Table of Contents

Understanding and Configuring Backbone Fast on Catalyst Switches

Document ID: 12014

Introduction

Backbone fast is a Cisco proprietary feature that, once enabled on all switches of a bridge network, can save a switch up to 20 seconds (max_age) when recovering from an indirect link failure. After a quick review of some Spanning−Tree Protocol (STP) basics, you will see the exact failure scenario to which backbone fast applies and how to configure it for Catalyst switches running both CatOS and Integrated Cisco IOS (Native Mode) (Catalyst 6000 Series switches, Catalyst 4000 switches, and Catalyst 2950/3550 series switches).

Before You Begin

Conventions

For more information on document conventions, see the Cisco Technical Tips Conventions.

Prerequisites

There are no specific prerequisites for this document.

Components Used

The information in this document is based on the software and hardware versions below.

- Catalyst 2950 Series Switches 12.1(6)EA2 and later
- Catalyst 3550 Series Switches 12.1(4)EA1 and later
- Catalyst 4000 Series Switches 5.1(1a) and later

- Catalyst 4000 Switch running Integrated Cisco IOS (Native Mode) 12.1(8a)EW and later
- Catalyst 5000 Series Switches CatOS Version 4.1(1) and later
- Catalyst 6000 Series Switches CatOS Version 5.1(1)CSX and later
- Catalyst 6000 Series switches Integrated Cisco IOS (Native Mode) Version 12.0−7XE and later

BPDUs and How to Compare Them

Bridge Protocol Data Units (BPDUs) can be strictly classified using the fields they carry. Among these fields are the root bridge ID, path cost to the root, and sender bridge ID. A BPDU is considered better than another BDPU for the following reasons:

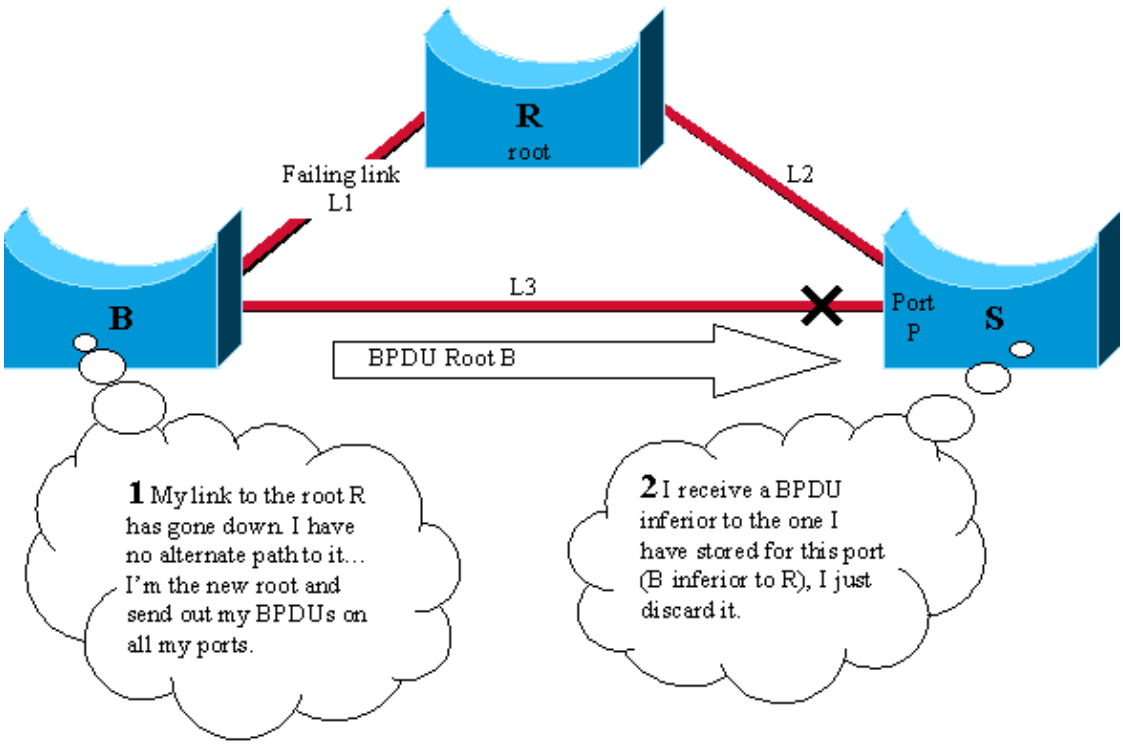
- When one BPDU carries a better root bridge ID than another (the lower the value, the better).
- When the root bridge ID values are equal, then the BPDU with the lowest path cost to the root is better.
- When the root bridge ID values are equal and the costs to the root are the same, then the BPDU with the better sender bridge ID is better (the lower the value, the better).

