# H.264 SVC performance evaluation over SDN (2)

**[Goal]**

Based on "Using Bellman-Ford to find a shortest path (version 2)" and myEvalSVC-Mininet, I re-write some tools so that users can create any topology and do the H.264/SVC performance evaluations. Moreover, metrics, such as packet loss rate, packet end-to-end delay, and throughput, can be obtained.

[Tools]

1. l2_bellmardford.py (put this file under /pox/ext)
2. mystg_svc.c (H.264 sender: This version can generate the sending packet trace, i.e. sender_trace.txt)
3. myrtg_svc.c (H.264 receiver: This version can generate the receiving packet trace, i.e received_trace.txt)
4. measure-throughput.pl (for measuring the throughput)

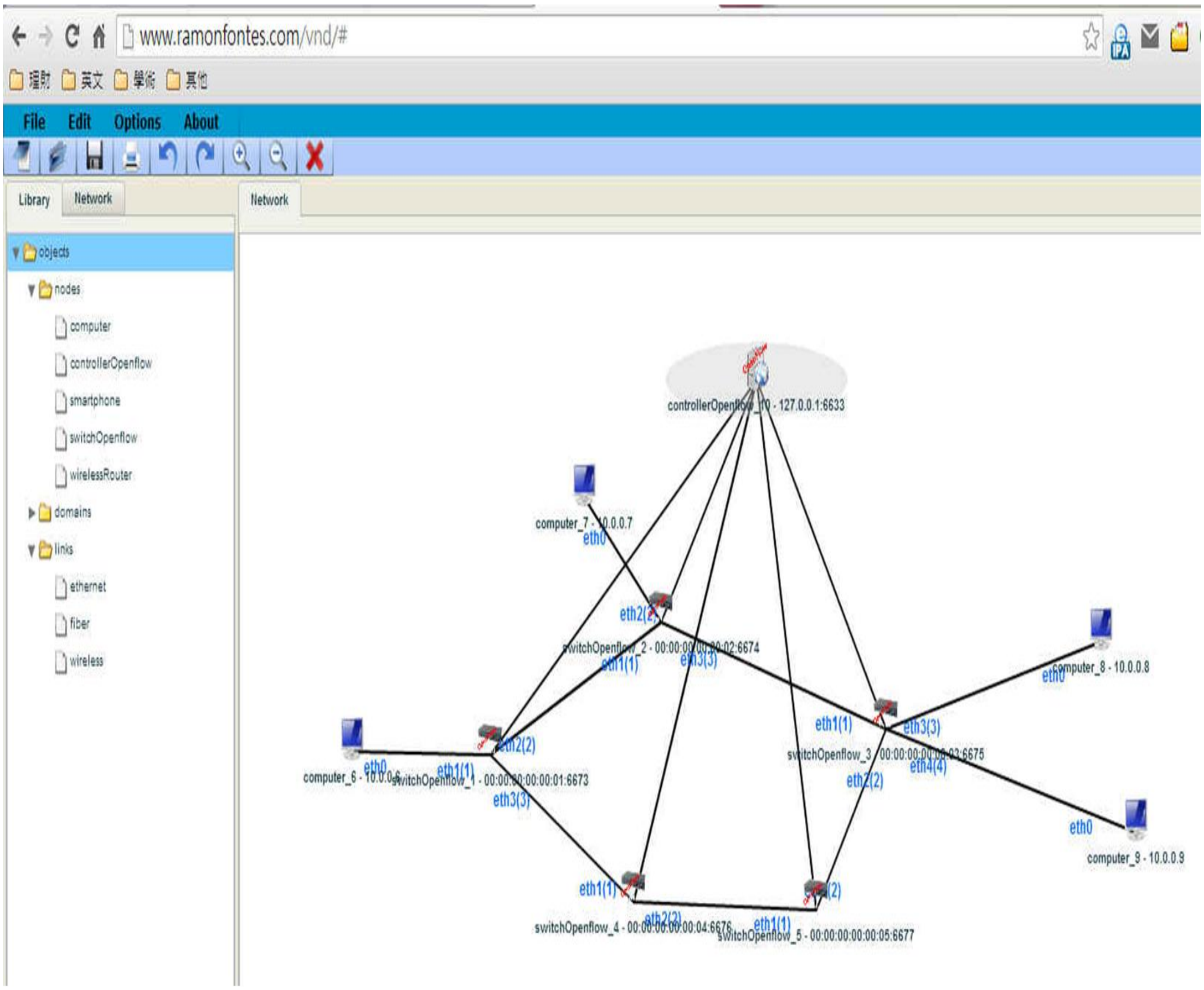
(above files can be downloaded from here)

5. Other related can refer to myEvalSVC-Mininet.

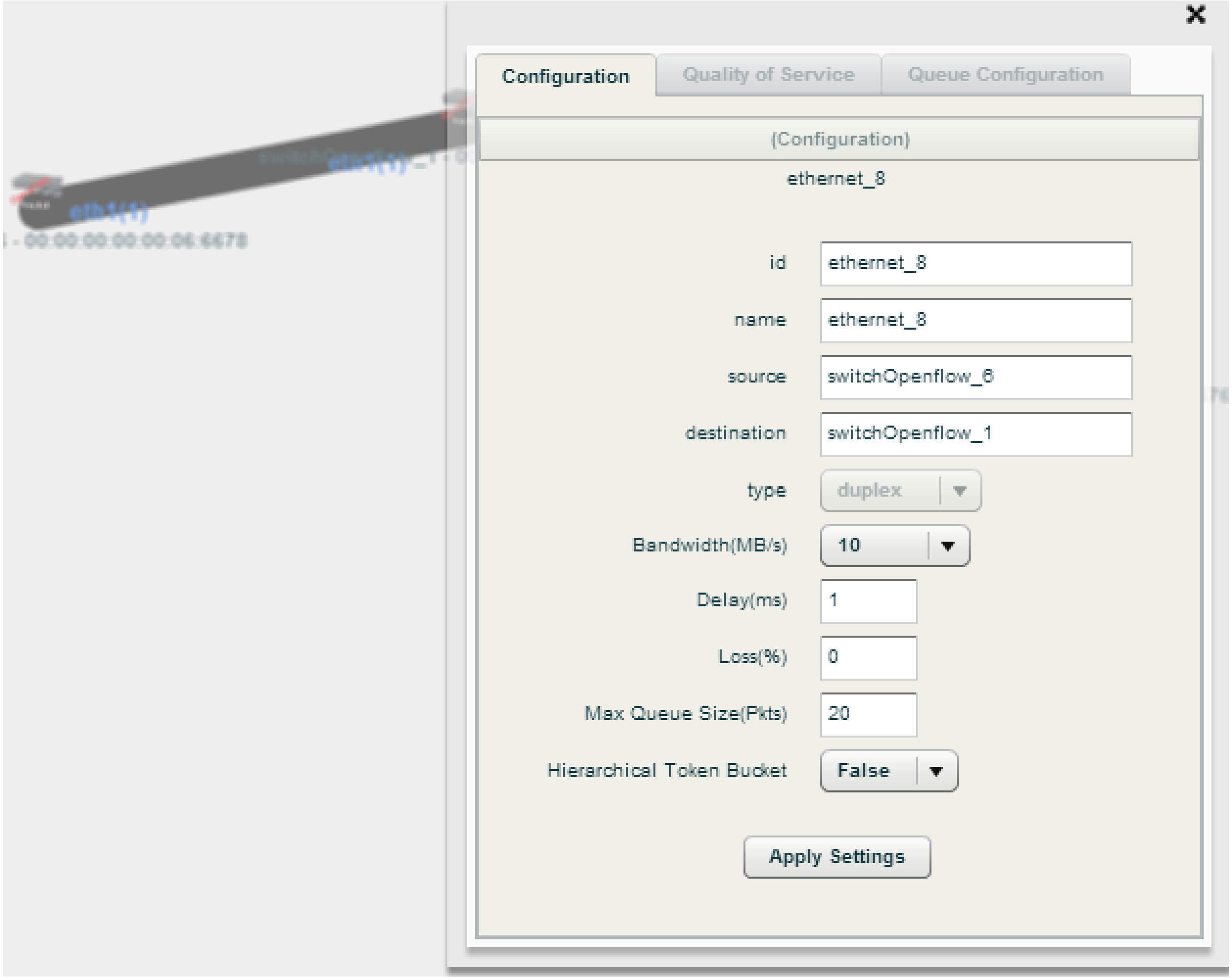
[Steps]

