LAPORAN SIMULASI PADA MININET

Tugas Besar Mata Kuliah Jaringan Komputer

Disusun oleh:

Berlian Muhammad Galin Al Awienoor (1301204378)

PROGRAM STUDI S1 INFORMATIKA
FAKULTAS INFORMATIKA
UNIVERSITAS TELKOM
2021/2022

DAFTAR ISI

PENDAHULUAN

Selama hampir satu dekade terakhir, jumlah dari pengguna atau perangkat yang terkoneksi dengan internet atau aplikasi serta layanan yang berjalan pada sebuah jaringan di seluruh dunia meningkat dengan pesat. Akan tetapi, konfigurasi jaringan atau protokol yang mendasarinya tidak banyak berubah secara signifikan. Contohnya, proses routing yang biasanya didasarkan pada IP prefix tujuan dan manajemen jaringan yang masih kaku karena dibatasi oleh perbedaan konfigurasi masing-masing vendor. Oleh karena itu, ketika sebuah aturan telah diterapkan pada sebuah sistem jaringan, satu-satunya cara untuk mengubah konfigurasi yaitu dengan mengubah konfigurasi pada keseluruhan perangkat yang terkoneksi. Sehingga sangat menyita waktu dan keterbatasan pada skalabilitas dan mobilitas jaringan dan pengguna. Sedangkan saat ini, kebutuhan akan jaringan yang cepat, fleksibel, handal dan tanpa batasan vendor sangat diperlukan. Pada laporan ini, saya akan membahas tentang hasil tugas besar jaringan komputer yang telah saya kerjakan sesuai dengan spesifikasi tugas besar yang diberikan.

A. Topologi Jaringan

Topologi adalah suatu aturan bagaimana menghubungkan komputer satu sama lain secara fisik dan pola hubungannya antara komponen - komponen yang berkomunikasi melalui media atau peralatan jaringan baik secara fisik (*physical topology*) maupun secara logika (*logic topology*).

B. Mininet

Mininet adalah emulator jaringan SDN yang dapat mensimulasikan kinerja antara *end-host, switch, router, controler*, dan *link* dalam sebuah kernel Linux. Mininet merupakan sebuah sistem virtualisasi yang dapat menggambarkan jaringan yang besar dengan hanya menggunakan sebuah laptop. Mininet bersifat open source, sehingga proyek yang telah dilakukan berupa *source code, scripts*, dan dokumentasi yang dapat dikembangkan oleh siapapun. Tiap perangkat virtual yang dijalankan oleh Mininet berperilaku seperti mesin sungguhan, yang berarti pengguna dapat mengkonfigurasi tiap perangkat virtual tersebut layaknya *physical device*. Pengguna juga dapat menentukan kecepatan, bandwidth, serta delay.

C. *Routing*

Routing merupakan metode yang digunakan untuk meneruskan paket-paket dari satu jaringan ke jaringan yang berbeda melalui sebuah internetwork. Untuk melakukan hal ini, digunakan sebuah perangkat jaringan bernama router dimana router-router yang berada pada jaringan akan menghubungkan jaringan satu dan yang lainnya agar dapat saling bertukar paket. Agar routing yang dilakukan dapat secara benar mengantarkan paket yang dimaksud, router diharuskan mengenal topologi jaringan yang ada. Terdapat 2 fungsi dasar dari routing yakni sebagai penentu jalur yang akan dilalui paket hingga ke tujuan dan melakukan fungsi *switching* karena sifatnya yang dapat meneruskan paket. Terdapat beberapa jenis metode routing yang ada, yaitu *Static Routing* dan *Dynamic Routing*. Pada tugas besar jaringan komputer ini saya menggunakan pengimplementasian *Static Routing*, dimana metode perutean terhadap jaringan dengan tabel jaringan dibangun secara manual, sehingga diharuskan untuk memasukan *command* secara manual di router setiap yang akan dilakukan perubahan jalur pengantaran paket.

D. *Transmission Control Protocol*

TCP adalah standar komunikasi yang memungkinkan program aplikasi dan perangkat komputer dapat bertukar informasi melalui jaringan. TCP dirancang untuk mengirimkan data/informasi dan memastikannya terkirim lewat jaringan. Bisa dibilang, *Transmission Control Protocol* ini adalah inti dari sebuah internet protocol. Server dan klien dapat saling mentransmisikan data yang telah diatur oleh TCP, integritas data/informasi yang dikirimkan melalui jaringan juga akan terjamin. Sebelum mentransmisikan data, TCP membuat koneksi antara sumber dan tujuannya, kemudian barulah menguraikan data besar menjadi lebih kecil dan memastikan integritas data selama proses transmisi berlangsung.

E. *Network Scheduler*

Network Scheduler atau juga disebut *packet scheduler*, disiplin antrian, *qdisc* atau *queue* adalah sebuah pengatur pada node dalam jaringan komunikasi *packet switching*. *Network Scheduler* mengatur *sequence* dari paket-paket jaringan dalam antrian dikirim dan diterima dari interface jaringan. Logika dari *Network Scheduler* memutuskan untuk memilih paket yang mana untuk diteruskan terlebih dahulu. Sistem mungkin memiliki satu atau lebih antrian yang mungkin menyimpan paket dalam satu alur, klasifikasi, atau prioritas. Terdapat banyak jenis *queue* yang dapat digunakan pada jaringan, contoh jenis *queue* yaitu CBQ (*Class Based Queue*) dan HTB (*Hierarchical Token Bucket*). CBQ merupakan disiplin antrian yang memungkinkan *traffic* untuk membagi *bandwidth* secara merata, setelah dibagi ke beberapa kelas. HTB merupakan pengganti antrian yang lebih bagus dari CBQ di Linux yang berguna untuk membatasi kecepatan unduh atau unggah klien sehingga klien terbatas tidak dapat memenuhi total *bandwidth*.

