Laporan Tugas Besar
Jaringan Komputer

Oleh:

Artisa Bunga Syahputri

1301194007

Fakultas Informatika

Program Studi S1 Informatika

Universitas Telkom

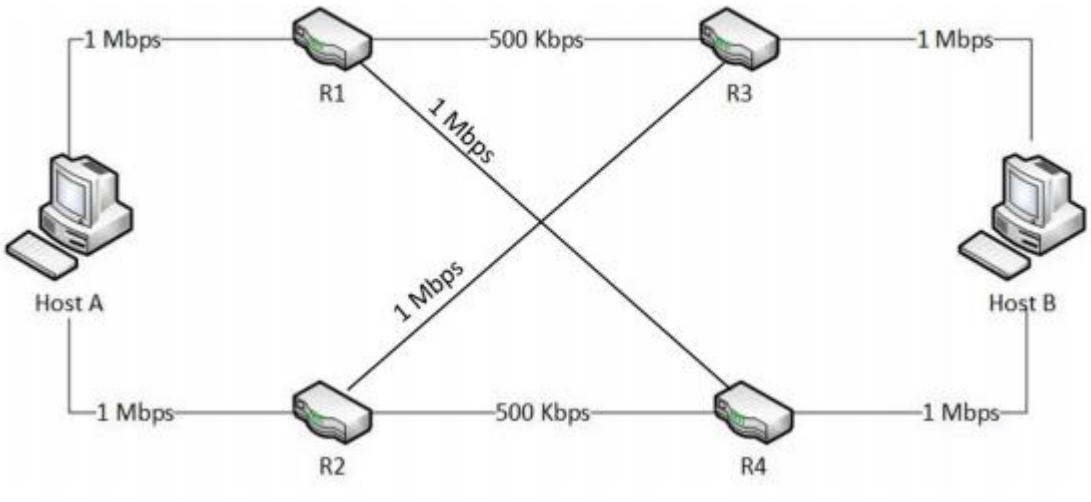
Daftar Isi

CLO 1: Membangun Topologi

Goals:

- membangun topologi sesuai dengan yang diberikan
- mendesign subnet masing-masing network
- melakukan konfigurasi IP address sesuai dengan subnet yang dibuat
- uji konektivitas dari topologi yang dibangun

Topologi yang akan dibangun:



Topologi terdiri dari 2 host dan 4 router yang saling terhubung seperti gambar diatas

1. membuat design subnetting yang akan digunakan untuk mengkonfigurasi IP address pada setiap host dan router

IP yang digunakan: 192.168.0.0/24

Tabel Subneting:

Nama	Needs	Alokasi	Network ID	Host Range	Broadcast	Prefiks	Subnet Mask
Network 1	2	256	192.168.0.0	192.168.0.1 - 192.168.0.254	192.168.0.255	\24	255.255.255.0
Network 2	2	256	192.168.1.0	192.168.1.1 - 192.168.1.254	192.168.1.255	\24	255.255.255.0
Network 3	2	256	192.168.2.0	192.168.2.1 - 192.168.2.254	192.168.2.255	\24	255.255.255.0
Network 4	2	256	192.168.3.0	192.168.3.1 - 192.168.3.254	192.168.3.255	\24	255.255.255.0
Network 5	2	256	192.168.4.0	192.168.4.1 - 192.168.4 254	192.168.4.255	\24	255.255.255.0
Network 6	2	256	192.168.5.0	192.168.5.1 - 192.168.5.254	192.168.5.255	\24	255.255.255.0
Network 7	2	256	192.168.6.0	192.168.6.1 - 192.168.6.254	192.168.6.255	\24	255.255.255.0
Network 8	2	256	192.168.7.0	192.168.7.1 - 192.168.7.254	192.168.7.255	\24	255.255.255.0

Network yang saling terhubung:

Network 1: Host 1 – R1 → h1-eth0 – r1eth0 →192.168.0.1 – 192.168.0.2

Network 2: Host 1 – R2 → h1-eth1 – r2eth0 →192.168.1.1 – 192.168.1.2

Network 3: R1 – R3 → r1-eth1 – r3eth0 →192.168.2.1 – 192.168.2.2

Network 4: R2 – R4 → r2-eth1 – r4eth0 →192.168.3.1 – 192.168.3.2

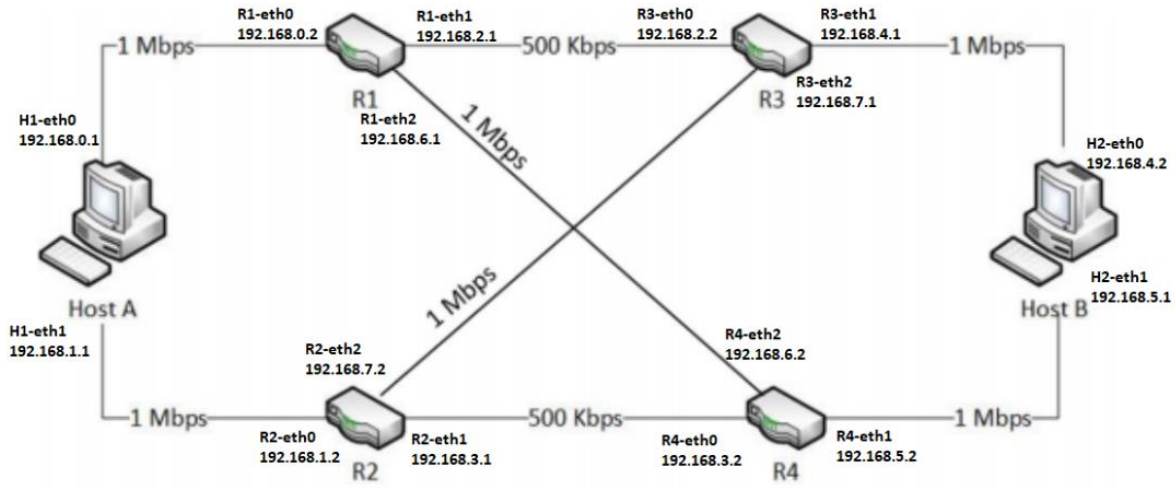
Network 5: R3 – Host 2 → r3-eth1 – h2eth0 →192.168.4.1 – 192.168.4.2