1. Go to http://www.ramonfontes.com/vnd/# to create your own SDN topology. (More detail operations can refer to https://www.youtube.com/playlist?list=PLccoFREVAt_4nEtrkl59mjjf5ZzRX8DZA)

2. You can create a topology like the following figure.



3. click the link and set the parameters, such as bandwidth, delay, loss, and etc.



ethernet_8

id: ethernet_8
name: ethernet_8
source: switchOpenflow_6
destination: switchOpenflow_1
type: duplex
Bandwidth(MB/s): 10
Delay(ms): 1
Loss(%): 0
Max Queue Size(Pkts): 20
Hierarchical Token Bucket: False

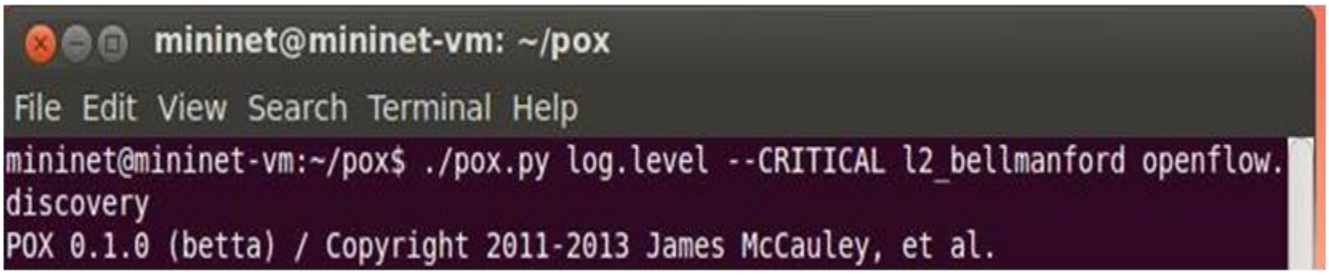
4. click File-> Export -> Export to Mininet

(mininetScript0224.sh)

```
#!/usr/bin/python

"""
Script created by VND - Visual Network Description (SDN version)
"""
from mininet.net import Mininet
from mininet.node import Controller, RemoteController, OVSKernelSwitch,
OVSLegacyKernelSwitch, UserSwitch
from mininet.cli import CLI
from mininet.log import setLogLevel
from mininet.link import Link, TCLink

def topology():
   "Create a network."
   net = Mininet( controller=RemoteController, link=TCLink,
switch=OVSKernelSwitch )

   print "*** Creating nodes"
```

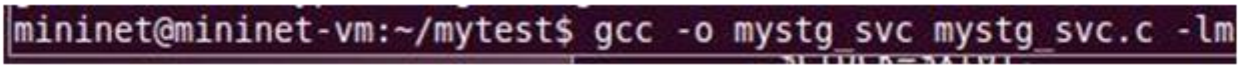
```
    s1 = net.addSwitch( 's1', listenPort=6673, mac='00:00:00:00:00:01' )
    s2 = net.addSwitch( 's2', listenPort=6674, mac='00:00:00:00:00:02' )
    s3 = net.addSwitch( 's3', listenPort=6675, mac='00:00:00:00:00:03' )
    s4 = net.addSwitch( 's4', listenPort=6676, mac='00:00:00:00:00:04' )
    s5 = net.addSwitch( 's5', listenPort=6677, mac='00:00:00:00:00:05' )
    h6 = net.addHost( 'h6', mac='00:00:00:00:00:06', ip='10.0.0.6/8' )
    h7 = net.addHost( 'h7', mac='00:00:00:00:00:07', ip='10.0.0.7/8' )
    h8 = net.addHost( 'h8', mac='00:00:00:00:00:08', ip='10.0.0.8/8' )
    h9 = net.addHost( 'h9', mac='00:00:00:00:00:09', ip='10.0.0.9/8' )
    c10 = net.addController( 'c10', controller=RemoteController, ip='127.0.0.1',
port=6633 )

    print "*** Creating links"
    net.addLink(s3, h9, 4, 0, bw=10, delay='1ms', max_queue_size=20, loss=0)
    net.addLink(s3, h8, 3, 0, bw=10, delay='1ms', max_queue_size=20, loss=0)
    net.addLink(s5, s3, 2, 2, bw=10, delay='1ms', max_queue_size=20, loss=0)
    net.addLink(s4, s5, 2, 1, bw=10, delay='1ms', max_queue_size=20, loss=0)
    net.addLink(s2, s3, 3, 1, bw=10, delay='1ms', max_queue_size=20, loss=0)
    net.addLink(h7, s2, 0, 2, bw=10, delay='1ms', max_queue_size=20, loss=0)
    net.addLink(s1, s4, 3, 1, bw=10, delay='1ms', max_queue_size=20, loss=0)
    net.addLink(s1, s2, 2, 1, bw=10, delay='1ms', max_queue_size=20, loss=0)
    net.addLink(h6, s1, 0, 1, bw=10, delay='1ms', max_queue_size=20, loss=0)

    print "*** Starting network"
    net.build()
    c10.start()
    s3.start( [c10] )
    s5.start( [c10] )
    s4.start( [c10] )
    s2.start( [c10] )
    s1.start( [c10] )

    print "*** Running CLI"
    CLI( net )

    print "*** Stopping network"
    net.stop()

if __name__ == '__main__':
    setLogLevel( 'info' )
    topology()
```

5. Open a terminal and run pox controller.

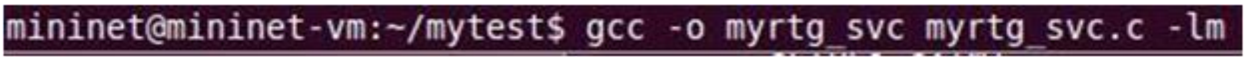

mininet@mininet-vm: ~/pox
File Edit View Search Terminal Help

```
mininet@mininet-vm:~/pox$ ./pox.py log.level --CRITICAL l2_bellmanford openflow.
discovery
POX 0.1.0 (betta) / Copyright 2011-2013 James McCauley, et al.
```

6. compile the H.264 sender and receiver program.



