

[2021 Advanced Computer Networks Homework 2]

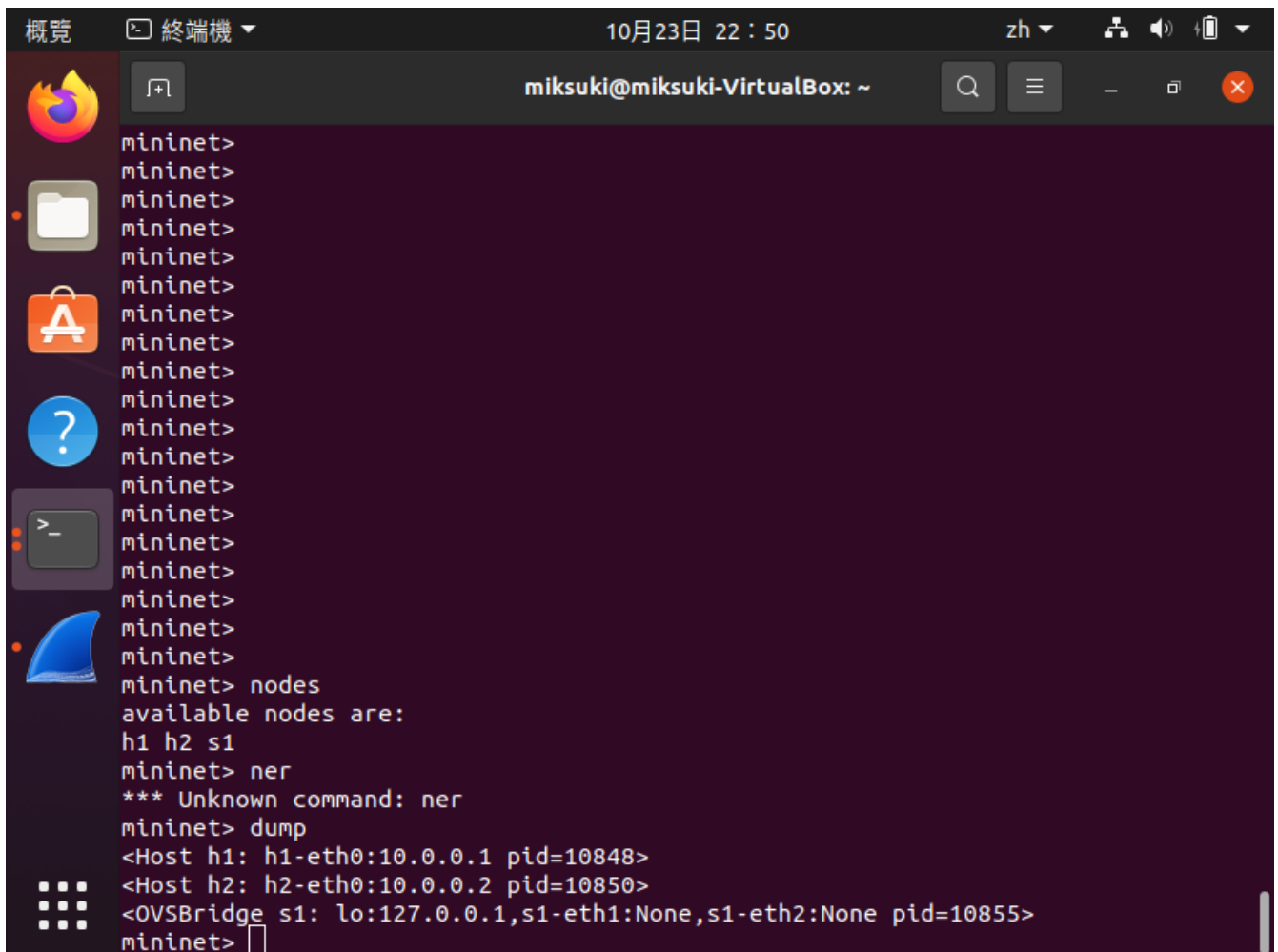
Part1:

Mininet

本部分將讓同學練習安裝mininet，並利用mininet 建立一個簡單的虛擬網路平台。請上網查詢安裝方式及基本使用的指令。

- 1 安裝mininet，可藉由**apt-get install mininet** 或其他方式完成安裝。
- 2 使用**"mn"**可建立基本的虛擬拓樸。
- 3 有幾個基本指令可以顯示現在的虛擬拓樸中節點資訊或鏈結的訊息等，如**"nodes"**、**"net"**、**"dump"**等，請嘗試使用這些指令，並觀察顯示的訊息，可使用**help** 查詢mininet 可支援的指令。
- 4 請打開**wireshark**，擷取兩個虛擬host 的網卡，再使用mininet 從**h1 ping h2**，將你所看到的wireshark 畫面擷取下來。

mininet 指令



```
概覽 終端機 10月23日 22:50 zh
miksuki@miksuki-VirtualBox: ~
mininet>
mininet>
mininet>
mininet>
mininet>
mininet>
mininet>
mininet>
mininet>
mininet>
mininet>
mininet>
mininet>
mininet>
mininet>
mininet>
mininet>
mininet>
mininet>
mininet>
mininet> nodes
available nodes are:
h1 h2 s1
mininet> ner
*** Unknown command: ner
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=10848>
<Host h2: h2-eth0:10.0.0.2 pid=10850>
<OVSBridge s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=10855>
mininet>
```

Winshark 畫面

Wireshark 10月23日 22:48 zh

Capturing from s1-eth1

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

Interface Channel 802.11 Preferences

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request
2	0.000128088	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply
3	1.024825137	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request
4	1.024851980	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply
5	2.048558666	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request
6	2.048585447	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply
7	3.072848371	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request
8	3.072875146	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply
9	4.096541129	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request
10	4.096570753	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply

Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface s1-eth1, id 0
Ethernet II, Src: 3e:2b:ce:18:67:32 (3e:2b:ce:18:67:32), Dst: 86:12:9f:7a:12:a2 (86:12:9f:7a:12:a2)
Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.0.2
Internet Control Message Protocol

```
0000 86 12 9f 7a 12 a2 3e 2b ce 18 67 32 08 00 45 00  ...z...>+...g2...E.
0010 00 54 ae 8c 40 00 40 01 78 1a 0a 00 00 01 0a 00  .T...@...x.....
0020 00 02 08 00 e9 a1 2a f6 00 01 c3 20 74 61 00 00  .....*...ta..
0030 00 00 e6 11 07 00 00 00 00 00 10 11 12 13 14 15  .....
0040 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25  .....!""#$%
0050 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35  &'()*+,-./012345
0060 36 37 67
```

s1-eth1: <live capture in progress> Packets: 22 • Displayed: 22 (100.0%) Profile: Default

1. tshark

請使用tshark指令搭配正確的參數達到以下的要求：

- 抓取“**icmp**”的封包，並且來源或目的是“**8.8.8.8**”
- 將擷取的封包儲存為一個檔案，名稱為“**packet01**”
- 下達正確的指令後，**開啟另一個視窗**ping 8.8.8.8 及 ping 208.67.220.220
- 結束後使用**tshark** 指令查看擷取的檔案
- 以上請寫出正確的指令並截圖證明

tshark 抓取 icmp 封包，且寫入檔案 packet01 指令:

```
sudo tshark -f icmp -w packet01.pcap
```

The screenshot shows a Kali Linux desktop environment. The top panel displays the date and time as '10月23日 23:34' and the language as 'zh'. The terminal window is titled 'miksuki@miksuki-VirtualBox: /home'. The terminal output shows the command 'sudo tshark -f icmp -w packet01.pcap' being executed, resulting in the message: 'Running as user "root" and group "root". This could be dangerous. Capturing on 'enp0s3''.

