TUGAS TERTULIS

JARINGAN KOMPUTER

SIMULASI MININET

Disusun untuk memenuhi Tugas Besar
Mata Kuliah Jaringan Komputer
Dosen Pengampu Immanuel Sembiring

Disusun Oleh :

1. Talitha Nabila 1301204516

Kelas IF-44-10
Fakultas Informatika
Jurusan S1 Informatika

DAFTAR PUSTAKA

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1
PENDAHULUAN

1.1 Tujuan Penulisan Laporan

Laporan ini disusun untuk memenuhi salah satu tugas mata kuliah Jaringan Komputer mengenai materi CLO-1, CLO-2, CLO-4, dan CLO-4 yang berisikan pembangunan topology dengan subnetting, uji konektivitas dengan routing, melakukan *capture traffic* dengan iPerf, dan inspeksi penggunaan *queue* pada router jaringan. Laporan ini dikerjakan berdasarkan spesifikasi tugas besar Jaringan Komputer yang telah diberikan.

1.2 Landasan Teori

1.2.1 Sistem Operasi Linux

Linux merupakan salah satu sistem operasi *open source* di bawah naungan lisensi GNU dan dibangun di atas kernel Linux. Sistem operasi yang terbuka menjadi salah satu sistem operasi yang dapat diunduh secara gratis menjadikan Linux populer di kalangan yang bergerak pada bidang *engineering* seperti *computing engineer* dan *network engineer.*

Linux pertama kali dibuat oleh Linus Torvalds pada tanggal 25 Agustus 1991 yang saat itu merupakan sebuah hobi belaka waktu menempuh jurusan ilmu komputer di Universitas Helsinki, Finlandia. Proses pengembangan linux yang awalnya sebuah hobi terus dikembangkan olehnya dan pada bulan Maret tahun 1994, Linux versi awal dirilis yaitu Linux versi 1.0. Linux dapat dikenal melalui logonya yang unik yaitu pinguin yang bernama Tux yang di desain oleh Larry Ewing pada tahun 1996. Logo 6 pinguin dipilih dikarenakan sang pencipta Linux itu sendiri yaitu Linus Torvalds menyukai pinguin.

Gambar 1 Logo Linux (Tux)

1.2.2 Ubuntu

Secara garis besar, Ubuntu merupakan sistem operasi turunan dari Linux yang diciptakan pada tanggal 20 Oktober 2004 oleh CEO perangkat lunak komputer swasta yang berbasis di Afrika Selatan yang bernama Mark Richard Shuttleworth. Proses pengembangan Ubuntu menurut sumber dari laman https://id.wikipedia.org/wiki/Ubuntu dilakukan tiap 6 bulan sekali untuk versi terbaru, tiap 18 bulan sekali untuk pembaharuan sistem, setiap 2 tahun sekali (versi xx.04 dengan x angka genap) akan mendapatkan Long Term Support(LTS) selama 3 tahun untuk desktop dan 5 tahun untuk edisi server. Terdapat 4 divisi resmi yang dikeluarkan oleh Ubuntu, yaitu Ubuntu Desktop, Ubuntu Server, Ubuntu for IoT, dan Ubuntu Cloud.

Gambar 2 Logo Ubuntu

1.2.3 Jaringan Komputer

Jaringan komputer adalah jaringan telekomunikasi yang memungkinkan antar komputer untuk saling berkomunikasi dengan bertukar data. Tujuan dari jaringan komputer adalah agar dapat mencapai tujuannya, setiap bagian dari jaringan komputer dapat meminta dan memberikan layanan 7 (service). Pihak yang meminta/menerima layanan disebut klien (client) dan yang memberikan/mengirim layanan disebut peladen (server). Desain ini disebut dengan sistem client-server, dan digunakan pada hampir seluruh aplikasi jaringan komputer. Selain itu, pengertian lain dari jaringan komputer adalah dua atau lebih komputer yang terhubung satu sama lain dan digunakan untuk berbagi data.

1.2.4 Mininet

Mininet adalah emulator berbasis CLI yang digunakan untuk membuat sebuah topologi jaringan pada Software Defined Network. Pada Mininetsudah terdapat beberapa topologi bawaan yang dapat langsung digunakan dengan menggunakan perintah (command) tertentu. Beberapa topologi bawaan tersebut antara lain topologi single, tree dan linear

1.2.5 Wireshark

Wireshark merupakan salah satu tools atau aplikasi “Network Analyzer” atau Penganalisa Jaringan. Penganalisaan Kinerja Jaringan itu dapat melingkupi berbagai hal, mulai dari proses menangkap paket-paket data atau informasi yang berlalu-lalang dalam jaringan, sampai pada digunakan pula untuk sniffing (memperoleh informasi penting seperti password email, dll). Wireshark sendiri merupakan free tools untuk Network Analyzer yang ada saat ini. Dan tampilan dari wireshark ini sendiri terbilang sangat bersahabat dengan user karena menggunakan tampilan grafis atau GUI (Graphical User Interface).

Gambar 3 Logo Wireshark

1.2.6 TCPDump

TCPDump adalah sebuah tools command line atau terminal paket sniffer atau paket analisis yang biasa dipakai untuk menyaring paket tcp/ip yang diterima atau yang dikirimkan melalui jaringan pada interface tertentu. TCPdump banyak digunakan untuk mencari masalah-masalah di jaringan atau untuk memonitor aktivitas jaringan, Tcpdump menggunakan API pcap (packet capture) yaitu LibPcap (Unix) atau winPcap (windows) untuk menangkap paket. Data hasil capture tcpdump sering disebut sebagai dumpfile. TCPdump banyak digunakan untuk mencari masalahmasalah di jaringan atau untuk memonitor aktivitas jaringan, Tcpdump menggunakan API pcap (packet capture) yaitu LibPcap (Unix) atau winPcap (windows) untuk menangkap paket. Data hasil capture tcpdum sering disebut sebagai dumpfile.

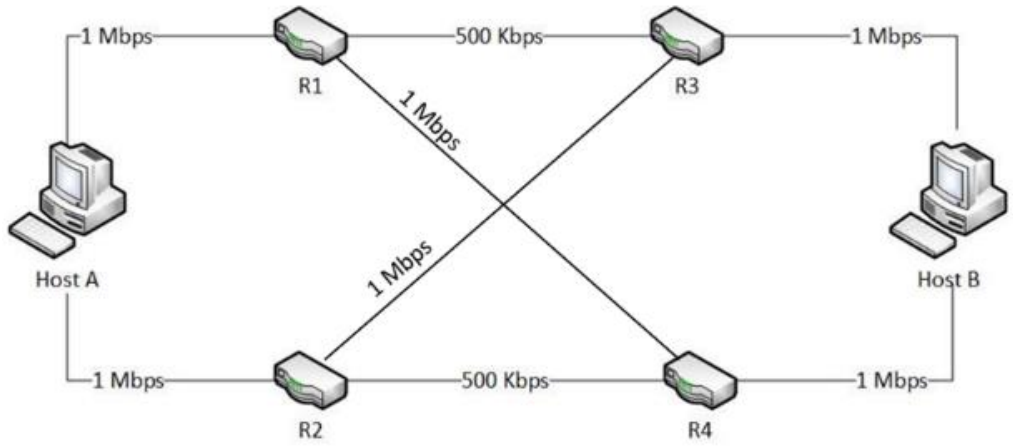
1.2.7 Packet Tracer

Packet Tracer adalah alat simulasi visual lintas platform yang dirancang oleh Cisco Systems yang memungkinkan pengguna membuat topologi jaringan dan meniru jaringan komputer modern. Perangkat lunak ini memungkinkan pengguna untuk mensimulasikan konfigurasi router dan switch. Cisco menggunakan antarmuka command line yang disimulasikan.

BAB 2
PEMBAHASAN

2.1 Spesifikasi Tugas Besar

Spesifikasi tugas besar yaitu memenuhi 4 CLO yaitu pembangunan topology dengan subnetting, uji konektivitas dengan routing, melakukan *capture traffic* dengan iPerf, dan inspeksi penggunaan *queue* pada router jaringan. Tugas besar ini memiliki topologi yang disediakan sebagai berikut.

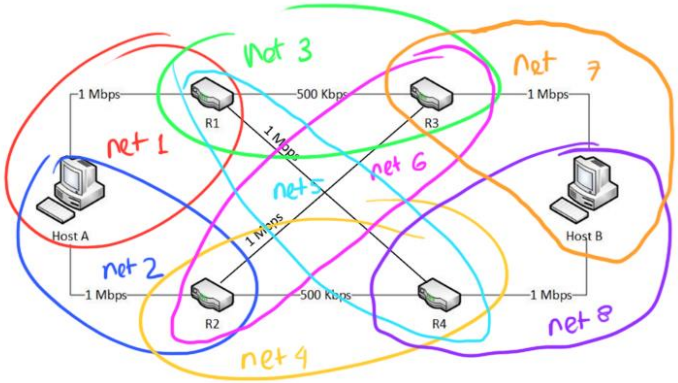


Gambar 4 Topologi untuk Tugas Besar

2.2 CLO 1

Pada CLO 1 akan melakukan pembangunan topology jaringan sesuai dengan spesifikasi yang diberikan. Pada langkah pertama akan melakukan desain subnet masing-masing network menggunakan VLSM agar menghasilkan IP subnet yang digunakan semua.

