1 Overview

This document introduces the combination of Quagga with Mininet(SDN emulator). Quagga is an application which can make a Linux system a software router, due to its feature of application we can integrate Quagga into Mininet and make Mininet able to emulate traditional routers which are non-SDN components. This integration extends the functionality of our SDN testbed to support more traditional network experiments than only SDN experiments.

This documents mainly consists of two parts, the first part is focused on the installation and configuration of Quagga on Linux system and run ping tests to check whether Quagga works well with other network components. In the second part, we designed different experiment topologies and conducted in different scenarios to prove the good performance with Quagga integrated.

2 Quagga-Mininet Install and Configuration

2.1 Quagga Install

2.1.1 Download Quagga:

http://download.savannah.gnu.org/releases/quagga/quagga-1.2.1.tar.gz

2.1.2 Unzip and Configure:

$ ./configure –enable-vtysh –enable-user=root –enable-group=root –enable-vty-group=root

If GNU awk is required:

$ sudo apt-get install gawk

If libreadline is required:

$ sudo apt-get install libreadline6 libreadline6-dev

If libcares is required:

$ sudo apt-get install libc-ares-dev

2.1.3 Make:

$ make

If aclocal-1.15 is missing:

$ sudo apt-get install automake

If makeinfo is missing:

$ sudo apt-get install texinfo

2.1.4 Make install:

$ sudo make install

2.1.5 Make new file, copy zebra.conf.sample to zebra.conf:

$ cd /usr/local/etc

$ sudo cp zebra.conf.sample zebra.conf

2.1.6 Quagga(Zebra) Service Start:

$ sudo zebra -d

If zebra: error while loading shared libraries: libzebra.so.1: cannot open shared object file: No such file or directory

$ cd /usr/local/lib

$ sudo cp libzebra.* /lib

$ sudo rm libzebra.*

2.1.7 Connect to zebra using telnet(password zebra):

$ telnet localhost 2601

2.2 Ping Test on OSPF Experiment (One Mininet)

2.2.1 Experiment Topo:

2.2.2 Create Zebra Configuration File for router r1 and r2:

$ cd /usr/local/etc

$ cp zebra.conf.sample r1zebra.conf

$ cp zebra.conf.sample r2zebra.conf

2.2.3 Create OSPF Configuration File for router r1 and r2:

$ cd /usr/local/etc

$ cp ospfd.conf r1ospfd.conf

$ cp ospfd.conf r2ospfd.conf

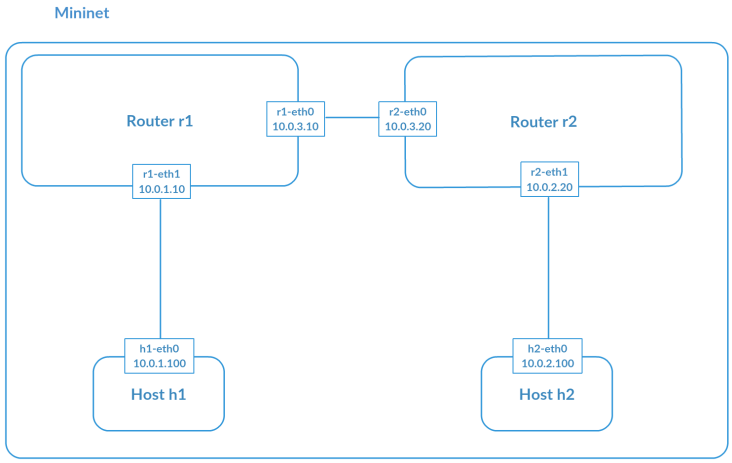


Figure 1: Ping Test on OSPF Experiment (One Mininet)

2.2.4 Edit OSPF configuration File:

Edit r1ospfd.conf

```
hostname r1_ospfd
password 123
enable password 123

router ospf
  ospf router-id 10.0.3.10
  network 10.0.3.0/24 area 0
  network 10.0.1.0/24 area 0
debug ospf event
log file /usr/local/etc/r1ospfd.log
```

Edit r2ospfd.conf

```
hostname r2_ospfd
password 123
enable password 123

router ospf
  ospf router-id 10.0.3.20
  network 10.0.3.0/24 area 0
  network 10.0.2.0/24 area 0
debug ospf event
log file /usr/local/etc/r2ospfd.log
```

2.2.5 Start Mininet Script:

$ sudo python QuaggaOSPF.py

2.2.6 Ping test:

mininet> pingall

It may take a while(40s) to add route.