Network 6: R4 – Host 2 → r4-eth1 – h2eth1 →192.168.5.2 – 192.168.5.1

Network 7: R1 – R4 → r1-eth2 – r4eth2 →192.168.6.1 – 192.168.6.2

Network 8: R3 – R2 → r3-eth2 – r2eth2 →192.168.7.1 – 192.168.7.2

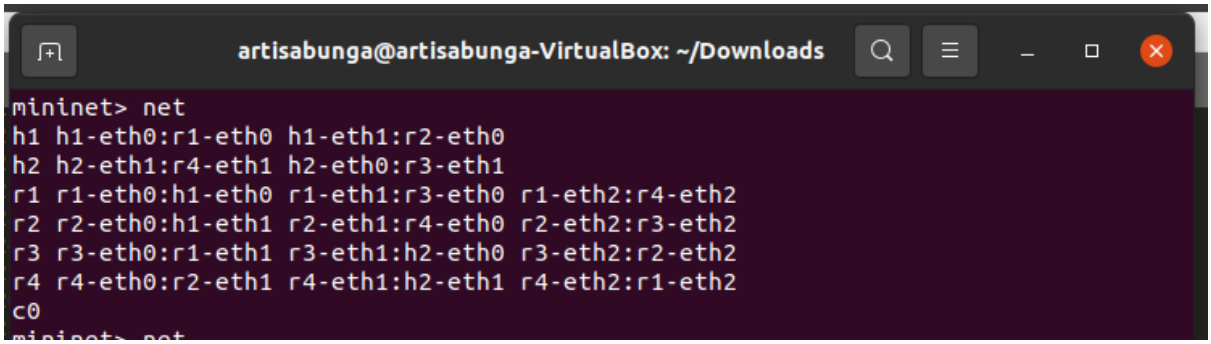
Topologi dengan IP address:



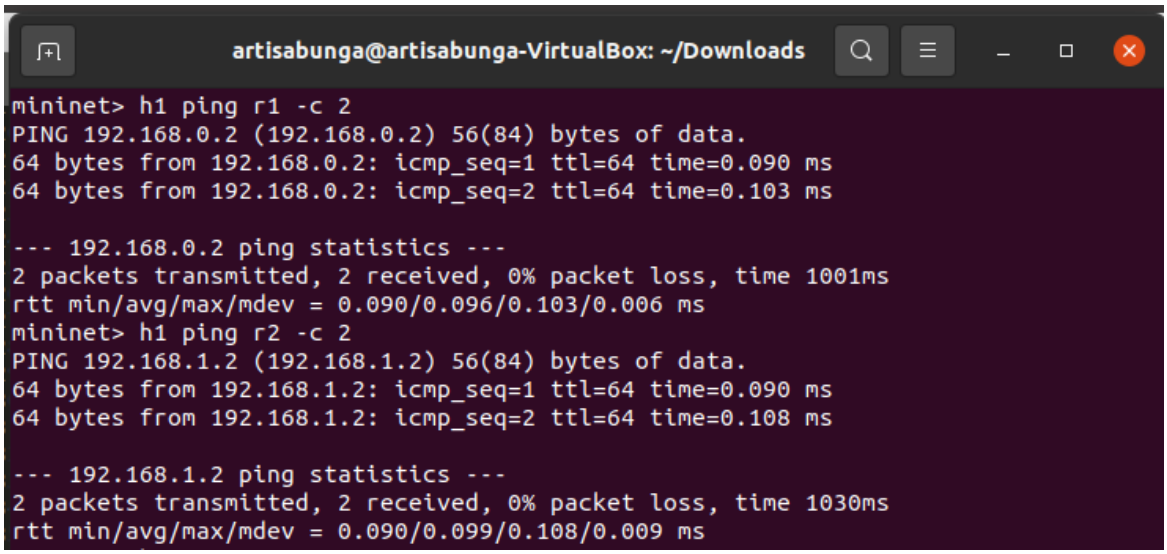
Membangun Topologi pada mininet:

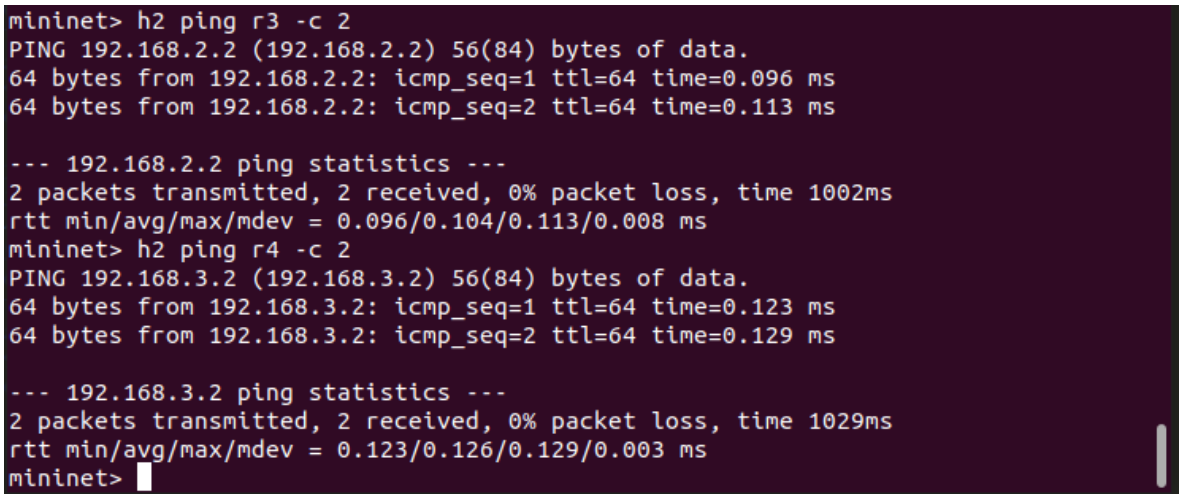
```
class NetworkTopo(Topo) :
    def __init__(self, **opts):
        Topo.__init__(self,**opts)

        #tambahin host
        h1 = self.addHost("h1" )
        h2 = self.addHost("h2")
        #tambahkan router
        r1 = self.addHost("r1")
        r2 = self.addHost("r2")
        r3 = self.addHost("r3")
        r4 = self.addHost("r4")
        #buat tipe bandwitch
        bw1k ={"bw": 1}
        bw500= {"bw": 0.5}
        #sambungkan
        self.addLink(h1,r1, intfName1 = 'h1-eth0', intfName2 = 'r1-eth0', cls = TCLink, **bw1k
        self.addLink(h1,r2, intfName1 = 'h1-eth1', intfName2 = 'r2-eth0', cls = TCLink, **bw1k
        self.addLink(r1,r3, intfName1 = 'r1-eth1', intfName2 = 'r3-eth0', cls = TCLink, **bw500
        self.addLink(r2,r4, intfName1 = 'r2-eth1', intfName2 = 'r4-eth0', cls = TCLink, **bw500
        self.addLink(r4,h2, intfName1 = 'r4-eth1', intfName2 = 'h2-eth1', cls = TCLink, **bw1k
        self.addLink(r3,h2, intfName1 = 'r3-eth1', intfName2 = 'h2-eth0', cls = TCLink, **bw1k
        self.addLink(r1,r4, intfName1 = 'r1-eth2', intfName2 = 'r4-eth2', cls = TCLink, **bw1k
        self.addLink(r2,r3, intfName1 = 'r2-eth2', intfName2 = 'r3-eth2', cls = TCLink, **bw1k
```

