



THE HONG KONG
POLYTECHNIC UNIVERSITY
香港理工大學

Virtual Switching in an Era of Advanced Edges

Group6 member: Ng Chi Hong
Leung Chun Yue
Leung Kai Yin



2010

Abbreviation:

Kernel

The kernel is the essential center of a computer operating system, the core that provides basic services for all other parts of the operating system. A synonym is *nucleus*. A kernel can be contrasted with a shell, the outermost part of an operating system that interacts with user commands.

Load Balancing

Load balancing is a technique to distribute workload evenly across two or more computers, network links, CPUs, hard drives, or other resources, in order to get optimal resource utilization, maximize throughput, minimize response time, and avoid overload.

Virtual Machine (VM)

A virtual machine (VM) is an environment, usually a program or operating system, which does not physically exist but is created within another environment.

Access Control List

An access control list (ACL) is a table that tells a computer operating system which access rights each user has to a particular system object, such as a file directory or individual file.

Hypervisor

A hypervisor, also called a virtual machine manager, is a program that allows multiple operating systems to share a single hardware host. Each operating system appears to have the host's processor, memory, and other resources all to itself.

Packet

A packet is the unit of data that is routed between an origin and a destination on the Internet or any other packet-switched network.

INTRODUCTION.....	4
OBJECTIVES	7
METHODOLOGY FOR TEST	7
OUR WORK ON TESTING OPEN VSWITCH	9
PLAN OF IMPLEMENTATION	9
ENVIRONMENT CONFIGURATION AND INSTALLATION:.....	10
SETUP ENVIRONMENT AND INSTALLATION.....	11
TEST 1 - SECURITY ON VLAN:	13
<i>Cleanup and checking before testing:</i>	<i>14</i>
<i>Test 1 – Security Test</i>	<i>16</i>
<i>Test 2 - Performance Test:</i>	<i>19</i>
<i>TCP stream.....</i>	<i>22</i>
<i>UDP stream.....</i>	<i>23</i>
<i>TCP Connect/Request/Respond</i>	<i>24</i>
<i>UDP Request/Respond.....</i>	<i>25</i>
DIFFICULTIES FROM TESTING.....	29
TEAM’S WORK ALLOCATED	30
CONCLUSION.....	30
FUTURE TREND	31
REFERENCES	31

Introduction

In the modern world, most of the organizations have a challenge to capacity problem. The Traditional networking infrastructure (figure.1) will be replaced by Virtual Networking infrastructure which will be more advantage on server consolidation. That means a single physical hardware allow multiple operating systems and applications.

Although, the virtualization technology is mature and it will be rapidly in use, it might have other issues that the engineers mainly concern such as management, policy enforcement, security and scalability.

In this paper, we had studied the problem occurred in the traditional problems issued in virtualization environment and the functionalities of Open vSwitch as virtual switch. For testing Open vSwitch, we had constructed the Open vSwitch environment to test the security issue and performance issue, when Open vSwitch is used in the Xen Server network. Base on the result of the testing, we had analyzed the results and pointed out the problems we discovered and the difficulties of the testing. Final, a briefly conclusion had sum up our progression in this report.

Figure 1

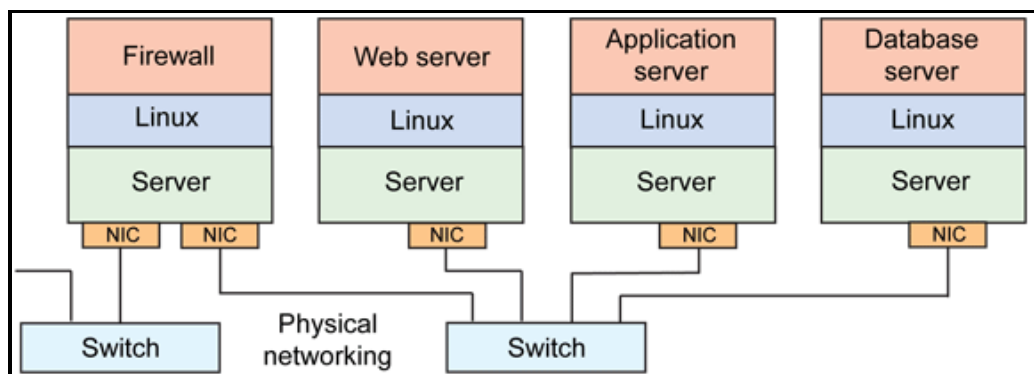
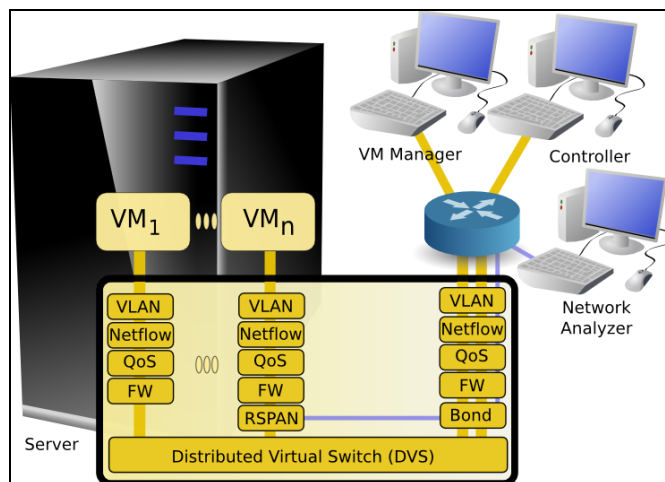


Figure 2



Why use Virtual Switching

The Virtual Switching is a technology which it can allow multiple switching transactions at physical machine or hardware with the following point of views.

Technical view

By using virtualization technology, the engineer no more concerns the hardware environment such as configuration for new physical server which can be consolidated the existing system, so it can make system architecture to be simple.

However, the traditional virtualization technology is that the network traffic will send directly from VM to other VM on the same physical host without pass though physical switch. As a result, the network administrators cannot monitor or manage network traffic. Thus, the objective of Open vSwitch is to address this issue.

Open vSwitch (OVS) is a kernel module, integrating in Linux kernel which helps to solve this virtual network management problem, because OVS takes advantage, by using hypervisor bridge as multilayer virtual switch designed (figure 2). It can be able to controls network packets with VMs. Moreover, it can provide a feature for monitoring and controlling the traffic by NetFlow, sFlow and mirroring (SPAN and RSPAN), fine-grained ACLs and QoS. In additional, it supports for centralized control and port bonding GRE and IPSec tunneling and per-VM traffic policing.

Business view

The organization can reduce the expense on manpower such as engineer and Server consolidation is a key component in reducing your overall expenditures for hardware costs.

Choice of tool for test – What is Open vSwitch

In this project, we will talk about the virtual switching and use a software called “Open vSwitch”. It is a multiplayer software switch licensed under the open source Apache 2 license. It supports standard management interfaces and opens the forwarding functions to programmatic extension and control.

Functionalities of Open vSwitch

- Support multiple Linux-based virtualization technologies (Xen/XenServer, KVM and VirtualBox)
- Visibility into inter-VM communication via NetFlow, sFlow, SPAN and RSPAN.
- Setting QoS rate limiting
- Standard 802.1Q VLAN model with trucking
- Per VM policing
- NIC bonding with source-MAC load balancing
- Kernel-based forwarding
- Support for OpenFlow
- Compatibility layer for the Linux bridging code.

There are comparisons with different virtual switches

Brand	Citrix	Microsoft	VMware	Cisco
Product	XenServer 5.6	Hyper-V 1.0	ESX 3.5	Nexus 1000V
Maxium of Virtual Switch	N/A	127	248	N/A
Support 10GBE Networ card	yes	yes	yes	yes
QOS	yes	no	yes	yes
VLAN	yes	yes	yes	yes
Application of manage	XenCenter 5.6	Hyper-V	Virtual Center 2.5	Virtual Supervisor Module

Objectives

In this project, we setup a testing environment for virtual switching. It is using Open vSwitch at Xen Server with sFlow and QoS. To analysis the features of Open vSwitch, the testing scopes such as security and network performance are defined with the following details.