There are other variables that then can act as a tie−breaker, however, the better a BPDU, the better the access to the best root bridge.

A bridge that receives a BPDU on a port better than the one it would send out, puts this port in blocking mode unless it is its root port. This means that on the segment connected to this port, there is another bridge that is a designated bridge. A bridge stores the value of the BPDU on a port sent by the current designated bridge.

How STP Recovers From an Indirect Link Failure

Below is illustrated how STP behaves when it has to recalculate after an indirect link failure, that is, when a bridge has to change the status of some of its ports because of a failure on a link that is not directly attached to it.



Consider the above diagram involving three switches R, B, and S in a fully meshed topology. Assume that R is the root bridge and B is backup root bridge. S is blocking its port P and B is the designated bridge for link L3.

1. If link L1 goes down, switch B immediately detects the failure and assumes it is the root. It starts sending BPDUs to S claiming to be the new root.
2. When S receives this new BPDU from B, it realizes it is inferior to the one it had stored for port P and ignores it.
3. After max_age timer has expired (20 seconds by default), the BPDU stored on S for port P ages out. The port goes immediately to listening and S starts sending its better BPDU to B.
4. As soon as B receives the BPDU from S, it stops sending its BPDU.
5. Port P moves to the forwarding state through listening and learning states. This takes twice the fw_delay value, an additional 30 seconds. Full connectivity is then restored.

It took the max_age value (20 seconds) plus twice the fw_delay value (2x15 seconds) to recover from this indirect link failure. This is 50 seconds with the default parameters. The backbone fast feature proposes to save max_age (20 seconds) by aging out immediately after the port receive inferior BPDUs.

Backbone Fast Enhancements to Standard STP

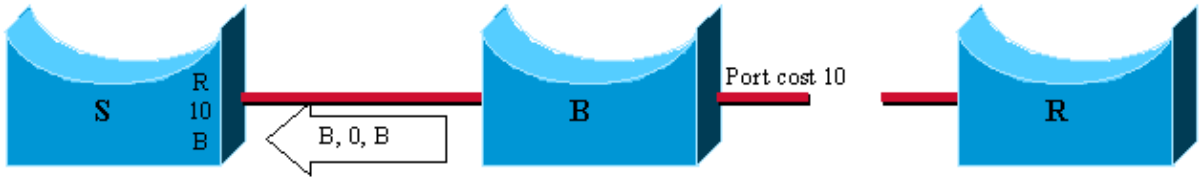
With the previous example, STP invalidates information that became wrong because of an indirect link failure by passively waiting for max_age. To get rid of this max_age delay, backbone fast introduces two enhancements:

- The ability to detect an indirect link failure as soon as possible. This is achieved by tracking the inferior BPDUs that a designated bridge sends when it experiences a direct link failure.
- A mechanism allowing to check immediately if the BPDU information stored on a port is still valid. This is implemented introducing a new protocol data unit (PDU) and the Root Link Query (referred to in this document as the RLQ PDU).

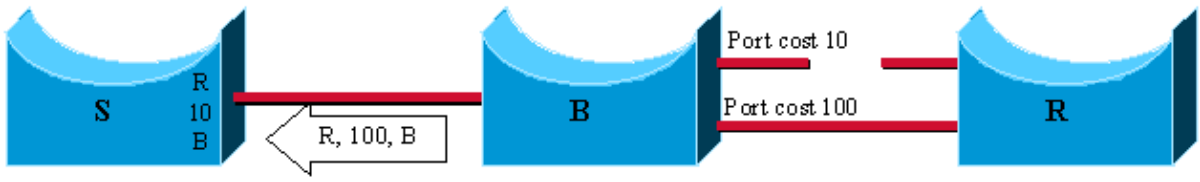
Detecting Indirect Link Failures

If an inferior BPDU is received on a port from our designated bridge, then this bridge has:

1. Lost the root and starts advertising a root with a higher bridge ID (a worse root than ours).
2. Or its path to the root has increased above ours.