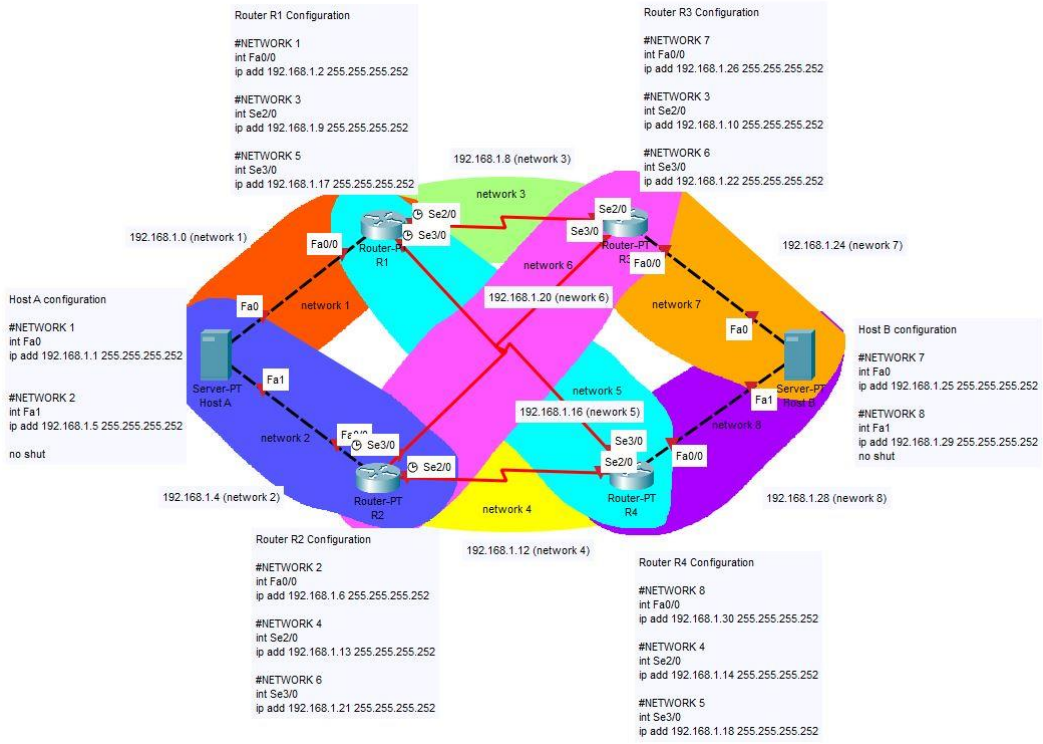
2.2.1 Subnetting



Gambar 5 Pembagian Network

Subnet Name	Needed Size	Allocated Size	Address	Mask	Dec Mask	Assignable Range	Broadcast
Net 1	2	2	192.168.1.0	/30	255.255.255.252	192.168.1.1 - 192.168.1.2	192.168.1.3
Net 2	2	2	192.168.1.4	/30	255.255.255.252	192.168.1.5 - 192.168.1.6	192.168.1.7
Net 3	2	2	192.168.1.8	/30	255.255.255.252	192.168.1.9 - 192.168.1.10	192.168.1.11
Net 4	2	2	192.168.1.12	/30	255.255.255.252	192.168.1.13 - 192.168.1.14	192.168.1.15
Net 5	2	2	192.168.1.16	/30	255.255.255.252	192.168.1.17 - 192.168.1.18	192.168.1.19
Net 6	2	2	192.168.1.20	/30	255.255.255.252	192.168.1.21 - 192.168.1.22	192.168.1.23
Net 7	2	2	192.168.1.24	/30	255.255.255.252	192.168.1.25 - 192.168.1.26	192.168.1.27
Net 8	2	2	192.168.1.28	/30	255.255.255.252	192.168.1.29 - 192.168.1.30	192.168.1.31

Tabel 1 Table Subnetting



Gambar 6 Topologi pada Packet Tracer

Pada subnetting digunakan network utama dengan IP 192.168.1.0, dalam topologi ini dibutuhkan 16 IP address untuk ke-6 node yang disediakan. Jumlah IP address ini ditentukan karena kebutuhan 8 jaringan atau network untuk menghubungan node satu sama lain. Pada setiap network tersebut akan membutuhkan 2 IP address.

Dengan kebutuhan 2 IP address pada setiap jaringan, maka digunakanlah prefix CIDR bernilai /30 yang akan menyediakan available IP berjumlah 2, network address berjumlah 1, dan broadcast address berjumlah 1. Pemilihan prefix CIDR ini juga berpengaruh untuk memperkecil jangkauan IP agar tidak melakukan pemborosan IP. Dengan itu, penggunaan prefic /30 paling cocok karena semua IP subnet yang tersedia akan digunakan.

2.2.2 Membangun Topologi Network

```python
class MyTopo(Topo):       You, 2 weeks ago • Base Topology
    '''Topology to be instantiated in Mininet'''
    def __init__(self, **opts):
        # Initialize topology and default options
        Topo.__init__(self, **opts)

        #Define bandwidth
        bw1 = 1
        bw2 = 0.5

        #Creating the nodes/hosts
        hostA = self.addHost('hostA')
        hostB = self.addHost('hostB')

        #Creating the routers
        r1 = self.addHost('r1')
        r2 = self.addHost('r2')
        r3 = self.addHost('r3')
        r4 = self.addHost('r4')

        #Creating the links between host and routers

        #Host A
        self.addLink(hostA, r1,
                     intfName1 = 'hostA-fa0',
                     intfName2 = 'r1-fa0',
                     cls = TCLink,
                     bw=bw1) #network 1
        self.addLink(hostA, r2,
                     intfName1 = 'hostA-fa1',
                     intfName2 = 'r2-fa0',
                     cls = TCLink,
                     bw=bw1) #network 2
        #Host B
        self.addLink(hostB, r3,
                     intfName1 = 'hostB-fa0',
                     intfName2 = 'r3-fa0',
                     cls = TCLink,
                     bw=bw1) #network 7
        self.addLink(hostB, r4,
                     intfName1 = 'hostB-fa1',
                     intfName2 = 'r4-fa0',
                     cls = TCLink,
                     bw=bw1) #network 8

        #Creating the links between routers

        #Network 3
        self.addLink(r1, r3, intfName1='r1-se2', intfName2='r3-se2', cls = TCLink, bw=bw2)
        #Network 5
        self.addLink(r1, r4, intfName1='r1-se3', intfName2='r4-se3', cls = TCLink, bw=bw1)

        #Network 4
        self.addLink(r2, r4, intfName1='r2-se2', intfName2='r4-se2', cls = TCLink, bw=bw2)
        #Network 6
        self.addLink(r2, r3, intfName1='r2-se3', intfName2='r3-se3', cls = TCLink, bw=bw1)
```

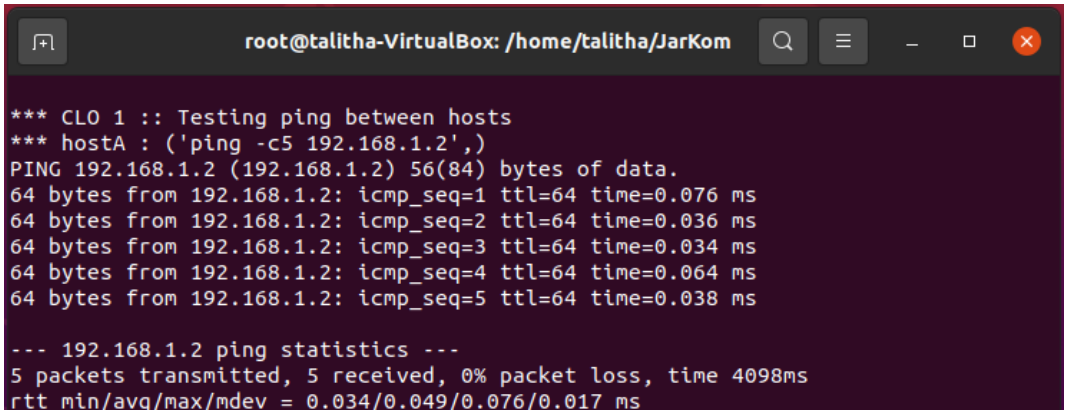
2.2.3 Assign IP Address

```
72  def assign_IP(h1,h2,r1,r2,r3,r4):       You, 2 days ago • cleaning code & hardcode
73      '''Assign IP addresses to the hosts & routers'''
74
75      #Configure IP addresses for hosts
76
77      #define NIC for hostA
78      h1.cmd('ifcongfig hostA-fa0 0')
79      h1.cmd('ifcongfig hostA-fa1 0')
80      #define IP address for hostA interfaces
81      h1.cmd('ifconfig hostA-fa0 192.168.1.1 netmask 255.255.255.252') #network 1
82      h1.cmd('ifconfig hostA-fa1 192.168.1.5 netmask 255.255.255.252') #network 2
83
84      #define NIC for hostB
85      h2.cmd('ifcongfig hostB-fa0 0')
86      h2.cmd('ifcongfig hostB-fa1 0')
87      #define IP address for hostB interfaces
88      h2.cmd('ifconfig hostB-fa0 192.168.1.25 netmask 255.255.255.252') #network 7
89      h2.cmd('ifconfig hostB-fa1 192.168.1.29 netmask 255.255.255.252') #network 8
90
91      #Enabling routers IP Forwarding for hosts
92      r1.cmd('sysctl -w net.ipv4.ip_forward=1')
93      r2.cmd('sysctl -w net.ipv4.ip_forward=1')
94      r3.cmd('sysctl -w net.ipv4.ip_forward=1')
95      r4.cmd('sysctl -w net.ipv4.ip_forward=1')
96
97      #Configure IP addresses for routers
98
99      #define NIC for r1
100     r1.cmd('ifcongfig r1-fa0 0')
101     r1.cmd('ifcongfig r1-se2 0')
102     r1.cmd('ifcongfig r1-se3 0')
103     #define IP address for r1 interfaces
104     r1.cmd('ifconfig r1-fa0 192.168.1.2 netmask 255.255.255.252') #network 1
105     r1.cmd('ifconfig r1-se2 192.168.1.9 netmask 255.255.255.252') #network 3
106     r1.cmd('ifconfig r1-se3 192.168.1.17 netmask 255.255.255.252') #network 5
107
108     #define NIC for r2
109     r2.cmd('ifcongfig r2-fa0 0')
110     r2.cmd('ifcongfig r2-se2 0')
111     r2.cmd('ifcongfig r2-se3 0')
112     #define IP address for r2 interfaces
113     r2.cmd('ifconfig r2-fa0 192.168.1.6 netmask 255.255.255.252') #network 2
114     r2.cmd('ifconfig r2-se2 192.168.1.13 netmask 255.255.255.252') #network 4
115     r2.cmd('ifconfig r2-se3 192.168.1.21 netmask 255.255.255.252') #network 6
```

```
116
117     #define NIC for r3
118     r3.cmd('ifcongfig r3-fa0 0')
119     r3.cmd('ifcongfig r3-se2 0')
120     r3.cmd('ifcongfig r3-se3 0')
121     #define IP address for r3 interfaces
122     r3.cmd('ifconfig r3-fa0 192.168.1.26 netmask 255.255.255.252') #network 7
123     r3.cmd('ifconfig r3-se2 192.168.1.10 netmask 255.255.255.252') #network 3
124     r3.cmd('ifconfig r3-se3 192.168.1.22 netmask 255.255.255.252') #network 6
125
126     #define NIC for r4
127     r4.cmd('ifcongfig r4-fa0 0')
128     r4.cmd('ifcongfig r4-se2 0')
129     r4.cmd('ifcongfig r4-se3 0')
130     #define IP address for r4 interfaces
131     r4.cmd('ifconfig r4-fa0 192.168.1.30 netmask 255.255.255.252') #network 8
132     r4.cmd('ifconfig r4-se2 192.168.1.14 netmask 255.255.255.252') #network 4
133     r4.cmd('ifconfig r4-se3 192.168.1.18 netmask 255.255.255.252') #network 5
```