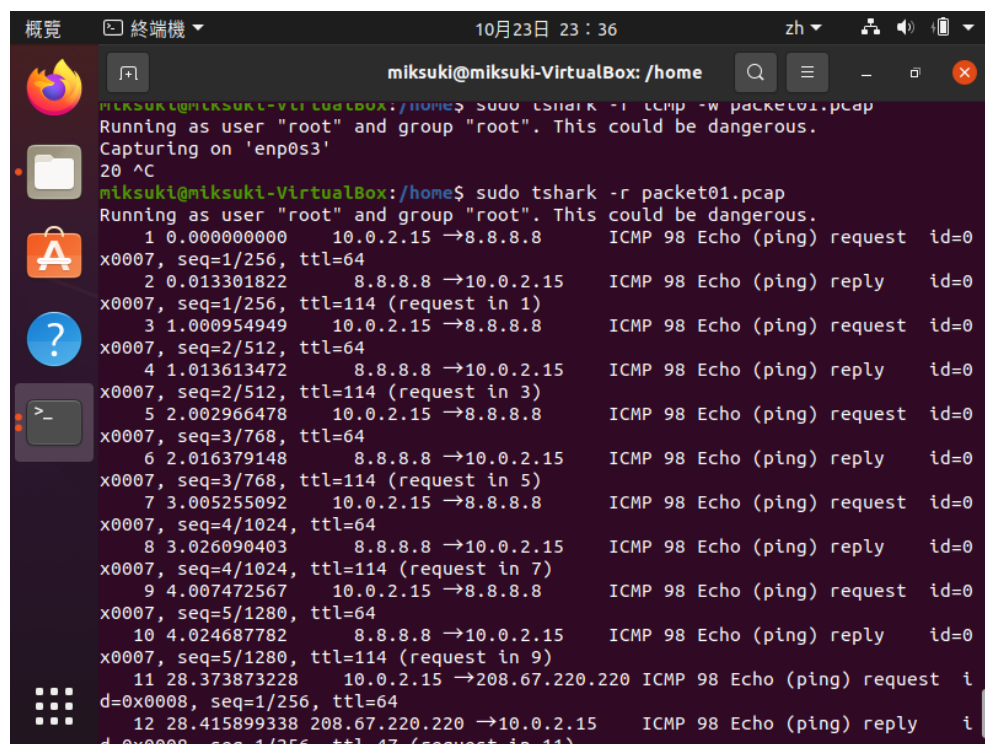
ping 8.8.8.8 及 ping 208.67.220.220

The screenshot shows a terminal window with the following content:

```
miksuki@miksuki-VirtualBox: ~
64 bytes from 8.8.8.8: icmp_seq=3 ttl=114 time=13.0 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=114 time=15.2 ms
^C
--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 12.594/14.736/18.111/2.190 ms
miksuki@miksuki-VirtualBox:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=114 time=13.3 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=114 time=12.7 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=114 time=13.5 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=114 time=20.9 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=114 time=17.3 ms
^C
--- 8.8.8.8 ping statistics: ---
5 packets transmitted, 5 received, 0% packet loss, time 4007ms
rtt min/avg/max/mdev = 12.683/15.512/20.855/3.119 ms
miksuki@miksuki-VirtualBox:~$ ping 208.67.220.220
PING 208.67.220.220 (208.67.220.220) 56(84) bytes of data.
64 bytes from 208.67.220.220: icmp_seq=1 ttl=47 time=42.0 ms
64 bytes from 208.67.220.220: icmp_seq=2 ttl=47 time=39.4 ms
64 bytes from 208.67.220.220: icmp_seq=3 ttl=47 time=40.6 ms
64 bytes from 208.67.220.220: icmp_seq=4 ttl=47 time=41.0 ms
64 bytes from 208.67.220.220: icmp_seq=5 ttl=47 time=41.4 ms
^C
--- 208.67.220.220 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4007ms
rtt min/avg/max/mdev = 39.423/40.881/42.044/0.870 ms
miksuki@miksuki-VirtualBox:~$
```

查看擷取檔案指令:

`sudo tshark -r packet01.pcap`



```
miksuki@miksuki-VirtualBox: /home
miksuki@miksuki-VirtualBox: /home$ sudo tshark -i temp -w packet01.pcap
Running as user "root" and group "root". This could be dangerous.
Capturing on 'enp0s3'
20 ^C
miksuki@miksuki-VirtualBox: /home$ sudo tshark -r packet01.pcap
Running as user "root" and group "root". This could be dangerous.
  1 0.000000000 10.0.2.15 → 8.8.8.8 ICMP 98 Echo (ping) request id=0
x0007, seq=1/256, ttl=64
  2 0.013301822 8.8.8.8 → 10.0.2.15 ICMP 98 Echo (ping) reply id=0
x0007, seq=1/256, ttl=114 (request in 1)
  3 1.000954949 10.0.2.15 → 8.8.8.8 ICMP 98 Echo (ping) request id=0
x0007, seq=2/512, ttl=64
  4 1.013613472 8.8.8.8 → 10.0.2.15 ICMP 98 Echo (ping) reply id=0
x0007, seq=2/512, ttl=114 (request in 3)
  5 2.002966478 10.0.2.15 → 8.8.8.8 ICMP 98 Echo (ping) request id=0
x0007, seq=3/768, ttl=64
  6 2.016379148 8.8.8.8 → 10.0.2.15 ICMP 98 Echo (ping) reply id=0
x0007, seq=3/768, ttl=114 (request in 5)
  7 3.005255092 10.0.2.15 → 8.8.8.8 ICMP 98 Echo (ping) request id=0
x0007, seq=4/1024, ttl=64
  8 3.026090403 8.8.8.8 → 10.0.2.15 ICMP 98 Echo (ping) reply id=0
x0007, seq=4/1024, ttl=114 (request in 7)
  9 4.007472567 10.0.2.15 → 8.8.8.8 ICMP 98 Echo (ping) request id=0
x0007, seq=5/1280, ttl=64
 10 4.024687782 8.8.8.8 → 10.0.2.15 ICMP 98 Echo (ping) reply id=0
x0007, seq=5/1280, ttl=114 (request in 9)
 11 28.373873228 10.0.2.15 → 208.67.220.220 ICMP 98 Echo (ping) request i
d=0x0008, seq=1/256, ttl=64
 12 28.415899338 208.67.220.220 → 10.0.2.15 ICMP 98 Echo (ping) reply i
d=0x0008, seq=1/256, ttl=47 (request in 11)
```

