edges that offer oversubscription to customers, based on PIR and CIR rates.

To verify this generate a single traffic flow from R1 down to SW2 across R4. To accomplish this, you need to shut down the Frame-Relay interface of R5. The general idea is that link between R4 and R5 is set to 128Kbps and the traffic flows will eventually oversubscribe it. The task configuration ensures that an individual flow across the serial link never exceeds 128Kbps, but allows every flow to send traffic above guaranteed 64Kbps.

Generate an HTTP traffic flow from R1 to SW2 and pre-shape it down to 128Kbps on R6. Ensure the traffic shaper burst is smaller than the configured policer burst by at least one average packet size.

```
R1:
ip http server
ip http path flash:
!
! Bc = 12800 bits = 1600 bytes
!
interface FastEthernet 0/0
  traffic-shape rate 128000 12800 0

R4:
interface FastEthernet 0/1
  load-interval 30

R5:
interface Serial 0/0
 shutdown
```

```
Rack1SW2#copy http://admin:cisco@155.1.146.1/c2600-adventerprisek9-
mz.124-10.bin null:
```

Copyright © 2009 Internetwork Expert

Note that traffic from R1 to SW2 exceeds the policer average rate but not the peak rate. Therefore, the policer marks half of the packets as exceeding. Since the traffic is TCP based, it adapts to the maximum allowable rate.

```
Rack1R4#show policy-map interface fastEthernet 0/1
 FastEthernet0/1

  Service-policy input: POLICE_VLAN146

    Class-map: HTTP (match-all)
      2331 packets, 1373671 bytes
      30 second offered rate 121000 bps, drop rate 0 bps
      Match: access-group name HTTP

      Service-policy : SUBRATE_POLICER

        Class-map: FROM_R1 (match-all)
          2331 packets, 1373671 bytes
          30 second offered rate 121000 bps, drop rate 0 bps
          Match: access-group name FROM_R1
          police:
              cir 64000 bps, bc 3200 bytes
              pir 128000 bps, be 6400 bytes
            conformed 1183 packets, 696351 bytes; actions:
              set-prec-transmit 1
            exceeded 1148 packets, 677320 bytes; actions:
              set-prec-transmit 0
            violated 0 packets, 0 bytes; actions:
              drop
            conformed 62000 bps, exceed 61000 bps, violate 0 bps

        Class-map: FROM_R6 (match-all)
          0 packets, 0 bytes
          30 second offered rate 0 bps, drop rate 0 bps
          Match: access-group name FROM_R6
          police:
              cir 64000 bps, bc 3200 bytes
              pir 128000 bps, be 6400 bytes
            conformed 0 packets, 0 bytes; actions:
              set-prec-transmit 1
            exceeded 0 packets, 0 bytes; actions:
              set-prec-transmit 0
            violated 0 packets, 0 bytes; actions:
              drop
            conformed 0 bps, exceed 0 bps, violate 0 bps

        Class-map: class-default (match-any)
          0 packets, 0 bytes
          30 second offered rate 0 bps, drop rate 0 bps
          Match: any

    Class-map: class-default (match-any)
      1081 packets, 318426 bytes
      30 second offered rate 0 bps, drop rate 0 bps
      Match: any
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Now add another traffic flow from R6 down to SW4 and across the serial link between R4 and R5.

```
R6:
ip http server
ip http path flash:
!
interface FastEthernet 0/0.146
  traffic-shape rate 128000 12800 0
```

```
Rack1SW4#copy http://admin:cisco@155.1.146.6/c2600-adventerprisek9-mz.124-
10.bin null:
```

Verify the policer statistics once again. Now both policers meter the same average rate, and there is no exceeding traffic. This is because the oversubscribed serial link between R4 and R5 uses fair queuing and divides the bandwidth equally between the two flows.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
Rack1R4#show policy-map interface fastEthernet 0/1
 FastEthernet0/1

  Service-policy input: POLICE_VLAN146

    Class-map: HTTP (match-all)
      15682 packets, 9247646 bytes
      30 second offered rate 127000 bps, drop rate 0 bps
      Match: access-group name HTTP

      Service-policy : SUBRATE_POLICER

        Class-map: FROM_R1 (match-all)
          14228 packets, 8392901 bytes
          30 second offered rate 63000 bps, drop rate 0 bps
          Match: access-group name FROM_R1
          police:
              cir 64000 bps, bc 3200 bytes
              pir 128000 bps, be 6400 bytes
            conformed 7917 packets, 4669411 bytes; actions:
              set-prec-transmit 1
            exceeded 6311 packets, 3723490 bytes; actions:
              set-prec-transmit 0
            violated 0 packets, 0 bytes; actions:
              drop
            conformed 63000 bps, exceed 0 bps, violate 0 bps

        Class-map: FROM_R6 (match-all)
          1454 packets, 854745 bytes
          30 second offered rate 63000 bps, drop rate 0 bps
          Match: access-group name FROM_R6
          police:
              cir 64000 bps, bc 3200 bytes
              pir 128000 bps, be 6400 bytes
            conformed 1451 packets, 852975 bytes; actions:
              set-prec-transmit 1
            exceeded 3 packets, 1770 bytes; actions:
              set-prec-transmit 0
            violated 0 packets, 0 bytes; actions:
              drop
            conformed 63000 bps, exceed 0 bps, violate 0 bps

        Class-map: class-default (match-any)
          0 packets, 0 bytes
          30 second offered rate 0 bps, drop rate 0 bps
          Match: any

    Class-map: class-default (match-any)
      1190 packets, 341980 bytes
      30 second offered rate 0 bps, drop rate 0 bps
      Match: any
```

Note that traffic never violated the peak rate condition, thanks to the fact that the link between R4 and R5 is 128Kbps.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.45 MQC Peak Shaping

- Configure R1 and R6 to shape HTTP traffic to a peak rate of 128Kbps as it is sent out to VLAN 146.
- Use B_c and B_e bursts based on a 10ms interval.

Configuration

```
R1:
access-list 180 permit tcp any eq 80 any
!
class-map HTTP
 match access-group 180
!
policy-map POLICY_VLAN146_OUT
  class HTTP
    shape peak 64000 6400 6400
!
interface FastEthernet 0/0
  service-policy output POLICY_VLAN146_OUT

R6:
access-list 180 permit tcp any eq 80 any
!
class-map HTTP
 match access-group 180
!
policy-map POLICY_VLAN146_OUT
  class HTTP
    shape peak 64000 6400 6400
!
interface FastEthernet 0/0.146
  service-policy output POLICY_VLAN146_OUT
```

Verification

Note

Peak shaping may look confusing at first sight, however, its function becomes clear once you think of oversubscription. As we discussed before, oversubscription means selling customers more bandwidth than a network can supply, hoping that not all connections would use their maximum sending rate at the same time. With oversubscription, a traffic contract usually specifies three parameters: PIR, CIR and T_c – peak rate, committed rate, and averaging time interval for rate measurements. The SP allows customers to send traffic at rates up to PIR, but only guarantees CIR rate in case of network congestion.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Inside the network, the SP uses any of the max-min scheduling procedures to implement bandwidth sharing in such manner that oversubscribed traffic has lower preference than conforming traffic. Additionally, the SP generally assumes that customers respond to notifications of traffic congestion in the network (either explicit, such as FECN/BECN/TCP ECN or implicit such as packet drops in TCP) by slowing down sending rate.

Commonly customers implement traffic shaping to conform to their traffic contract, and the provider uses traffic policing to enforce the contract. If a contract specifies PIR, then it makes sense for customer to shape traffic at PIR rate. However, this makes difficult to deduce the CIR value just by looking at the router configuration. In some circumstances, like with Frame-Relay networks, a secondary parameter, known as minCIR, may help to understand the configuration quickly. In general, it would benefit to see CIR and PIR in the shaping configuration at the same time. This is exactly the idea behind shape peak. When you configure **`shape peak <CIR> <Bc> <Be>`** the actual maximum sending rate is limited to $PIR = CIR*(1+B_e/B_c)$.

That is, each time interval $T_c=B_c/CIR$ the shaper allows sending up to B_c+B_e bits of data. By default, if you omit the value for B_e, it equals to B_c and thus PIR=2*CIR by default. However, due to a cosmetic IOS bug in the show output, this is NOT reflected unless you explicitly specify the Be value in command line. With shape peak configured this way, you can see both CIR as the “average rate” and PIR as the “target rate” when issuing “show policy-map” command.

```
Rack1R6#show policy-map interface fastEthernet 0/0.146
 FastEthernet0/0.146
<snip>
      Traffic Shaping
           Target/Average   Byte   Sustain   Excess    Interval  Increment
             Rate           Limit  bits/int  bits/int  (ms)      (bytes)
           128000/64000     1600   6400      6400      100       1600
...
```

All other shaping functions remain the same as with the classic GTS - shape peak is just more suited for use with oversubscription scenarios. Also, in Frame-Relay networks you may want to use configuration similar to the following to respond to congestion notifications:

```
shape peak <CIR> <Bc> <Be>
shape adaptive <CIR>
```

To verify the configuration, ensure that traffic from VLAN146 goes across R4 to reach R5. Then, start transferring a file from R1 to SW2.

```
R1 & R6:
ip http server
ip http path flash:

Rack1SW2#copy http://admin:cisco@155.1.146.1/c2600-adventerprisek9-
mz.124-10.bin null:
```

Check the policer statistics on R4 (remember we have two-rate policers admitting traffic flows from R1 and R6). Note that the policer considers half of the packets as exceeding, and that the offered rate is close to 128Kbps (peak).

```
Rack1R4#show policy-map interface fastEthernet 0/1
 FastEthernet0/1

  Service-policy input: POLICE_VLAN146

    Class-map: HTTP (match-all)
      20451 packets, 12066090 bytes
      30 second offered rate 126000 bps, drop rate 0 bps
      Match: access-group name HTTP

      Service-policy : SUBRATE_POLICER

        Class-map: FROM_R1 (match-all)
          20451 packets, 12066090 bytes
          30 second offered rate 126000 bps, drop rate 0 bps
          Match: access-group name FROM_R1
          police:
              cir 64000 bps, bc 3200 bytes
              pir 128000 bps, be 6400 bytes
            conformed 11113 packets, 6556670 bytes; actions:
              set-prec-transmit 1
            exceeded 9338 packets, 5509420 bytes; actions:
              set-prec-transmit 0
            violated 0 packets, 0 bytes; actions:
              drop
            conformed 64000 bps, exceed 62000 bps, violate 0 bps

        Class-map: FROM_R6 (match-all)
          0 packets, 0 bytes
          30 second offered rate 0 bps, drop rate 0 bps
          Match: access-group name FROM_R6
          police:
              cir 64000 bps, bc 3200 bytes
              pir 128000 bps, be 6400 bytes
            conformed 0 packets, 0 bytes; actions:
              set-prec-transmit 1
            exceeded 0 packets, 0 bytes; actions:
              set-prec-transmit 0
            violated 0 packets, 0 bytes; actions:
              drop
            conformed 0 bps, exceed 0 bps, violate 0 bps
<snip>
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Check the shaper statistics on R1. Note the average and target traffic rates. Also, note that shaping is no longer active, although some packets have been delayed. This is because the initial stage of TCP transmission is usually aggressive, and it takes time to adapt to the enforced rates and stabilize the sending rate. The sustained sending rate is close to 128Kbps.

```
Rack1R1#show policy-map interface fastEthernet 0/0
 FastEthernet0/0

  Service-policy output: POLICY_VLAN146_OUT

    Class-map: HTTP (match-all)
      3225 packets, 1897929 bytes
      30 second offered rate 124000 bps, drop rate 0 bps
      Match: access-group 180
      Traffic Shaping
           Target/Average   Byte   Sustain   Excess    Interval  Increment
             Rate           Limit  bits/int  bits/int  (ms)      (bytes)
           128000/64000     1600   6400      6400      100       1600

        Adapt  Queue     Packets   Bytes     Packets   Bytes     Shaping
        Active Depth                         Delayed   Delayed   Active
        -      0         3225      1897929   348       205320    no

    Class-map: class-default (match-any)
      29 packets, 4378 bytes
      30 second offered rate 0 bps, drop rate 0 bps
      Match: any
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Now start the second file transfer from R6 to SW4.

```
Rack1SW4#copy http://admin:cisco@155.1.146.6/c2600-adventerprisek9-mz.124-
10.bin null:
```

Check the accumulated statistics on R4 once again. Note that now both classes conform to their guaranteed rate, thanks to the fair-sharing of the serial link between R4 and R5.

```
Rack1R4#show policy-map interface fastEthernet 0/1
 FastEthernet0/1

  Service-policy input: POLICE_VLAN146

    Class-map: HTTP (match-all)
      35113 packets, 20715559 bytes
      30 second offered rate 126000 bps, drop rate 0 bps
      Match: access-group name HTTP

      Service-policy : SUBRATE_POLICER

        Class-map: FROM_R1 (match-all)
          29986 packets, 17691740 bytes
          30 second offered rate 63000 bps, drop rate 0 bps
          Match: access-group name FROM_R1
          police:
              cir 64000 bps, bc 3200 bytes
              pir 128000 bps, be 6400 bytes
            conformed 18466 packets, 10894940 bytes; actions:
              set-prec-transmit 1
            exceeded 11520 packets, 6796800 bytes; actions:
              set-prec-transmit 0
            violated 0 packets, 0 bytes; actions:
              drop
            conformed 63000 bps, exceed 0 bps, violate 0 bps

        Class-map: FROM_R6 (match-all)
          5127 packets, 3023819 bytes
          30 second offered rate 63000 bps, drop rate 0 bps
          Match: access-group name FROM_R6
          police:
              cir 64000 bps, bc 3200 bytes
              pir 128000 bps, be 6400 bytes
            conformed 5124 packets, 3022049 bytes; actions:
              set-prec-transmit 1
            exceeded 3 packets, 1770 bytes; actions:
              set-prec-transmit 0
            violated 0 packets, 0 bytes; actions:
              drop
            conformed 63000 bps, exceed 0 bps, violate 0 bps

        Class-map: class-default (match-any)
          0 packets, 0 bytes
          30 second offered rate 0 bps, drop rate 0 bps
          Match: any
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Now check the statistics for traffic shapers at R1 and R6. This time, both shapers are inactive, and offered rate is close to 64Kbps (the CIR value).

```
Rack1R6#show policy-map interface fastEthernet 0/0.146
 FastEthernet0/0.146

  Service-policy output: POLICY_VLAN146_OUT

    Class-map: HTTP (match-all)
      6846 packets, 4065413 bytes
      5 minute offered rate 63000 bps, drop rate 0 bps
      Match: access-group 180
      Traffic Shaping
           Target/Average   Byte   Sustain   Excess    Interval  Increment
             Rate           Limit  bits/int  bits/int  (ms)      (bytes)
           128000/64000     1600   6400      6400      100       1600

        Adapt  Queue     Packets   Bytes     Packets   Bytes     Shaping
        Active Depth                         Delayed   Delayed   Active
        -      0         6846      4065413   3         1782      no

    Class-map: class-default (match-any)
      191 packets, 43930 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any

Rack1R1#show policy-map interface fastEthernet 0/0
 FastEthernet0/0

 Service-policy output: POLICY_VLAN146_OUT

    Class-map: HTTP (match-all)
      33062 packets, 19505469 bytes
      30 second offered rate 63000 bps, drop rate 0 bps
      Match: access-group 180
      Traffic Shaping
           Target/Average   Byte   Sustain   Excess    Interval  Increment
             Rate           Limit  bits/int  bits/int  (ms)      (bytes)
           128000/64000     1600   6400      6400      100       1600

        Adapt  Queue     Packets   Bytes     Packets   Bytes     Shaping
        Active Depth                         Delayed   Delayed   Active
        -      0         33062     19505469  2632      1552858   no

    Class-map: class-default (match-any)
      7641 packets, 7385752 bytes
      30 second offered rate 0 bps, drop rate 0 bps
      Match: any
```

10.46 MQC Percent-Based Policing

- Hard code R1's FastEthernet link to a speed of 10Mbps.
- Limit the traffic entering this link to 10% of this rate with a burst value of 125ms.

Configuration

```
R1:
policy-map POLICE_VLAN146
 class class-default
   police rate percent 10 burst 125 ms
!
interface FastEthernet 0/0
 speed 10
 service-policy input POLICE_VLAN146
```

Verification

Note

Percent based policing allows specifying the policer rate as a percentage of the interface speed. At the same time, you may configure burst sizes in milliseconds duration. The actual burst size will be *InterfaceSpeed*BurstDuration* based on the interface's speed. This method does not use "absolute" values but rather relative settings based on interface parameters. The drawback is limited range of speed values, since CLI does not allow fractional percent values. Thus, you may only step by one percent configuring policed rates.

The metering procedure remains the same as with srTCM or trTCM, you just change the way to set configuration parameters.

To verify, send a stream of 1000 byte ICMP packets from R4 to R1 and shape it to 2Mbps. Select the burst size for the shaper to be less than the policer's burst by approximately one packet size (1000+14 bytes). You may learn the policer burst size by using the **`show policy-map`** command.

Based on that, the shaper burst is (15625-1014)*8 = 117000 bps, but we'll make it a bit smaller, around 116000 bps.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
R4:
interface FastEthernet 0/1
 traffic-shape rate 2000000 116000
```

```
Rack1R4#ping 155.1.146.1 size 1000 timeout 0 repeat 100000000
```

Check the policer statistics to view number of conforming and exceeding packets. Note the amount is approximately the same, and conformed rate is getting close to 1Mbps (which is 10% of interface bandwidth), however we cannot reach the exact metering precision due to small burst size mismatches and jitter.

```
Rack1R1#show policy-map interface fastEthernet 0/0
 FastEthernet0/0

  Service-policy input: POLICE_VLAN146

    Class-map: class-default (match-any)
      40769 packets, 41308718 bytes
      30 second offered rate 1791000 bps, drop rate 889000 bps
      Match: any
      police:
          rate 10 % burst 125 ms
          rate 1000000 bps, burst 15625 bytes
        conformed 20535 packets, 20791442 bytes; actions:
          transmit
        exceeded 20234 packets, 20517276 bytes; actions:
          drop
        conformed 900000 bps, exceed 889000 bps
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.47 MQC Header Compression

- Enable MQC based RTP header compression on the serial link between R1 and R3.
- Assume that RTP packets use the port range 16384-32767.
- Allocate enough priority bandwidth to allow 2 G.729 compressed calls over this link, for a total of 24Kbps.

Configuration

```
R1:
class-map VOICE_BEARER
 match ip rtp 16384 16383
!
policy-map SERIAL_LINK_OUT
 class VOICE_BEARER
   priority 24
   compression header ip rtp
!
interface Serial 0/1
 bandwidth 128
 service-policy output SERIAL_LINK_OUT

R3:
class-map VOICE_BEARER
 match ip rtp 16384 16383
!
policy-map SERIAL_LINK_OUT
 class VOICE_BEARER
   priority 24
   compression header ip rtp
!
interface Serial 1/2
 bandwidth 128
 clock rate 128000
 service-policy output SERIAL_LINK_OUT
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Verification

> **Note**
>
> The functionality of MQC based RTP header compression is the same as in the legacy case. It is important that the MQC policer for the priority queue accounts for the effect of header compression, therefore when configuring bandwidth for the priority queue, base your calculations on compressed packet sizes.
>
> The RTP header compression process reduces IP/UDP/RTP headers from 40 bytes down to 2 bytes, as Cisco does not preserve the UDP checksum. In the case of G.729 and HDLC the resulting bandwidth for one calls is (2*10+2+7)*50*8 = 11600 bps.
>
> The computation assumes the fact that G.729 generates 50pps with 20 bytes of payload each. The total overhead for HDLC is 7 bytes. We round up the value in bps to the next whole number, which is 12Kbps. Two calls will then consume 24Kbps.
>
> Note that MQC based TCP header compression can be configured in this same manner.

```
Rack1R1#show policy-map interface serial 0/1
 Serial0/1

  Service-policy output: SERIAL_LINK_OUT

    Class-map: VOICE_BEARER (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: ip rtp 16384 16383
      Queueing
        Strict Priority
        Output Queue: Conversation 40
        Bandwidth 24 (kbps) Burst 600 (Bytes)
        (pkts matched/bytes matched) 0/0
        (total drops/bytes drops) 0/0
      compress:
          header ip rtp
          UDP/RTP (compression on, IPHC, RTP)
            Sent:    0 total, 0 compressed,
                     0 bytes saved, 0 bytes sent
                     rate 0 bps

    Class-map: class-default (match-any)
      108 packets, 23328 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
Rack1R3# show policy-map interface serial 1/2
 Serial1/2

  Service-policy output: SERIAL_LINK_OUT

    Class-map: VOICE_BEARER (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: ip rtp 16384 16383
      Queueing
        Strict Priority
        Output Queue: Conversation 40
        Bandwidth 24 (kbps) Burst 600 (Bytes)
        (pkts matched/bytes matched) 0/0
        (total drops/bytes drops) 0/0
      compress:
          header ip rtp
          UDP/RTP (compression on, IPHC, RTP)
            Sent:    0 total, 0 compressed,
                     0 bytes saved, 0 bytes sent
                     rate 0 bps


    Class-map: class-default (match-any)
      91 packets, 15148 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any
```

Use the legacy commands to verify the configuration as well.

```
Rack1R1#show ip rtp header-compression
RTP/UDP/IP header compression statistics:
We're compressing using MQC profiles, use the
MQC commands to see the stats for each class.
  Interface Serial0/1 (compression on, IPHC, RTP)
    Rcvd:    0 total, 0 compressed, 0 errors, 0 status msgs
             0 dropped, 0 buffer copies, 0 buffer failures
    Sent:    0 total, 0 compressed, 0 status msgs, 0 not predicted
             0 bytes saved, 0 bytes sent
    Connect: 32 rx slots, 32 tx slots,
             0 misses, 0 collisions, 0 negative cache hits, 32 free
contexts
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.48 Using Class-Based GTS for FRTS

- Configure R3 and R5 to shape traffic going out to the Frame Relay network using MQC based GTS.
- R3's policy should match PVC 305, and shape to an average rate of 256Kbps.
- R5's policy should match PVC 503 with a peak shaping rate of 128Kbps, and match PVC 502 with a peak shaping rate of 256Kbps.
- R3 should use a Bc and Be of 12800 and 6400 for the shaper to R5.
- R5 should use a Bc and Be of 6400 for the shaper to R3, and a Bc and Be of 12800 for the shaper to R2.

Configuration

```
R3:
class-map DLCI_305
 match fr-dlci 305
!
policy-map POLICY_FR_OUT
 class DLCI_305
  shape average 256000 12800 6400
!
interface Serial 1/0
 service-policy output POLICY_FR_OUT

R5:
class-map DLCI_502
 match fr-dlci 502
!
class-map DLCI_503
 match fr-dlci 503
!
policy-map POLICY_FR_OUT
 class DLCI_503
  shape peak 128000 6400 6400
 class DLCI_502
  shape peak 256000 12800 12800
!
interface Serial 0/0
  service-policy output POLICY_FR_OUT
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Verification

Note

You can use Class-Based GTS (CBTS) to simulate Frame-Relay traffic shaping. The core of this functionality is the **`match fr-dlci`** command that allows matching a specific DLCI and classifying traffic on per-VC basis. However, some important functionality is missing, for example the **`shape adaptive`** command does not work. Unlike legacy GTS, CBTS allows shaping on per-VC basis. Some of the differences between legacy GTS, legacy FRTS, and CBTS are as follows.

- You can apply legacy GTS per interface or sub-interface only, and the queuing strategy is fixed to WFQ. You may apply CBTS to sub-interfaces as well as to selected PVCs.
- FRTS turns interface queue into FIFO, but allows using PQ/CQ/WFQ/CBWFQ as per-VC queues. With CBTS, interface queue is either FIFO or CBWFQ. However, you may also use the legacy queuing methods. The per-VC queue is limited to CBWFQ with CBTS.
- CBTS does not affect PVCs not matched by configured class-maps. Legacy FRTS enforces a default rate on all PVCs bound to an interface.
- Legacy FRTS allows per-VC fragmentation, while CBTS only supports per-interface fragmentation.
- CBTS does not support adaptive Frame-Relay traffic shaping.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

This configuration can be verified as follows:

```
Rack1R5#show policy-map interface serial 0/0

 Serial0/0

  Service-policy output: POLICY_FR_OUT

    Class-map: DLCI_503 (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: fr-dlci 503
      Traffic Shaping
           Target/Average   Byte   Sustain   Excess    Interval  Increment
             Rate           Limit  bits/int  bits/int  (ms)      (bytes)
           256000/128000    1600   6400      6400      50        1600

        Adapt  Queue     Packets   Bytes     Packets   Bytes     Shaping
        Active Depth                         Delayed   Delayed   Active
        -      0         0         0         0         0         no

    Class-map: DLCI_502 (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: fr-dlci 502
      Traffic Shaping
           Target/Average   Byte   Sustain   Excess    Interval  Increment
             Rate           Limit  bits/int  bits/int  (ms)      (bytes)
           512000/256000    3200   12800     12800     50        3200

        Adapt  Queue     Packets   Bytes     Packets   Bytes     Shaping
        Active Depth                         Delayed   Delayed   Active
        -      0         0         0         0         0         no

    Class-map: class-default (match-any)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any
```

Note the PIR/CIR rate in the output above. Now verify traffic-shaping queues. Note that CBWFQ bandwidth bases on the CIR value, not the PIR rate (128 and 256K respectively).

Copyright © 2009 Internetwork Expert

```
Rack1R5#show traffic-shape queue
Traffic queued in shaping queue on Serial0/0
 Traffic shape class: DLCI_503
  Queueing strategy: weighted fair
  Queueing Stats: 0/1000/64/0 (size/max total/threshold/drops)
     Conversations  0/0/16 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 128 kilobits/sec


 Traffic shape class: DLCI_502
  Queueing strategy: weighted fair
  Queueing Stats: 0/1000/64/0 (size/max total/threshold/drops)
     Conversations  0/0/32 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 256 kilobits/sec

Rack1R3#show policy-map interface serial 1/0

 Serial1/0

  Service-policy output: POLICY_FR_OUT

    Class-map: DLCI_305 (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: fr-dlci 305
      Traffic Shaping
           Target/Average   Byte   Sustain   Excess    Interval  Increment
             Rate           Limit  bits/int  bits/int  (ms)      (bytes)
           256000/256000    2400   12800     6400      50        1600

        Adapt  Queue     Packets   Bytes     Packets   Bytes     Shaping
        Active Depth                         Delayed   Delayed   Active
        -      0         0         0         0         0         no

    Class-map: class-default (match-any)
      645 packets, 8385 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any

Rack1R3#show traffic-shape queue
Traffic queued in shaping queue on Serial1/0
 Traffic shape class: DLCI_305
  Queueing strategy: weighted fair
  Queueing Stats: 0/1000/64/0 (size/max total/threshold/drops)
     Conversations  0/0/32 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
     Available Bandwidth 256 kilobits/sec
```

10.49 MQC Based Frame-Relay DE-Marking

- Modify R5's shaper so that the Frame Relay Discard Eligible bit is set on all frames going out DLCI 502.

Configuration

```
R5:
policy-map POLICY_FR_OUT
 class DLCI_502
   shape peak 256000 12800 12800
   set fr-de
!
interface Serial 0/0
  service-policy output POLICY_FR_OUT
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Verification

Note

Unlike the legacy DE-marking procedure, MQC-based DE marking applies to all packet switching paths, including CEF. In this particular case, we apply DE-marking to all traffic that exits DLCI 502. To verify this generate some packets over DLCI 502 and see if they are marked.

```
Rack1R5#show policy-map interface serial 0/0

 Serial0/0

  Service-policy output: POLICY_FR_OUT

    Class-map: DLCI_503 (match-all)
      7 packets, 1672 bytes
      5 minute offered rate 1000 bps, drop rate 0 bps
      Match: fr-dlci 503
      Traffic Shaping
           Target/Average   Byte   Sustain   Excess    Interval  Increment
             Rate           Limit  bits/int  bits/int  (ms)      (bytes)
           256000/128000    1600   6400      6400      50        1600

        Adapt  Queue     Packets   Bytes     Packets   Bytes     Shaping
        Active Depth                         Delayed   Delayed   Active
        -      0         6         1616      0         0         no

    Class-map: DLCI_502 (match-all)
      17 packets, 2712 bytes
      5 minute offered rate 1000 bps, drop rate 0 bps
      Match: fr-dlci 502
      Traffic Shaping
           Target/Average   Byte   Sustain   Excess    Interval  Increment
             Rate           Limit  bits/int  bits/int  (ms)      (bytes)
           512000/256000    3200   12800     12800     50        3200

        Adapt  Queue     Packets   Bytes     Packets   Bytes     Shaping
        Active Depth                         Delayed   Delayed   Active
        -      0         16        2656      0         0         no
      QoS Set
        fr-de
          Packets marked 17

    Class-map: class-default (match-any)
      20 packets, 3423 bytes
      5 minute offered rate 2000 bps, drop rate 0 bps
      Match: any
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.50 Using MQC CBWFQ with Legacy FRTS

- Configure legacy FRTS on R2's connection to DLCI 205 as follows:
  - Shape to the average rate of 512Kbps
  - Use an interval of 10ms
  - Adaptively shape down to 256Kbps if BECNs are received
- Configure an MQC policy inside this legacy shaper as follows:
  - Guarantee 64Kbps of PVC bandwidth to HTTP flows
  - Other unmatched flows should be guaranteed 192Kbps

Configuration

```
R2:
class-map HTTP
 match protocol http
!
policy-map CBWFQ
 class HTTP
  bandwidth 64
 class class-default
  bandwidth 192
!
map-class frame-relay DLCI_205
 frame-relay cir 512000
 frame-relay bc 25600
 frame-relay be 0
 frame-relay mincir 256000
 frame-relay adaptive-shaping becn
 service-policy output CBWFQ
!
interface Serial 0/0
 frame-relay traffic-shaping
 frame-relay interface-dlci 205
  class DLCI_205
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Verification

> ✐ **Note**
>
> Even though the preferred way of configuring Frame-Relay Traffic Shaping and CBWFQ is through MQC Based FRTS, a CBWFQ policy may still be applied as a per-VC queuing policy by calling it from a legacy FRTS map-class. The MQC policy specifies CBWFQ weights using the unified MQC syntax. In this case, we use NBAR to classify HTTP traffic.
>
> CBWFQ as the per-VC queue is an extension of using simple WFQ or WFQ with IP RTP priority. You may use all CBWFQ features, including LLQ, however the legacy FRTS still performs the shaping function. This means that the system automatically enforces a 56Kbps rate on all PVCs unless you explicitly assign map-classes for them.
>
> One important feature for per-VC CBWFQ is that IOS deduces the "available bandwidth" for CBWFQ based on the **mincir** settings. By default minCIR=CIR/2, so you may need to adjust this value if you do not do oversubscription.
>
> To verify the policy-map status and statistics you may no longer use the **show policy-map** command. Instead, the **show frame-relay pvc** command is used to see the policy-map attached to the respective PVC.

```
Rack1R2#show frame-relay pvc 205

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 205, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

  input pkts 14337         output pkts 7194         in bytes 4527272
  out bytes 1982084        dropped pkts 0           in pkts dropped 0
  out pkts dropped 0                out bytes dropped 0
  in FECN pkts 0           in BECN pkts 0           out FECN pkts 0
  out BECN pkts 0          in DE pkts 12778         out DE pkts 0
  out bcast pkts 7184      out bcast bytes 1981044
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
  Shaping adapts to BECN
  pvc create time 1d01h, last time pvc status changed 16:25:16
  cir 512000    bc 25600     be 0         byte limit 3200   interval 50
  mincir 256000    byte increment 3200  Adaptive Shaping BECN
  pkts 16        bytes 4416      pkts delayed 0         bytes delayed 0
  shaping inactive
  traffic shaping drops 0
  service policy CBWFQ
 Serial0/0: DLCI 205 -
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
 Service-policy output: CBWFQ

    Class-map: HTTP (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: protocol http
      Queueing
        Output Queue: Conversation 41
        Bandwidth 64 (kbps)Max Threshold 64 (packets)
        (pkts matched/bytes matched) 0/0
        (depth/total drops/no-buffer drops) 0/0/0

    Class-map: class-default (match-any)
      7 packets, 1932 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any
      Queueing
        Output Queue: Conversation 42
        Bandwidth 192 (kbps)Max Threshold 64 (packets)
        (pkts matched/bytes matched) 0/0
        (depth/total drops/no-buffer drops) 0/0/0
  Output queue size 0/max total 600/drops 0
```

Generate some HTTP traffic across R2 and add another ICMP flow to verify if traffic matches the configured policy.

```
Rack1R5#copy http://admin:cisco@155.1.0.2/c2600-adventerprisek9-mz.124-10.bin
null:
Loading http://***********@155.1.0.2/c2600-adventerprisek9-mz.124-10.bin !!!!!

Rack1SW2#ping 155.1.0.2 size 500 repeat 10000 timeout 1

Type escape sequence to abort.
Sending 10000, 500-byte ICMP Echos to 155.1.0.2, timeout is 1 seconds:
!!!!!
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Verify queue statistics once again and see traffic matches for both classes in the PVC service-policy.

```
Rack1R2#show frame-relay pvc 205

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 205, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

  input pkts 22008          output pkts 12045        in bytes 5466236
  out bytes 7718533         dropped pkts 0           in pkts dropped 0
  out pkts dropped 0                 out bytes dropped 0
  in FECN pkts 0            in BECN pkts 0           out FECN pkts 0
  out BECN pkts 0           in DE pkts 20449         out DE pkts 0
  out bcast pkts 7375       out bcast bytes 2033760
  5 minute input rate 10000 bits/sec, 10 packets/sec
  5 minute output rate 61000 bits/sec, 6 packets/sec
  Shaping adapts to BECN
  pvc create time 1d02h, last time pvc status changed 16:54:44
  cir 512000    bc 25600     be 0          byte limit 3200   interval 50
  mincir 256000    byte increment 3200  Adaptive Shaping BECN
  pkts 4867      bytes 5740865   pkts delayed 2         bytes delayed 3008
  shaping inactive
  traffic shaping drops 0
  service policy CBWFQ
 Serial0/0: DLCI 205 -

  Service-policy output: CBWFQ

    Class-map: HTTP (match-all)
      3584 packets, 5146633 bytes
      5 minute offered rate 52000 bps, drop rate 0 bps
      Match: protocol http
      Queueing
        Output Queue: Conversation 41
        Bandwidth 64 (kbps)Max Threshold 64 (packets)
        (pkts matched/bytes matched) 2/3008
        (depth/total drops/no-buffer drops) 0/0/0

    Class-map: class-default (match-any)
      1282 packets, 600300 bytes
      5 minute offered rate 7000 bps, drop rate 0 bps
      Match: any
      Queueing
        Output Queue: Conversation 42
        Bandwidth 192 (kbps)Max Threshold 64 (packets)
        (pkts matched/bytes matched) 0/0
        (depth/total drops/no-buffer drops) 0/0/0
  Output queue size 0/max total 600/drops 0
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.51 MQC Compatible FRF.12 Fragmentation

- Configure R2, R3, and R5 to fragment packets to 480 bytes as they are sent out to the Frame Relay network.
- Enable fragmentation on the main Frame-Relay interfaces of R3 and R4.
- Enable fragmentation under the map-class for DLCI 205 on R2.

Configuration

```
R2:
map-class frame-relay DLCI_205
 frame-relay fragment 480

R3:
interface Serial 1/0
  frame-relay fragment 480 end-to-end

R5:
interface Serial 0/0
  frame-relay fragment 480 end-to-end
```

Verification

Note

The optimal fragment size ensures the fragment serialization delay of 10ms. Based on the *slowest* link speed we can calculate the fragment size as:

FragSize = 384000*0,01/8 = 480 bytes

MQC only allows enabling FRF.12 fragmentation at the interface level. This command is incompatible with legacy FRTS enabled at interface level. FRF.12 with MQC performs the same function it does with legacy FRTS. However, this time fragment-interleaving queue is hidden, and you cannot observe it with show commands. MQC restricts this feature for use with the whole interface only – you cannot enable it per VC. However, it makes perfect sense, since enabling fragmentation on just one PVC is not effective, until you enable it on all other PVCs.

To verify, ensure you send and receive fragmented packets on all participating routers.

```
Rack1R5#show frame-relay fragment
interface                 dlci frag-type  size in-frag    out-frag   dropped-
frag
Se0/0                     501  end-to-end 480  0          130        0
Se0/0                     502  end-to-end 480  4378       4506       0
Se0/0                     503  end-to-end 480  0          130        0
Se0/0                     504  end-to-end 480  0          130        0
Se0/0                     513  end-to-end 480  0          0          0

Rack1R2#show frame-relay fragment
interface                 dlci frag-type  size in-frag    out-frag   dropped-
frag
Se0/0                     205  end-to-end 480  4814       4837       0

Rack1R3#ping 155.1.0.5 size 500

Type escape sequence to abort.
Sending 5, 500-byte ICMP Echos to 155.1.0.5, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 260/271/316 ms

Rack1R3#show frame-relay fragment
interface                 dlci frag-type  size in-frag    out-frag   dropped-
frag
Se1/0                     301  end-to-end 480  0          0          0
Se1/0                     302  end-to-end 480  0          0          0
Se1/0                     304  end-to-end 480  0          0          0
Se1/0                     305  end-to-end 480  328        10         0
```

Note that we had to stream large ICMP packets off R3 to increase the fragment counters, as the RIP updates are to small to fragment them.

10.52 MQC Based Frame-Relay Traffic Shaping

- Modify the traffic-shaping configuration on R5 to remove the classes matching the Frame-Relay DLCIs.
- Configure MQC shaping on R5 to apply per-VC as follows, but do not enable legacy FRTS on the main interface
  - Shape peak with CIR values 128000 and 256000 for DLCI 503 and DLCI 502
  - Select a B_c value based on a T_c of 50ms, and a B_e value so that PIR=2*CIR
  - Set the Frame-Relay DE bit for traffic going out DLCI 502
- Replace the interface-based fragmentation with per-VC fragmentation.

Configuration

```
R5:
policy-map SHAPE_DLCI_503
 class class-default
  shape peak 128000 6400 6400
  shape adaptive 128000
!
policy-map SHAPE_DLCI_502
 class class-default
  shape peak 256000 12800 12800
  shape adaptive 256000
  set fr-de
!
! MQC based FRTS still uses map-classes
! but applies shaping using CBTS commands
! inside a service-policy.
! Fragmentation also applies inside a map-class
!
map-class frame-relay DLCI_503
 frame-relay fragment 480
 service-policy output SHAPE_DLCI_503
!
map-class frame-relay DLCI_502
 frame-relay fragment 480
 service-policy output SHAPE_DLCI_502
!
! We no longer configure fragmentation at the interface level
!
interface Serial 0/0
 no frame-relay fragment 480 end-to-end
 no service-policy output POLICY_FR_OUT
 frame-relay interface-dlci 503
  class DLCI_503
 frame-relay interface-dlci 502
  class DLCI_502
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Verification

✐ **Note**

MQC based Frame-Relay traffic shaping is a hybrid form of legacy FRTS and Class-Based Traffic Shaping (CBTS). Instead of using MQC-based syntax for classification, it relies on map-classes bound to the interface's DLCIs. MQC statements specify the shaping parameters, such as average/peak rate. Additionally, it is possible to enable Frame-Relay adaptive traffic shaping and FECN reflection, which CBTS does not support. Additionally, you should not use the **frame-relay traffic-shaping** interface level command with the MQC-based FRTS.