PEMBAHASAN

A. CLO 1

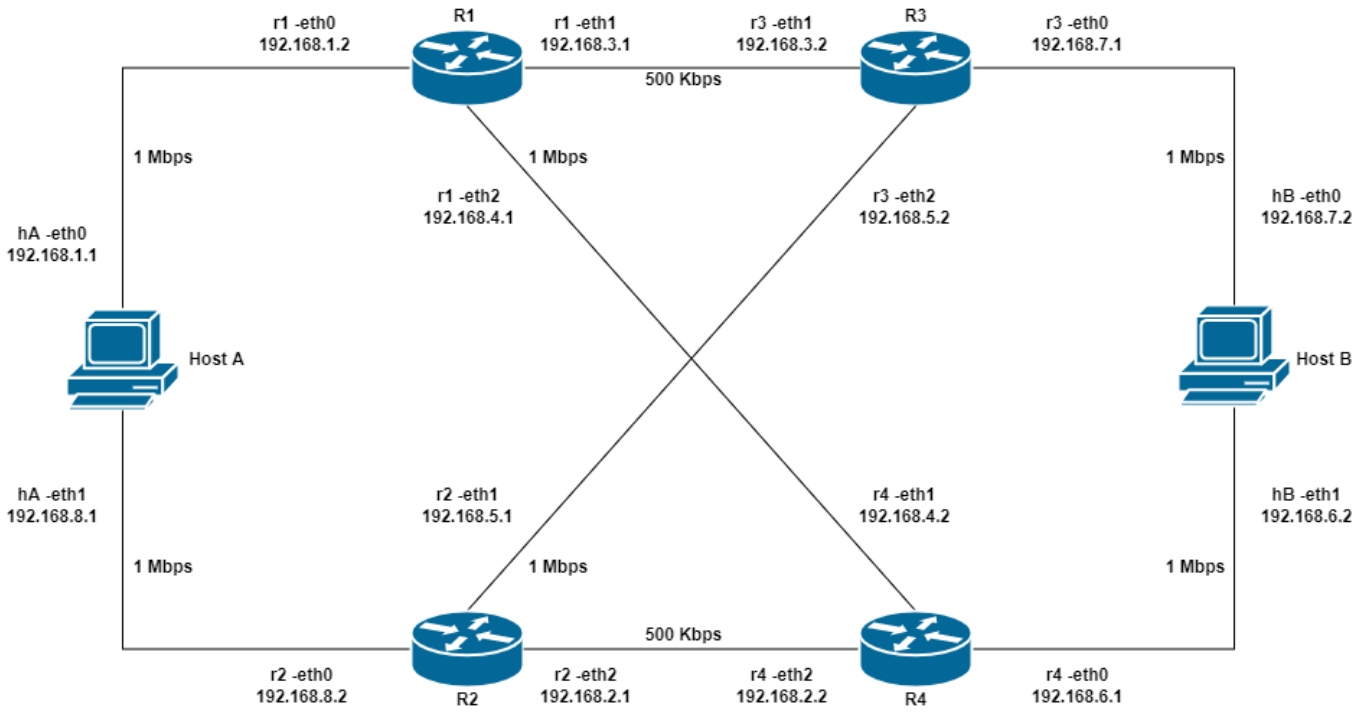
Pada CLO ini terdapat spesifikasi pengerjaan dan kriteria penilaian yang akan dilakukan.

Goal :

- *Build topology* sesuai dengan soal.
- Desain subnet masing-masing *network*.
- *Assign* IP sesuai subnet.
- Uji konektivitas dengan ping antara 2 host yang berada dalam 1 network.

1. Build Topologi

Berikut adalah topologi yang saya bangun sedemikian rupa dengan mengikuti ketentuan skenario pada soal tugas besar jaringan komputer.



2. Tabel Subnetting

Berikut adalah tabel subnetting yang saya kerjakan sesuai dengan topologi yang dibangun.

NAMA	NEEDS	ALOKASI	IP ADDRESS	HOST RANGE	PREFIX	SUBNET MASK	BROADCAST
NET 1	2	256	192.168.1.0	192.168.1.1 - 192.168.1.254	/24	255.255.255.0	192.168.1.255
NET 2	2	256	192.168.2.0	192.168.2.1 - 192.168.2.254	/24	255.255.255.0	192.168.2.255
NET 3	2	256	192.168.3.0	192.168.3.1 - 192.168.3.254	/24	255.255.255.0	192.168.3.255
NET 4	2	256	192.168.4.0	192.168.4.1 - 192.168.4.254	/24	255.255.255.0	192.168.4.255
NET 5	2	256	192.168.5.0	192.168.5.1 - 192.168.5.254	/24	255.255.255.0	192.168.5.255
NET 6	2	256	192.168.6.0	192.168.6.1 - 192.168.6.254	/24	255.255.255.0	192.168.6.255
NET 7	2	256	192.168.7.0	192.168.7.1 - 192.168.7.254	/24	255.255.255.0	192.168.7.255
NET 8	2	256	192.168.8.0	192.168.8.1 - 192.168.8.254	/24	255.255.255.0	192.168.8.255