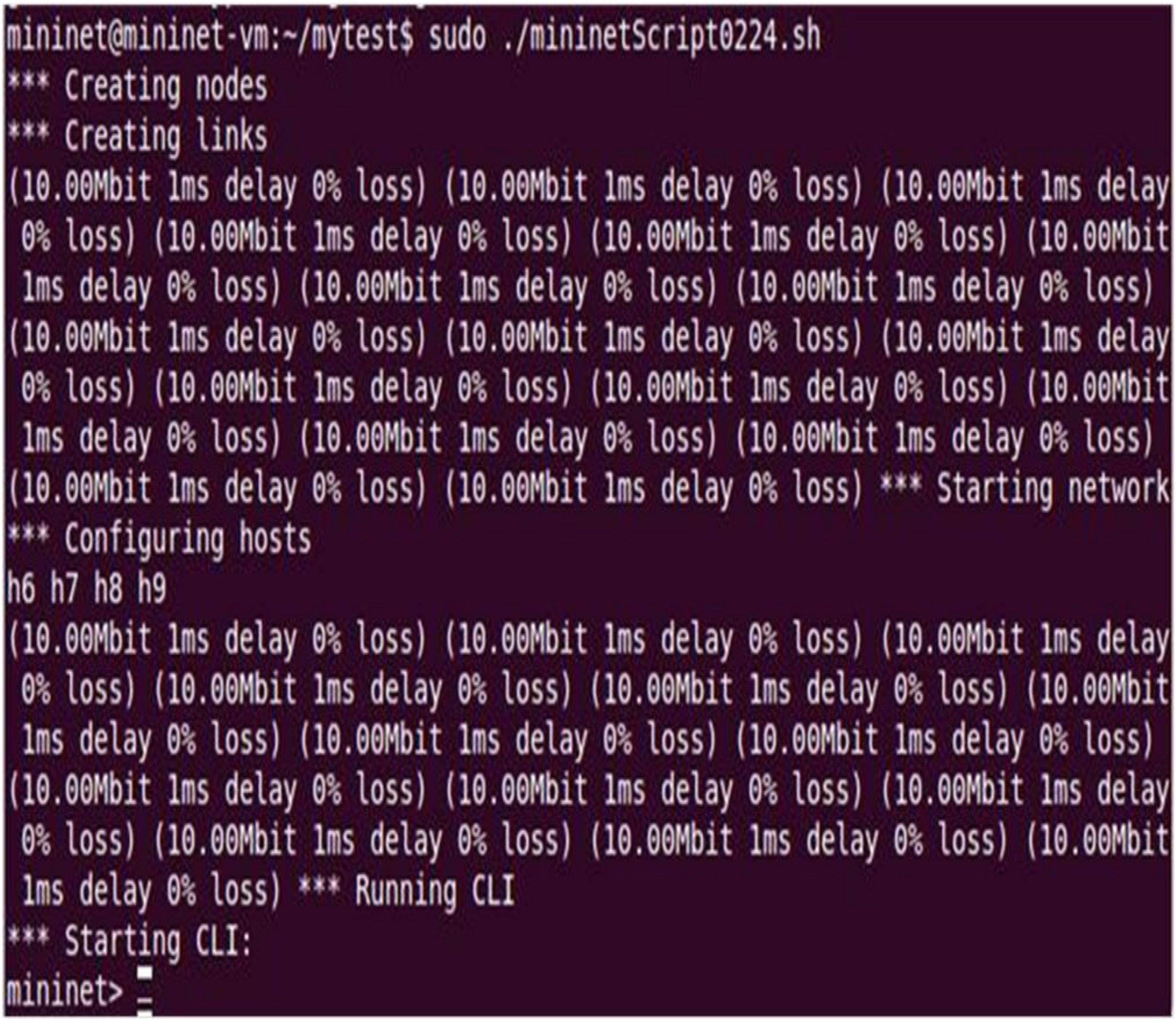
```
mininet@mininet-vm:~/mytest$ gcc -o mystg_svc mystg_svc.c -lm
```



```
mininet@mininet-vm:~/mytest$ gcc -o myrtg_svc myrtg_svc.c -lm
```

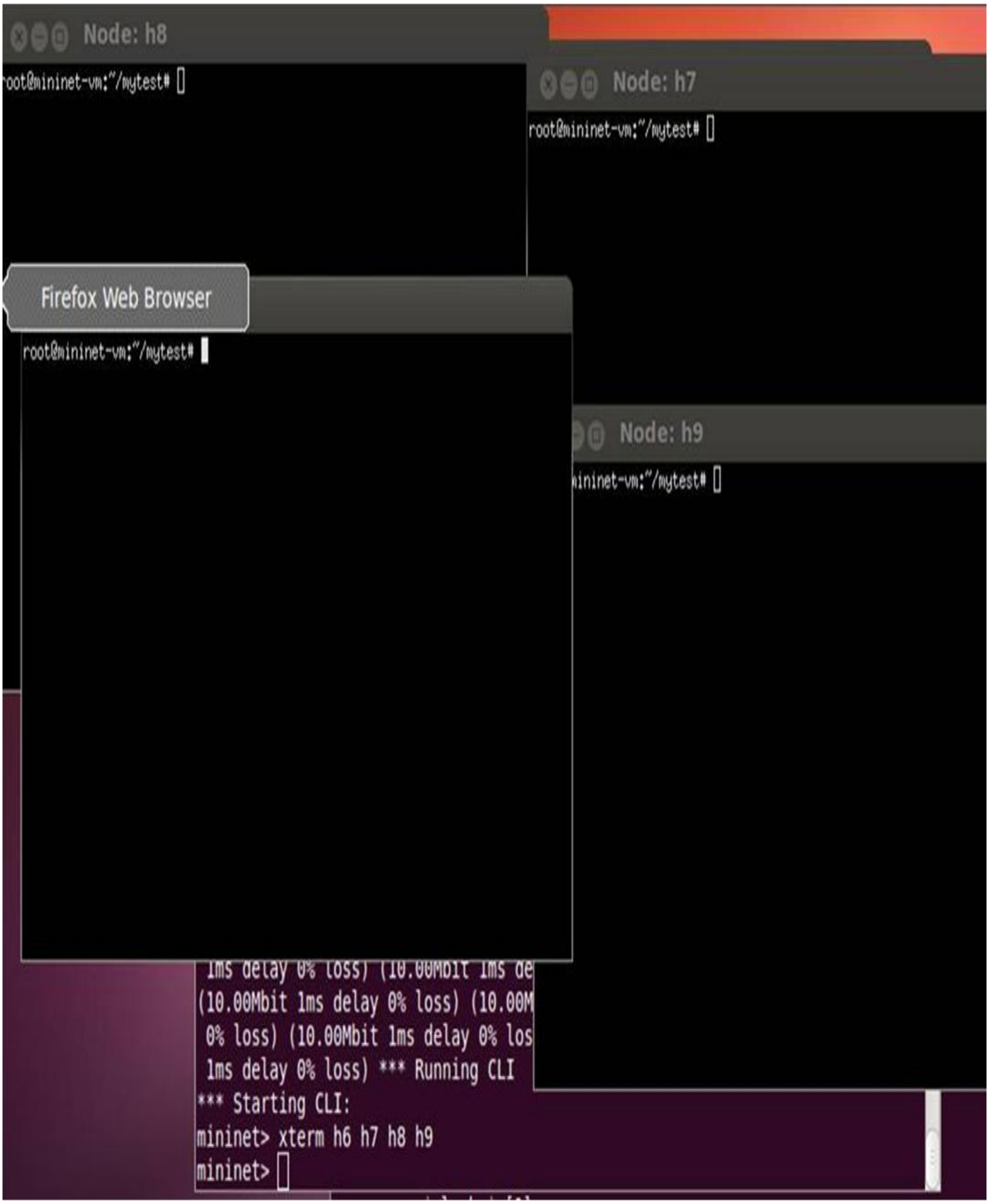
Video encoding and some related operations can refer to myEvalSVC-Mininet.