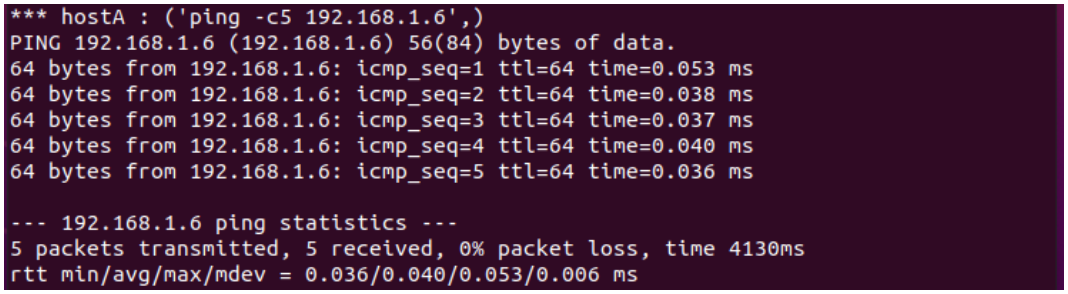
2.2.4 Uji Konektivitas Subnet Sama

2.2.4.1 Konektivitas Network 1



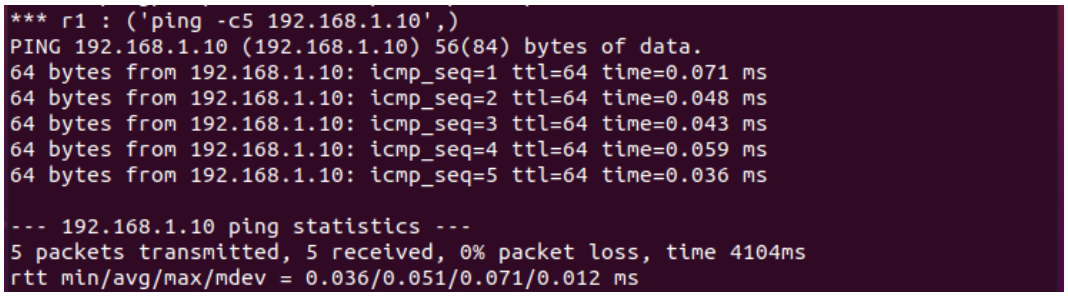
Gambar 7 Konektivitas Network 1

2.2.4.2 Konektivitas Network 2



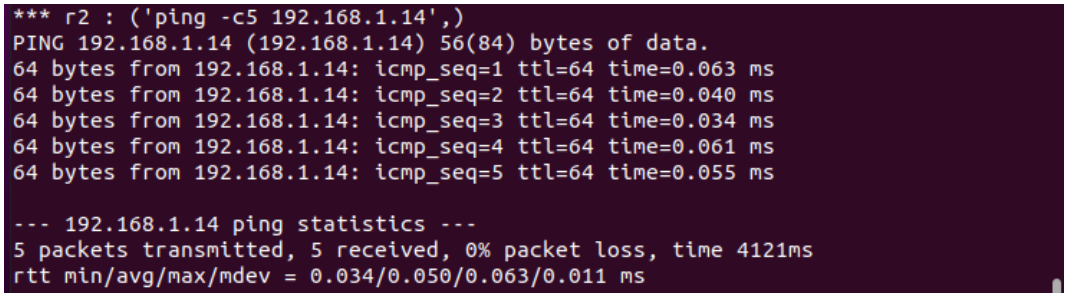
Gambar 8 Konektivitas Network 2

2.2.4.3 Konektivitas Network 3



Gambar 9 Konektivitas Network 3

2.2.4.4 Konektivitas Network 4



Gambar 10 Konektivitas Network 4

2.2.4.5 Konektivitas Network 5

```
*** r1 : ('ping -c5 192.168.1.18',)
PING 192.168.1.18 (192.168.1.18) 56(84) bytes of data.
64 bytes from 192.168.1.18: icmp_seq=1 ttl=64 time=0.044 ms
64 bytes from 192.168.1.18: icmp_seq=2 ttl=64 time=0.038 ms
64 bytes from 192.168.1.18: icmp_seq=3 ttl=64 time=0.036 ms
64 bytes from 192.168.1.18: icmp_seq=4 ttl=64 time=0.046 ms
64 bytes from 192.168.1.18: icmp_seq=5 ttl=64 time=0.038 ms

--- 192.168.1.18 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4108ms
rtt min/avg/max/mdev = 0.036/0.040/0.046/0.004 ms
```

Gambar 11 Konektivitas Network 5

2.2.4.6 Konektivitas Network 6

```
*** r2 : ('ping -c5 192.168.1.22',)
PING 192.168.1.22 (192.168.1.22) 56(84) bytes of data.
64 bytes from 192.168.1.22: icmp_seq=1 ttl=64 time=0.054 ms
64 bytes from 192.168.1.22: icmp_seq=2 ttl=64 time=0.046 ms
64 bytes from 192.168.1.22: icmp_seq=3 ttl=64 time=0.037 ms
64 bytes from 192.168.1.22: icmp_seq=4 ttl=64 time=0.070 ms
64 bytes from 192.168.1.22: icmp_seq=5 ttl=64 time=0.039 ms

--- 192.168.1.22 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4111ms
rtt min/avg/max/mdev = 0.037/0.049/0.070/0.012 ms
```

Gambar 12 Konektivitas Network 6

2.2.4.7 Konektivitas Network 7

```
*** hostB : ('ping -c5 192.168.1.26',)
PING 192.168.1.26 (192.168.1.26) 56(84) bytes of data.
64 bytes from 192.168.1.26: icmp_seq=1 ttl=64 time=0.051 ms
64 bytes from 192.168.1.26: icmp_seq=2 ttl=64 time=0.039 ms
64 bytes from 192.168.1.26: icmp_seq=3 ttl=64 time=0.037 ms
64 bytes from 192.168.1.26: icmp_seq=4 ttl=64 time=0.035 ms
64 bytes from 192.168.1.26: icmp_seq=5 ttl=64 time=0.060 ms

--- 192.168.1.26 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4123ms
rtt min/avg/max/mdev = 0.035/0.044/0.060/0.009 ms
```

Gambar 13 Konektivitas Network 7

2.2.4.8 Konektivitas Network 8

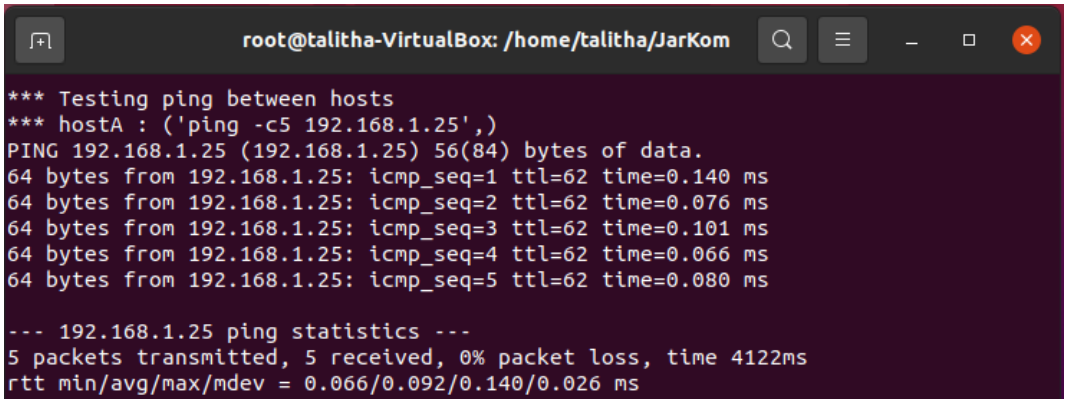
```
*** hostB : ('ping -c5 192.168.1.30',)
PING 192.168.1.30 (192.168.1.30) 56(84) bytes of data.
64 bytes from 192.168.1.30: icmp_seq=1 ttl=64 time=0.069 ms
64 bytes from 192.168.1.30: icmp_seq=2 ttl=64 time=0.098 ms
64 bytes from 192.168.1.30: icmp_seq=3 ttl=64 time=0.037 ms
64 bytes from 192.168.1.30: icmp_seq=4 ttl=64 time=0.049 ms
64 bytes from 192.168.1.30: icmp_seq=5 ttl=64 time=0.036 ms

--- 192.168.1.30 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4110ms
rtt min/avg/max/mdev = 0.036/0.057/0.098/0.023 ms
```