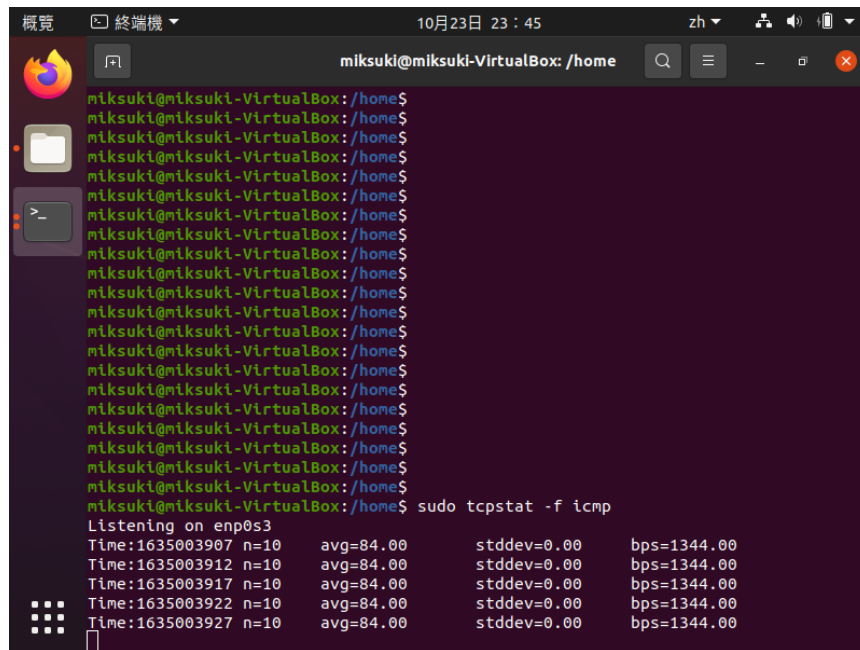
2. tcpstat

請使用 tcpstat 指令搭配正確的參數達到以下的要求：

- 抓取“icmp”的封包
- 開啟另一個視窗 ping 任意位址
- 完成後中斷tcpstat，將顯示的結果截圖，並寫出正確的指令

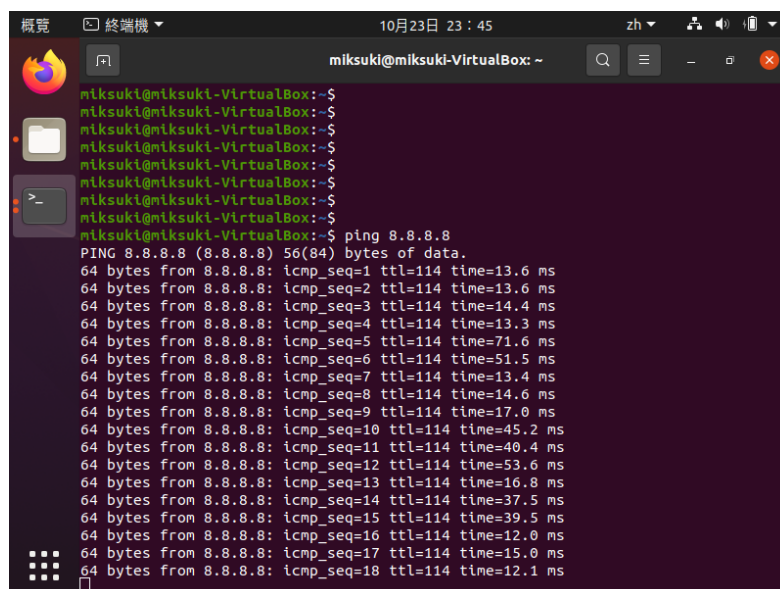
tcpstat 指令:

[sudo tcpstat -f icmp](#)

A terminal window titled 'miksuki@miksuki-VirtualBox: /home' showing the execution of 'sudo tcpstat -f icmp'. The command starts 'Listening on enp0s3'. The output shows five rows of statistics, each with a timestamp, sample size (n=10), average (avg=84.00), standard deviation (stddev=0.00), and bits per second (bps=1344.00).

```
miksuki@miksuki-VirtualBox: /home$  
miksuki@miksuki-VirtualBox: /home$  
miksuki@miksuki-VirtualBox: /home$  
miksuki@miksuki-VirtualBox: /home$  
miksuki@miksuki-VirtualBox: /home$  
miksuki@miksuki-VirtualBox: /home$  
miksuki@miksuki-VirtualBox: /home$  
miksuki@miksuki-VirtualBox: /home$  
miksuki@miksuki-VirtualBox: /home$  
miksuki@miksuki-VirtualBox: /home$  
miksuki@miksuki-VirtualBox: /home$  
miksuki@miksuki-VirtualBox: /home$  
miksuki@miksuki-VirtualBox: /home$  
miksuki@miksuki-VirtualBox: /home$  
miksuki@miksuki-VirtualBox: /home$  
miksuki@miksuki-VirtualBox: /home$  
miksuki@miksuki-VirtualBox: /home$  
miksuki@miksuki-VirtualBox: /home$  
miksuki@miksuki-VirtualBox: /home$  
miksuki@miksuki-VirtualBox: /home$ sudo tcpstat -f icmp  
Listening on enp0s3  
Time:1635003907 n=10 avg=84.00 stddev=0.00 bps=1344.00  
Time:1635003912 n=10 avg=84.00 stddev=0.00 bps=1344.00  
Time:1635003917 n=10 avg=84.00 stddev=0.00 bps=1344.00  
Time:1635003922 n=10 avg=84.00 stddev=0.00 bps=1344.00  
Time:1635003927 n=10 avg=84.00 stddev=0.00 bps=1344.00
```

ping任意位址: ping 8.8.8.8

A terminal window titled 'miksuki@miksuki-VirtualBox: ~' showing the execution of 'ping 8.8.8.8'. The output shows 18 successful ping requests, each with a 64-byte payload, TTL of 114, and various response times ranging from 12.1 ms to 71.6 ms.

```
miksuki@miksuki-VirtualBox: ~$  
miksuki@miksuki-VirtualBox: ~$  
miksuki@miksuki-VirtualBox: ~$  
miksuki@miksuki-VirtualBox: ~$  
miksuki@miksuki-VirtualBox: ~$  
miksuki@miksuki-VirtualBox: ~$  
miksuki@miksuki-VirtualBox: ~$  
miksuki@miksuki-VirtualBox: ~$  
miksuki@miksuki-VirtualBox: ~$ ping 8.8.8.8  
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.  
64 bytes from 8.8.8.8: icmp_seq=1 ttl=114 time=13.6 ms  
64 bytes from 8.8.8.8: icmp_seq=2 ttl=114 time=13.6 ms  
64 bytes from 8.8.8.8: icmp_seq=3 ttl=114 time=14.4 ms  
64 bytes from 8.8.8.8: icmp_seq=4 ttl=114 time=13.3 ms  
64 bytes from 8.8.8.8: icmp_seq=5 ttl=114 time=71.6 ms  
64 bytes from 8.8.8.8: icmp_seq=6 ttl=114 time=51.5 ms  
64 bytes from 8.8.8.8: icmp_seq=7 ttl=114 time=13.4 ms  
64 bytes from 8.8.8.8: icmp_seq=8 ttl=114 time=14.6 ms  
64 bytes from 8.8.8.8: icmp_seq=9 ttl=114 time=17.0 ms  
64 bytes from 8.8.8.8: icmp_seq=10 ttl=114 time=45.2 ms  
64 bytes from 8.8.8.8: icmp_seq=11 ttl=114 time=40.4 ms  
64 bytes from 8.8.8.8: icmp_seq=12 ttl=114 time=53.6 ms  
64 bytes from 8.8.8.8: icmp_seq=13 ttl=114 time=16.8 ms  
64 bytes from 8.8.8.8: icmp_seq=14 ttl=114 time=37.5 ms  
64 bytes from 8.8.8.8: icmp_seq=15 ttl=114 time=39.5 ms  
64 bytes from 8.8.8.8: icmp_seq=16 ttl=114 time=12.0 ms  
64 bytes from 8.8.8.8: icmp_seq=17 ttl=114 time=15.0 ms  
64 bytes from 8.8.8.8: icmp_seq=18 ttl=114 time=12.1 ms
```

3. tcpdump & tcpstat & gnuplot

本題要讓同學練習使用以上三個工具，將網路流量監測的結果繪製成圖表，繳交作業時請一併附上繪製出的圖檔，請依照下列步驟操作

- a. 使用tcpdump 擷取網路封包，每台電腦的網卡代號可能不同

```
tcpdump -i eth1 -w rawdata.dmp
```

- b. 開啟瀏覽器瀏覽網頁約一分鐘
- c. 中斷 tcpdump
- d. 使用tcpstat 將擷取的檔案做格式化

```
tcpstat -r rawdata.dmp -o "%r %A %T %U %l %b\n" > tcpstat.log
```