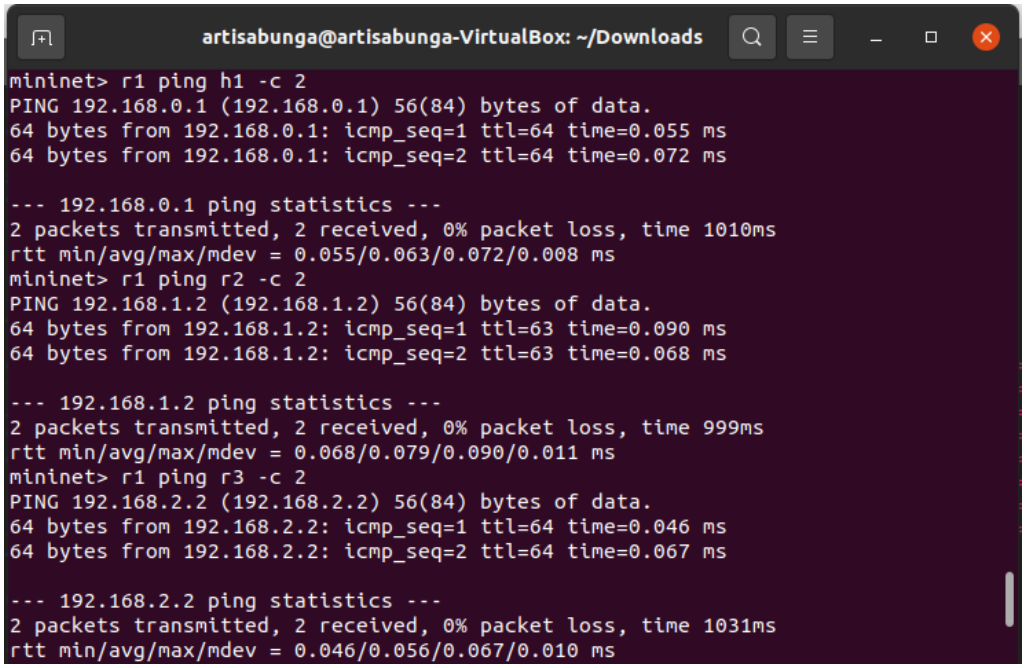
Hasil Running topologi yang terbentuk

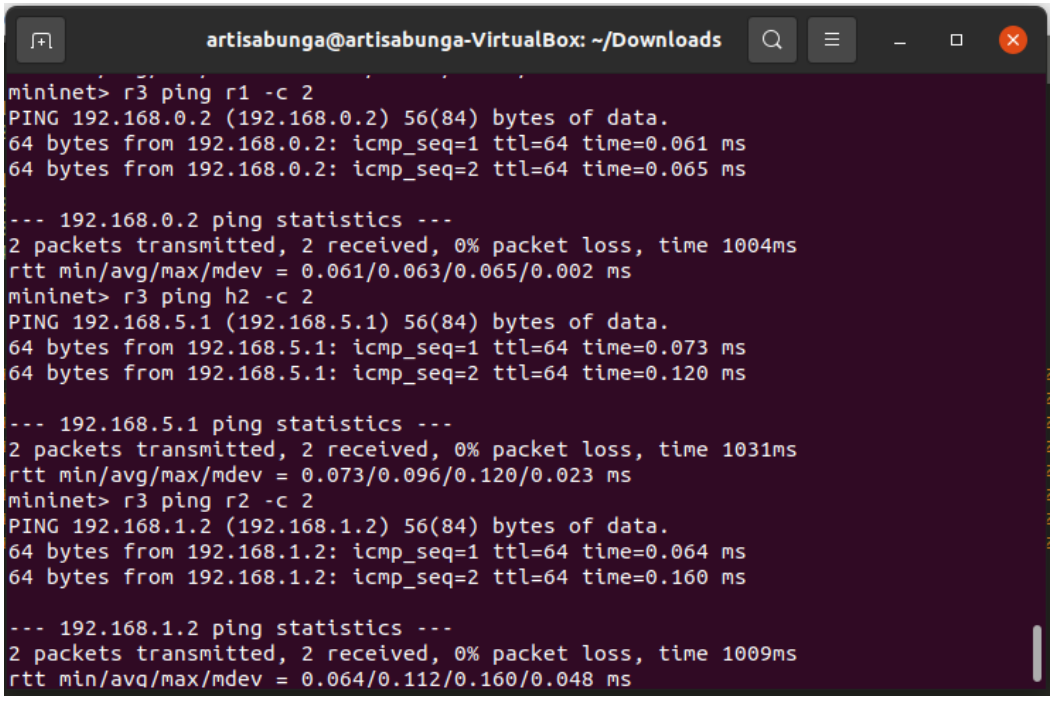


Topologi sudah terbangun






artisabunga@artisabunga-VirtualBox: ~/Downloads
mininet> r1 ping h1 -c 2
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_seq=1 ttl=64 time=0.055 ms
64 bytes from 192.168.0.1: icmp_seq=2 ttl=64 time=0.072 ms
--- 192.168.0.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1010ms
rtt min/avg/max/mdev = 0.055/0.063/0.072/0.008 ms
mininet> r1 ping r2 -c 2
PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data.
64 bytes from 192.168.1.2: icmp_seq=1 ttl=63 time=0.090 ms
64 bytes from 192.168.1.2: icmp_seq=2 ttl=63 time=0.068 ms
--- 192.168.1.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.068/0.079/0.090/0.011 ms
mininet> r1 ping r3 -c 2
PING 192.168.2.2 (192.168.2.2) 56(84) bytes of data.
64 bytes from 192.168.2.2: icmp_seq=1 ttl=64 time=0.046 ms
64 bytes from 192.168.2.2: icmp_seq=2 ttl=64 time=0.067 ms
--- 192.168.2.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1031ms
rtt min/avg/max/mdev = 0.046/0.056/0.067/0.010 ms


artisabunga@artisabunga-VirtualBox: ~/Downloads
mininet> r3 ping r1 -c 2
PING 192.168.0.2 (192.168.0.2) 56(84) bytes of data.
64 bytes from 192.168.0.2: icmp_seq=1 ttl=64 time=0.061 ms
64 bytes from 192.168.0.2: icmp_seq=2 ttl=64 time=0.065 ms
--- 192.168.0.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1004ms
rtt min/avg/max/mdev = 0.061/0.063/0.065/0.002 ms
mininet> r3 ping h2 -c 2
PING 192.168.5.1 (192.168.5.1) 56(84) bytes of data.
64 bytes from 192.168.5.1: icmp_seq=1 ttl=64 time=0.073 ms
64 bytes from 192.168.5.1: icmp_seq=2 ttl=64 time=0.120 ms
--- 192.168.5.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1031ms
rtt min/avg/max/mdev = 0.073/0.096/0.120/0.023 ms
mininet> r3 ping r2 -c 2
PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data.
64 bytes from 192.168.1.2: icmp_seq=1 ttl=64 time=0.064 ms
64 bytes from 192.168.1.2: icmp_seq=2 ttl=64 time=0.160 ms
--- 192.168.1.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1009ms
rtt min/avg/max/mdev = 0.064/0.112/0.160/0.048 ms

CLO 2: Pengimplementasian Routing pada Topologi

Goals:

- Mengimplementasikan mekanisme Routing pada topologi yang ada
- Uji konektivitas menggunakan ping. Membuat table routing di semua host Dibuktikan dengan ping antar host, pakai traceroute

Routing atau perutean adalah proses untuk proses untuk menghantarkan paket dari satu jaringan ke jaringan lain hingga paket dapat sampai ke tujuan melalui *inter-network*

Ada 2 jenis routing yang dipelajari, yaitu static routing dan dynamic routing

Static routing adalah metode perutean terhadap jairngan dimana table jaringan dibangun secara manual, sehingga diharuskan untuk memasukan command secara manual di router setiap akan dilakukan perubahan jalur pengantaran paket.

