**Virtual switching with Open vSwitch**

**Switching Station**

*Ralf Spenneberg*

Virtualization with Vmware, KVM, and Xen is here to stay. But up to now, no virtual switch has supported complex scenarios. Open vSwitch supports flows, VLANS, trunking, and port aggregation just like major league switches.

Many corporations are moving their whole infrastructure to virtual systems. This process involves virtualizing centralized components like SAP systems, Oracle database servers, email systems, and fileservers, thus facilitating administration. Additionally, administrators no longer need to shut down systems for maintenance, because the workloads can be migrated on the fly to other virtual hosts.

One of the biggest disadvantages of a virtual environment has always been the simplistic network structure. Although physical network switches support VLANs, trunking, QoS, port aggregation, firewalling, and Layer 3 functionality, virtual switches are very simple affairs. VMware provided a solution in cooperation with Cisco, which now offers the virtual Cisco Nexus 1000V switch for VMware environments. The switch integrates with the VMware environment and offers advanced functionality.

An open source product of this caliber previously has not been available, but Open vSwitch tackles the problem. Open vSwitch supports Xen, KVM, and VirtualBox, as well as XenServer. The next generation of Citrix will also be moving to Open vSwitch.

Open vSwitch [[1]](http://www.admin-magazine.com/CloudAge/content/view/full/3868/(offset)/4#article_i1), which is based on Stanford University's OpenFlow project [[2]](http://www.admin-magazine.com/CloudAge/content/view/full/3868/(offset)/4#article_i2), is a new open standard designed to support the management of switches and routers with arbitrary software (see the "OpenFlow" box).

**OpenFlow**

The OpenFlow project aims to revolutionize the world of routers and switches. A classical router or switch combines two functions in a single device:

* Fast packet forwarding (data path)
* Decisionmaking on how and where to forward packets (control path)

These two systems typically work independently on the same device. The data path component only asks the control path component if it doesn't know how and where to route a packet. The control path component then determines the path/route and stores it in the flow table. All other packets in the same flow can then be forwarded quickly by the data path engine.

OpenFlow offloads the control path onto a separate controller, which can be a simple server. The OpenFlow switch (data path) and controller then communicate over a secure channel.

The OpenFlow switch stores the flow table where the controller saves the individual flows. Each flow describes the properties of the packets that make up the flow and how the switch should handle the packets (drop, sendout port, and so on). Once the switch receives a packet for which it doesn't have a matching entry in the table, it sends the packet to the controller, which analyzes the packet, makes a decision and stores the decision in the flow table.

Because of cooperation between multiple manufacturers, the developers have been able to achieve OpenFlow support in several commercial network appliances. Customized firmware exists for several switches by HP, NEC, Toroki, and Pronto [[3]](http://www.admin-magazine.com/CloudAge/content/view/full/3868/(offset)/4#article_i3). Open vSwitch is a software implementation that provides both functionalities (data path and controller).

Open vSwitch gives the administrator the following features on a Linux system:

* Fully functional Layer 2 switch
* NetFlow, sFlow, SPAN, and RSPAN support
* 802.1Q VLANs with trunking
* QoS
* Port aggregation
* GRE tunneling
* Compatibility with the Linux bridge code (brctl)
* Kernel and userspace switch implementation

Before you can benefit from these features, however, you first need to install Open vSwitch. Prebuilt packages exist for Debian Sid (unstable). I have released packages for Fedora/Red Hat that you can download from my own website [[4]](http://www.admin-magazine.com/CloudAge/content/view/full/3868/(offset)/4#article_i4). You can also install from the source code (see the "Installation" box).

**Installation**

After unpacking the source code, you'll need to build and install Open vSwitch using the typical commands:

./configure with l26=U

/lib/modules/$(uname r)/build

make

sudo make install

You'll need the kernel headers to build the kernel module. In recent distributions, you will find typically find the headers in a package named kerneldevel, or something similar. After the build, you should check the installation and launch the software for the first time. To do so, load the kernel module manually:

modprobe datapath/linux 2.6/U

openvswitch\_mod.ko

If this command fails, you probably need to unload a bridge module: **rmmod bridge**.

The kernel module version may not match your current kernel, which can be a problem if you use prebuilt packages. In this case, you need to rebuild the module. After doing so, you can initialize the Open vSwitch configuration database:

ovsdb tool create U

/usr/local/etc/ovs vswitchd.conf.db U

vswitchd/vswitch.ovsschema

In case of other issues, the INSTALL.Linux file provides troubleshooting tips.

Although the packages provide start scripts for simple use, you will need to launch manually or create your own start script in case of a manual installation. The configuration database handles switch management (see [Listing 1](http://www.admin-magazine.com/CloudAge/Articles/Virtual-switching-with-Open-vSwitch#article_l1)).

**Listing 1**

**Configuration**

01 ovsdb server /usr/local/etc/ovs vswitchd.conf.db \

02 remote=punix:/usr/local/var/run/openvswitch/db.sock \

03 remote=db:Open\_vSwitch,managers \

04 private key=db:SSL,private\_key \

05 certificate=db:SSL,certificate \

06 bootstrap ca cert=db:SSL,ca\_cert

The next step is to launch the Open vSwitch service:

ovs vswitchd unix:/usr/local/var/U

run/openvswitch/db.sock

You can now run the **ovs vsctl** command to create new switches or add and configure ports. Because most scripts for Xen and KVM rely on the bridge utilities, and on the **brctl** command to manage the bridge, you will need to start the bridge compatibility daemon. To do this, load the kernel module and then start the service:

modprobe U

datapath/linux 2.6/brcompat\_mod.ko

ovs brcompatd U

pidfile detach U

vANY:console:EMER unix:/usr/U

local/var/run/openvswitch/db.sock

You can now use the bridge utilities to manage your Open vSwitch:

brctl addbr extern0

brctl addif extern0 eth0

Distribution scripts for creating bridges will work in the normal way. You can also use **ovs vsctl** to manage the bridge. In fact, you can use both commands at the same time ([Listing 2](http://www.admin-magazine.com/CloudAge/Articles/Virtual-switching-with-Open-vSwitch#article_l2)).