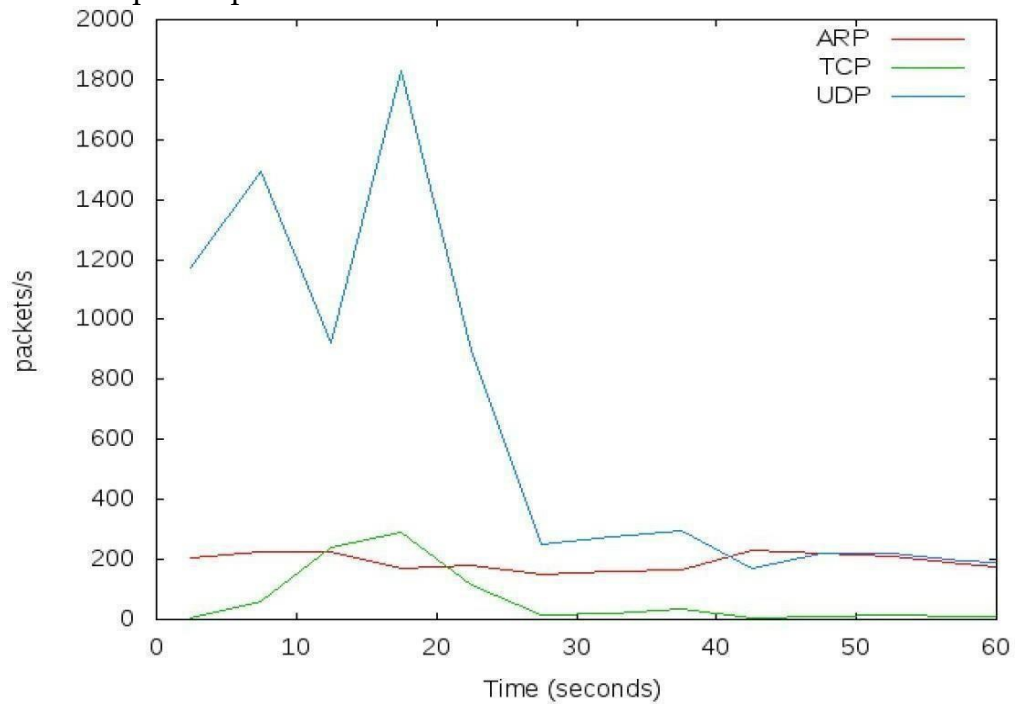
- d. 使用vim 寫一個script，名稱為“**script1**”，script 內容如下圖

```
set terminal png
set style data lines
set xlabel "Time (seconds)"
set ylabel "packets/s"
plot [00:60] "tcpstat.log" using 1:2 title "ARP", \
            "tcpstat.log" using 1:3 title "TCP", \
            "tcpstat.log" using 1:4 title "UDP"
```

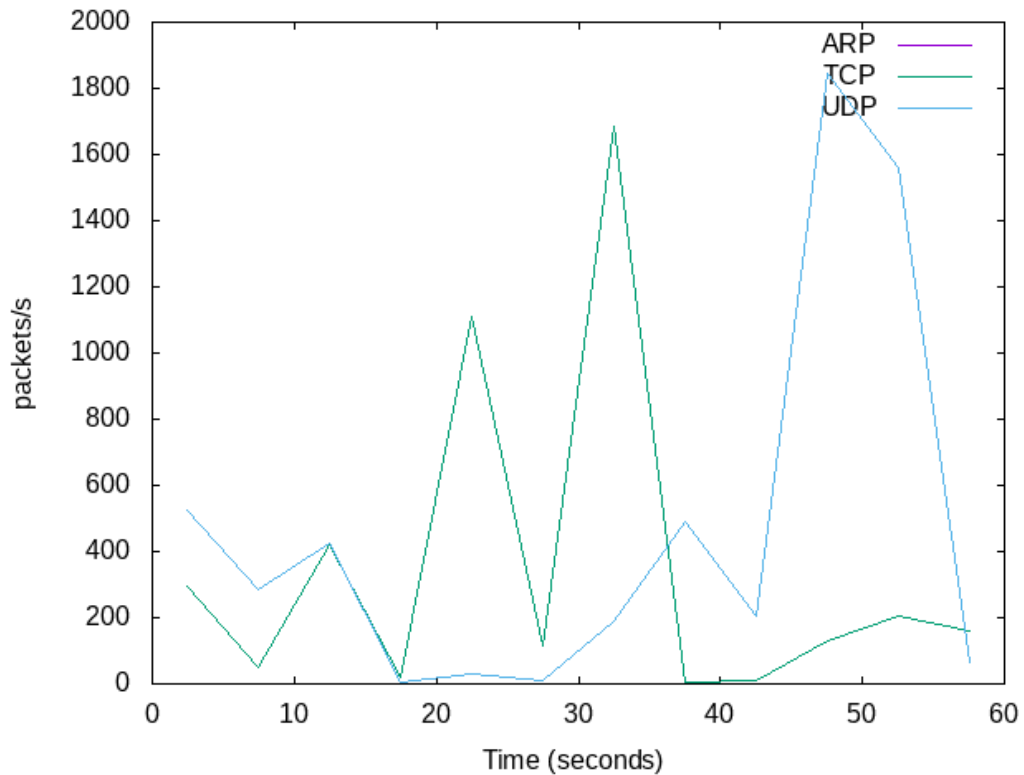
e. 利用gnuplot 繪圖，產生如下的圖表

gnuplot script1 > graph1.png

Sample output:



繪圖結果:



4. mininet & iperf & gnuplot

本題要讓同學練習在mininet下使用iperf，將分為以下 4個部份：

<請使用iperf3 指令完成作業>

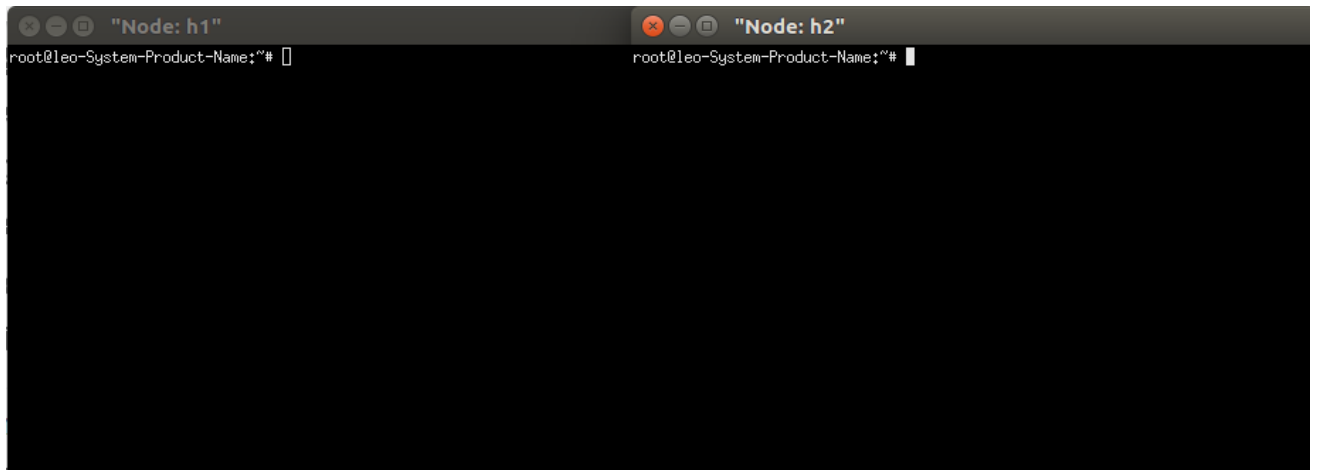
4-1.

請說明iperf 的用途，以及在什麼情況下你會需要使用它？

4-2.