7. Open another terminal to run the mininet script.



```
mininet@mininet-vm:~/mytest$ sudo ./mininetScript0224.sh
*** Creating nodes
*** Creating links
(10.00Mbit 1ms delay 0% loss) (10.00Mbit 1ms delay 0% loss) (10.00Mbit 1ms delay
 0% loss) (10.00Mbit 1ms delay 0% loss) (10.00Mbit 1ms delay 0% loss) (10.00Mbit
 1ms delay 0% loss) (10.00Mbit 1ms delay 0% loss) (10.00Mbit 1ms delay 0% loss)
(10.00Mbit 1ms delay 0% loss) (10.00Mbit 1ms delay 0% loss) (10.00Mbit 1ms delay
 0% loss) (10.00Mbit 1ms delay 0% loss) (10.00Mbit 1ms delay 0% loss) (10.00Mbit
 1ms delay 0% loss) (10.00Mbit 1ms delay 0% loss) (10.00Mbit 1ms delay 0% loss)
(10.00Mbit 1ms delay 0% loss) (10.00Mbit 1ms delay 0% loss) *** Starting network
*** Configuring hosts
h6 h7 h8 h9
(10.00Mbit 1ms delay 0% loss) (10.00Mbit 1ms delay 0% loss) (10.00Mbit 1ms delay
 0% loss) (10.00Mbit 1ms delay 0% loss) (10.00Mbit 1ms delay 0% loss) (10.00Mbit
 1ms delay 0% loss) (10.00Mbit 1ms delay 0% loss) (10.00Mbit 1ms delay 0% loss)
(10.00Mbit 1ms delay 0% loss) (10.00Mbit 1ms delay 0% loss) (10.00Mbit 1ms delay
 0% loss) (10.00Mbit 1ms delay 0% loss) (10.00Mbit 1ms delay 0% loss) (10.00Mbit
 1ms delay 0% loss) *** Running CLI
*** Starting CLI:
mininet>
```

8. type xterm h6 h7 h8 h9 to open four xterm windows. (h6->h8: H.264/SVC video transmission, h7->h9: iperf for background traffic)



```
root@mininet-vm:~/mytest# 
```

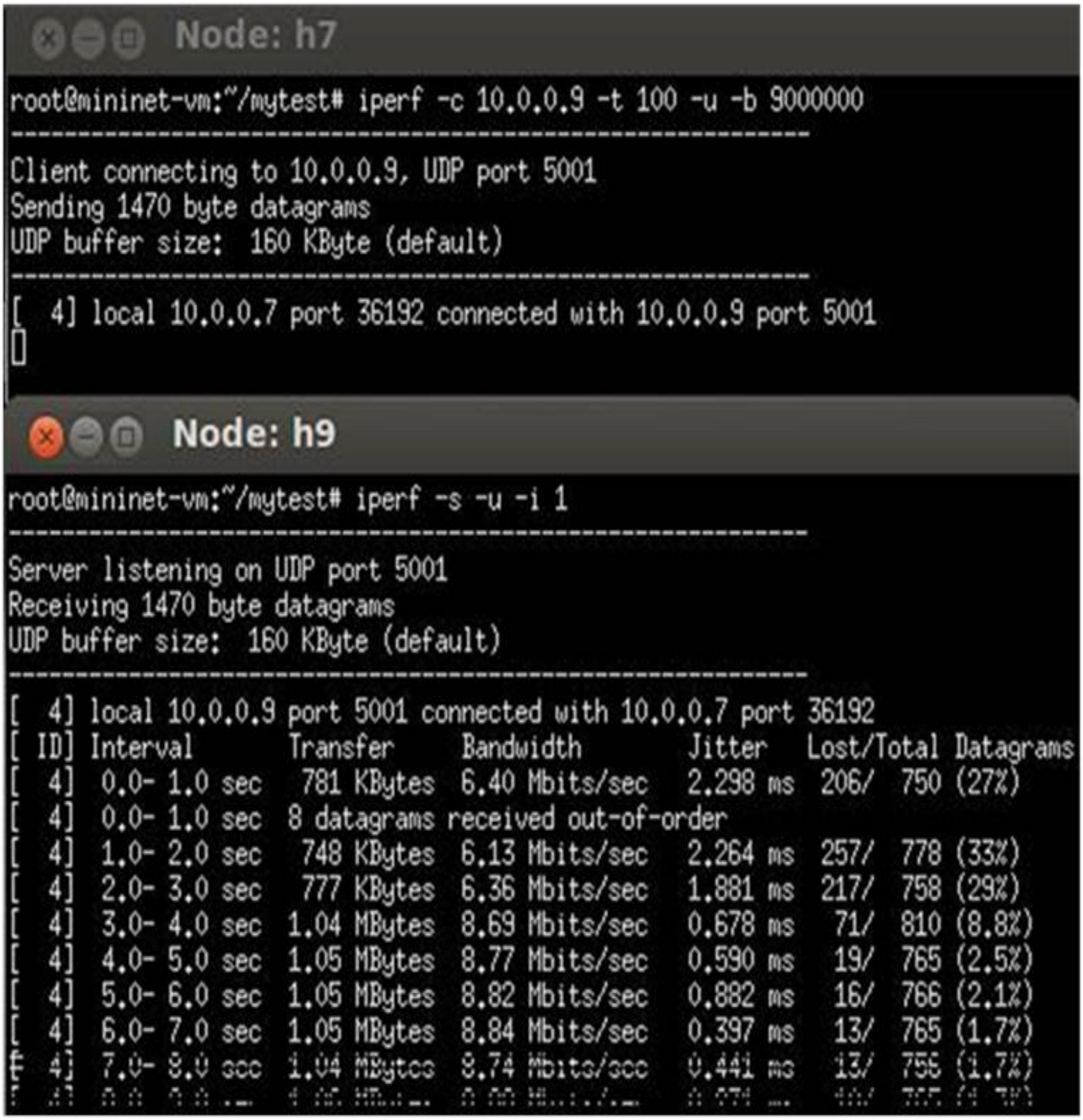
```
root@mininet-vm:~/mytest# 
```

```
root@mininet-vm:~/mytest# 
```

```
mininet-vm:~/mytest# 
```

```
 1ms delay 0% loss) (10.00Mbit 1ms de
(10.00Mbit 1ms delay 0% loss) (10.00M
 0% loss) (10.00Mbit 1ms delay 0% los
 1ms delay 0% loss) *** Running CLI
*** Starting CLI:
mininet> xterm h6 h7 h8 h9
mininet> 
```