Gambar 14 Konektivitas Network 8

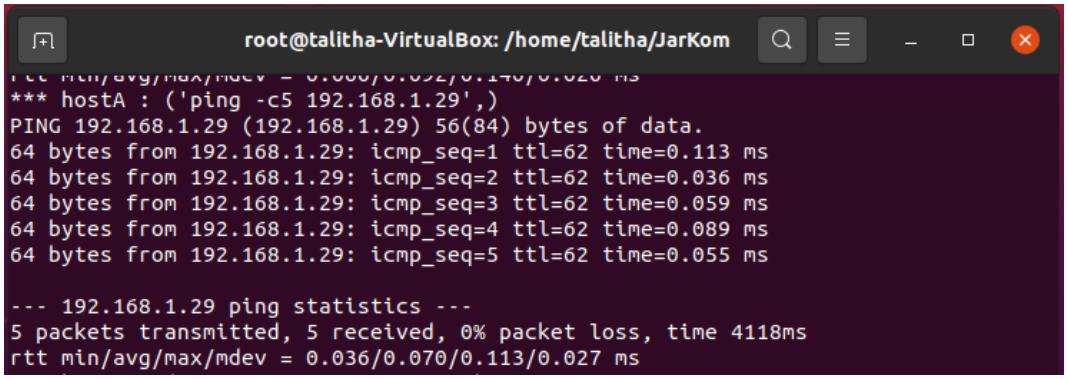
2.3 CLO 2 – Static Routing

2.3.1 Uji Konektivitas Static Routing

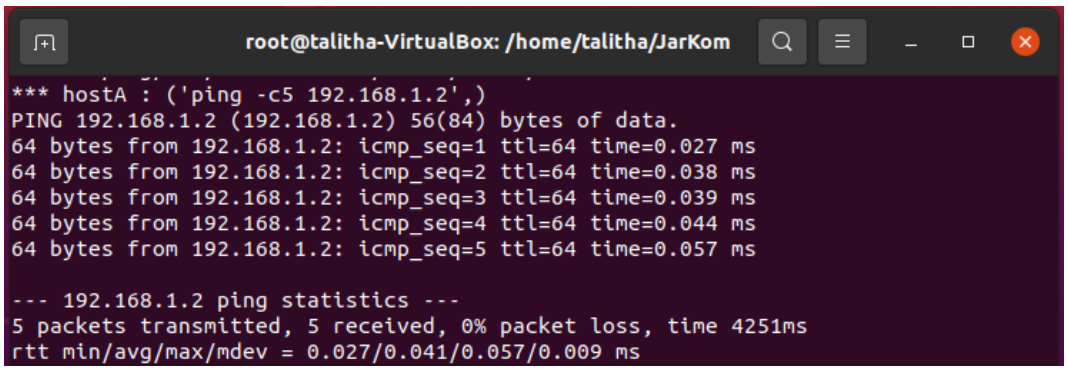
2.3.1.1 HostA to All



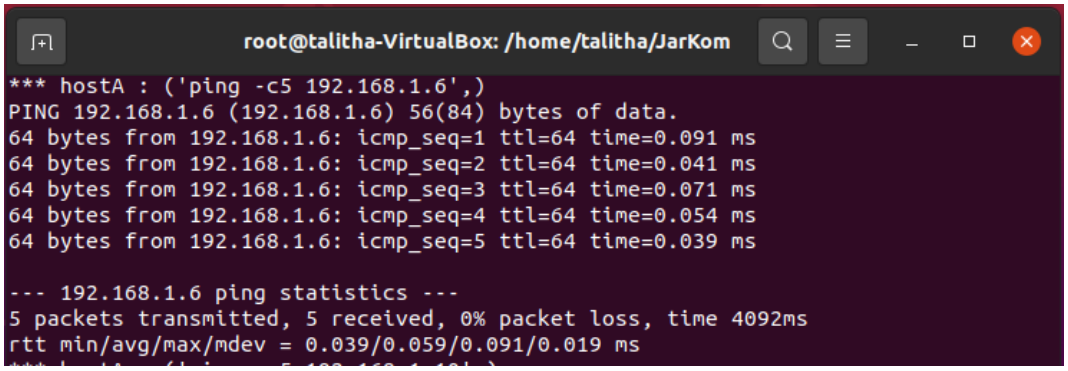
Gambar 15 hostA ping hostB (via network 7)



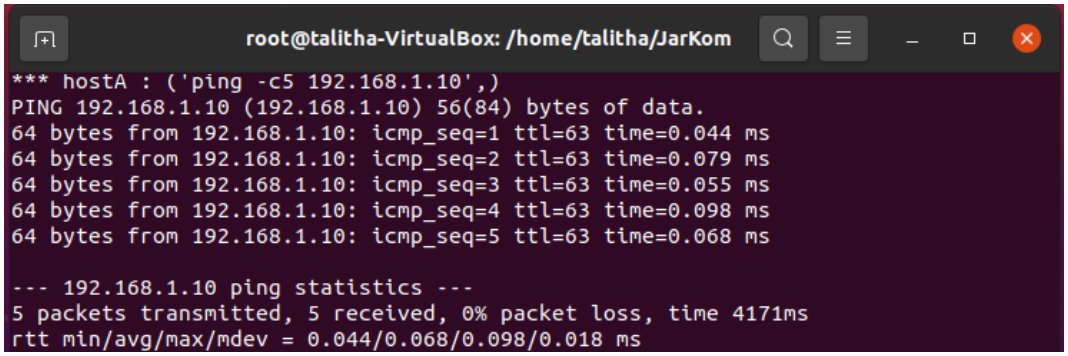
Gambar 16 hostA ping hostB (via network 8)



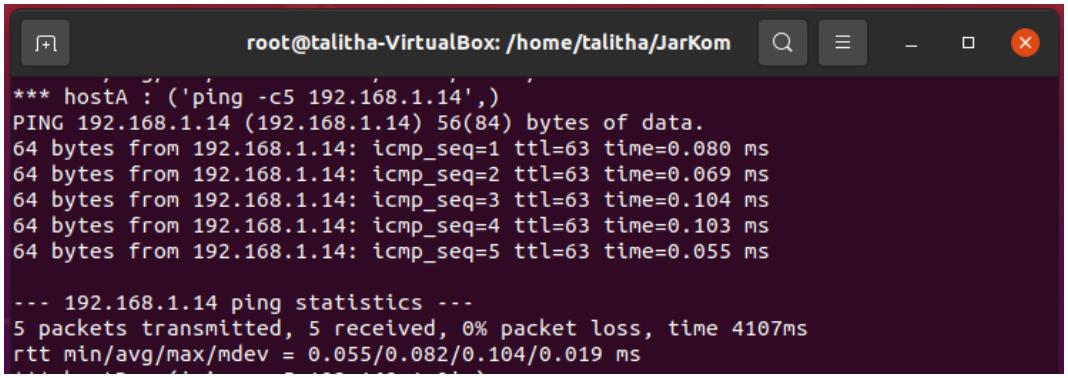
Gambar 17 hostA ping r1 (network 1)



Gambar 18 hostA ping r2 (network 2)

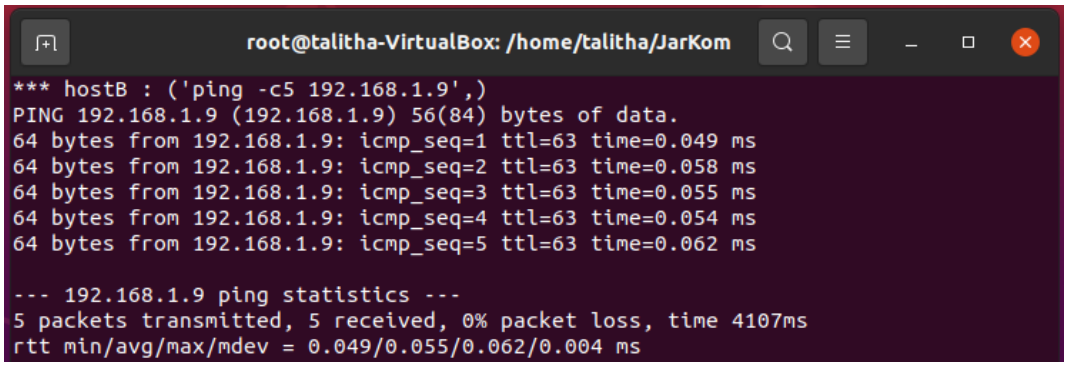


Gambar 19 hostA ping r3 (via network 3)