請在mininet 下設計topology，使得hosts(Server 及Client)之間有1、3、5個節點，測量TCP 及UDP 傳輸時，不同數量節點的頻寬變化，並將結果存成檔案。【利用xterm <host>指令開啟host 視窗，分別在兩個不同host 的xterm視窗下iperf3 指令，即可開始測量】

此題有以下幾點事項注意:

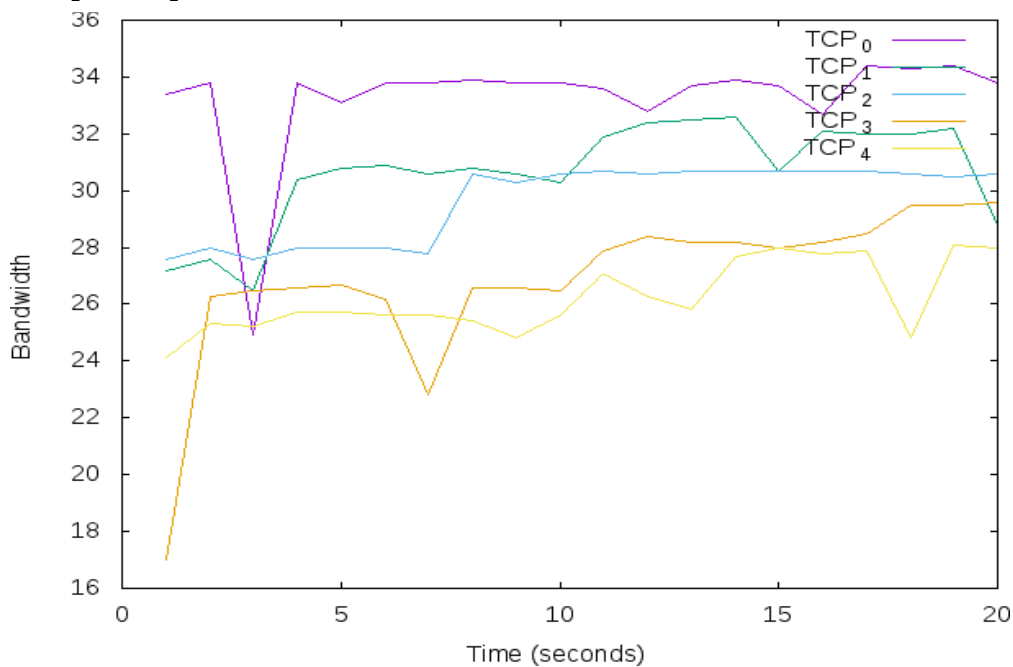


- (1) 需在mininet nodes 上完成，mininet nodes 如下圖所示。
- (2) 1、3、5 個節點表示Server 和Client 之間有幾個hops。
- (3) 請在Client 端指令後加入"> [檔案名稱]"將輸出導入檔案中。
- (4) 請注意UDP 頻寬有預設的最大值，需調整頻寬最大值才能看出差異。

4-3.

請參考Part2 第三小題自行修改script，將第二步tcp 及udp 的結果使用參考指令處理後，利用gnuplot 繪製兩張結果圖。輸出圖會類似此範例，UDP 請同學自行繪製。

Sample output :



4-4.

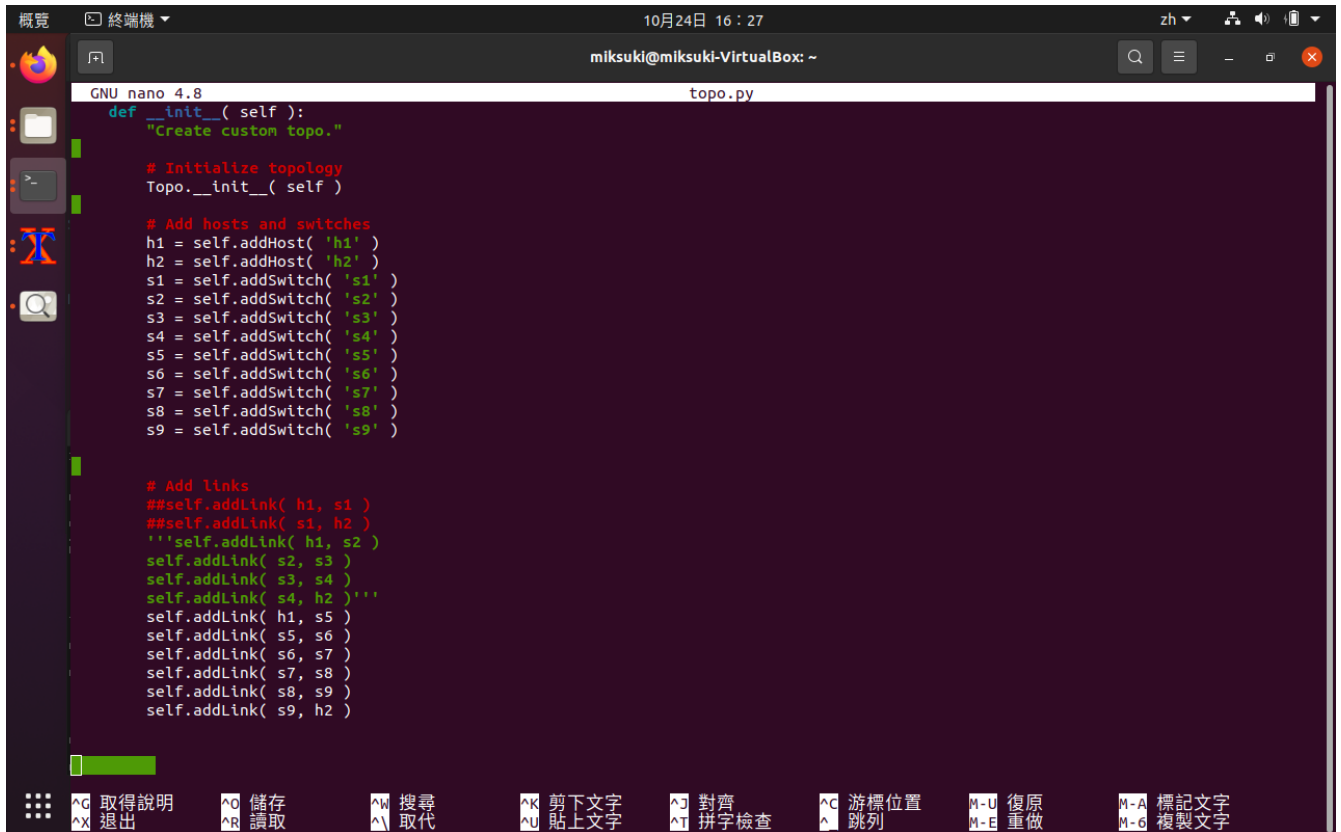
請說明TCP 及 UDP 產生結果差異的原因。

4.1

iperf 可用來測試網路效能，
當你覺得網速異常慢時，可用 **iperf** 來看看是哪邊出問題。

4.2 , 4.3

使用 python 建立 topology



```
GNU nano 4.8 topo.py
def __init__( self ):
    "Create custom topo."

    # Initialize topology
    Topo.__init__( self )

    # Add hosts and switches
    h1 = self.addHost( 'h1' )
    h2 = self.addHost( 'h2' )
    s1 = self.addSwitch( 's1' )
    s2 = self.addSwitch( 's2' )
    s3 = self.addSwitch( 's3' )
    s4 = self.addSwitch( 's4' )
    s5 = self.addSwitch( 's5' )
    s6 = self.addSwitch( 's6' )
    s7 = self.addSwitch( 's7' )
    s8 = self.addSwitch( 's8' )
    s9 = self.addSwitch( 's9' )

    # Add links
    #self.addLink( h1, s1 )
    ##self.addLink( s1, h2 )
    '''self.addLink( h1, s2 )
    self.addLink( s2, s3 )
    self.addLink( s3, s4 )
    self.addLink( s4, h2 )'''
    self.addLink( h1, s5 )
    self.addLink( s5, s6 )
    self.addLink( s6, s7 )
    self.addLink( s7, s8 )
    self.addLink( s8, s9 )
    self.addLink( s9, h2 )
```