Dynamic routing adalah perutean yang adaptif dimana rute dapat meneruskan data melalui rute yang berbeda dengan yang didaftarkan berdasarkan kondisi saat ini dari sirkuit komunikasi jaringan dalam sistem jaringan tersebut.

Pada Tubes ini akan dilakuan pengimplementasian staitic routing dari topologi yang sudah dibangun dan dilakukan konfigurasi IP address pada clo sebelumnya

Pembangunan static routing pada mininet:

Routing pada Host:

```
h1.cmd("ifconfig h1-eth0 192.168.0.1 netmask 255.255.255.0")
h1.cmd("ifconfig h1-eth1 192.168.1.1 netmask 255.255.255.0")

h1.cmd("ip rule add from 192.168.0.1 table 1")
h1.cmd("ip rule add from 192.168.1.1 table 2")
h1.cmd("ip route add 192.168.0.0 netmask 255.255.255.0 dev h1-eth0 scope link table 1")
h1.cmd("ip route add default via 192.168.0.2 dev h1-eth0 ")
h1.cmd("ip route add 192.168.1.0 netmask 255.255.255.0 dev h1-eth1 scope link table 2")
h1.cmd("ip route add default via 192.168.1.2 dev h1-eth1")
h1.cmd("ip route add default scope global nexthop via 192.168.0.2 dev h1-eth0")
h1.cmd("ip route add default scope global nexthop via 192.168.1.2 dev h1-eth1")
```

```
h2.cmd("ifconfig h2-eth0 192.168.4.2 netmask 255.255.255.0")
h2.cmd("ifconfig h2-eth1 192.168.5.1 netmask 255.255.255.0")
#routing

h2.cmd("ip rule add from 192.168.4.2 table 1")
h2.cmd("ip rule add from 192.168.5.1 table 2")
h2.cmd("ip route add 192.168.4.0 netmask 255.255.255.0 dev h2-eth0 scope link table 1")
h2.cmd("ip route add default via 192.168.4.1 dev h2-eth0 ")
h2.cmd("ip route add 192.168.5.0 netmask 255.255.255.0 dev h2-eth1 link table 2")
h2.cmd("ip route add default via 192.168.5.2 dev h2-eth1 ")
h2.cmd("ip route add default scope global nexthop via 192.168.4.1 dev h2-eth0")
h2.cmd("ip route add default scope global nexthop via 192.168.5.2 dev h2-eth1")
```

Routing pada Router:

```
r1.cmd("route add -net 192.168.1.0/24 gw 192.168.0.1")
r1.cmd("route add -net 192.168.1.0/24 gw 192.168.6.2")
r1.cmd("route add -net 192.168.4.0/24 gw 192.168.2.2")
r1.cmd("route add -net 192.168.5.0/24 gw 192.168.6.2")
r1.cmd("route add -net 192.168.7.0/24 gw 192.168.2.2")
r1.cmd("route add -net 192.168.3.0/24 gw 192.168.6.2")
```

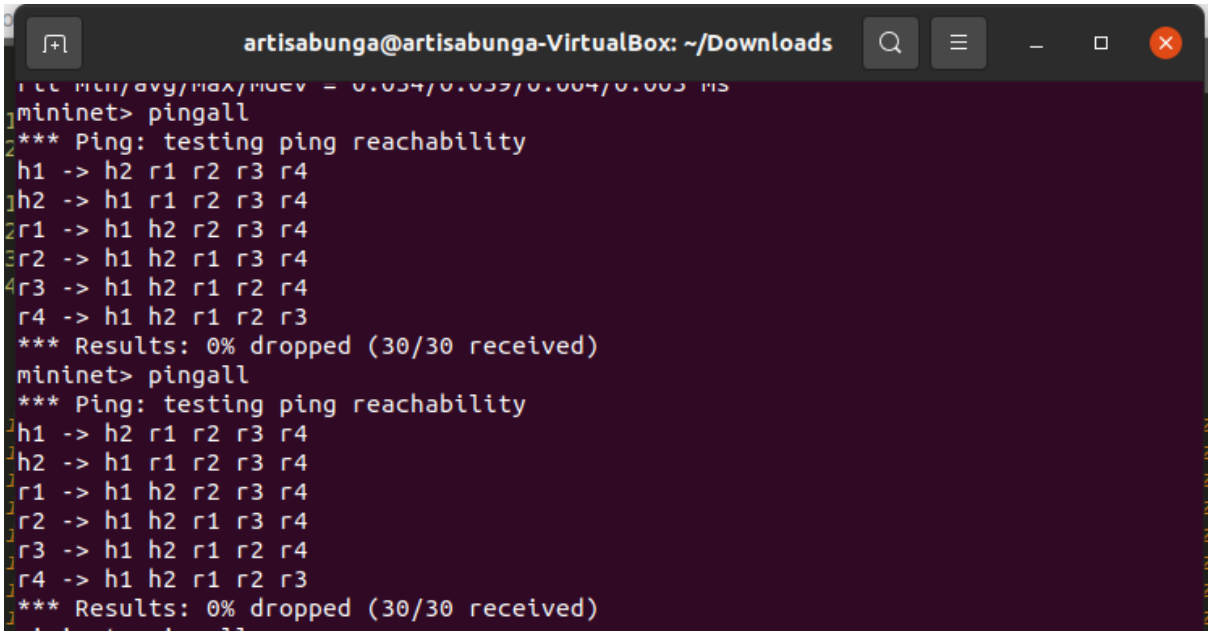
```
r2.cmd("route add -net 192.168.0.0/24 gw 192.168.1.1")
r2.cmd("route add -net 192.168.0.0/24 gw 192.168.7.1")
r2.cmd("route add -net 192.168.2.0/24 gw 192.168.7.1")
r2.cmd("route add -net 192.168.6.0/24 gw 192.168.3.2")
r2.cmd("route add -net 192.168.5.0/24 gw 192.168.3.2")
r2.cmd("route add -net 192.168.4.0/24 gw 192.168.7.1")
```

```
r3.cmd("route add -net 192.168.0.0/24 gw 192.168.2.1")
r3.cmd("route add -net 192.168.1.0/24 gw 192.168.7.2")
r3.cmd("route add -net 192.168.3.0/24 gw 192.168.7.2")
r3.cmd("route add -net 192.168.5.0/24 gw 192.168.4.2")
r3.cmd("route add -net 192.168.6.0/24 gw 192.168.2.1")
```

```
r4.cmd("route add -net 192.168.0.0/24 gw 192.168.6.1")
r4.cmd("route add -net 192.168.1.0/24 gw 192.168.3.1")
r4.cmd("route add -net 192.168.7.0/24 gw 192.168.3.1")
r4.cmd("route add -net 192.168.2.0/24 gw 192.168.6.1")
r4.cmd("route add -net 192.168.4.0/24 gw 192.168.5.1")
```