- Security network by virtual LAN - create two subnets and test by broadcast message
- Stress test for network performance - use 4 scopes of network socket (TCP Steam, UDP Steam, TCP CRR, UDP RR)
- Analysis the results, based on testing from Open vSwitch.

Methodology for test

The tests perform for using OpenvSwitch (OVS) to simulate virtual switching. It is based on two ways. Firstly, we test the security of network architecture. It illustrates the network traffic isolation in VLAN environment. Secondly, we provide a stress test to test performance and retransmission in the virtual switch environment. The detail test procedure and result has been written in page 12.

Test Case	Objective	Method
Security	This test has used ARP broadcast to send out to the different subnet. We observe the ARP broadcast can transport different subnet or not, after performing VLAN.	To ensure there is no ARP cache, all Ethernet interfaces of the nodes has restart and used ARP -n to enquire the ARP cache table. Besides, to monitor the ARP request and reply status in Linux network environment, tcpdump is used as the monitor tool for each host.
Performance	This test tried to find out the performance of OVS and the occurrence of the	There are four scopes of network socket that we would like to analyze because those network

	retransmission in the stress network traffic.	<p>sockets are mostly common appear in the network:</p> <ol style="list-style-type: none"> 1. TCP steam (TCP STREAM) 2. UDP steam (UDP STREAM) 3. TCP Connect/Request/Response (TCP CRR) 4. UDP Request/Response UDP RR <p>To analyze the date in the same situation, we have created the test in the same virtual network infrastructure.</p>
--	---	--

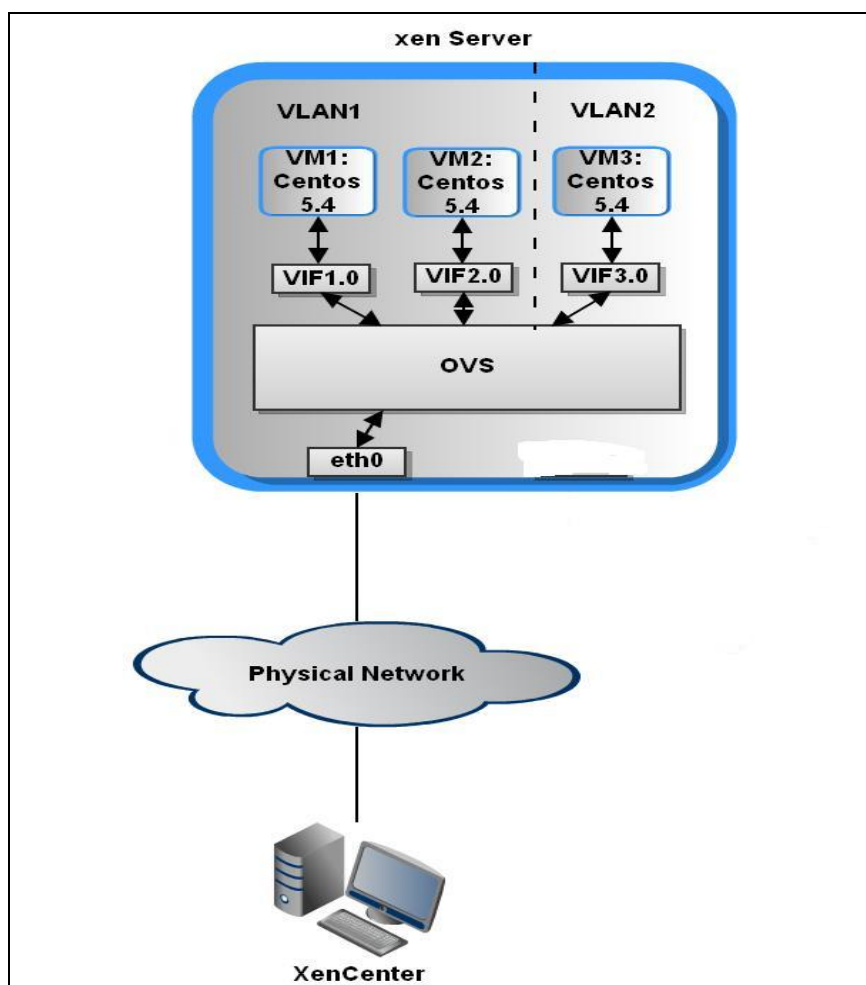
Test Result	Result
Security	After used the "tcpdump arp" to monitor the ARP flow for each node, L2 & L3 are isolated under the test of VLAN environment. It cannot discover any ARP package in data link layer, even it has 1 PIF in the physical host.
Performance	Using client machine to send large amount of different concurrent stream to Server with four stream socket that would like to test. We have found that the retransmission has occurred in TCP STEAM, UDP STEAM and UDP RR. Only TCP CRR has not package lost. However, comparing with the same network configuration, UDP STEAM is more consistence than TCP stream and TCP CRR has lower but steadier transaction rate then UDP RR.

Our Work on testing Open vSwitch

1. Plan of implementation
2. Environment configuration and installation
3. Place Linux kernel bridge with Open vSwitch
4. Using different method to test.

Plan of implementation

The infrastructure of the testing network is shown as below:



1. Install 3 Virtual Machine (VM1,VM2,VM3) in Xen Server
2. Build 2 Virtual LAN (VLAN1, VLAN2) in Xen Server
3. Setup VM1, VM2 at VLAN1
4. Setup VM3 at VLAN2

Environment configuration and installation:

The following configuration list only shows the main components for each system. The other essential modules, such as gcc++, glibc-devel and libXp will be counted as the modules that had included in the OS.

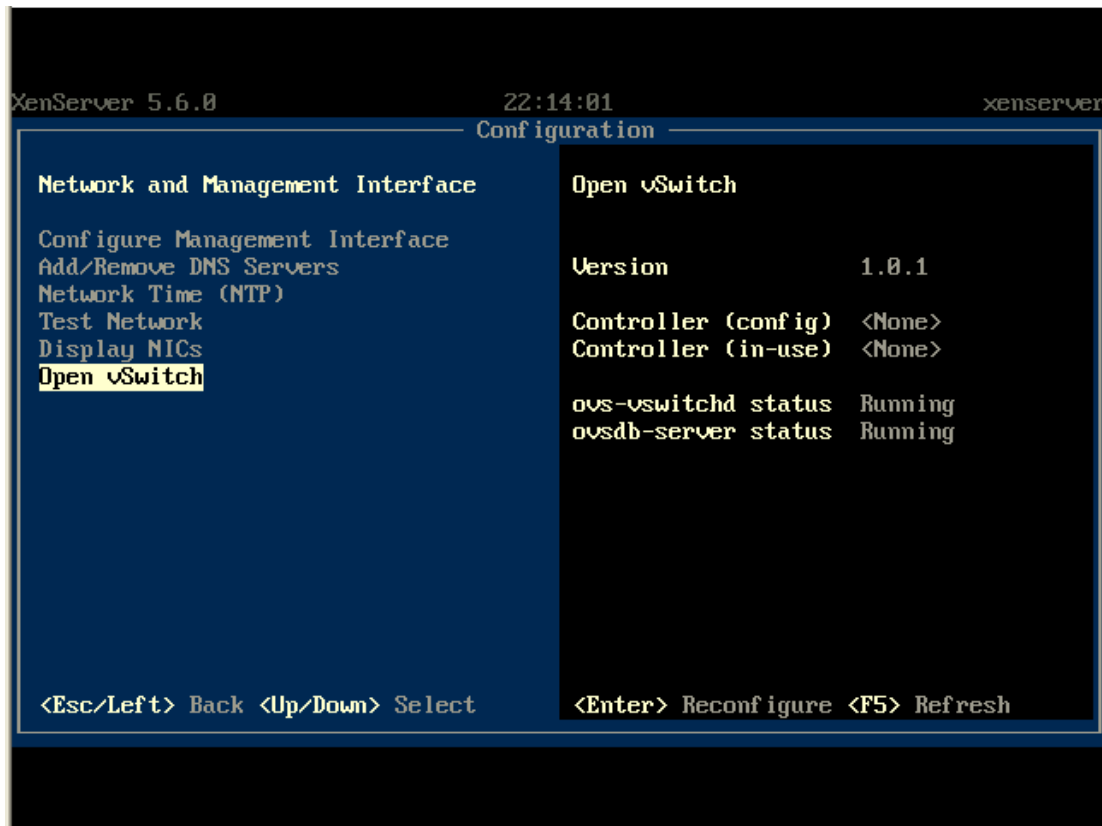
Server configuration	Hardware configuration: Intel® Core™2 Quad Q8200 @ 2.33MHZ 4GB RAM 500GB SATA HardDisk Software configuration: Citrix Xen Server 5.6 Replace Linux kernel Bridge Openvswitch 1.0.1
VM configuration	Hardware configuration: Intel® Core 2.4GHZ 512M RAM 20GB SCSI HardDisk 10/100 Ethernet Card Software configuration: Centos 5.4 Netperf 2.5.4
Remote host configuration	Hardware configuration: AMD Athlon™ 64X2 Dual Core Processor 4400+ 2G RAM 250GB IDE HardDisk 10/100 Ethernet Card Software configuration: Xen Center 5.6 SSH Client 2.9.3