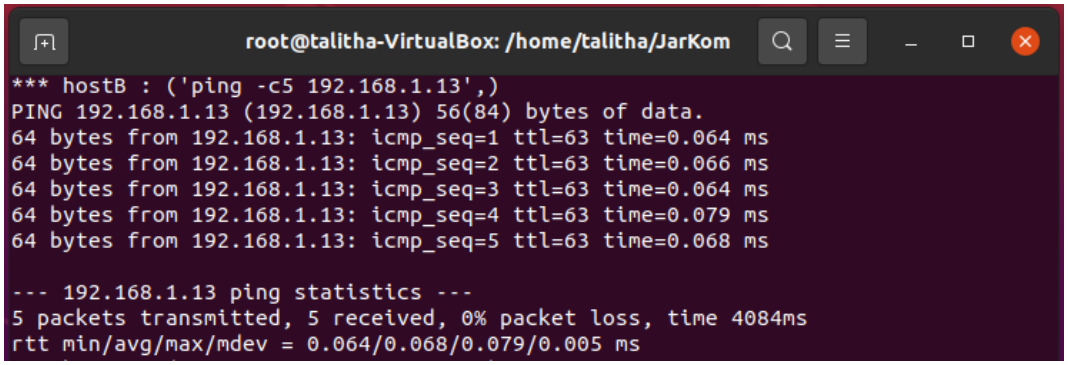


Gambar 20 hostA ping r4 (via network 5)

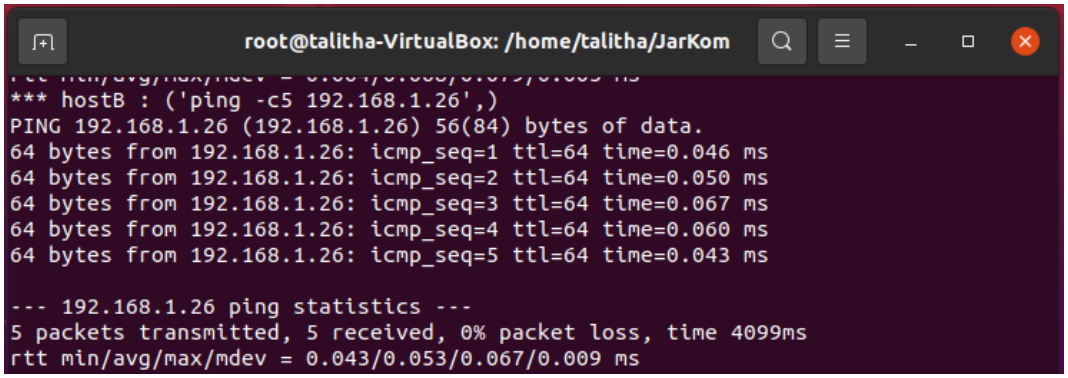
2.3.1.2 HostB to All Routers



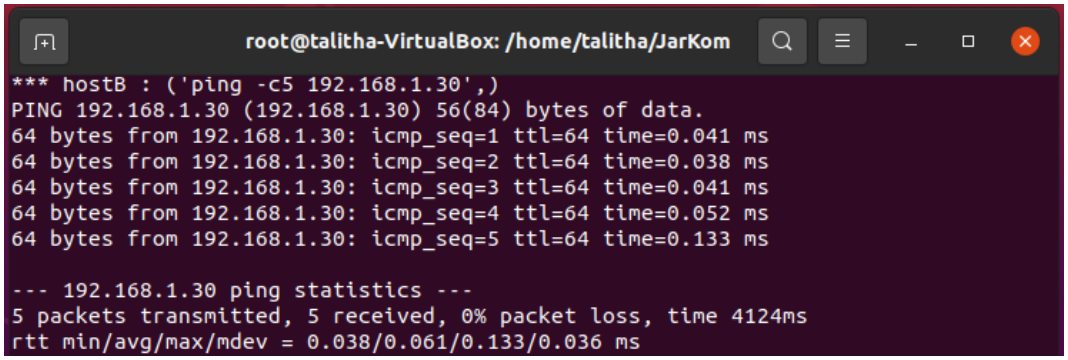
Gambar 21 hostB ping r1 (via network 3)



Gambar 22 hostB ping r2 (via network 6)

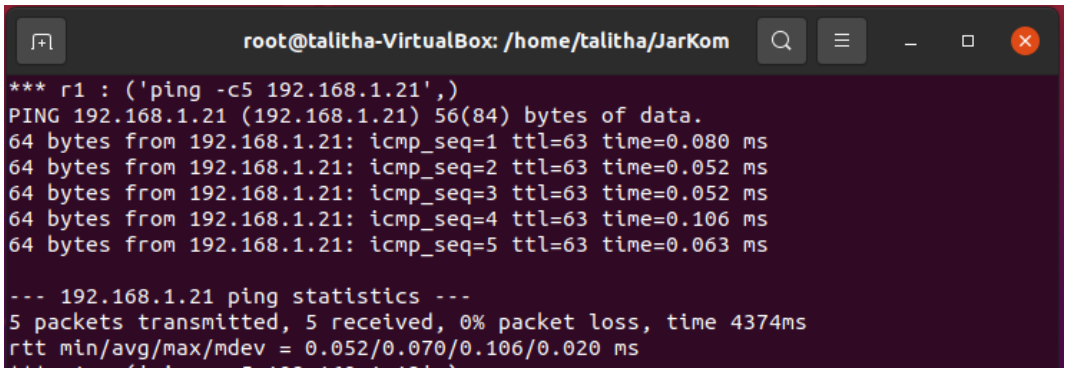


Gambar 23 hostB ping r3 (network 7)

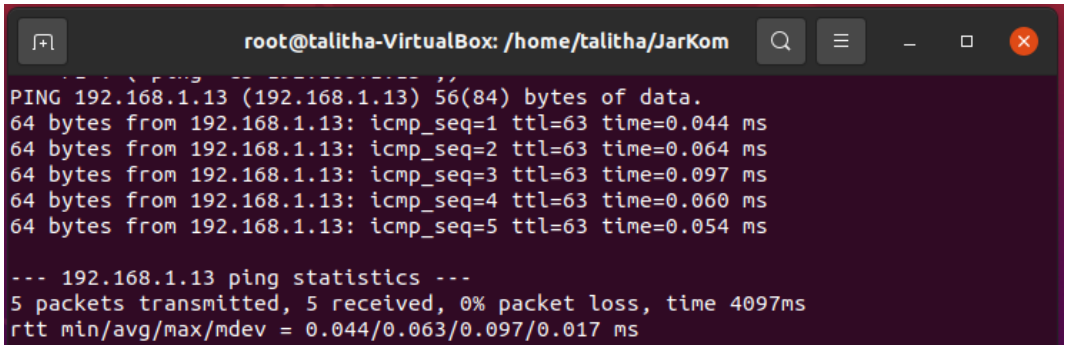


Gambar 24 hostB ping r4 (network 8)

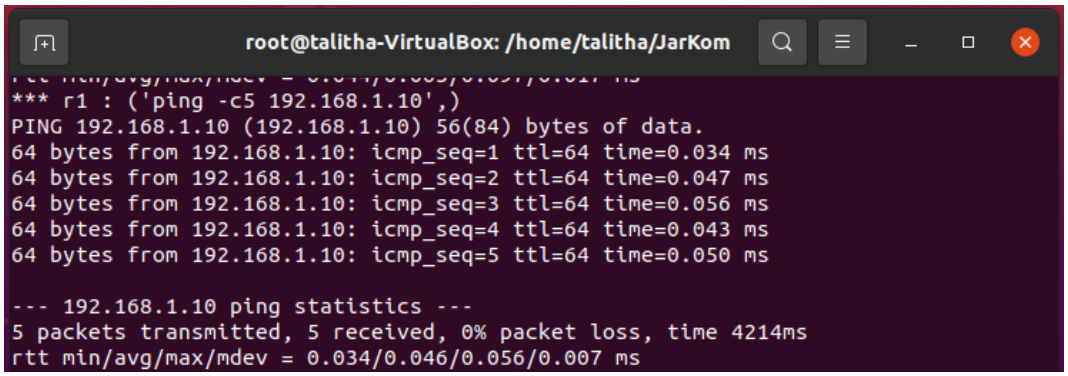
2.3.1.3 Router r1 to All Routers



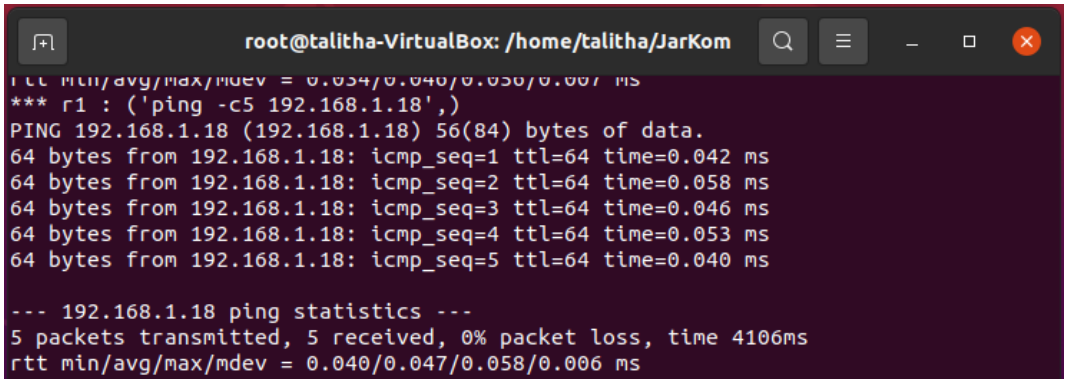
Gambar 25 r1 ping r2 (via network 6)



Gambar 26 r1 ping r2 (via network 4)