mininet 指令:

```
sudo mn --custom topo.py --topo mytopo
```

iperf-tcp 指令:

```
iperf3 -c 10.0.0.1 -i 3 -t 20 > filename
```

iperf-udp 指令:

```
iperf3 -u -c 10.0.0.1 -i 3 -t 20 -b 512G > filename
```

1個節點的 TCP

```
root@miksuki-VirtualBox:/home/miksuki# iperf3 -s -i 3
Server listening on 5201
Accepted connection from 10.0.0.2, port 48576
[ 29] local 10.0.0.1 port 5201 connected to 10.0.0.2 port 48578
[ ID] Interval      Transfer     Bitrate
[ 29] 0.00-3.00 sec  12.5 GBytes  35.8 Gbits/sec
[ 29] 3.00-6.00 sec  12.5 GBytes  35.7 Gbits/sec
[ 29] 6.00-9.00 sec  13.9 GBytes  39.5 Gbits/sec
[ 29] 9.00-12.00 sec  13.0 GBytes  37.1 Gbits/sec
[ 29] 12.00-15.00 sec  13.5 GBytes  38.7 Gbits/sec
[ 29] 15.00-18.00 sec  12.4 GBytes  35.6 Gbits/sec
[ 29] 18.00-20.00 sec  8.54 GBytes  36.7 Gbits/sec
[ ID] Interval      Transfer     Bitrate
[ 29] 0.00-20.00 sec  86.2 GBytes  37.0 Gbits/sec
Server listening on 5201
[]

root@miksuki-VirtualBox:/home/miksuki# iperf3 -c 10.0.0.1 -i 3 -t 20 > tcp1
root@miksuki-VirtualBox:/home/miksuki#

miksuki@miksuki-VirtualBox:~$
miksuki@miksuki-VirtualBox:~$
miksuki@miksuki-VirtualBox:~$ sudo mn --custom topo.py --topo mytopo
*** No default OpenFlow controller found for default switch!
*** Falling back to OVS Bridge
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7 s8 s9
*** Adding links:
(h1, s1) (s1, h2)
*** Configuring hosts
h1 h2
*** Starting controller
*** Starting 9 switches
s1 s2 s3 s4 s5 s6 s7 s8 s9 ...
*** Starting CLI:
mininet> xterm h1
mininet> xterm h2
mininet> []
```

已選取「tcp3」 (999 位元組)

3個節點的 TCP

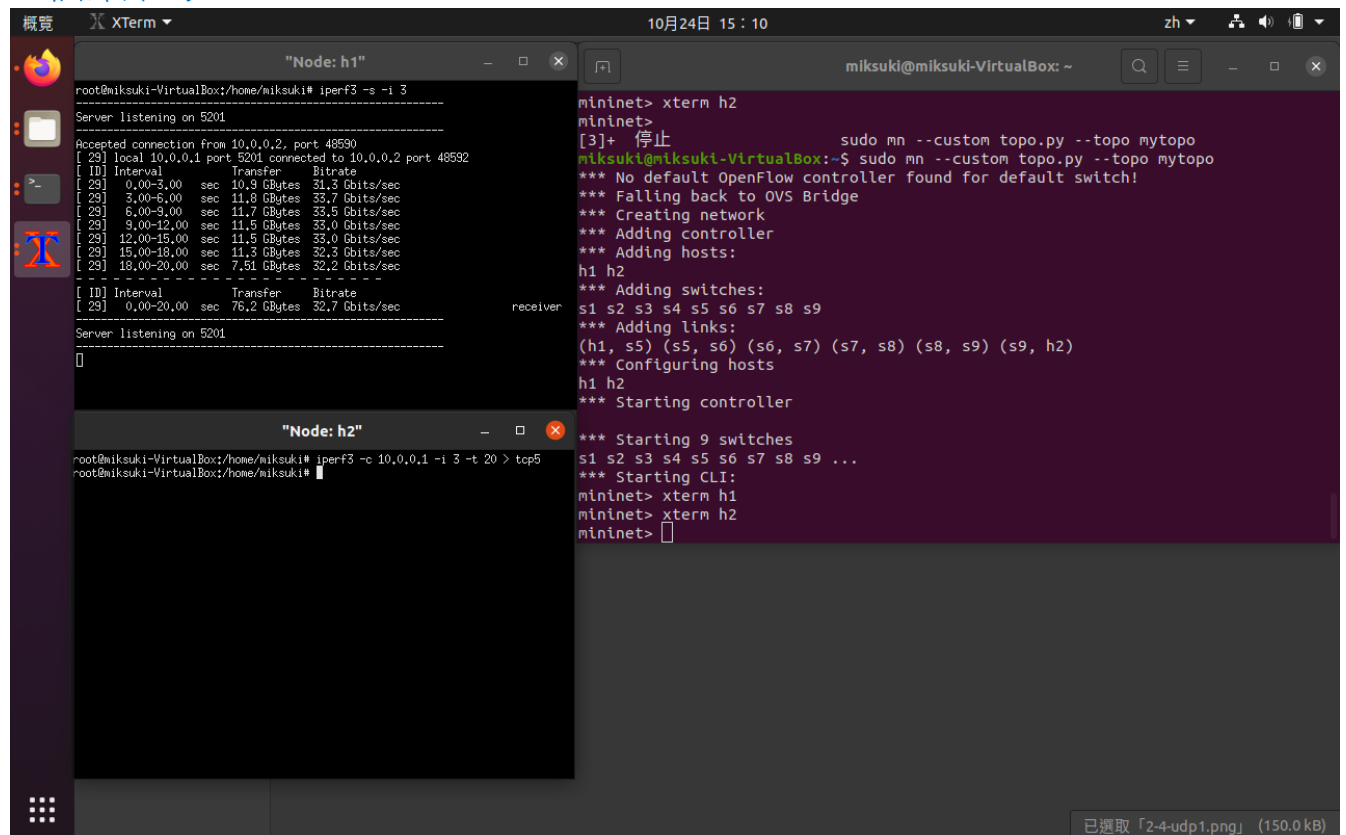
```
[ 29] 18.00-20.00 sec  7.67 GBytes  32.9 Gbits/sec
[ ID] Interval      Transfer     Bitrate
[ 29] 0.00-20.00 sec  78.1 GBytes  33.6 Gbits/sec
Server listening on 5201
Accepted connection from 10.0.0.2, port 48580
[ 29] local 10.0.0.1 port 5201 connected to 10.0.0.2 port 48582
[ ID] Interval      Transfer     Bitrate
[ 29] 0.00-3.00 sec  11.8 GBytes  33.7 Gbits/sec
[ 29] 3.00-6.00 sec  10.0 GBytes  28.7 Gbits/sec
[ 29] 6.00-9.00 sec  12.5 GBytes  35.7 Gbits/sec
[ 29] 9.00-12.00 sec  12.9 GBytes  37.0 Gbits/sec
[ 29] 12.00-15.00 sec  13.2 GBytes  37.7 Gbits/sec
[ 29] 15.00-18.00 sec  13.1 GBytes  37.5 Gbits/sec
[ 29] 18.00-20.00 sec  8.69 GBytes  37.3 Gbits/sec
[ ID] Interval      Transfer     Bitrate
[ 29] 0.00-20.00 sec  82.2 GBytes  35.3 Gbits/sec
Server listening on 5201
[]

root@miksuki-VirtualBox:/home/miksuki# iperf3 -c 10.0.0.1 -i 3 -t 20 > tcp3
root@miksuki-VirtualBox:/home/miksuki#

miksuki@miksuki-VirtualBox:~$
miksuki@miksuki-VirtualBox:~$
miksuki@miksuki-VirtualBox:~$ sudo mn --custom topo.py --topo mytopo
*** No default OpenFlow controller found for default switch!
*** Falling back to OVS Bridge
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7 s8 s9
*** Adding links:
(h1, s2) (s2, s3) (s3, s4) (s4, h2)
*** Configuring hosts
h1 h2
*** Starting controller
*** Starting 9 switches
s1 s2 s3 s4 s5 s6 s7 s8 s9 ...
*** Starting CLI:
mininet> xterm h1
mininet> xterm h2
mininet> []
```

已選取「2-4-tcp3.png」 (145.9 kB)