3. Implementasi Topologi pada Mininet

Selanjutnya adalah membuat atau membangun topologi serta mengkonfigurasi IP Address dan router pada Mininet yang sesuai dengan tabel subnetting dan topologi yang dibangun sebelumnya. Berikut kode program yang telah saya kerjakan.

```
#Build Topologi
class MyTopo(Topo):
  def __init__(self, **opts):
    Topo.__init__(self, **opts)

    #membuat host
    #menambahkan host A
    hA = self.addHost( 'hA' )
    #menambahkan host B
    hB = self.addHost( 'hB' )

    #membuat router
    #menambahkan router 1
    r1 = self.addHost( 'r1' )
    #menambahkan router 2
    r2 = self.addHost( 'r2' )
    #menambahkan router 3
    r3 = self.addHost( 'r3' )
    #menambahkan router 4
    r4 = self.addHost( 'r4' )
```

```
#membuat link
#link host - router
#menghubungkan hA dengan r1
self.addLink( 'hA', 'r1', intfName1='hA-eth0', intfName2='r1-eth0', cls=TCLink, bw=1 )
#menghubungkan hA dengan r2
self.addLink( 'hA', 'r2', intfName1='hA-eth1', intfName2='r2-eth0', cls=TCLink, bw=1 )
#menghubungkan hB dengan r3
self.addLink( 'hB', 'r3', intfName1='hB-eth0', intfName2='r3-eth0', cls=TCLink, bw=1 )
#menghubungkan hB dengan r4
self.addLink( 'hB', 'r4', intfName1='hB-eth1', intfName2='r4-eth0', cls=TCLink, bw=1 )

#link router - router
#menghubungkan r1 dengan r3
self.addLink( 'r1', 'r3', intfName1='r1-eth1', intfName2='r3-eth1', cls=TCLink, bw=0.5 )
#menghubungkan r1 dengan r4
self.addLink( 'r1', 'r4', intfName1='r1-eth2', intfName2='r4-eth1', cls=TCLink, bw=1 )
#menghubungkan r2 dengan r3
self.addLink( 'r2', 'r3', intfName1='r2-eth1', intfName2='r3-eth2', cls=TCLink, bw=1 )
#menghubungkan r2 dengan r4
self.addLink( 'r2', 'r4', intfName1='r2-eth2', intfName2='r4-eth2', cls=TCLink, bw=0.5 )

#konfigurasi IP hA
hA.cmd( "ifconfig hA-eth0 0" )
hA.cmd( "ifconfig hA-eth1 0" )
hA.cmd( "ifconfig hA-eth0 192.168.1.1 netmask 255.255.255.0" )
hA.cmd( "ifconfig hA-eth1 192.168.8.1 netmask 255.255.255.0" )

#konfigurasi IP hB
hB.cmd( "ifconfig hB-eth0 0" )
hB.cmd( "ifconfig hB-eth1 0" )
hB.cmd( "ifconfig hB-eth0 192.168.7.2 netmask 255.255.255.0" )
hB.cmd( "ifconfig hB-eth1 192.168.6.2 netmask 255.255.255.0" )

#konfigurasi IP r1
r1.cmd( "ifconfig r1-eth0 0" )
r1.cmd( "ifconfig r1-eth1 0" )
r1.cmd( "ifconfig r1-eth2 0" )
r1.cmd( "ifconfig r1-eth0 192.168.1.2 netmask 255.255.255.0" )
r1.cmd( "ifconfig r1-eth1 192.168.3.1 netmask 255.255.255.0" )
r1.cmd( "ifconfig r1-eth2 192.168.4.1 netmask 255.255.255.0" )

#konfigurasi IP r2
r2.cmd( "ifconfig r2-eth0 0" )
r2.cmd( "ifconfig r2-eth1 0" )
r2.cmd( "ifconfig r2-eth2 0" )
r2.cmd( "ifconfig r2-eth0 192.168.8.2 netmask 255.255.255.0" )
r2.cmd( "ifconfig r2-eth1 192.168.5.1 netmask 255.255.255.0" )
r2.cmd( "ifconfig r2-eth2 192.168.2.1 netmask 255.255.255.0" )

#konfigurasi IP r3
r3.cmd( "ifconfig r3-eth0 0" )
r3.cmd( "ifconfig r3-eth1 0" )
r3.cmd( "ifconfig r3-eth2 0" )
r3.cmd( "ifconfig r3-eth0 192.168.7.1 netmask 255.255.255.0" )
r3.cmd( "ifconfig r3-eth1 192.168.3.2 netmask 255.255.255.0" )
r3.cmd( "ifconfig r3-eth2 192.168.5.2 netmask 255.255.255.0" )

#konfigurasi IP r4
r4.cmd( "ifconfig r4-eth0 0" )
r4.cmd( "ifconfig r4-eth1 0" )
r4.cmd( "ifconfig r4-eth2 0" )
r4.cmd( "ifconfig r4-eth0 192.168.6.1 netmask 255.255.255.0" )
r4.cmd( "ifconfig r4-eth1 192.168.4.2 netmask 255.255.255.0" )
r4.cmd( "ifconfig r4-eth2 192.168.2.2 netmask 255.255.255.0" )
```

```
#konfigurasi router
r1.cmd( "echo 1 > /proc/sys/net/ipv4/ip_forward" )
r2.cmd( "echo 1 > /proc/sys/net/ipv4/ip_forward" )
r3.cmd( "echo 1 > /proc/sys/net/ipv4/ip_forward" )
r4.cmd( "echo 1 > /proc/sys/net/ipv4/ip_forward" )
```