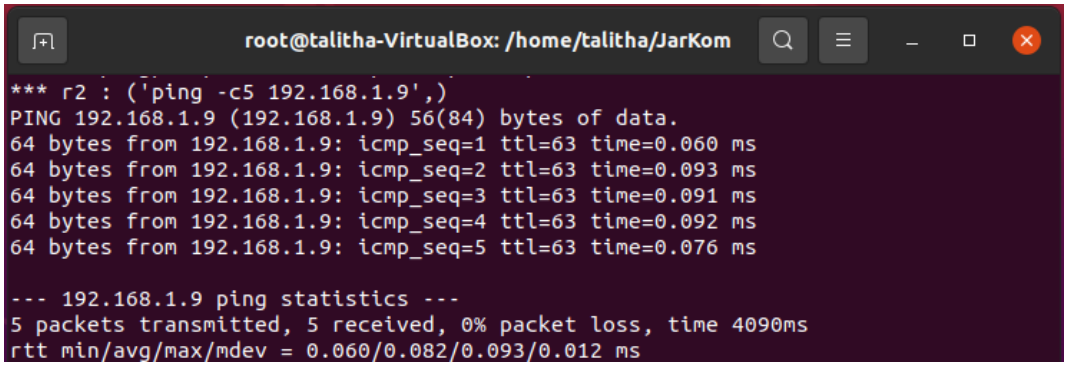


Gambar 27 r1 ping r3 (network 3)

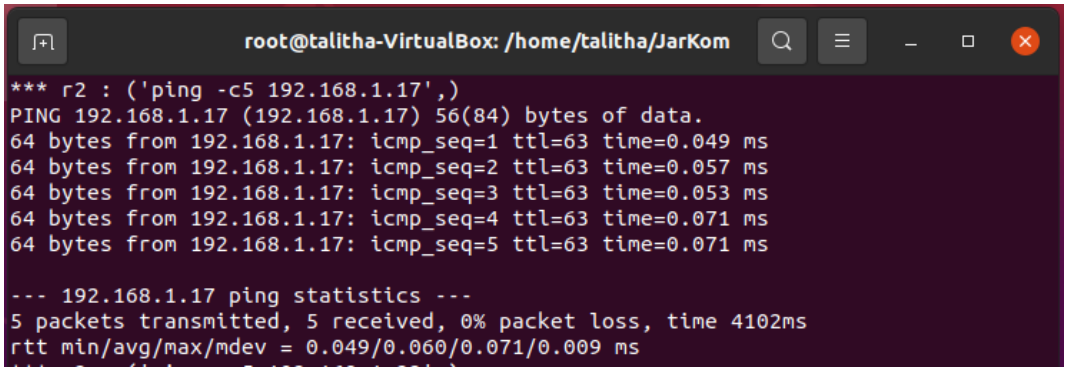


Gambar 28 r1 ping r4 (network 5)

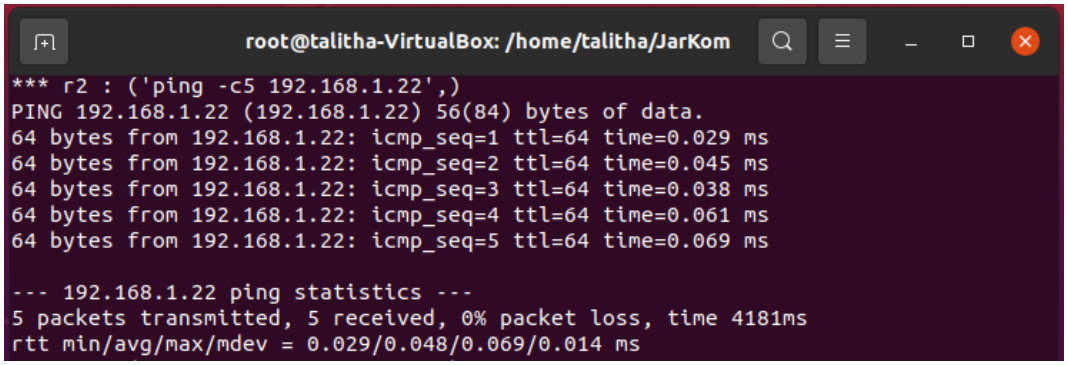
2.3.1.4 Router r2 to All Routers



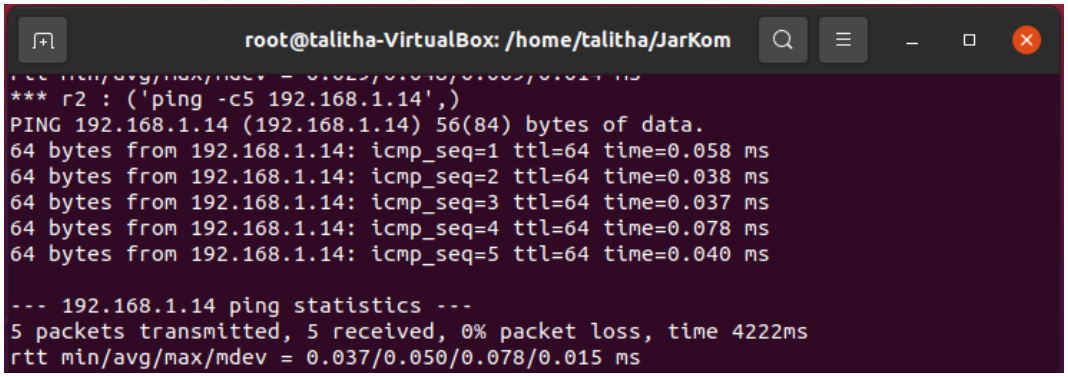
Gambar 29 r2 ping r1 (via network 3)



Gambar 30 r2 ping r1 (via network 5)

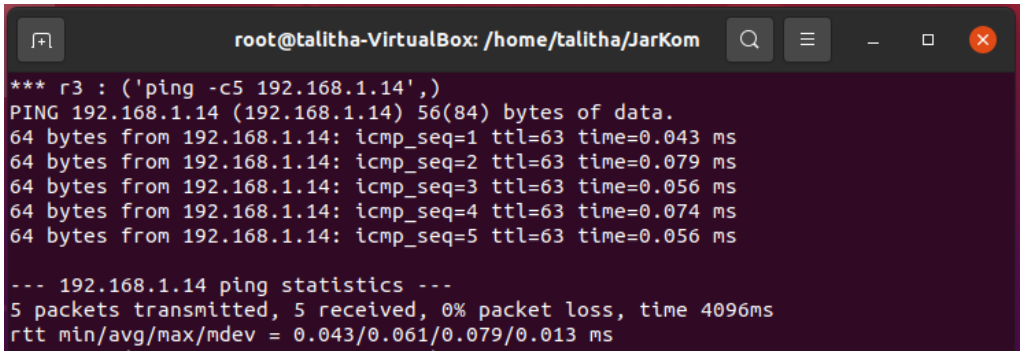


Gambar 31 r2 ping r3 (network 6)

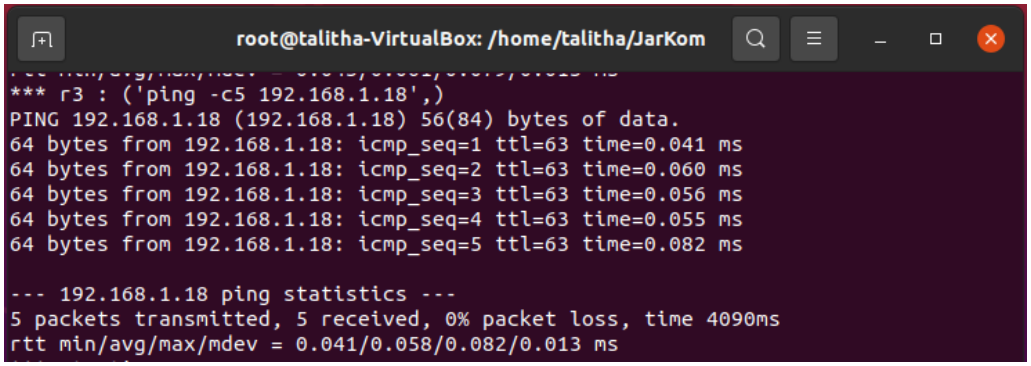


Gambar 32 r2 ping r4 (network 4)

2.3.1.4 Router r3 to r4



Gambar 33 r3 ping r4 (via network 4)



Gambar 34 r3 ping r4 (via network 5)

2.3.2 Traceroute HostA dan HostB

```
mininet> hostA traceroute hostB
traceroute to 192.168.1.25 (192.168.1.25), 30 hops max, 60 byte packets
 1  192.168.1.2 (192.168.1.2)  0.755 ms  0.673 ms  0.630 ms
 2  192.168.1.10 (192.168.1.10)  0.607 ms  0.556 ms  0.526 ms
 3  192.168.1.25 (192.168.1.25)  0.492 ms  0.418 ms  0.381 ms
mininet>
```