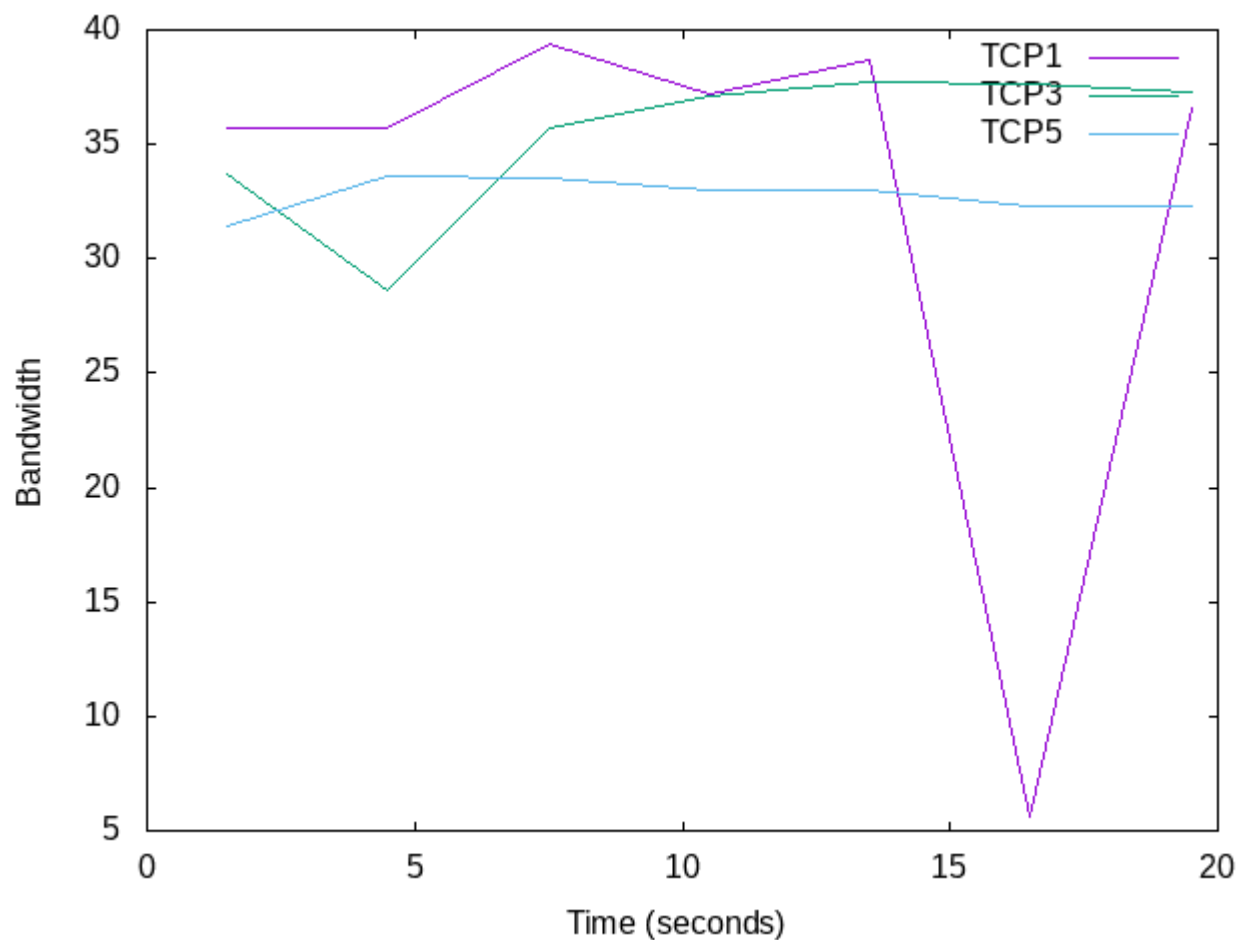
五個節點的 TCP



```
root@miksuki-VirtualBox:~/home/miksuki# iperf3 -s -i 3
Server listening on 5201
Accepted connection from 10.0.0.2, port 48590
[ 29] local 10.0.0.1 port 5201 connected to 10.0.0.2 port 48592
[ ID] Interval      Transfer     Bitrate
[ 29] 0.00-3.00  sec  10.9 GBytes  31.3 Gbits/sec
[ 29] 3.00-6.00  sec  11.8 GBytes  33.7 Gbits/sec
[ 29] 6.00-9.00  sec  11.7 GBytes  33.5 Gbits/sec
[ 29] 9.00-12.00 sec  11.5 GBytes  33.0 Gbits/sec
[ 29] 12.00-15.00 sec  11.5 GBytes  33.0 Gbits/sec
[ 29] 15.00-18.00 sec  11.3 GBytes  32.3 Gbits/sec
[ 29] 18.00-20.00 sec  7.51 GBytes  32.2 Gbits/sec
[ ID] Interval      Transfer     Bitrate
[ 29] 0.00-20.00  sec  76.2 GBytes  32.7 Gbits/sec
Server listening on 5201
[ 0]
```

```
mininet> xterm h2
mininet>
[3]+ 停止      sudo mn --custom topo.py --topo mytopo
miksuki@miksuki-VirtualBox:~$ sudo mn --custom topo.py --topo mytopo
*** No default OpenFlow controller found for default switch!
*** Falling back to OVS Bridge
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7 s8 s9
*** Adding links:
(h1, s5) (s5, s6) (s6, s7) (s7, s8) (s8, s9) (s9, h2)
*** Configuring hosts
h1 h2
*** Starting controller
*** Starting 9 switches
s1 s2 s3 s4 s5 s6 s7 s8 s9 ...
*** Starting CLI:
mininet> xterm h1
mininet> xterm h2
mininet>
```

TCP 繪圖結果

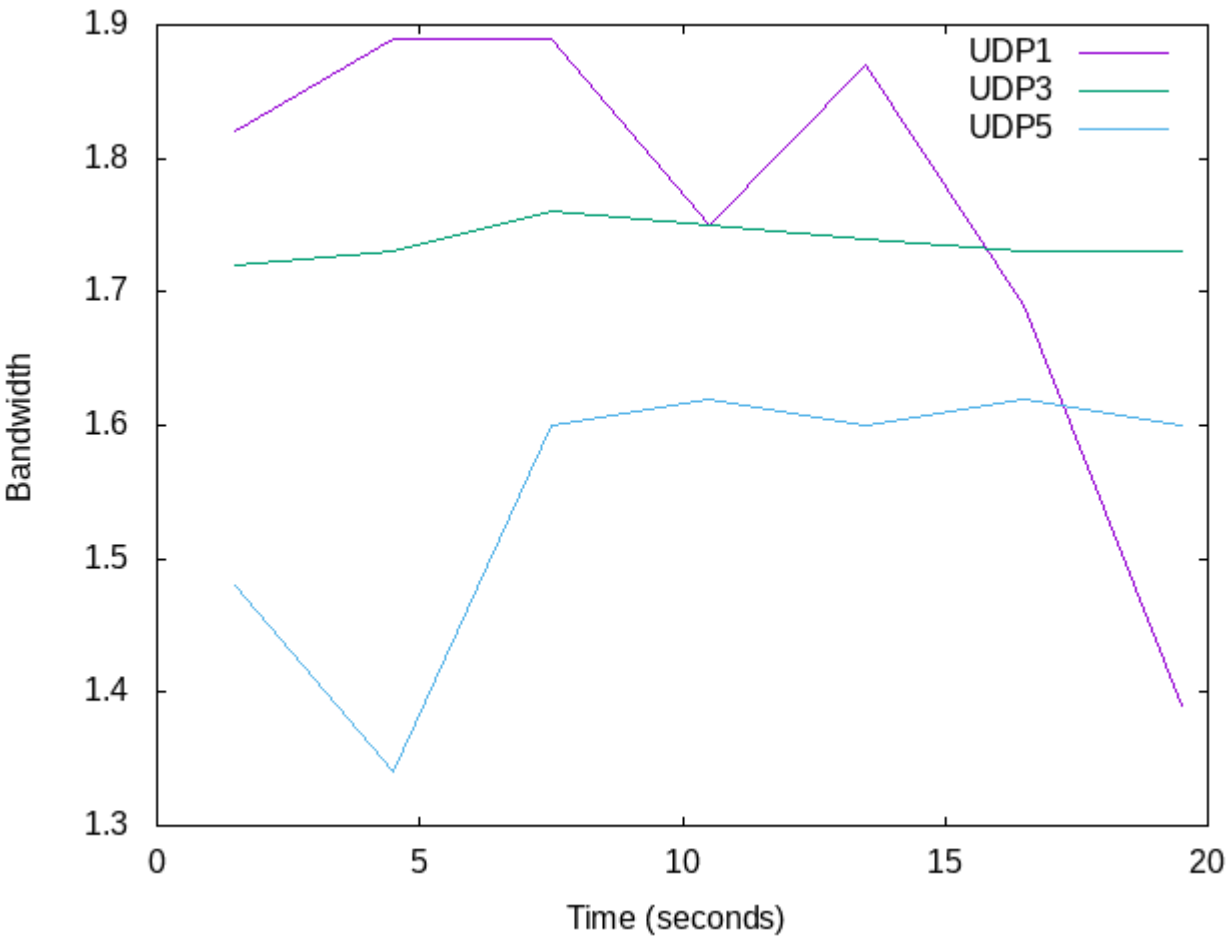


[illegible][illegible]