As in the case of CBTS, you may adjust shaper queue by applying a second-level policy map with CBWFQ parameters. This is similar to the use of CBWFQ with legacy FRTS but this time you use MQC syntax to define shaping parameters.

In this scenario, we oversubscribe both PVCs, thus we use the **shape peak** command combined with **shape adaptive** to configure a PVC to send at PIR, while ensuring it throttles down to CIR in case of congestion notification.

The verification commands are slightly different from the ones you use to verify legacy FRTS, as seen below.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
Rack1R5#show frame-relay pvc 502

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 502, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

  input pkts 29950         output pkts 46932        in bytes 22814609
  out bytes 11804996       dropped pkts 0           in pkts dropped 0
  out pkts dropped 0                out bytes dropped 0
  in FECN pkts 0           in BECN pkts 0           out FECN pkts 0
  out BECN pkts 0          in DE pkts 0             out DE pkts 45369
  out bcast pkts 18637     out bcast bytes 5888192
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
  pvc create time 1d07h, last time pvc status changed 21:56:46
  fragment type end-to-end fragment size 480
  service policy SHAPE_DLCI_502
 Serial0/0: DLCI 502 -

  Service-policy output: SHAPE_DLCI_502

    Class-map: class-default (match-any)
      26 packets, 8216 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any
      Traffic Shaping
           Target/Average   Byte   Sustain   Excess    Interval  Increment
             Rate           Limit  bits/int  bits/int  (ms)      (bytes)
           512000/256000    3200   12800     12800     50        3200

        Adapt  Queue     Packets   Bytes     Packets   Bytes     Shaping
        Active Depth                         Delayed   Delayed   Active
        BECN   0         1019      976568    0         0         no
      QoS Set
        fr-de
          Packets marked 26
  Output queue size 0/max total 600/drops 0
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
Rack1R5#show frame-relay pvc 503

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 503, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

  input pkts 107786        output pkts 99529        in bytes 9809512
  out bytes 51081671       dropped pkts 0           in pkts dropped 0
  out pkts dropped 0                out bytes dropped 0
  in FECN pkts 0           in BECN pkts 0           out FECN pkts 0
  out BECN pkts 0          in DE pkts 0             out DE pkts 164
  out bcast pkts 18639     out bcast bytes 5888824
  5 minute input rate 5000 bits/sec, 9 packets/sec
  5 minute output rate 57000 bits/sec, 6 packets/sec
  pvc create time 1d07h, last time pvc status changed 21:56:51
  fragment type end-to-end fragment size 480
  service policy SHAPE_DLCI_503
 Serial0/0: DLCI 503 -

  Service-policy output: SHAPE_DLCI_503

    Class-map: class-default (match-any)
      818 packets, 954140 bytes
      5 minute offered rate 30000 bps, drop rate 0 bps
      Match: any
      Traffic Shaping
           Target/Average   Byte   Sustain   Excess    Interval  Increment
             Rate           Limit  bits/int  bits/int  (ms)      (bytes)
           256000/128000    1600   6400      6400      50        1600

        Adapt  Queue     Packets   Bytes     Packets   Bytes     Shaping
        Active Depth                         Delayed   Delayed   Active
        BECN   0         0         0         0         0         no
  Output queue size 0/max total 600/drops 0
```

Note the CIR/PIR values in this output, as well as “Adapt Active” field, which tells that PVCs will respond to BECN notifications. FRF.12 fragmentation can be verified as follows.

```
Rack1R5#show frame-relay fragment
interface                dlci frag-type  size in-frag    out-frag   dropped-frag
Se0/0                    502  end-to-end 480  0          734        0
Se0/0                    503  end-to-end 480  0          19         0
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.53 Voice Adaptive Traffic Shaping

- Since the provider oversubscribes both PVCs on R5's interface, voice quality may degrade when sending traffic at rates over the CIR. Since it is undesirable to turn off oversubscription and lose this extra bandwidth, provide a solution to throttle down the sending rates on R3 and R5 to the CIR, but only when the IOS detects traffic in the LLQ queue.
- Leave the shaping time interval at 50ms for simplicity.
- Assume that VoIP traffic is already marked as DSCP EF.
- This traffic should be prioritized up to 64Kbps on R2, R3, and R5.

Configuration

```
R2:
!
! R2 uses legacy FRTS
!
class-map VOICE_BEARER
 match dscp ef
!
policy-map CBWFQ
 class class-default
  bandwidth 128
 class VOICE_BEARER
  priority 64
!
map-class frame-relay DLCI_205
 service-policy output CBWFQ

R3:
!
! R3 uses CBTS for traffic shaping
!
class-map VOICE_BEARER
 match dscp ef
!
policy-map LLQ
 class VOICE_BEARER
  priority 64
!
policy-map POLICY_FR_OUT
 class DLCI_305
  shape average 256000 12800 6400
  service-policy LLQ
```

Copyright © 2009 Internetwork Expert

```
R5:
!
! R5 uses MQC-based FRTS, VATS applies here
!
class-map VOICE_BEARER
 match dscp ef
!
policy-map LLQ
 class VOICE_BEARER
  priority 64
!
! Add LLQ to shaper queue
! Enable VATS in the class
!
policy-map SHAPE_DLCI_503
 class class-default
  shape peak 128000 6400 6400
  shape adaptive 128000
  shape fr-voice-adapt
  service-policy LLQ
!
policy-map SHAPE_DLCI_502
 class class-default
  shape peak 256000 12800 12800
  shape adaptive 256000
  shape fr-voice-adapt
  service-policy LLQ
```

Verification

Note

Voice Adaptive Traffic Shaping (VATS) is only available with MQC-based Frame-Relay traffic shaping, and only with LLQ enabled in shaper-queue. Thus, in our scenario, only R5 is capable of doing VATS.

The goal of VATS is to throttle the PVC's sending rate once the system detects voice packets in the priority queue. The VATS feature becomes useful in oversubscription scenarios, where the customer constantly sends traffic over the CIR. In cases when the LLQ enqueues voice packets, the system signals the shaper to slow its sending rate. This ensures that router only sends at the guaranteed rate, and the provider will never drop or delay any voice packets.

The syntax for the command applied under policy-map class is **shape fr-voice-adapt [deactivation** <*N*>**]**, where *N* specifies the number of seconds required to deactivate shaping after the last packet has been seen in priority queue (LLQ). The default value is 30 seconds.

To verify this scenario, adjust the PIR for DLCI 503 to 56Kbps, and the CIR to 32Kbps. This will match them properly against the connection's physical rate of

64Kbps. Next, generate a flow of ICMP packets marked with DSCP EF from SW2 to R3 to simulate voice traffic. Note that the TOS byte of 184 corresponds to DSCP 46 (EF). In addition to that, start transferring the IOS image from R5 down to R3 to cause congestion in the shaper.

```
R5:
policy-map LLQ
 class VOICE_BEARER
  priority 32
!
policy-map SHAPE_DLCI_503
 class class-default
  shape peak 32000 6400 4800
  shape adaptive 32000
  shape fr-voice-adapt
  service-policy LLQ

Rack1SW2#ping
Protocol [ip]:
Target IP address: 155.1.0.3
Repeat count [5]: 10000000
Datagram size [100]: 60
Timeout in seconds [2]: 1
Extended commands [n]: y
Source address or interface:
Type of service [0]: 184
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 10000000, 60-byte ICMP Echos to 155.1.0.3, timeout is 1
seconds:
!!!!!!!!!!

R5:
ip http server
ip http path flash:

Rack1R3#copy http://admin:cisco@155.1.0.5/c2600-adventerprisek9-mz.124-
10.bin null:
Loading http://***********@155.1.0.5/c2600-adventerprisek9-mz.124-
10.bin !!!
```

Now check the policy-map statistics on R5. Note that the shaper's offered rate is close to 32Kbps, since the system activated VATS on this PVC. Every new voice packet refreshes VATS times back to 30 seconds.

```
Rack1R5#show policy-map interface serial 0/0
 Serial0/0: DLCI 503 -

  Service-policy output: SHAPE_DLCI_503

    Class-map: class-default (match-any)
      737754 packets, 73689969 bytes
      5 minute offered rate 32000 bps, drop rate 0 bps
      Match: any
      Traffic Shaping
           Target/Average   Byte   Sustain   Excess    Interval  Increment
             Rate           Limit  bits/int  bits/int  (ms)      (bytes)
            56000/32000     1400   6400      4800      200       1400

        Adapt  Queue     Packets   Bytes     Packets   Bytes     Shaping
        Active Depth                         Delayed   Delayed   Active
        BECN   2         737752    73688401  3132      267876    yes
        Voice Adaptive Shaping active, time left 29 secs

      Service-policy : LLQ

        Class-map: VOICE_BEARER (match-all)
          555731 packets, 35566784 bytes
          5 minute offered rate 12000 bps, drop rate 0 bps
          Match:  dscp ef (46)
          Queueing
            Strict Priority
            Output Queue: Conversation 24
            Bandwidth 32 (kbps) Burst 800 (Bytes)
            (pkts matched/bytes matched) 555731/35566784
            (total drops/bytes drops) 0/0

        Class-map: class-default (match-any)
          182023 packets, 38123185 bytes
          5 minute offered rate 0 bps, drop rate 0 bps
          Match: any
```

Now stop the flow of ICMP packets from SW2 (this flow simulated the voice traffic), and wait for the system to deactivate VATS. Observe shaping statistics once again after this delay:

```
Rack1R5#show policy-map interface serial 0/0
 Serial0/0: DLCI 503 -

  Service-policy output: SHAPE_DLCI_503

    Class-map: class-default (match-any)
      739663 packets, 75354437 bytes
      5 minute offered rate 53000 bps, drop rate 0 bps
      Match: any
      Traffic Shaping
           Target/Average   Byte   Sustain   Excess    Interval  Increment
             Rate           Limit  bits/int  bits/int  (ms)      (bytes)
            56000/32000     1400   6400      4800      200       1400

        Adapt  Queue     Packets   Bytes     Packets   Bytes     Shaping
        Active Depth                         Delayed   Delayed   Active
        BECN   2         739660    75352301  4898      1747468   yes
        Voice Adaptive Shaping inactive

      Service-policy : LLQ

        Class-map: VOICE_BEARER (match-all)
          556296 packets, 35602944 bytes
          5 minute offered rate 0 bps, drop rate 0 bps
          Match:  dscp ef (46)
          Queueing
            Strict Priority
            Output Queue: Conversation 24
            Bandwidth 32 (kbps) Burst 800 (Bytes)
            (pkts matched/bytes matched) 556296/35602944
            (total drops/bytes drops) 0/0

        Class-map: class-default (match-any)
          183367 packets, 39751493 bytes
          5 minute offered rate 53000 bps, drop rate 0 bps
          Match: any
```

Note that the shaping rate is now getting close to 56Kbps, since VATS is inactive. As soon as voice packets are back, the system will return to shaping back to CIR.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.54 Voice Adaptive Fragmentation

- Configure R5 to activate fragmentation, but only if voice packets are present in the LLQ queue.

Configuration

```
R5:
interface Serial 0/0
 frame-relay fragmentation voice-adaptive
```

Verification

Note

Fragmentation is useful only in the presence of VoIP packets – otherwise it just wastes router resources and link bandwidth due to additional overhead. Voice Adaptive Fragmentation (VAF) allows fragmenting of large data packets only in the presence of voice packets in the LLQ. Based on this the VAF feature helps in situations when processing a high load of fragmented packets becomes CPU intensive.

VAF only works with MQC-based FRTS, and supports fragmentation configured in a map-class (legacy style) or at the interface level (MQC-compatible style). The syntax for interface-level command is **frame-relay fragmentation voice-adaptive [deactivation** <*N*>**]**, where *N* specifies the number of seconds required to deactivate fragmentation after the last packet has been seen in priority queue (LLQ). The default value is 30 seconds.

To verify this scenario adjust the PIR for DLCI 503 to 56Kbps and the CIR to 32Kbps. This will match them properly against the connection’s physical rate of 64Kbps. After that, generate a flow of ICMP packets marked with DSCP EF from SW2 to R3 to simulate voice traffic. Note that TOS byte of 184 corresponds to DSCP 46 (EF).

```
R5:
policy-map LLQ
 class VOICE_BEARER
  priority 32
!
policy-map SHAPE_DLCI_503
 class class-default
  shape peak 32000 6400 4800
  shape adaptive 32000
!
map-class frame-relay DLCI_503
 frame-relay fragment 80
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
Rack1SW2#ping
Protocol [ip]:
Target IP address: 155.1.0.3
Repeat count [5]: 10000000
Datagram size [100]: 60
Timeout in seconds [2]: 1
Extended commands [n]: y
Source address or interface:
Type of service [0]: 184
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 10000000, 60-byte ICMP Echos to 155.1.0.3, timeout is 1 seconds:
!!!!!!!!!!!
```

Check that VAF is active on PVC 503 now:

```
Rack1R5#show frame-relay pvc 503

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 503, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

  input pkts 769006        output pkts 767494       in bytes 74548828
  out bytes 87236829       dropped pkts 0           in pkts dropped 0
  out pkts dropped 0                out bytes dropped 0
  in FECN pkts 0           in BECN pkts 0           out FECN pkts 0
  out BECN pkts 0          in DE pkts 0             out DE pkts 0
  out bcast pkts 9361      out bcast bytes 2957736
  5 minute input rate 12000 bits/sec, 24 packets/sec
  5 minute output rate 13000 bits/sec, 24 packets/sec
  pvc create time 12:01:37, last time pvc status changed 12:01:37
  fragment type end-to-end fragment size 80 adaptive active, time left 29 secs
```

Now reset the Frame-Relay statistics and send packets larger than the configured fragment size across the PVC. Verify that the system fragmented the packets:

```
Rack1R5#clear frame-relay counter
Rack1R5#show frame-relay fragment
interface                dlci frag-type  size in-frag    out-frag   dropped-frag
Se0/0                    502  end-to-end 480  0          2          0
Se0/0                    503  end-to-end 80   0          9          0

Rack1R5#ping 155.1.0.3 repeat 100

Type escape sequence to abort.
Sending 100, 100-byte ICMP Echos to 155.1.0.3, timeout is 2 seconds:
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Success rate is 100 percent (100/100), round-trip min/avg/max = 56/64/368 ms
```

```
Rack1R5#show frame-relay fragment
interface                  dlci frag-type  size in-frag    out-frag   dropped-
frag
Se0/0                      502  end-to-end 480  0          6          0
Se0/0                      503  end-to-end 80   0          227        0
```

Now stop the flow of ICMP packets from SW2 (this flow simulated the voice traffic) and wait for the system to deactivate VAF. Observe fragments statistics again:

```
Rack1R5#show frame-relay pvc 503

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

DLCI = 503, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0

  input pkts 1040          output pkts 1135         in bytes 95088
  out bytes 120326         dropped pkts 0           in pkts dropped 0
  out pkts dropped 0                out bytes dropped 0
  in FECN pkts 0           in BECN pkts 0           out FECN pkts 0
  out BECN pkts 0          in DE pkts 0             out DE pkts 0
  out bcast pkts 168       out bcast bytes 53088
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
  pvc create time 12:15:38, last time pvc status changed 12:15:38
  fragment type end-to-end fragment size 80 adaptive inactive

Rack1R5#show frame-relay fragment
interface                  dlci frag-type  size in-frag    out-frag   dropped-
frag
Se0/0                      502  end-to-end 480  0          0          0
Se0/0                      503  end-to-end 80   0          0          0

Rack1R5#ping 155.1.0.3 repeat 100

Type escape sequence to abort.
Sending 100, 100-byte ICMP Echos to 155.1.0.3, timeout is 2 seconds:
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Success rate is 100 percent (100/100), round-trip min/avg/max = 56/59/81 ms
Rack1R5#show frame-relay fragment
interface                  dlci frag-type  size in-frag    out-frag   dropped-
frag
Se0/0                      502  end-to-end 480  0          0          0
Se0/0                      503  end-to-end 80   0          0          0
```

Based on this we can conclude that fragmentation only occurs when voice packets are present in the LLQ.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.55 MLPPP LFI over Frame Relay

- Configure PPP over Frame Relay on the back-to-back Serial link between R4 and R5 using DLCI 100.
- Configure FRTS on both routers assuming a PVC CIR of 256Kbps, and an interface rate of 512Kbps.
- Create two class-maps named VOICE_BEARER and VOICE_SIGNALING matching the DSCP values of EF and CS3 respectively on both routers.
- Create a policy-map named MLPPP_LLQ and configure it as follows:
    - VOICE_BEARER class should provide enough priority bandwidth to accommodate two G.729 calls (56Kbps)
    - VOICE_SIGNALING class should receive 5% of the remaining bandwidth
    - Class-default should receive 95% of the remaining bandwidth
- Enable PPP multilink and interleave on the virtual-templates and configure delay of 10ms.
- Apply the service-policy to the virtual-templates.

Configuration

```
R4:
class-map match-all VOICE_BEARER
 match  dscp ef
!
class-map match-all VOICE_SIGNALING
 match  dscp cs3
!
! 56Kbps for 2 G.729 calls
!
policy-map MLPPP_LLQ
 class VOICE_BEARER
  priority 56
 class VOICE_SIGNALING
  bandwidth remaining percent 5
 class class-default
  bandwidth remaining percent 95
!
! Use MQC FRTS for traffic-shaping
!
policy-map SHAPE_DLCI_100
 class class-default
  shape average 256000 2560 0
!
map-class frame-relay DLCI_100
 service-policy output SHAPE_DLCI_100
!
interface Serial 0/1
 encapsulation frame-relay
 no keepalive
 no ip address
```

```
 frame-relay interface-dlci 100 ppp Virtual-Template 1
  class DLCI_100

!
! Apply LLQ to the virtual-template
! Adjust bandwidth to 512Kbps
!
interface Virtual-Template 1
 ip address 155.1.45.4 255.255.255.0
 bandwidth 512
 service-policy output MLPPP_LLQ
 ppp multilink
 ppp multilink interleave
 ppp multilink fragment delay 10

R5:
class-map match-all VOICE_BEARER
 match  dscp ef
!
class-map match-all VOICE_SIGNALING
 match  dscp cs3
!
! 56Kbps for voice traffic
!
policy-map MLPPP_LLQ
 class VOICE_BEARER
  priority 56
 class VOICE_SIGNALING
  bandwidth remaining percent 5
 class class-default
  bandwidth remaining percent 95
!
map-class frame-relay DLCI_100
 frame-relay cir 256000
 frame-relay mincir 25600
 frame-relay bc 2560
 frame-relay be 0
!
interface Serial 0/1
 encapsulation frame-relay
 no keepalive
 no ip address
 frame-relay traffic-shaping
 frame-relay interface-dlci 100 ppp Virtual-Template 1
  class DLCI_100
!
! Apply LLQ to the virtual-template
! Adjust bandwidth to 512Kbps
!
interface Virtual-Template 1
 ip address 155.1.45.5 255.255.255.0
 bandwidth 512
 service-policy output MLPPP_LLQ
 ppp multilink
 ppp multilink interleave
 ppp multilink fragment delay 10
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Verification

Note

MLPPP (Multilink PPP) LFI (Link Fragmentation and Interleaving) provides a way to implement a fragmentation and interleaving scheme supported across both Frame-Relay and ATM PVCs. As both Frame-Relay and ATM are considered legacy technologies nowadays, the need for a common fragmentation scheme does not seem to be important. However, many installations still utilize Frame-Relay and ATM interworking schemes, and which is why Cisco IOS still supports this feature.

Keep in mind the following key points about MLPPP.

1) MLPPP requires Frame-Relay traffic shaping for PVCs. Although MLPPP would work without FRTS, it will not provide proper QoS properties, since queuing will not engage at the PPP level.

2) MLPPP requires the LLQ at virtual-interface level. LLQ is essential for interleaving, since it ensures that scheduler services VoIP packets ahead of others.

3) Calculate the fragment size based on interface physical rate, not the PVC CIR. In our case, we simply changed the virtual-template interface bandwidth to match physical rate.

4) PPP performs fragmentation and interleaving using PPP multilink feature. MLPPP never fragments packets smaller than the configured fragments size, but rather send them as regular PPP packets interleaved with fragments of larger packets. The interleaving procedure uses special interface-level dual-FIFO queue, similar to the one used in FRF.12 process.

For voice bandwidth calculation, take into account the MLPPPoFR encapsulation overhead of 9 bytes: 1 byte for FR flag, 2 bytes for Frame-Relay header, 1 byte for PPP LLC Control field, 1 byte for PPP NLPID and 2 bytes for Frame-Relay FCS.

In our case, voice packets are uncompressed G.729 which takes 60 bytes for Payload + IP/UDP/RTP headers, 9 bytes for overhead and have a 50 pps rate. Thus the total Layer 2 bandwidth per call is:

Bandwidth = (60+9)*50*8=27600 bps

Multiply it by 2 to get approximately 56Kbps.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

When provisioning CBWFQ bandwidth values, note that we scaled the bandwidth available on the PVC (CIR) to the physical interface rate. This may require scaling the classes' absolute bandwidth values to match the new interface bandwidth, or simply use bandwidth allocation based on remaining percents. As for voice bandwidth, remember the extra overhead added by MLPPP header. We demonstrated the calculations in configuration section.

To verify your configuration, issue the following commands on R4 side (MQC-based FRTS configuration):

```
Rack1R4#show policy-map interface serial 0/1
 Serial0/1: DLCI 100 -

  Service-policy output: SHAPE_DLCI_100

    Class-map: class-default (match-any)
      19 packets, 2484 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any
      Traffic Shaping
           Target/Average   Byte   Sustain   Excess    Interval  Increment
             Rate           Limit  bits/int  bits/int  (ms)      (bytes)
           256000/256000    320    2560      0         10        320

        Adapt  Queue     Packets   Bytes     Packets   Bytes     Shaping
        Active Depth                         Delayed   Delayed   Active
        -      0         19        2484      0         0         no

Rack1R4#show ppp multilink

Virtual-Access3, bundle name is Rack1R5
  Endpoint discriminator is Rack1R5
  Bundle up for 01:43:46, total bandwidth 512, load 1/255
  Receive buffer limit 12192 bytes, frag timeout 1000 ms
  Interleaving enabled
    0/0 fragments/bytes in reassembly list
    0 lost fragments, 0 reordered
    0/0 discarded fragments/bytes, 0 lost received
    0x2B3 received sequence, 0x2B3 sent sequence
  Member links: 1 (max not set, min not set)
    Vi1, since 01:43:47, 640 weight, 630 frag size
No inactive multilink interfaces

Rack1R4#show policy-map interface virtual-access 3
 Virtual-Access3

  Service-policy output: MLPPP_LLQ

    Class-map: VOICE_BEARER (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match:  dscp ef (46)
      Queueing
        Strict Priority
        Output Queue: Conversation 136
        Bandwidth 56 (kbps) Burst 1400 (Bytes)
        (pkts matched/bytes matched) 0/0
        (total drops/bytes drops) 0/0
```

```
    Class-map: VOICE_SIGNALING (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match:  dscp cs3 (24)
      Queueing
        Output Queue: Conversation 137
        Bandwidth remaining 5 (%)Max Threshold 64 (packets)
        (pkts matched/bytes matched) 0/0
        (depth/total drops/no-buffer drops) 0/0/0

    Class-map: class-default (match-any)
      13 packets, 4082 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any
      Queueing
        Output Queue: Conversation 138
        Bandwidth remaining 95 (%)Max Threshold 64 (packets)
        (pkts matched/bytes matched) 0/0
        (depth/total drops/no-buffer drops) 0/0/0
```

Note the CIR value, fragment size and policy-map attached to virtual bundle interface. The **`show ppp multilink`** command displays the bundle interface “virtual-access3” and the member link “virtual-access1”. The policy map applies to the bundle interface, while member link queue is FIFO.

Now repeat the similar verification on the side configured for Frame-Relay traffic shaping.

```
Rack1R5#show frame-relay pvc 100

PVC Statistics for interface Serial0/1 (Frame Relay DTE)

DLCI = 100, DLCI USAGE = LOCAL, PVC STATUS = STATIC, INTERFACE = Serial0/1

  input pkts 2037          output pkts 2152         in bytes 266496
  out bytes 400154         dropped pkts 0           in pkts dropped 0
  out pkts dropped 0                out bytes dropped 0
  in FECN pkts 0           in BECN pkts 0           out FECN pkts 0
  out BECN pkts 0          in DE pkts 0             out DE pkts 0
  out bcast pkts 154       out bcast bytes 48664
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
  pvc create time 01:59:33, last time pvc status changed 01:59:33
  Bound to Virtual-Access1 (up, cloned from Virtual-Template1)
  cir 256000    bc 2560      be 0         byte limit 320    interval 10
  mincir 25600     byte increment 320   Adaptive Shaping none
  pkts 1998      bytes 351490    pkts delayed 8         bytes delayed 468
  shaping inactive
  traffic shaping drops 0
  Queueing strategy: fifo
  Output queue 0/40, 0 drop, 0 dequeued
```

```
Rack1R5#show ppp multilink

Virtual-Access3, bundle name is Rack1R4
  Endpoint discriminator is Rack1R4
  Bundle up for 01:48:01, total bandwidth 512, load 1/255
  Receive buffer limit 12192 bytes, frag timeout 1000 ms
  Interleaving enabled
    0/0 fragments/bytes in reassembly list
    1 lost fragments, 1 reordered
    0/0 discarded fragments/bytes, 0 lost received
    0x2CE received sequence, 0x2CF sent sequence
  Member links: 1 (max not set, min not set)
    Vi1, since 01:48:01, 640 weight, 630 frag size
No inactive multilink interfaces

Rack1R5#show policy-map interface Virtual-Access 3
 Virtual-Access3

  Service-policy output: MLPPP_LLQ

    Class-map: VOICE_BEARER (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match:  dscp ef (46)
      Queueing
        Strict Priority
        Output Queue: Conversation 136
        Bandwidth 56 (kbps) Burst 1400 (Bytes)
        (pkts matched/bytes matched) 0/0
        (total drops/bytes drops) 0/0

    Class-map: VOICE_SIGNALING (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match:  dscp cs3 (24)
      Queueing
        Output Queue: Conversation 137
        Bandwidth remaining 5 (%)Max Threshold 64 (packets)
        (pkts matched/bytes matched) 0/0
        (depth/total drops/no-buffer drops) 0/0/0

    Class-map: class-default (match-any)
      34 packets, 15436 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any
      Queueing
        Output Queue: Conversation 138
        Bandwidth remaining 95 (%)Max Threshold 64 (packets)
        (pkts matched/bytes matched) 0/0
        (depth/total drops/no-buffer drops) 0/0/0
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Now identify the interface-level interleaving queue, for example on R4. Note that the queue is essentially the same priority queue seen with FRF.12:

```
Rack1R4#show interfaces serial 0/1
Serial0/1 is up, line protocol is up
  Hardware is PowerQUICC Serial
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation FRAME-RELAY, loopback not set
  Keepalive not set
  LMI DLCI 1023  LMI type is CISCO  frame relay DTE
  FR SVC disabled, LAPF state down
  Broadcast queue 0/64, broadcasts sent/dropped 39/0, interface broadcasts 38
  Last input 02:04:14, output 00:00:07, output hang never
  Last clearing of "show interface" counters 02:16:55
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 10
  Queueing strategy: dual fifo
  Output queue: high size/max/dropped 0/256/0
  Output queue: 0/128 (size/max)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     2491 packets input, 462842 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     1 input errors, 0 CRC, 1 frame, 0 overrun, 0 ignored, 0 abort
     2488 packets output, 322180 bytes, 0 underruns
     0 output errors, 0 collisions, 2 interface resets
     0 output buffer failures, 0 output buffers swapped out
     14 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up

Rack1R4#show queueing interface serial 0/1
Interface Serial0/1 queueing strategy: priority

Output queue utilization (queue/count)
        high/0 medium/0 normal/183 low/0
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.56 QoS Pre-Classify

- Create a GRE tunnel between R1 and R6 over VLAN 146.
- Route traffic between the Loopback0 subnets of R1 and R6 across the tunnel using static routes.
- Limit the rate of traffic leaving the VLAN 146 interface of R6 to 256Kbps.
- Configure R6 to guarantee 64Kbps of priority bandwidth to IP traffic between the Loopback0 subnets of R1 and R6 marked with DSCP EF on the VLAN146 interface.

Configuration

```
R1:
interface Tunnel0
 tunnel source 155.1.146.1
 tunnel destination 155.1.146.6
 ip unnumbered FastEthernet 0/0
!
ip route 150.1.6.0 255.255.255.0 Tunnel0

R6:
interface Tunnel0
 tunnel source 155.1.146.6
 tunnel destination 155.1.146.1
 ip unnumbered FastEthernet 0/0.146
 qos pre-classify
!
ip route 150.1.1.0 255.255.255.0 Tunnel0
!
ip access-list extended LOOPBACKS
 permit ip 150.1.6.0 0.0.0.255 150.1.1.0 0.0.0.255
!
class-map LOOPBACKS_DSCP_EF
 match access-group name LOOPBACKS
 match dscp ef
!
policy-map LLQ
 class LOOPBACKS_DSCP_EF
!
policy-map SHAPE_VLAN_146
 class class-default
  shape average 256000
  service-policy LLQ
!
interface FastEthernet 0/0.146
 service-policy output SHAPE_VLAN_146
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Verification

Note

From a QoS classification perspective, traffic routed inside tunnels, such as GRE or IPsec, may pose a serious problem. Imagine a situation where multiple tunnels go across the same physical interface of a router, in addition to non-tunneled traffic leaving the same interface. Additionally you need to limit the aggregate outgoing traffic rate, and share bandwidth between tunneled and non-tunneled traffic. If you apply a service policy to the physical interface, the policy will treat traffic inside the tunnel as a single flow, sourced off the tunnel endpoints addresses. Since the classification takes into account only the top-most header, the system is not able to distinguish traffic flows inside the tunnel. A number of solutions exist to make the router aware of the “tunneled” traffic structure.

1) ToS reflection. When IOS encapsulates an IP packet with a tunnel header, it copies the ToS byte to the tunnel header. This makes possible to distinguish traffic “classes” between the tunnel endpoints. However, specific flows remain “hidden” from the physical interface level policy, limiting the usefulness of flow-based schedulers like WFQ.

2) Applying service-policy to the tunnel interface. IOS allows applying GTS (Generic Traffic Shaping) and CBTS (Class Based Shaping) to GRE/IPIP tunnels. Using CBTS you may shape traffic going across the tunnel and tune the shaper queue the way you want. This method has its limitations. First, not all tunnel technologies support CBTS. For example, you cannot apply CBTS inside an IPsec tunnel or apply it to Multipoint GRE tunnel. More seriously, even if the previous issue could be resolved, you still cannot create one unified aggregate policy for all traffic. That is, you need to limit each tunnel up to its maximum rate and apply CBWFQ inside the tunnel. In addition, you have to configure some policy at the physical interface level, e.g. a bandwidth reservation for tunnel traffic flow. This makes the configuration process complicated.

The last solution is QoS pre-classification. When you turn on this feature on a tunnel interface (GRE/mGRE, IPIP, IPsec, Virtual-Template) you no longer need to apply a service policy inside the tunnel interface. Thanks to QoS pre-classification, the service-policy applied at the interface level can “see” the tunnel encapsulated packets as is they cross the interface without any encapsulation. However, the physical interface level policy still accounts for tunnel header overhead, thus allowing for fair scheduling.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

The QoS pre-classify feature is useful in scenarios similar to the described above, e.g. you need to share the physical interface between tunneled and non-tunneled traffic. With pre-classification you apply the service-policy to the physical interface and enable the feature on all tunnel interfaces, treating all traffic (from a QoS prospective) like it is not encapsulated.

To verify this, generate packets from the Loopback0 of R6 to the Loopback0 of R1, since these packets are tunnel encapsulated.

```
Rack1R6#ping
Protocol [ip]:
Target IP address: 150.1.1.1
Repeat count [5]: 100
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface: 150.1.6.6
Type of service [0]: 184
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 100, 100-byte ICMP Echos to 150.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 150.1.6.6
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Success rate is 100 percent (100/100), round-trip min/avg/max = 1/3/8 ms

Rack1R6#show ip route 150.1.1.1
Routing entry for 150.1.1.0/24
  Known via "static", distance 1, metric 0 (connected)
  Redistributing via rip
  Advertised by rip
  Routing Descriptor Blocks:
  * directly connected, via Tunnel0
      Route metric is 0, traffic share count is 1

Rack1R6#show policy-map interface fastEthernet 0/0.146
 FastEthernet0/0.146

  Service-policy output: SHAPE_VLAN_146

    Class-map: class-default (match-any)
      150 packets, 29092 bytes
      5 minute offered rate 2000 bps, drop rate 0 bps
      Match: any
      Traffic Shaping
           Target/Average   Byte   Sustain   Excess    Interval  Increment
             Rate           Limit  bits/int  bits/int  (ms)      (bytes)
           256000/256000    1984   7936      7936      31        992

        Adapt  Queue     Packets   Bytes     Packets   Bytes     Shaping
        Active Depth                         Delayed   Delayed   Active
        -      0         150       29092     0         0         no
```

```
    Service-policy : LLQ

      Class-map: VOICE (match-all)
        100 packets, 14200 bytes
        5 minute offered rate 2000 bps, drop rate 0 bps
        Match:  dscp ef (46)
        Queueing
          Output Queue: Conversation 41
          Bandwidth 64 (kbps)Max Threshold 64 (packets)
          (pkts matched/bytes matched) 0/0
      (depth/total drops/no-buffer drops) 0/0/0

      Class-map: LOOPBACKS_DSCP_EF (match-all)
        0 packets, 0 bytes
        5 minute offered rate 0 bps
        Match: access-group name LOOPBACKS
        Match:  dscp ef (46)

      Class-map: class-default (match-any)
        50 packets, 14892 bytes
        5 minute offered rate 0 bps, drop rate 0 bps
        Match: any
```

Note the size of the ICMP packets classified by the service-policy. A total of 14200 bytes for 100 packets means a size of 142 bytes for each packet. Subtract 18 bytes for the tagged Ethernet frame, and the remaining 24 bytes are the for GRE encapsulation header. This means that QoS pre-classification correctly accounts for the tunnel overhead.

10.57 RSVP and WFQ

- Shutdown R5's Frame Relay connection.
- Configure the link bandwidth between R4 and R5 as 128Kbps.
- Enable RSVP on the connections to VLAN 146 on R1, R4, and R6.
- Allow RSVP to use the maximum possible bandwidth on the link between R4 and R5, but do not allow it to reserve more than 64Kbps per flow.