Gambar 35 hostA traceroute hostB

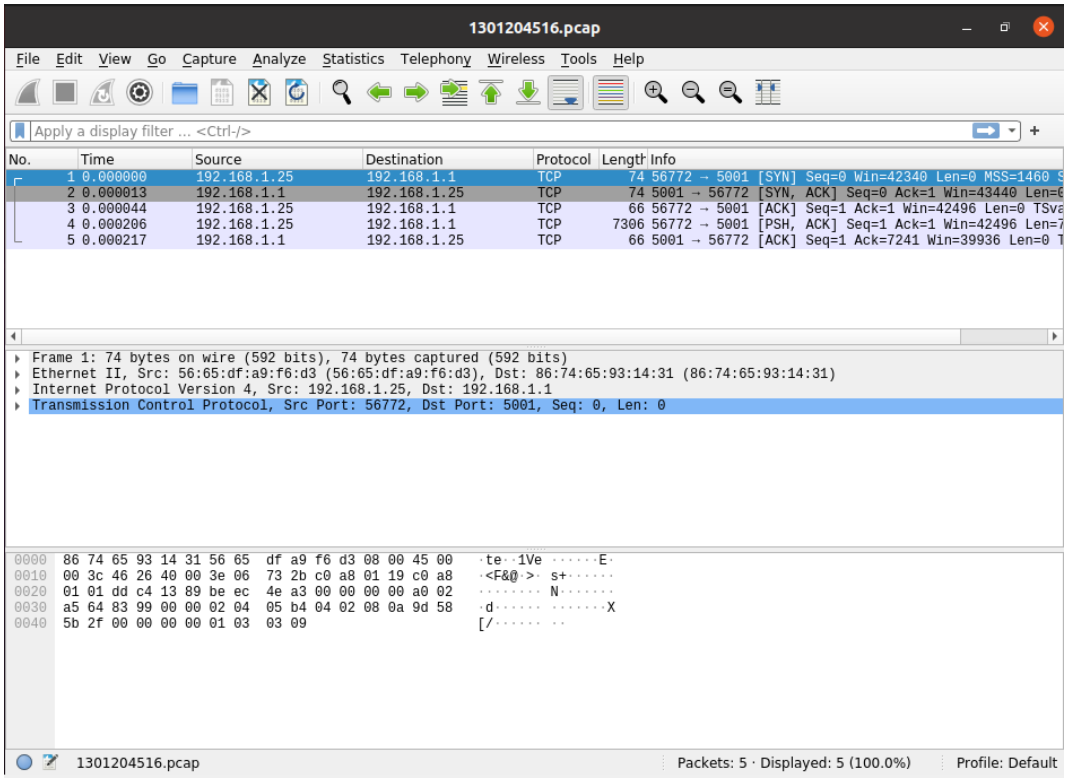
2.4 CLO 3 Capture iPerf

```
330 def tcp_traffic(net):
331     h1, h2 = net.get('hostA', 'hostB')
332     h1.cmd('iperf -s &')
333     h1.cmd('tcpdump tcp -c 5 -w 1301204516.pcap &')
334     sleep(1)
335     h2.cmdPrint('iperf -c 192.168.1.1 -t 5')
336     h1.cmdPrint('tcpdump -r 1301204516.pcap')
```

Gambar 36 Code Capture iPerf using tcpdump

```
root@talitha-VirtualBox: /home/talitha/JarKom
*** hostB : ('iperf -c 192.168.1.1 -t 5',)
------------------------------------------------------------
Client connecting to 192.168.1.1, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.25 port 56772 connected with 192.168.1.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 5.6 sec   768 KBytes  1.12 Mbits/sec
*** hostA : ('tcpdump -r 1301204516.pcap',)
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
tcpdump: listening on hostA-fa0, link-type EN10MB (Ethernet), capture size 26214
4 bytes
[  4] local 192.168.1.1 port 5001 connected with 192.168.1.25 port 56772
5 packets captured
5 packets received by filter
0 packets dropped by kernel
reading from file 1301204516.pcap, link-type EN10MB (Ethernet)
18:28:41.967280 IP 192.168.1.25.56772 > 192.168.1.1.5001: Flags [S], seq 3203157
667, win 42340, options [mss 1460,sackOK,TS val 2639813423 ecr 0,nop,wscale 9],
length 0
18:28:41.967293 IP 192.168.1.1.5001 > 192.168.1.25.56772: Flags [S.], seq 250886
491, ack 3203157668, win 43440, options [mss 1460,sackOK,TS val 837191780 ecr 26
39813423,nop,wscale 9], length 0
18:28:41.967324 IP 192.168.1.25.56772 > 192.168.1.1.5001: Flags [.], ack 1, win
83, options [nop,nop,TS val 2639813423 ecr 837191780], length 0
18:28:41.967486 IP 192.168.1.25.56772 > 192.168.1.1.5001: Flags [P.], seq 1:7241
, ack 1, win 83, options [nop,nop,TS val 2639813423 ecr 837191780], length 7240
18:28:41.967497 IP 192.168.1.1.5001 > 192.168.1.25.56772: Flags [.], ack 7241, w
in 78, options [nop,nop,TS val 837191780 ecr 2639813423], length 0
*** Starting CLI:
mininet>
```

Gambar 37 Hasil capture iPerf



Gambar 38 hasil capture iPerf pada wireshark

2.5 CLO 4

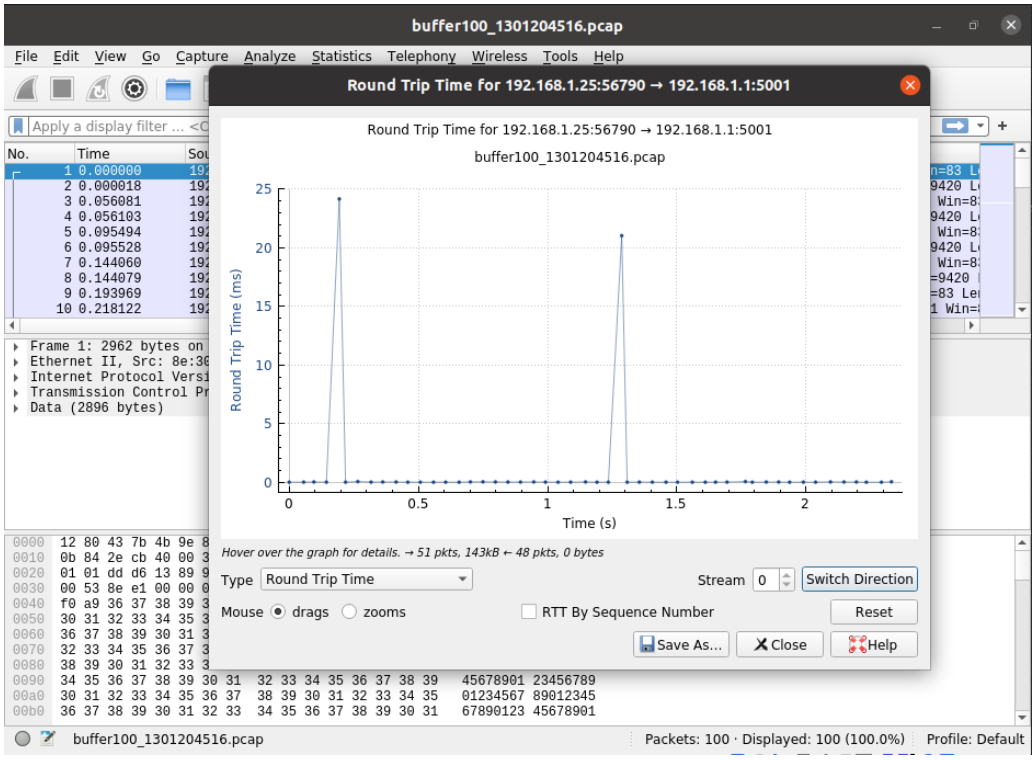
2.5.1 Setting up buffer

```
338  def buffered_traffic(h1,h2,r1,r2,r3,r4):
339      #buffer = 100
340      r1.cmdPrint("tc qdisc del dev r1-fa0 root")
341      r1.cmdPrint("tc qdisc add dev r1-fa0 root handle 1: pfifo limit 100")
342      h1.cmdPrint("iperf -s &")
343      h1.cmdPrint("tcpdump tcp -c 100 -w buffer100_1301204516.pcap &")
344      sleep(2)
345      h2.cmdPrint("iperf -c 192.168.1.1 -t 60")
346      h1.cmdPrint("tcpdump -r buffer100_1301204516.pcap")
347
348      #buffer = 60
349      r2.cmdPrint("tc qdisc del dev r2-fa0 root")
350      r2.cmdPrint("tc qdisc add dev r2-fa0 root handle 1: pfifo limit 60")
351      h1.cmdPrint("iperf -s &")
352      h1.cmdPrint("tcpdump tcp -c 100 -w buffer60_1301204516.pcap &")
353      sleep(2)
354      h2.cmdPrint("iperf -c 192.168.1.5 -t 60")
355      h1.cmdPrint("tcpdump -r buffer60_1301204516.pcap")
356
357      #buffer = 40
358      r1.cmdPrint("tc qdisc del dev r1-fa0 root")
359      r1.cmdPrint("tc qdisc add dev r1-fa0 root handle 1: pfifo limit 40")
360      h1.cmdPrint("iperf -s &")
361      h1.cmdPrint("tcpdump tcp -c 100 -w buffer40_1301204516.pcap &")
362      sleep(2)
363      h2.cmdPrint("iperf -c 192.168.1.1 -t 60")
364      h1.cmdPrint("tcpdump -r buffer40_1301204516.pcap")
365
366      #buffer = 20
367      r2.cmdPrint("tc qdisc del dev r2-fa0 root")
368      r2.cmdPrint("tc qdisc add dev r2-fa0 root handle 1: pfifo limit 20")
369      h1.cmdPrint("iperf -s &")
370      h1.cmdPrint("tcpdump tcp -c 100 -w buffer20_1301204516.pcap &")
371      sleep(2)
372      h2.cmdPrint("iperf -c 192.168.1.5 -t 60")
373      h1.cmdPrint("tcpdump -r buffer20_1301204516.pcap")
```