4. Uji Konektivitas

Berikut adalah hasil uji konektivitas dengan menggunakan net dan ping termasuk antara dua host yaitu hostA dan hostB sesuai dengan topologi yang dibangun dari kode program yang sudah saya kerjakan untuk membangun topologi pada Mininet.

```
*** Creating network
*** Adding controller
*** Adding hosts:
hA hB r1 r2 r3 r4
*** Adding switches:

*** Adding links:
(1.00Mbit) (1.00Mbit) (hA, r1) (1.00Mbit) (1.00Mbit) (hA, r2) (1.00Mbit) (1.00Mbit) (hB, r3) (1.00Mbit)
(1.00Mbit) (hB, r4) (0.50Mbit) (0.50Mbit) (r1, r3) (1.00Mbit) (1.00Mbit) (r1, r4) (1.00Mbit) (1.00Mbit)
(r2, r3) (0.50Mbit) (0.50Mbit) (r2, r4)
*** Configuring hosts
hA (cfs -1/100000us) hB (cfs -1/100000us) r1 (cfs -1/100000us) r2 (cfs -1/100000us) r3 (cfs -1/100000us)
 r4 (cfs -1/100000us)
*** Starting controller
c0
*** Starting 0 switches

*** Starting CLI:
mininet> net
hA hA-eth0:r1-eth0 hA-eth1:r2-eth0
hB hB-eth0:r3-eth0 hB-eth1:r4-eth0
r1 r1-eth0:hA-eth0 r1-eth1:r3-eth1 r1-eth2:r4-eth1
r2 r2-eth0:hA-eth1 r2-eth1:r3-eth2 r2-eth2:r4-eth2
r3 r3-eth0:hB-eth0 r3-eth1:r1-eth1 r3-eth2:r2-eth1
r4 r4-eth0:hB-eth1 r4-eth1:r1-eth2 r4-eth2:r2-eth2
c0
```

```
mininet> hA ping r1 -c 3
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=0.055 ms
64 bytes from 192.168.1.2: icmp_seq=2 ttl=64 time=0.058 ms
64 bytes from 192.168.1.2: icmp_seq=3 ttl=64 time=0.048 ms

--- 192.168.1.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2031ms
rtt min/avg/max/mdev = 0.048/0.053/0.058/0.004 ms
mininet> hA ping r2 -c 3
PING 192.168.8.2 (192.168.8.2) 56(84) bytes of data.
64 bytes from 192.168.8.2: icmp_seq=1 ttl=64 time=0.090 ms
64 bytes from 192.168.8.2: icmp_seq=2 ttl=64 time=0.065 ms
64 bytes from 192.168.8.2: icmp_seq=3 ttl=64 time=0.061 ms

--- 192.168.8.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2036ms
rtt min/avg/max/mdev = 0.061/0.072/0.090/0.012 ms
```

```
mininet> hB ping r3 -c 3
PING 192.168.7.1 (192.168.7.1) 56(84) bytes of data.
64 bytes from 192.168.7.1: icmp_seq=1 ttl=64 time=0.081 ms
64 bytes from 192.168.7.1: icmp_seq=2 ttl=64 time=0.066 ms
64 bytes from 192.168.7.1: icmp_seq=3 ttl=64 time=0.057 ms

--- 192.168.7.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2035ms
rtt min/avg/max/mdev = 0.057/0.068/0.081/0.009 ms
mininet> hB ping r4 -c 3
PING 192.168.6.1 (192.168.6.1) 56(84) bytes of data.
64 bytes from 192.168.6.1: icmp_seq=1 ttl=64 time=0.055 ms
64 bytes from 192.168.6.1: icmp_seq=2 ttl=64 time=0.053 ms
64 bytes from 192.168.6.1: icmp_seq=3 ttl=64 time=0.085 ms

--- 192.168.6.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2035ms
rtt min/avg/max/mdev = 0.053/0.064/0.085/0.014 ms
```

```
mininet> r1 ping r3 -c 3
PING 192.168.7.1 (192.168.7.1) 56(84) bytes of data.
64 bytes from 192.168.7.1: icmp_seq=1 ttl=64 time=0.051 ms
64 bytes from 192.168.7.1: icmp_seq=2 ttl=64 time=0.058 ms
64 bytes from 192.168.7.1: icmp_seq=3 ttl=64 time=0.071 ms

--- 192.168.7.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2045ms
rtt min/avg/max/mdev = 0.051/0.060/0.071/0.008 ms
mininet> r1 ping r4 -c 3
PING 192.168.6.1 (192.168.6.1) 56(84) bytes of data.
64 bytes from 192.168.6.1: icmp_seq=1 ttl=64 time=0.055 ms
64 bytes from 192.168.6.1: icmp_seq=2 ttl=64 time=0.058 ms
64 bytes from 192.168.6.1: icmp_seq=3 ttl=64 time=0.066 ms

--- 192.168.6.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2029ms
rtt min/avg/max/mdev = 0.055/0.059/0.066/0.004 ms
```

```
mininet> r2 ping r3 -c 3
PING 192.168.7.1 (192.168.7.1) 56(84) bytes of data.
64 bytes from 192.168.7.1: icmp_seq=1 ttl=64 time=0.063 ms
64 bytes from 192.168.7.1: icmp_seq=2 ttl=64 time=0.049 ms
64 bytes from 192.168.7.1: icmp_seq=3 ttl=64 time=0.065 ms

--- 192.168.7.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2046ms
rtt min/avg/max/mdev = 0.049/0.059/0.065/0.007 ms
mininet> r2 ping r4 -c 3
PING 192.168.6.1 (192.168.6.1) 56(84) bytes of data.
64 bytes from 192.168.6.1: icmp_seq=1 ttl=64 time=0.058 ms
64 bytes from 192.168.6.1: icmp_seq=2 ttl=64 time=0.066 ms
64 bytes from 192.168.6.1: icmp_seq=3 ttl=64 time=0.064 ms

--- 192.168.6.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2026ms
rtt min/avg/max/mdev = 0.058/0.062/0.066/0.003 ms
```

B. CLO 2

Pada CLO ini terdapat spesifikasi pengerjaan dan kriteria penilaian yang akan dilakukan.

Goal :

- Mengimplementasikan mekanisme Routing pada topologi yang ada.
- Uji konektivitas menggunakan ping.
- Membuat tabel routing di semua host, dibuktikan dengan ping antar host.
- Menganalisis routing yang digunakan menggunakan traceroute