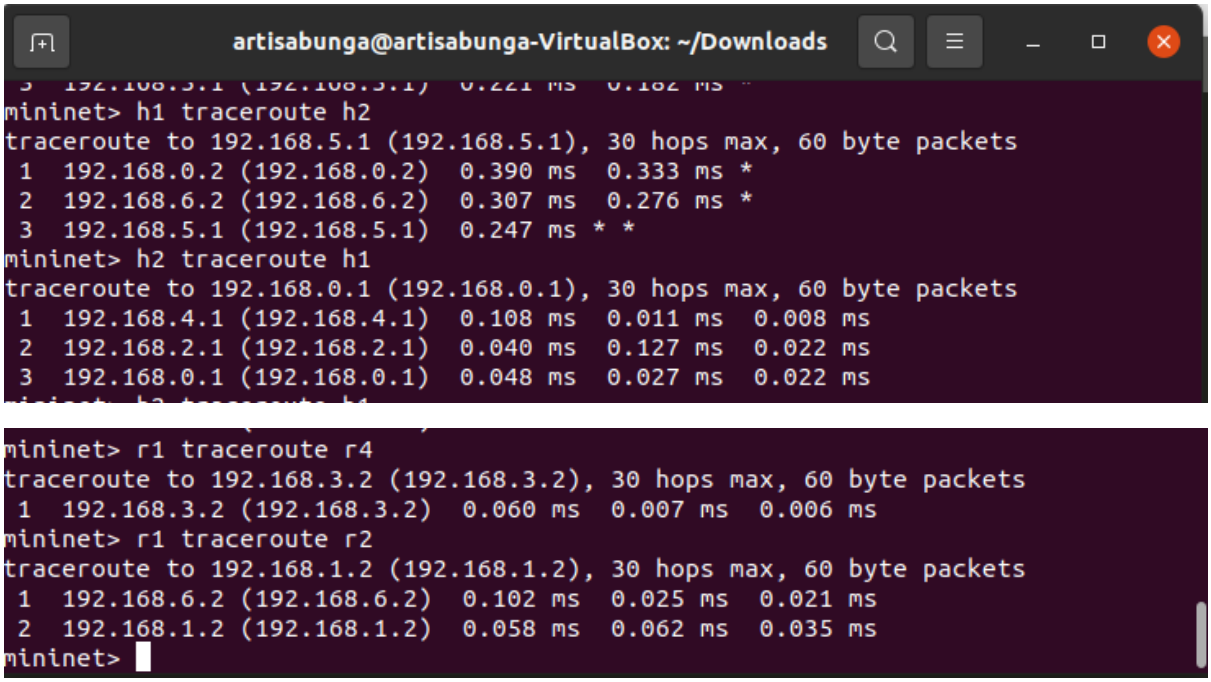
Hasil Running:

Saat di pingall



Semua host dan router sudah terhubung dengan metode Static routing

Melihat traceroute dari h1 ke h2 dan sebaliknya:



Dari hasil running diatas, untuk dapat mengirim paket ke Host 2 maka paket harus dihantarkan sebanyak 3 kali dari IP satu ke IP berikutnya hingga sampai ke Host 2, begitu juga sebaliknya.

CLO 3: Generate Traffic TCP

Goals:

- Membuktikan bahwa TCP telah di-implementasikan dengan benar pada topologi
- Generate trafik dari h1 ke h2 menggunakan iperf.
- Inspeksi trafik pakai wireshark, dibuktikan dengan trafik TCP di wireshark/tcp dump

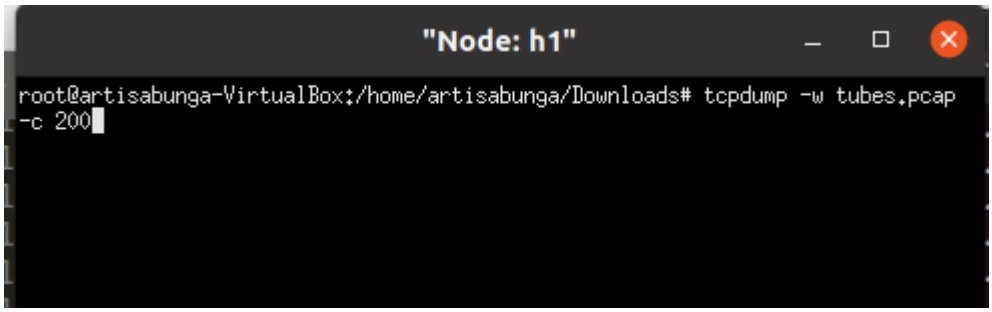
TCP (*Transmission Control Protocol)* adalah standar protocol yang digunakan sebagai protocol pertukaran data yang berorientasi pada sambungan

Perbedaan umum antara TCP dan UDP adalah cara pengiriman datanya, dimana UDP mengirimkan data tidak secara berurutan sehingga tidak dapat dipastikan paket mana yang sampai di tujuan terlebih dahulu, sedangkan TCP mengirimkan data secara berurutan dari awal hingga akhir

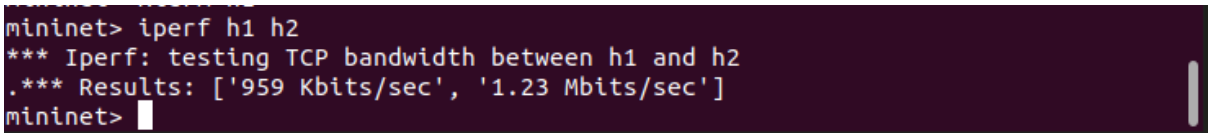
TCP berada pada layer transport pada pada model OSI layer. Layer transport berfungsi untuk memecah data dan mentransmisikan nya ke network layer dari session layer maupun sebaliknya.

Pada clo ini ingin dibuktikan bahwa protocol TCP diimplementasikan pad topologi, yang dilakukan adalah melakukan capture traffic dari h1 ke h2 untuk meliha protocol yang diterapkan pada pengiriman paketnya.