Configuration

```
R1:
interface FastEthernet 0/0
 ip rsvp bandwidth

R4:
interface FastEthernet 0/1
 ip rsvp bandwidth
!
interface Serial 0/1
 bandwidth 128
 ip rsvp bandwidth 96 64

R5:
interface Serial 0/1
 bandwidth 128
 clock rate 128000
 ip rsvp bandwidth 96 64
!
interface Serial 0/0
 shutdown

R6:
interface FastEthernet 0/0
 ip rsvp bandwidth
!
interface FastEthernet 0/0.146
 ip rsvp bandwidth
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Verification

Note

RSVP, or Resource Reservation Protocol, is the core of Integrated Services (IntServ) QoS model, also known as the end-to-end QoS model. The general idea is to mimic the behavior of circuit-switched networks, using RSVP as a signaling protocol to request certain QoS capabilities from the network. Compare this to a PSTN cloud, where the network reserves resources for a call for the duration of the conversation and releases them as the call ends. This in contrast with classic packet switched network (PSN) behavior, where the network statistically multiplexes packets and provides per-hop treatment for each packet, irrespective of the “flow” the packet belongs to. The latter behavior evolved into the Differentiated Services (DiffServ) QoS model.

Central to RSVP is the concept of a “flow” – a unidirectional stream of packets. A flow could be either one-to-one or one-to-many - e.g. if a sender streams a multicast flow with many receivers. When a host X wants to stream a flow toward host Y (Y could be a multicast address), it sends RSVP request (PATH message) to the nearest router, asking to establish QoS-aware path downstream to Y. The request contains information describing the flow the sender is going to originate. The router compares the parameters in the message with its available resources and, if everything is okay, forwards the request downstream to the router one hop closer to Y. Eventually, when this chain of requests reaches Y, it responds with a reservation request (RESV message) towards host X.

It is up to the receiver to request specific QoS properties from the network (e.g. specify the bitrate and quality of service class). After that, every router in path checks if it can provide the requested QoS parameters, and forwards the requests upstream to the sender. When sender receives the reservation message, it knows that it can now send data along the QoS-capable path. Of course, each router on the path must pre-configure proper instrumentation to provide the requested QoS parameters. Note that since flows are unidirectional it requires installing two reservations for each direction of the traffic if the packet exchange is duplex.

Within the Integrated Services model, the network may provide the following classes of service: Best Effort, Guaranteed Rate, and Controlled Load. Best effort flows do not require any specific QoS treatment. Guaranteed rate provides a flow with bandwidth and delay guarantee – this is the strictest type of service. Controlled load is the type of service you may expects on a lightly loaded network – it provides you the bandwidth but no guarantees on delay.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

The RSVP reservation request contains two structures called ‘Flowspec’ and ‘Filterspec’. These two structures describe the reservation that receiver requests from the network. The ‘Flowspec’ consists of ‘Rspec’ and ‘Tspec’ structures. ‘Rspec’ (reservation specification) defines the class of services that reservation requests and ‘Tspec’ (traffic specification) defines token bucket parameter for traffic metering – such as average rate and burst size. Note that those structures are the part of the requests sent from the receiver upstream to the sender. RSVP path requests (originated by sender) also contain ‘Tspec’ structure, so that routers on the way may check their resources to see if they can accommodate the new flow’s bitrate.

The ‘Filterspec’ structure defines the senders filter. Essentially, it specifies the sources that are eligible to use the reservation installed by the receiver. Usually, communications are one to one. However, sometimes you may want to send RESV message to multiple senders and make them share a single reservation. For example, this may be the case of many-to-many conference. Three types of filter exist in RSVP: Fixed Filter (FF), Shared Explicit (SE), and Wildcard Filter (WF). FF allows only one explicitly specified source to use the reservation, and the ‘Tspec’ structure parameters (rate, burst etc) apply only to that single flow. If there are multiple senders, the receiver needs to establish a reservation for every one of them. SE allows multiple explicitly specified sources to use the same reservation. The receiver specifies the IP addresses of the sources in the reservation message. WF allows any sender to use the reservation, without explicitly specifying the IP addresses.

Note that RSVP is inherently poorly scalable, in the sense that number of flows grows as N^2 where N is the number of endpoints. However, if the receivers use multicast addressing, RSVP supports feature knows as “reservation merging”. That is, if many recipients sharing the same multicast group create reservations for the same sender, RVSP will merge all reservations into one with the largest bandwidth, and propagate it upstream the tree towards the sender. This feature makes RSVP scalable, when working with multicast. Other enhancements to RSVP also allow scalability by adding “aggregate” reservations; the standard RSVP implementation does not support them.

RSVP is a soft-state protocol – that is, a router would keep an RSVP reservation only when it keeps receiving RSVP PATH/RESV messages. By default, all routers send those messages asynchronously, every 30 seconds. That is, if a router has particular RSVP state, it would keep sending PATH and RESV messages for it. If it does not receive PATH from upstream or RESV from downstream node for some it, the router will time out the state.

RSVP configuration is simple, with only a single command needed to specify the RSVP bandwidth and the per-flow bandwidth using the syntax **ip rsvp bandwidth *<total-bw> <per-flow-bw>***. Note that you need this command to enable RSVP as well as specify bandwidth available to RSVP flows. Without RSVP enabled on interface, a router will not accept nor originate RSVP messages on the interface. If you omit any parameters to the above-described command, the default is to use 75% of interface bandwidth for RSVP and allow the same amount per-flow. Note that you cannot set the RSVP bandwidth to a value higher than 75% of the interface configured bandwidth.

Also note that RSVP needs WFQ/CBWFQ as the interface queue to provide required QoS parameters. On interfaces that do not support WFQ/CBWFQ, RSVP performs admission control only. The system uses 'Filterspec' structure parameters to define traffic classifier for WFQ/CBWFQ flow. Look at this as an enhancement to automatic logic of WFQ flows classification. In fact, RSVP flows are very similar to CBWFQ user-defined classes. They even share the same flow numbers with user-defined classes (refer to the task on CBWFQ bandwidth reservation for details). So what is the formula for WFQ weigh of an RSVP flow? Here is the procedure for weigh calculation:

1) The RSVP flow with maximum bandwidth reservation has WFQ weight of 6.
2) All other flows, have weights based on the formula:

Weight = 6*max_reservation_bw/flow_reservation_bw

Where "max_reservation_bw" is the maximum RSVP bandwidth reservation on the link and "flow_reservation_bw" is the RSVP bandwidth reservation for the flow in question.

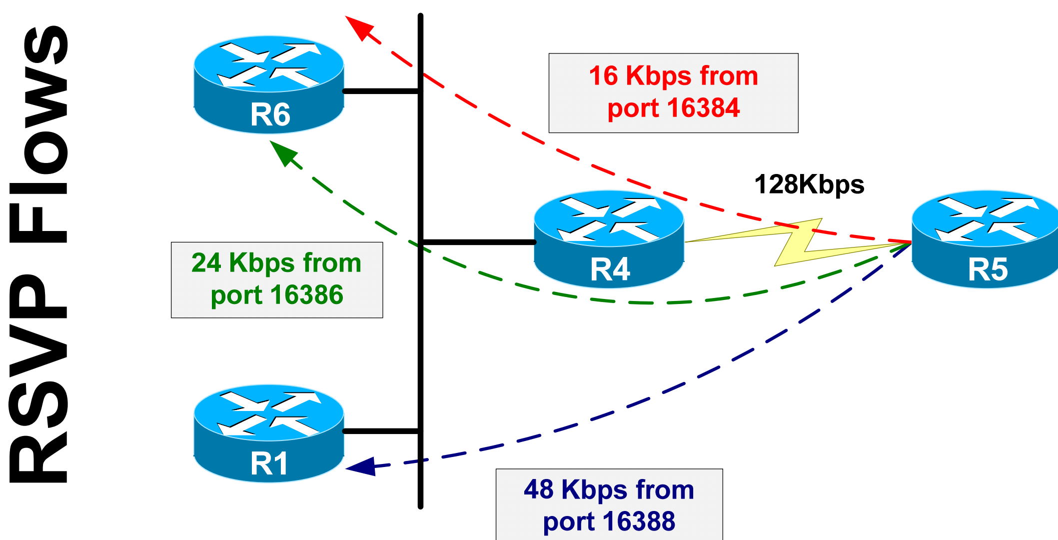
For example if there are two reservations of 12Kbps and 36Kbps, then the second flow will have a WFQ weight of 6, and the first flow will have a WFQ weight of 18.

We see that those weights are small. So how does WFQ ensure that RSVP flows do not monopolize the WFQ bandwidth? The answer is that RSVP also installs special per-flow token-bucket meters, configured in accordance to 'Tspec' parameters. The scheduler serves packets that *exceed* the configured parameters like if they have the default WFQ weight corresponding to IP Precedence of 0 (the highest weight of 32384, lowest scheduling priority).

Note that WFQ subtracts the bandwidth reserved by RSVP from the bandwidth available on the interface, so you may see the approximate amount of bandwidth remaining to other flows on the link.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

To verify the working of RSVP, consider the diagram below. R5 is the sender for 3 RSVP sessions going down to VLAN146 hosts. R6 reserves bandwidth for two sessions of 16Kbps and 24Kbps, while R1 reserves just one session of 48Kbps. Both sessions going to R6 will ask for guaranteed rate and R1 will ask for controlled load services



We source the first flow from UDP port 16384, second flow from UDP port of 16386 and the third flow from UDP port of 16388.

Note that in the configuration below burst sizes are in kilobytes. The burst size is taken as “Rate*1s” which is 6Kbyte for 48Kbps, 3Kbyte for 24Kbps, and 2Kbyte for 16Kbps. You will see the reason for selecting the averaging interval of 1s later, when we start configuring traffic sources.

```
R1:
ip rsvp reservation-host 150.1.1.1 150.1.5.5 UDP 16388 16388 FF LOAD 48 6

R5:
ip rsvp sender-host 150.1.1.1 150.1.5.5 UDP 16388 16388 48 6
ip rsvp sender-host 150.1.6.6 150.1.5.5 UDP 16384 16384 16 2
ip rsvp sender-host 150.1.6.6 150.1.5.5 UDP 16384 16386 24 3

R6:
ip rsvp reservation-host 150.1.6.6 150.1.5.5 UDP 16384 16384 FF RATE 16 2
ip rsvp reservation-host 150.1.6.6 150.1.5.5 UDP 16384 16386 FF RATE 24 3
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Verify if R5 has installed the RSVP reservations from R6 and R6 on the Serial interface to R4. Note the WFQ weights assigned to the flows as well as policing parameters for each flow. Also, look at the interface RSVP status, to see the amount of already allocated bandwidth.

```
Rack1R5#show ip rsvp installed
RSVP: Serial0/1
BPS    To              From            Protoc DPort  Sport  Weight Conversation
48K    150.1.1.1       150.1.5.5       UDP    16388  16388  6      265
16K    150.1.6.6       150.1.5.5       UDP    16384  16384  18     266
24K    150.1.6.6       150.1.5.5       UDP    16384  16386  12     267

Rack1R5#show ip rsvp interface detail

 Se0/1:
   Interface State: Up
   Bandwidth:
     Curr allocated: 88K bits/sec
     Max. allowed (total): 128K bits/sec
     Max. allowed (per flow): 64K bits/sec
     Max. allowed for LSP tunnels using sub-pools: 0 bits/sec
     Set aside by policy (total): 0 bits/sec
   Admission Control:
     Header Compression methods supported:
       rtp (36 bytes-saved), udp (20 bytes-saved)
   Traffic Control:
     RSVP Data Packet Classification is ON via CEF callbacks
   Signalling:
     DSCP value used in RSVP msgs: 0x3F
     Number of refresh intervals to enforce blockade state: 4
     Number of missed refresh messages: 4
     Refresh interval: 30
   Authentication: disabled
```

We have three WFQ flows, with conversation numbers 265, 266 and 267. That means that there are 256 dynamic WFQ flows on the interface. The weights are 6, 12, and 18 for flows that reserved 48K, 24K, and 16K respectively.

Now we configure three traffic sources, to simulate sender traffic. We will use the IP SLA jitter operation to generate bursts of 100-byte packets every 1 second. By sending 2000 bytes every 1 second we would achieve the 16Kbps average rate. Similarly, bursts of 4000 bytes and 6000 bytes result in average rates of 24Kbps and 48Kbps.

Note that IP SLA operation sends UDP packets. The combined size of IP + UDP header is 20 + 8 bytes. The HDLC overhead is 4 bytes. Thus, to get 100 bytes of layer 2 size, we need the payload size of 68 bytes (68 + 28 + 4=100). In addition, we set the inter-packet interval to the minimum 1ms, to generate packet bursts.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

The three operations correspond to the three RSVP reservations:

```
R1& R6:
ip sla monitor responder

R5:
ip sla monitor 1
 type jitter dest-ipaddr 150.1.6.6 dest-port 16384 source-ipaddr 150.1.5.5
source-port 16384 num-packets 20 interval 1
 request-data-size 68
 timeout 500
 frequency 1
ip sla monitor schedule 1 life forever start-time now

ip sla monitor 2
 type jitter dest-ipaddr 150.1.6.6 dest-port 16384 source-ipaddr 150.1.5.5
source-port 16386 num-packets 30 interval 1
 request-data-size 68
 timeout 500
 frequency 1
ip sla monitor schedule 2 life forever start-time now

ip sla monitor 3
 type jitter dest-ipaddr 150.1.1.1 dest-port 16388 source-ipaddr 150.1.5.5
source-port 16388 num-packets 60 interval 1
 request-data-size 68
 timeout 500
 frequency 1
ip sla monitor schedule 3 life forever start-time now
```

Now create a special policy-map on R5 to meter the input rate for RSVP reserved flows. Start transferring the IOS image from R5 to R6 across the slow serial link to see that WFQ guarantees reservations even when with congested link.

```
R4:
ip access-list extended FLOW_1
 permit udp host 150.1.5.5 eq 16384 any
!
ip access-list extended FLOW_2
 permit udp host 150.1.5.5 eq 16386 any
!
ip access-list extended FLOW_3
 permit udp host 150.1.5.5 eq 16388 any
!
class-map match-all FLOW_1
 match access-group name FLOW_1
!
class-map match-all FLOW_3
 match access-group name FLOW_3
!
class-map match-all FLOW_2
 match access-group name FLOW_2
!
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
policy-map METER
 class FLOW_1
 class FLOW_2
 class FLOW_3
!
interface Serial0/1
 service-policy input METER
 load-interval 30

R5:
ip http server
ip htt path flash:
```

```
Rack1R6#$copy http://admin:cisco@150.1.5.5/c2600-adventerprisek9-
mz.124-10.bin null:
Loading http://***********@150.1.5.5/c2600-adventerprisek9-mz.124-
10.bin !!!
```

```
Rack1R4#show policy-map interface serial 0/1
 Serial0/1

  Service-policy input: METER

    Class-map: FLOW_1 (match-all)
      124181 packets, 11182928 bytes
      30 second offered rate 16000 bps
      Match: access-group name FLOW_1

    Class-map: FLOW_2 (match-all)
      151863 packets, 14924736 bytes
      30 second offered rate 24000 bps
      Match: access-group name FLOW_2

    Class-map: FLOW_3 (match-all)
      298729 packets, 29432584 bytes
      30 second offered rate 48000 bps
      Match: access-group name FLOW_3

    Class-map: class-default (match-any)
      1776 packets, 553343 bytes
      30 second offered rate 33000 bps, drop rate 0 bps
      Match: any
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

See that the metered rate is the same that we reserved, even though the HTTP packets overload the link. Now look at the contents of the WFQ queue on R5:

```
Rack1R5#show queueing interface serial 0/1
Interface Serial0/1 queueing strategy: fair
  Input queue: 1/75/0/0 (size/max/drops/flushes); Total output drops:
15470
  Queueing strategy: weighted fair
  Output queue: 27/1000/64/15470 (size/max total/threshold/drops)
     Conversations  2/7/256 (active/max active/max total)
     Reserved Conversations 3/3 (allocated/max allocated)
     Available Bandwidth 8 kilobits/sec

  (depth/weight/total drops/no-buffer drops/interleaves) 20/6/0/0/0
  Conversation 265, linktype: ip, length: 100
  source: 150.1.5.5, destination: 150.1.1.1, id: 0x000D, ttl: 255,
  TOS: 0 prot: 17, source port 16388, destination port 16388

  (depth/weight/total drops/no-buffer drops/interleaves) 7/32384/0/0/0
  Conversation 70, linktype: ip, length: 580
  source: 150.1.5.5, destination: 155.1.146.6, id: 0xA68F, ttl: 255,
  TOS: 0 prot: 6, source port 80, destination port 51793
```

Note that RSVP flow for port 16388 has weigh of 6, since it’s the maximum reservation flow on the interface. Also, note the remaining bandwidth and the number of dynamic flows on the interface. Now look at the detailed statistics for installed RSVP flows:

```
Rack1R5#show ip rsvp installed detail

RSVP: Serial0/1 has the following installed reservations
RSVP Reservation. Destination is 150.1.1.1. Source is 150.1.5.5,
  Protocol is UDP, Destination port is 16388, Source port is 16388
  Traffic Control ID handle: 0B00040E
  Created: 23:52:56 UTC Sat Mar 2 2002
  Admitted flowspec:
    Reserved bandwidth: 48K bits/sec, Maximum burst: 6K bytes, Peak rate: 48K
bits/sec
    Min Policed Unit: 0 bytes, Max Pkt Size: 0 bytes
  Resource provider for this flow:
    WFQ on hw idb Se0/1:  RESERVED queue 265.  Weight: 6, BW 48 kbps
  Conversation supports 1 reservations [0xD000400]
  Data given reserved service: 44760 packets (4296960 bytes)
  Data given best-effort service: 0 packets (0 bytes)
  Reserved traffic classified for 746 seconds
  Long-term average bitrate (bits/sec): 46061 reserved, 0 best-effort
  Policy: INSTALL. Policy source(s): Default

RSVP Reservation. Destination is 150.1.6.6. Source is 150.1.5.5,
  Protocol is UDP, Destination port is 16384, Source port is 16384
  Traffic Control ID handle: 15000409
  Created: 23:52:57 UTC Sat Mar 2 2002
  Admitted flowspec:
    Reserved bandwidth: 16K bits/sec, Maximum burst: 2K bytes, Peak rate: 16K
bits/sec
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
    Min Policed Unit: 0 bytes, Max Pkt Size: 0 bytes
  Resource provider for this flow:
    WFQ on hw idb Se0/1:  RESERVED queue 266.  Weight: 18, BW 16 kbps
  Conversation supports 1 reservations [0x36000402]
  Data given reserved service: 14960 packets (1436160 bytes)
  Data given best-effort service: 0 packets (0 bytes)
  Reserved traffic classified for 748 seconds
  Long-term average bitrate (bits/sec): 15351 reserved, 0 best-effort
  Policy: INSTALL. Policy source(s): Default

RSVP Reservation. Destination is 150.1.6.6. Source is 150.1.5.5,
  Protocol is UDP, Destination port is 16384, Source port is 16386
  Traffic Control ID handle: C100040F
  Created: 23:52:57 UTC Sat Mar 2 2002
  Admitted flowspec:
    Reserved bandwidth: 24K bits/sec, Maximum burst: 3K bytes, Peak rate: 24K
bits/sec
    Min Policed Unit: 0 bytes, Max Pkt Size: 0 bytes
  Resource provider for this flow:
    WFQ on hw idb Se0/1:  RESERVED queue 267.  Weight: 12, BW 24 kbps
  Conversation supports 1 reservations [0x6900040B]
  Data given reserved service: 22530 packets (2162880 bytes)
  Data given best-effort service: 0 packets (0 bytes)
  Reserved traffic classified for 750 seconds
  Long-term average bitrate (bits/sec): 23047 reserved, 0 best-effort
  Policy: INSTALL. Policy source(s): Default
```

Note the long-term average bit-rates for each flow. They are close to the values we set in the reservations: 16K, 24K, and 48K.

Now check RSVP states in R4. Note that “Resource Provider” is set to “None”, and thus the router actually does nothing to provide the required QoS treatment on the FastEthernet interface. However, commonly no one needs that, since high-speed interfaces usually provide more bandwidth than a router can forward, and the congestion points are WAN links.

```
Rack1R4#show ip rsvp installed detail

RSVP: FastEthernet0/1 has the following installed reservations
RSVP Reservation. Destination is 150.1.1.1. Source is 150.1.5.5,
  Protocol is UDP, Destination port is 16388, Source port is 16388
  Traffic Control ID handle: 0400040C
  Created: 18:13:28 UTC Fri Mar 1 2002
  Admitted flowspec:
    Reserved bandwidth: 48K bits/sec, Maximum burst: 6K bytes, Peak rate: 48K
bits/sec
    Min Policed Unit: 0 bytes, Max Pkt Size: 0 bytes
  Resource provider for this flow: None
  Conversation supports 1 reservations [0x36000406]
  Data given reserved service: 47481 packets (4558176 bytes)
  Data given best-effort service: 0 packets (0 bytes)
  Reserved traffic classified for 791 seconds
  Long-term average bitrate (bits/sec): 46073 reserved, 0 best-effort
  Policy: INSTALL. Policy source(s): Default

RSVP Reservation. Destination is 150.1.6.6. Source is 150.1.5.5,
  Protocol is UDP, Destination port is 16384, Source port is 16384
  Traffic Control ID handle: 0A000404
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
  Created: 18:13:13 UTC Fri Mar 1 2002
  Admitted flowspec:
    Reserved bandwidth: 16K bits/sec, Maximum burst: 2K bytes, Peak rate: 16K
bits/sec
    Min Policed Unit: 0 bytes, Max Pkt Size: 0 bytes
  Resource provider for this flow: None
  Conversation supports 1 reservations [0x800040A]
  Data given reserved service: 16161 packets (1551456 bytes)
  Data given best-effort service: 0 packets (0 bytes)
  Reserved traffic classified for 808 seconds
  Long-term average bitrate (bits/sec): 15359 reserved, 0 best-effort
  Policy: INSTALL. Policy source(s): Default

RSVP Reservation. Destination is 150.1.6.6. Source is 150.1.5.5,
  Protocol is UDP, Destination port is 16384, Source port is 16386
  Traffic Control ID handle: 03000403
  Created: 18:13:35 UTC Fri Mar 1 2002
  Admitted flowspec:
    Reserved bandwidth: 24K bits/sec, Maximum burst: 3K bytes, Peak rate: 24K
bits/sec
    Min Policed Unit: 0 bytes, Max Pkt Size: 0 bytes
  Resource provider for this flow: None
  Conversation supports 1 reservations [0x7000402]
  Data given reserved service: 23615 packets (2267040 bytes)
  Data given best-effort service: 0 packets (0 bytes)
  Reserved traffic classified for 787 seconds
  Long-term average bitrate (bits/sec): 23067 reserved, 0 best-effort
  Policy: INSTALL. Policy source(s): Default
```

Finally, observe the flow of RSVP messages on R4. Note the periodic refresh messages for all three flows and the default refresh interval value of 30 seconds.

```
Rack1R4#debug ip rsvp

RSVP 150.1.5.5_16386->150.1.6.6_16384[0.0.0.0]: Received Path message from
155.1.45.5 (on Serial0/1)
RSVP session 150.1.6.6_16384[0.0.0.0]: Received RESV for 150.1.6.6
(FastEthernet0/1) from 155.1.146.6
RSVP 150.1.5.5_16384->150.1.6.6_16384[0.0.0.0]: reservation found--processing
possible change: 848EA964
RSVP 150.1.5.5_16384->150.1.6.6_16384[0.0.0.0]: No change in reservation
RSVP 150.1.5.5_16384->150.1.6.6_16384[0.0.0.0]: Refresh RESV, req=84917B78,
refresh interval=30000mSec [cleanup timer is not awake]
RSVP 150.1.5.5_16384->150.1.6.6_16384[0.0.0.0]: Sending Resv message to
155.1.45.5
RSVP session 150.1.1.1_16388[0.0.0.0]: Received RESV for 150.1.1.1
(FastEthernet0/1) from 155.1.146.1
RSVP 150.1.5.5_16388->150.1.1.1_16388[0.0.0.0]: reservation found--processing
possible change: 848EA824
RSVP 150.1.5.5_16388->150.1.1.1_16388[0.0.0.0]: No change in reservation
RSVP 150.1.5.5_16386->150.1.6.6_16384[0.0.0.0]: Refresh Path psb = 8490C884
refresh interval = 30000mSec
RSVP 150.1.5.5_16386->150.1.6.6_16384[0.0.0.0]: Sending Path message to
155.1.146.6
RSVP session 150.1.6.6_16384[0.0.0.0]: Received RESV for 150.1.6.6
(FastEthernet0/1) from 155.1.146.6
RSVP 150.1.5.5_16386->150.1.6.6_16384[0.0.0.0]: reservation found--processing
possible change: 848EAAA4
RSVP 150.1.5.5_16386->150.1.6.6_16384[0.0.0.0]: No change in reservation
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Finally, let's see how WFQ deals with flows that exceed their reservations. Re-configure the 16Kbps flow to send 32Kbps instead. All you need is double the number of packets send per second. In addition to that, ensure that the file transfer between R5 and R6 is still ongoing.

```
R5:
no ip sla monitor 1
ip sla monitor 1
 type jitter dest-ipaddr 150.1.6.6 dest-port 16384 source-ipaddr 150.1.5.5
source-port 16384 num-packets 40 interval 1
 request-data-size 68
 timeout 500
 frequency 1
ip sla monitor schedule 1 life forever start-time now


Rack1R6#$copy http://admin:cisco@150.1.5.5/c2600-adventerprisek9-mz.124-10.bin
null:
Loading http://***********@150.1.5.5/c2600-adventerprisek9-mz.124-10.bin !!!

Rack1R4#show policy-map interface serial 0/1
 Serial0/1

  Service-policy input: METER

    Class-map: FLOW_1 (match-all)
      1303558 packets, 128241332 bytes
      30 second offered rate 32000 bps
      Match: access-group name FLOW_1

    Class-map: FLOW_2 (match-all)
      1894879 packets, 188349552 bytes
      30 second offered rate 23000 bps
      Match: access-group name FLOW_2

    Class-map: FLOW_3 (match-all)
      3729529 packets, 371635916 bytes
      30 second offered rate 48000 bps
      Match: access-group name FLOW_3

    Class-map: class-default (match-any)
      104206 packets, 55282992 bytes
      30 second offered rate 19000 bps, drop rate 0 bps
      Match: any
```

Now we see that FLOW_1 traffic rate is 32Kbps – just as we configured it. The explanation to that fact is that RSVP scheduler re-assigns the exceeding traffic to a flow queue with a very high weigh 32384. However, this weight is on par with the weight of the HTTP traffic flow. Thus, they share the bandwidth remaining on the link in equal proportions. The bandwidth remaining on the link is approximately 40Kbps, so it's enough for FLOW_1 to get up to 32Kbps.

Let's look at the WFQ queue contents on R5 once again:

```
Rack1R5#show queueing interface serial 0/1
Interface Serial0/1 queueing strategy: fair
  Input queue: 2/75/0/0 (size/max/drops/flushes); Total output drops: 473
  Queueing strategy: Class-based queueing
  Output queue: 17/1000/64/473 (size/max total/threshold/drops)
     Conversations  2/10/256 (active/max active/max total)
     Reserved Conversations 3/3 (allocated/max allocated)
     Available Bandwidth 8 kilobits/sec

  (depth/weight/total drops/no-buffer drops/interleaves) 10/32384/0/0/0
  Conversation 213, linktype: ip, length: 100
  source: 150.1.5.5, destination: 150.1.6.6, id: 0x001F, ttl: 255,
  TOS: 0 prot: 17, source port 16384, destination port 16384

  (depth/weight/total drops/no-buffer drops/interleaves) 6/32384/0/0/0
  Conversation 223, linktype: ip, length: 580
  source: 150.1.5.5, destination: 155.1.146.6, id: 0x0A5C, ttl: 255,
  TOS: 0 prot: 6, source port 80, destination port 54751
```

We see that the snapshot of the queue contents containing flows for "exceeding" part of the violating flow and the flow for HTTP traffic. They both have the same weight of 32384. Now look at the statistics for the RSVP flow and note the exceeding traffic that uses best-effort services:

```
Rack1R5#show ip rsvp installed detail

RSVP: Serial0/1 has the following installed reservations
<snip>

RSVP Reservation. Destination is 150.1.6.6. Source is 150.1.5.5,
  Protocol is UDP, Destination port is 16384, Source port is 16384
  Traffic Control ID handle: 0500040E
  Created: 00:45:33 UTC Fri Mar 1 2002
  Admitted flowspec:
    Reserved bandwidth: 16K bits/sec, Maximum burst: 2K bytes, Peak rate: 16K
bits/sec
    Min Policed Unit: 0 bytes, Max Pkt Size: 0 bytes
  Resource provider for this flow:
    WFQ on hw idb Se0/1:  RESERVED queue 266.  Weight: 18, BW 16 kbps
  Conversation supports 1 reservations [0x600040B]
  Data given reserved service: 17449 packets (1659816 bytes)
  Data given best-effort service: 12071 packets (1156100 bytes)
  Reserved traffic classified for 847 seconds
  Long-term average bitrate (bits/sec): 15677 reserved, 10919 best-effort
  Policy: INSTALL. Policy source(s): Default

<snip>
```

Note that metering is high from being accurate: RSVP considers just 11Kbps as exceeding, instead of 16Kbps. The reason is that we send traffic in bursts of 4Kbyte while we configured the RSVP policer for 2Kbyte bursts. This results in policer metering the rate below its actual value. Adjust the reservation burst sizes and see what happens:

```
R5:
no ip rsvp sender-host 150.1.6.6 150.1.5.5 UDP 16384 16384 16 2
ip rsvp sender-host 150.1.6.6 150.1.5.5 UDP 16384 16384 16 4

R6:
no ip rsvp reservation-host 150.1.6.6 150.1.5.5 UDP 16384 16384 FF RATE 16 2
ip rsvp reservation-host 150.1.6.6 150.1.5.5 UDP 16384 16384 FF RATE 16 4
```

```
Rack1R5#show ip rsvp installed detail

RSVP: Serial0/1 has the following installed reservations
<snip>

RSVP Reservation. Destination is 150.1.6.6. Source is 150.1.5.5,
  Protocol is UDP, Destination port is 16384, Source port is 16384
  Traffic Control ID handle: F200040D
  Created: 01:06:10 UTC Fri Mar 1 2002
  Admitted flowspec:
    Reserved bandwidth: 16K bits/sec, Maximum burst: 4K bytes, Peak rate: 16K
bits/sec
    Min Policed Unit: 0 bytes, Max Pkt Size: 0 bytes
  Resource provider for this flow:
    WFQ on hw idb Se0/1:  RESERVED queue 266.  Weight: 18, BW 16 kbps
  Conversation supports 1 reservations [0x700040E]
  Data given reserved service: 584 packets (56064 bytes)
  Data given best-effort service: 456 packets (43776 bytes)
  Reserved traffic classified for 25 seconds
  Long-term average bitrate (bits/sec): 17407 reserved, 13591 best-effort
  Policy: INSTALL. Policy source(s): Default

<snip>
```

Metering is much more accurate now, when the policer bursts size equals to traffic bursts.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.58 RSVP and SBM

- Configure R1 as the DSBM candidate to coordinate the bandwidth usage on the segment shared by R1, R4, and R6.
- Allocate just 1024Kbps to RSVP flows on this segment
- Configure R1 to inform other routers on the segment that they may not send above 50Kbyte/sec of non-RSVP traffic into the segment. Limit the burst size to 5Kbyte.

Configuration

```
R1:
interface FastEthernet 0/0
 ip rsvp dsbm candidate
 ip rsvp bandwidth 1024
 ip rsvp dsbm non-resv-send-limit rate 50
 ip rsvp dsbm non-resv-send-limit burst 5
```

Verification

Note

RSVP works nicely when you need to reserve bandwidth on the link between two routers. However, on shared segments like Ethernet, one router is not aware of reservations made through another router. This may lead to oversubscription, where the total amount of reserved bandwidth on all routers exceeds the capacity of the shared network. The problem described is only important for topologies that use shared media, such as half-duplex Ethernet or Token-Ring. However, one may imagine a similar situation in a ring-like Ethernet topology built using switches.

To resolve the above problem, one may introduce a special centralized bandwidth management agent on the segment. This agent is RSVP Subnetwork Bandwidth Manager and we call the segment with DSBM (designated SBM) as “managed”. DSBM is one of the many candidate routers on the segment, elected as the SBM. All other RSVP-capable routers on the segment will discover the DSBM and will use it as a “proxy” for RSVP PATH and RESV messages. This way the DSBM will know about all reservations and may perform centralized admission control. The traffic flows will take the direct path of course, and RSVP will install proper bandwidth reservations, if allowed by DSBM.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

There is one another set of parameters, distributed by DSBM to all routers on the segment. It is the amount of non-reservable traffic allowed in the segment. The DSBM specifies average/peak rate for non-RSVP traffic as well as other parameters such as burst size. It's up to the routers on the segment to provide proper treatment and implement this policy. The DSBM's main function is performing admission control, not to implement any data plane policing.

So far, since you may rarely see Ethernet segments using WFQ (though properly configured CBWFQ is a viable queuing solution), RSVP/DSBM performs mainly an admission control function on Ethernet segments. In addition, Cisco IOS does not seem to implement any policing for non-RSVP traffic flowing into the shared segment.

You may configure multiple SBM candidates on a segment, and the one with numerically highest priority wins the election and becomes Designated SBM or DSBM. Note that if a DSBM already exist on a segment, newly configured SBM will not try to preempt it. If two candidate SMBs have the same priority, the one with highest IP address wins.

To verify your configuration, issue the following show commands:

```
Rack1R1#show ip rsvp sbm detail

Interface: FastEthernet0/0
Local Configuration              Current DSBM
  IP Address: 155.1.146.1          IP Address: 155.1.146.1
  DSBM candidate: yes              I Am DSBM: yes
  Priority: 64                     Priority: 64
  Non Resv Send Limit              Non Resv Send Limit
    Rate: 50 Kbytes/sec              Rate: 50 Kbytes/sec
    Burst: 5 Kbytes                  Burst: 5 Kbytes
    Peak: unlimited                  Peak: unlimited
    Min Unit: unlimited              Min Unit: unlimited
    Max Unit: unlimited              Max Unit: unlimited

Rack1R4#show ip rsvp sbm detail

Interface: FastEthernet0/1
Local Configuration              Current DSBM
  IP Address: 155.1.146.4          IP Address: 155.1.146.1
  DSBM candidate: no               I Am DSBM: no
  Priority: 0                      Priority: 64
  Non Resv Send Limit              Non Resv Send Limit
    Rate: unlimited                  Rate: 50 Kbytes/sec
    Burst: unlimited                 Burst: 5 Kbytes
    Peak: unlimited                  Peak: unlimited
    Min Unit: unlimited              Min Unit: unlimited
    Max Unit: unlimited              Max Unit: unlimited
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
Rack1R6#show ip rsvp sbm detail

Interface: FastEthernet0/0.146
Local Configuration             Current DSBM
  IP Address: 155.1.146.6         IP Address: 155.1.146.1
  DSBM candidate: no              I Am DSBM: no
  Priority: 64                    Priority: 64
  Non Resv Send Limit             Non Resv Send Limit
    Rate: unlimited                 Rate: 50 Kbytes/sec
    Burst: unlimited                Burst: 5 Kbytes
    Peak: unlimited                 Peak: unlimited
    Min Unit: unlimited             Min Unit: unlimited
    Max Unit: unlimited             Max Unit: unlimited
```

Verify that R1 is now in the path of all RSVP messages on the shared segment. Note that the reservation originated at R1 itself is not take in account by DSBM:

```
Rack1R1#show ip rsvp installed
RSVP: FastEthernet0/0
BPS    To              From            Protoc DPort  Sport
16K    150.1.6.6       150.1.5.5       UDP    16384  16384
24K    150.1.6.6       150.1.5.5       UDP    16384  16386

Rack1R1#show ip rsvp interface detail fastEthernet 0/0

 Fa0/0:
   Interface State: Up
   Bandwidth:
     Curr allocated: 40K bits/sec
     Max. allowed (total): 1024K bits/sec
     Max. allowed (per flow): 1024K bits/sec
     Max. allowed for LSP tunnels using sub-pools: 0 bits/sec
     Set aside by policy (total): 0 bits/sec
   Admission Control:
     Header Compression methods supported:
       rtp (36 bytes-saved), udp (20 bytes-saved)
   Traffic Control:
     RSVP Data Packet Classification is ON via CEF callbacks
   Signalling:
     DSCP value used in RSVP msgs: 0x3F
     Number of refresh intervals to enforce blockade state: 4
     Number of missed refresh messages: 4
     Refresh interval: 30
   Authentication: disabled
```

10.59 RSVP and CBWFQ

- Configure CBWFQ on the Serial link of R5 to ensure that HTTP server replies have a guarantee of all interface bandwidth not reserved to RSVP flows.
- Allow for 64 dynamic flow queues with CBWFQ.

Configuration

```
R5:
ip access-list extended HTTP
 permit tcp any eq www any
!
class-map HTTP
 match access-group name HTTP
!
policy-map CBWFQ
 class HTTP
  bandwidth remaining percent 100
 class class-default
  fair-queue 64
!
interface Serial 0/1
 no fair-queue
 service-policy output CBWFQ
```

Verification

Note

As previously discussed, RSVP treats all flow traffic exceeding its reservation as traffic with an IP precedence of zero. However, that means it may directly compete with all other best effort traffic. In this case, we want to ensure that HTTP traffic may claim all bandwidth not guaranteed to RSVP flows. To accomplish that, we set the bandwidth reservation for HTTP class to 100%. Simply speaking, we reserve all bandwidth not need by higher priority flows to the HTTP class. Technically speaking, CBWFQ implements this by assigning a weight of 64 to the HTTP class, as outlined below.

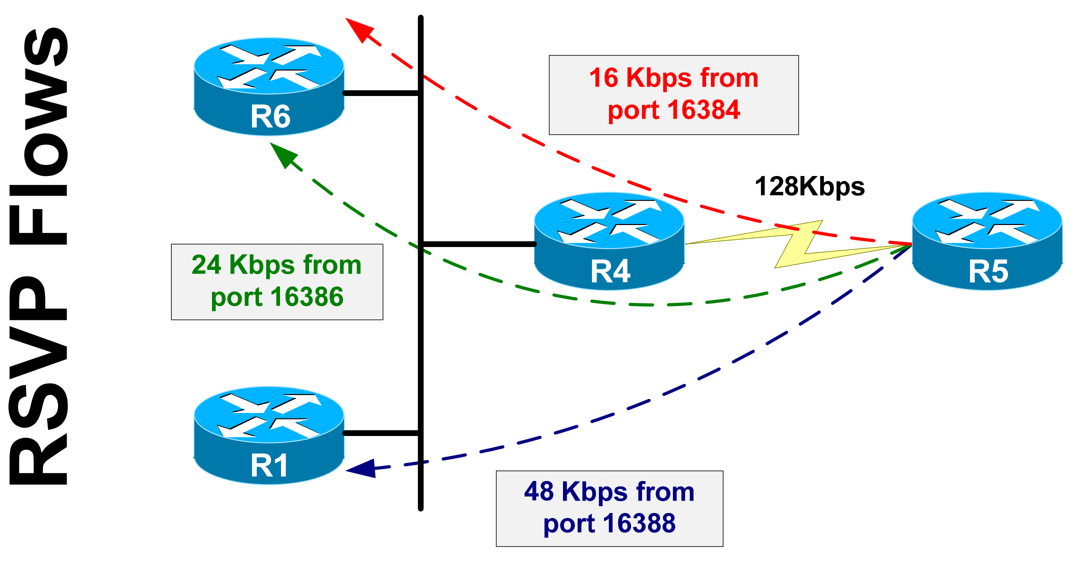
Copyright © 2009 Internetwork Expert

First, we know that CBWFQ is the same as WFQ from the perspective of RSVP. Thus RSVP flows look much like CBWFQ user-defined classes with relatively low weights. However, RSVP flows that exceed their reservation parameters have their treatment changed to “best-effort”. This is the key difference between handling of RSVP flows and CBWFQ user-defined class. A user-defined class with bandwidth reservation may grow up to the full interface bandwidth, if other classes permit. CBWFQ never treats it as best-effort traffic with the lowest priority weight. Thus, exceeding traffic from RVSP flows always “loses” against traffic from a user-defined class, since user-defined traffic always have the same weight.

Secondly, if we have a traffic flow matching the RSVP classifier and a user defined class, the RSVP classifier will take precedence. Even though you may see matches in the policy map, the RSVP reservation will provide QoS services.