1 in this case, B lost the root and sends a BPDU with root id B, path cost 0 and bridge id B. It is inferior to the one that S had stored, because R is a better root than B.



2 In this case, B still has the root R, but the failure implied the cost to increase from 10 to 100. The BPDU sent is, again, inferior to the one stored by S (path cost 100 > value stored = 10)

The usual behavior in regards to the Institute of Electrical and Electronics Engineers (IEEE) specifications is to simply ignore any inferior BPDUs. Backbone fast will use them because as soon as one is received, it is certain that a failure occurred on the path to the root and that you will have to age out at least one port.

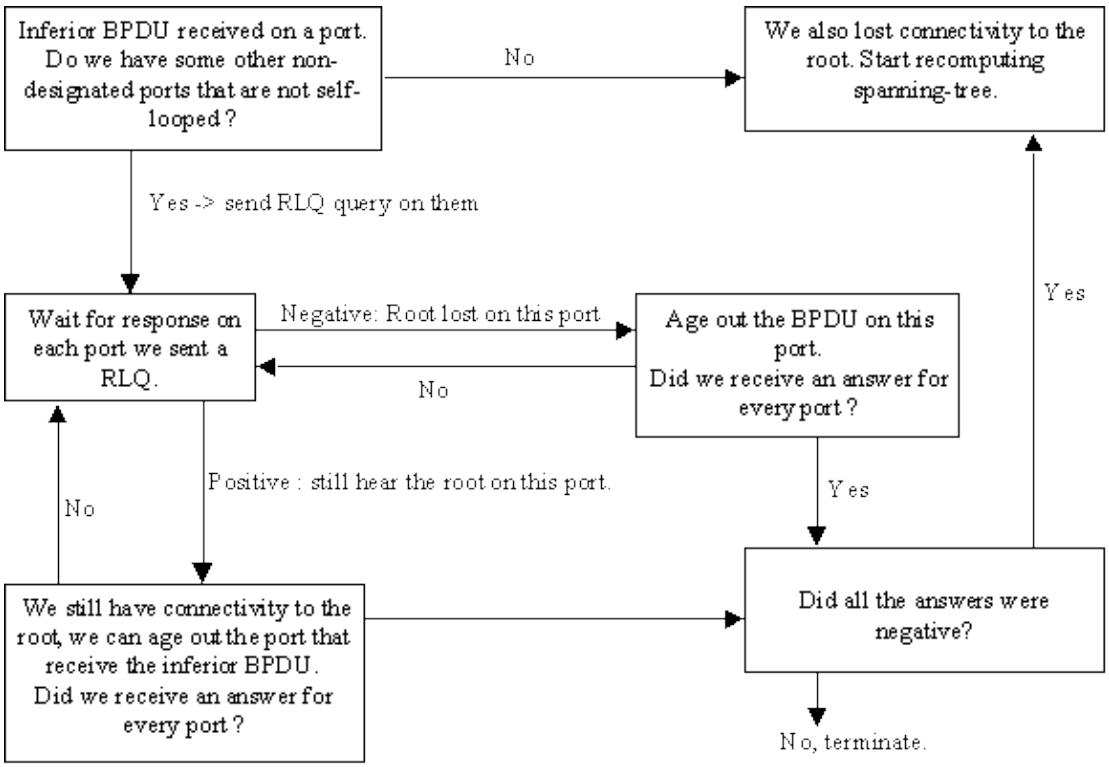
Note: An indirect link failure can happen without any inferior BPDU being generated in the network. Simply add a hub in the previous diagram:

The link failure occurs between the root bridge R and the hub. B doesn't detect the link going down and will wait max_age before claiming to be the new root. Remember that the mechanism will only work if a bridge detects a direct link failure.

Only keep track of inferior BPDUs sent by the designated bridge. Since this is the BPDU that is stored on the port. If, for instance, a newly inserted bridge starts sending inferior BPDU, it will not start the backbone fast feature.

Reacting to Indirect Link Failures

When an inferior BPDU has been detected on a non−designated port, the second phase of backbone fast is triggered. Instead of passively waiting max_age to age out ports that could have been affected by the failure, a proactive way of testing them immediately is introduced by means of the RLQ PDU. The RLQ is used to achieve a kind of ping for the root on a non−designated port and allowed to quickly confirm if the BPDU stored on a port is still valid or needs to be discarded.