Setup Environment and Installation

Install Xen Server

1. Insert Xen Server installation disk and boot up the system.
2. After the hardware detection and initialization is finished, use the default keyboard setting and choose OK to the next step.
3. The installation of Xen Server will be started to configure when “Welcome to XenServer” screen is shown. Choose OK to continue.
4. The next screen is Xen Server End User License Agreement (EULA), select “Accept EULA” to continue.
5. In this moment, system will ask for the Supplement disk to install later on. In order to install openvswitch, choose “Yes” to continue.
6. Select “Skip verification”.
7. Set up the root password.
8. In the test environment, the following IP configuration is used:
IP address: 192.168.0.157
Subnet mask: 255.255.255.0
Default Gateway: 192.168.0.1
9. Configure the DNS to 192.168.0.1 in the DNS configuration and click OK.
10. Choose “Asia/Hong Kong” as the default time zone.
11. Select “Manual time entry” for clock configuration.
12. After all configurations are finish, a message is displayed that the installation is ready to proceed. Select Install XenServer to continue.
During the installation, a progress bar is shown the installation status.
13. When the installation is finished, eject the installation disk and reboot the system.

Install and replace Linux Kernel Bridge by OVS



Download the OVS source file from OVS official site.

1. Download the corresponding DDK VM image from Xen server official site.
2. Import the DDK VM image into the Xen server.
3. In the DDK VM, compile the OVS source to rpm file.

```
VERSION=<Open vSwitch version>
XENKERNEL=<Xen kernel version>
cd /tmp
tar xzf /usr/src/redhat/SOURCES/openvswitch-$VERSION.tar.gz
rpmbuild \
    -D "openvswitch_version $VERSION" \
    -D "xen_version $XENKERNEL" \
    -bb openvswitch-$VERSION/xenserver/openvswitch-xen.spec
```

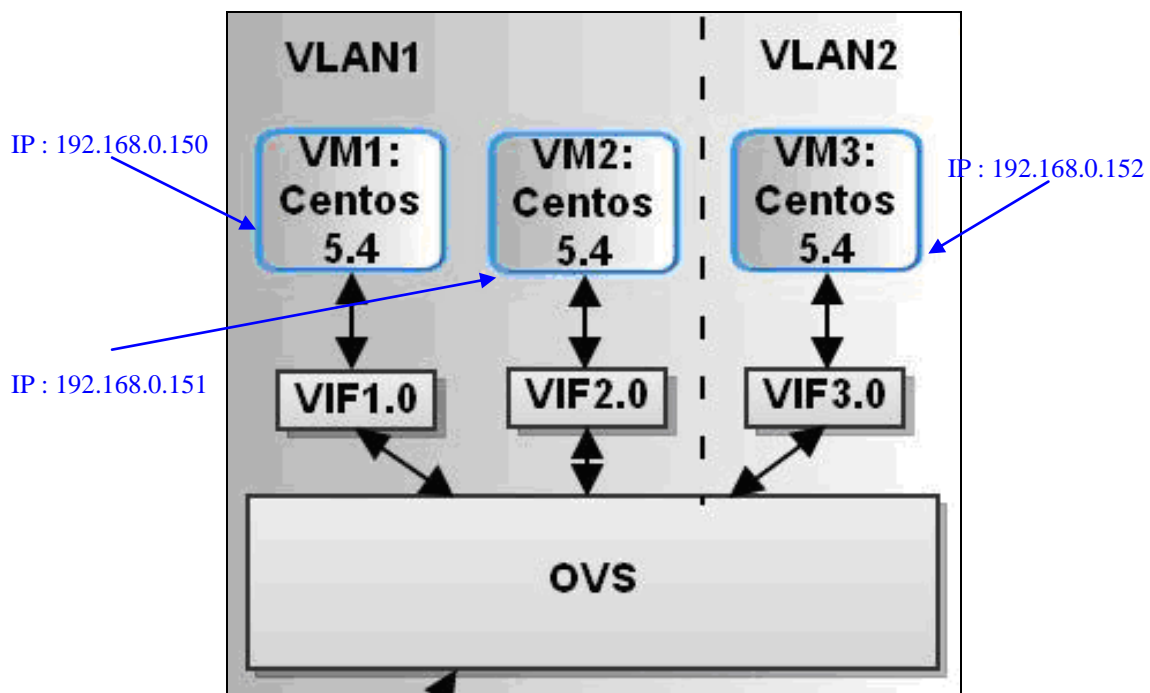
4. Transfer the OVS rpm file and install it in Xen server
5. Reboot the system after completed OVS installation.
6. In Xen server local console, enable the OVS switch function.

Test 1 - Security on VLAN:

In the actual network environment, there is a security problem if two clients from different networks but connect to the same port in one single switch they will transfer the layer2 (L2) ARP data frame from each other. This approach is not a well network setting. When it failed to isolate the ARP data frame, in terms of diversify to network package into different network. Besides, a VM server have a similar approach as the example above, which has communicated all VM inside and have one single physical interface. Therefore, it should be questioned that if VM have the same problem.

When the switch can group the subnet and isolates the domain broadcast package from each other, this approach is called Virtual LAN (VLAN). For the security concern, the ARP request and respond package should not been transferred to the VLAN divided network in OVS. Thus, OVS should isolate the ARP protocol from two different VLAN. In this test, we have created a network environment as the table below shown and try to figure out that may the ARP request and respond can be transferred to the opposite VLAN.