Let’s run a simple simulation to see how CBWFQ divides bandwidth between RSVP and user-defined classes. You may want to refer back to task on CBWFQ bandwidth reservation to recall how the system assigns weights to user-defined classes.

Consider the diagram below. R5 is the sender for three RSVP sessions going down to VLAN146 hosts. R6 reserves bandwidth for two sessions of 16Kbps and 24Kbps, while R1 reserves just one session of 48Kbps. Both sessions going to R6 will ask for guaranteed rate and R1 will ask for controlled load services.



We source the first flow from UDP port 16384, second flow from UDP port of 16386, and the third flow from UDP port of 16388. Note that in the configuration below burst sizes are in kilobytes. The burst size is taken as "Rate*1s" which is 6Kbyte for 48Kbps, 3Kbyte for 24Kbps, and 2Kbyte for 16Kbps.

```
R1:
ip rsvp reservation-host 150.1.1.1 150.1.5.5 UDP 16388 16388 FF LOAD 48 6

R5:
ip rsvp sender-host 150.1.1.1 150.1.5.5 UDP 16388 16388 48 6
ip rsvp sender-host 150.1.6.6 150.1.5.5 UDP 16384 16384 16 2
ip rsvp sender-host 150.1.6.6 150.1.5.5 UDP 16384 16386 24 3

R6:
ip rsvp reservation-host 150.1.6.6 150.1.5.5 UDP 16384 16384 FF RATE 16 2
ip rsvp reservation-host 150.1.6.6 150.1.5.5 UDP 16384 16386 FF RATE 24 3
```

Now configure R5 to sends IP SLA jitter probes. See the task on RSVP/WFQ for more explanation on the burst sizes and number of packets in burst. So far, just remember that three SLA probes send packets in bursts separated by 1-second intervals and have average rates of 16K, 24K, and 48K. Note that we disabled the "control" connection, since it may be jammed on the serial link by bulky HTTP file transfer we are going to start between R5 and R6 as well.

```
R1& R6:
ip sla monitor responder

R5:
no ip sla monitor 1
no ip sla monitor 2
no ip sla monitor 3

ip sla monitor 1
 type jitter dest-ipaddr 150.1.6.6 dest-port 16384 source-ipaddr 150.1.5.5
source-port 16384 num-packets 20 interval 1 control disable
 request-data-size 68
 timeout 500
 frequency 1
ip sla monitor schedule 1 life forever start-time now

ip sla monitor 2
 type jitter dest-ipaddr 150.1.6.6 dest-port 16384 source-ipaddr 150.1.5.5
source-port 16386 num-packets 30 interval 1 control disable
 request-data-size 68
 timeout 500
 frequency 1
ip sla monitor schedule 2 life forever start-time now

ip sla monitor 3
 type jitter dest-ipaddr 150.1.1.1 dest-port 16388 source-ipaddr 150.1.5.5
source-port 16388 num-packets 60 interval 1 control disable
 request-data-size 68
 timeout 500
 frequency 1
ip sla monitor schedule 3 life forever start-time now
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
R5:
ip http server
ip htt path flash:
```

```
Rack1R6#copy http://admin:cisco@150.1.5.5/c2600-adventerprisek9-mz.124-10.bin
null:
Loading http://***********@150.1.5.5/c2600-adventerprisek9-mz.124-10.bin !!!
```

Now configure a special policy map on R4 to meter incoming traffic and check the resulting statistics:

```
R4:
ip access-list extended FLOW_1
 permit udp host 150.1.5.5 eq 16384 any
!
ip access-list extended FLOW_2
 permit udp host 150.1.5.5 eq 16386 any
!
ip access-list extended FLOW_3
 permit udp host 150.1.5.5 eq 16388 any
!
class-map match-all FLOW_1
 match access-group name FLOW_1
!
class-map match-all FLOW_3
 match access-group name FLOW_3
!
class-map match-all FLOW_2
 match access-group name FLOW_2
!
policy-map METER
 class FLOW_1
 class FLOW_2
 class FLOW_3
!
interface Serial0/1
 service-policy input METER
 load-interval 30
```

```
Rack1R4#show policy-map interface serial 0/1
 Serial0/1

  Service-policy input: METER

    Class-map: FLOW_1 (match-all)
      1415938 packets, 139293444 bytes
      30 second offered rate 16000 bps
      Match: access-group name FLOW_1

    Class-map: FLOW_2 (match-all)
      1978603 packets, 196699272 bytes
      30 second offered rate 24000 bps
      Match: access-group name FLOW_2

    Class-map: FLOW_3 (match-all)
      3896920 packets, 388329656 bytes
      30 second offered rate 48000 bps
      Match: access-group name FLOW_3

    Class-map: class-default (match-any)
      115838 packets, 61721521 bytes
      30 second offered rate 35000 bps, drop rate 0 bps
      Match: any
```

The above seems to be in accordance with the configuration – all flows get their reserved rates and HTTP gets all the remaining bandwidth which is close to 40Kbps (128-16-24-48), but is definitely less due to various overhead and layer 1 procedures such as HDCL bit-stuffing.

Now let's configure one flow to send above its reserved rate, and see how CBWFQ shares bandwidth in this case. We are going to increase the burst size per second for one flow and change the reservations to reflect that burst size adjustment:

```
R5:
no ip sla monitor 1

ip sla monitor 1
 type jitter dest-ipaddr 150.1.6.6 dest-port 16384 source-ipaddr 150.1.5.5
source-port 16384 num-packets 40 interval 1 control disable
 request-data-size 68
 timeout 500
 frequency 1
ip sla monitor schedule 1 life forever start-time now

R5:
no ip rsvp sender-host 150.1.6.6 150.1.5.5 UDP 16384 16384 16 2
ip rsvp sender-host 150.1.6.6 150.1.5.5 UDP 16384 16384 16 4

R6:
no ip rsvp reservation-host 150.1.6.6 150.1.5.5 UDP 16384 16384 FF RATE 16 2
ip rsvp reservation-host 150.1.6.6 150.1.5.5 UDP 16384 16384 FF RATE 16 4
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Now look at the policy-map statistics on R4. Note that even though FLOW_1 now sends twice as much, it still gets 16Kbps from CBWFQ. This is because RSVP tells CBWFQ to treat exceeding traffic as best effort.

```
Rack1R4#show policy-map interface serial 0/1
 Serial0/1

  Service-policy input: METER

    Class-map: FLOW_1 (match-all)
      1473895 packets, 145089144 bytes
      30 second offered rate 16000 bps
      Match: access-group name FLOW_1

    Class-map: FLOW_2 (match-all)
      2047904 packets, 203629472 bytes
      30 second offered rate 24000 bps
      Match: access-group name FLOW_2

    Class-map: FLOW_3 (match-all)
      4006490 packets, 399286656 bytes
      30 second offered rate 46000 bps
      Match: access-group name FLOW_3

    Class-map: class-default (match-any)
      195068 packets, 107394026 bytes
      30 second offered rate 37000 bps, drop rate 0 bps
      Match: any
```

Look at the CBWFQ queue contents to see the flow weights:

```
Rack1R5#show queueing interface serial 0/1
Interface Serial0/1 queueing strategy: fair
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 292205
  Queueing strategy: Class-based queueing
  Output queue: 142/1000/64/292205 (size/max total/threshold/drops)
     Conversations  8/9/256 (active/max active/max total)
     Reserved Conversations 4/4 (allocated/max allocated)
     Available Bandwidth 8 kilobits/sec

  (depth/weight/total drops/no-buffer drops/interleaves) 3/12/0/0/0
  Conversation 267, linktype: ip, length: 100
  source: 150.1.5.5, destination: 150.1.6.6, id: 0x001B, ttl: 255,
  TOS: 0 prot: 17, source port 16386, destination port 16384

  (depth/weight/total drops/no-buffer drops/interleaves) 36/6/0/0/0
  Conversation 268, linktype: ip, length: 100
  source: 150.1.5.5, destination: 150.1.1.1, id: 0x0018, ttl: 255,
  TOS: 0 prot: 17, source port 16388, destination port 16388

  (depth/weight/total drops/no-buffer drops/interleaves) 22/18/0/0/0
  Conversation 266, linktype: ip, length: 100
  source: 150.1.5.5, destination: 150.1.6.6, id: 0x0012, ttl: 255,
  TOS: 0 prot: 17, source port 16384, destination port 16384
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
(depth/weight/total drops/no-buffer drops/interleaves) 7/16/0/0/0
Conversation 265, linktype: ip, length: 580
source: 150.1.5.5, destination: 155.1.146.6, id: 0x0D42, ttl: 255,
TOS: 0 prot: 6, source port 80, destination port 35205

(depth/weight/total drops/no-buffer drops/interleaves) 1/1024/0/0/0
Conversation 263, linktype: ip, length: 216
source: 155.1.45.5, destination: 224.0.0.9, id: 0x0000, ttl: 2,
TOS: 192 prot: 17, source port 520, destination port 520

(depth/weight/total drops/no-buffer drops/interleaves) 8/32384/0/0/0
Conversation 257, linktype: cdp, length: 328

(depth/weight/total drops/no-buffer drops/interleaves) 44/32384/15805/0/0
Conversation 213, linktype: ip, length: 100
source: 150.1.5.5, destination: 150.1.6.6, id: 0x000D, ttl: 255,
TOS: 0 prot: 17, source port 16384, destination port 16384

(depth/weight/total drops/no-buffer drops/interleaves) 20/32384/6485/0/0
Conversation 211, linktype: ip, length: 100
source: 150.1.5.5, destination: 150.1.1.1, id: 0x0008, ttl: 255,
TOS: 0 prot: 17, source port 16388, destination port 16388
```

The three highlighted flows are the conforming FLOW_1, HTTP traffic flow (which has CBWFQ weight maximum for user-defined classes, since it has 100% of bandwidth), and lastly the exceeding part of FLOW_1. Since HTTP has a better weight than the exceeding part of FLOW_1, it claims all the available bandwidth.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Now stop the HTTP file transfer, and check flow rates once again:

```
Rack1R4#show policy-map interface serial 0/1
 Serial0/1

  Service-policy input: METER

    Class-map: FLOW_1 (match-all)
      1495428 packets, 147242444 bytes
      30 second offered rate 32000 bps
      Match: access-group name FLOW_1

    Class-map: FLOW_2 (match-all)
      2077330 packets, 206572072 bytes
      30 second offered rate 24000 bps
      Match: access-group name FLOW_2

    Class-map: FLOW_3 (match-all)
      4065348 packets, 405172456 bytes
      30 second offered rate 48000 bps
      Match: access-group name FLOW_3

    Class-map: class-default (match-any)
      202138 packets, 111410986 bytes
      30 second offered rate 3000 bps, drop rate 0 bps
      Match: any
```

This time, FLOW_1 takes all the bandwidth it needs, even though it exceeds the reserved bandwidth.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.60 RSVP and LLQ

- Configure R5 to classify RSVP flows with a PQ-profile that ignores peak-rate, and classifies all flows below 2000Kbyte/s with the burst sizes below or equal to 2000 bytes as eligible for LLQ.

Configuration

```
R5:
ip rsvp pq-profile 2000 2000 ignore-peak-value
```

Verification

Note

The RSVP PQ profile defines parameters for RSVP flows that should use the CBWFQ priority queue (LLQ). The parameters are maximum rate in bytes per second, maximum burst size in bytes, and peak-to-average rate ratio in percents:

```
ip rsvp pq-profile <max-rate> <max-burst> <peak-to-avg-ratio>
```

If you configure **`ip rsvp pq-profile voice-like`** then the values are 12288 byte/s, burst 592 bytes, and 110%, which are the defaults.

RSVP if often used with VoIP applications to reserve resources for voice bearer flows. However, the WFQ services do not provide priority treatment for RSVP flows and thus do not guarantee low jitter values. This may result in lower voice quality, compared to the use of LLQ.

The RSVP PQ-profile enhancement allows the RSVP classifier to direct certain flows into the CBWFQ priority queue (LLQ). RSVP makes decisions based on the ‘flowspec’ structure of the reservation. If the parameters meet certain requirements specified in RSVP PQ-profile then the respective flow’s traffic uses LLQ. However, unlike the LLQ policer, RSVP does not drop the exceeding packets, but rather marks them for best-effort services. Thus you don’t need to configure any class in the CBWFQ policy with “priority” keyword – RSVP will classify flows automatically and will use their ‘flowspecs’ to define the policers.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

When a router receives an RSVP reservation flowspec, it compares the parameters of the flow with the configured PQ-profile. If the parameters (avg. rate, peak rate, burst size) are less than or equal to the values specified in the global PQ profile, the RSVP scheduler considers the flow to be voice-like and directs it to LLQ for service.

To verify the feature, configure R5 to reserve bandwidth for three traffic flows, just as we did in previous tasks. Note that only the flow sourced off the port 16384 of R5 satisfies the PQ-profile requirements. After that, check the installed RSVP reservations on R5:

```
R1:
ip rsvp reservation-host 150.1.1.1 150.1.5.5 UDP 16388 16388 FF LOAD 48 6

R5:
ip rsvp sender-host 150.1.1.1 150.1.5.5 UDP 16388 16388 48 6
ip rsvp sender-host 150.1.6.6 150.1.5.5 UDP 16384 16384 16 2
ip rsvp sender-host 150.1.6.6 150.1.5.5 UDP 16384 16386 24 3

R6:
ip rsvp reservation-host 150.1.6.6 150.1.5.5 UDP 16384 16384 FF RATE 16 2
ip rsvp reservation-host 150.1.6.6 150.1.5.5 UDP 16384 16386 FF RATE 24 3

Rack1R5#show ip rsvp installed detail
<snip>

RSVP Reservation. Destination is 150.1.6.6. Source is 150.1.5.5,
  Protocol is UDP, Destination port is 16384, Source port is 16384
  Traffic Control ID handle: 0900040B
  Created: 02:09:17 UTC Fri Mar 1 2002
  Admitted flowspec:
    Reserved bandwidth: 16K bits/sec, Maximum burst: 2K bytes, Peak rate: 16K
bits/sec
    Min Policed Unit: 0 bytes, Max Pkt Size: 0 bytes
  Resource provider for this flow:
    WFQ on hw idb Se0/1:  PRIORITY queue 72.  Weight: 0, BW 16 kbps
  Conversation supports 1 reservations [0x1800040C]
<snip>
```

We see that the flow uses the priority queue inside CBWFQ. Now configure R5 to sends IP SLA jitter probes. See the task on RSVP/WFQ for more explanation on the burst sizes and number of packets in burst. So far, just remember that three SLA probes send packets in bursts separated by 1-second intervals and have average rates of 32K, 24K, and 48K. Note that the first source (source port 16384), which corresponds to voice-like RSVP reservation, is deliberately set to source twice as much packet as the reservation expects.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Also note that we disabled the "control" connection, since it may be jammed on the serial link by the bulky HTTP file transfer we are going to start between R5 and R6 as well.

```
R1& R6:
ip sla monitor responder

R5:
no ip sla monitor 1
no ip sla monitor 2
no ip sla monitor 3

ip sla monitor 1
 type jitter dest-ipaddr 150.1.6.6 dest-port 16384 source-ipaddr 150.1.5.5
source-port 16384 num-packets 40 interval 1 control disable
 request-data-size 68
 timeout 500
 frequency 1
ip sla monitor schedule 1 life forever start-time now

ip sla monitor 2
 type jitter dest-ipaddr 150.1.6.6 dest-port 16384 source-ipaddr 150.1.5.5
source-port 16386 num-packets 30 interval 1 control disable
 request-data-size 68
 timeout 500
 frequency 1
ip sla monitor schedule 2 life forever start-time now

ip sla monitor 3
 type jitter dest-ipaddr 150.1.1.1 dest-port 16388 source-ipaddr 150.1.5.5
source-port 16388 num-packets 60 interval 1 control disable
 request-data-size 68
 timeout 500
 frequency 1
ip sla monitor schedule 3 life forever start-time now

R5:
ip http server
ip htt path flash:

Rack1R6#copy http://admin:cisco@150.1.5.5/c2600-adventerprisek9-mz.124-10.bin
null:
Loading http://***********@150.1.5.5/c2600-adventerprisek9-mz.124-10.bin !!!
```

Now configure a special policy map on R4 to meter incoming traffic and check the resulting statistics:

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
R4:
ip access-list extended FLOW_1
 permit udp host 150.1.5.5 eq 16384 any
!
ip access-list extended FLOW_2
 permit udp host 150.1.5.5 eq 16386 any
!
ip access-list extended FLOW_3
 permit udp host 150.1.5.5 eq 16388 any
!
class-map match-all FLOW_1
 match access-group name FLOW_1
!
class-map match-all FLOW_3
 match access-group name FLOW_3
!
class-map match-all FLOW_2
 match access-group name FLOW_2
!
policy-map METER
 class FLOW_1
 class FLOW_2
 class FLOW_3
!
interface Serial0/1
 service-policy input METER
 load-interval 30
```

```
Rack1R4#show policy-map interface serial 0/1
 Serial0/1

  Service-policy input: METER

    Class-map: FLOW_1 (match-all)
      1636262 packets, 161325844 bytes
      30 second offered rate 16000 bps
      Match: access-group name FLOW_1

    Class-map: FLOW_2 (match-all)
      2231819 packets, 222020972 bytes
      30 second offered rate 24000 bps
     Match: access-group name FLOW_2

    Class-map: FLOW_3 (match-all)
      4373557 packets, 435993356 bytes
      30 second offered rate 48000 bps
      Match: access-group name FLOW_3

    Class-map: class-default (match-any)
      228882 packets, 126489279 bytes
      30 second offered rate 34000 bps, drop rate 0 bps
      Match: any
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Even though FLOW_1 uses priority queue on R5, it does not exceed the 16Kbps limit. This is, of course, is thanks to the RSVP policer. Now let's check the CBWFQ queue contents on R5 to confirm that the "voice-like" flow uses LLQ instead of a regular conversation:

```
Rack1R5#show queueing interface serial 0/1
Interface Serial0/1 queueing strategy: fair
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 374535
  Queueing strategy: Class-based queueing
  Output queue: 105/1000/64/374535 (size/max total/threshold/drops)
     Conversations  5/7/64 (active/max active/max total)
     Reserved Conversations 3/3 (allocated/max allocated)
     Available Bandwidth 8 kilobits/sec

  (depth/weight/total drops/no-buffer drops/interleaves) 12/0/0/0/0
  Conversation 72, linktype: ip, length: 100
  source: 150.1.5.5, destination: 150.1.6.6, id: 0x0009, ttl: 255,
  TOS: 0 prot: 17, source port 16384, destination port 16384

  (depth/weight/total drops/no-buffer drops/interleaves) 20/12/0/0/0
  Conversation 75, linktype: ip, length: 100
  source: 150.1.5.5, destination: 150.1.6.6, id: 0x0000, ttl: 255,
  TOS: 0 prot: 17, source port 16386, destination port 16384

  (depth/weight/total drops/no-buffer drops/interleaves) 7/57/0/0/0
  Conversation 73, linktype: ip, length: 580
  source: 150.1.5.5, destination: 155.1.146.6, id: 0x0FDE, ttl: 255,
  TOS: 0 prot: 6, source port 80, destination port 11251

  (depth/weight/total drops/no-buffer drops/interleaves) 64/32384/29363/0/0
  Conversation 21, linktype: ip, length: 100
  source: 150.1.5.5, destination: 150.1.6.6, id: 0x001D, ttl: 255,
  TOS: 0 prot: 17, source port 16384, destination port 16384

  (depth/weight/total drops/no-buffer drops/interleaves) 2/32384/0/0/0
  Conversation 65, linktype: cdp, length: 328
```

From this output we see that some packets of the flow use the priority queue (weight 0) while the exceeding packets have a WFQ weight of 32384 and have the lowest scheduling priority.

Copyright © 2009 Internetwork Expert

10.61 RSVP and Per-VC WFQ

- Configure the link between R4 and R5 for Frame-Relay encapsulation using DLCI 100.
- Enable legacy Frame-Relay traffic shaping on R5 and shape DLCI 100 down to 96Kbps, with the burst size corresponding to time-interval of 10ms.
- Allow RSVP to use up to 96Kbps of interface bandwidth with 64Kbps per-flow.
- Enable WFQ as the VC queue and configure per-VC WFQ as the RSVP resource provider.

Configuration

```
R4:
interface Serial0/1
 bandwidth 128
 encapsulation frame-relay
 load-interval 30
 no keepalive
 fair-queue 64 256 3
 frame-relay map ip 155.1.45.5 100 broadcast
 ip rsvp bandwidth 96 64

R5:
!
! Map class contains shaping parameters
! and defines WFQ as per-VC queue
!
map-class frame-relay DLCI_100
 frame-relay cir 96000
 frame-relay bc 9600
 frame-relay be 0
 frame-relay mincir 96000
 frame-relay fair-queue
!
interface Serial0/1
 bandwidth 128
 encapsulation frame-relay
 no keepalive
 no fair-queue
 clock rate 128000
 frame-relay traffic-shaping
 frame-relay map ip 155.1.45.4 100 broadcast
 frame-relay interface-dlci 100
  class DLCI_100
 ip rsvp bandwidth 96 64
 ip rsvp resource-provider wfq pvc
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Verification

> **Note**
>
> RSVP uses a “resource provider” to implement the QoS policy for flows. The Resource Provider is either a per-interface WFQ or per-VC WFQ, when working with technologies like Frame-Relay. In order to engage per-VC WFQ or CBWFQ you need to enable legacy Frame-Relay traffic shaping (RSVP does not work any other form of traffic shaping) and enable WFQ as per-VC queue. Additionally, you need to set the resource provider to Per-VC WFQ, using the special interface-level command.
>
> Remember that RSVP bandwidth configuration takes place an interface level. Thus, if you want to use a per-VC RSVP bandwidth admission granularity, create a separate sub-interface for each VC. If you have a hub site terminating many spoke VCs that all use RSVP, then consider using the lowest PVC CIR to deduce the RSVP bandwidth value. Otherwise some RSVP flows may reserve more bandwidth than low-speed PVCs may supply. (This process is similar to the bandwidth calculations you do when configuring EIGRP bandwidth pacing).
>
> To verify your configuration, configure three bandwidth reservations as you did in the previous tasks. Keep in mind that we still have the PQ profile configured, and this will affect flow weights:

```
R1:
ip rsvp reservation-host 150.1.1.1 150.1.5.5 UDP 16388 16388 FF LOAD 48 6

R5:
ip rsvp sender-host 150.1.1.1 150.1.5.5 UDP 16388 16388 48 6
ip rsvp sender-host 150.1.6.6 150.1.5.5 UDP 16384 16384 16 2
ip rsvp sender-host 150.1.6.6 150.1.5.5 UDP 16384 16386 24 3

R6:
ip rsvp reservation-host 150.1.6.6 150.1.5.5 UDP 16384 16384 FF RATE 16 2
ip rsvp reservation-host 150.1.6.6 150.1.5.5 UDP 16384 16386 FF RATE 24 3
```

```
Rack1R5#show ip rsvp installed detail

RSVP: Serial0/1 has the following installed reservations
RSVP Reservation. Destination is 150.1.1.1. Source is 150.1.5.5,
  Protocol is UDP, Destination port is 16388, Source port is 16388
  Traffic Control ID handle: 4500040D
  Created: 03:06:15 UTC Fri Mar 1 2002
  Admitted flowspec:
    Reserved bandwidth: 48K bits/sec, Maximum burst: 6K bytes, Peak rate: 48K
bits/sec
    Min Policed Unit: 0 bytes, Max Pkt Size: 0 bytes
  Resource provider for this flow:
    WFQ on FR PVC dlci 100 on Se0/1:  RESERVED queue 25.  Weight: 6, BW 48 kbps
  Conversation supports 1 reservations [0xE700040A]
  Data given reserved service: 67920 packets (6520320 bytes)
  Data given best-effort service: 0 packets (0 bytes)
```

```
  Reserved traffic classified for 1131 seconds
  Long-term average bitrate (bits/sec): 46081 reserved, 0 best-effort
  Policy: INSTALL. Policy source(s): Default

RSVP Reservation. Destination is 150.1.6.6. Source is 150.1.5.5,
  Protocol is UDP, Destination port is 16384, Source port is 16384
  Traffic Control ID handle: A700040F
  Created: 03:06:15 UTC Fri Mar 1 2002
  Admitted flowspec:
    Reserved bandwidth: 16K bits/sec, Maximum burst: 2K bytes, Peak rate: 16K
bits/sec
    Min Policed Unit: 0 bytes, Max Pkt Size: 0 bytes
  Resource provider for this flow:
    WFQ on FR PVC dlci 100 on Se0/1:  PRIORITY queue 24.  Weight: 0, BW 16 kbps
  Conversation supports 1 reservations [0x2200040C]
  Data given reserved service: 23646 packets (2270016 bytes)
  Data given best-effort service: 21714 packets (2084544 bytes)
  Reserved traffic classified for 1134 seconds
  Long-term average bitrate (bits/sec): 16012 reserved, 14703 best-effort
  Policy: INSTALL. Policy source(s): Default

RSVP Reservation. Destination is 150.1.6.6. Source is 150.1.5.5,
  Protocol is UDP, Destination port is 16384, Source port is 16386
  Traffic Control ID handle: 5C000409
  Created: 03:06:15 UTC Fri Mar 1 2002
  Admitted flowspec:
    Reserved bandwidth: 24K bits/sec, Maximum burst: 3K bytes, Peak rate: 24K
bits/sec
    Min Policed Unit: 0 bytes, Max Pkt Size: 0 bytes
  Resource provider for this flow:
    WFQ on FR PVC dlci 100 on Se0/1:  RESERVED queue 26.  Weight: 12, BW 24
kbps
  Conversation supports 1 reservations [0xB00040E]
  Data given reserved service: 34080 packets (3271680 bytes)
  Data given best-effort service: 0 packets (0 bytes)
  Reserved traffic classified for 1136 seconds
  Long-term average bitrate (bits/sec): 23034 reserved, 0 best-effort
  Policy: INSTALL. Policy source(s): Default

Rack1R5#show traffic-shape queue
Traffic queued in shaping queue on Serial0/1 dlci 100
  Queueing strategy: weighted fair
  Queueing Stats: 94/600/64/14893 (size/max total/threshold/drops)
     Conversations  3/6/16 (active/max active/max total)
     Reserved Conversations 2/0 (allocated/max allocated)
     Available Bandwidth 8 kilobits/se
```

We see that flows use WFQ conversations on the PVC corresponding to DLCI 100. The number of reserved conversation is two, since one flow uses priority queue. Note the reserved conversation numbers, which correspond to the number of WFQ flows equal to 16. The IOS picked up this number automatically, based on PVC CIR, when we configured **`fair-queue`** on the PVC.

You may replace the per-VC WFQ with CBWFQ, since RSVP also interoperates with the latter one. For example, configure the following:

```
R5:
ip access-list extended HTTP
 permit tcp any eq www any
!
policy-map CBWFQ
 class HTTP
  bandwidth remaining percent 100
 class class-default
  fair-queue 64
!
map-class frame-relay DLCI_100
 no fair-queue
 service-policy output CBWFQ
```

Verify the installed RSVP flows once again. Note the new conversation numbers, based on the new number of dynamic WFQ flows (64):

```
Rack1R5#show ip rsvp installed detail

RSVP: Serial0/1 has the following installed reservations
RSVP Reservation. Destination is 150.1.1.1. Source is 150.1.5.5,
  Protocol is UDP, Destination port is 16388, Source port is 16388
  Traffic Control ID handle: 4700040D
  Created: 03:31:20 UTC Fri Mar 1 2002
  Admitted flowspec:
    Reserved bandwidth: 48K bits/sec, Maximum burst: 6K bytes, Peak rate: 48K
bits/sec
    Min Policed Unit: 0 bytes, Max Pkt Size: 0 bytes
  Resource provider for this flow:
    WFQ on FR PVC dlci 100 on Se0/1:  RESERVED queue 74.  Weight: 6, BW 48 kbps
  Conversation supports 1 reservations [0x2300040C]
  Data given reserved service: 12960 packets (1244160 bytes)
  Data given best-effort service: 0 packets (0 bytes)
  Reserved traffic classified for 216 seconds
  Long-term average bitrate (bits/sec): 45987 reserved, 0 best-effort
  Policy: INSTALL. Policy source(s): Default

RSVP Reservation. Destination is 150.1.6.6. Source is 150.1.5.5,
  Protocol is UDP, Destination port is 16384, Source port is 16384
  Traffic Control ID handle: A900040F
  Created: 03:31:17 UTC Fri Mar 1 2002
  Admitted flowspec:
    Reserved bandwidth: 16K bits/sec, Maximum burst: 2K bytes, Peak rate: 16K
bits/sec
    Min Policed Unit: 0 bytes, Max Pkt Size: 0 bytes
  Resource provider for this flow:
    WFQ on FR PVC dlci 100 on Se0/1:  PRIORITY queue 72.  Weight: 0, BW 16 kbps
  Conversation supports 1 reservations [0xC00040E]
  Data given reserved service: 4626 packets (444096 bytes)
  Data given best-effort service: 4214 packets (404544 bytes)
  Reserved traffic classified for 220 seconds
  Long-term average bitrate (bits/sec): 16101 reserved, 14667 best-effort
  Policy: INSTALL. Policy source(s): Default
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
RSVP Reservation. Destination is 150.1.6.6. Source is 150.1.5.5,
  Protocol is UDP, Destination port is 16384, Source port is 16386
  Traffic Control ID handle: 34000401
  Created: 03:31:25 UTC Fri Mar 1 2002
  Admitted flowspec:
    Reserved bandwidth: 24K bits/sec, Maximum burst: 3K bytes, Peak rate: 24K
bits/sec
    Min Policed Unit: 0 bytes, Max Pkt Size: 0 bytes
  Resource provider for this flow:
    WFQ on FR PVC dlci 100 on Se0/1:  RESERVED queue 75.  Weight: 12, BW 24
kbps
  Conversation supports 1 reservations [0xE800040A]
  Data given reserved service: 6450 packets (619200 bytes)
  Data given best-effort service: 0 packets (0 bytes)
  Reserved traffic classified for 214 seconds
  Long-term average bitrate (bits/sec): 23082 reserved, 0 best-effort
  Policy: INSTALL. Policy source(s): Default
```

Now you see how RSVP may apply to per-VC queues when using WFQ or CBWFQ as the queuing discipline.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.62 Catalyst QoS Port-Based Classification

- Enable MLS QoS in all switches.
- Configure CoS to DSCP maps on SW1 and SW2 so that CoS 3 maps to DSCP 26 and CoS 5 maps to DSCP EF.
- Ensure CoS 4 maps to DSCP 44 on SW1.
- Configure IP Precedence to DSCP map in SW4 to map IP Precedence values 3 and 5 into DSCP 26 and DSCP EF.
- Trust DSCP values in packets coming from R1's interface connected to SW1.
- Trust IP Precedence on SW4 port connected to R4 and set the default CoS to 2.
- Override CoS to a value of four on the port connected to R5's VLAN58 interface.
- Use 802.1p bits in the Ethernet headers for classification of packets coming from R6; packets on native VLAN should use a CoS value of one.
- Use interface-level commands only to accomplish the task.

Configuration

```
SW1:
mls qos
!
! This map is indexed by CoS values. That is, for CoS 3 take the value
! with offset 3 in this list. Indexes start at zero and end at 7.
! AF31 is DSCP value of 26 and DSCP EFhas DSCP value of 46.
!
! Note that CoS 4 maps to DSCP 44
!
mls qos map cos-dscp 0 8 16 26 44 46 48 56
!
interface FastEthernet 0/1
 mls qos trust dscp
!
! Force CoS value of 4 on all packets. The switch will modify the DSCP
! value according to the CoS to DSCP table and set it to 44
!
interface FastEthernet 0/5
 mls qos cos 4
 mls qos cos override
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
SW2:
mls qos
!
mls qos map cos-dscp 0 8 16 26 32 46 48 56
!
! Trust CoS bits in 802.1p headers. For packets on native VLAN
! (e.g. CDP) apply the CoS value of 1, since they don't have 802.1p
! bits of their own
!
interface FastEthernet 0/6
 mls qos trust cos
 mls qos cos 1

SW3:
mls qos

SW4:
mls qos
!
! IP precedence to DSCP map is similar to CoS to DSCP
!
mls qos map ip-prec-dscp 0 8 16 26 32 46 48 56
!
interface FastEthernet 0/4
 mls qos trust ip-precedence
 mls qos cos 2
```

Verification

Note

Before you can configure any QoS settings in the 3550 or 3560 Catalyst series switches you need to issue the global **`mls qos`** command. Remember that with a clean QoS configuration the default is not to trust any marking, but reset it to zero. In this task, we implement the simplest form of classification based on port-level trusting.

Note the use of QoS tables to attain the desired markings. The trick is that you can infer DSCP/IP Precedence for untagged packets based on the default interface value, or using the CoS override functionality. The mapping tables maps the L2 to L3 mappings thus achieving the desired goal assigning the DSCP 44 marking.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

The Catalyst QoS implementation uses the Differentiated Services model with the following design goals:

1) Edge network nodes classify input packets using locally configured policy and QoS marking found in packets.
2) Edge nodes encode traffic class using a special field in a packet to let the "core" nodes know of the packet class.
3) Core and edge nodes allocate resources and provide services based on the packet class.

A peculiar feature is that each node acts on its own, according to the configured policy. In order for the QoS policy to be consistent end-to-end, a network administrator must ensure that all nodes use the same classification and resource allocation policy. Another feature is that Differentiated Services model scales well, because only edge nodes classify traffic.

The following are options for encoding marking in IP/Ethernet networks:

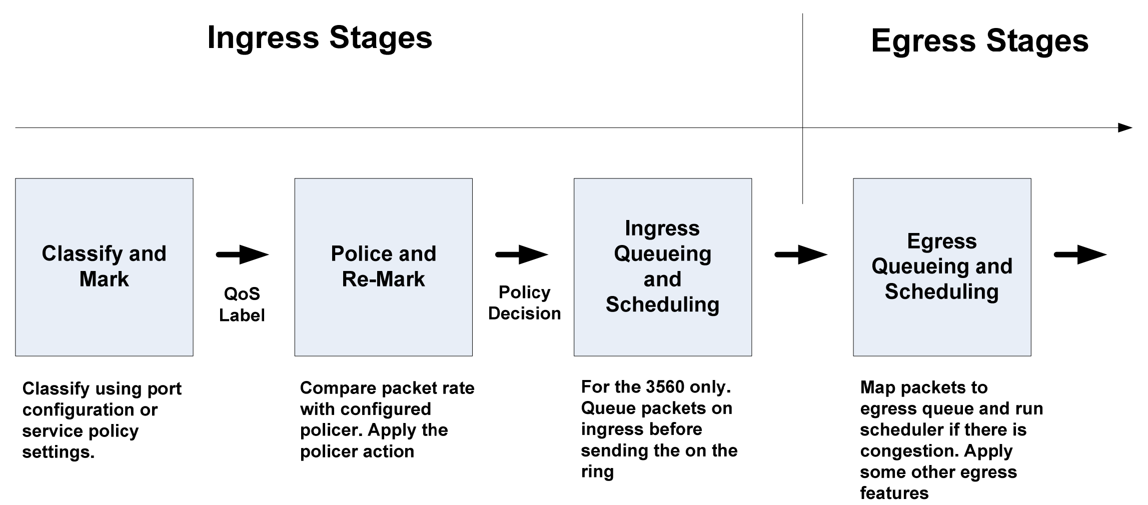
1) ToS byte in IP packet or Traffic Class byte in IPv6 packet. The switch may interpret this field in two different ways:
1.1) Interpret the topmost six bits of the byte as DSCP code point. This is in full compliance with Diff-Serv model.
1.2) Interpret the topmost three bits of the byte as IP Precedence value. This complies with the old, "military-style" QoS model, where higher precedence traffic preempts the lower precedences. Note that the IP Precedence numbers form a subspace of DSCP numbers.

2) The CoS bits (three bits) in 802.1q/ISL header of a tagged Ethernet frame. Those bits are also known as 802.1p priority bits, and should be interpreted under the respective QoS model of relative traffic priorities.

The first two types of marking are Layer 3, and the CoS based marking is Layer 2. The Layer 2 marking is only possible on trunk links. The Catalyst switches have no special intelligence to implement any of the QoS models by themselves. It's up to you to define the policy and encode classes using any marking you find more flexible. Due to the large number of available markings, the Catalyst switches assign a special QoS label to packets and use this label for internal QoS processing. Both switches represent this label as an "internal DSCP" value, associated with a packet.

Keep in mind the distinction that the Catalyst Switches make between IP and non-IP packets. As we know, layer 3 switches are hardware optimized to route IP or IPv6 packets using their ASICs. This results in differences of handling the IP and non-IP packets. Specifically, you may use MAC-based ACLs (MAC addresses, Ether-types, LSAPs) only with non-IP packets (ARP, DECNet, IPX). The MAC ACLs will not match any of IP packets, even if you specify the correct MAC addresses. Also, note that the 3560 models treats IPv6 as “IP” traffic, while the 3550 treats the IPv6 packets as “non-IP” and matches them with MAC ACLs.

The following diagram displays the stage of QoS processing on a typical Catalyst switch.



We are now considering the **Classification and Marking** stage. The switch uses the local policy configuration to classify input packets. The local policy may be as simple as port QoS settings or as complex as policy-maps using QoS ACLs. The result of the classification stage is that the internal QoS label (e.g. internal DSCP value with the 3550) is created. At this stage, the switch extensively uses special globally configurable QoS mapping tables. The tables map one QoS marking to another, e.g. CoS values to DSCP, or DSCP to CoS. The switch uses those tables for “synchronization” of different types of QoS markings. For example, if you instruct the switch to set the CoS field to “3”, the switch will look up the CoS-to-DSCP mapping table and set the DSCP value in the packet header respectively. In addition, the switch uses those tables to translate input marking (e.g. CoS, or IP Precedence) into the Internal DSCP label and vice versa: on egress, the switch maps internal DSCP value to CoS marking if needed.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

You may classify using either interface level settings or by applying a pre-configured policy-map. Those two options are mutually exclusive, i.e. if you configure interface level classification settings and later apply a service-policy, then the latter removes any interface-level QoS classification settings (there are some exceptions though).