五個節點的 UDP

The image shows a Windows 10 desktop with a taskbar at the bottom containing icons for Edge, File Explorer, and other applications. Two virtual machine windows are open. The top window, titled "Node: h1", shows a terminal output for a network setup. It includes a table of network statistics for a receiver, followed by a series of commands and their outputs in a Mininet environment. The commands include starting the controller, adding switches (s1-s9), adding links, and starting the CLI. The bottom window, titled "Node: h2", shows a terminal prompt at the root of a Miksuki VM. It contains a list of commands for setting up network interfaces, installing iperf3, and running iperf3 tests between hosts h1 and h2. The desktop background is a dark blue gradient with a faint grid pattern.

UDP 繪圖結果



4.4

buffer size 會影響傳輸的 performance

當 buffer size 大的時候，傳輸量增加，但相對的延遲也高

而在 linux kernel 裡，TCP 追求的是高速、大量的資料傳輸，buffer size 大；

UDP 則是追求低延遲傳輸，buffer size 小，

所以會導致在大量傳輸時，TCP 比 UDP 快

5. netperf

請利用netperf 完成以下要求【自行使用mininet 產生hosts，利用xterm <host> 指令開啟個別 host 視窗，即可進行量測】

- 測量Client 與Server 間的TCP 網路效能
- 測量Client 與Server 間的UDP 網路效能
- 請寫出正確的指令並截圖證明。

Server 指令:

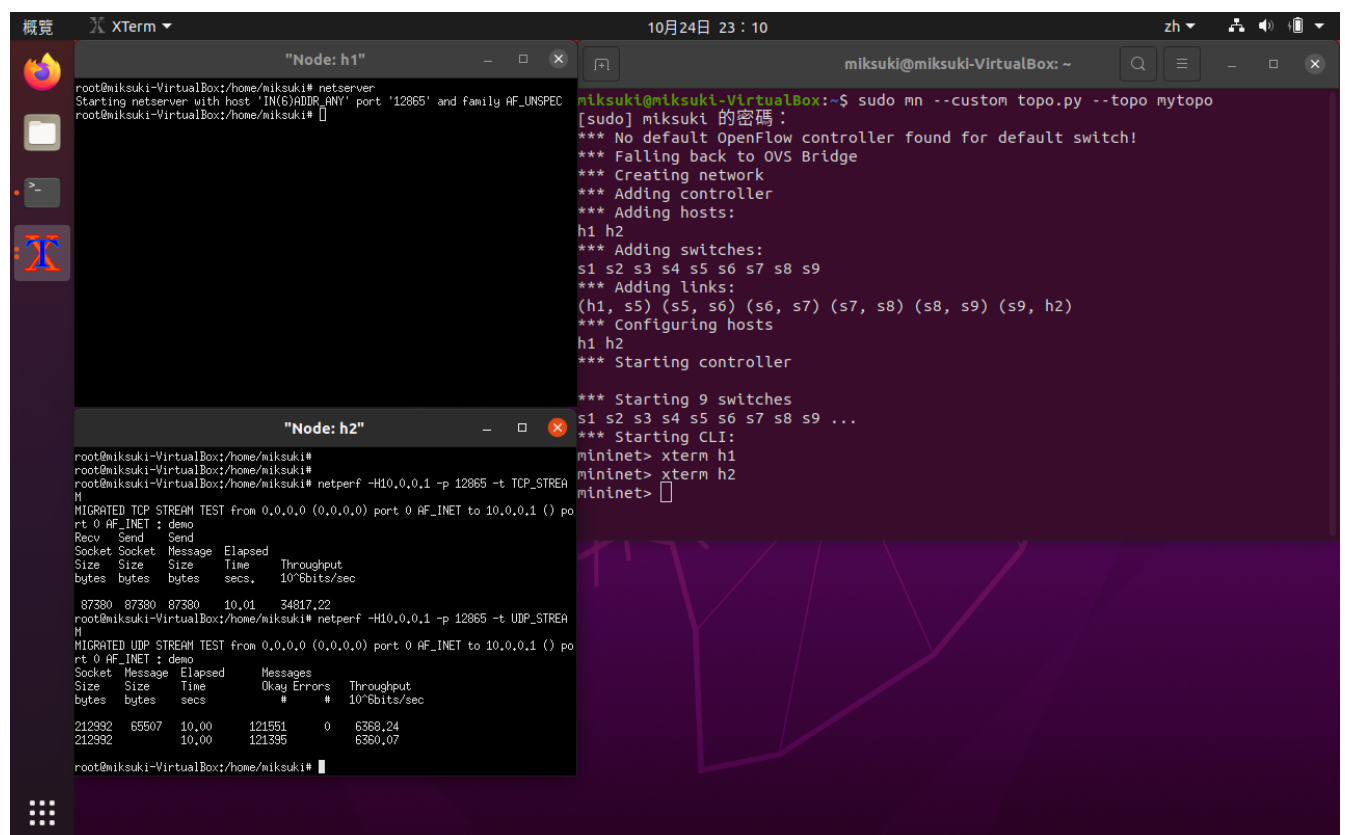
```
netserver
```

Client-TCP 指令:

```
netperf -H 10.0.0.1 -p 12865 -t TCP_STREAM
```

Client-UDP 指令:

```
netperf -H 10.0.0.1 -p 12865 -t UDP_STREAM
```



```
root@miksuki-VirtualBox:~# netserver
Starting netserver with host 'IN(6)ADDR_ANY' port '12865' and family AF_UNSPEC
root@miksuki-VirtualBox:~#

miksuki@miksuki-VirtualBox:~$ sudo mn --custom topo.py --topo mytopo
[sudo] miksuki 的密碼:
*** No default OpenFlow controller found for default switch!
*** Falling back to OVS Bridge
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7 s8 s9
*** Adding links:
(h1, s5) (s5, s6) (s6, s7) (s7, s8) (s8, s9) (s9, h2)
*** Configuring hosts
h1 h2
*** Starting controller
*** Starting 9 switches
s1 s2 s3 s4 s5 s6 s7 s8 s9 ...
*** Starting CLI:
mininet> xterm h1
mininet> xterm h2
mininet>

root@miksuki-VirtualBox:~# netperf -H 10.0.0.1 -p 12865 -t TCP_STREAM
MIGRATED TCP STREAM TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to 10.0.0.1 () port 0 AF_INET : demo
Recv Send Send
Socket Socket Message Elapsed
Size Size Size Time Throughput
bytes bytes bytes secs. 10^6bits/sec
87380 87380 87380 10.01 34817.22
root@miksuki-VirtualBox:~# netperf -H 10.0.0.1 -p 12865 -t UDP_STREAM
MIGRATED UDP STREAM TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to 10.0.0.1 () port 0 AF_INET : demo
Socket Message Elapsed Messages
Size Size Time # Errors Throughput
bytes bytes secs # # 10^6bits/sec
212932 65507 10.00 121561 0 6368.24
212932 65507 10.00 121395 0 6360.07
root@miksuki-VirtualBox:~#
```