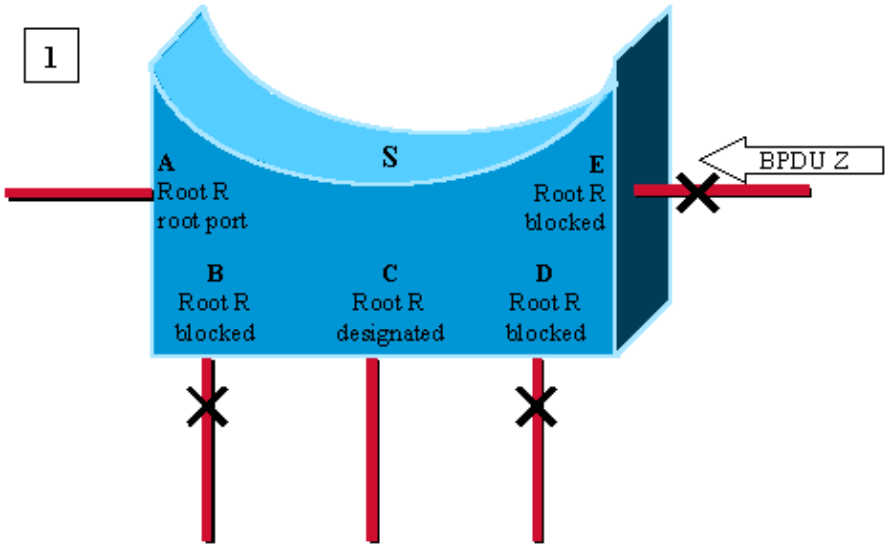
On receipt of an inferior BPDU from a designated bridge, send a RLQ PDU on all non−designated ports except the port where you received the inferior BPDU and self−looped ports. This in order to check that you still hear from the root on ports where you are used to receiving BPDUs. The port where you received the inferior BPDU is excluded because you should already be aware that it suffered from a failure, self−looped and designated ports are not useful, as they don't lead to the root.

On the receipt of a RLQ response on a port, if the answer is negative, the port lost connection to the root and you can age out its BPDU. Furthermore, if all other non−designated ports have already received a negative answer, the whole bridge has lost the root and can start the STP calculation from scratch.

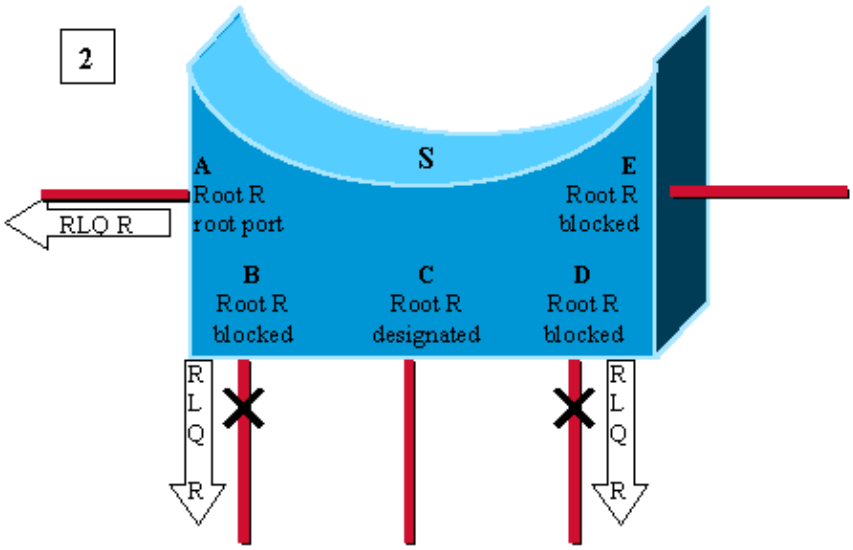
If the answer confirms you can still access the root bridge via this port, you can immediately age out the port on which we initially received the inferior BPDU.

In the following example, ports A, B, D, and E are non−designated ports for switch S. A is the root port and the others are blocking. When E receives an inferior BPDU (1), backbone fast kicks in to speed up STP recalculation.