Gambar 39 Setting buffer

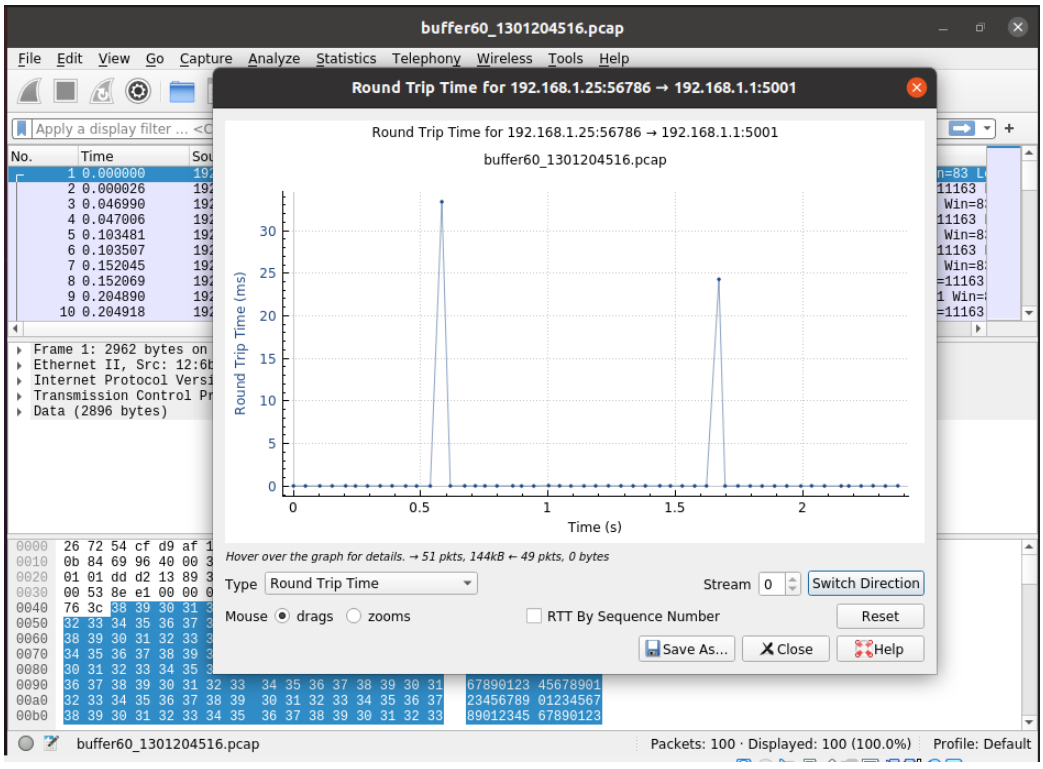
2.5.2 Wireshark Output

2.5.2.1 Buffer 100



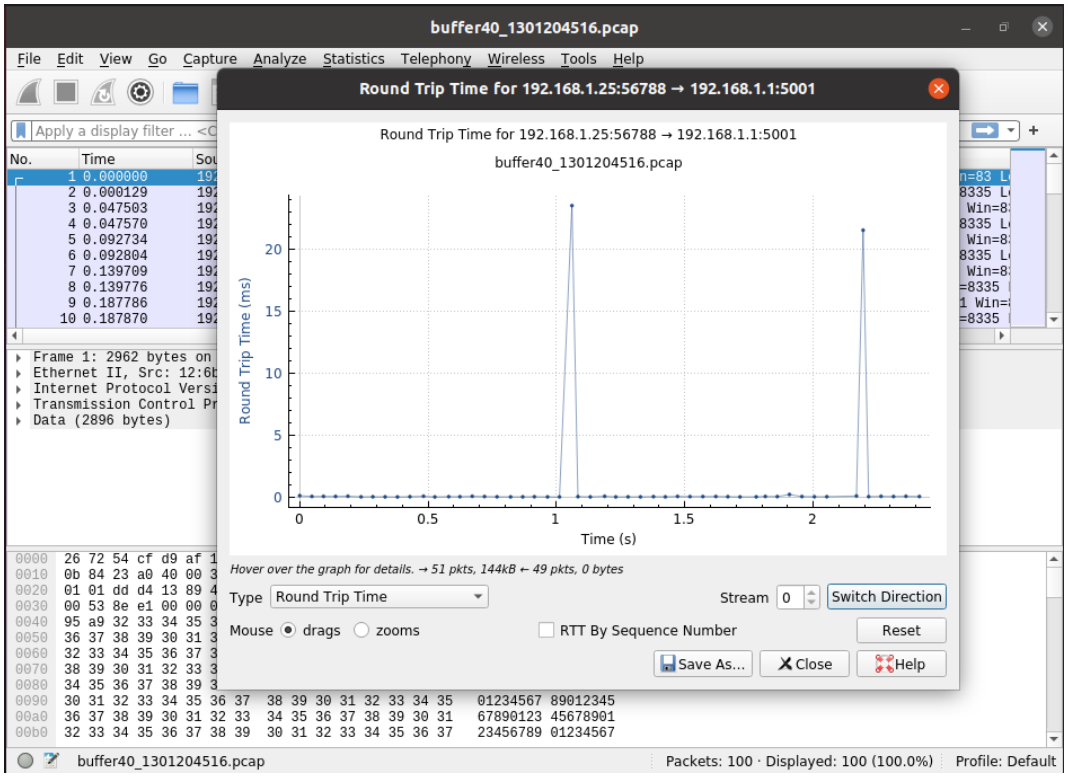
Gambar 40 Wireshark - Hasil Buffer 100

2.5.2.2 Buffer 60



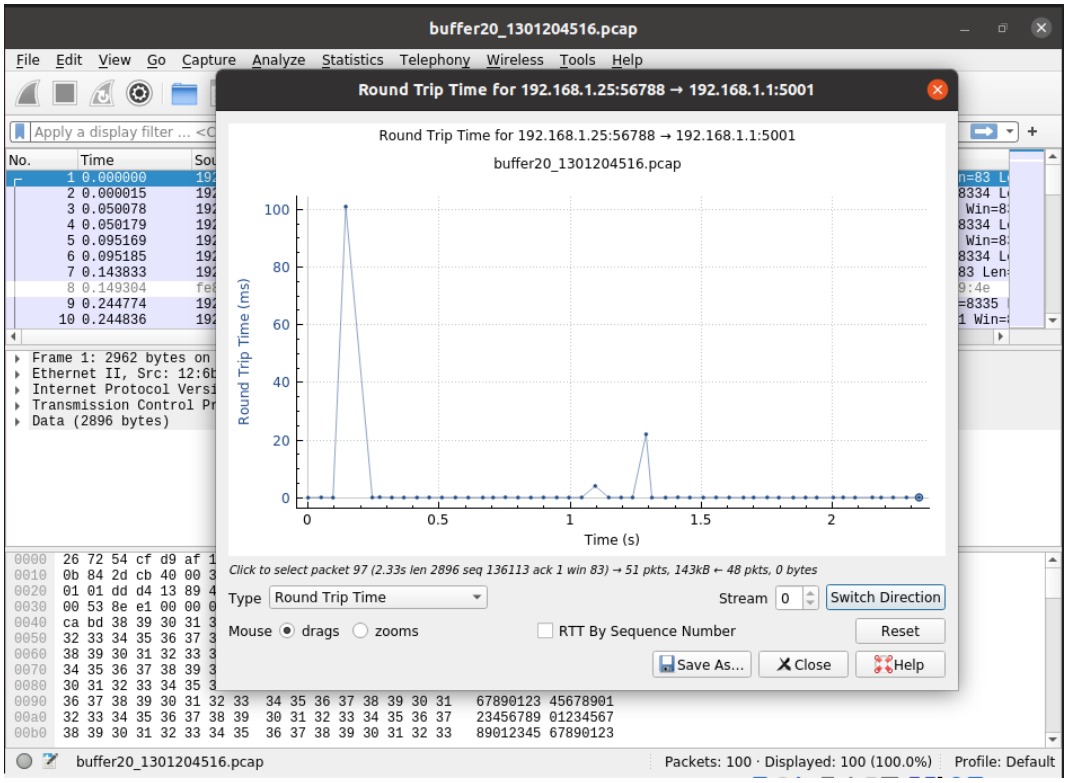
Gambar 41 Wireshark - Hasil Buffer 60

2.5.2.3 Buffer 40



Gambar 42 Wireshark - Hasil Buffer 40

2.5.2.4 Buffer 20



Gambar 43 Wireshark - Hasil Buffer 20

2.6 Video Pembahasan

Video pembahasan terteda pada link berikut :

https://youtu.be/gVrs7tGZ0ug

BAB 3
KESIMPULAN

3. Kesimpulan

Penggunaan mininet memerlukan keahlian dalam menulis kode dan memahami beberapa command line pada mininet. Berdasarkan hasil pengerjaan tugas Besar jaringan komputer ini maka dapat disimpulkan mininet dapat mensimulasikan sebuah topolog dengan host yang memiliki 2 NIC. Dalam melakukan simulasi, dibutuhkan pembangunan topologi terlebih dahulu seperti pada CLO-1. Setelah itu, pada CLO-2 dilakukan routing static untuk menghubungkan jaringan antar node dan telah terbukti dengan cara melakukan ping dan traceroute.

Selain itu, pada CLO-3 dibuktikan bahwa mininet dapat melakukan simulasi pengujian pengiriman packet protocol TCP menggunakan iPerf yang dapat di packet sniff oleh command TCPDump dan divisualisasikan dalam bentuk file .pcap yang dapat dibuka oleh wireshark. Terakhir, pada CLO-4 terbukti bahwa mininet dapat mensimulasikan sebuah buffer yang telah ditetapkan, dalam interface yang di-set menggunakan buffer tinggi akan memiliki RTT yang lebih fluktuatif dan memiliki perlambatan dalam mengirimkan packet.

REFERENSI

https://id.wikipedia.org/wiki/Jaringan_komputer

http://wulandari25.blogspot.com/2012/03/laporanpratikum-2-pengertian-wireshark.html