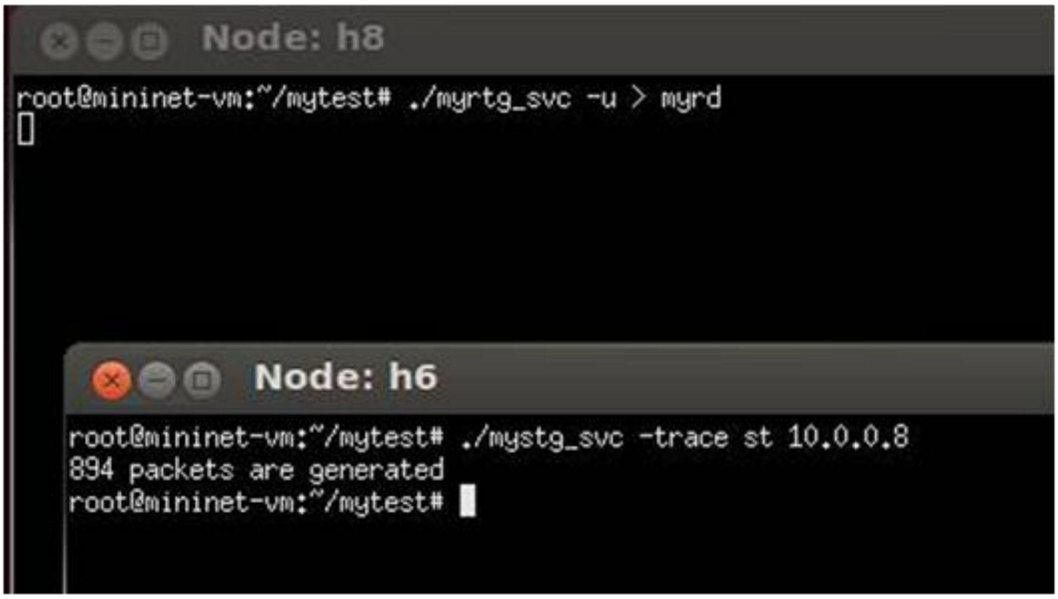
9. use iperf in h7 and h9 to generate the background traffic



```
root@mininet-vm:~/mytest# iperf -c 10.0.0.9 -t 100 -u -b 9000000
------------------------------------------------------------
Client connecting to 10.0.0.9, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size:  160 KByte (default)
------------------------------------------------------------
[  4] local 10.0.0.7 port 36192 connected with 10.0.0.9 port 5001
```

```
root@mininet-vm:~/mytest# iperf -s -u -i 1
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:  160 KByte (default)
------------------------------------------------------------
[  4] local 10.0.0.9 port 5001 connected with 10.0.0.7 port 36192
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  4]  0.0- 1.0 sec   781 KBytes  6.40 Mbits/sec   2.298 ms  206/  750 (27%)
[  4]  0.0- 1.0 sec  8 datagrams received out-of-order
[  4]  1.0- 2.0 sec   748 KBytes  6.13 Mbits/sec   2.264 ms  257/  778 (33%)
[  4]  2.0- 3.0 sec   777 KBytes  6.36 Mbits/sec   1.881 ms  217/  758 (29%)
[  4]  3.0- 4.0 sec  1.04 MBytes  8.69 Mbits/sec   0.678 ms   71/  810 (8.8%)
[  4]  4.0- 5.0 sec  1.05 MBytes  8.77 Mbits/sec   0.590 ms   19/  765 (2.5%)
[  4]  5.0- 6.0 sec  1.05 MBytes  8.82 Mbits/sec   0.882 ms   16/  766 (2.1%)
[  4]  6.0- 7.0 sec  1.05 MBytes  8.84 Mbits/sec   0.397 ms   13/  765 (1.7%)
[  4]  7.0- 8.0 sec  1.04 MBytes  8.74 Mbits/sec   0.441 ms   13/  756 (1.7%)
[illegible]
```

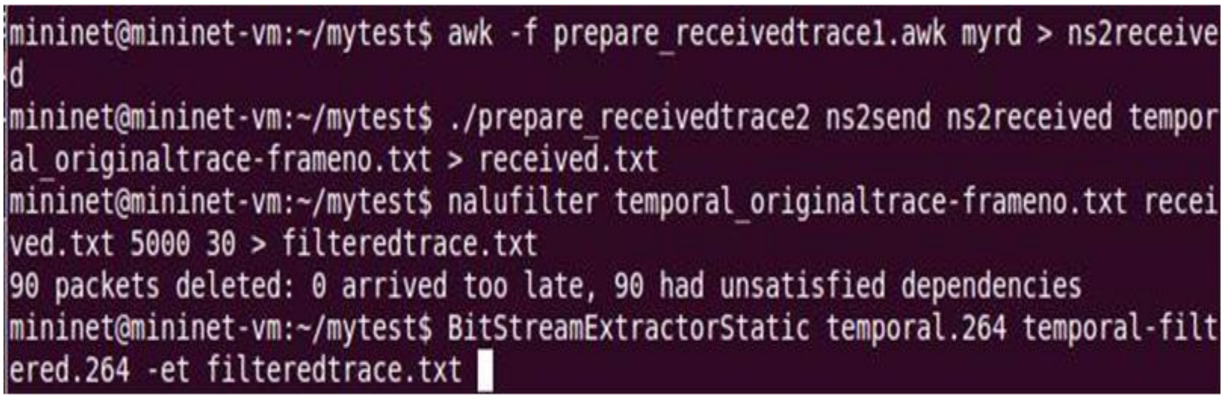
10. use mystg_svc in h6 and myrtg_svc in h8 to transmit the video packets.



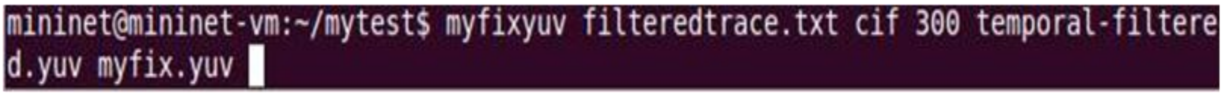
```
root@mininet-vm:~/mytest# ./myrtg_svc -u > myrd
```

```
root@mininet-vm:~/mytest# ./mystg_svc -trace st 10.0.0.8
894 packets are generated
root@mininet-vm:~/mytest#
```