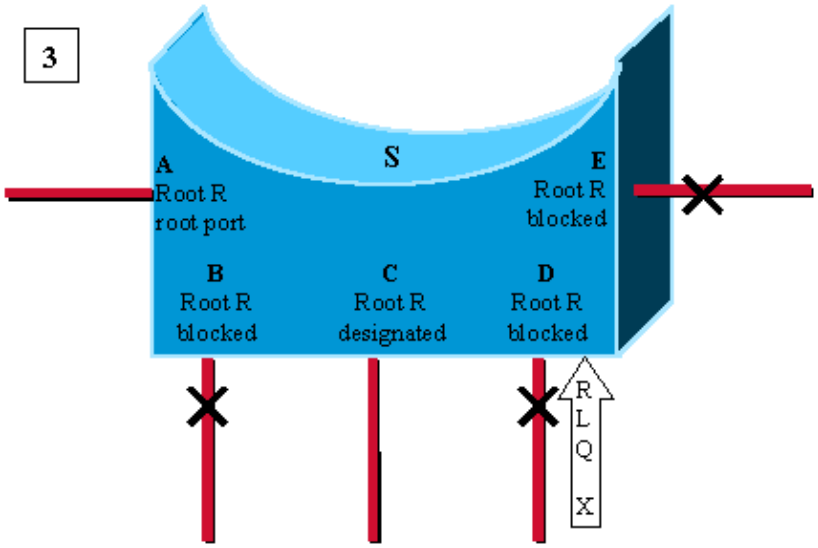
Send out a RLQ request, looking for root R on all non−designated ports but E (2). The replies will specify which root is accessible via these ports. The RLQ response that D receives specifies that D lost its path to the root R. Age its BPDU out immediately (3). Ports A and B receive confirmation that they still have a path to R (4). So, as the switch S still has connectivity to the root, immediately age out port E and carry on with regular STP rules (5).



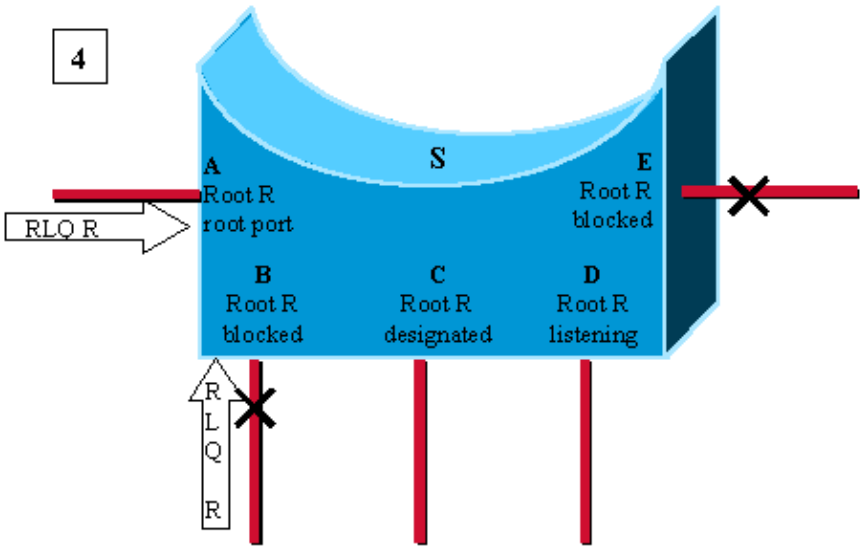
Port E receives an inferior BPDU, advertising root Z instead of root R stored on the different ports.



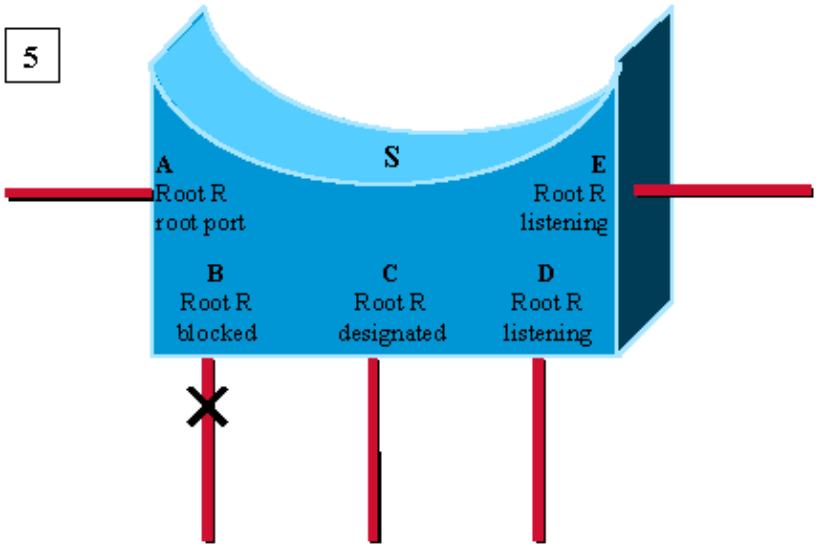
Switch S needs to recheck all its other non-designated ports. It sends out a RLQ request for root R on ports A,B and D.