1. Implementasi Mekanisme Routing

Berikut adalah kode program implementasi mekanisme *routing* yang telah saya kerjakan. Pada tugas besar jaringan komputer ini, saya menggunakan pengimplementasian *static routing* dari topologi yang sudah dibangun serta konfigurasi IP Address dan router yang telah saya kerjakan pada CLO 1.

```
#static routing
#host A
hA.cmd( "ip rule add from 192.168.1.1 table 1" )
hA.cmd( "ip rule add from 192.168.8.1 table 2" )

hA.cmd( "ip route add 192.168.1.0/24 dev hA-eth0 scope link table 1" )
hA.cmd( "ip route add default via 192.168.1.2 dev hA-eth0 table 1" )

hA.cmd( "ip route add 192.168.8.0/24 dev hA-eth1 scope link table 2" )
hA.cmd( "ip route add default via 192.168.8.2 dev hA-eth1 table 2" )

hA.cmd( "ip route add default scope global nexthop via 192.168.1.2 dev hA-eth0" )
hA.cmd( "ip route add default scope global nexthop via 192.168.8.2 dev hA-eth1" )

#host B
hB.cmd( "ip rule add from 192.168.7.2 table 1" )
hB.cmd( "ip rule add from 192.168.6.2 table 2" )

hB.cmd( "ip route add 192.168.7.0/24 dev hB-eth0 scope link table 1" )
hB.cmd( "ip route add default via 192.168.7.1 dev hB-eth0 table 1" )

hB.cmd( "ip route add 192.168.6.0/24 dev hB-eth1 scope link table 2" )
hB.cmd( "ip route add default via 192.168.6.1 dev hB-eth1 table 2" )

hB.cmd( "ip route add default scope global nexthop via 192.168.7.1 dev hB-eth0" )
hB.cmd( "ip route add default scope global nexthop via 192.168.6.1 dev hB-eth1" )
```

```
#menyetting gateaway router
#router 1
r1.cmd( "route add -net 192.168.8.0/24 gw 192.168.3.2" )
r1.cmd( "route add -net 192.168.5.0/24 gw 192.168.3.2" )
r1.cmd( "route add -net 192.168.2.0/24 gw 192.168.4.2" )
r1.cmd( "route add -net 192.168.7.0/24 gw 192.168.3.2" )
r1.cmd( "route add -net 192.168.6.0/24 gw 192.168.4.2" )

#router 2
r2.cmd( "route add -net 192.168.1.0/24 gw 192.168.2.2" )
r2.cmd( "route add -net 192.168.4.0/24 gw 192.168.2.2" )
r2.cmd( "route add -net 192.168.3.0/24 gw 192.168.5.2" )
r2.cmd( "route add -net 192.168.7.0/24 gw 192.168.5.2" )
r2.cmd( "route add -net 192.168.6.0/24 gw 192.168.2.2" )

#router 3
r3.cmd( "route add -net 192.168.6.0/24 gw 192.168.3.1" )
r3.cmd( "route add -net 192.168.4.0/24 gw 192.168.3.1" )
r3.cmd( "route add -net 192.168.2.0/24 gw 192.168.5.1" )
r3.cmd( "route add -net 192.168.8.0/24 gw 192.168.5.1" )
r3.cmd( "route add -net 192.168.1.0/24 gw 192.168.3.1" )

#router 4
r4.cmd( "route add -net 192.168.7.0/24 gw 192.168.2.1" )
r4.cmd( "route add -net 192.168.5.0/24 gw 192.168.2.1" )
r4.cmd( "route add -net 192.168.3.0/24 gw 192.168.4.1" )
r4.cmd( "route add -net 192.168.1.0/24 gw 192.168.4.1" )
r4.cmd( "route add -net 192.168.8.0/24 gw 192.168.2.1" )
```

2. Uji Konektivitas

Berikut adalah hasil uji konektivitas antara semua *network* di dalam topologi yang sudah dibangun dari kode program yang sudah saya kerjakan dengan menggunakan pingall.

```
mininet> pingall
*** Ping: testing ping reachability
hA -> hB r1 r2 r3 r4
hB -> hA r1 r2 r3 r4
r1 -> hA hB r2 r3 r4
r2 -> hA hB r1 r3 r4
r3 -> hA hB r1 r2 r4
r4 -> hA hB r1 r2 r3
*** Results: 0% dropped (30/30 received)
```

3. Ping antar Host

Berikut adalah hasil uji konektivitas antara dua host yaitu dari hostA ke hostB, begitu juga sebaliknya dengan menggunakan ping.

```
mininet> hA ping hB -c 7
PING 192.168.7.2 (192.168.7.2) 56(84) bytes of data.
64 bytes from 192.168.7.2: icmp_seq=1 ttl=62 time=0.056 ms
64 bytes from 192.168.7.2: icmp_seq=2 ttl=62 time=0.080 ms
64 bytes from 192.168.7.2: icmp_seq=3 ttl=62 time=0.064 ms
64 bytes from 192.168.7.2: icmp_seq=4 ttl=62 time=0.060 ms
64 bytes from 192.168.7.2: icmp_seq=5 ttl=62 time=0.067 ms
64 bytes from 192.168.7.2: icmp_seq=6 ttl=62 time=0.054 ms
64 bytes from 192.168.7.2: icmp_seq=7 ttl=62 time=0.064 ms

--- 192.168.7.2 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6128ms
rtt min/avg/max/mdev = 0.054/0.063/0.080/0.008 ms
```

```
mininet> hB ping hA -c 7
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=62 time=0.060 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=62 time=0.066 ms
64 bytes from 192.168.1.1: icmp_seq=3 ttl=62 time=0.065 ms
64 bytes from 192.168.1.1: icmp_seq=4 ttl=62 time=0.067 ms
64 bytes from 192.168.1.1: icmp_seq=5 ttl=62 time=0.079 ms
64 bytes from 192.168.1.1: icmp_seq=6 ttl=62 time=0.073 ms
64 bytes from 192.168.1.1: icmp_seq=7 ttl=62 time=0.078 ms

--- 192.168.1.1 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6133ms
rtt min/avg/max/mdev = 0.060/0.069/0.079/0.006 ms
```

4. Traceroute

Berikut adalah hasil uji konektivitas antara dua host yaitu dari hostA ke hostB, begitu juga sebaliknya dengan menggunakan traceroute.

```
mininet> hA traceroute hB
traceroute to 192.168.7.2 (192.168.7.2), 30 hops max, 60 byte packets
 1  192.168.1.2 (192.168.1.2)  0.265 ms  0.221 ms  0.212 ms
 2  192.168.3.2 (192.168.3.2)  0.202 ms  0.183 ms  0.171 ms
 3  192.168.7.2 (192.168.7.2)  0.161 ms  0.141 ms  0.127 ms
mininet> hB traceroute hA
traceroute to 192.168.1.1 (192.168.1.1), 30 hops max, 60 byte packets
 1  192.168.7.1 (192.168.7.1)  0.307 ms  0.261 ms  0.240 ms
 2  192.168.3.1 (192.168.3.1)  0.230 ms  0.210 ms  0.198 ms
 3  192.168.1.1 (192.168.1.1)  0.169 ms  0.137 ms  0.123 ms
```

Jika dilihat dari *screenshot* diatas, terdapat perbedaan mengenai uji konektivitas antara penggunaan ping dengan traceroute. Jika menggunakan ping hanya akan menampilkan hasil host pengecekan *network* (terhubung atau tidak), sedangkan jika menggunakan traceroute, akan menampilkan urutan jalur yang dilalui untuk mencapai host yang dituju agar bisa terhubung.