2.3 Ping Test on OSPF Experiment (Hybrid)

2.3.1 Experiment Topo:

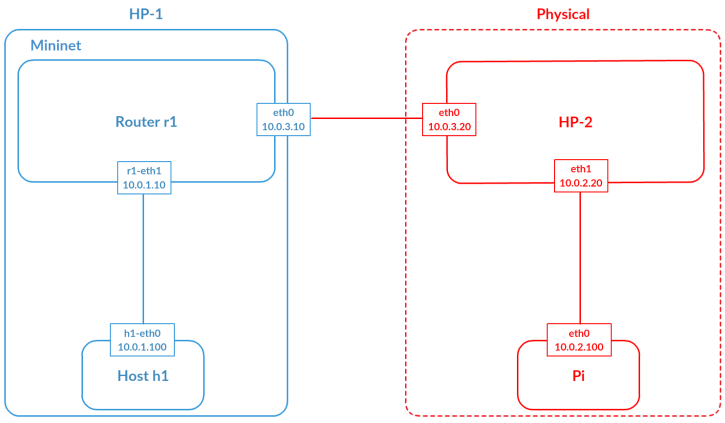


Figure 2: Ping Test on OSPF Experiment (Hybrid)

2.3.2 Run Mininet Script on PC1:

$ sudo python QuaggaOSPF (hybrid).py

2.3.3 Copy Zebra Configuration File on PC2:

$ cp zebra.conf.sample zebra.conf

2.3.4 Copy and Edit OSPF Configuration File on PC2:

$ cd /usr/local/etc

$ cp ospfd.conf.sample ospfd.conf

$ nano ospfd.conf

```
hostname hp2_ospfd
password 123
enable password 123

router ospf
 ospf router-id 10.0.3.20
 network 10.0.3.0/24 area 0
 network 10.0.2.0/24 area 0
```

2.3.5 Run Zebra and OSPF on PC2:

$ sudo zebra -d

$ sudo ospfd -d

2.3.6 Configure IP address and gateway on Pi:

$ sudo ifconfig eth0 10.0.2.100/24

$ sudo route add default gw 10.0.2.20

2.3.7 Ping Test:

mininet> h1 ping 10.0.2.100

It may take a while(40s) to add route.

2.4 Ping Test on SDN and Non-SDN Combination Experiment

2.4.1 Experiment Topo:

2.4.2 Edit OSPF Configuration Files:

$ cd /usr/local/etc

$ sudo nano r1ospfd.conf

```
hostname r1_ospfd
password 123
enable password 123

router ospf
  ospf router-id 10.0.3.10
  network 10.0.3.0/24 area 0
  network 10.0.1.0/24 area 0
  network 10.0.4.0/24 area 0
debug ospf event
log file /usr/local/etc/r1ospfd.log
```