Port D is the first to receive and RLQ response from bridge X claiming to be the root. It is a negative response: D has lost connectivity to the root R. We age out immediately the BPDU on port D and go to listening. As we don't know if we still have connectivity to the root R, we don't age out port E yet.



Here, A and B receive a RLQ response confirming R as being the root. As switch S still has connectivity to the root, we can age out immediately the BPDU stored on port E.



Port E transitions to listening, without waiting for max_age. Usual spanning-tree rules then apply to determine whether E and D will eventually go to blocking or forwarding.

In a case where the switch received only responses with a root different from R, consider the root as lost and restarted STP calculation from scratch immediately. Note that this case also occurs when the only non−designated (and non self−looped) port on the bridge is the root port and you receive an inferior BPDU on this port.

The Root Link Query PDU

The two forms of RLQs are RLQ requests and RLQ responses.

The RLQ request is sent out on a port where you usually receive BPDUs, in order to check that you still have connectivity to the root via this port. Specify in the request which bridge is your root and the RLQ response will eventually come back with root bridge that can be accessed via this port. If the two roots are the same, connectivity is still alive, else, it is lost.

A bridge receiving a RLQ request immediately answers if it knows it has lost connection to the root queried (because it has a root bridge different to the one specified in the RLQ query), and if it is the root.

If this is not the case, then, it forwards the query toward the root via its root port.

RLQ responses are flooded on designated ports. The sender of the RLQ request puts its bridge ID in the PDU. This is to ensure that when it receives back a reply to its own query, it does not flood the response on its designated ports.

The RLQ PDU has the same packet structure as a normal STP BPDU. The only difference is that two different Cisco−specific SNAP addresses are used: one for the request and one for the reply.

Below is the standard BPDU format:

DA	SA	Length	DSAP	SSAP	CNTL	SNAP	PDU

Below is the PDU field is:

Protocol identifier	Version	Message type	Flags	Root ID	Root path cost
Sender ID	Port ID	Message age	Max age	Hello time	Forward delay

The message type used in the PDU is also different from the standard BPDU.

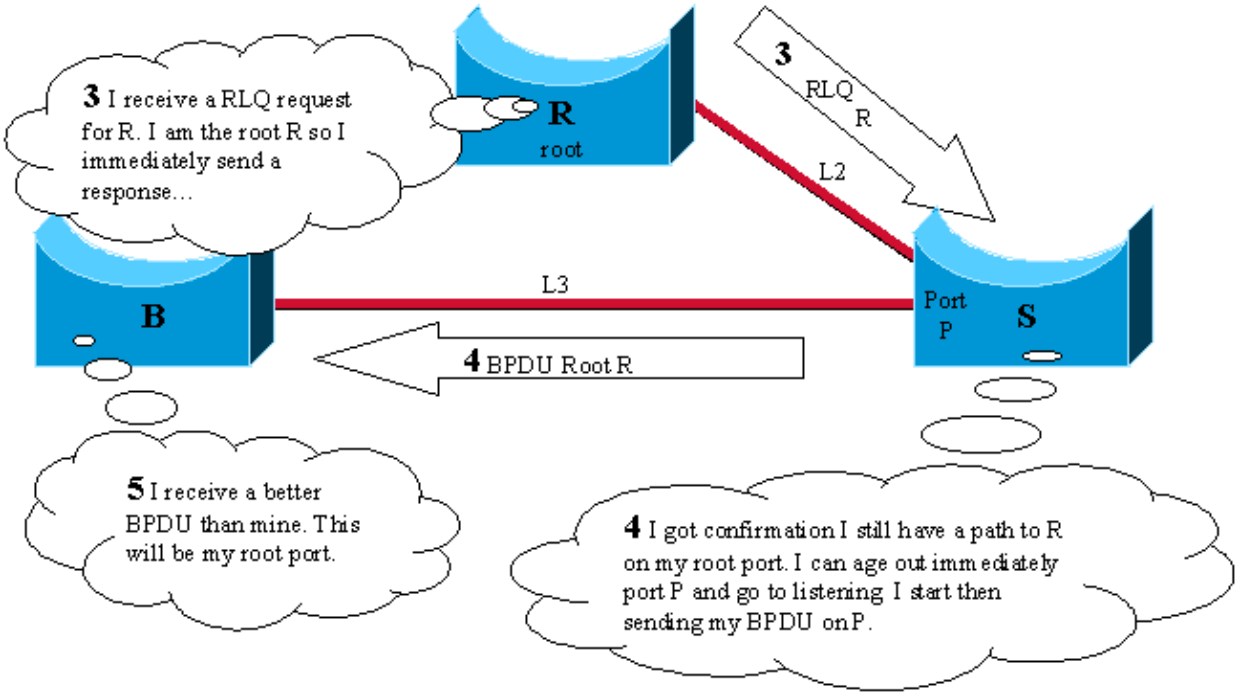
The only fields used are the root ID and the sender bridge ID.