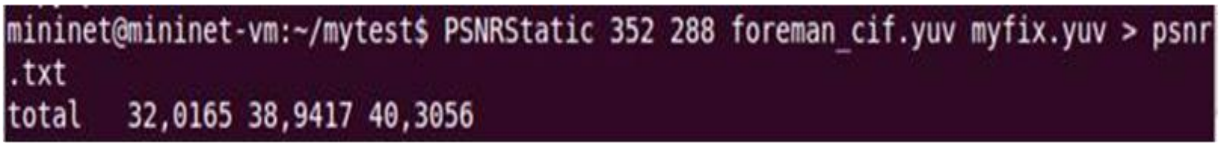
11. After evaluation, run the following commands to do performance evaluation.



```
mininet@mininet-vm:~/mytest$ awk -f prepare_receivedtrace1.awk myrd > ns2receive
d
mininet@mininet-vm:~/mytest$ ./prepare_receivedtrace2 ns2send ns2received tempor
al_originaltrace-frameno.txt > received.txt
mininet@mininet-vm:~/mytest$ nalufilter temporal_originaltrace-frameno.txt recei
ved.txt 5000 30 > filteredtrace.txt
90 packets deleted: 0 arrived too late, 90 had unsatisfied dependencies
mininet@mininet-vm:~/mytest$ BitStreamExtractorStatic temporal.264 temporal-filt
ered.264 -et filteredtrace.txt
```

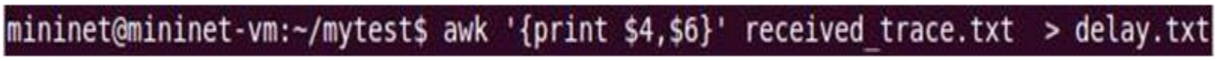


```
mininet@mininet-vm:~/mytest$ myfixyuv filteredtrace.txt cif 300 temporal-filtere
d.yuv myfix.yuv
```

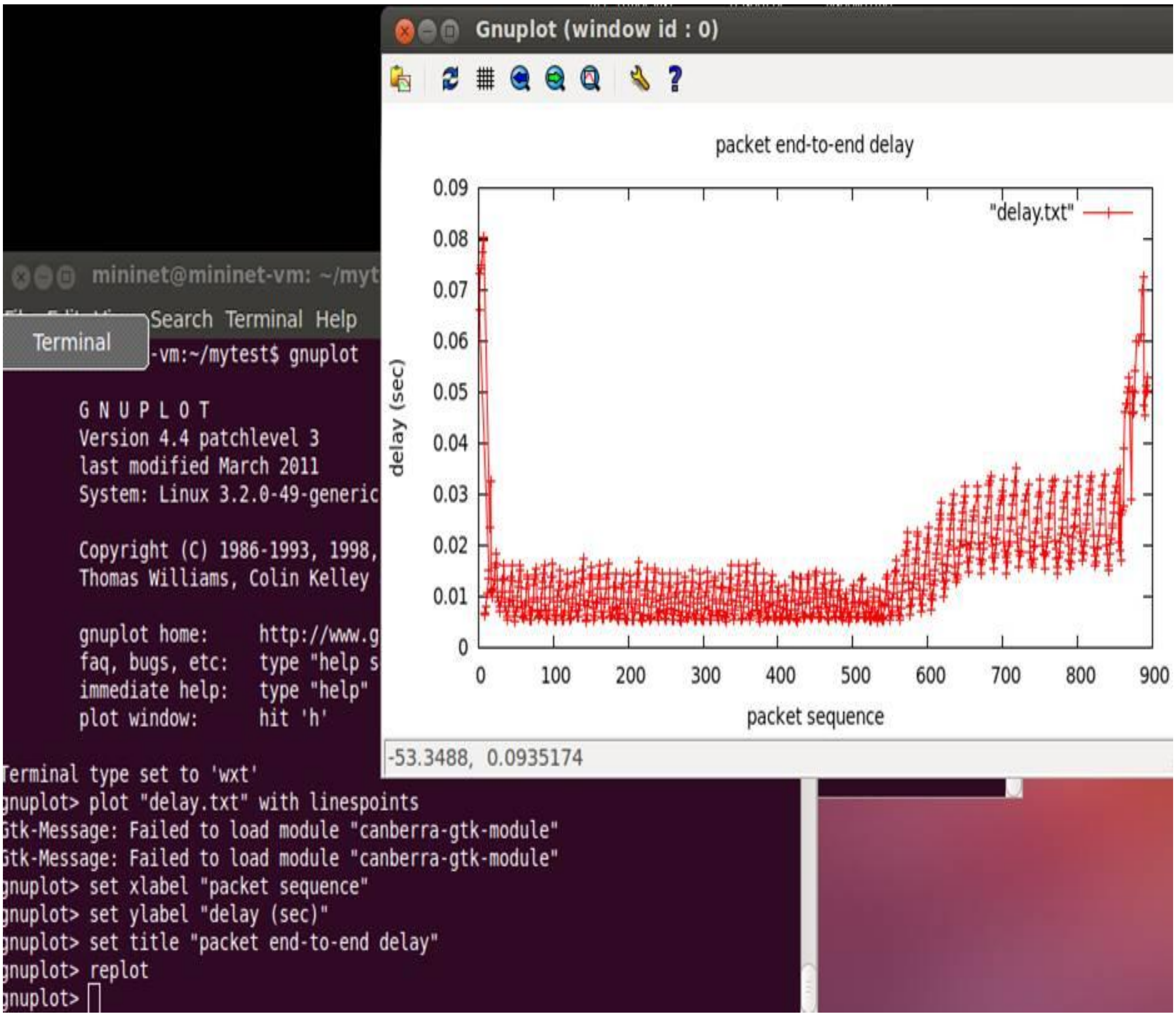


```
mininet@mininet-vm:~/mytest$ PSNRStatic 352 288 foreman_cif.yuv myfix.yuv > psnr
.txt
total   32,0165 38,9417 40,3056
```

(delay)

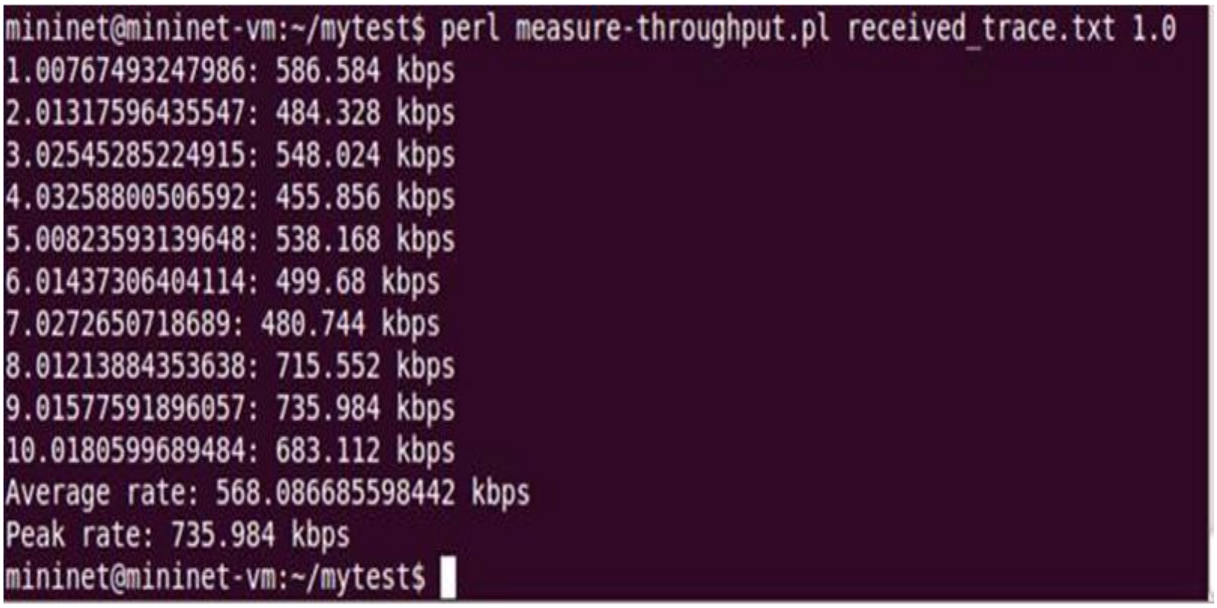


```
mininet@mininet-vm:~/mytest$ awk '{print $4,$6}' received_trace.txt  > delay.txt
```



```
-vm:~/mytest$ gnuplot

        G N U P L O T
        Version 4.4 patchlevel 3
        last modified March 2011
        System: Linux 3.2.0-49-generic

        Copyright (C) 1986-1993, 1998,
        Thomas Williams, Colin Kelley

        gnuplot home:     http://www.g
        faq, bugs, etc:   type "help s
        immediate help:   type "help"
        plot window:      hit 'h'

Terminal type set to 'wxt'
gnuplot> plot "delay.txt" with linespoints
Gtk-Message: Failed to load module "canberra-gtk-module"
Gtk-Message: Failed to load module "canberra-gtk-module"
gnuplot> set xlabel "packet sequence"
gnuplot> set ylabel "delay (sec)"
gnuplot> set title "packet end-to-end delay"
gnuplot> replot
gnuplot> 
```