C. CLO 3

Pada CLO ini terdapat spesifikasi pengerjaan dan kriteria penilaian yang akan dilakukan.
Goal :

- Membuktikan bahwa TCP telah diimplementasikan dengan benar pada topologi.
- Generate *traffic* menggunakan iPerf.
- Capture trafik menggunakan custom script atau Wireshark untuk diinspeksi, dibuktikan dengan trafik di Wireshark/tcpdump.

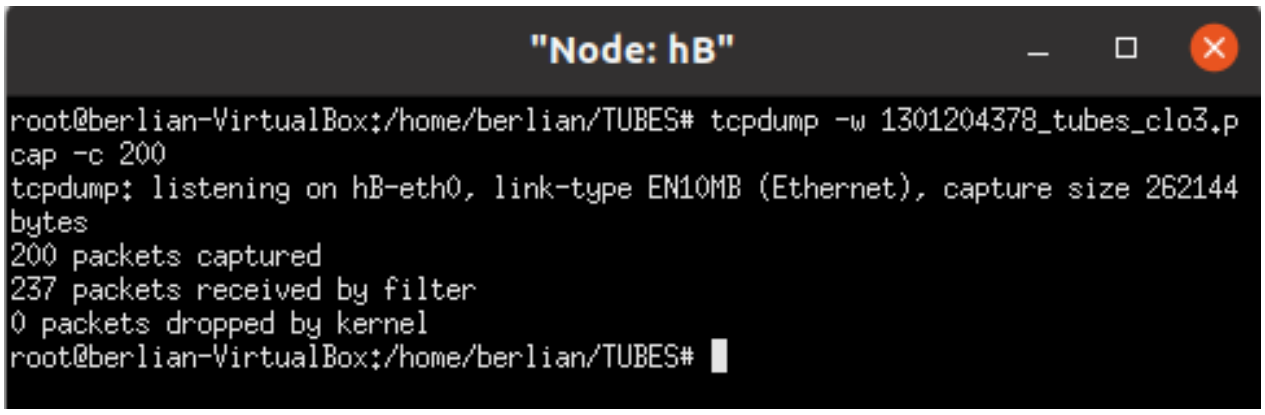
1. Traffic dengan menggunakan iPerf

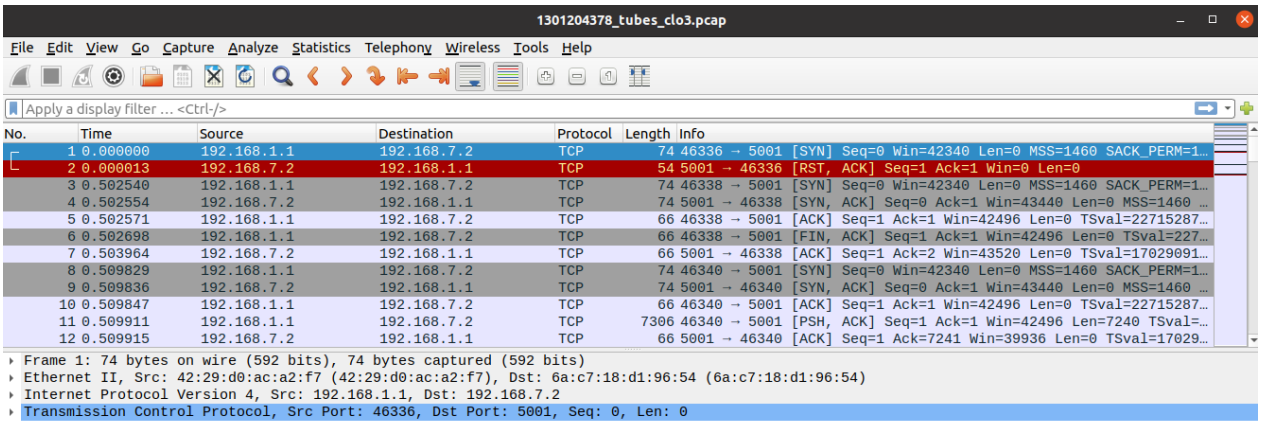
Berikut adalah hasil *generate traffic* dengan menggunakan iPerf pada kode program yang telah saya kerjakan.

```
mininet> xterm hB
mininet> iperf hA hB
*** Iperf: testing TCP bandwidth between hA and hB
.*** Results: ['478 Kbits/sec', '1.12 Mbits/sec']
```

2. Traffic dengan menggunakan Wireshark

Berikut adalah hasil dari meng-*capture traffic* dengan menggunakan Wireshark untuk diinspeksi.






```
0000  6a c7 18 d1 96 54 42 29  d0 ac a2 f7 08 00 45 10   j····TB) ·····E·
0010  00 3c c2 50 40 00 3e 06  f1 07 c0 a8 01 01 c0 a8   ·<·P@·>· ········
0020  07 02 b5 00 13 89 2e 11  bd cd 00 00 00 00 a0 02   ·····.· ········
0030  a5 64 89 82 00 00 02 04  05 b4 04 02 08 0a 87 64   ·d····· ·······d
0040  c5 15 00 00 00 00 01 03  03 09                     ····· ··
```

```
Ethernet II, Src: 42:29:d0:ac:a2:f7 (42:29:d0:ac:a2:f7), Dst: 6a:c7:18:d1:96:54 (6a:c7:18:d1:96:54)
  Destination: 6a:c7:18:d1:96:54 (6a:c7:18:d1:96:54)
  Source: 42:29:d0:ac:a2:f7 (42:29:d0:ac:a2:f7)
  Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 192.168.1.1, Dst: 192.168.7.2
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  Differentiated Services Field: 0x10 (DSCP: Unknown, ECN: Not-ECT)
  Total Length: 60
  Identification: 0xc250 (49744)
  Flags: 0x4000, Don't fragment
  Fragment offset: 0
  Time to live: 62
  Protocol: TCP (6)
  Header checksum: 0xf107 [validation disabled]
  [Header checksum status: Unverified]
  Source: 192.168.1.1
  Destination: 192.168.7.2

Transmission Control Protocol, Src Port: 46336, Dst Port: 5001, Seq: 0, Len: 0
  Source Port: 46336
  Destination Port: 5001
  [Stream index: 0]
  [TCP Segment Len: 0]
  Sequence number: 0    (relative sequence number)
  Sequence number (raw): 772914637
  [Next sequence number: 1    (relative sequence number)]
  Acknowledgment number: 0
  Acknowledgment number (raw): 0
  1010 .... = Header Length: 40 bytes (10)
  Flags: 0x002 (SYN)
  Window size value: 42340
  [Calculated window size: 42340]
  Checksum: 0x8982 [unverified]
  [Checksum Status: Unverified]
  Urgent pointer: 0
  Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale
  [Timestamps]
```

Jika dilihat dari hasil *capture* sesuai *screenshot* diatas, dapat ditentukan bahwa protokol yang digunakan pada topologi yang dibuat menggunakan protokol TCP (dilihat dengan menggunakan Wireshark).