The following is the list of the classification options:

1) Trust one of the marking types already found in the packet (**mls qos trust {dscp|ip-precedence|cos}**). For IP packets, it is possible to trust either IP Precedence or DSCP value, and trusting Precedence or DSCP makes sense only for IP packets. If the switch trusts IP markings and the incoming packet is non-IP, the switch will attempt to classify based on the CoS value in the Ethernet header. If there is a CoS value in the packet (i.e. the port is trunk) the switch uses this value to build the QoS label. However, if there is not a CoS field in the packet, the switch will look at the default CoS value configured on the interface (**mls qos cos *x***). The switch builds the corresponding DSCP value for the label using the CoS to DSCP table. When you trust IP precedence, the switch builds the DSCP value using the IP-Precedence to DSCP mapping table and the CoS value using the DSCP to CoS mapping table.

2) Trusting CoS (**mls qos trust cos**) is a little bit different. First, it works for both IP and non-IP packets, since they *both* may carry CoS bits in the Ethernet header. Thus, when the switch trusts CoS on interface, it attempts to build the QoS label based on the CoS bits. If there is no 802.1q/ISL header, the default CoS value from the interface settings is used instead (not the IP/DSCP value from the packet header!). This procedure works for both IP and a non-IP packets, as the switch does not take into account the IP Precedence/DSCP values. The corresponding mapping table allows the switch to compute the internal DSCP value and rewrite it in the IP packet header.

3) You may explicitly assign a specific CoS value to all packet entering the interface using the **mls qos cos override** command. This command will enforce the use of CoS value specified via **mls qos cos *x*** command for all IP and non-IP packets. Note that this only works with CoS marking, and the switch deduces DSCP marking using the CoS to DSCP mapping table. Also any trust configuration on the interface conflicts with the override settings and thus the switch removes it.

4) The most flexible option is to use QoS ACLs to perform classification. These are be covered in a separate task.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

A special type of trust is conditional trust. That means trusting QoS marking only if the switch detects certain device connected to the port. For example, if the switch sees a Cisco IP Phone via CDP or a Cisco SoftPhone sending CDP packets from the attached PC. The command for conditional trust is **mls qos trust device <*device-name*>** and it instructs the switch to trust the packet marking only if the certain device reports itself via CDP.

To verify your configuration, first try using the switch-specific show commands. Start with R6's port, which uses CoS-based classification. The IOS code will by default map DSCP/IP Precedence values to CoS bits using a simple mapping table – highest 3 bits of the ToS byte maps to the CoS value.

```
Rack1SW2#show mls qos interface fastEthernet 0/6 statistics
FastEthernet0/6

  dscp: incoming
-------------------------------

  0 -  4 :           0            0            0            0            0
  5 -  9 :           0            0            0            0            0
 10 - 14 :           0            0            0            0            0
 15 - 19 :           0            0            0            0            0
 20 - 24 :           0            0            0            0            0
 25 - 29 :           0            0            0            0            0
 30 - 34 :           0            0            0            0            0
 35 - 39 :           0            0            0            0            0
 40 - 44 :           0            0            0            0            0
 45 - 49 :           0            0            0           11            0
 50 - 54 :           0            0            0            0            0
 55 - 59 :           0            0            0            0            0
 60 - 64 :           0            0            0            0
  dscp: outgoing
-------------------------------

  0 -  4 :          16            0            0            0            0
  5 -  9 :           0            0            0            0            0
 10 - 14 :           0            0            0            0            0
 15 - 19 :           0            0            0            0            0
 20 - 24 :           0            0            0            0            0
 25 - 29 :           0            0            0            0            0
 30 - 34 :           0            0            0            0            0
 35 - 39 :           0            0            0            0            0
 40 - 44 :           0            0            0            0            0
 45 - 49 :           0            0            0            0            0
 50 - 54 :           0            0            0            0            0
 55 - 59 :           0            0            0            0            0
 60 - 64 :           0            0            0            0
  cos: incoming
-------------------------------

  0 -  4 :          11            5            0            0            0
  5 -  7 :           0            0            0
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
  cos: outgoing
-------------------------------

  0 -  4 :           16            0             0             0             0
  5 -  7 :            0            0             0
Policer: Inprofile:            0 OutofProfile:            0
```

The output consists of two sections: DSCP and CoS values. Those are the values observed in the packets entering leaving the switch interface (i.e. incoming traffic – coming from the router, outgoing traffic – coming to the router). To find statistics for a particular DSCP or CoS value, locate the row corresponding to the value. For example, DSCP 3 maps to the fourth column of the first row:

```
  dscp: incoming
-------------------------------

  0 -  4 :           21            0             0           <0>             0
```

And the CoS value of 2 maps to the third column of the correspond row:

```
  cos: outgoing
-------------------------------

  0 -  4 :         2288            0             0             0             0
```

From the output above, you may note the incoming values corresponding to DSCP 48 (this is the RIP updates, since routing traffic uses IP Precedence of six, which is DSCP 48). Also note, that all incoming routing updates bear CoS value of zero, since IOS does not set the CoS field by default.

Since the switch is set to classify based on CoS it will reset DSCP 48 value in the header to DSCP value of zero. However, the output above does show the DSCP values before the rewrite based on the mapping tables. Note the count of packets with CoS value of “1” – five packets. Those are the Layer 2 keepalives and CDP packets. Since they travel on native VLAN, they have no CoS fields of their own and thus the switch assigns the interface default value to them.

Now configure R6 to mark all outgoing tagged packets with the CoS value of 4, and stream some packets between R6 and R1. To ensure that other switches do not erase our marking, disable MLS QoS on SW1, SW3, and SW4 temporarily. Configure R1 to track packets with DSCP values of CS4 (corresponds to CoS 4 in our CoS to DSCP mapping table):

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
R6:
policy-map SET_COS
 class class-default
  set cos 4
!
interface FastEthernet 0/0
 service-policy output SET_COS

SW1, SW3, & SW4:
no mls qos

R1:
ip access-list extended TEST
 permit icmp any any dscp cs4
 permit ip any any
!
interface FastEthernet 0/0
 ip access-group TEST in
```

```
Rack1R6#ping 155.1.146.1 repeat 100 size 36

Type escape sequence to abort.
Sending 100, 36-byte ICMP Echos to 155.1.146.1, timeout is 2 seconds:
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Success rate is 100 percent (100/100), round-trip min/avg/max = 1/2/4
ms

Rack1R1#sh ip access-lists TEST
Extended IP access list TEST
    10 permit icmp any any dscp cs4 (100 matches)
    20 permit ip any any (927 matches)
```

This test demonstrates that the Catalyst switch actually classifies the IP packets based on the CoS value in Ethernet header. Re-enable MLS QoS on SW1 and see how we can verify QoS marking on SW4, which is a 3550.

The 3550 switch supports a special feature known as the DSCP monitor. The DSCP values monitored are the internal DSCP values (QoS labels) assigned by the switch for incoming packets and seen by the switch in the outgoing packets. You can monitor either packet count or bytes count. You set the monitor mode on special “master” interfaces (e.g. interface Fa 0/1 controls mode for interfaces 1-12 inclusive and interface Fa 0/13 controls mode for interfaces 13-24). You may specify up to 8 DSCP values to monitor in both directions.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
SW4:
interface FastEthernet 0/1
 mls qos monitor packets
!
interface FastEthernet 0/4
 mls qos monitor dscp 0 16 26 46 48
```

Rack1SW4#clear mls qos interface fastEthernet 0/4 statistics

Let's see some preliminary statistics. So far, you can see Ingress packets marked with DSCP 16 and DSCP 48. They correspond to the CDP/L2 Keepalives (CoS 2 maps to DSCP 16) and RIP routing updates (IP Precedence of 6 maps to DSCP of 48). Note that all packets are listed as "no_change" since they have been classified on trusted values. Also, note that "Egress" direction does not show any counters for classified packets, since classification is only possible ingress.

```
Rack1SW4#show mls qos interface fastEthernet 0/4 statistics
FastEthernet0/4
Ingress
  dscp: incoming   no_change  classified policed    dropped (in pkts)
    0 : 0          0          0          0          0
    16: 7          7          0          0          0
    26: 0          0          0          0          0
    46: 0          0          0          0          0
    48: 7          7          0          0          0
Others: 0          0          0          0          0
Egress
  dscp: incoming   no_change  classified policed    dropped (in pkts)
    0 : 14            n/a        n/a      0          0
    16: 0             n/a        n/a      0          0
    26: 0             n/a        n/a      0          0
    46: 0             n/a        n/a      0          0
    48: 0             n/a        n/a      0          0
Others: 37            n/a        n/a      0          0
```

Now generate some packets with IP precedence of 3 (ToS byte 96) and see how the switch re-marks them:

```
Rack1R4#ping
Protocol [ip]:
Target IP address: 155.1.146.1
Repeat count [5]: 100
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface:
Type of service [0]: 96
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
```

Copyright © 2009 Internetwork Expert

```
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 100, 100-byte ICMP Echos to 155.1.146.1, timeout is 2 seconds:
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Success rate is 100 percent (100/100), round-trip min/avg/max = 1/3/5
ms

Rack1SW4#show mls qos interface fastEthernet 0/4 statistics
FastEthernet0/4
Ingress
  dscp: incoming   no_change  classified policed    dropped (in pkts)
    0 : 0          0          0          0          0
    16: 60         60         0          0          0
    26: 0          0          100        0          0
    46: 0          0          0          0          0
    48: 55         55         0          0          0
Others: 100        0          0          0          0
Egress
  dscp: incoming   no_change  classified policed    dropped (in pkts)
    0 : 211           n/a        n/a     0          0
    16: 0             n/a        n/a     0          0
    26: 0             n/a        n/a     0          0
    46: 0             n/a        n/a     0          0
    48: 0             n/a        n/a     0          0
Others: 315           n/a        n/a     0          0
```

This time 100 packets have been classified to the DSCP value of 26. This is because the switch used its mapping table to deduce the internal DSCP value. Of course, the incoming packets had a DSCP value of 96 and you can see them under the “Others” row of the table.

Now, verify the CoS override feature on the port connected to R5. Configure SW2 to detect packets marked with DSCP value of 44 and disable MLS QoS on all switches with except to SW1:

```
SW3, SW4:
no mls qos

SW2:
no mls qos
!
ip access-list extended TEST
 permit ip any any dscp 44
 permit ip any any
!
interface Vlan 58
 ip access-group TEST in
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
Rack1R5#ping 155.1.58.8 repeat 100

Type escape sequence to abort.
Sending 100, 100-byte ICMP Echos to 155.1.58.8, timeout is 2 seconds:
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Success rate is 100 percent (100/100), round-trip min/avg/max = 1/2/4
ms
Rack1R5#

Rack1SW2#show ip access-list TEST
Extended IP access list TEST
    10 permit ip any any dscp 44 (101 matches)
    20 permit ip any any (735 matches)
```

The verification of DSCP trust is the simplest of them. Source packets off R1 marked with DSCP value of EF (ToS bytes 184, DSCP numerical value of 46).

```
Rack1R1#ping
Protocol [ip]:
Target IP address: 155.1.146.4
Repeat count [5]: 100
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface:
Type of service [0]: 184
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 100, 100-byte ICMP Echos to 155.1.146.4, timeout is 2 seconds:
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Success rate is 100 percent (100/100), round-trip min/avg/max = 1/3/12
ms
```

Next, verify packets matching on SW1 side. Note the 100 packets matched for DSCP value of 46. Note that CDP packets and keepalives increment counters for the CoS value of 1.

```
Rack1SW1#show mls qos interface fastEthernet 0/1 statistics
FastEthernet0/1

  dscp: incoming
-------------------------------

  0 -  4 :           0            0            0            0            0
  5 -  9 :           0            0            0            0            0
 10 - 14 :           0            0            0            0            0
 15 - 19 :           0            0            0            0            0
 20 - 24 :           0            0            0            0            0
 25 - 29 :           0            0            0            0            0
 30 - 34 :           0            0            0            0            0
 35 - 39 :           0            0            0            0            0
 40 - 44 :           0            0            0            0            0
 45 - 49 :           0          100            0           13            0
 50 - 54 :           0            0            0            0            0
 55 - 59 :           0            0            0            0            0
 60 - 64 :           0            0            0            0
  dscp: outgoing
-------------------------------

  0 -  4 :         126            0            0            0            0
  5 -  9 :           0            0            0            0            0
 10 - 14 :           0            0            0            0            0
 15 - 19 :           0            0            0            0            0
 20 - 24 :           0            0            0            0            0
 25 - 29 :           0            0            0            0            0
 30 - 34 :           0            0            0            0            0
 35 - 39 :           0            0            0            0            0
 40 - 44 :           0            0            0            0            0
 45 - 49 :           0            0            0            0            0
 50 - 54 :           0            0            0            0            0
 55 - 59 :           0            0            0            0            0
 60 - 64 :           0            0            0            0
  cos: incoming
-------------------------------

  0 -  4 :           0          127            0            0            0
  5 -  7 :           0            0            0
  cos: outgoing
-------------------------------

  0 -  4 :         126            0            0            0            0
  5 -  7 :           0            0            0
Policer: Inprofile:            0 OutofProfile:            0
```

Now confirm that the switch actually sends DSCP 46 packets out to R4. To do that, you need to find the exit trunk for the MAC address of R4. Find out the MAC address of R4 and look up the exit port. After that, check the MLS QoS statistics for the output port.

```
Rack1R4#show interfaces fastEthernet 0/1
FastEthernet0/1 is up, line protocol is up
  Hardware is AmdFE, address is 0012.43c1.6f01 (bia 0012.43c1.6f01)
<snip>

Rack1SW1#show mac-address-table address 0012.43c1.6f01
          Mac Address Table
-------------------------------------------

Vlan    Mac Address       Type        Ports
----    -----------       --------    -----
 146    0012.43c1.6f01    DYNAMIC     Fa0/16
Total Mac Addresses for this criterion: 1

Rack1SW1#show mls qos interface fastEthernet 0/16 statistics
FastEthernet0/16

  dscp: incoming
-------------------------------

  0 -  4 :        8660            0            0            0            0
  5 -  9 :           0            0            0            0            0
 10 - 14 :           0            0            0            0            0
 15 - 19 :           0            0            0            0            0
 20 - 24 :           0            0            0            0            0
 25 - 29 :           0          100            0            0            0
 30 - 34 :           0            0         1159            0            0
 35 - 39 :           0            0            0            0            0
 40 - 44 :           0            0            0            0          100
 45 - 49 :           0          100            0        59898            0
 50 - 54 :           0            0            0            0            0
 55 - 59 :           0         9491            0            0            0
 60 - 64 :           0            0            0            0
  dscp: outgoing
-------------------------------

  0 -  4 :        1629            0            0            0            0
  5 -  9 :           0            0            0            0            0
 10 - 14 :           0            0            0            0            0
 15 - 19 :           0            0            0            0            0
 20 - 24 :           0            0            0            0            0
 25 - 29 :           0          100            0            0            0
 30 - 34 :           0            0          700            0            0
 35 - 39 :           0            0            0            0            0
 40 - 44 :           0            0            0            0          213
 45 - 49 :           0          100            0        24082            0
 50 - 54 :           0            0            0            0            0
 55 - 59 :           0            0            0            0            0
 60 - 64 :           0            0            0            0
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
  cos: incoming
-------------------------------

  0 -  4 :       69688            0           342           100          2160
  5 -  7 :         100        28168        208834
  cos: outgoing
-------------------------------

  0 -  4 :       10796           11             0             0           709
  5 -  7 :         100        15335             4
Policer: Inprofile:            0 OutofProfile:            0
```

Note the number of DSCP EF packets as well as the count of CoS 5 packets that were switched out of the port. Note that may need to clear statistics on this link as well before sending the sample packets sif you want to get more accurate statistics.

10.63 Catalyst QoS Marking Pass-Through

- Ensure SW2 classifies packets coming from R6's VLANs 67 and VLAN 146 based on CoS values, but does not change the DSCP value in IP packets.
- Additionally ensure that IP-precedence based classification on the SW4's connection to R4 does not change the CoS value in Ethernet headers.

Configuration

```
SW2:
no mls qos rewrite ip dscp

SW4:
interface FastEthernet 0/4
 mls qos trust dscp pass-through cos
```

Verification

Note

The automatic rewrite feature ensures the uniform marking - that is it takes care of synchronizing L2 and L3 code points. Is it possible to trust DSCP and built the internal QoS label based on its value, but retain the CoS bits in the packet? Alternatively – trust CoS bits and retain the DSCP values? You may need this capability occasionally, if you want to "tunnel" one type of QoS marking through your network, while using the other type for your needs. One possible example could be a Metro Ethernet provider that uses CoS bits for QoS marking in core, but preserves customer DSCP marking.

In the Catalyst 3550, you may set one type of marking as "pass-through". For example, when you trust CoS you may enable DSCP pass-through with the command **`mls qos trust cos pass-through dscp`**. The command with reversed logic is obviously **`mls qos trust dscp pass-through cos`**.

On the 3560 switches, you may only disable DSCP rewrite in IP headers, when you trust the CoS values, using the global command no **`mls qos rewrite ip dscp`**.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

To verify QoS "tunneling" on the 3560, perform the following steps. First, temporarily disable MLS QoS in all switches except for SW2 (which connects to R6) to ensure that other switches do not change our QoS marking. Next, configure R6 to mark all outgoing packets with a CoS value of four. After that, configure R1 to "detect" ICMP packets with a DSCP value of 24 using an extended ACL. Lastly, source ICMP packets with a ToS byte of 96 (DSCP 24) out of R6 to R1 and check the QoS statistics.

```
R6:
policy-map SET_COS
 class class-default
  set cos 4
!
interface FastEthernet 0/0
 service-policy output SET_COS

SW1,SW3,SW4:
no mls qos

R1:
no ip access-list extended TEST
ip access-list extended TEST
 permit icmp any any dscp cs3
 permit ip any any
!
interface FastEthernet 0/0
 ip access-group TEST in

Rack1SW2#clear mls qos interface fastEthernet 0/6 statistics

Rack1R6#ping
Protocol [ip]:
Target IP address: 155.1.146.1
Repeat count [5]: 100
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface:
Type of service [0]: 96
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 100, 100-byte ICMP Echos to 155.1.146.1, timeout is 2 seconds:
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Success rate is 100 percent (100/100), round-trip min/avg/max = 1/2/4
ms
Rack1R6#
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
Rack1R1#show ip access-list TEST
Extended IP access list TEST
    10 permit icmp any any dscp cs3 (100 matches)
    20 permit ip any any (114 matches)

Rack1SW2#show mls qos interface fastEthernet 0/6 statistics
FastEthernet0/6

  dscp: incoming
-------------------------------

  0 -  4 :           0            0            0            0            0
  5 -  9 :           0            0            0            0            0
 10 - 14 :           0            0            0            0            0
 15 - 19 :           0            0            0            0            0
 20 - 24 :           0            0            0            0          100
 25 - 29 :           0            0            0            0            0
 30 - 34 :           0            0            0            0            0
 35 - 39 :           0            0            0            0            0
 40 - 44 :           0            0            0            0            0
 45 - 49 :           0            0            0           16            0
 50 - 54 :           0            0            0            0            0
 55 - 59 :           0            0            0            0            0
 60 - 64 :           0            0            0            0
  dscp: outgoing
-------------------------------

  0 -  4 :           0            0            0            0            0
  5 -  9 :           0            0            0            0            0
 10 - 14 :           0            0            0            0            0
 15 - 19 :           0            0            0            0            0
 20 - 24 :           0            0            0            0          100
 25 - 29 :           0            0            0            0            0
 30 - 34 :           0            0            0            0            0
 35 - 39 :           0            0            0            0            0
 40 - 44 :           0            0            0            0            0
 45 - 49 :           0            0            0           26            0
 50 - 54 :           0            0            0            0            0
 55 - 59 :           0            0            0            0            0
 60 - 64 :           0            0            0            0
  cos: incoming
-------------------------------

  0 -  4 :           0           10            0            0          116
  5 -  7 :           0            0            0
  cos: outgoing
-------------------------------

  0 -  4 :         126            0            0            0            0
  5 -  7 :           0            0            0
Policer: Inprofile:            0 OutofProfile:            0
```

From the output above you can see that SW2 accepts packets with DSCP 24 and CoS 4 from R6. In accordance with CoS to DSCP mapping table, DSCP should have changed to 32. However, as we see R1 still receives packets marked with DSCP 24 (CS3).

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

To verify CoS pass-through, we temporarily need to convert the port connected to R4 VLAN146 interface into a trunk and configure a policy-map that sets some CoS value, e.g. 3, in outgoing packets:

```
R4:
policy-map SET_COS
 class class-default
  set cos 3
!
interface FastEthernet 0/1
 no ip address
 service-policy output SET_COS
!
interface FastEthernet 0/1.146
 encapsulation dot1q 146
 ip address 155.1.146.4 255.255.255.0

SW4:
interface FastEthernet0/4
 switchport trunk encapsulation dot1q
 switchport mode trunk
 mls qos trust dscp pass-through cos
```

Configure R6 to “detect” frames marked with CoS 3 and IP packets marked with DSCP EF. After that, source IP packets marked with DSCP EF from R4 to R6. Be sure to disable MLS QoS temporarily on SW1, SW2, and SW3:

```
SW1, SW2, SW3:
no mls qos

R6:
ip access-list extended TEST
 permit icmp any any dscp ef
!
interface FastEthernet 0/0.146
 ip access-group TEST in

class-map COS_3
 match cos 3
!
policy-map MATCH_COS
 class COS_3
!
interface FastEthernet 0/0
 service-policy input MATCH_COS
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
Rack1R4#ping
Protocol [ip]:
Target IP address: 155.1.146.6
Repeat count [5]: 100
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface:
Type of service [0]: 184
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 100, 100-byte ICMP Echos to 155.1.146.6, timeout is 2 seconds:
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Success rate is 100 percent (100/100), round-trip min/avg/max = 1/3/4
ms
Rack1R4#
```

Observe the resulting statistics on R6. Note that incoming packets have DSCP value of EF and CoS marking of 3. That means SW4 did not change the CoS field in accordance with DSCP.

```
Rack1R6#show ip access-lists TEST
Extended IP access list TEST
    10 permit icmp any any dscp ef (100 matches)
    20 permit ip any any (339 matches)

Rack1R6#show policy-map int fastEthernet 0/0
 FastEthernet0/0

  Service-policy input: MATCH_COS

    Class-map: COS_3 (match-all)
      104 packets, 13120 bytes
      5 minute offered rate 2000 bps
      Match: cos  3

    Class-map: class-default (match-any)
      8 packets, 3280 bytes
      5 minute offered rate 1000 bps, drop rate 0 bps
      Match: any
```

10.64 Catalyst QoS ACL Based Classification & Marking

- Create a policy-map on SW2 that performs the following on traffic received from R6.
    - Set a DSCP value of EF on IPX packets (EtherType value 0x8137)
    - Set a DSCP value of CS3 on ICMP packets
    - Set an IP Precedence of 2 on TELNET packets
    - Trust CoS under the default class
- Configure SW4 and R4 to change the port connected to R4 into a trunk.
- Configure SW4 to trust IP DSCP values in packets coming from R4, and set the CoS to a value of 2.

Configuration

```
SW2:
mls qos rewrite ip dscp
!
mac access-list extended IPX
 permit any any 0x8137 0x0
!
ip access-list extended ICMP
 permit icmp any any
!
ip access-list extended TELNET
 permit tcp any any eq telnet
 permit tcp any eq telnet any
!
class-map match-all TELNET
  match access-group name TELNET
!
class-map match-all ICMP
  match access-group name ICMP
!
class-map match-all IPX
  match access-group name IPX
!
! Note that we set DSCP under non-IP traffic. The switch computes
! the respective CoS value using DSCP-to-CoS table.
!
policy-map CLASSIFY
  class IPX
   set dscp ef
  class ICMP
   set dscp cs3
  class TELNET
   set precedence 2
  class class-default
    trust cos
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
!
interface FastEthernet 0/6
 mls qos cos 1
 service-policy input CLASSIFY

SW4:
mls qos cos policy-map
!
ip access-list standard IP_ANY
 permit any
!
class-map IP_ANY
 match access-group name IP_ANY
!
! Note that we can't set in class-default with the 3550
! you need a special class that matches all IP traffic
! in order to implement the policy
!
policy-map CLASSIFY
 class IP_ANY
  trust dscp
  set cos 2
!
interface FastEthernet 0/4
 switchport trunk encapsulation dot1q
 switchport mode trunk
 service-policy input CLASSIFY

R4:
interface FastEthernet 0/1
 no ip address
!
interface FastEthernet 0/1.146
 encapsulation dot1q 146
 ip address 155.1.146.4 255.255.255.0
```

Verification

Note

The most flexible classification option for the Catalyst is using QoS ACLs. Even though they are called QoS ACLs (the term has been borrowed from CatOS) you actually apply them using MQC syntax using class-maps and policy-map. You create MQC classes by matching one of the following:

1) MAC or IP/IPv6 access-list. The MAC access-list matches non-IP packets and IP/IPv6 match IP or IPv6 packets respectively. You can only match IPv6 packets on the 3560. You cannot use MAC ACLs to match IP traffic (though you can use MAC ACLs on the 3550 to match IPv6 traffic).

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

2) DSCP and IP Precedence values. Note that you cannot match CoS values in Ethernet headers.

3) Additional platform-specific matches like **`match input-interface`** and **`match vlan`**. We will consider them in separate tasks.

Note that the classification criteria is very limited compared to IOS routers. You cannot match size or packet contents. Additionally, you cannot do hierarchical matching, with the exception of the per-VLAN classification feature discussed in a separate task. The limitations are the result of hardware optimization in the Layer 3 switches.

Pay attention to the behavior of the class "class-default" with the Catalyst QoS. This class works fine in the 3560 switches, matching both IP and non-IP traffic. However, it seems to work inconsistently or does not work at all with the 3550 switches. In the latter model, as a workaround, create a special class to match either all IP or all non-IP traffic using ACLs.

Under a class in the policy-map you can either trust (CoS, DSCP, IP Precedence) or set (DSCP, IP Precedence) the respective QoS marking. Note that you cannot set CoS value directly in the 3560 switches, but you can set DSCP for non-IP packets. The switch translates DSCP into CoS using the DSCP-to-CoS mapping table. The same holds true for the 3550 switches – you can set DSCP on the non-IP packets and it is translated to the CoS.

You cat set CoS directly in the 3550 switches in one special case. First, you need the global command **`mls qos cos policy-map`**. Using this feature, you *must* set the DSCP marking *and* set Layer 2 marking using the **`set cos`** command. This simulates the behavior of "CoS pass-through" feature available with interface-level settings. The set cos feature only works when you trust DSCP. Furthermore, the 3550 performs all QoS processing and deduces internal DSCP based on the *trusted* DSCP value, not the CoS value set.

On both switch models, you can either configure **`set dscp`** or **`set ip precedence`** but not both – the one configured later erases the previous one. Also, if a class contains "trust" and "set" statements for the same type of marking (e.g. L3 or L2) the trust statement takes precedence over explicit set.

Aside from that, the trust feature works the same way as it works on the interface, just it has the scope limited to a class defined by ACL.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Remember that applying a service-policy will remove any interface-level QoS settings on the interface, with the exception of the default CoS (which is used by the policy map when you trust CoS inside a class). If the input packet does not match any class in the service-policy, the switch will set all markings to zero. If you trust CoS for IP or non-IP packets and there is no 802.1q/ISL header, the switch will take the default CoS value from the interface settings or use the zero markings.

The verification procedure is somewhat complicated and we will try to do it in systematic manner. First, we configure IPX on both R4 and R6, using IPX SNAP encapsulation. This way we can generate non-IP traffic, understood both by the 3550 and 3560 switches.

```
R4:
ipx routing
!
interface FastEthernet 0/1.146
 ipx network 146 encapsulation snap

R6:
ipx routing
!
interface FastEthernet 0/0.146
 ipx network 146 encapsulation snap
```

The first thing we want to test is to see how the 3560 assign QoS marking. We expect the IPX packets to bear CoS value of 5 as they leave the switch. In addition, we expect the switch to enforce marking for telnet and ICMP packets.

First, prepare SW4 to count CoS 5 packets coming to R4. You will need to trust CoS values on all trunk links and disable QoS on SW3 and SW1 (so that they don’t interfere):

```
SW1, SW3:
no mls qos

SW4:
mls qos
!
interface FastEthernet 0/1
 mls qos monitor packets
!
interface FastEthernet 0/4
 mls qos monitor dscp 0 40 46
!
interface range FastEthernet 0/13 – 21
 mls qos trust cos
```

Now send IPX ping packets from R6 to R4. To do that, you need to learn the IPX address of R4 first. After that, check the count of DSCP 40 packets sent to R4. Remember that DSCP values are not the actual, but internal switch DSCP (QoS labels). Thus, we ensure that set DSCP operation works for non-IP packets by the virtue of the mapping tables.

```
Rack1R4#show ipx interface fastEthernet 0/1.146
FastEthernet0/1.146 is up, line protocol is up
  IPX address is 146.000e.8475.1341, 802.1Q vLAN (SNAP) [up]
  Delay of this IPX network, in ticks is 1

Rack1SW4#clear mls qos interface fastEthernet 0/4 statistics

Rack1R6#ping 146.000e.8475.1341

Translating "146.000e.8475.1341"

Type escape sequence to abort.
Sending 5, 100-byte IPX Novell Echoes to 146.000e.8475.1341, timeout is
2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
Rack1R6#

Rack1SW4#show mls qos interface fastEthernet 0/4 statistics
FastEthernet0/4
Ingress
  dscp: incoming   no_change  classified policed    dropped (in pkts)
    0 : 3          3          7          0          0
    40: 0          0          0          0          0
    46: 0          0          0          0          0
Others: 7          0          0          0          0
Egress
  dscp: incoming   no_change  classified policed    dropped (in pkts)
    0 : 15            n/a       n/a      0          0
    40: 5             n/a       n/a      0          0
    46: 0             n/a       n/a      0          0
Others: 137           n/a       n/a      0          0
```

Now we need to ensure marking for telnet and ICMP packets going to R1.

```
R1:
ip access-list extended TEST
 permit tcp any any eq telnet precedence immediate
 permit icmp any any dscp cs3
 permit ip any any
!
interface FastEthernet 0/0
 ip access-group TEST in
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
Rack1R6#telnet 155.1.146.1
Trying 155.1.146.1 ... Open


User Access Verification

Password: cisco
Rack1R1>en
Password: cisco

Rack1R1#show ip access-list TEST
Extended IP access list TEST
    10 permit tcp any any eq telnet precedence immediate (108 matches)
    20 permit icmp any any dscp cs3
    30 permit ip any any (18 matches)

Rack1R6#ping 155.1.146.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 155.1.146.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
Rack1R6#

Rack1R1#show ip access-lists TEST
Extended IP access list TEST
    10 permit tcp any any eq telnet precedence immediate (207 matches)
    20 permit icmp any any dscp cs3 (5 matches)
    30 permit ip any any (216 matches)
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Ensure that packets not matching any of the user-defined classes match class-default and have their CoS value trusted or assigned from the interface default setting. The output shows CoS value of 1, assigned to non-tagged packets such as CDP or L2 keepalives received from R6.

```
Rack1SW2#show mls qos interface fastEthernet 0/6 statistics
FastEthernet0/6

  dscp: incoming
-------------------------------

  0 -  4 :         110            0            0            0            0
  5 -  9 :           0            0            0            0            0
 10 - 14 :           0            0            0            0            0
 15 - 19 :           0            0            0            0            0
 20 - 24 :           0            0            0            0          100
 25 - 29 :           0            0            0            0            0
 30 - 34 :           0            0            0            0            0
 35 - 39 :           0            0            0            0            0
 40 - 44 :           0            0            0            0            0
 45 - 49 :           0          200            0         4761            0
 50 - 54 :           0            0            0            0            0
 55 - 59 :           0            0            0            0            0
 60 - 64 :           0            0            0            0
  dscp: outgoing
-------------------------------

  0 -  4 :        2094            0            0            0            0
  5 -  9 :           0            0            0            0            0
 10 - 14 :           0            0            0            0            0
 15 - 19 :           0            0            0            0            0
 20 - 24 :           0            0            0            0          100
 25 - 29 :           0            0            0            0            0
 30 - 34 :           0            0            0            0            0
 35 - 39 :           0            0            0            0            0
 40 - 44 :           0            0            0            0            0
 45 - 49 :           0          200            0         4988            0
 50 - 54 :           0            0            0            0            0
 55 - 59 :           0            0            0            0            0
 60 - 64 :           0            0            0            0
  cos: incoming
-------------------------------

  0 -  4 :        1205         1401            0            0         5184
  5 -  7 :           0            0            0
  cos: outgoing
-------------------------------

  0 -  4 :        5089            0            0         1313            0
  5 -  7 :           0         1064            0
Policer: Inprofile:            0 OutofProfile:           0
```

Now configure R6 to mark all outgoing *tagged* packets (the router may only set CoS in trunk tagged packets) with a CoS value of 4. After that, send HTTP traffic to R1 from R6 and ensure R1 sees it marked with DSCP value of CS4, which corresponds to automatic rewrite from CoS value of 4. This procedure verifies that the 3560 actually trusts CoS values in class-default packets even for IP packets.

```
R1:
no ip access-list extended TEST
ip access-list extended TEST
 permit tcp any any eq 80 dscp cs4
 permit ip any any
!
interface FastEthernet 0/0
 ip access-group TEST in
!
ip http server

R6:
policy-map SET_COS
 class class-default
   set cos 4
!
interface FastEthernet 0/0
 service-policy output SET_COS
```

```
Rack1R6#telnet 155.1.146.1 80
Trying 155.1.146.1, 80 ... Open
GET /

WWW-Authenticate: Basic realm="level_15_access"

401 Unauthorized

[Connection to 155.1.146.1 closed by foreign host]
Rack1R6#

Rack1R1#show ip access-list TEST
Extended IP access list TEST
    10 permit tcp any any eq www dscp cs4 (5 matches)
    20 permit ip any any (10 matches)
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

The last thing to verify is the simultaneous DSCP trusting and CoS setting in the 3550. We need to make sure that an IP packet sent from R4 reaches R6 with non-synchronized Layer 3 and Layer 2 markings. To verify that, shut down the VLAN 67 interface of R6 and VLAN 146 interface of R1. This is to ensure that R6 only receives routing updates from R4.

Now R4 generate routing updates with an IP precedence of six. The switch trusts this value but sets CoS to “2”. Therefore, we should expect packets with DSCP value of CS6 and CoS 2 on R6.

Configure R6 to “detect” both types of marking, and disable MLS QoS in SW1, SW2 and SW3:

```
SW1, SW2, SW3:
no mls qos

R6:
class-map COS_2
 match cos 2
class-map COS_6
 match cos 6
!
policy-map MATCH_COS
 class COS_2
 class COS_6
!
ip access-list extended TEST
 permit udp any any eq rip dscp 48
 permit ip any any
!
interface FastEthernet 0/0
 service-policy input MATCH_COS
 ip access-group TEST in

Rack1R6#clear access-list counters

Rack1R6#clear counters fastEthernet 0/0
Clear "show interface" counters on this interface [confirm]

Rack1R6#show ip access-list TEST
Extended IP access list TEST
    10 permit udp any any eq rip dscp cs6 (3 matches)
    20 permit ip any any
```

```
Rack1R6#show policy-map int fa 0/0
 FastEthernet0/0

  Service-policy input: MATCH_COS

    Class-map: COS_2 (match-all)
      3 packets, 1650 bytes
      5 minute offered rate 0 bps
      Match: cos  2

    Class-map: COS_6 (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps
      Match: cos  6

    Class-map: class-default (match-any)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any

  Service-policy output: SET_COS

    Class-map: class-default (match-any)
      6 packets, 450 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any
      QoS Set
        cos 4
          Packets marked 3
```

This example demonstrates that it is possible in the 3550 to set CoS and Trust DSCP independently for the same packet.

10.65 Catalyst 3550 Per-Port Per-VLAN Classification

- Configure SW4 so that ICMP traffic coming in on the port connected to R4 has the DSCP value set to CS3.
- For IPX traffic with EtherType 0x8137 set the CoS field to 2.
- Ensure your classification only affects traffic on VLAN 146 but not any VLANs trunked to R4 in the future.

Configuration

```
SW4:
mls qos
!
ip access-list extended ICMP
 permit icmp any any
!
mac access-list extended IPX
 permit any any 0x8137 0x0
!
! Second-level class-maps
!
class-map ICMP
 match access-group name ICMP
!
class-map IPX
 match access-group name IPX
!
! First-level class-maps
!
class-map VLAN_146_ICMP
 match vlan 146
 match class-map ICMP
!
class-map VLAN_146_IPX
 match vlan 146
 match class-map IPX
!
! Note that we set DSCP for IPX packets
! In fact, this is the internal DSCP value, not the IP DSCP.
! You cannot set CoS directly - only when trusting DSCP.
! DSCP 16 translates to CoS 2 by the virtue of the default
! DSCP-to-CoS table.
!
policy-map PER_PORT_PER_VLAN
 class VLAN_146_ICMP
   set dscp CS3
 class VLAN_146_IPX
   set dscp 16
!
interface FastEthernet 0/4
  no service-policy input CLASSIFY
  service-policy input PER_PORT_PER_VLAN
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Verification

Note

Per-Port Per-VLAN classification adds granularity to the classification process based on ACLs. You may create a policy-map that matches specific traffic classes insides specific VLANs. The only limitation is that the policy-map is applied at the port level, not globally to the VLAN.

Usually you apply a per-port per-VLAN (PPPV) policy-map to a trunk port. However, it is possible to apply the PPPV classification to an access-port, if the access-vlan matches any of the VLANs configured in the policy-map.

Remember to observe the following strict syntax rules when configuring PPPV classification in the 3550 model:

1) You must use two-levels of class-maps. The first level class-map must have the following format:

```
class-map L1_CLASS
  match vlan <VLAN ID>
  match class-map L2_CLASS
```

The **match vlan** statement must always be the first statement in the class-map, and the only allowed second statement is match class-map. In addition, both first level and second-level class-maps must be of type **match-any**.

2) All first-level class-maps must have the **match vlan** statement, and you cannot mix them with regular class-maps.

3) The second level class map is only allowed a *single* match statement, and it cannot be **match class-map**. For example,

```
class-map L2_CLASS
 match access-group name <NAME>
```

4) The policy-map lists the above-defined first-level classes with respective actions (e.g. set/trust statements):

```
policy-map PPPV
 class L1_CLASS
  set dscp ef
```

Copyright © 2009 Internetwork Expert

To start verification, enable IPX on R4 and R6. We will need that to generate non-IP packets. After that, configure SW4 to monitor DSCP values of interest in packets entering the switchports.