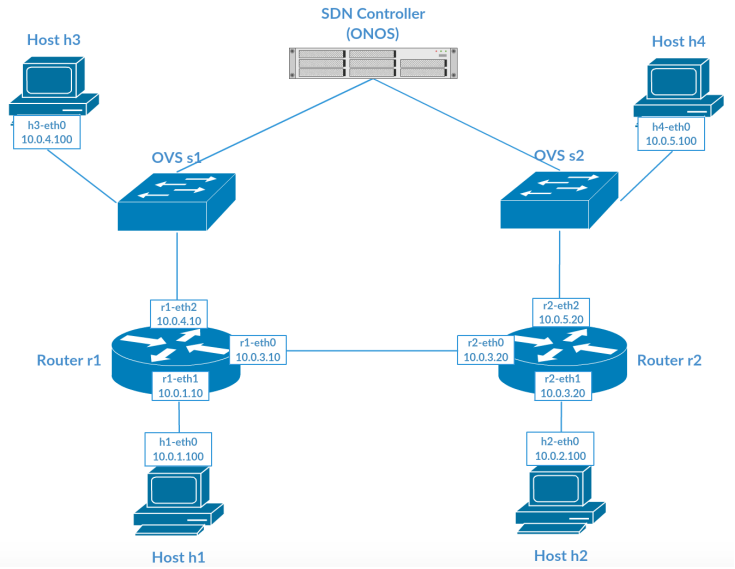


Figure 3: Ping Test on SDN and Non-SDN Combination Experiment

$ sudo nano r2ospfd.conf

```
hostname r2_ospfd
password 123
enable password 123

router ospf
  ospf router-id 10.0.3.20
  network 10.0.3.0/24 area 0
  network 10.0.2.0/24 area 0
  network 10.0.5.0/24 area 0
debug ospf event
log file /usr/local/etc/r2ospfd.log
```

2.4.3 Run Mininet Script:

$ sudo python QuaggaOSPF (SDN+NONSDN).py

2.4.4 Ping Test:

mininet> pingall

It may take a while(40s) to add route.

2.5 Trouble Shooting

2.5.1 OSPF service wait a long time before adding route.

3 Experiments and Performance Evaluation

We designed topologies based on our previous testbed validation. The new coponent is software router which make virtual host in mininet a router. In our performance evaluation, all software routers are running OSPF routing protocols and we want to prove that our testbed can maintain good performance when adding non-SDN components into our hardware-in-the-loop SDN testbed.

3.1 One Mininet

3.1.1 Two-way Topology(OneMininet):

Two-way test is the basic experiment to validate the fidelity of our Mininet-Quagga testbed. We conduct a two-way perf test on a simple topology to check the impact of non-SDN components (Quagga router) in SDN network.

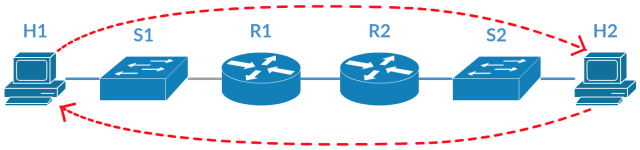
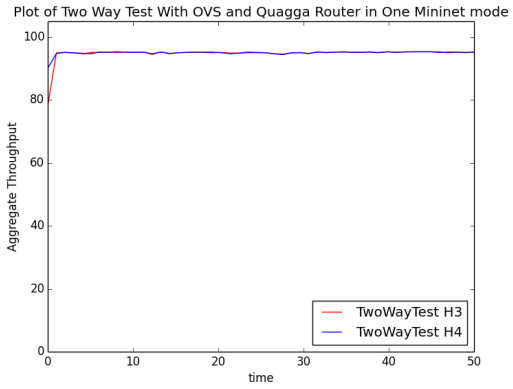


Figure 4: Two-way Topology(OneMininet)



From the result we can see both connections have an average throughput above 90 Mbits/sec.

3.1.2 Fork-in Topology:

We did fork-in test to evaluate the in-boud traffic queuing processing performance of Quagga routers.

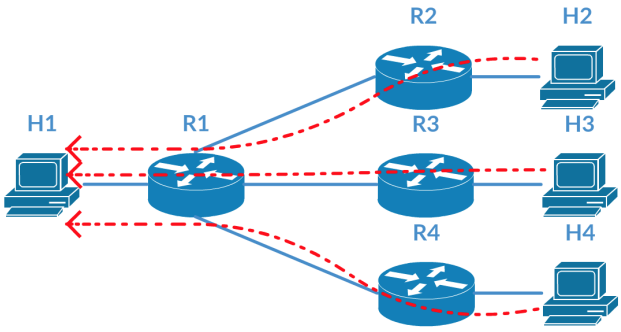
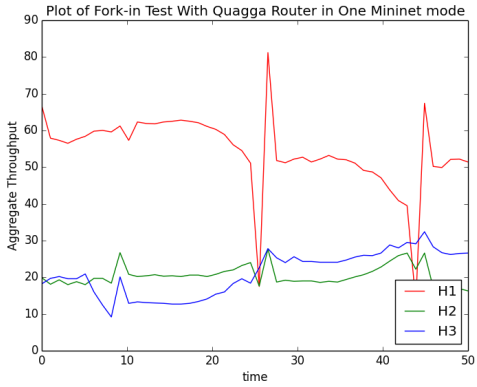


Figure 5: Fork-in Topology



3.1.3 Fork-out Topology:

Fork-out test aims to check whether Quagga router can make out-boud traffic get a fair share of bandwdith resources.