VM (Virtual Machine)	VIF (Virtual Interface)	VLAN Tag
VM1	vif1.0	1
VM2	vif2.0	1
VM3	vif3.0	2

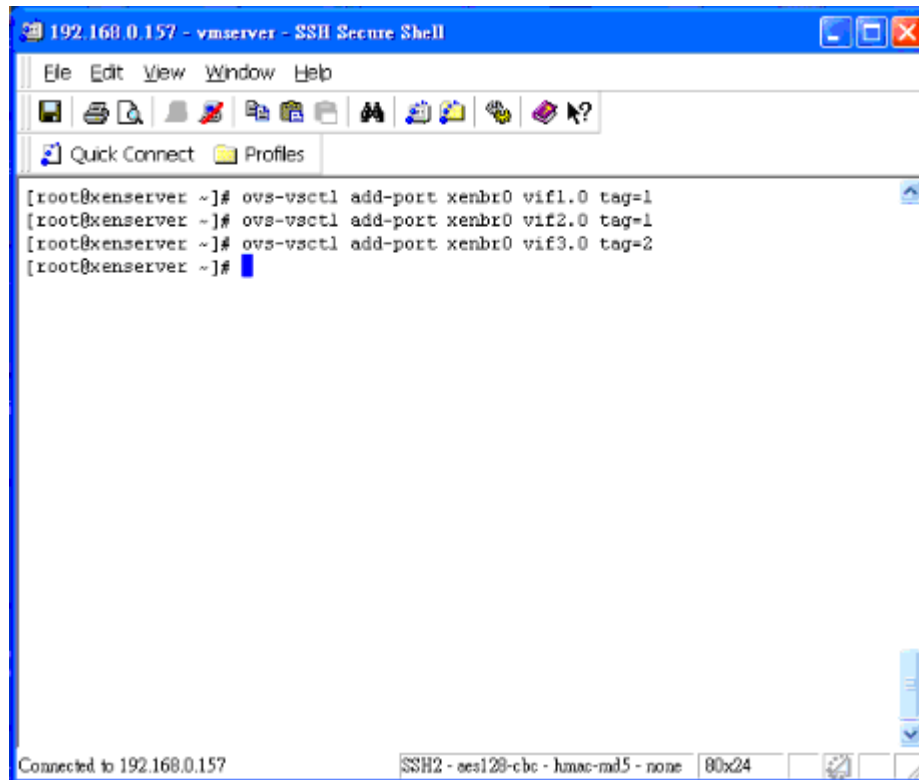


Cleanup and checking before testing:

To divide the VM from different VLAN, we have added the VIFs to the corresponding VLAN as following (figure 3):

ovs-vsctl add-port <bridge name> <vif name> tag=<tag number>

Figure 3 – Xen Server IP: 192.168.0.157

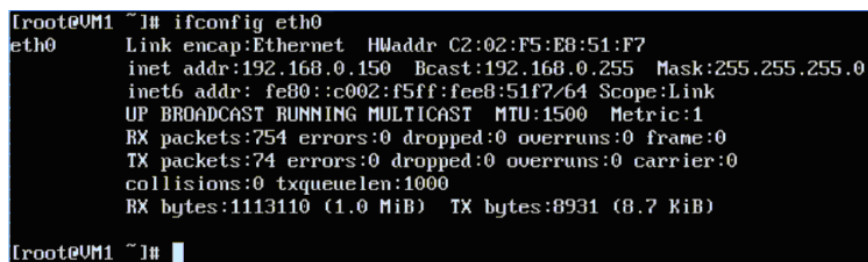


```
[root@xenserver ~]# ovs-vsctl add-port xenbr0 vif1.0 tag=1
[root@xenserver ~]# ovs-vsctl add-port xenbr0 vif2.0 tag=1
[root@xenserver ~]# ovs-vsctl add-port xenbr0 vif3.0 tag=2
[root@xenserver ~]#
```

In this test, the VMs has the following IP configuration:

VM1 IP configure(figure 4):

Figure 4 – VM1 IP: 192.168.0.150



```
[root@VM1 ~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr C2:02:F5:E8:51:F7
          inet addr:192.168.0.150  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::c002:f5ff:fee8:51f7/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:754 errors:0 dropped:0 overruns:0 frame:0
          TX packets:74 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1113110 (1.0 MiB)  TX bytes:8931 (8.7 KiB)

[root@VM1 ~]#
```

VM2 IP configuration (figure 5):

Figure 5 – VM2 IP: 192.168.0.151

```
[root@VM2 ~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr FA:F7:B3:F5:D4:0C
          inet addr:192.168.0.151  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::f8f7:b3ff:fe5:d40c/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:753 errors:0 dropped:0 overruns:0 frame:0
          TX packets:71 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1112559 (1.0 MiB)  TX bytes:8805 (8.5 KiB)

[root@VM2 ~]#
```

VM3 IP configuration (figure 6):

Figure 6 – VM3 IP: 192.168.0.152

```
[root@VM3 ~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 4A:CC:23:C4:84:F1
          inet addr:192.168.0.152  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::48cc:23ff:fec4:84f1/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:734 errors:0 dropped:0 overruns:0 frame:0
          TX packets:54 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1112619 (1.0 MiB)  TX bytes:8301 (8.1 KiB)

[root@VM3 ~]#
```

Test 1 – Security Test

To check the ARP table in VM2 and VM3, `arp -n` is used to list out the ARP table in both VMs. If there is an ARP record in the ARP table, the restarting virtual Ethernet card commands “`ifdown eth0`” and “`ifup eth0`” will be used to cleanup the ARP table. Apart from that, we also used “`tcpdump arp`” to monitor the incoming ARP request for both VMs (figure7, figure8).

Figure 7 – VM2 IP 192.168.0.151

```
Starting hpssd: [ OK ]
Starting sshd: [ OK ]
Starting cups: [ OK ]
Starting console mouse services: [ OK ]
Starting crond: [ OK ]
Starting xfs: [ OK ]
Starting anacron: [ OK ]
Starting atd: [ OK ]
Starting yum-updatesd: [ OK ]
Starting Avahi daemon... [ OK ]
Starting HAL daemon: [ OK ]
Starting smartd: [ OK ]

CentOS release 5.3 (Final)
Kernel 2.6.18-128.el5xen on an i686

VM2 login: root
Password:
Last login: Sat Nov 20 12:33:36 from 192.168.0.101
[root@VM2 ~]# arp -n
[root@VM2 ~]# tcpdump arp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
```

Figure 8 – VM3 IP 192.168.0.152

```
login: root
Password:
Last login: Sat Nov 13 04:14:41 from 192.168.0.101
[root@VM3 ~]# arp -n
[root@VM3 ~]# tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes

[1]+  Stopped                  tcpdump
[root@VM3 ~]# tcpdump -arp
tcpdump: p: No such file or directory
[root@VM3 ~]# tcpdump -h
tcpdump version 3.9.4
libpcap version 0.9.4
Usage: tcpdump [-aAdDefILnNDpqRStuUvxX] [-c count] [-C file_size]
             [-E algo:secret] [-F file] [-i interface] [-M secret]
             [-r file] [-s snaplen] [-T type] [-w file]
             [-W filecount] [-y datalinktype] [-Z user]
             [ expression ]
[root@VM3 ~]# tcpdump arp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
```


Since we had divided VM1 and VM3 into different network, VM1 tried to “ping” VM3 and looked for the status of ARP request package from VM1 to VM2 and VM3 (figure 9).

Figure 9 – VM1 IP 192.168.0.150

```
[root@VM1 ~]# ping 192.168.0.152
PING 192.168.0.152 (192.168.0.152) 56(84) bytes of data.
From 192.168.0.150 icmp_seq=1 Destination Host Unreachable
From 192.168.0.150 icmp_seq=2 Destination Host Unreachable
From 192.168.0.150 icmp_seq=3 Destination Host Unreachable
From 192.168.0.150 icmp_seq=4 Destination Host Unreachable
From 192.168.0.150 icmp_seq=5 Destination Host Unreachable
From 192.168.0.150 icmp_seq=6 Destination Host Unreachable
From 192.168.0.150 icmp_seq=7 Destination Host Unreachable
From 192.168.0.150 icmp_seq=8 Destination Host Unreachable
From 192.168.0.150 icmp_seq=9 Destination Host Unreachable
From 192.168.0.150 icmp_seq=10 Destination Host Unreachable
From 192.168.0.150 icmp_seq=11 Destination Host Unreachable
From 192.168.0.150 icmp_seq=12 Destination Host Unreachable
From 192.168.0.150 icmp_seq=13 Destination Host Unreachable
From 192.168.0.150 icmp_seq=14 Destination Host Unreachable
From 192.168.0.150 icmp_seq=15 Destination Host Unreachable
From 192.168.0.150 icmp_seq=16 Destination Host Unreachable
From 192.168.0.150 icmp_seq=17 Destination Host Unreachable
From 192.168.0.150 icmp_seq=18 Destination Host Unreachable

[1]+  Stopped                  ping 192.168.0.152
[root@VM1 ~]#
```

Even though VM1 does not “ping” VM2, VM2 still receive the ARP broadcast request because VM1 and VM2 are in the same VLAN (figure 10).