Ditemukan pula TCP yang menggunakan metode *Three-Way Handshake* (diketahui SYN - SYN ACK - ACK) itu adalah cara yang dilakukan oleh protokol TCP untuk melakukan pertukaran data/paket antar host atau antar jaringan. Proses *Three-Way Handshake* dari hasil *capture traffic* diatas yaitu paket nomor 3,4,5 dan paket nomor 8,9,10. Berikut penjelasan proses *Three-Way Handshake* pada paket 3,4,5.

1. Host pertama 192.168.1.1 dengan tujuan paket 192.168.7.2 akan mengirim segmen TCP dengan flag SYN ke host kedua yaitu 192.168.7.2
2. Host kedua kemudian akan merespon dengan mengirim kembali segmen TCP dengan flag SYN kepada host pertama (SYN-ACK)
3. Terjadilah pertukaran data antara host pertama dan kedua yang telah terhubung (ACK)

3. Traffic dengan menggunakan Tcpdump

Berikut adalah hasil dari meng-*capture traffic* dengan menggunakan tcpdump untuk diinspeksi.

```
root@berlian-VirtualBox:/home/berlian# cd TUBES/
root@berlian-VirtualBox:/home/berlian/TUBES# tcpdump -r 1301204378_tubes_clo3.pcap tcp
reading from file 1301204378_tubes_clo3.pcap, link-type EN10MB (Ethernet)
14:30:32.931573 IP 192.168.1.1.46336 > 192.168.7.2.5001: Flags [S], seq 772914637, win 42340, options [mss 1460,sackOK,TS val 2271528213 ecr 0,nop,wscale 9], length 0
14:30:32.931586 IP 192.168.7.2.5001 > 192.168.1.1.46336: Flags [R.], seq 0, ack 772914638, win 0, length 0
14:30:33.434113 IP 192.168.1.1.46338 > 192.168.7.2.5001: Flags [S], seq 1065517037, win 42340, options [mss 1460,sackOK,TS val 2271528716 ecr 0,nop,wscale 9], length 0
14:30:33.434127 IP 192.168.7.2.5001 > 192.168.1.1.46338: Flags [S.], seq 3430428567, ack 1065517038, win 43440, options [mss 1460,sackOK,TS val 1702909168 ecr 2271528716,nop,wscale 9], length 0
14:30:33.434144 IP 192.168.1.1.46338 > 192.168.7.2.5001: Flags [.], ack 1, win 83, options [nop,nop,TS val 2271528716 ecr 1702909168], length 0
14:30:33.434271 IP 192.168.1.1.46338 > 192.168.7.2.5001: Flags [F.], seq 1, ack 1, win 83, options [nop,nop,TS val 2271528716 ecr 1702909168], length 0
14:30:33.435537 IP 192.168.7.2.5001 > 192.168.1.1.46338: Flags [.], ack 2, win 85, options [nop,nop,TS val 1702909169 ecr 2271528716], length 0
14:30:33.441402 IP 192.168.1.1.46340 > 192.168.7.2.5001: Flags [S], seq 1918377683, win 42340, options [mss 1460,sackOK,TS val 2271528723 ecr 0,nop,wscale 9], length 0
14:30:33.441409 IP 192.168.7.2.5001 > 192.168.1.1.46340: Flags [S.], seq 4131138101, ack 1918377684, win 43440, options [mss 1460,sackOK,TS val 1702909175 ecr 2271528723,nop,wscale 9], length 0
14:30:33.441420 IP 192.168.1.1.46340 > 192.168.7.2.5001: Flags [.], ack 1, win 83, options [nop,nop,TS val 2271528723 ecr 1702909175], length 0
14:30:33.441484 IP 192.168.1.1.46340 > 192.168.7.2.5001: Flags [P.], seq 1:7241, ack 1, win 83, options [nop,nop,TS val 2271528723 ecr 1702909175], length 7240
14:30:33.441488 IP 192.168.7.2.5001 > 192.168.1.1.46340: Flags [.], ack 7241, win 78, options [nop,nop,TS val 1702909175 ecr 2271528723], length 0
14:30:33.444590 IP 192.168.7.2.5001 > 192.168.1.1.46338: Flags [F.], seq 1, ack 2, win 85, options [nop,nop,TS val 1702909178 ecr 2271528716], length 0
14:30:33.539182 IP 192.168.1.1.46340 > 192.168.7.2.5001: Flags [P.], seq 7241:14481, ack 1, win 83, options [nop,nop,TS val 2271528723 ecr 1702909175], length 7240
14:30:33.539199 IP 192.168.7.2.5001 > 192.168.1.1.46340: Flags [.], ack 14481, win 78, options [nop,nop,TS val 1702909273 ecr 2271528723], length 0
14:30:33.651552 IP 192.168.7.2.5001 > 192.168.1.1.46338: Flags [F.], seq 1, ack 2, win 85, options [nop,nop,TS val 1702909385 ecr 2271528716], length 0
14:30:33.660368 IP 192.168.1.1.46340 > 192.168.7.2.5001: Flags [P.], seq 14481:24617, ack 1, win 83, options [nop,nop,TS val 2271528723 ecr 1702909175], length 10136
14:30:33.660382 IP 192.168.7.2.5001 > 192.168.1.1.46340: Flags [.], ack 24617, win 75, options [nop,nop,TS val 1702909394 ecr 2271528723], length 0
14:30:33.829866 IP 192.168.1.1.46340 > 192.168.7.2.5001: Flags [P.], seq 24617:28961, ack 1, win 83, options [nop,nop,TS val 2271528723 ecr 1702909175], length 4344
14:30:33.829882 IP 192.168.7.2.5001 > 192.168.1.1.46340: Flags [.], ack 28961, win 80, options [nop,nop,TS val 1702909564 ecr 2271528723], length 0