3.2 Two Mininet

3.2.1 Two-way Topology(Two Mininet):

We put two-way test components in two mininet by spliting the one mininet topology symmetrically. The goal is to prove a substitution of virtual link into physical link will not make performance down.

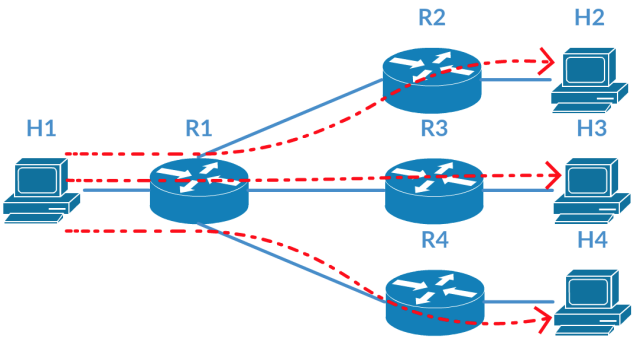
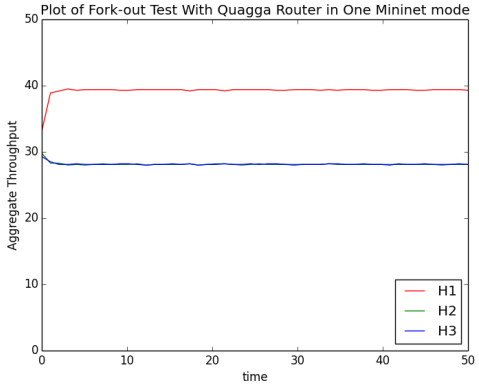


Figure 6: Fork-out Topology



3.3 Hybrid

3.3.1 Two-way Topology(Add Pica8):

There are 2 aspects that need to be evaluated in Hybrid mode: the physical port and Pica8 switch. By introducing non-SDN components into SDN network topology, we made our testbed capable of emulating traditional network experiment other than SDN network experiment. With both virtual and physical components inside, we have maintain the fidelity as well as scalability and reproducibility.

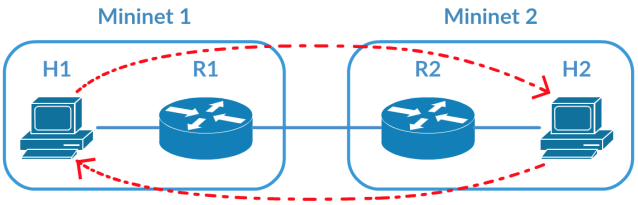


Figure 7: Two-way Topology(Two Mininet)

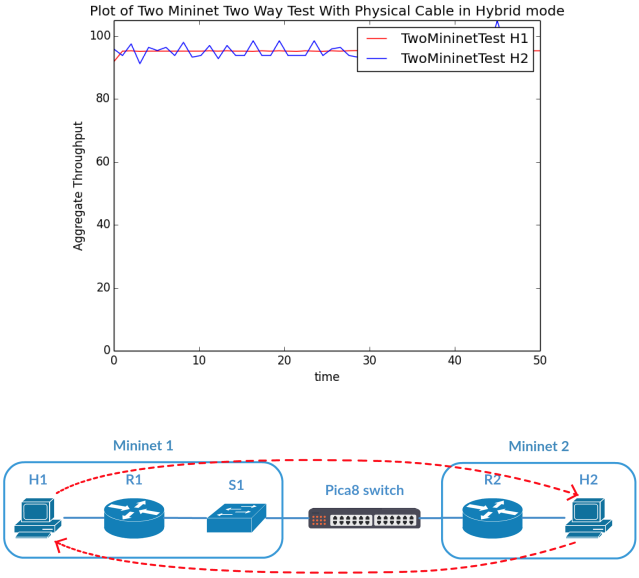


Figure 8: Two-way Topology(Add Pica8)