```
R4:
ipx routing
!
interface FastEthernet 0/1.146
 ipx network 146 encapsulation snap

R6:
ipx routing
!
interface FastEthernet 0/0.146
 ipx network 146 encapsulation snap

SW4:
interface FastEthernet 0/1
 mls qos monitor packets
!
interface FastEthernet 0/4
 mls qos monitor dscp 0 16 24 48
```

Now send ICMP packets to R1 and IPX packets to R6. Before you start, learn the IPX addres of R6.

```
Rack1R6#show ipx interface fastEthernet 0/0.146
FastEthernet0/0.146 is up, line protocol is up
  IPX address is 146.0011.9221.da80, 802.1Q vLAN (SNAP) [up]
<snip>

Rack1SW4#clear mls qos interface fastEthernet 0/4 statistics
Rack1SW4#

Rack1R4#ping 155.1.146.1 repeat 50

Type escape sequence to abort.
Sending 50, 100-byte ICMP Echos to 155.1.146.1, timeout is 2 seconds:
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Success rate is 100 percent (50/50), round-trip min/avg/max = 1/2/4 ms

Rack1R4#ping 146.0011.9221.da80

Translating "146.0011.9221.da80"

Type escape sequence to abort.
Sending 5, 100-byte IPX Novell Echoes to 146.0011.9221.da80, timeout is
2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

Now verify the statistics on SW4 port connected to R4:

```
Rack1SW4#show mls qos interface fastEthernet 0/4 statistics
FastEthernet0/4
Ingress
  dscp: incoming   no_change  classified policed    dropped (in pkts)
    0 : 72         17         16         0          0
    16: 0          0          5          0          0
    24: 0          0          50         0          0
    48: 16         0          0          0          0
Others: 0          0          0          0          0
Egress
  dscp: incoming   no_change  classified policed    dropped (in pkts)
    0 : 510           n/a       n/a      0          0
    16: 0             n/a       n/a      0          0
    24: 0             n/a       n/a      0          0
    48: 32            n/a       n/a      0          0
Others: 971           n/a       n/a      0          0
```

Note the packet counters in “classified” column. The column list packets that have their DSCP value changed from their original value. Remember that from the internal DSCP standpoint both IP and non-IP packets have a DSCP value associated with them.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.66 Catalyst 3560 Per-VLAN Classification

- Enable VLAN-based QoS on the trunk ports of SW1 and on the port connected to R1.
- Configure SW1 to mark all TCP traffic on VLAN 146 with a DSCP value of EF.
- IPX SNAP-encapsulated traffic should be marked with a CoS value of 4.

Configuration

```
SW1:
mls qos
!
interface FastEthernet 0/1
 mls qos vlan-based
!
! Enable VLAN-based QoS on the trunks
!
interface range FastEthernet 0/13 - 21
 mls qos vlan-based
!
ip access-list extended TCP
 permit tcp any any
!
mac access-list extended IPX
 permit any any 0x8137 0x0
!
! First-level class-maps
!
class-map TCP
 match access-group name TCP
!
class-map IPX
 match access-group name IPX
!
! VLAN-based policy-map
!
policy-map PER_VLAN
 class TCP
  set dscp ef
 class IPX
  set dscp 32
!
! Apply the policy-map to SVI
!
interface VLAN 146
 service-policy input PER_VLAN
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Verification

> **Note**
>
> The 3560 model configuration differs significantly from the 3550 with respect to per-VLAN classification. You cannot use the **`match vlan`** statement anymore. Rather, you may enable certain ports for per-VLAN QoS services using the command **`mls qos vlan-based`**. The mentioned ports will inherit all QoS configurations from the respective SVIs. This approach allows for true switch-wide VLAN QoS. With the VLAN-based QoS model you apply the service policy to an SVI, and all ports enabled for VLAN QoS that have this VLAN active (trunks or access ports) will use the service policy on the respective VLAN.
>
> The major difference from the 3550 model is that you apply classification to all ports enabled for VLAN QoS. You may selectively disable VLAN-wide classification on specific ports by using **`no mls qos vlan-based`** command.
>
> Verifications of this configuration is tricky due to the limited set of show commands. We need to verify the following:
>
> 1) Traffic coming from R1 (IP and non-IP) receives proper marking in SW1
> 2) VLAN146 traffic coming across SW1 on the trunk ports also receives proper marking
>
> Let's start with R1's connection. Enable IPX on R1, R4, and R6 to generate non-IP traffic. At the same time, Configure SW4 to monitor DSCP values 0, 32, 46, and 48 on the connection to R4's port. Note that DSCP values of 32 and 46 correspond to IPX and TCP traffic. Lastly, configure SW4 to trust DSCP markings on all trunk connections - this will ensure it does not erase the marking in IP/non-IP packets. (Remember when you trust DSCP, the switch classifies non-IP packets trusting their CoS).

```
R1:
ipx routing
!
interface FastEthernet 0/0
 ipx network 146 encapsulation snap

R4:
ipx routing
!
interface FastEthernet 0/1.146
 ipx network 146 encapsulation snap
```

```
R6:
ipx routing
!
interface FastEthernet 0/0.146
 ipx network 146 encapsulation snap

SW4:
interface FastEthernet 0/1
 mls qos monitor packets
!
interface FastEthernet 0/4
 mls qos monitor dscp 0 32 46 48
!
interface range FastEthernet 0/13 - 21
 mls qos trust dscp
```

At this point, clear the MLS QoS statistics on the connection to R4. After that, find the IPX address of R4 in VLAN146 and then generate IPX and TCP traffic from R1 to R4. Also, make sure you disabled MLS QoS in all non-participating switches, i.e. SW2 and SW3:

```
SW2, SW3:
no mls qos

Rack1SW4#clear mls qos interface fastEthernet 0/4 statistics

Rack1R4#show ipx interface fastEthernet 0/1.146
FastEthernet0/1.146 is up, line protocol is up
  IPX address is 146.000f.8f0f.6101, 802.1Q vLAN (SNAP) [up]
<snip>

Rack1R1#ping 146.000f.8f0f.6101

Translating "146.000f.8f0f.6101"

Type escape sequence to abort.
Sending 5, 100-byte IPX Novell Echoes to 146.000f.8f0f.6101, timeout is
2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms

Rack1R1#telnet 155.1.146.4
Trying 155.1.146.4 ... Open


User Access Verification

Password: cisco
Rack1R4>exit

[Connection to 155.1.146.4 closed by foreign host]
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
Rack1SW4#show mls qos interface fastEthernet 0/4 statistics
FastEthernet0/4
Ingress
  dscp: incoming   no_change  classified policed    dropped (in pkts)
    0 : 26         21         4          0          0
    32: 0          0          0          0          0
    46: 1          0          0          0          0
    48: 3          0          0          0          0
Others: 0          0          5          0          0
Egress
  dscp: incoming   no_change  classified policed    dropped (in pkts)
    0 : 62            n/a       n/a      0          0
    32: 5             n/a       n/a      0          0
    46: 21            n/a       n/a      0          0
    48: 77            n/a       n/a      0          0
Others: 221           n/a       n/a      0          0
```

Note that R4 received 5 packets marked with internal DSCP of 32 which corresponds to the input CoS of 4:

```
Rack1SW4#show mls qos maps cos-dscp
   Cos-dscp map:
        cos:   0  1  2  3  4  5  6  7
     --------------------------------
       dscp:   0  8 16 24 32 40 48 56
```

There are also 21 packets from the TCP sessions initiated off R1 and marked with DSCP value of EF.

Now we are going to send IPX packets from R6 to R4 across SW1. To enforce this path, we shutdown all SW2 links connected to other switches in the topology. After that, send IPX packets from R6 to R4.

```
SW2:
interface range FastEthernet 0/16 - 21
 shutdown

Rack1R6#ping 146.000f.8f0f.6101

Translating "146.000f.8f0f.6101"

Type escape sequence to abort.
Sending 5, 100-byte IPX Novell Echoes to 146.000f.8f0f.6101, timeout is
2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
Rack1R6#
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
Rack1SW4#show mls qos interface fastEthernet 0/4 statistics
FastEthernet0/4
Ingress
  dscp: incoming   no_change  classified policed    dropped (in pkts)
    0 : 200        190        158        0          0
    32: 0          0          0          0          0
    46: 1          0          0          0          0
    48: 157        0          0          0          0
Others: 0          0          10         0          0
Egress
  dscp: incoming   no_change  classified policed    dropped (in pkts)
    0 : 1521          n/a        n/a     0          0
    32: 10            n/a        n/a     0          0
    46: 21            n/a        n/a     0          0
    48: 3291          n/a        n/a     0          0
Others: 9683          n/a        n/a     0          0
```

As you can see from the statistics output on SW4, now we got 5 more packets with internal DSCP of 32. Those are the packets sent from R6 across SW1. Since they traversed trunks that have VLAN-based QoS enabled, the switch classified them according to the VLAN policy.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.67 Catalyst QoS Port-Based Policing and Marking

- Configure SW4's port connected to R4 as follows:
    - Limit the input rate of ICMP packets to 64Kbps. Meter using a burst size value of 16Kbytes. Trust DSCP values in conforming packets and re-mark exceeding traffic down to CS1.
    - Irrespective of the DSCP value, use a CoS value of two for ICMP packets entering the switch.
    - Mark traffic from WWW servers with DSCP CS2. If the traffic exceeds 256 Kbps, re-mark it down to CS1. Assume the burst size of 32000 bytes.
    - Meter the rate of non-IP traffic at 128Kbps using the minimum burst size. Set a CoS value of 0 for conforming traffic, and re-mark exceeding packets with a CoS value of 1.
    - Assume that incoming ICMP packets are pre-colored either with DSCP AF31 or DSCP CS3.
- Configure SW1 to limit the total rate of traffic coming from R1 to 128Kbps. Use a large enough burst size to accommodate 10ms of traffic generated by R1 transmitting at the full 100Mbps interface rate.

Configuration

```
SW1:
mls qos
!
policy-map POLICE
 class class-default
  police 128000 125000
!
interface FastEthernet 0/1
 service-policy input POLICE

SW4:
mls qos
!
! Allow setting CoS in policy-map
!
mls qos cos policy-map
!
ip access-list extended ICMP
 permit icmp any any
!
ip access-list extended HTTP
 permit tcp any eq 80 any
!
mac access-list extended ALL_NON_IP
 permit any any 0x0 0xFFFF
!
class-map ICMP
 match access-group name ICMP
!
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
class-map HTTP
 match access-group name HTTP
!
class-map ALL_NON_IP
 match access-group name ALL_NON_IP
!
policy-map POLICE_INBOUND
 class ICMP
  trust dscp
  set cos 2
  police 64000 16000 exceed-action policed-dscp-transmit
 class HTTP
  set dscp cs2
  police 256000 32000 exceed-action policed-dscp-transmit
 class ALL_NON_IP
  set dscp 0
  police 128000 8000 exceed-action policed-dscp-transmit
!
! Mapping original DSCP values to new DSCP values
! We mention all DSCP values that could be potentially
! assigned to inbound traffic.
!
mls qos map policed-dscp 0 16 24 26 to 8
!
interface FastEthernet 0/4
 no service-policy input PER_PORT_PER_VLAN
 service-policy input POLICE_INBOUND
```

Verification

Note

Along with the classification settings in policy-maps, you may also set policer parameters to control the rates of input traffic. The policer applies only to the traffic matching the respective class. The only two metering parameters are average rate and burst size. The switch always accepts conforming traffic, and for exceeding traffic, you have only two options: drop or set policed DSCP and transmit.

The switch determines the markdown DSCP based on the global mapping table. This table specifies original DSCP and the DSCP applied when policer implements markdown actions. As usual, if the packet carries Layer 2 and Layer 3 markings, the switch uses DSCP to CoS mapping table to find the new CoS value.

If you apply a per-port policy to a trunk port, it applies to all VLANs traffic at the same time (e.g. IP traffic from VLAN1, VLAN2 etc).

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

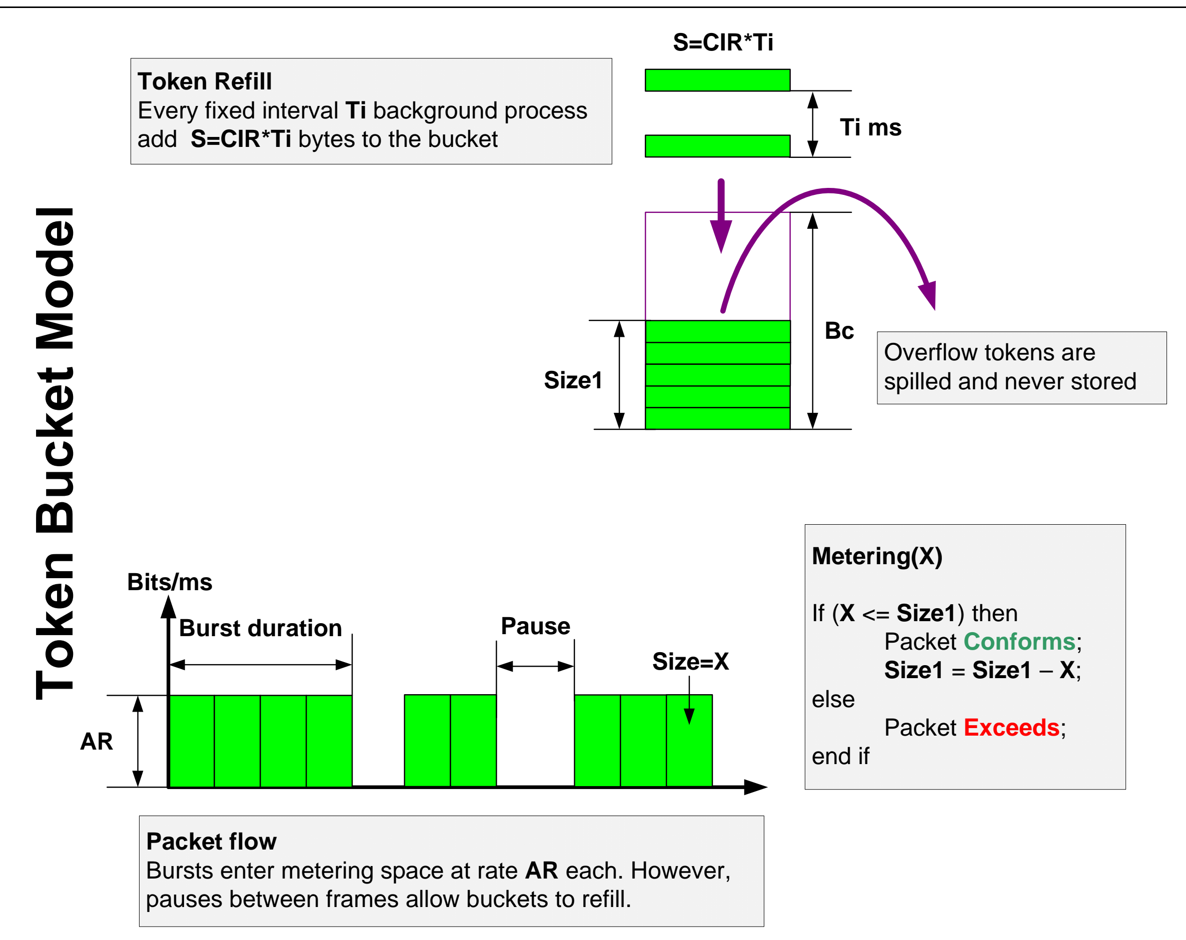
In our task, the burst size for R1 is 100Mbps*10ms/8 = 125000. Note that the switch will round up the burst size to the closest value acceptable for hardware optimization.

Another special thing is handling ICMP traffic. Per the requirements, we trust the DSCP value and remark it down to CS1 if the traffic exceeds. The problem is in general we don't know the DSCP values ahead of time, unless someone else pre-colors them. In this particular case, we know that traffic may carry either CS3 or AF31 DSCP code-points and we can configure policed DSCP mappings accordingly (24 and 26 map to 8). In real life, it's common to first set a fixed DSCP value and mark it down for exceeding traffic, like with do with HTTP class (16 maps to 8).

Finally, we set the CoS value in parallel with trusting DSCP value for the ICMP traffic. This results in Layer 2 marking calculated independently of DSCP. Effectively, not even the markdown action affects the CoS value – it remains the same irrespective of the policer action and DSCP changes.

As for the actual implementation of policing on the Catalyst switches, ASICs allow policing at high rates. Such optimization results in certain limitations imposed on rates and bursts sizes. The Catalyst switches use the familiar token bucket algorithm, which we discussed in the section dedicated to CAR, as follows.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com



Here **Ti** value is fixed at 8000 times per second (1/8000, or 0,125ms). From the equation S=CIR*Ti and based on the fact that Bc>=S we can see that Bc should be not less than CIR/(8*8000). For example with CIR=100Mbps this value is around 1600 bytes. However, the minimum allowed burst size in the 3550/3560 models is 8000 bytes so you don't have to worry about those details.

The switch's capability of remarking traffic allows changing DSCP values only. You cannot customize markdown settings per policy-map but only set the markdown settings globally using policed-dscp mapping table. This is the limitation of the hardware-optimized platform. The table maps the original DSCP values to their markdown equivalents. Note that the "original" DSCP value is the value taken from the QoS label, not the packet itself. Thus, if you have either a **`trust`** or **`set`** commands under a class map, the switch will take DSCP values for markdown lookup based on the "set " command value or on the trusted element value.

Copyright © 2009 Internetwork Expert

Note one important limitation – policing only applies to hardware switched traffic. If you have traffic destined to the switch's SVI and it ends up on switch's CPU, the policing configuration will not affect such traffic.

For this task, we split verification procedure into three steps. First, we verify the ICMP packets markdown. Per our configuration, the switch should trust the DSCP value, and remark it in the case it exceeds the configured rate.

```
SW1, SW2, SW3:
no mls qos

R1:
interface FastEthernet 0/0
 shutdown

R6:
class-map COS_2
 match cos 2
!
class-map COS_1
 match cos 1
!
! Policy map to count CoS values
!
policy-map MATCH_COS
 class COS_1
 class COS_2
!
! Policy map to detect marked ICMP packets
!
ip access-list extended DETECT
 permit icmp any any dscp af31
 permit icmp any any dscp cs1
 permit ip any any
!
interface FastEthernet 0/0
 service-policy input MATCH_COS
 load-interval 30
!
interface FastEthernet 0/0.146
 ip access-group DETECT in
!
interface FastEthernet 0/0.67
 shutdown
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Now configure R4 to source a stream of packets marked with DSCP AF31 and shaped down to 128Kbps. The switch should remark approximately half of the packets coming into the port down to CS1. However, the packets should remain marked with CoS value of two, no matter what.

```
R4:
interface FastEthernet 0/1.145
 traffic-shape rate 128000 7936 7936 1000

Rack1R6#clear access-list counters

Rack1R6#clear counters fastEthernet 0/0
Clear "show interface" counters on this interface [confirm]

Rack1R4#ping
Protocol [ip]:
Target IP address: 155.1.146.6
Repeat count [5]: 10000000
Datagram size [100]: 1500
Timeout in seconds [2]: 0
Extended commands [n]: y
Source address or interface:
Type of service [0]: 104
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 10000000, 1500-byte ICMP Echos to 155.1.146.6, timeout is 0
seconds:
......................................................................
......................................................................
......................................................................
......................................................................
```

```
Rack1R6#show policy-map interface fastEthernet 0/0
 FastEthernet0/0

  Service-policy input: MATCH_COS

    Class-map: COS_1 (match-all)
      0 packets, 0 bytes
      30 second offered rate 0 bps
      Match: cos  1

    Class-map: COS_2 (match-all)
      892 packets, 1354056 bytes
      30 second offered rate 124000 bps
      Match: cos  2

    Class-map: class-default (match-any)
      9 packets, 4410 bytes
      30 second offered rate 0 bps, drop rate 0 bps
      Match: any

Rack1R6#show ip access-lists
Extended IP access list DETECT
    10 permit icmp any any dscp af31 (437 matches)
    20 permit icmp any any dscp cs1 (435 matches)
    30 permit ip any any (27 matches)
```

From the statistics, it is clear that half of the packets exceed and SW4 remarks them down to DSCP value of CS1. At the same time, we don't see any packet with a CoS value of 1, which means switch does not change the CoS value.

You can verify the HTTP re-marking in the same way. However, this time the switch will modify CoS values according to the policed DSCP value. We will demonstrate a simplified verification. First, set R4 as an HTTP server and start downloading a file from R4 down to R6. Before that, clear MLS QoS statistics counters on the port of SW4 connected to R4 and enable monitoring of DSCP values "0 CS1 and CS2". Ensure that no other switches with except for SW4 have MLS QoS enabled. Don't forget to remove traffic-shaping configuration on R4's VLAN146 interface for this verification.

```
SW1, SW2, SW3:
no mls qos

SW4:
mls qos
!
interface FastEthernet 0/1
 mls qos monitor packets
!
interface FastEthernet 0/4
 mls qos monitor dscp 0 8 16
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
R4:
ip http server
ip http path flash:

Rack1SW4#clear mls qos interface fastEthernet 0/4 statistics

Rack1R6#copy http://admin:cisco@155.1.146.4/c2600-adventerprisek9-
mz.124-18.bin null:
Loading http://***********@155.1.146.4/c2600-adventerprisek9-mz.124-
18.bin
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Rack1SW4#show mls qos interface fastEthernet 0/4 statistics
FastEthernet0/4
Ingress
  dscp: incoming   no_change  classified policed    dropped (in pkts)
    0 : 1142       1          1          0          0
    8 : 0          0          0          0          0
    16: 0          0          1141       731        0
Others: 1          0          0          0          0
Egress
  dscp: incoming   no_change  classified policed    dropped (in pkts)
    0 : 1746          n/a       n/a      0          0
    8 : 0             n/a       n/a      0          0
    16: 0             n/a       n/a      0          0
Others: 84            n/a       n/a      0          0
```

From this output, we can see that all incoming packets are DSCP 0, but the switch classifies them as CS2. Some packets are policed down, though we can't see the new DSCP value.

Now check the CoS monitoring setup at R6 to ensure it receives packets marked with a CoS value of 1 (marked down packets):

```
Rack1R6#show policy-map interface fastEthernet 0/0
 FastEthernet0/0

  Service-policy input: MATCH_COS

    Class-map: COS_1 (match-all)
      32758 packets, 47265183 bytes
      30 second offered rate 1134000 bps
      Match: cos  1
<snip>
```

The last thing to verify now is markdown of non-IP packets. Generate a barrage of IPX packets from R4 to R6 and ensure that you can see CoS 1 packets (exceeding) going down from SW2 on its connection to R6:

```
Rack1R6#show ipx interface fastEthernet 0/0.146
FastEthernet0/0.146 is up, line protocol is up
  IPX address is 146.0011.9221.da80, 802.1Q vLAN (SNAP) [up]

Rack1R4#ping
Protocol [ip]: ipx
Target IPX address: 146.0011.9221.da80
Repeat count [5]: 1000000
Datagram size [100]: 1000
Timeout in seconds [2]: 0
Verbose [n]:
Type escape sequence to abort.
Sending 1000000, 1000-byte IPX Novell Echoes to 146.0011.9221.da80, timeout is
0 seconds:
<snip>

Rack1SW2#show mls qos interface fastEthernet 0/6 statistics
FastEthernet0/6

<snip>
-------------------------------

  0 -  4 :        3258            0            0            0            0
  5 -  7 :           0            0            0
  cos: outgoing
-------------------------------

  0 -  4 :          78         3179            0            0            0
  5 -  7 :           0            0            0
Policer: Inprofile:            0 OutofProfile:            0
```

Note that we constantly get CoS 1 marked packets transmitted on the port connected to R6. Those IPX packets exceeded the configured policer rate in SW4.

The last and the simplest thing to verify is the policer configured on the port connected to R1. Simply send stream of ICMP packets to R6, using large packet sizes.

```
Rack1R1#ping 155.1.146.6 size 1500 repeat 10000

Type escape sequence to abort.
Sending 10000, 1500-byte ICMP Echos to 155.1.146.6, timeout is 2
seconds:
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!.!!!!!!!!!!!!!!!!!!!!.!!!!!!!!!!!!!!!!!!!!.
Success rate is 97 percent (128/131), round-trip min/avg/max = 4/5/9 ms
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Note that the length of the first “burst” is much larger compared to the length of sequential bursts. Here are the calculations why this occurs:

First, we set a policer burst size to 125000 bytes. Divide this value by 1514 byte sized packets and you get 83 packets in a single burst. That means we can “instantly” send 83 packets in the network. However, the router sends each packet with 5ms interval (average RTT shown in the output). This means, while sending those packets, the switch accumulates the following credit:

CIR*Interval = 128000*5ms*83 = 53120 bits = 6640 which is 4 packets of size 1514 bytes.

This means that initially we can send around 87 packets (which is easy to verify by counting the “!” signs).

Next, the policer drops a packet and the router waits 2 seconds of timeout. During this interval, the switch accumulates

CIR*Interval = 128000*2s = 256000 bits = 32000 bytes or 21 packet of size 1514 bytes.

This means that every next burst would be of size 21 packets or a bit larger, thanks to extra-accumulated credit. This fact is easy to verify counting the “!” signs in the second and third “bursts”.

10.68 Catalyst 3560 Per-Port Per-VLAN Policing

- Remove any previous policing configuration on VLAN146 in SW2.
- Configure SW2 to limit the rate of IP traffic entering VLAN146 on every trunk port to 128Kbps.
- Mark all conforming packets with AF21 and exceeding with CS1.
- Use the burst size of twice the minimal size.
- Restrict the amount of IP traffic entering from R6 on VLAN146 to 256Kbps, and drop any exceeding packets. Use the burst size of 32000 bytes.

Configuration

```
SW2:
mls qos map policed-dscp 18 to 8
!
! For second level policy you can only match input interfaces
!
class-map TRUNKS
 match input-interface FastEthernet 0/13 -  FastEthernet 0/21
!
! The second class-map matches a group of ports or a single port
!
class-map PORT_TO_R6
 match input-interface FastEthernet 0/6
!
! IP traffic: ACL and class-map
!
ip access-list extended IP_ANY
 permit ip any any
!
class-map IP_ANY
 match access-group name IP_ANY
!
! Second-level policy-map may only police, but not mark
!
policy-map INTERFACE_POLICY
 class TRUNKS
  police 128000 16000 exceed policed-dscp-transmit
 class PORT_TO_R6
  police 256000 32000
!
! First level policy-map may only mark, not police
! VLAN aggregate policing is not possible in the 3560
!
policy-map VLAN_POLICY
 class IP_ANY
  set dscp af21
  service-policy INTERFACE_POLICY
!
interface Vlan 146
 service-policy input VLAN_POLICY
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
!
! Enable VLAN-based QoS on the ports
!
interface FastEthernet 0/6
 mls qos vlan-based
!
! Some of these are port-channel members, but it's OK
!
interface range FastEthernet 0/13 - 21
 mls qos vlan-based
```

Verification

Note

The first thing to remember about VLAN-based QoS in the 3560 is that you can use either single-level or two-level policy-maps. In the case of single-level policy map, you match classes of traffic globally, on all ports with the VLAN active and enabled for VLAN QoS. You cannot police traffic using single-level policy map, that is, VLAN-wide aggregate policing is not possible in the 3560. However, it is allowed to restrict the amount of traffic entering the VLAN on every specific port.

Think of a VLAN as cloud, with switch ports connected to this cloud. The first level policy applies to VLAN traffic globally, where the second-level policy may only match port ranges and applies policing to individual ports. It is not possible to apply aggregate policing among different ports, even if they are under the same range of a class map.

The syntax is pretty much MQC based, and the policed DSCP configuration is the same as with the port-level policy maps. You can match up to six ports in a class-map assigned to 2nd level policy-map. Separate different ports with spaces and separate ranges by a hyphen. A port range counts as two entries in match clause.

However, there is one limitation specific to dual-level policy-maps. You cannot apply the 2nd level policy map under the class-default, only under user defined classes. Even though in such configuration the class-default will match and mark any traffic, the second-level policy map will not be active.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

The verification consists of two parts. First, we need to verify the limit imposed on traffic entering from a VLAN 146 port. We should verify that it applies only to VLAN 146 traffic and does not affect VLAN 67. To accomplish this, send some traffic destined to R1. Disable MLS QoS in SW1, SW3, and SW4 to avoid any side effects from existing QoS configuration.

```
SW1, SW3, SW4:
no mls qos

Rack1R6#ping 155.1.146.1 repeat 100000 size 1500 timeout 1

Type escape sequence to abort.
Sending 10000, 1500-byte ICMP Echos to 155.1.146.1, timeout is 2
seconds:
!!!!!!!!!!!!!!!!!!!!!!!!.!!!!!!!!!!!!!!!!!!!!!!!!.!!!!!!!!!!!!!!!!!!!!!
!.!!!!!!!!!!!!!!!!!!!!!!!!.!!!!!!!!!!!!!!!!!!!!!!!.
Success rate is 95 percent (115/120), round-trip min/avg/max = 4/5/9 ms

Rack1R6#ping 155.1.67.7 repeat 10000 size 1500 timeout 1

Type escape sequence to abort.
Sending 10000, 1500-byte ICMP Echos to 155.1.67.7, timeout is 1
seconds:
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
<snip>
Success rate is 100 percent (725/725), round-trip min/avg/max = 1/3/8
ms
```

Next, configure R1 to generate a packet flow of rate 256Kbps. Use traffic shaping and ping flooding to accomplish this. After that, observe the statistics for packets leaving towards R6:

```
R1:
interface FastEthernet 0/0
 traffic-shape rate 256000

Rack1R1#ping 155.1.146.6 repeat 100000000 size 1000 timeout 0

Type escape sequence to abort.
Sending 100000000, 1000-byte ICMP Echos to 155.1.146.6, timeout is 0 seconds:
........................................................................
........................................................................
........................................................................
........................................................................
........................................................................
<snip>
```

```
Rack1SW2#show mls qos interface fastEthernet 0/6 statistics
FastEthernet0/6

  dscp: incoming
-------------------------------

<snip>
  dscp: outgoing
-------------------------------

  0 -  4 :           4            0            0            0            0
  5 -  9 :           0            0            0          577            0
 10 - 14 :           0            0            0            0            0
 15 - 19 :           0            0            0          535            0
 20 - 24 :           0            0            0            0            0
 25 - 29 :           0            0            0            0            0
 30 - 34 :           0            0            0            0            0
 35 - 39 :           0            0            0            0            0
 40 - 44 :           0            0            0            0            0
 45 - 49 :           0            0            0            0            0
 50 - 54 :           0            0            0            0            0
 55 - 59 :           0            0            0            0            0
 60 - 64 :           0            0            0            0
  cos: incoming
-------------------------------

  0 -  4 :        1115            0            0            0            0
  5 -  7 :           0            0            0
  cos: outgoing
-------------------------------

  0 -  4 :           5          658          612            0            0
  5 -  7 :           0            0            0
Policer: Inprofile:            0 OutofProfile:            0
```

Note that the number of CS1 and AF21 packet going to R6 is approximately the same. This is because the switch marks approximately half of the packets as exceeding and changes their DSCP values according to the mapping table.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.69 Catalyst 3550 Per-Port Per-VLAN Policing

- Remove any previously attached service-policy on the port connected to the VLAN 146 interface of SW4
- Limit the rate of IP traffic from R4 on VLAN 146 to 128Kbps. Mark conforming packets as AF11 and exceeding as CS1
- Limit the rate of non-IP traffic from R4 on VLAN 146 to 256Kbps and drop the exceeding packets. Mark conforming packets with a CoS value of 3
- The configuration should not affect any future VLANs trunked to R4.

Configuration

```
SW4:
mls qos
mls qos map policed-dscp 10 to 8
!
! Acess-Lists for IP and non-IP traffic
!
ip access-list extended IP_ANY
 permit IP any any
!
mac access-list extended NON_IP_ANY
 permit any any 0x0 0xFFFF
!
! Second-level class-maps
!
class-map IP_ANY
 match access-group name IP_ANY
!
class-map NON_IP_ANY
 match access-group name NON_IP_ANY
!
! First-level class-maps
!
class-map VLAN_146_IP
 match vlan 146
 match class-map IP_ANY
!
class-map VLAN_146_NON_IP
 match vlan 146
 match class-map NON_IP_ANY
!
! Since we are not limited, we use any burst values
! Policy map matches first-level class-maps
!
policy-map POLICE_PER_PORT_PER_VLAN
 class VLAN_146_IP
   set dscp af11
   police 128000 16000 exceed-action policed-dscp-transmit
 class VLAN_146_NON_IP
   set dscp cs3
   police 256000 32000
```

```
!
! Apply the policy-map
!
interface FastEthernet 0/4
  no service-policy input POLICE_INBOUND
  service-policy input POLICE_PER_PORT_PER_VLAN
```

Verification

> **Note**
>
> Per-Port per-VLAN policing on the 3550 is very similar to the per-port per-VLAN classification model. You simply add policing commands under the 1st level class-maps in the policy map. The restrictions for per-VLAN classification syntax remain the same.
>
> For the purpose of verification, we will temporarily trunk another VLAN between R4 and SW1.

```
SW1, SW2, SW3, SW4:
vlan 47

SW1:
interface Vlan 47
 ip address 155.1.47.7 255.255.255.0

SW4:
interface FastEthernet 0/4
 switchport trunk encapsulation dot1q
 switchport mode trunk
 switchport trunk allowed vlan 47,146

R4:
interface FastEthernet 0/1
 no ip address
!
interface FastEthernet 0/1.146
 encapsulation dot1q 146
 ip address 155.1.146.4 255.255.255.0
!
interface FastEthernet 0/1.47
 encapsulation dot1q 47
 ip address 155.1.47.4 255.255.255.0
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

We need to verify three things:

1) IP traffic on VLAN146 from R4 is marked with AF11 and remarked with CS1 when it exceeds the configured rate
2) non-IP traffic is marked as CS3 and dropped when it exceeds configured rate
3) Traffic on VLAN14 is not affected by the policing and classification process

To test the non-IP traffic we need to enable IPX on the interfaces of R4 and R6.

```
R4:
ipx routing
!
interface FastEthernet 0/1.146
 ipx network 146 encapsulation snap

R6:
ipx routing
!
interface FastEthernet 0/0.146
 ipx network 146 encapsulation snap
```

We start by verifying traffic on VLAN 47:

```
Rack1R4#ping 155.1.47.7 repeat 1000 size 1500

Type escape sequence to abort.
Sending 1000, 1500-byte ICMP Echos to 155.1.47.7, timeout is 2 seconds:
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!
Success rate is 100 percent (1000/1000), round-trip min/avg/max = 1/3/12 ms
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Now configure R4 to source ICMP packets at rate of 256Kbps to R6. Also, enable DSCP monitoring in SW4 to watch packets marked with internal DSCP values of 10 (AF11) and other DSCP values used in the task:

```
SW4:
interface FastEthernet 0/1
 mls qos monitor packets
!
interface FastEthernet 0/4
 mls qos monitor dscp 0 8 10 24

R4:
interface FastEthernet 0/1.146
 traffic-shape rate 256000

Rack1SW4#clear mls qos interface fastEthernet 0/4 statistics

Rack1R4#ping 155.1.146.6 repeat 10000 size 1500 timeout 0

Type escape sequence to abort.
Sending 10000, 1500-byte ICMP Echos to 155.1.146.6, timeout is 0 seconds:
......................................................................
......................................................................
......................................................................
......................................................................
......................................................................
<snip>

Rack1SW4#show mls qos interface fastEthernet 0/4 statistics
FastEthernet0/4
Ingress
  dscp: incoming   no_change  classified policed    dropped (in pkts)
    0 : 332        6          18         0          0
    8 : 0          0          0          0          0
    10: 0          0          331        153        0
    24: 0          0          0          0          0
Others: 23         0          0          0          0
Egress
  dscp: incoming   no_change  classified policed    dropped (in pkts)
    0 : 28            n/a       n/a      0          0
    8 : 0             n/a       n/a      0          0
    10: 0             n/a       n/a      0          0
    24: 0             n/a       n/a      0          0
Others: 57            n/a       n/a      0          0
```

Note that count of the “policed” packets is close to the half of “classified” packets. The latter are the packets we assigned DSCP to, and the former are the packets that have their DSCP marking changed by the policer.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

The last thing to verify is checking the limit imposed on the IPX packets rate. Ensure you removed the policer on the VLAN146 interface of R4 before running the test:

```
Rack1R6#show ipx interface fastEthernet 0/0.146
FastEthernet0/0.146 is up, line protocol is up
  IPX address is 146.0012.d9ed.8c80, 802.1Q vLAN (SNAP) [up]
<snip>

R4:
interface FastEthernet 0/1.146
 no traffic-shape rate

Rack1R4#ping
Protocol [ip]: ipx
Target IPX address: 146.0012.d9ed.8c80
Repeat count [5]: 10000
Datagram size [100]: 1200
Timeout in seconds [2]: 1
Verbose [n]:
Type escape sequence to abort.
Sending 10000, 1200-byte IPX Novell Echoes to 146.0012.d9ed.8c80, timeout is 1
seconds:
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!.!!!!!!!!!!!!!!!!!!!!!!!!!!!!.!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!.!!!!!!!!!!!!!!!!!!!!!!!!!!!!!.!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!.!!!!!!!!!!!!!!!!!!!!!!!!!!!!!.!!!!!!!!!!!!!!!!!!!!!!!!!!!!!.
!!!!!!!!!!!!!!!!!!!!!!!!!!!!.!!!!!!!!!!!!!!!!!!!!!!!!!!!!!.!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!.!!!!!!!!!!!!!!!!!!!!!!!!!!!!!.
Success rate is 96 percent (318/329), round-trip min/avg/max = 1/3/5 ms

Rack1SW4#show mls qos interface fastEthernet 0/4 statistics
FastEthernet0/4
Ingress
  dscp: incoming   no_change  classified policed    dropped (in pkts)
    0 : 2183       211        581        0          0
    8 : 0          0          0          0          0
    10: 0          0          519        153        0
    24: 0          0          329        0          11
Others: 774        0          0          0          0
Egress
  dscp: incoming   no_change  classified policed    dropped (in pkts)
    0 : 2215          n/a       n/a      0          0
    8 : 0             n/a       n/a      0          0
    10: 0             n/a       n/a      0          0
    24: 0             n/a       n/a      0          0
Others: 2047          n/a       n/a      0          0
```

Note the number of packets classified with internal DSCP value of 24 (CS3) and the number of dropped packets.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.70 Catalyst QoS Aggregate Policers

- Limit the aggregate rate of ICMP and IPX traffic coming from R1 on SW1 to 256Kbps, and drop exceeding packets.
- Trunk a new VLAN 47 to the port of R4 connected to SW4.
- Limit the aggregate rate of IP traffic to 128Kbps in both VLAN 146 and the new VLAN 47. Mark conforming packets with DSCP AF11 and remark packets to CS1 if they exceed.

Configuration

```
SW1:
mls qos
!
! Access-lists for ICMP pakets
!
ip access-list extended ICMP
 permit icmp any any
!
! Access-list for IPX packets
!
mac access-list extended IPX
 permit any any 0x8137 0x0
!
! Class-maps for traffic flows
!
class-map ICMP
 match access-group name ICMP
!
class-map IPX
 match access-group name IPX
!
! Aggregate policer
!
mls qos aggregate-policer AGG128 128000 16000 exceed-action drop
!
! Policy-map that uses the aggregate policer
!
policy-map POLICE_AGGREGATE
 class ICMP
  police aggregate AGG128
 class IPX
  police aggregate AGG128
 class class-default
  set dscp cs1
!
interface FastEthernet 0/1
 service-policy input POLICE_AGGREGATE
```

```
SW4:
!
! IP traffic on the new VLAN (first level class-map)
! Access-lists and second level class-maps
! have been already created
!
class-map VLAN_47_IP
 match vlan 47
 match class-map IP_ANY
!
mls qos aggregate-policer AGG256 256000 32000 exceed-action drop
!
! Since we are not limited, we may use any burst values
! Policy map matches first-level class-maps as usual
!
policy-map POLICE_PER_PORT_PER_VLAN
 class VLAN_146_IP
   set dscp af11
   no police 128000 16000 exceed-action policed-dscp-transmit
   police aggregate AGG256
 class VLAN_47_IP
   set dscp af11
   police aggregate AGG256
!
! Apply the policy-map
!
interface FastEthernet 0/4
  switchport trunk allowed vlan add 47
  no service-policy input POLICE_PER_PORT_PER_VLAN
  service-policy input POLICE_PER_PORT_PER_VLAN
```

Verification

Note

An aggregate policer applies to multiple traffic classes on the same interface. You define the aggregate policer rate, burst, and action globally using a unique name. Later, you apply this policer under the respective classes of a policy-map. All classes configured with the same aggregate policer share the configured rate.

The restrictions are that you cannot aggregate metered rate across different physical ports. In the 3560 model it is impossible to insert an aggregate policer inside a second-level policy map. Thus, you may *only* aggregate the bitrate between VLANs on the same interface if you are using the 3550 model.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

For the purpose of verification, we will configure IP addressing in VLAN 47 on SW1 and R4:

```
SW1, SW2, SW3, SW4:
vlan 47

SW1:
interface Vlan 47
 ip address 155.1.47.7 255.255.255.0

SW4:
interface FastEthernet 0/4
 switchport trunk encapsulation dot1q
 switchport mode trunk
 switchport trunk allowed vlan 47,146

R4:
interface FastEthernet 0/1
 no ip address
!
interface FastEthernet 0/1.146
 encapsulation dot1q 146
 ip address 155.1.146.4 255.255.255.0
!
interface FastEthernet 0/1.47
 encapsulation dot1q 47
 ip address 155.1.47.4 255.255.255.0
```

Now start transferring a large file from R4 across both VLANs 146 and 47.