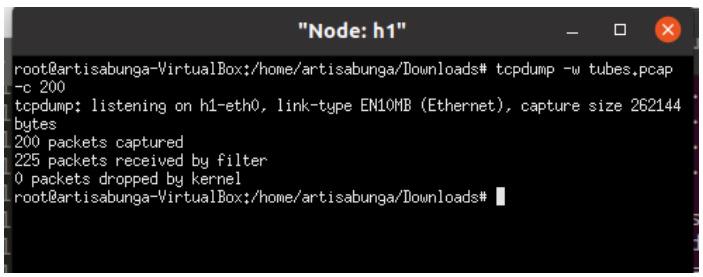
Proses untuk melakukan Capture grafik dari mininet ke wireshark:



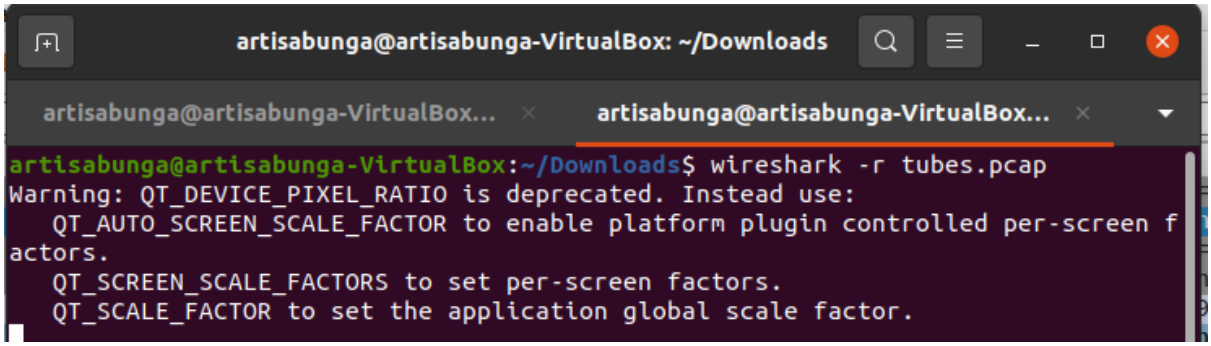
Melakukan generate trafik h1 ke h2 menggunakan iperf:



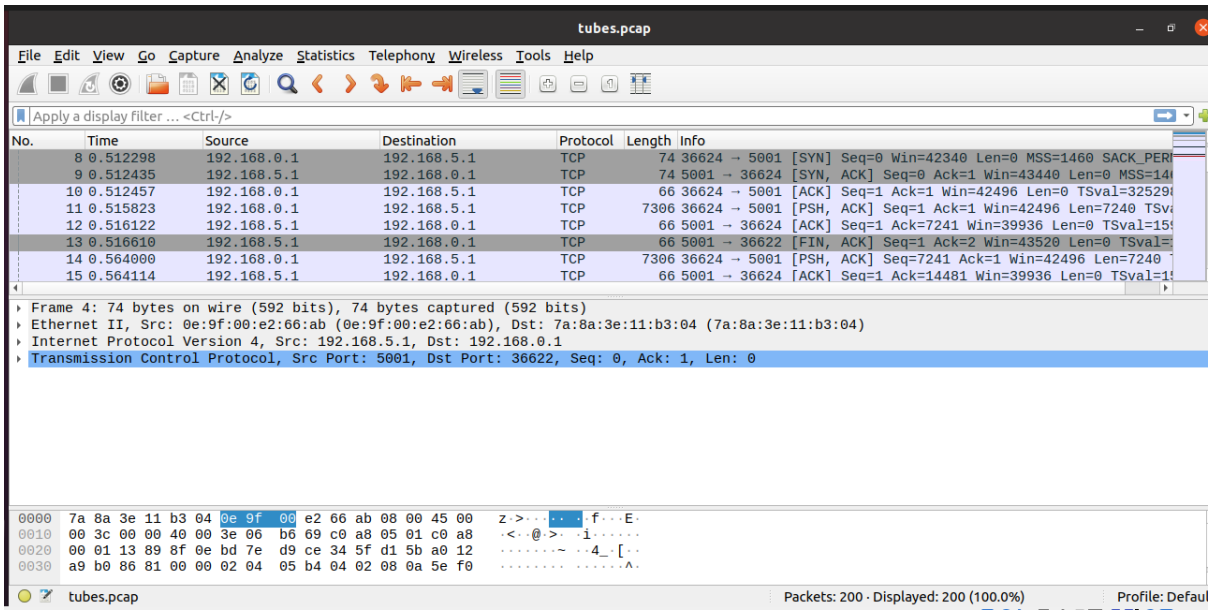
Hasil pada xterm setelah di generate trafik:



Buka hasil capture trafik TCP pada wireshark:



Hasil capture traffic di wireshark:



Pada hasil diatas dapat dilihat bahwa protocol yang digunakan pada topologi yang dibuat menggunakan protocol TCP.

TCP menggunakan metode three-way handshake(SYN- SYN ACK – ACK) itu adalah cara yang dilakukan oke protocol TCP untuk melakukan pertikaran data antar host atau antar jaringan. Proses three way dari capture traffic diatas dicontohkan pada nomor 8,9,san 10

1. host pertama 192.168.0.1 (ingin membuat koneksi dengan 192.168.5.1) akan mengirim segmen TCP dengan flag SYN ke host kedua yaitu 192.168.5.1
2. host kesua kemudian akan merespon dengan mengirim Kembali segmen dengan flag SYN kepada host pertama (SYN-ACK)
3. terjadilah pertukaran data antara host pertama dan kedua (ACK)

CLO 4: Analisis Pengaruh Buffer dan Penggunaan Queue pada jaringan

GOALS:

- Menginspeksi penggunaan queue pada router jaringan
- Set ukuran buffer pada router : 20, 40, 60 dan 100
- Generate *traffic* dan *background traffic* menggunakan iPerf
- Capture pengaruh ukuran buffer terhadap *delay*
- Analisis eksperimen hasil variasi ukuran buffer
- Mahasiswa mengerti caranya mengubah buffer dan mengenai pengaruh besar buffer

queue adalah pengaturan antrian yang mengatur *sequence* dari paket jaringan yang doterima pada interface jaringan

Buffer adalah Teknik yang digunakan untuk meningkatkan efisiensi dari sistem operrasi dalam melakukan proses input outpu, maupun proses lainnya

Buffer berpengaruh pada delay dan juga packet loss pada saat melakukan transfer paket

Berikut adalah percobaan dari transfer data untuk bilai buffer 20, 40, 60, dan 100