**Listing 2**

**Bridge Management**

01 [root@kvm1 ~]# brctl show

02 bridge name bridge id STP enabled interfaces

03 extern0 0000.00304879668c no eth0

04 vnet0

05 [root@kvm1 ~]# ovs vsctl list ports extern0

06 eth0

07 vnet0

If the **brctl show** command says it can't find some files in the **/sys/** directory, the bridge utilities may be too new (e.g., on RHEL 6). In this case, you might want to downgrade to the latest version of RHEL 5. Up to this point, Open vSwitch has acted exactly like a bridge set up using the bridge utilities. Some additional configuration steps are necessary to use the advanced features. All of the settings in the Open vSwitch configuration database can be handled using the **ovs vsctl** command.

**NetFlow**

Open vSwitch can export the NetFlows within the switch. To allow this to happen, you first need to create a new NetFlow probe.

# ovs vsctl create netflow target=U

"192.168.0.5\:5000"

75545802 675f 45b2 814e 0875921e7ede

Then link the probe with the **extern0** bridge:

# ovs vsctl add bridge extern0 netflow U

75545802 675f 45b2 814e 0875921e7ede

If you previously launched a NetFlow collector (such as Ntop) on port 5000 of a machine with the address of 192.168.0.5, you can now view the file ([Figure 1](http://www.admin-magazine.com/CloudAge/Articles/Virtual-switching-with-Open-vSwitch#article_f1)).

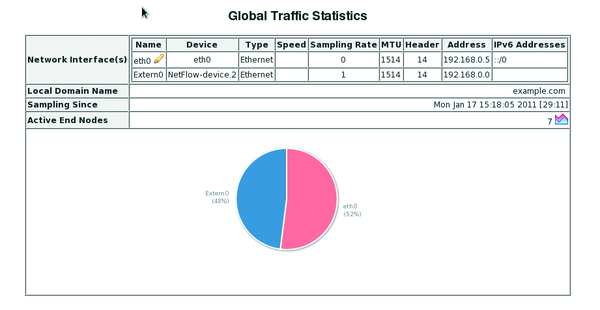


Figure 1: Ntop showing the flows for the bridge.

The configuration settings in the database can be managed using

ovs vsctl list bridge

and

ovs vsctl list netflow

commands and removed using ovs vsctl destroy ….

**QoS**

In many cases, administrators need to restrict the bandwidth of individual virtual guests, particularly when different customers use the same virtual environment. Different guests receive the performance they pay for, based on their Service Level Agreements.

Open vSwitch gives administrators a fairly simple option for restricting the maximum transmit performance of individual guests. To test this, you should first measure the normal throughput. The **iperf** tool is useful for doing so. You can launch iperf as a server on one system and as a client on a virtual guest ([Listing 3](http://www.admin-magazine.com/CloudAge/Articles/Virtual-switching-with-Open-vSwitch#article_l3)).

**Listing 3**

**Performance Measurement**

01 ## Server:

02 ## iperf s

03 ## Client:

04 # iperf c 192.168.0.5 t 60

05

06 Client connecting to 192.168.0.5, TCP port 5001

07 TCP window size: 16.0 KByte (default)

08

09 [ 3] local 192.168.0.210 port 60654 connected with 192.168.0.5 port 5001

10 [ ID] Interval Transfer Bandwidth

11 [ 3] 0.0 60.0 sec 5.80 GBytes 830 Mbits/sec

You can now restrict the send performance. Note that the command expects you to enter the send performance in kbps. Besides the send performance, you will also need to specify the burst speed, which should be about a tenth of the send performance. The **vnet0** interface in this case in this switch port to which the virtual guest is connected.

# ovs vsctl set Interface vnet0 U

ingress\_policing\_rate=1000

# ovs vsctl set Interface vnet0 U

ingress\_policing\_burst=100

You can test the results directly using iperf. In this case, the restrictions work ([Figure 2](http://www.admin-magazine.com/CloudAge/Articles/Virtual-switching-with-Open-vSwitch#article_f2)).

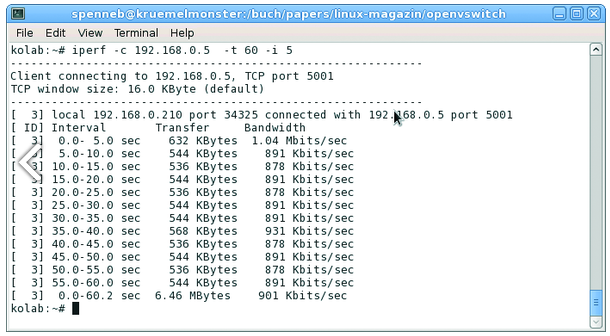


Figure 2: Using iperf to check the effectiveness of send performance restrictions.

If you are familiar with the tc command and classbased QoS on Linux with various queuing disciplines, you can use this tool in combination with Open vSwitch. The man page provides various examples.