```
R4:
ip http server
ip http path flash:

Rack1R1#$copy http:/admin:cisco@155.1.146.4/c2600-adventerprisek9-
mz.124-10.bin null:
Loading http://***********@155.1.146.4/c2600-adventerprisek9-mz.124-
10.bin
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!

Rack1SW1#copy http:/admin:cisco@155.1.47.4/c2600-adventerprisek9-
mz.124-10.bin null:
Loading http://admin:cisco@155.1.47.4/c2600-adventerprisek9-mz.124-
10.bin !!!
```

Wait a few minutes for transfer rates to stabilize. Now collect input and output rate statistics as shown below (note that all load-intervals have value of 30 seconds):

```
Rack1R1#show interfaces fastEthernet 0/0
FastEthernet0/0 is up, line protocol is up
<snip>
  30 second input rate 125000 bits/sec, 16 packets/sec
  30 second output rate 11000 bits/sec, 22 packets/sec

Rack1SW1#show interfaces vlan47
Vlan47 is up, line protocol is up
<snip>
  30 second input rate 142000 bits/sec, 18 packets/sec
  30 second output rate 14000 bits/sec, 27 packets/sec
<snip>

Rack1R4#show interface fastEthernet 0/1
FastEthernet0/1 is up, line protocol is up
<snip>
  30 second input rate 25000 bits/sec, 48 packets/sec
  30 second output rate 340000 bits/sec, 38 packets/sec
<snip>

Rack1SW4#show mls qos interface fastEthernet 0/4 statistics
FastEthernet0/4
Ingress
  dscp: incoming   no_change  classified policed    dropped (in pkts)
    0 : 8645       27         50         0          0
    8 : 0          0          0          0          0
    10: 0          0          8667       0          1693
    24: 0          0          0          0          0
Others: 99         0          0          0          0
Egress
  dscp: incoming   no_change  classified policed    dropped (in pkts)
    0 : 10647         n/a       n/a      0          0
    8 : 0             n/a       n/a      0          0
    10: 0             n/a       n/a      0          0
    24: 0             n/a       n/a      0          0
Others: 269           n/a       n/a      0          0
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Calculate the following:

R1_Input_Rate + SW1_Input_Rate = 142000+125000 = 267000
R4_Output_Rate = 340000
Packet_Drop_Rate=1693/8667*100%=20%

If you multiply *R4_Output_Rate* by *Packet_Drop_Rate* this value will be close to *R4_OutputRate – R1_Input_Rate – SW1_Input_Rate*. The conclusion is that switch limits aggregate rate for traffic from both VLANs.

Now we need to ensure the traffic rate limit imposed on IP and non-IP packets from R1. To verify that, send a flood of IPX and ICMP packets from R1 towards R6. Configure R1 and R6 for IPX packet exchange first:

```
R1:
ipx routing
!
interface FastEthernet 0/0
 ipx network 146 encapsulation snap

R6:
ipx routing
!
interface FastEthernet 0/0.146
 ipx network 146 encapsulation snap
```

Now generate two traffic flows from R1: one with ICMP packets and the other one with IPX packets.

```
Rack1R1#ping 155.1.146.6 repeat 1000000 size 1000 timeout 0

Type escape sequence to abort.
Sending 1000000, 1000-byte ICMP Echos to 155.1.146.6, timeout is 0 seconds:
......................................................................
......................................................................
......................................................................
......................................................................
......................................................................

Rack1R6#show ipx interface fastEthernet 0/0.146
FastEthernet0/0.146 is up, line protocol is up
  IPX address is 146.0011.2042.55e0, 802.1Q vLAN (SNAP) [up]
```

```
Rack1R1#ping
Protocol [ip]: ipx
Target IPX address: 146.0011.2042.55e0
Repeat count [5]: 10000000
Datagram size [100]: 1000
Timeout in seconds [2]: 0
Verbose [n]:
Type escape sequence to abort.
Sending 10000000, 1000-byte IPX Novell Echoes to 146.0011.2042.55e0, timeout is
0 seconds:
......................................................................
......................................................................
......................................................................
......................................................................
......................................................................
......................................................................
......................................................................
.......................
```

Check the output packet rate on the connection to R6 and the policer statistics on connection to R1:

```
Rack1SW2#show interfaces fastEthernet 0/6
FastEthernet0/6 is up, line protocol is up (connected)
  Hardware is Fast Ethernet, address is 001b.d4df.c308 (bia 001b.d4df.c308)
  MTU 1500 bytes, BW 100000 Kbit, DLY 100 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  Keepalive set (10 sec)
  Full-duplex, 100Mb/s, media type is 10/100BaseTX
  input flow-control is off, output flow-control is unsupported
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input 00:00:42, output 00:00:01, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: fifo
  Output queue: 0/40 (size/max)
  30 second input rate 124000 bits/sec, 15 packets/sec
  30 second output rate 125000 bits/sec, 16 packets/sec
<snip>
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
Rack1SW1#show mls qos interface fastEthernet 0/1 statistics
FastEthernet0/1

<snip>

  cos: incoming
-------------------------------

  0 -  4 :      429084           0            0            0            0
  5 -  7 :           0           0            0
  cos: outgoing
-------------------------------

  0 -  4 :       16289           0            0            0            0
  5 -  7 :           0           1            1
Policer: Inprofile:        12527 OutofProfile:       372720
```

We can see that SW1 limits the aggregate input rate to a value close to the target 128Kbps by dropping the exceeding packets in both classes.

10.71 Catalyst 3560 Ingress Queueing

- Enable MLS QoS on SW1 and trust IP precedence on the port connected to R1.
- Configure ingress queue 1 as priority queue with 15% of ring bandwidth.
- Map all packets marked with CoS values of 5 to the priority queue.
- Ensure all other packets map to the non-priority queue.
- The bandwidth remaining after the priority queue should be shared between ingress queues in proportions 1:3 for priority and non-priority queues respectively.

Configuration

```
SW1:
mls qos
!
! Set queue 1 as priority queue
! Set the bandwidth limit to 15%
!
mls qos srr-queue input priority-queue 1 bandwidth 15
!
! SRR weights for input queue 1 and 2
!
mls qos srr-queue input bandwidth 10 30
!
! CoS to Queue-ID mapping. Queue ID is followed by CoS list
!
mls qos srr-queue input cos-map queue 1 5
mls qos srr-queue input cos-map queue 2 1 2 3 4 6 7
!
! Trust precedence on the port connected to R1
!
interface FastEthernet 0/1
 mls qos trust ip-precedence
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Verification

Note

Ingress queues are unique to the 3560 switches due to their internal ring architecture. There are two ingress queues acting as a “shim’ layer between all ports and the internal ring. The switch queues input traffic after the classification and marking stage and uses the QoS label assigned to a packet to map it to the respective input queue, if the internal backplane experiences congestion. There are exist two separate mapping tables: CoS to Queue-ID and DSCP to Queue-ID. Depending on the value you trusted or set on ingress (DSCP, IP Precedence or CoS) the switch selects the respective mapping table. You may learn the existing mappings using the commands **`show mls qos maps dscp-input-q`** or **`show mls qos maps cos-input-q`**.

All ingress queue settings are global. In this task, we configure mappings for ingress queues as well as queue weights. The queue weights define the logic of input packet scheduler. Input packet scheduler is a variation of Round Robin (RR) procedure called Shared Round Robin or SRR. It has some additional modifications over classic Round Robin to increase accuracy and swiching performance. The documentation is really scarce on this topic, but the logic is as follows:

1) You may configure one of the queues (Queue 1 or Queue 2) as priority.
2) If queue is a priority queue, you assign it a bandwith threshold value expressed in percents ranging from 0% to 40%. Those percents define the amount of *internal ring bandwidth* available to priority queue.
3) Both Queue 1 and Queue 2 has additional SRR weights, used by the scheduler.

The scheduling process works as follows, every time quantum:

a) If a queue is the priority queue, service it in exclusive mode up to the amount of packets enough not to fill the intenal ring over the configured threshold.
b) If the scheduler reaches the threshold for the priority queue, it stops servicing it in exclusive mode.
c) At that point the scheduler uses the weighted round robin procedure between Queue 1 and Queue 2, sharing the remaining bandwidth between queues proportional to their SRR weights.
d) If there is no priority queue defined, service both queues using SRR weighted round robin logic.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

The general logic with priority queue mode is to service the priority queue up to the maximum rate but then schedule all exceeding packets using “fair” round robin scheduling. This is similar to the process used by RSVP policer, which does not drop exceeding flow traffic but rather services it as best-effort using lowest WFQ weight.

The actual implementation of SRR scheduler is not important to us. Rather, the general idea of SRR is that “n” number of queues of weights W(1)…W(n) will have shares of bandwidth proportional to

Share(i)=W(i)/(W(1)+...W(n)) for queue “i”.

Therefore, the absolute values of SRR weight do no matter, but only their proportions (relative values). The actual implemtnation may use those weights as extra credits added to deficit counter of every queue.

Consider the example where Queue 1 and Queue 2 are set with SRR weights “10” and “20” while Queue 2 is in priority mode with bandwidth limit of 20%. Assuming that internal ring bandwidth is 10Gbps we get the following:

1) Queue1 may send traffic up to 20%*10Gbps=2Gbps.
2) The remaining bandwidth (8Gbps) is shared in proportions 10:20 between the exceeding packets from Queue 1 and all packets from Queue 2.

Thus in result Queue 1 has guaranteed 8Gbps*10/(10+20)=2.6Gbps of ring bandwidth and Queue 2 has guranteed 2Gbps+8Gbps*20(10+20)=7.3Gbps of ring bandwidth. At the same time, if Queue 2 traffic does not exceed 2Gbps the scheduler treats it as priority.

As another example, consider both queues configured in regular mode with SRR weights of 33 and 66. Assuming that internal ring bandwidth is 5Gbps, we conclude that Queue 1 guaranteed bandwidth is 1.65Gbps and Queue 2 guaranteed bandwidth is 3.3Gbps.

As usual with max-min sharing procedure, if any queue does not use its minimum guarantee bandwidth, the other queue may use the remaining bandwidth.

In the configuration above, we trust IP precedence in the incoming packets. This will automatically enable trusting CoS in non-IP packets. In both cases, the switch deduces the equivalent CoS value using the DSCP to CoS mapping table. Thus we only map CoS values to the respective queues, performing “aggregate”.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

There is no easy way to verify the actual working of ingress queues unless you really congest the switch. Therefore, we will limit verifications just to some show commands.

```
Rack1SW1#show mls qos input-queue
Queue     :       1       2
----------------------------------------------
buffers   :      90      10
bandwidth :      10      30
priority  :      15       0
threshold1:     100     100
threshold2:     100     100

Rack1SW2#show mls qos maps dscp-input-q
   Dscp-inputq-threshold map:
     d1 :d2    0     1     2     3     4     5     6     7     8     9
     ------------------------------------------------------------
      0 :    02-01 01-01 01-01 01-01 01-01 01-01 01-01 01-01 02-01 01-01
      1 :    01-01 01-01 01-01 01-01 01-01 01-01 02-01 01-01 01-01 01-01
      2 :    01-01 01-01 01-01 01-01 02-01 01-01 01-01 01-01 01-01 01-01
      3 :    01-01 01-01 02-01 01-01 01-01 01-01 01-01 01-01 01-01 01-01
      4 :    01-01 02-01 02-01 02-01 02-01 02-01 02-01 02-01 02-01 01-01
      5 :    01-01 01-01 01-01 01-01 01-01 01-01 02-01 01-01 01-01 01-01
      6 :    01-01 01-01 01-01 01-01

Rack1SW2#show mls qos maps cos-input-q
   Cos-inputq-threshold map:
              cos:  0   1   2   3   4   5   6   7
              ------------------------------------
  queue-threshold: 1-1 2-1 2-1 2-1 2-1 1-1 2-1 1-1
```

As you can see, the SRR weights are 10 and 30, while priority bandwidth for queue 1 is set to 15% of the internal ring bandwidth.

The mapping tables are for DSCP and CoS values. In the DSCP to queue-threshold mapping table you see DSCP values split in rows and columns. Rows are for the first digits of DSCP codepoints and columns are for the second digits. There are 64 DSCP codepoints and the table is partitioned accordingly. In the cells, you can see “queue_id-theshold_id” pairs, which specify the queue number and the threshold the value maps to. Note that DSCPs corresponding to IP Precedence 6 and 7 map to queue 2, not the default queue 1. We will discuss thresholds in more details in the next task.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.72 Catalyst 3560 Ingress Queue Tuning

- Configure drop thresholds on SW1 for the non-priority ingress queue to 80% and 100%.
- Ensure that routing and network control traffic mapped to queue 2 is less likely to be dropped than regular traffic.
- Ensure that the queue 2 buffer space is 3 times larger than the priority queue's size.

Configuration

```
SW1:
mls qos
!
! Change the threholds for Queue 2. Thresholds are in percents
! Queue ID is folloed by two threshold values
!
mls qos srr-queue input threshold 2 80 100
!
! Map CoS corresponding to network control and IP routing
! traffic to Threshold 2
!
mls qos srr-queue input cos-map threshold 2 6 7
!
! Share buffers in proportions 1:3 between Q1 and Q2
!
mls qos srr-queue input buffers 25 75
```

Verification

Note

Each input queue has the following configurable parameters:

1) Drop thresholds. There are *three* thresholds per queue, and by default, they are set to 100% of the queue size. You can only set thresholds in percents, not the absolute values. Note that the third threshold value is locked to "queue full" state or 100% of queue size and *cannot* be changed. You can still map DSCP and CoS values to the third threshold.

2) Buffer space partitioning. Both queues share the same buffer space, and you can divide it using relative weights expressed in percents. By default, first queue has 90% of buffer space and the second queue has 10% of queue space.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

The drop strategy used by SRR is known as WTD – weighted tail drop. That is, every queue has a number of drop thresholds (three in the 3560) and you map codepoints to these values. For example, you may want to map voice or control packets to the maximum drop threshold and map less priority traffic to a lower threshold. Per the default configuration, all codepoints map to the first drop threshold, which is set to 100% of the queue size.

Note that you can map DSCP and CoS values to threshold in two different ways: using commands:

```
mls qos srr-queue input {dscp-map|cos-map} threshold <t-id>
<List>
```

And

```
mls qos srr-queue input {dscp-map|cos-map} queue <q-id>
threshold <t-id> <List>.
```

The first form seems to be deprectated as if you enter the first command, the CLI will automatically convert it to the second format.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Use the following show commands to display your new settings:

```
Rack1SW1#show mls qos input-queue
Queue     :       1       2
----------------------------------------------
buffers   :      25      75
bandwidth :      10      30
priority  :      15       0
threshold1:     100      80
threshold2:     100     100

Rack1SW1#show mls qos maps dscp-input-q
   Dscp-inputq-threshold map:
     d1 :d2    0     1     2     3     4     5     6     7     8     9
     ------------------------------------------------------------
      0 :    02-01 01-01 01-01 01-01 01-01 01-01 01-01 01-01 02-01 01-01
      1 :    01-01 01-01 01-01 01-01 01-01 01-01 02-01 01-01 01-01 01-01
      2 :    01-01 01-01 01-01 01-01 02-01 01-01 01-01 01-01 01-01 01-01
      3 :    01-01 01-01 02-01 01-01 01-01 01-01 01-01 01-01 01-01 01-01
      4 :    01-01 02-01 02-01 02-01 02-01 02-01 02-01 02-01 02-02 01-01
      5 :    01-01 01-01 01-01 01-01 01-01 01-01 02-02 01-01 01-01 01-01
      6 :    01-01 01-01 01-01 01-01

Rack1SW1#show mls qos maps cos-input-q
   Cos-inputq-threshold map:
              cos:  0   1   2   3   4   5   6   7
              ------------------------------------
  queue-threshold: 1-1 2-1 2-1 2-1 2-1 1-1 2-2 1-2
```

You can see the new buffer settings, and the new queue-id and threshold mappings for DSCP and CoS values.

Copyright © 2009 Internetwork Expert

10.73 Catalyst 3550 Egress Queueing

- Network traffic consists of the following traffic classes:
  - VoIP bearer packets marked with DSCP EF
  - Interactive video traffic marked with DSCP AF41
  - VoIP signaling traffic marked with DSCP CS3
  - Interactive Citrix traffic marked with DSCP AF21
  - Best-effort traffic marked with DSCP 0
- Implement the following traffic policy on SW4’s connection to R4:
  - Provide priority treatment for VoIP bearer packets
  - Provide a separate queue for video traffic and guarantee it 30% of the remaining bandwidth
  - Provide a separate queue for interactive traffic and VoIP signaling with 20% of the bandwidth
  - The remaining bandwidth should be used by best-effort traffic
- Ensure no other traffic but voice can use the priority queue

Configuration

```
SW4:
mls qos
!
interface FastEthernet 0/4
 wrr-queue bandwidth 50 20 30 1
 wrr-queue cos-map 4 5
 wrr-queue cos-map 3 4 6 7
 wrr-queue cos-map 2 2 3
 wrr-queue cos-map 1 0 1
 priority-queue out
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Verification

Note

The 3550 switches have two types of ports: Gigabit and FastEthernet, and they use different ASICs. As we see later this allows for more features on Gigabit ports. However, both types use the same queuing scheduler known as WRR (Weighted Round Robin). It is a modification of well-known Round Robin procedure, which is similar to SRR used in the 3560 but has different implementation. So far, we only know that SRR is more effective than WRR.

WRR works with four egress queues per port. That means you can define up to four traffic classes and define bandwidth sharing between them. Compare this to dynamic queue capabilites of WFQ, where you can have up to 4096 flows of traffic or CBWFQ where you can define 64 classes. Obviously, this is the limitation of highly optimized hardware-based QoS.

Each of the four egress queues has numerical weight values assigned. The WRR scheduler uses these values to share the bandwidth between classes. The absolute values of the weights are not important, for the bandwidth share of class "j" is

S(j)=W(j)/(W(1)+...W(4)),

where W(1),...,W(4) are the weight values of all four classes. If you are interested in the actualy implementation, it seems that the weight values are used to increment the deficit round robin counter by an amount of tokens proportional to the weight value. Thus, setting larger absolute weight values may slightly increase packet latency, though is hardly possible on high-speed interfaces. You can set the WRR weight for a queue using the interface-level command **`wrr-queue bandwidth <w1> <w2> <w3> <w4>`** where **`<wj>`** is the weight for queue "j". Remember that different absolute weight values may result in the same relative shares of bandwidth.

It is possible to designate queue 4 as priority queue on the 3550 switch. In this case, WRR scheduler always empties this queue first. The danger is that there is no restriction on the priority queue rate, and you must ensure the use of ingress policing for admisstion control on the priority traffic. When you enable priority-queue services using the interface-level command **`priority-queue out`** the weight assigned to queue 4 with the command **`wrr-queue bandwidth`** does not play any role for the WRR scheduler and you may set it to zero when calculating bandwidth share.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

For example, consider the following situation. The interface rate is 10Mbps and WRR weights are 10 20 30 40 for queues from 1 to 4. The interface uses a priority queue. Imagine that voice traffic (priority queue) rate is 1Mbps (that's way too much, but it's an imaginary situation). In this case, the minimum bandwidth guarantees are:

Queue 1: 9Mbps*10/(10+20+30+0)=1/6*9Mbs=1.5 Mbps.
Queue 2: 9Mbps*20/(10+20+30+0)=3 Mbps.
Queue 3: 9Mbps*30/(10+20+30+0)=4.5 Mbps.

In case if priority traffic is not present, the extra 1Mbps could be shared in the same proportions between the remaining queues.

The last question is mapping packets to queues. In the 3550 model you can only do that based on the CoS values. However, as we remember, the 3550 assigns internal DSCP value for all classified packets. In order to come up with CoS value for a packet, the switch consults the DSCP to CoS mapping table. It is important to note, that if the switch policed down the internal DSCP, it uses the new, policed value for mapping. The command to map a CoS to Queue-ID is interface level command **`wrr-queue cos-map <q-id> <CoS List>`**. You may find out the current mappings using the command **`show mls qos interface queueing`**.

In our case, we use the default DSCP to CoS mapping table, which results in the following mappings:

DSCP EF -> CoS 5
DSCP AF41 -> CoS 4
DSCP CS3 -> CoS 3
DSCP AF21 -> CoS 2
DSCP 0 - > CoS 0

We configure the fourth queue as the priority queue and map CoS 5 to it. Next we map CoS 4 to queue 3 and both CoS 2 and CoS 3 to queue 2. Finally, CoS 0 is mapped to Queue 1 by default. The bandwidth weight values are easy to find from the bandwidth percentage: 30% for video traffic, 20% for interactive and voice signaling and 50% for best-effort traffic. Assign any WRR weight value to queue 4 since it is not used when the priority queue is engaged.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

We also need to deal with the control plane traffic, which usually bears CoS values of 6 or 7. For our scenario we'll just map them to the queue used by video traffic. Use the following show commands to display the new mappings and WRR weights:

```
Rack1SW4#show mls qos interface fastEthernet 0/4 queueing
FastEthernet0/4
Egress expedite queue: ena
wrr bandwidth weights:
qid-weights
 1 - 50
 2 - 20
 3 - 30
 4 - 1    when expedite queue is disabled
Cos-queue map:
cos-qid
 0 - 1
 1 - 1
 2 - 2
 3 - 2
 4 - 3
 5 - 4
 6 - 3
 7 - 3
```

Now let's run a stress test for WRR. We will configure the connection to R4 for a speed of 10Mbps and send a barrage of ICMP packets from R6 and R1 to saturate the link. One of the flows will be marked with DSCP AF21 and the other will use the default DSCP 0. Per the WRR algorithm, the bandwidth should be shared in proportions of 50:20 between DSCP 0 and DSCP AF21 (ToS byte 72) traffic. The resulting bandwidth values should be

50/(20+50)*10Mbps = 7.1 Mbps for DSCP 0 traffic
20/(20+50)*10Mbps = 2.9 Mbps for DSCP AF21 traffic.

SW4 should trust DSCP marking for packets coming from other switches. R4 should block ICMP packets from getting to the CPU and meter the input rate for DSCP AF21 packets.

```
SW4:
interface range FastEthernet 0/13 - 21
 mls qos trust dscp
!
R4:
class-map DSCP_AF21
 match dscp af21
!
policy-map METER
 class DSCP_AF21
!
ip access-list extended FILTER
 deny icmp any any
 permit ip any any
!
interface FastEthernet 0/1
 ip access-group FILTER in
 service-policy input METER
 load-interval 30
```

```
Rack1R6#ping
Protocol [ip]:
Target IP address: 155.1.146.4
Repeat count [5]: 100000000
Datagram size [100]: 1500
Timeout in seconds [2]: 0
Extended commands [n]: y
Source address or interface:
Type of service [0]: 72
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 100000000, 1500-byte ICMP Echos to 155.1.146.4, timeout is 0
seconds:
......................................................................
......................................................................
......................................................................
```

```
Rack1R1#ping 155.1.146.4 repeat 1000000 size 1500 timeout 0

Type escape sequence to abort.
Sending 1000000, 1500-byte ICMP Echos to 155.1.146.4, timeout is 0
seconds:
......................................................................
......................................................................
......................................................................
......................................................................
.....................................................................
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Verify the statistics accumulated at R4 and note how close it is to the values calculated above:

```
Rack1R4#show policy-map interface fastEthernet 0/1
 FastEthernet0/1

  Service-policy input: METER

    Class-map: DSCP_AF21 (match-all)
      41104 packets, 58211273 bytes
      30 second offered rate 2730000 bps
      Match:  dscp af21 (18)

    Class-map: class-default (match-any)
      228668 packets, 346115656 bytes
      30 second offered rate 7104000 bps, drop rate 0 bps
      Match: any
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.74 Catalyst 3550 Regular Queues Tuning

- Configure SW4 to set the global min-reserve levels 7 and 8 to 170 (max) and 10 (min) buffers respectively.
- Under the interface connected to R4 map global level 7 to queue 1 and global level 8 to queue 4.

Configuration

```
SW4:
mls qos min-reserve 7 170
mls qos min-reserve 8 10
!
interface FastEthernet 0/4
 wrr-queue min-reserve 1 7
 wrr-queue min-reserve 4 8
```

Verification

Note

As mentioned earlier, the 3550 makes a clear distinction between FastEthernet and GigabitEthernet ports. The buffer space allocation for regular FastEthernet ports differs significantly from the buffer allocation scheme for GigabitEthernet Port.

Every queue on a regular interface has its own buffer limit. You cannot change it for every specific queue on any interface. Rather, there are eight global variables knows as “minimal reserve levels”. Each of these values is the number of packet buffers. You can map the global constants to a queue-id on each interface.

By default, queues 1 through 4 on every interface use global minimal reserve levels 1 through 4 as well. Initially, all eight levels have value of 100 buffers. The maximum level size is 170 buffers and the minimum size is 10. The buffer tuning commands for regular interfaces are as follows:

Global command defines values any of 8 levels

```
mls qos min-reserve <level> <size>
```

Interface-level command:

```
wrr-queue min-reserve <q-id> <level>
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Note that you cannot change the drop policy on regular interface. It's fixed to tail drop, with no advanced features. For this task, we should tune the levels that are not used by default, e.g. levels 7 and 8. Use the following show command to verify your settings:

```
Rack1SW4#show mls qos interface fastEthernet 0/4 buffers
FastEthernet0/4
Minimum reserve buffer size:
 100 100 100 100 100 100 170 10
Minimum reserve buffer level select:
 7 2 3 8
```

Here you can see the minimum reserve levels and their assignment for the given interface.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.75 Catalyst 3550 Gigabit Interface Queues Tuning

- Configure the first GigbitEthernet interface on SW1 as follows:
    - Set thresholds to 50 and 100 for all four queues
    - Map DSCP values 46, 48, and 56 to threshold 2
    - Set WRR queue weights to 20, 20, 20 and 40 for queues 1 through 4
- Enable WRED on the second GigabitEthernet interface of SW1 and configure as following:
    - Set WRED minimum thresholds to 50 and 80 for all queues
    - Map DSCP values 46, 48 and 56 to WRED threshold 2
    - Set WRR queue weights to 20, 20, 20 and 40 for queues 1 through 4

Configuration

```
SW4:
!
! Use WTD on the first interface
!
interface GigabitEthernet0/1
 wrr-queue threshold 1 50 100
 wrr-queue threshold 2 50 100
 wrr-queue threshold 3 50 100
 wrr-queue threshold 4 50 100
 wrr-queue dscp-map 2 46 48 56
 wrr-queue queue-limit 20 20 20 40
!
! Use WRED on the second interface
!
interface GigabitEthernet0/2
 wrr-queue random-detect max-threshold 1 50 80
 wrr-queue random-detect max-threshold 2 50 80
 wrr-queue random-detect max-threshold 3 50 80
 wrr-queue random-detect max-threshold 4 50 80
 wrr-queue dscp-map 2 46 48 56
 wrr-queue queue-limit 20 20 20 40
```

Copyright © 2009 Internetwork Expert

Verification

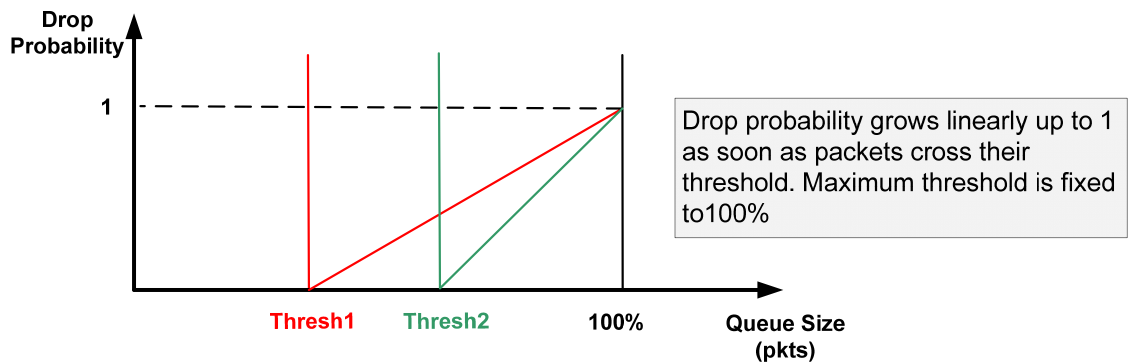
Note

GigabitEthernet ports support two queue management techniques. The first one is similar to the 3560 WTD (Weighted Tail Drop). Each of the four queues has two thresholds, expressed in percents. By default, both thresholds are set to 100%, simulating tail drop. You can tune the thresholds per-queue and map internal DSCP values to any of the thresholds. Thus, packets with different internal DSCP values have different chances of being dropped in the output queue. By default all DSCP values map to the first threshold.

The interface-level commands to set WTD thresholds are as follows:

```
 wrr-queue threshold <q-id> <thresh1> <thresh2>
wrr-queue dscp-map <thresh> <DSCP List>
```

The second drop mode for any queue on GigabitEthernet interface is WRED. You can enable WRED per-queue and set two thresholds where the system starts dropping packets. Unlike the classic WRED found in routers, this WRED implementation does not allow you to set the maximum threshold. It is fixed to 100%. However, you can map DSCP values to minimum (starting) thresholds, and thus provide differentiated drop behavior.



On the diagram, you see two different thresholds and the graph of dropping probability for each starting value. There is no Marking Probability like the one you see in IOS WRED, not there is average queue size weighting parameter. By default all DSCP values maps to threshold 1.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

The commands to configure WRED in the 3550 are as follows:

```
 wrr-queue random-detect max-threshold <q-id> <thresh1>
<thresh2>
 wrr-queue dscp-map <thresh> <DSCP List>.
```

You can see that the threshold mapping is shared by both queue modes. The default queue mode for gigabit interface is WTD with both thresholds set to 100%.

Lasty, the buffer space allocation for the Gigabit ports is also different. Each port has common buffer space shared between all queues. You can not specify the number of buffers available to each specific queue but you can set the weighted part of shared space allocated to each queue. The command to share the buffers is:

```
wrr-queue queue-limit <w1> <w2> <w3> <w4>
```

Queue “j” will get the share of buffer space equals to

s(j)=w(j)/(w(1)+...w(4).

Try to configure the weights so that they sum to 100, so that you can interpret weights as percents of buffer pool size.
Use the following show command to verify your new settings for both ports.

For the first port, notice how the DSCP values map to the thresholds and new threshold values.

```
Rack1SW4#show mls qos interface gigabitEthernet 0/1 queueing
GigabitEthernet0/1
Egress expedite queue: dis
wrr bandwidth weights:
qid-weights
 1 - 25
 2 - 25
 3 - 25
 4 - 25
Dscp-threshold map:
     d1 :  d2 0  1  2  3  4  5  6  7  8  9
     ---------------------------------------
      0 :    01 01 01 01 01 01 01 01 01 01
      1 :    01 01 01 01 01 01 01 01 01 01
      2 :    01 01 01 01 01 01 01 01 01 01
      3 :    01 01 01 01 01 01 01 01 01 01
      4 :    01 01 01 01 01 01 02 01 02 01
      5 :    01 01 01 01 01 01 02 01 01 01
      6 :    01 01 01 01
Cos-queue map:
cos-qid
 0 - 1
 1 - 1
 2 - 2
 3 - 2
 4 - 3
 5 - 3
 6 - 4
 7 - 4

Rack1SW4#show mls qos interface gigabitEthernet 0/1 buffers
GigabitEthernet0/1
Notify Q depth:
qid-size
 1 - 20
 2 - 20
 3 - 20
 4 - 40
qid WRED thresh1 thresh2
1   dis  50      100
2   dis  50      100
3   dis  50      100
4   dis  50      100
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

For the second port, notice that WRED is the drop mode for all four queues.

```
Rack1SW4#show mls qos interface gigabitEthernet 0/2 queueing
GigabitEthernet0/2
Egress expedite queue: dis
wrr bandwidth weights:
qid-weights
 1 - 25
 2 - 25
 3 - 25
 4 - 25
Dscp-threshold map:
     d1 :  d2 0  1  2  3  4  5  6  7  8  9
     ---------------------------------------
      0 :    01 01 01 01 01 01 01 01 01 01
      1 :    01 01 01 01 01 01 01 01 01 01
      2 :    01 01 01 01 01 01 01 01 01 01
      3 :    01 01 01 01 01 01 01 01 01 01
      4 :    01 01 01 01 01 01 02 01 02 01
      5 :    01 01 01 01 01 01 02 01 01 01
      6 :    01 01 01 01
Cos-queue map:
cos-qid
 0 - 1
 1 - 1
 2 - 2
 3 - 2
 4 - 3
 5 - 3
 6 - 4
 7 - 4

Rack1SW4#show mls qos interface gigabitEthernet 0/2 buffers
GigabitEthernet0/2
Notify Q depth:
qid-size
 1 - 20
 2 - 20
 3 - 20
 4 - 40
qid WRED thresh1 thresh2
1   ena  50      80
2   ena  50      80
3   ena  50      80
4   ena  50      80
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.76 Catalyst 3550 Egress Policing

- Enable the expedite queue on SW4's connection to R4.
- Ensure only CoS 5 maps to this queue.
- Limit the rate of the expedite queue to 128Kbps with the burst size of 8000 bytes.
- Limit the rate of traffic marked with CS0 to 384Kbps using a burst size of 32000 bytes for metering.

Configuration

```
SW4:
mls qos
!
class-map DSCP_EF
 match ip dscp ef
!
class-map DSCP_CS0
 match ip dscp 0
!
policy-map EGRESS_POLICING
 class DSCP_EF
  police 128000 16000
 class DSCP_CS0
  police 384000 32000
!
interface FastEthernet 0/4
 priority-queue out
!
! Ensure that CoS 6 and 7 map to non-priority queue
!
 wrr-queue cos-map 3 6 7
 service-policy output EGRESS_POLICING
```

Verification

Note

The egress policing feature is unique to the 3550. You may configure up to eight egress policers on any port. However, the limitation is that you can only match DSCP values as the policer classification criterion. The DSCP values are the internal DSCP and thus apply both to IP and non-IP traffic.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

For verification, configure SW4 to trust DSCP values on all trunk ports and send two ping floods from R6 towards R4.