1. Nilai buffer 20

```
        #sambungkan
        buff = 20

        self.addLink(h1,r1, intfName1 = 'h1-eth0', intfName2 = 'r1-eth0', cls = TCLink, **bw1k  , max_queue_size=buff,use_htb=True)
        self.addLink(h1,r2, intfName1 = 'h1-eth1', intfName2 = 'r2-eth0', cls = TCLink, **bw1k  , max_queue_size=buff,use_htb=True)
        self.addLink(r1,r3, intfName1 = 'r1-eth1', intfName2 = 'r3-eth0', cls = TCLink, **bw500 , max_queue_size=buff,use_htb=True)
        self.addLink(r2,r4, intfName1 = 'r2-eth1', intfName2 = 'r4-eth0', cls = TCLink, **bw500 , max_queue_size=buff,use_htb=True)
        self.addLink(r4,h2, intfName1 = 'r4-eth1', intfName2 = 'h2-eth1', cls = TCLink, **bw1k  , max_queue_size=buff,use_htb=True)
        self.addLink(r3,h2, intfName1 = 'r3-eth1', intfName2 = 'h2-eth0', cls = TCLink, **bw1k  , max_queue_size=buff,use_htb=True)
        self.addLink(r1,r4, intfName1 = 'r1-eth2', intfName2 = 'r4-eth2', cls = TCLink, **bw1k  , max_queue_size=buff,use_htb=True)
        self.addLink(r2,r3, intfName1 = 'r2-eth2', intfName2 = 'r3-eth2', cls = TCLink, **bw1k  , max_queue_size=buff,use_htb=True)
```

Hasil running:

```
artisabunga@artisabunga-VirtualBox: ~/Downloads
*** Starting 0 switches

*** h2 : ('iperf -s &',)
*** h1 : ('iperf -t 60 -c 192.168.4.2 &',)
*** Starting CLI:
mininet> h2 ping h1 -c 20
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 192.168.4.2 port 5001 connected with 192.168.0.1 port 33506
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_seq=4 ttl=62 time=633 ms
64 bytes from 192.168.0.1: icmp_seq=5 ttl=62 time=409 ms
64 bytes from 192.168.0.1: icmp_seq=6 ttl=62 time=476 ms
64 bytes from 192.168.0.1: icmp_seq=7 ttl=62 time=446 ms
64 bytes from 192.168.0.1: icmp_seq=8 ttl=62 time=488 ms
64 bytes from 192.168.0.1: icmp_seq=9 ttl=62 time=580 ms
64 bytes from 192.168.0.1: icmp_seq=10 ttl=62 time=720 ms
64 bytes from 192.168.0.1: icmp_seq=14 ttl=62 time=689 ms
64 bytes from 192.168.0.1: icmp_seq=15 ttl=62 time=685 ms
64 bytes from 192.168.0.1: icmp_seq=16 ttl=62 time=509 ms
64 bytes from 192.168.0.1: icmp_seq=17 ttl=62 time=603 ms
64 bytes from 192.168.0.1: icmp_seq=18 ttl=62 time=621 ms
64 bytes from 192.168.0.1: icmp_seq=19 ttl=62 time=615 ms
64 bytes from 192.168.0.1: icmp_seq=20 ttl=62 time=659 ms

--- 192.168.0.1 ping statistics ---
20 packets transmitted, 14 received, 30% packet loss, time 19133ms
rtt min/avg/max/mdev = 409.396/580.951/719.630/94.908 ms
mininet>
```

2. Nilai buffer 40

```
#sambungkan
buff = 40

self.addLink(h1,r1, intfName1 = 'h1-eth0', intfName2 = 'r1-eth0', cls = TCLink, **bw1k  , max_queue_size=buff,use_htb=True)
self.addLink(h1,r2, intfName1 = 'h1-eth1', intfName2 = 'r2-eth0', cls = TCLink, **bw1k  , max_queue_size=buff,use_htb=True)
self.addLink(r1,r3, intfName1 = 'r1-eth1', intfName2 = 'r3-eth0', cls = TCLink, **bw500 , max_queue_size=buff,use_htb=True)
self.addLink(r2,r4, intfName1 = 'r2-eth1', intfName2 = 'r4-eth0', cls = TCLink, **bw500 , max_queue_size=buff,use_htb=True)
self.addLink(r4,h2, intfName1 = 'r4-eth1', intfName2 = 'h2-eth1', cls = TCLink, **bw1k  , max_queue_size=buff,use_htb=True)
self.addLink(r3,h2, intfName1 = 'r3-eth1', intfName2 = 'h2-eth0', cls = TCLink, **bw1k  , max_queue_size=buff,use_htb=True)
self.addLink(r1,r4, intfName1 = 'r1-eth2', intfName2 = 'r4-eth2', cls = TCLink, **bw1k  , max_queue_size=buff,use_htb=True)
self.addLink(r2,r3, intfName1 = 'r2-eth2', intfName2 = 'r3-eth2', cls = TCLink, **bw1k  , max_queue_size=buff,use_htb=True)
```

Hasil Running:

```
artisabunga@artisabunga-VirtualBox: ~/Downloads
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 192.168.4.2 port 5001 connected with 192.168.0.1 port 33502
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_seq=9 ttl=62 time=1157 ms
64 bytes from 192.168.0.1: icmp_seq=10 ttl=62 time=934 ms

--- 192.168.0.1 ping statistics ---
10 packets transmitted, 2 received, 80% packet loss, time 9205ms
rtt min/avg/max/mdev = 934.415/1045.641/1156.867/111.226 ms, pipe 2
mininet> h2 ping h1 -c 20
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_seq=1 ttl=62 time=1323 ms
64 bytes from 192.168.0.1: icmp_seq=3 ttl=62 time=1353 ms
64 bytes from 192.168.0.1: icmp_seq=4 ttl=62 time=1139 ms
64 bytes from 192.168.0.1: icmp_seq=5 ttl=62 time=987 ms
64 bytes from 192.168.0.1: icmp_seq=6 ttl=62 time=1104 ms
64 bytes from 192.168.0.1: icmp_seq=7 ttl=62 time=1188 ms
64 bytes from 192.168.0.1: icmp_seq=8 ttl=62 time=1181 ms
64 bytes from 192.168.0.1: icmp_seq=9 ttl=62 time=1222 ms
64 bytes from 192.168.0.1: icmp_seq=10 ttl=62 time=1265 ms
64 bytes from 192.168.0.1: icmp_seq=11 ttl=62 time=1403 ms
64 bytes from 192.168.0.1: icmp_seq=12 ttl=62 time=1687 ms
64 bytes from 192.168.0.1: icmp_seq=18 ttl=62 time=1263 ms
64 bytes from 192.168.0.1: icmp_seq=19 ttl=62 time=1257 ms
64 bytes from 192.168.0.1: icmp_seq=20 ttl=62 time=1250 ms

--- 192.168.0.1 ping statistics ---
20 packets transmitted, 14 received, 30% packet loss, time 19183ms
rtt min/avg/max/mdev = 987.116/1258.774/1687.076/156.215 ms, pipe 2
mininet>
```