Besides, due to VM2 has been setup the default gateway in the IP configuration, so it can be seen in the monitor report of VM2 that VM2 also sent the ARP request to look for the MAC address of default gateway 192.168.0.1.

Figure 10 – VM2 IP 192.168.0.151

```
14:50:33.725686 arp who-has 192.168.0.152 tell 192.168.0.150
14:50:34.725748 arp who-has 192.168.0.152 tell 192.168.0.150
14:50:35.732762 arp who-has 192.168.0.152 tell 192.168.0.150
14:50:35.733793 arp who-has 192.168.0.1 tell 192.168.0.151
14:50:36.729876 arp who-has 192.168.0.152 tell 192.168.0.150
14:50:36.733857 arp who-has 192.168.0.1 tell 192.168.0.151
14:50:37.729949 arp who-has 192.168.0.152 tell 192.168.0.150
14:50:37.733919 arp who-has 192.168.0.1 tell 192.168.0.151
14:50:38.736957 arp who-has 192.168.0.152 tell 192.168.0.150
14:50:39.734064 arp who-has 192.168.0.152 tell 192.168.0.150
14:50:40.734123 arp who-has 192.168.0.152 tell 192.168.0.150
14:50:40.738109 arp who-has 192.168.0.1 tell 192.168.0.151
14:50:41.735735 arp who-has 192.168.0.1 tell 192.168.0.151
14:50:42.735812 arp who-has 192.168.0.1 tell 192.168.0.151
14:50:45.751988 arp who-has 192.168.0.1 tell 192.168.0.151
14:50:46.752062 arp who-has 192.168.0.1 tell 192.168.0.151
14:50:47.752125 arp who-has 192.168.0.1 tell 192.168.0.151
14:50:50.756302 arp who-has 192.168.0.1 tell 192.168.0.151
14:50:51.756361 arp who-has 192.168.0.1 tell 192.168.0.151
14:50:52.756424 arp who-has 192.168.0.1 tell 192.168.0.151
14:50:55.756612 arp who-has 192.168.0.1 tell 192.168.0.151
14:50:56.756674 arp who-has 192.168.0.1 tell 192.168.0.151
14:50:57.756737 arp who-has 192.168.0.1 tell 192.168.0.151
```

However, in VM3, it does not receive any ARP package in the separated VLAN (figure 11 – VM3 IP 192.168.0.152).

```
login: root
Password:
Last login: Sat Nov 13 04:14:41 from 192.168.0.101
[root@VM3 ~]# arp -n
[root@VM3 ~]# tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes

[1]+  Stopped                  tcpdump
[root@VM3 ~]# tcpdump -arp
tcpdump: p: No such file or directory
[root@VM3 ~]# tcpdump -h
tcpdump version 3.9.4
libpcap version 0.9.4
Usage: tcpdump [-aAdDefILnMOpqRStuUvxX] [-c count] [-C file_size ]
              [-E algo:secret ] [-F file ] [-i interface ] [-M secret ]
              [-r file ] [-s snaplen ] [-T type ] [-w file ]
              [-W filecount ] [-y datalinktype ] [-Z user ]
              [ expression ]
[root@VM3 ~]# tcpdump arp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
```

From the testing shown above, it can be seen that the OVS VLAN can isolate the VMs traffic between different groups of VLAN. Thus, in terms of security, OVS can isolate the ARP L2 packets in different VLAN to grantee the ARP request and respond packet should not been transferred to the VLAN divided network.

Test1 - Summary of Security Test

(Yes – receive traffic, No – cannot receive)

Virtual Machine	VM1	VM2	VM3
VM1	/	Yes	No
VM2	Yes	/	No
VM3	No	No	/

Test 2 - Performance Test:

To testing a network total performance, there are 5 standards that can measure and archive:

1. Availability
2. Response time
3. Network utilization
4. Network throughput
5. Network bandwidth capacity

In the above standards, availability measures the connectivity between the node in the network; response time measures how quick does the transferred package can do; network utilization measures the time that the network is in use; network throughput measures the remaining bandwidth between two nodes are transferring data; and network bandwidth capacity measures the maximum bandwidth between two nodes are transferring data.

The reason that carries out a performance test of OVS is that network utilization and availability of a VM that provides system services is very important, whenever OVS as role of the middle point in the virtual environment and virtual equipment in the Virtualization server. Therefore, to prepare a performance test, there are some setting need to setup in the OVS and the VMs.

In the performance test there are four types of network sockets that will test:

1. TCP stream
2. UDP stream
3. TCP Connect/Request/Respond (TCP CRR)
4. UDP Request/Respond (UDP RR)

The reason that chooses those four network sockets is that those four network sockets are most common in use in the actual network environment for client server communication. They are commonly appearing in the connection-oriented service like video streaming technology or HTTP protocol.

Testing setup:

In this test, VM1 will act as client to send the package to the server VM2, and VM2 will calculate this package transaction performance and send the data back to client.

In VM1 and VM2, “netperf 2.4.5” has been installed to create the client-server environment.

```
# cd ~  
# wget ftp://ftp.netperf.org/netperf/netperf-2.4.5.tar.bz2  
# tar jxf ./netperf-2.4.5.tar.bz2  
# cd netperf-2.4.5  
# ./configure  
# make  
# make install
```

To ensure the same condition of VM1 and VM2, setup QoS by using the following command to limit the ingress policing rate and burst in OVS for both vif1.0 and vif2.0.

```
# ovs-vsctl set Interface vif1.0 ingress_policing_rate=10240  
# ovs-vsctl set Interface vif1.0 ingress_policing_burst=1024  
# ovs-vsctl set Interface vif2.0 ingress_policing_rate=10240  
# ovs-vsctl set Interface vif2.0 ingress_policing_burst=1024
```

After that, in VM2, start up the netserver to prepare receiving client package.
netserver

The OVS performance will be carry out like this: a stress test will try to transfer packets as much as it can within 60 seconds. It will take several rounds and increase the concurrent stream test with the fixed message size 10000 bytes that sent for each test, e.g.: 10, 20, 30 and so on. Therefore, to simplify the testing, we had coded a shell script to run netperf and collect the data for each

testing socket scope.

netscript:

```
#!/bin/bash

time=60
server=192.168.0.151

if test $# -eq 0; then
    echo "no data provide"
#   $1=socket type, $2= loop size
    echo "usage: netscript <socket type> <loop size>"
    exit 0
fi

for (( i=1; i<=$2; i=i+1 ))
do
    ts=$(date +%H%M%S)
    netperf -t $1 -H $server -l $time -- -m 10000 >> netreport$i-$time-$2-$ts &
done
```