```
SW4:
interface range FastEthernet 0/13 - 21
 mls qos trust dscp

Rack1R6#ping
Protocol [ip]:
Target IP address: 155.1.146.4
Repeat count [5]: 100000
Datagram size [100]: 1500
Timeout in seconds [2]: 1
Extended commands [n]: y
Source address or interface:
Type of service [0]: 184
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 100000, 1500-byte ICMP Echos to 155.1.146.4, timeout is 1
seconds:
!!!!!!!!!!!!.!!!!!!!!!!!.!!!!!!!!!!!.!!!!!!!!!!!.!!!!!!!!!!!.!!!!!!!!!
!!!.!!!!!!!!!!!.!!!!!!!!!!!.!!!!!!!!!!!.!!!!!!!!!!!.!!!!!!!!!!!!.!!!!!
!!!!!!.!!!!!!!!!!!.!!!!!!!!!!!.!!!!!!!!!!!.!!!!!!!!!!!!.!!!!!!!!!!!.!!
Success rate is 91 percent (639/696), round-trip min/avg/max = 4/4/21
ms
```

Note the count of “!” signs in the selected sample which is 11. Multiply this number by the packet size, which is 1514 on Ethernet.

Bits = 11*1514*8=133232. It took 11*4ms to send those packets, plus one second of pause waiting for the dropped packet. Effectively, we send all those bits in the time interval of approximately 1 second. This is close to the desired value of 128Kilobits per second.

```
Rack1R6#ping 155.1.146.4 repeat 10000 size 1500 timeout 1
Type escape sequence to abort.
Sending 10000, 1500-byte ICMP Echos to 155.1.146.4, timeout is 1
seconds:
!!!!!!!!!!!!!!!!!!!!!!!!!!.!!!!!!!!!!!!!!!!!!!!!!!!!!.!!!!!!!!!!!!!!!!
!!!!!!!!!!.!!!!!!!!!!!!!!!!!!!!!!!!!!.!!!!!!!!!!!!!!!!!!!!!!!!!!.!!!!!
!!!!!!!!!!!!!!!!!!!!!.!!!!!!!!!!!!!!!!!!!!!!!!!!.!!!!!!!!!!!!!!!!!!!!!
!!!!!.!!!!!!!!!!!!!!!!!!!!!!!!!!.!!!!!!!!!!!!!!!!!!!!!!!!!!.!!!!!!!!!!
!!!!!!!!!!!!!!!!.!!!!!!!!!!!!!!!!!!!!!!!!!!.!!!!!!!!!!!!!!!!!!!!!!!!!!
.
Success rate is 96 percent (338/351), round-trip min/avg/max = 4/4/12
ms
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Take the sample burst of 26 packets. The size of 26 packets in bits is

26*1514*8=314912.

This is far below the target value of 384 Kbps. This is probably due to the fact that now the token bucket refills much faster, and our procedure is not accurate enough to meter precisely the average rate.

Now verify that policing applies to IPX traffic as well. In this case, we source IPX packets that are going to have CoS value of 0 in SW4. This value maps to DSCP 0.

```
R4:
ipx routing
!
interface FastEthernet 0/1
 ipx network 146 encapsulation snap

R6:
ipx routing
!
interface FastEthernet 0/0.146
 ipx network 146 encapsulation snap

Rack1R4#show ipx interface fastEthernet 0/1
FastEthernet0/1 is up, line protocol is up
  IPX address is 146.0011.2031.0901, SNAP [up]

Rack1R6#ping
Protocol [ip]: ipx
Target IPX address: 146.0011.2031.0901
Repeat count [5]: 100000
Datagram size [100]: 1000
Timeout in seconds [2]: 1
Verbose [n]:
Type escape sequence to abort.
Sending 100000, 1000-byte IPX Novell Echoes to 146.0011.2031.0901,
timeout is 1 seconds:
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!.!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!.!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!.!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!.!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!.!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!.
Success rate is 97 percent (222/228), round-trip min/avg/max = 1/3/5 ms
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.77 Catalyst 3560 SRR Shared Mode

- Configure SW1 to set the speed of the port connected to R1 to 10 Mbps, and limit the bandwidth to 20%.
- Preserve the default DSCP to Queue ID mapping.
- Set all SRR shaped weights to zero.
- Set the SRR shared weights to 1, 10, 20, and 30 for queues 1 through 4.
- Enable expedite queue on the interface.

Configuration

```
SW1:
mls qos
!
! Those mappings are the default but we still present
! them for the sake of clarity
!
mls qos srr-queue output dscp-map queue 1 46
mls qos srr-queue output dscp-map queue 2 0
mls qos srr-queue output dscp-map queue 3 18
mls qos srr-queue output dscp-map queue 4 34
!
! Apply SRR weights
!
interface FastEthernet 0/1
 speed 10
 srr-queue bandwidth limit 20
 srr-queue bandwidth shape  0  0  0  0
 srr-queue bandwidth share 1 10 20 30
 priority-queue out
```

Verification

✎ **Note**

Like the 3550 model, the 3560 has four egress queues per interface. However, the 3560 does not make any distinction between GigibitEthernet or FastEthernet ports. All of them have uniform scheduler, called SRR (which we mentioned earlier). Depending on the modes of operation, SRR stands for either Shared Round Robin (we discussed the use of SRR for ingress queues) or Shaped Round Robin.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

You can configure each of the four egress queues in either Shaped or Shared mode. Here is how the scheduler works with each of them:

1) All queues configured in shaped mode have an absolute weight assigned. The interface-level command to assign the weights is

```
srr-queue bandwidth shape <w1> <w2> <w3> <w4>
```

If some of the weights are zero, then the respective queues operate in *shared* mode. However, for non-zero weights the scheduler places the respective queue in *shaped* mode.

In shaped mode, the system limits the queue sending rate to **1/weight*interface-speed**. The speed here is the actual physical rate set with the **`speed`** interface-level command. At the same time, the SRR scheduler *guarantees* this rate to the queue. For example if the interface speed is 100Mbps and the shaped weight is 20, then the respective queue is shaped to 5Mbps. The switch meters output traffic and delays the exceeding traffic to conform to the configure rate. However, the details of the shaping procedure is not published.

2) All queues *not* configured in *shaped* mode operate in *shared* mode. The scheduler obtains weights for shared mode queues from the command

```
srr-queue bandwidth share <w1> <w2> <w3> <w4>
```

These weights are always non-zero. If it happens so that the corresponding *shaped* weight is non-zero also, then the scheduler *ignores* the shared weight and considers it zero in bandwidth share computations.

All shared queues share the bandwidth *remaining* after the shaped queues. In fact, the SRR scheduler “prefers” shaped queues to shared queues and ensures the former have the bandwidth they need up to their limit. However, you need to make sure that absolute bandwidth values assigned to shaped queues do not *exceed* the interface rate (or speed limit, which we discuss later). Otherwise, unexpected behavior may result.

Shared queues share the remaining bandwidth *proportional* to their configured weights. If there are no shaped queues on the interface, four queues simply divide the interface bandwidth in proportions:

<w1>:<w2>:<w3>:<w4>

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

That means bandwidth shares are

s(i)=w(i)/(w(1)+...w(4)) for queue number “i”.

If any of the queues operate in shaped mode, consider the respective *w(i)* to be zero, irrespective of the actual weight setting.

Consider the following example. A one hundred megabit interface has queues 1 and 2 in shaped mode and queues 3 and 4 in shared mode. Shaped weights are 10 and 20 (Qeueues 1 and 2), and shared weights are 30 and 70 (Queues 3 and 4):

Example:

```
interface FastEthernet 0/1
 speed 100
 srr-queue bandwidth shape 10 20 0 0
 srr-queue bandwidth share  1 1 30 70
```

In case of congestions the scheduler divides bandwidth as follows:

Queue 1: 10 Mbps.
Queue 2: 5 Mbps.
Queue 3: 30/(30+70)*(100-10-5)=85*0,3=25,5 Mbps.
Queue 4: 70/(30+70)*(100-10-5)=85,*0,7=59,5 Mbps.

Another feature supported by SRR is *priority* (or expedite) queue. Just as with the 3550, you can enable one queue for expedite services. However, with the 3560 this is queue number 1, not queue number 4. When queue 1 works as priority, it has unlimited bandwidth and may starve any other queue (either shared or shaped). In addition, shared and shaped weights set for the priority queue has no effect and the SRR scheduler does not take them in account.

A feature unique to the 3560 is limiting the egress port sending rate. You can use special interface level shaper to restrict the port sending rate to a value between 10% and 100% of its physical rate. The interface-level command is

```
srr-queue bandwidth limit <percent>
```

Note that the <percent> value ranges from 10% to 100%. Thus, the lowest sending rate for an interface is 1Mbps, achieved with physical speed 10Mbps and 10% limit. Note that shaped queue weights still apply to the physical speed, not the bandwidth limit, when calculating shaped queue rates.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

The default values for SRR shaped and shared weights are:

Queue 1: Shaped 25, Shared 25
Queue 2: Shaped 0, Shared 25
Queue 3: Shaped 0, Shared 25
Queue 3: Shaped 0, Shared 25

Note that Queue 1 uses shaped mode, while all other queues share the remaining bandwidth in equal proportions.

The next thing is mapping of DSCP and CoS values to Queue IDs. Recall that in the 3550 you can only map CoS values (DSCPs are mapped implicitly) and you have flexibility to do that at the interface level. However, in the 3560 all mappings are global and apply to all interfaces at the same time. Usually that makes sense in enterprise switches, where the users in the same QoS domain share one device.

```
mls qos srr-queue output dscp-map queue <q-id> <DSCP List>
mls qos srr-queue output cos-map queue <q-id> <CoS List>
```

Note that you can map DSCP and CoS values separately. The switch chooses the value you trusted or set at the classification stage to select the proper table. The default mappings exist and you can check them using the **show mls qos map** commands.

In this particular task, we use the default DSCP to Queue ID mappings, where the default DSCP maps to Queue 2, AF21 maps to Queue 3, and AF41 maps to Queue 4. Note that DSCP EF by default maps to Queue 1. In the example below, we enable queue 1 as a priority queue and configure shared weights to attain the required proportions.

We set the port speed to 10Mbps and bandwidth limit to 20% in order to achieve 2Mbps output rate. For verification purpose, we configure the following devices as traffic sources: R4, R6, SW3, and SW4. The switch connected to R1 (SW1) should trust DSCP on all it trunk ports.

R4 will send DSCP EF traffic (TOS 184);
R6 will source DSCP AF21 traffic (TOS 72).
R3 and R5 will source DSCP AF41 (TOS 136) and DSCP 0 packets respectively. Note that you need to reconfigure R3 and R5 temporarily to accomplish this.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

After that, we limit all sources sending rate to 1Mbps using traffic shaping on the routers. However, R4's sending rate will be limited down to 256Kbps, since it sources priority traffic (simulates a VoIP packet flow). This configuration will definitely oversubscribe the interface limited to 2Mbps.

Note that MLS QoS is disabled on all switches with except to SW1.

```
R3:
interface FastEthernet 0/1
 ip address 155.1.146.3 255.255.255.0
 no shutdown
 traffic-shape rate 1000000

R4:
interface FastEthernet 0/1
 traffic-shape rate 256000

R5:
interface FastEthernet 0/1
 ip address 155.1.146.5 255.255.255.0
 traffic-shape rate 1000000

R6:
interface FastEthernet 0/1.146
 traffic-shape rate 1000000

SW1:
interface range FastEthernet 0/13 - 21
 mls qos trust dscp

SW2:
no mls qos

SW3:
no mls qos
!
interface FastEthernet 0/3
 switchport access vlan 146
!
interface FastEthernet 0/5
 switchport access vlan 146

SW4:
no mls qos
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
Rack1R4#ping
Protocol [ip]:
Target IP address: 155.1.146.1
Repeat count [5]: 100000000
Datagram size [100]: 1500
Timeout in seconds [2]: 0
Extended commands [n]: y
Source address or interface:
Type of service [0]: 184
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 100000000, 1500-byte ICMP Echos to 155.1.146.1, timeout is 0
seconds:
......................................................................
......................................................................
......................................................................

Rack1R6#ping
Protocol [ip]:
Target IP address: 155.1.146.1
Repeat count [5]: 100000000
Datagram size [100]: 1500
Timeout in seconds [2]: 0
Extended commands [n]: y
Source address or interface:
Type of service [0]: 72
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 100000000, 1500-byte ICMP Echos to 155.1.146.1, timeout is 0
seconds:
......................................................................
......................................................................
......................................................................

Rack1R3#ping
Protocol [ip]:
Target IP address: 155.1.146.1
Repeat count [5]: 100000000
Datagram size [100]: 1500
Timeout in seconds [2]: 0
Extended commands [n]: y
Source address or interface:
Type of service [0]: 136
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
Sending 100000000, 1500-byte ICMP Echos to 155.1.146.1, timeout is 0
seconds:
......................................................................
......................................................................
......................................................................

Rack1R5#ping 155.1.146.1 repeat 10000000 size 1500 timeout 0

Type escape sequence to abort.
Sending 10000000, 1500-byte ICMP Echos to 155.1.146.1, timeout is 0
seconds:
......................................................................
......................................................................
......................................................................
......................................................................
```

Configure R1 for detection of input traffic marked with the DSCP values mentioned in the task.

```
R1:
class-map DSCP_EF
 match dscp ef
!
class-map DSCP_AF21
 match dscp af21
!
class-map DSCP_0
 match dscp 0
!
class-map DSCP_AF41
 match dscp af41
!
policy-map DETECT
 class DSCP_EF
 class DSCP_0
 class DSCP_AF21
 class DSCP_AF41
!
interface FastEthernet 0/0
 load-interval 30
 service-policy input DETECT
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Now observe the results on R1:

```
Rack1R1#show policy-map interface fastEthernet 0/0
 FastEthernet0/0

  Service-policy input: DETECT

    Class-map: DSCP_EF (match-all)
      12065 packets, 18266410 bytes
      30 second offered rate 255000 bps
      Match:  dscp ef (46)

    Class-map: DSCP_0 (match-all)
      22841 packets, 34581274 bytes
      30 second offered rate 284000 bps
      Match:  dscp default (0)

    Class-map: DSCP_AF21 (match-all)
      45835 packets, 69394190 bytes
      30 second offered rate 572000 bps
      Match:  dscp af21 (18)

    Class-map: DSCP_AF41 (match-all)
      68628 packets, 103902792 bytes
      30 second offered rate 860000 bps
      Match:  dscp af41 (34)

    Class-map: class-default (match-any)
      102 packets, 14972 bytes
      30 second offered rate 0 bps, drop rate 0 bps
      Match: any
```

Note that the proportion 1:2:3 holds true for DSCP 0, AF21, and AF41 traffic. The total aggregate input bandwidth is close to 2 Mbps.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.78 Catalyst 3560 SRR Shaped Mode

- Change the shaped weight for queue 4 to 1Mbps.
- Set the shared weights to 1, 10, 20, and 1 for queues 1 through 4 respectively.

Configuration

```
SW1:
!
! Apply SRR weights
!
interface FastEthernet 0/1
 speed 10
 srr-queue bandwidth limit 20
 srr-queue bandwidth shape  0  0  0  10
 srr-queue bandwidth share 1 10 20 1
 priority-queue out
```

Verification

Note

This task requires limiting one of the queues to 1Mbps. Since the interface rate is 10Mbps, the shaped weight would be 10. The other two classes get the weight values of 20 and 10. Those classes will share the bandwidth remaining after the priority queue and shaped queue in proportions specified.

For verification purpose, we configure the following devices as traffic sources: R4, R6, SW3, and SW4. The switch connected to R1 (SW1) should trust DSCP on all it trunk ports.

R4 will send DSCP EF traffic (TOS 184);
R6 will source DSCP AF21 traffic (TOS 72).
R3 and R5 will source DSCP AF41 (TOS 136) and DSCP 0 packets respectively. Note that you need to reconfigure R3 and R5 temporarily to accomplish this.

After that, we limit all sources sending rate to 1Mbps using traffic shaping in routers. However, R4 sending rate will be limited down to 256Kbs, since it sources priority traffic (simulates a VoIP packet flow). This configuration will definitely oversubscribe the interface limited to 2Mbps.

Note that MLS QoS is disabled on all switches with except to SW1.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
R3:
interface FastEthernet 0/1
 ip address 155.1.146.3 255.255.255.0
 no shutdown
 traffic-shape rate 1000000

R4:
interface FastEthernet 0/1
 traffic-shape rate 256000

R5:
interface FastEthernet 0/1
 ip address 155.1.146.5 255.255.255.0
 traffic-shape rate 1000000

R6:
interface FastEthernet 0/1.146
 traffic-shape rate 1000000

SW1:
interface range FastEthernet 0/13 - 21
 mls qos trust dscp

SW2:
no mls qos

SW3:
no mls qos
!
interface FastEthernet 0/3
 switchport access vlan 146
!
interface FastEthernet 0/5
 switchport access vlan 146

SW4:
no mls qos
```

```
Rack1R4#ping
Protocol [ip]:
Target IP address: 155.1.146.1
Repeat count [5]: 100000000
Datagram size [100]: 1500
Timeout in seconds [2]: 0
Extended commands [n]: y
Source address or interface:
Type of service [0]: 184
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 100000000, 1500-byte ICMP Echos to 155.1.146.1, timeout is 0
seconds:
......................................................................
......................................................................
```

Copyright © 2009 Internetwork Expert

```
Rack1R6#ping
Protocol [ip]:
Target IP address: 155.1.146.1
Repeat count [5]: 100000000
Datagram size [100]: 1500
Timeout in seconds [2]: 0
Extended commands [n]: y
Source address or interface:
Type of service [0]: 72
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 100000000, 1500-byte ICMP Echos to 155.1.146.1, timeout is 0
seconds:
......................................................................
......................................................................
......................................................................

Rack1R3#ping
Protocol [ip]:
Target IP address: 155.1.146.1
Repeat count [5]: 100000000
Datagram size [100]: 1500
Timeout in seconds [2]: 0
Extended commands [n]: y
Source address or interface:
Type of service [0]: 136
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 100000000, 1500-byte ICMP Echos to 155.1.146.1, timeout is 0
seconds:
......................................................................
......................................................................
......................................................................

Rack1R5#ping 155.1.146.1 repeat 10000000 size 1500 timeout 0

Type escape sequence to abort.
Sending 10000000, 1500-byte ICMP Echos to 155.1.146.1, timeout is 0
seconds:
......................................................................
......................................................................
......................................................................
......................................................................
```

Configure R1 for detection of input traffic marked with the DSCP values mentioned in the task.

```
R1:
class-map DSCP_EF
 match dscp ef
!
class-map DSCP_AF21
 match dscp af21
!
class-map DSCP_0
 match dscp 0
!
class-map DSCP_AF41
 match dscp af41
!
policy-map DETECT
 class DSCP_EF
 class DSCP_0
 class DSCP_AF21
 class DSCP_AF41
!
interface FastEthernet 0/0
 load-interval 30
 service-policy input DETECT
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Now observe the results on R1:

```
Rack1R1#show policy-map interface fastEthernet 0/0
 FastEthernet0/0

  Service-policy input: DETECT

    Class-map: DSCP_EF (match-all)
      6146 packets, 9305044 bytes
      30 second offered rate 255000 bps
      Match:  dscp ef (46)

    Class-map: DSCP_0 (match-all)
      9794 packets, 14828116 bytes
      30 second offered rate 408000 bps
      Match:  dscp default (0)

    Class-map: DSCP_AF21 (match-all)
      19671 packets, 29781894 bytes
      30 second offered rate 817000 bps
      Match:  dscp af21 (18)

    Class-map: DSCP_AF41 (match-all)
      11792 packets, 17853088 bytes
      30 second offered rate 491000 bps
      Match:  dscp af41 (34)

    Class-map: class-default (match-any)
      73 packets, 16098 bytes
      30 second offered rate 0 bps, drop rate 0 bps
      Match: any
```

Note that bandwidth proportions of DSCP 0 and DSCP AF21 classes is 1:2. The bandwidth guaranteed to AF41 class is close to target 500Kbps.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.79 Catalyst 3560 Egress Queues Tuning

- Configure SW1 to modify buffer sharing for queues on the interface connected to R1 only.
- Ensure that queue 2 has 70% of interface buffer pools and remaining queues share buffers in equal proportions.
- Reserve only 10% of allocated buffers for queue 2; the remaining queues should reserve all allocated buffers.
- Ensure that traffic marked with AF11, AF21, AF41, and CS1 maps to the second drop threshold in their respective queues.
- Set the first drop threshold to 200% and the second drop threshold to 150% for all queues.
- Ensure that the switch drops voice packets marked with DSCP EF only if the respective queue is exhausted.
- Set the maximum threshold to 250% for all queues

Configuration

```
SW1:
!
! Map DSCP EF to queue-full threshold
! Map other DSCPs to the lower threshold (2)
!
mls qos srr-queue output dscp-map threshold 3 46
mls qos srr-queue output dscp-map threshold 2 8 10 18 34
!
! Configure queue threshold. First two are the drop threshold
! The other two specify how much to reserve and how much a queue
! can borrow from a common pool.
!
mls qos queue-set output 2 threshold 1 200 150 100 250
mls qos queue-set output 2 threshold 2 200 150 10 250
mls qos queue-set output 2 threshold 3 200 150 100 250
mls qos queue-set output 2 threshold 4 200 150 100 250
!
! Partition reserved pool between queues in proportions 1:7:1:1
!
mls qos queue-set output 2 buffers 10 70 10 10
!
! Apply queue-set to the interface. By default, all interfaces
! use queue-set 1 so we modify only the port connected to R1.
!
interface FastEthernet 0/1
 queue-set 2
```

Verification

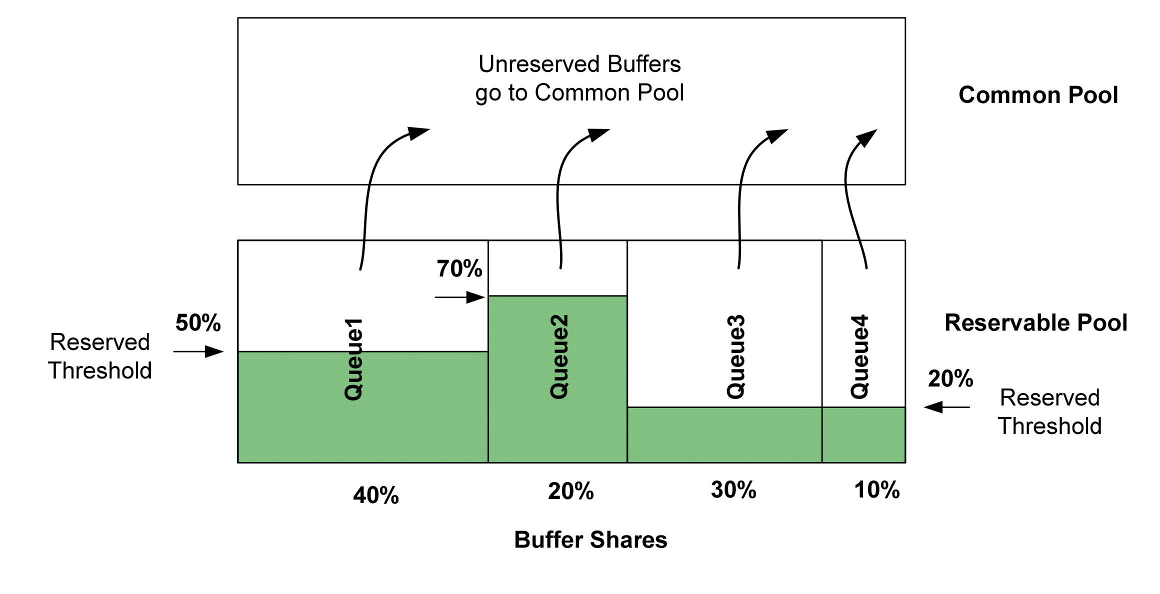
✎ Note

The queue buffer partitioning scheme in the 3560 is somewhat complicated. First of all, you cannot specify buffer allocation per interface. There exist two configurable global templates, which you can apply to interfaces individually. Those templates are called “queue-sets” and by default, all interfaces use queue-set 1. Within a queue-set you can define buffer allocation between interface queues and drop thresholds for each queue. However, remember that a queue set is merely a template that has effect only when applied to physical interface.

Physical interfaces have their own reservable buffers pool. You can partition this pool between four egress queues using the template command:

```
mls qos queue-set <qset-id> output <q-id> <w1> <w2> <w3>
<w4>
```

Where *<w1>...<w4>* are the weights in percents that specify how many buffers in the pool to allocate to the respective queue identified by number *<q-id>*. The sum of all weights should not exceed 100%. On the figure below, you can see those weights as buffer shares on the bottom of the reservable pool. There is also special common pool, shared by all interaces, which we discuss later.



For example if a reservalbe pool is 1000 buffers and you partition it in proportions 40%, 20%, 30%, and 10%, then queues 1 through 4 would get 400, 200, 300 and 100 buffers respectively.

After you allocated shares of reservable space to queues, you can set up queue thresholds. There are two types of threshold per queue, set by a single command.

```
mls qos queue-set output <qset-id> <threshold-id> <q-id>
<drop-threshold1> <drop-threshold2> <reserved-threshold>
<maximum-threshold>
```

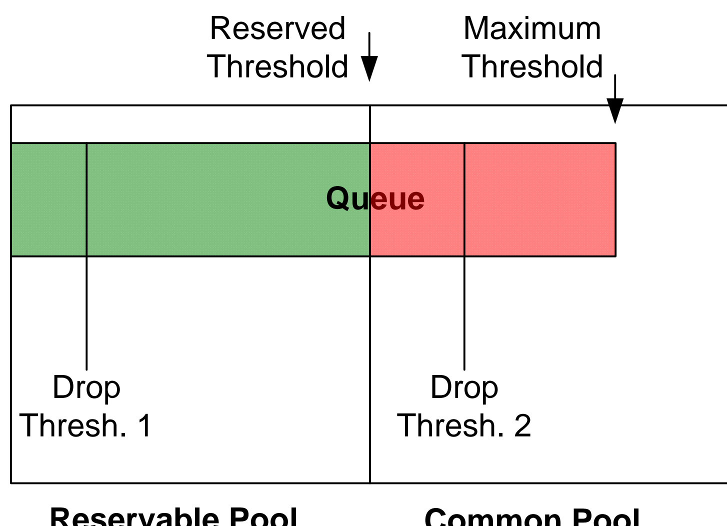
The first type defines the “drop” thresholds and specifies queue depth values where the scheduler drops certain packet types. There are two drop thresholds per queue and you can map DSCP/CoS values to drop thresholds.

The second threshold type defines queue size itself. As we know, each queue gets a part of reservable buffer space. This part may be even zero, as will see later. For every queue, you can define reserved threshold (in percents) and maximum threshold (in percents). The system uses both thresholds to “scale” the number of buffers allocated to a queue.

Let’s look at the reserved threshold first. This one defines the amount of buffer space the system actually reserves for the queue. For example, if the reservable pool is 1000 buffers, and you allocate 40% of the pool to queue 1, then the queue has 400 buffers as its base. Now if if you set the reservable threshold to 50%, then the system guarantees only 200 buffers to the queue. The system takes away and *shares* the remaining buffers in a common pool between other queues. Thus, the reservable threshold may vary from 0% to 100%, with the default value being 50%. Look at the figure above once again for an illustration.

The maximum threshold is also in percent. Like the reservable threshold, it applies to the number of buffers allocated to a queue during the partitioning process. This value ranges from 1% to 400% and specifies the maximum queue size in buffers. You multiply the number of buffers allocated to a queue by this value to get maximum queue size. Note that queue may grow *beyond* its reserved limit if the maximum threshold is above 100%. The queue borrows extra buffers from common buffer pool, if there are any. The queue may grow up to 4 times of its allocated size with this feature.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com



Now look at the new figure to see the relation between the drop thresholds and the reserved/maximum thresholds. As you can see, reserved threshold simpy defines the cut-off point inside the space allocated for a queue. The remaining space goes to the common buffer pool. Drop thresholds are also in percents, and they may not exceed the maximum threshold. The following is the list of restrictions that apply to the thresholds:

1) Maximum Thresholds <= 400%
2) Reserverd Threshold <= 100%
3) Drop Thresholds <= Maximum Threshold

You can even set the Reserved Threshold to 0%, putting all buffer space allocated to a queue to the common pool. In this condition, the queue will only rely on the common pool for the buffer space.

Here are the default settings for buffers in queue-set 1:

```
Rack1SW1#show mls qos queue-set
Queueset: 1
Queue     :       1       2       3       4
----------------------------------------------
buffers   :      25      25      25      25
threshold1:     100     200     100     100
threshold2:     100     200     100     100
reserved  :      50      50      50      50
maximum   :     400     400     400     400
```

Now the last piece is mapping of the DSCP/CoS value to drop thresholds. You do this using the familiar global mapping commands:

mls qos srr-queue output dscp-map threshold <thresh-id> <DSCP List>

Remember that threshold 3 is a non-configurable "queue-full" state threshold. Generally, you map important traffic to the larger threshold, implementing Weighted Tail Drop (WTD) discipline.

Verification can be performed as follows.

```
Rack1SW1#show mls qos queue-set 2
Queueset: 2
Queue     :       1       2       3       4
----------------------------------------------
buffers   :      10      70      10      10
threshold1:     200     200     200     200
threshold2:     150     150     150     150
reserved  :     100      10     100     100
maximum   :     250     250     250     250

Rack1SW1#show mls qos interface gigabitEthernet 0/1 buffers
GigabitEthernet0/1
The port is mapped to qset : 2
The allocations between the queues are : 10 70 10 10

Rack1SW1#show mls qos maps dscp-output-q
   Dscp-outputq-threshold map:
     d1 :d2    0     1     2     3     4     5     6     7     8     9
     ------------------------------------------------------------
      0 :    02-01 02-01 02-01 02-01 02-01 02-01 02-01 02-01 02-02 02-01
      1 :    02-02 02-01 02-01 02-01 02-01 02-01 03-01 03-01 03-02 03-01
      2 :    03-01 03-01 03-01 03-01 03-01 03-01 03-01 03-01 03-01 03-01
      3 :    03-01 03-01 04-01 04-01 04-02 04-01 04-01 04-01 04-01 04-01
      4 :    01-01 01-01 01-01 01-01 01-01 01-01 01-03 01-01 04-01 04-01
      5 :    04-01 04-01 04-01 04-01 04-01 04-01 04-01 04-01 04-01 04-01
      6 :    04-01 04-01 04-01 04-01
```

Note the DSCP values mapped to the new thresholds in the above output.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.80 Catalyst QoS DSCP Mutation

- Configure SW4 so that R1 sees packets marked at R4 with DSCP values CS0, AF31, and CS5 as DSCP values CS1, CS3, and EF.

Configuration

```
SW4:
mls qos
!
mls qos map dscp-mutation MUTATION 0 to 8
mls qos map dscp-mutation MUTATION 26 to 24
mls qos map dscp-mutation MUTATION 40 to 46
!
interface FastEthernet 0/4
 mls qos trust dscp
 mls qos dscp-mutation MUTATION
```

Verification

> **Note**
>
> A DSCP mutation map is a special function that may come in handy on the border of two QoS domains that use different markings. You may create many maps and apply them per-interface. The syntax for the mapping command is:
>
> **mls qos map dscp-mutation <NAME> <From: DSCP List> <To: DSCP Value>**
> **...**
>
> Each line in a DSCP mutation map translates a list of up to 8 DSCP values to a new value. One necessary condition to use the mutation maps is trusting DSCP on the ingress interface. To verify this create an access-list in R1 to detect the mentioned DSCP values. Disable MLS QoS in all switches except SW4.

```
R1:
no ip access-list extended DETECT
ip access-list extended DETECT
 permit icmp any any dscp 0
 permit icmp any any dscp cs1
 permit icmp any any dscp 26
 permit icmp any any dscp 24
 permit icmp any any dscp 40
 permit icmp any any dscp 46
 permit ip any any
!
interface FastEthernet 0/0
 ip access-group DETECT in
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
SW1, SW2, SW3:
no mls qos
```

Send ICMP packets from R4 marked with DSCP 0, DSCP AF31 (TOS 106), DSCP CS5 (TOS 160).

```
Rack1R4#ping 155.1.146.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 155.1.146.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/3/4 ms
Rack1R4#

Rack1R4#ping
Protocol [ip]:
Target IP address: 155.1.146.1
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface:
Type of service [0]: 104
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 155.1.146.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/4/4 ms

Rack1R4#ping
Protocol [ip]:
Target IP address: 155.1.146.1
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface:
Type of service [0]: 160
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 155.1.146.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/3/4 ms
```

```
Rack1R1#show ip access-lists
Extended IP access list DETECT
    10 permit icmp any any dscp default
    20 permit icmp any any dscp cs1 (15 matches)
    30 permit icmp any any dscp af31
    40 permit icmp any any dscp cs3 (15 matches)
    50 permit icmp any any dscp cs5
    60 permit icmp any any dscp ef (30 matches)
    70 permit ip any any (240 matches)
```

Note that the only access-list entries matched are the mutated (mapped) values.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

10.81 Advanced HTTP Classification with NBAR

- Configure R6's VLAN146 interface so that HTTP transfers of files with extensions ".bin", ".text" and ".taxt" are limited to 256Kbps.
- Use only one line to match the URL strings.
- Use the method most friendly for TCP connections.
- Do not apply this limit to the transfers of any other file types.

Configuration

```
R6:
class-map match-all EXTENSION
 match protocol http url "*.bin|*.t[ea]xt"
!
!
policy-map SHAPE
 class EXTENSION
  shape average 256000
!
interface FastEthernet 0/0.146
  service-policy output SHAPE
```

Verification

Note

NBAR (Network Based Application Recognition) protocol classification allows deep inspection for some protocol types. One most notable feature is matching various fileds inside HTTP headers. In recent IOS versions, this capability evolved into a full-blown Flexible Packet Matching feature, which allows for comprehensive inspection of IP packets.

Most commonly you will see HTTP inspection matching the URL or URI (Uniform resource Identified, the part after "hostname" in http://hostname/<URI>) fields against certain strings. While URL is commonly found in client requests, URI is often a part of server reply, and NBAR matches it as well.

The command to match URL is

```
match http protocol url  <pattern>
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

The URI matching feature uses special wildcard symbols:

"*" – matches any sequence of characters (non-empy)
"?"– matches any single character (you need to press Ctrl-V in order to enter "?")
"|" – alternation, logical OR
"[]" – grouping, e.g. [ab] matches either "a" or "b"

Matching is case-sensitive and you can use patterns like [aA] to match both cases. The engine matches the URL/URI in both client requests and server response, so you can use the same class-map to match both directions of the traffic (from client to server and from server to client) unless you matches the hostname there (e.g. match "http://www.cisco.com").

We are going to use the following single-line expression for our task:

"*.bin|*.t[ea]xt"

This pattern matches either of two strings that ends with ".bin" or ".text", ".taxt". You can also use "match-any" class map and match multiple URL strings on separate lines, resulting in the same effect as separating patterh with the symbol "|".

You can match other fields in the HTTP headers. Specifically you can match a pattern against Client and Server headers, using the expressions:

```
match http protocol c-header-field <string>
match http protocol s-header-field <string>
```

Those expressions match any of the client or server headers. Additionally you can match the "Host" header field separately using the command

```
match http protocol host <string>
```

Finally, you can match MIME types to distinguish certain "types" of files being transferred (e.g. audio/video) using the following command:

```
match http protocol mime <MIME-Type>
```

e.g.

```
match http protocol mime "image/jpeg"
```

However, matching URIs is the most common use of advanced HTTP classification. To verify the configuration, set SW1 as an HTTP server and create several file-names in the flash memory with extensions ".bin", ".text", ".taxt", "tsxt".

```
SW1:
ip http server
ip http server path flash:

Rack1SW1#copy flash:config.text flash:config.bin
Rack1SW1#copy flash:config.text flash:config.tsxt
Rack1SW1#copy flash:config.text flash:config.taxt
```

Now we have four files to try. Using R4 as HTTP client verify that ".bin", ".text" and ".taxt" match our configuration but ".tsxt" does not.

```
Rack1R6#clear counters fastEthernet 0/0
Clear "show interface" counters on this interface [confirm]

Rack1R4#copy http://admin:cisco@155.1.67.7/config.bin null:
Loading http://***********@155.1.67.7/config.BIN !
2438 bytes copied in 0.048 secs (50792 bytes/sec)

Rack1R6#show policy-map interface fastEthernet 0/0.146
 FastEthernet0/0.146

  Service-policy output: SHAPE

    Class-map: EXTENSION (match-all)
      4 packets, 1528 bytes
      5 minute offered rate 1000 bps
      Match: protocol http url "*bin|*t[ea]xt"

    Class-map: class-default (match-any)
      35 packets, 7957 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any

Rack1R4#copy http://admin:cisco@155.1.67.7/config.text null:
Loading http://***********@155.1.67.7/config.text !
2438 bytes copied in 0.044 secs (55409 bytes/sec)

Rack1R6#show policy-map interface fastEthernet 0/0.146
 FastEthernet0/0.146

  Service-policy output: SHAPE

    Class-map: EXTENSION (match-all)
      9 packets, 3047 bytes
      5 minute offered rate 1000 bps
      Match: protocol http url "*bin|*t[ea]xt"
```

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

```
Class-map: class-default (match-any)
      43 packets, 9637 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any
```

```
Rack1R4#copy http://admin:cisco@155.1.67.7/config.taxt null:
Loading http://***********@155.1.67.7/config.taxt !
2438 bytes copied in 0.048 secs (50792 bytes/sec)
```

```
Rack1R6#show policy-map interface fastEthernet 0/0.146
 FastEthernet0/0.146

  Service-policy output: SHAPE

    Class-map: EXTENSION (match-all)
      13 packets, 4642 bytes
      5 minute offered rate 1000 bps
      Match: protocol http url "*bin|*t[ea]xt"

    Class-map: class-default (match-any)
      49 packets, 10897 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any
```

```
Rack1R4#copy http://admin:cisco@155.1.67.7/config.tsxt null:
Loading http://***********@155.1.67.7/config.tsxt !
2438 bytes copied in 0.044 secs (55409 bytes/sec)
```

```
Rack1R6#show policy-map interface fastEthernet 0/0.146
 FastEthernet0/0.146

  Service-policy output: SHAPE

    Class-map: EXTENSION (match-all)
      13 packets, 4642 bytes
      5 minute offered rate 0 bps
      Match: protocol http url "*bin|*t[ea]xt"

    Class-map: class-default (match-any)
      58 packets, 13327 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any
```

Note how the packet counter for the class increases for the first three file names.

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com

Copyright © 2009 Internetwork Expert www.InternetworkExpert.com