3. Nilai buffer 60

```
#sambungkan
buff = 60

self.addLink(h1,r1, intfName1 = 'h1-eth0', intfName2 = 'r1-eth0', cls = TCLink, **bw1k  , max_queue_size=buff,use_htb=True)
self.addLink(h1,r2, intfName1 = 'h1-eth1', intfName2 = 'r2-eth0', cls = TCLink, **bw1k  , max_queue_size=buff,use_htb=True)
self.addLink(r1,r3, intfName1 = 'r1-eth1', intfName2 = 'r3-eth0', cls = TCLink, **bw500 , max_queue_size=buff,use_htb=True)
self.addLink(r2,r4, intfName1 = 'r2-eth1', intfName2 = 'r4-eth0', cls = TCLink, **bw500 , max_queue_size=buff,use_htb=True)
self.addLink(r4,h2, intfName1 = 'r4-eth1', intfName2 = 'h2-eth1', cls = TCLink, **bw1k  , max_queue_size=buff,use_htb=True)
self.addLink(r3,h2, intfName1 = 'r3-eth1', intfName2 = 'h2-eth0', cls = TCLink, **bw1k  , max_queue_size=buff,use_htb=True)
self.addLink(r1,r4, intfName1 = 'r1-eth2', intfName2 = 'r4-eth2', cls = TCLink, **bw1k  , max_queue_size=buff,use_htb=True)
self.addLink(r2,r3, intfName1 = 'r2-eth2', intfName2 = 'r3-eth2', cls = TCLink, **bw1k  , max_queue_size=buff,use_htb=True)
```

Hasil Running:

```
artisabunga@artisabunga-VirtualBox: ~/Downloads
*** Configuring hosts
h1 (cfs -1/100000us) h2 (cfs -1/100000us) r1 (cfs -1/100000us) r2 (cfs -1/100000
us) r3 (cfs -1/100000us) r4 (cfs -1/100000us)
*** Starting controller
c0
*** Starting 0 switches

*** h2 : ('iperf -s &',)
*** h1 : ('iperf -t 60 -c 192.168.4.2 &',)
*** Starting CLI:
mininet> h2 ping h1 -c 20
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 192.168.4.2 port 5001 connected with 192.168.0.1 port 33510
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_seq=1 ttl=62 time=1952 ms
64 bytes from 192.168.0.1: icmp_seq=2 ttl=62 time=2449 ms
64 bytes from 192.168.0.1: icmp_seq=11 ttl=62 time=2116 ms
64 bytes from 192.168.0.1: icmp_seq=15 ttl=62 time=1686 ms
64 bytes from 192.168.0.1: icmp_seq=16 ttl=62 time=1439 ms
64 bytes from 192.168.0.1: icmp_seq=17 ttl=62 time=1409 ms
64 bytes from 192.168.0.1: icmp_seq=18 ttl=62 time=1499 ms
64 bytes from 192.168.0.1: icmp_seq=19 ttl=62 time=1614 ms
64 bytes from 192.168.0.1: icmp_seq=20 ttl=62 time=1655 ms

--- 192.168.0.1 ping statistics ---
20 packets transmitted, 9 received, 55% packet loss, time 19277ms
rtt min/avg/max/mdev = 1409.039/1757.697/2448.569/328.297 ms, pipe 3
mininet>
```

4. Nilai buffer 100

```
#sambungkan
buff = 100

self.addLink(h1,r1, intfName1 = 'h1-eth0', intfName2 = 'r1-eth0', cls = TCLink, **bw1k  , max_queue_size=buff,use_htb=True)
self.addLink(h1,r2, intfName1 = 'h1-eth1', intfName2 = 'r2-eth0', cls = TCLink, **bw1k  , max_queue_size=buff,use_htb=True)
self.addLink(r1,r3, intfName1 = 'r1-eth1', intfName2 = 'r3-eth0', cls = TCLink, **bw500 , max_queue_size=buff,use_htb=True)
self.addLink(r2,r4, intfName1 = 'r2-eth1', intfName2 = 'r4-eth0', cls = TCLink, **bw500 , max_queue_size=buff,use_htb=True)
self.addLink(r4,h2, intfName1 = 'r4-eth1', intfName2 = 'h2-eth1', cls = TCLink, **bw1k  , max_queue_size=buff,use_htb=True)
self.addLink(r3,h2, intfName1 = 'r3-eth1', intfName2 = 'h2-eth0', cls = TCLink, **bw1k  , max_queue_size=buff,use_htb=True)
self.addLink(r1,r4, intfName1 = 'r1-eth2', intfName2 = 'r4-eth2', cls = TCLink, **bw1k  , max_queue_size=buff,use_htb=True)
self.addLink(r2,r3, intfName1 = 'r2-eth2', intfName2 = 'r3-eth2', cls = TCLink, **bw1k  , max_queue_size=buff,use_htb=True)
```

Hasil Running:

```
artisabunga@artisabunga-VirtualBox: ~/Downloads
h1 (cfs -1/100000us) h2 (cfs -1/100000us) r1 (cfs -1/100000us) r2 (cfs -1/100000
us) r3 (cfs -1/100000us) r4 (cfs -1/100000us)
*** Starting controller
c0
*** Starting 0 switches

*** h2 : ('iperf -s &',)
*** h1 : ('iperf -t 60 -c 192.168.4.2 &',)
*** Starting CLI:
mininet> h2 ping h1 -c 20
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 192.168.4.2 port 5001 connected with 192.168.0.1 port 33522
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_seq=1 ttl=62 time=1655 ms
64 bytes from 192.168.0.1: icmp_seq=2 ttl=62 time=1879 ms
64 bytes from 192.168.0.1: icmp_seq=3 ttl=62 time=2262 ms
64 bytes from 192.168.0.1: icmp_seq=4 ttl=62 time=2789 ms
64 bytes from 192.168.0.1: icmp_seq=5 ttl=62 time=3285 ms
64 bytes from 192.168.0.1: icmp_seq=6 ttl=62 time=3764 ms
64 bytes from 192.168.0.1: icmp_seq=7 ttl=62 time=4242 ms
64 bytes from 192.168.0.1: icmp_seq=15 ttl=62 time=3821 ms
64 bytes from 192.168.0.1: icmp_seq=17 ttl=62 time=3760 ms
64 bytes from 192.168.0.1: icmp_seq=19 ttl=62 time=3917 ms

--- 192.168.0.1 ping statistics ---
20 packets transmitted, 10 received, 50% packet loss, time 19190ms
rtt min/avg/max/mdev = 1655.499/3137.362/4242.477/881.250 ms, pipe 5
mininet>
```

Dari hasil percobaan Running dari 4 nilai buffer yang berbeda diatas didapat hasil data packet loss seperti pada table dibawah:

Besar buffer	Packet transmitted	Packet received	Packet loss	Waktu delay
20	20	14	30%	19133 ms
40	20	14	30%	19183 ms
60	20	9	55%	19277 ms
100	20	10	50%	19190 ms

Sehingga dapat disimpulkan bahwa pengaruh buffer yang digunakan terhadap paket loss adalah semakin besar buffer yang digunakan, maka persentase packet loss selama transfer data akan semakin besar

Dan pengaruh buffer terhadap waktu delay adalah, semakin besar buffer yang digunakan maka waktu delay dalam transfer data akan semakin lama juga.