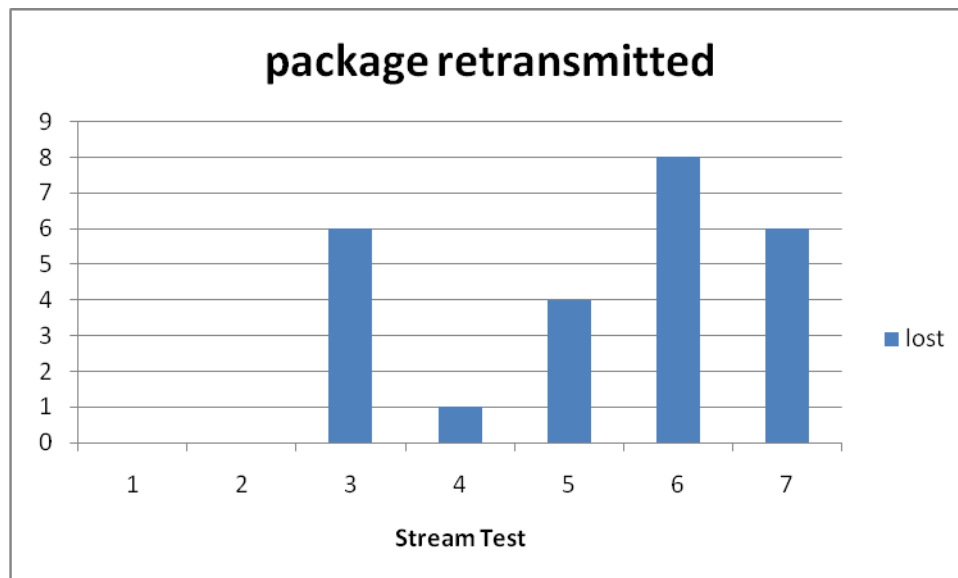
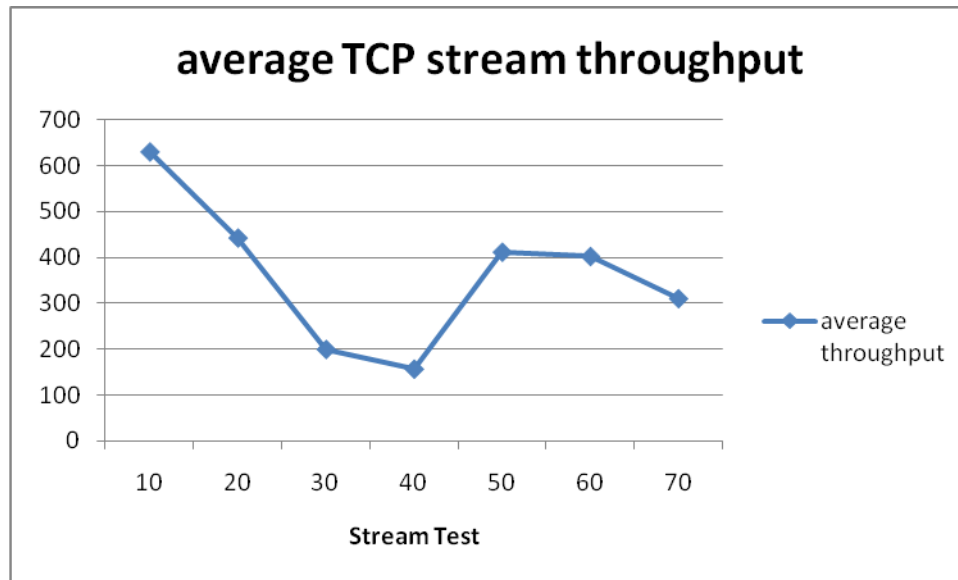
netconsolidate:

```
#!/bin/bash

for (( node=10; node<=70; node=node+10))
do
    for (( i=1; i<=$node; i=i+1 ))
    do
        mv ./netreport$i-60-$node* ./node
        nl ./node/netreport$i-60-$node* | sed '1,6d' | awk -F" " '{ print $6}' >> ./node/result.txt
    done
done
```

TCP stream

Stream Test	10	20	30	40	50	60	70
Average throughput	631.17	442.96	199.3	156.32	411.97	402.51	310.38
Lost	0	0	6	1	4	8	6

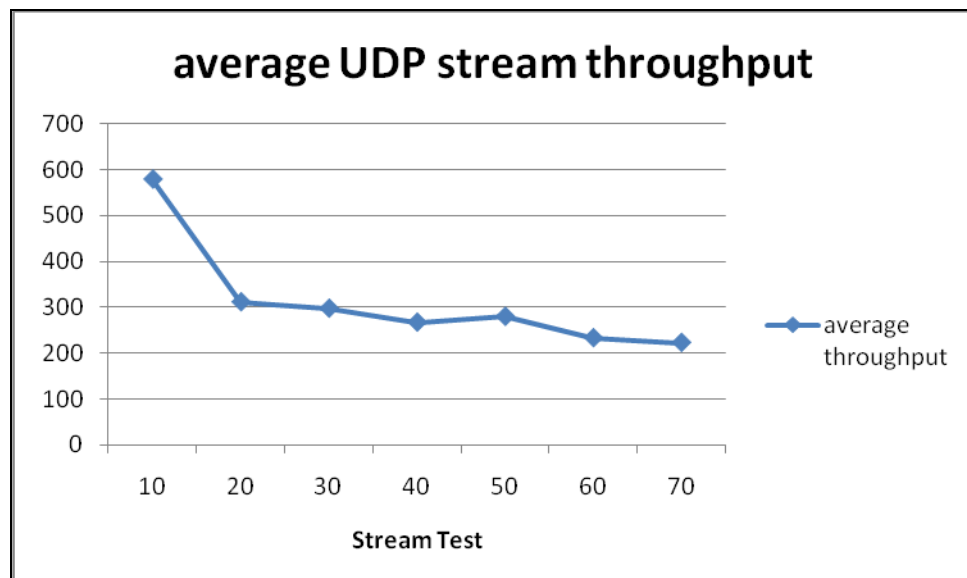


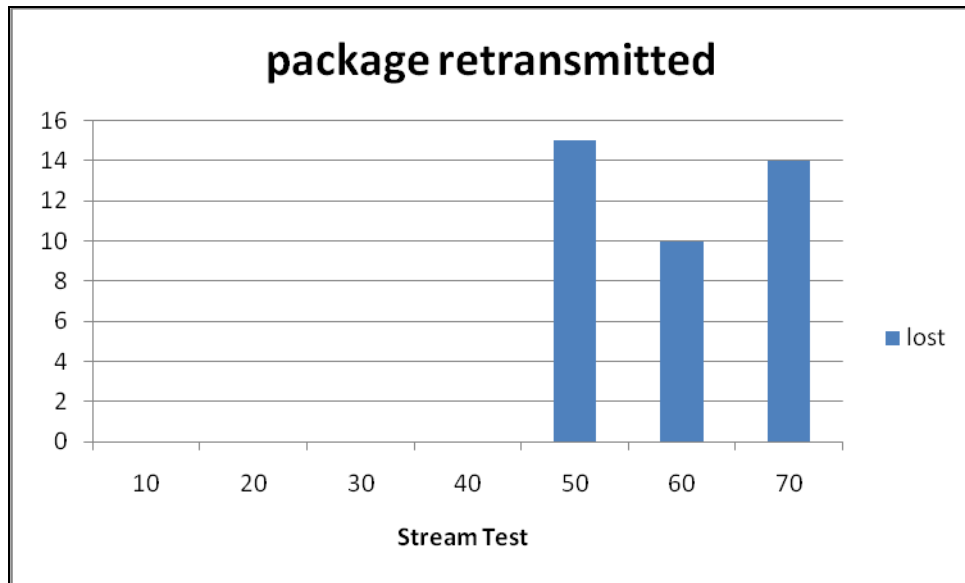
In TCP stream test, the average throughput is decreased from 631.17 to 156.32 when the package sent is increased from 10 to 40 stream tests in 60 seconds. However, the average throughput has raise up to 411.97, when 50 stream tests in the 60 seconds. Then, the average throughput is decrease steadily to 310.38, when the 60 seconds test has performed 70 times. Besides, the package lost

has been occurred when 30 stream tests or above has sent. There are 6 lost out of 30 stream test that cannot receive from server. The least lost of the above test is 40 stream tests that only have 1 lost. 50, 60 and 70 stream tests within 60 seconds have around 10% to 15% lost, which are 4, 8 and 6 lost respectively, during the transmission.