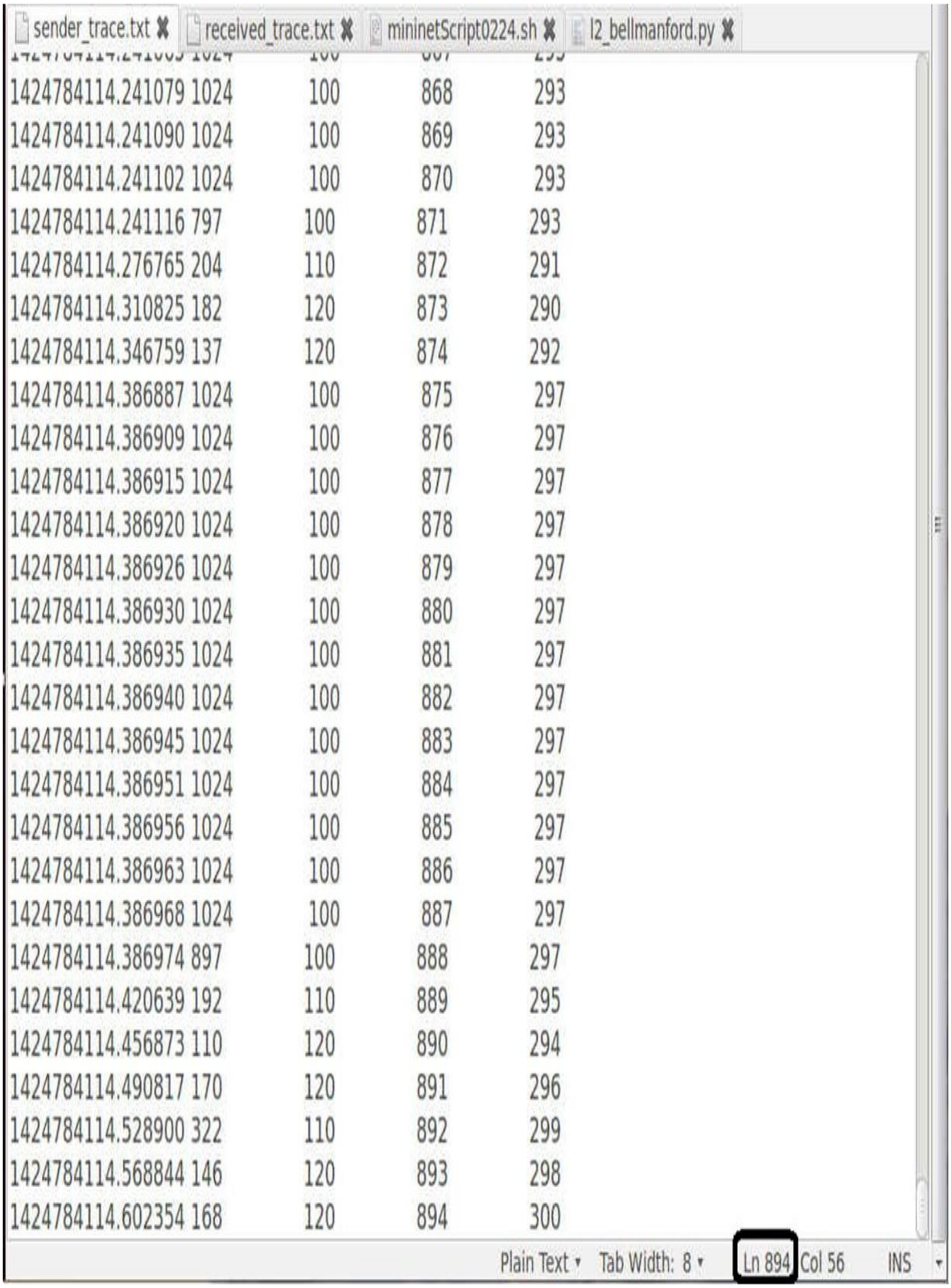
(throughput)



```
mininet@mininet-vm:~/mytest$ perl measure-throughput.pl received_trace.txt 1.0
1.00767493247986: 586.584 kbps
2.01317596435547: 484.328 kbps
3.02545285224915: 548.024 kbps
4.03258800506592: 455.856 kbps
5.00823593139648: 538.168 kbps
6.01437306404114: 499.68 kbps
7.0272650718689: 480.744 kbps
8.01213884353638: 715.552 kbps
9.01577591896057: 735.984 kbps
10.0180599689484: 683.112 kbps
Average rate: 568.086685598442 kbps
Peak rate: 735.984 kbps
mininet@mininet-vm:~/mytest$ 
```