This Cisco−specific feature needs to be configured on all switches in the network in order to process these PDUs.

Example Scenario with Backbone Fast Feature Enabled

The following scenario is based on the first example, however, this time with backbone fast enabled on the three switches.

1. The first stage is exactly the same as previously explained.
2. As soon as S receives the inferior BPDU from B, it starts reconfirming its non−designated ports instead of waiting max_age. It sends a RLQ query on its root port for root bridge R.
3. Root bridge R receives the query and immediately answers with a RLQ response specifying there is still a root R in that direction.
4. S has now checked all its non−designated ports, and it still has connectivity to the root. It can then age out immediately the information stored on port P. P transitions to listening and starts sending BPDUs. At that stage, you have already saved max_age seconds, and the standard Spanning−Tree Algorithm (STA) applies then.
5. B receives the better BPDU from S (R better root than B) and considers now the ports leading to L3 as its root port.



Configuring Backbone Fast for CatOS and Integrated Cisco IOS (Native Mode)

Configuration for CatOS

For Catalyst 4000, 5000, and 6000 series switches running CatOS, use the following commands to enable backbone fast globally for all ports and to verify the configuration.

```
Console> (enable) set spantree backbonefast enable
Backbonefast enabled for all VLANs
Console> (enable) show spantree backbonefast

! This command show that the backbonefast feature is enabled.

Backbonefast is enabled.
Console> (enable)
```

To display backbone fast statistics:

```
Console> (enable) show spantree summary
Summary of connected spanning tree ports by vlan
Uplinkfast disabled for bridge.
Backbonefast enabled for bridge.
Vlan  Blocking Listening Learning Forwarding STP Active
----- -------- --------- -------- ---------- ----------
 1      0        0          0         1         1

      Blocking Listening Learning Forwarding STP Active
----- -------- --------- -------- ---------- ----------
Total   0        0         0        1          1

BackboneFast statistics

! The show spantree summary command displays all backbonefast statistics.
```

```
----------------------
Number of inferior BPDUs received (all VLANs): 0
Number of RLQ req PDUs received (all VLANs): 0
Number of RLQ res PDUs received (all VLANs): 0
Number of RLQ req PDUs transmitted (all VLANs): 0
Number of RLQ res PDUs transmitted (all VLANs): 0
Console> (enable)
```

Configuration for Integrated Cisco IOS (Native Mode) (Catalyst 6000, Catalyst 4000, Catalyst 2950 Series, and Catalyst 3550 Series)

For Catalyst 6000 series switches , Catalyst 4000 series switches, Catalyst 2950 Series switches and Catalyst 3550 Series switches use the following commands to enable backbone fast globally for all interfaces.

```
CAT−IOS# configure terminal
CAT−IOS(config)# spanning−tree backbonefast
CAT−IOS(config)# end
CAT−IOS#
```

To verify that backbone fast is enabled and to show statistics:

```
CAT−IOS# show spanning−tree backbonefast

BackboneFast         is enabled

BackboneFast statistics
----------------------
Number of transition via backboneFast (all VLANs)          : 0
Number of inferior BPDUs received (all VLANs)              : 0
Number of RLQ request PDUs received (all VLANs)            : 0
Number of RLQ response PDUs received (all VLANs)           : 0
Number of RLQ request PDUs sent (all VLANs)                : 0
Number of RLQ response PDUs sent (all VLANs)               : 0
CAT−IOS#
```

Related Information

- **Technical Support − Cisco Systems**

Updated: May 10, 2005 Document ID: 12014