UDP stream

Stream Test	10	20	30	40	50	60	70
Average throughput	578.24	310.9	296.94	266.51	279.26	232.66	222.33
Lost	0	0	0	0	15	10	14

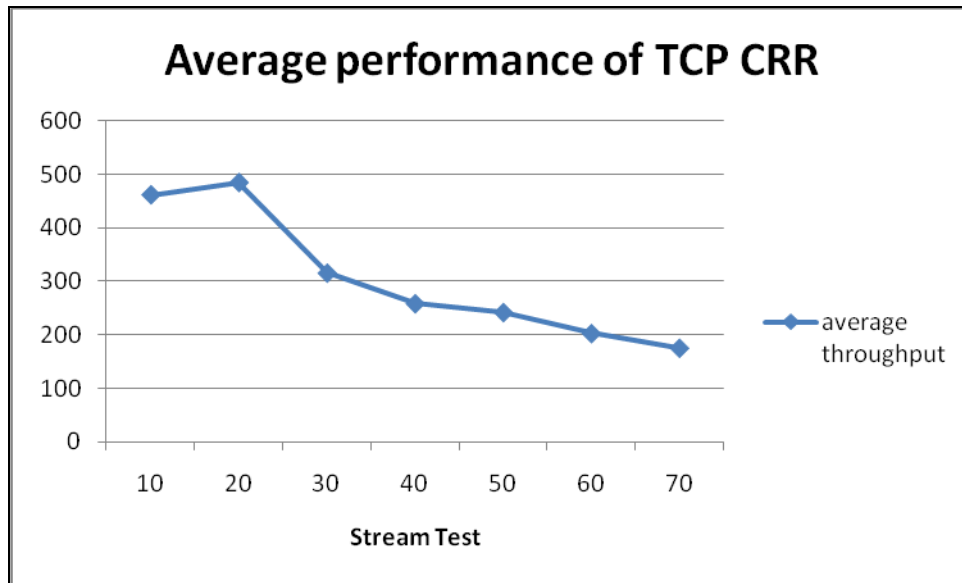




The stateless connection UDP stream has a rapidly fall of average throughput from 578.24 of 10 stream tests to 310.9 of 20 stream tests in the 60 seconds. Then the average throughput has decreased gently from 310.9 of 20 stream tests to 222.33 of 70 stream tests. On the other hand, the package retransmission has occurred when there are 50 stream tests or above has been sent. 50, 60 and 70 stream tests within 60 seconds has more than 15% lost, which are 15, 10 and 14 tests respectively, during the transmission.

TCP Connect/Request/Respond

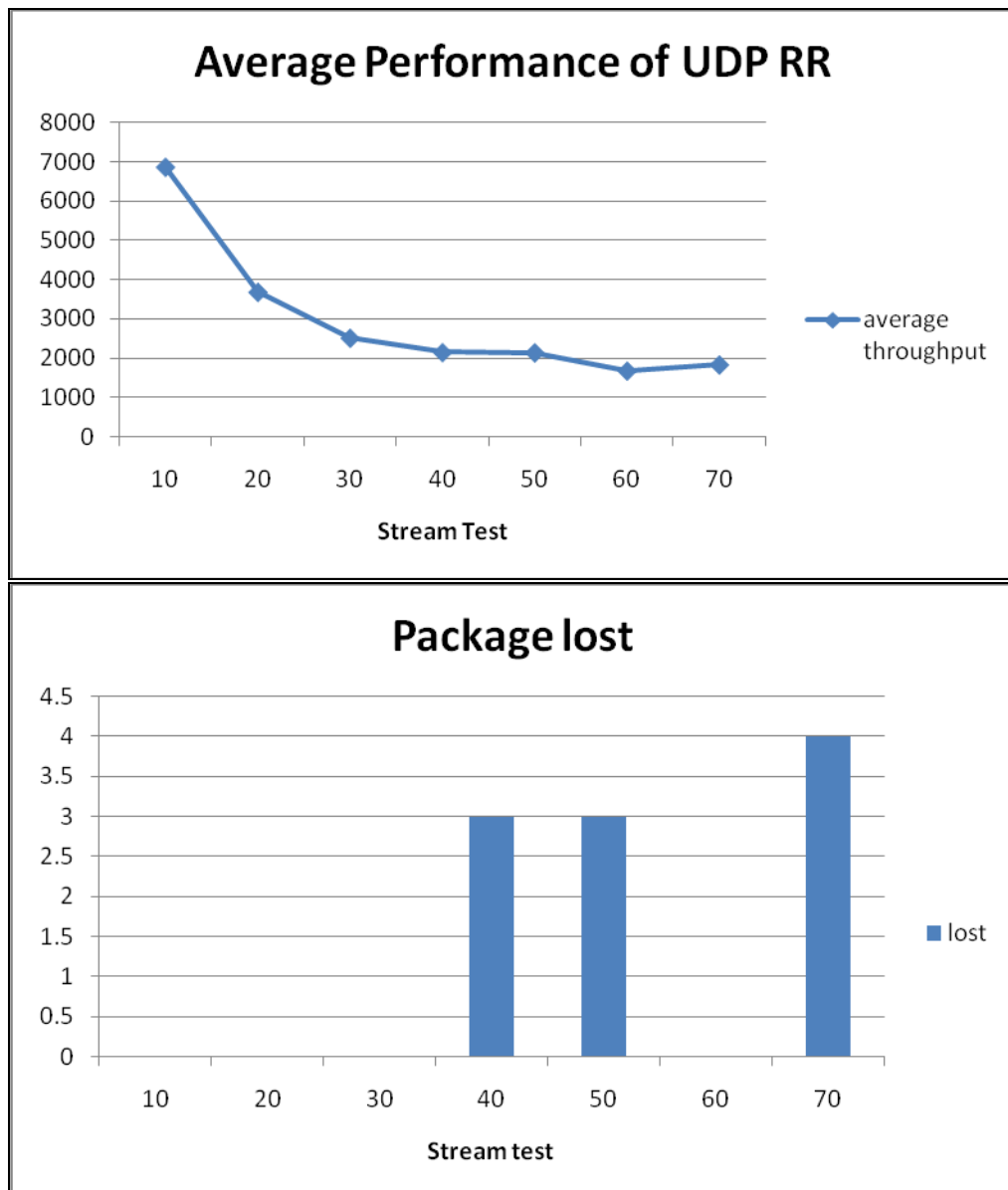
Stream Test	10	20	30	40	50	60	70
Transaction rate	461.36	484.16	315.43	258.44	242.04	202.85	175.55



The average transaction rate of the most stateful connection TCP CRR has increased from 461.36 of 10 stream tests to 484.16 of 20 stream tests. For the next 60 seconds test, the transaction rate has decreased slowly from 484.16 of 20 stream tests to 258.44 of 40 stream tests and decrease gently from 258.44 of 40 stream tests to 175.55 of 70 stream tests. By the way, the TCP CRR does not have any lost in any 60 seconds test.

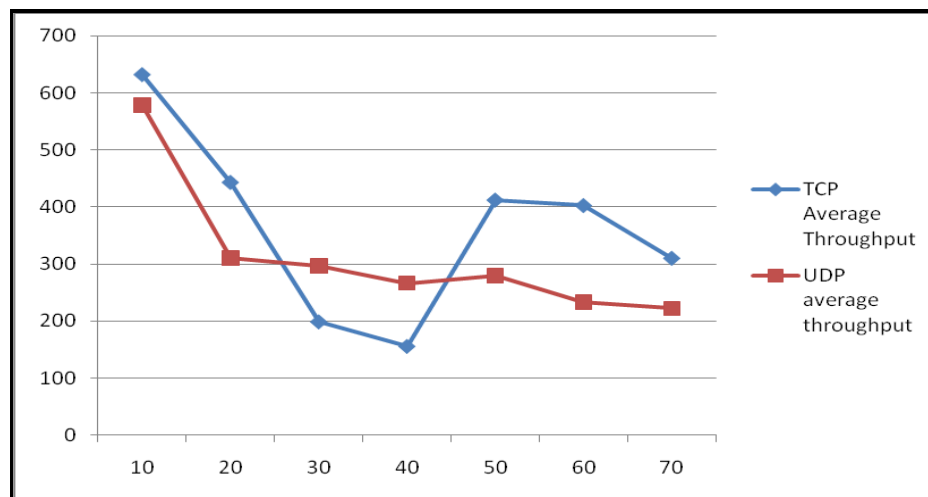
UDP Request/Respond

Stream Test	10	20	30	40	50	60	70
Transaction Rate	6856.13	3679.66	2514.36	2147.78	2141.96	1680.16	1835.07
lost	0	0	0	3	3	0	4