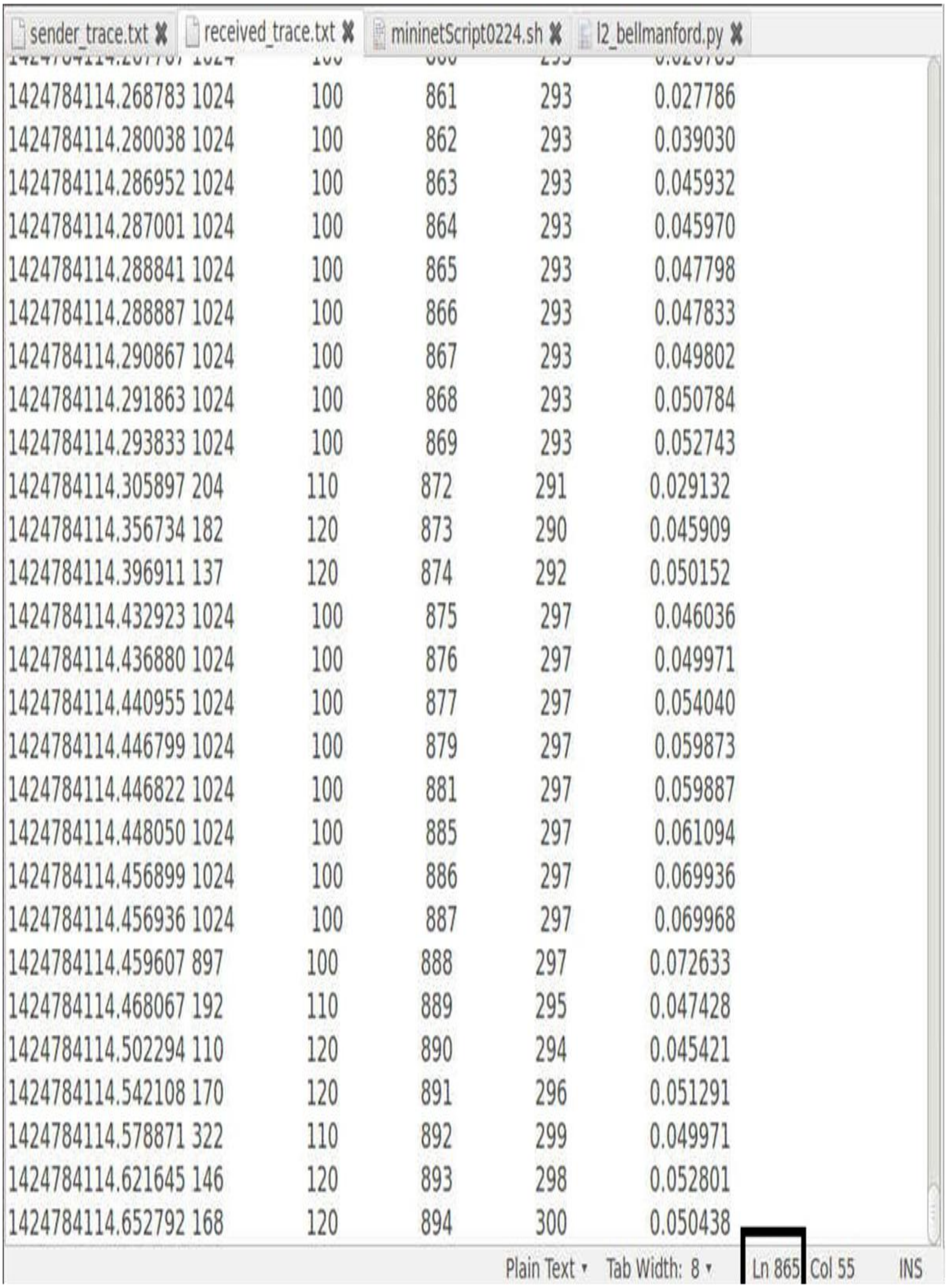
(packet loss rate)

count the number of record in sender_trace.txt



```
1424784114.241079 1024      100       868       293
1424784114.241090 1024      100       869       293
1424784114.241102 1024      100       870       293
1424784114.241116 797       100       871       293
1424784114.276765 204       110       872       291
1424784114.310825 182       120       873       290
1424784114.346759 137       120       874       292
1424784114.386887 1024      100       875       297
1424784114.386909 1024      100       876       297
1424784114.386915 1024      100       877       297
1424784114.386920 1024      100       878       297
1424784114.386926 1024      100       879       297
1424784114.386930 1024      100       880       297
1424784114.386935 1024      100       881       297
1424784114.386940 1024      100       882       297
1424784114.386945 1024      100       883       297
1424784114.386951 1024      100       884       297
1424784114.386956 1024      100       885       297
1424784114.386963 1024      100       886       297
1424784114.386968 1024      100       887       297
1424784114.386974 897       100       888       297
1424784114.420639 192       110       889       295
1424784114.456873 110       120       890       294
1424784114.490817 170       120       891       296
1424784114.528900 322       110       892       299
1424784114.568844 146       120       893       298
1424784114.602354 168       120       894       300
```

count the number of record in received_trace.txt.

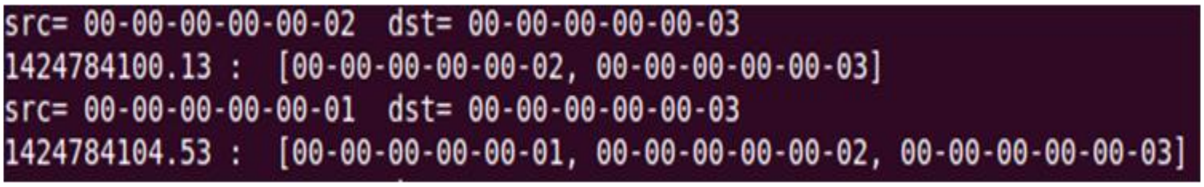


```
1424784114.268783 1024          100         861         293         0.027786
1424784114.280038 1024          100         862         293         0.039030
1424784114.286952 1024          100         863         293         0.045932
1424784114.287001 1024          100         864         293         0.045970
1424784114.288841 1024          100         865         293         0.047798
1424784114.288887 1024          100         866         293         0.047833
1424784114.290867 1024          100         867         293         0.049802
1424784114.291863 1024          100         868         293         0.050784
1424784114.293833 1024          100         869         293         0.052743
1424784114.305897 204           110         872         291         0.029132
1424784114.356734 182           120         873         290         0.045909
1424784114.396911 137           120         874         292         0.050152
1424784114.432923 1024          100         875         297         0.046036
1424784114.436880 1024          100         876         297         0.049971
1424784114.440955 1024          100         877         297         0.054040
1424784114.446799 1024          100         879         297         0.059873
1424784114.446822 1024          100         881         297         0.059887
1424784114.448050 1024          100         885         297         0.061094
1424784114.456899 1024          100         886         297         0.069936
1424784114.456936 1024          100         887         297         0.069968
1424784114.459607 897           100         888         297         0.072633
1424784114.468067 192           110         889         295         0.047428
1424784114.502294 110           120         890         294         0.045421
1424784114.542108 170           120         891         296         0.051291
1424784114.578871 322           110         892         299         0.049971
1424784114.621645 146           120         893         298         0.052801
1424784114.652792 168           120         894         300         0.050438
```

So the packet loss rate = (894-865)/894 *100 = 3.24%

[Discussion]

Check the output of pox controller and we can find out that video traffic will go from s1-s2-s3 and background traffic will go s2-s3. If the video traffic can choose another path, i.e. s1-s4-s5-s3, the video can get better video delivered quality.



```
src= 00-00-00-00-00-02  dst= 00-00-00-00-00-03
1424784100.13 :  [00-00-00-00-00-02, 00-00-00-00-00-03]
src= 00-00-00-00-00-01  dst= 00-00-00-00-00-03
1424784104.53 :  [00-00-00-00-00-01, 00-00-00-00-00-02, 00-00-00-00-00-03]
```

Dr. Chih-Heng Ke
Department of Computer Science and Information Engineering, National Quemoy University, Kinmen, Taiwan
Email: smallko@gmail.com / smallko@nqu.edu.tw