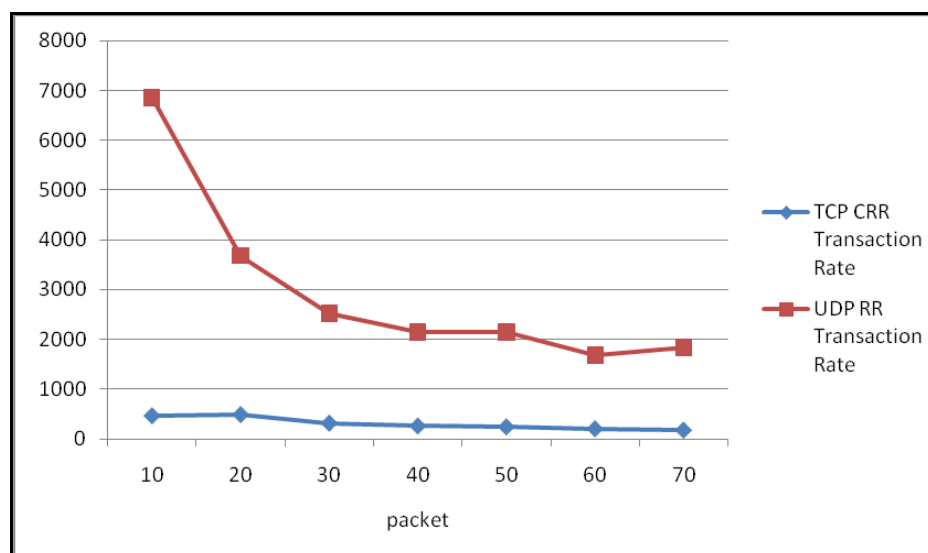


The UDP RR has a rapidly fall of average transaction rate from 6858.13 of 10 stream tests to 2514.36 of 30 stream tests. Apart from that, the average transaction rate has decreased gently from 2514.36 of 30 stream tests to 1835.07 of 70 stream test. On the other hand, the package retransmission has occurred when there are 40 stream tests or above has been sent within 60 seconds. 40, 50 and 70 streams tests test within 60 seconds has only 5% to 7% lost, which are 3, 3 and 4 lost respectively, during the transmission.

Test2 - Performance Test Summary



To compare with TCP stream and UDP stream, TCP stream performs more inconsistency than UDP stream. It can be seen from the average throughput of 20 stream tests to 50 stream tests, TCP is up and down between 446.92 and 156.32. However, UDP is more stable around 300 average throughputs. This contract has stopped and become stable around 400 to 300 average throughputs of TCP stream and around 230 to 220 of UDP stream. Moreover, even though UDP stream perform more consistency than TCP stream, TCP stream has lesser package lost than UDP stream. For each test, TCP stream has around 2% (1 lost out of 40) to 13% (8 lost out of 60) lost and UDP has higher packet lost around 16% (10 lost out of 60) to 28% (15 lost out of 50).



Rather than the comparison between TCP stream and UDP stream, the average transaction rate between TCP CRR and UDP RR have a great distance of performance. For TCP CRR, it performs very steady between 484.16 and 175.55.

However, UDP RR performs a huge falling from 6856.134 to 1680.163. Besides, TCP CRR is the only network socket that does not have any packet lost in the test. And UDP RR has a small lost between 5.7% (3 out of 40) and 7.5% (4 out of 70).

Question: where is the bottleneck? OVS? Or the netperf receive server?

From the above analyzing, it can be seen that there are 3 out 4 tests have shown that the lost has occurred during the test. It is an interesting part to find out where to makes the problems occur. The clue may discover in the returned package from the VM1.

*establish control: are you sure there is a netserver listening on 192.168.0.151 at port 12865?
establish_control could not establish the control connection from 0.0.0.0 port 0 address family
AF_UNSPEC to 192.168.0.151 port 12865 address family AF_UNSPEC*

The error above has shown that the client side VM1 cannot receive the testing packet, so there are two point of failure may occurred during the transmission:

1. OVS is the bottleneck in the transmission and it cannot send and receive the packets from both VM1 and VM2.
2. VM2 is the bottleneck in the transmission and it cannot receive the packets from VM1 via OVS.

However, to analyze the bottleneck in this case, OVS lack of a detail logs about those error. Apart from that, we lacks of time to study the third party tool appeared in OVS and VM2 to grab the data status that we need. Thus, it's hardly to define the point of failure between OVS and VM2, since time is the critical issue for us to study the tool to analyze the problems.

Difficulties from testing

There are several problems, when we setup the testing environment.

- First of all, it is not a good idea in all virtualization system that creating a VM server inside a VM because those VM are bad performance used this approach. Therefore, to grantee the performance for the VMs that will similar with the physical machines, we had to buy a middle class physical machine to act as the VM server.
- OVS official website has included a briefly, but not user friendly installation and configuration support to the user – OVS official website just provide a roughly and narrow installation guide and three beside case called “configuration cookbook”. Besides, the information about OVS installation is hard to find out on the Internet. One of the reasons was that OVS, or Virtualization, are both new to the network administrator. Thus, without any detail documents for installation, most of the hints is required to ask for help from OVS technicians.
- In our first testing approach, we decided to use Centos 5.5 as our Virtualization Server and KVM as our Virtualization Tools. However, we have failed in compiling the configuration files of OVS in Centos 5.5, because of lack of support document that mentioned above. After asked to the OVS technicians for a week and doing research about other method to create a Virtualization network with the OVS, we had change to use Xen server as our base VM environment to avoid the module installation problem.
- The most difficult part in installing OVS in Xen server is how to create an rpm package that is suitable for Xen server. The answer is that we need to download the DDK VM image from Xen official website first and create a VM from this image for compiling all the necessary rpm in this VM to Xen server.
- The testing scopes are hard to define after we have created the testing environment, because we have wasted much time to discover which testing data is useful to represent the security and performance issue. With the outdated support and information on the Internet, we are hardly to define which parameters of “netperf” are suitable for us to work out the testing in performance.

Team's work allocated

Name	Responsibility
Ng Chi Hong (08569944G)	Setup Environment for test Server Installation Testing
Leung Chun Yue (08565078G)	Testing Search and clarify information Consolidate the information
Leung Kai Yin (08564440G)	Setup Test Plan and scope Search and clarify information Consolidate the information

Conclusion

From this project, virtual switching will become more popular in the world. Based on the analysis of result with using Virtual vSwitch, it can be used to replace the physical switch. Actually, we learned how to build virtual network architecture, prepare setup facility, reading specification and analysis the technical issue. It is concluded from the result for the following:

Open vSwitch can separate the different subnet by VLAN. It will be secure to setup different networks. It is well management and controls the traffic, so it can preserve advantage of performance.

Future Trend

There are totally different between concepts of traditional networking and virtual networking. From nowadays, it is virtualized world. The physical devices will be replaced by virtual devices. The mainly developments of virtualization will be on virtual switching. The new type of virtual switch such as OVS which is called distributed virtual switch. It provides cross-server bridging and underlying server architecture transparent. It make much simpler. In addition, it can work under cloud environment to support the various environments in which hypervisors can coexist.

References

- <http://openvswitch.org/>
- <http://wiki.answers.com/>
- <http://www.ibm.com/developerworks/linux/library/l-virtual-networking/>
- <http://www.netperf.org/svn/netperf2/tags/netperf-2.4.5/doc/netperf.html>
- http://www.cisco.com/en/US/products/hw/switches/ps708/products_white_paper09186a008013159f.shtml