14:30:33.860274 IP 192.168.7.2.5001 > 192.168.1.1.46338: Flags [F.], seq 1, ack 2, win 85, options [nop,nop,TS val 1702909594 ecr 2271528716], length 0
14:30:33.903083 IP 192.168.1.1.46338 > 192.168.7.2.5001: Flags [.], ack 2, win 83, options [nop,nop,TS val 2271528726 ecr 1702909178], length 0
14:30:33.903586 IP 192.168.1.1.46340 > 192.168.7.2.5001: Flags [.], seq 28961:30409, ack 1, win 83, options [nop,nop,TS val 2271528737 ecr 1702909175], length 1448
14:30:33.903596 IP 192.168.7.2.5001 > 192.168.1.1.46340: Flags [.], ack 30409, win 83, options [nop,nop,TS val 1702909637 ecr 2271528737], length 0
14:30:33.927823 IP 192.168.1.1.46340 > 192.168.7.2.5001: Flags [P.], seq 30409:34753, ack 1, win 83, options [nop,nop,TS val 2271528918 ecr 1702909273], length 4344
14:30:33.927843 IP 192.168.7.2.5001 > 192.168.1.1.46340: Flags [.], ack 34753, win 80, options [nop,nop,TS val 1702909662 ecr 2271528918], length 0
14:30:34.000496 IP 192.168.1.1.46338 > 192.168.7.2.5001: Flags [.], ack 2, win 83, options [nop,nop,TS val 2271528933 ecr 1702909178], length 0
14:30:34.000518 IP 192.168.7.2.5001 > 192.168.1.1.46338: Flags [R], seq 3430428569, win 0, length 0
14:30:34.001542 IP 192.168.1.1.46340 > 192.168.7.2.5001: Flags [P.], seq 34753:36201, ack 1, win 83, options [nop,nop,TS val 2271528967 ecr 1702909394], length 1448
14:30:34.001551 IP 192.168.7.2.5001 > 192.168.1.1.46340: Flags [.], ack 36201, win 83, options [nop,nop,TS val 1702909735 ecr 2271528967], length 0
14:30:34.025774 IP 192.168.1.1.46340 > 192.168.7.2.5001: Flags [P.], seq 36201:39097, ack 1, win 83, options [nop,nop,TS val 2271528967 ecr 1702909394], length 2896
14:30:34.025782 IP 192.168.7.2.5001 > 192.168.1.1.46340: Flags [.], ack 39097, win 82, options [nop,nop,TS val 1702909760 ecr 2271528967], length 0
14:30:34.074419 IP 192.168.1.1.46340 > 192.168.7.2.5001: Flags [P.], seq 39097:41993, ack 1, win 83, options [nop,nop,TS val 2271528967 ecr 1702909394], length 2896
14:30:34.074432 IP 192.168.7.2.5001 > 192.168.1.1.46340: Flags [.], ack 41993, win 82, options [nop,nop,TS val 1702909808 ecr 2271528967], length 0
14:30:34.123102 IP 192.168.1.1.46340 > 192.168.7.2.5001: Flags [P.], seq 41993:44889, ack 1, win 83, options [nop,nop,TS val 2271529040 ecr 1702909394], length 2896
14:30:34.123116 IP 192.168.7.2.5001 > 192.168.1.1.46340: Flags [.], ack 44889, win 82, options [nop,nop,TS val 1702909857 ecr 2271529040], length 0
14:30:34.171773 IP 192.168.1.1.46340 > 192.168.7.2.5001: Flags [P.], seq 44889:47785, ack 1, win 83, options [nop,nop,TS val 2271529040 ecr 1702909394], length 2896
14:30:34.171794 IP 192.168.7.2.5001 > 192.168.1.1.46340: Flags [.], ack 47785, win 82, options [nop,nop,TS val 1702909906 ecr 2271529040], length 0
14:30:34.219571 IP 192.168.1.1.46340 > 192.168.7.2.5001: Flags [.], seq 24617:26065, ack 1, win 83, options [nop,nop,TS val 2271529053 ecr 1702909394], length 1448
14:30:34.219587 IP 192.168.7.2.5001 > 192.168.1.1.46340: Flags [.], ack 47785, win 83, options [nop,nop,TS val 1702909953 ecr 2271529040,nop,nop,sack 1 {24617:26065}], length 0
14:30:34.243795 IP 192.168.1.1.46340 > 192.168.7.2.5001: Flags [P.], seq 47785:50681, ack 1, win 83, options [nop,nop,TS val 2271529112 ecr 1702909564], length 2896
14:30:34.243811 IP 192.168.7.2.5001 > 192.168.1.1.46340: Flags [.], ack 50681, win 82, options [nop,nop,TS val 1702909978 ecr 2271529112], length 0
14:30:34.292415 IP 192.168.1.1.46340 > 192.168.7.2.5001: Flags [P.], seq 50681:53577, ack 1, win 83, options [nop,nop,TS val 2271529112 ecr 1702909564], length 2896
14:30:34.292433 IP 192.168.7.2.5001 > 192.168.1.1.46340: Flags [.], ack 53577, win 82, options [nop,nop,TS val 1702910026 ecr 2271529112], length 0
```

D. CLO 4

Pada CLO ini terdapat spesifikasi pengerjaan dan kriteria penilaian yang akan dilakukan.
Goal :

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

1. Nilai Buffer 20

```
time.sleep(1)
print("\n*** Bandwidth test")
time.sleep(1)

#iperf
hB.cmd('iperf -s &')
time.sleep(1)

hA.cmdPrint('iperf -t 60 -c 192.168.6.2 &')

#untuk tes percobaan transfer data buffer 20, 40, 60, 100
r1.cmdPrint("tc qdisc del dev r1-eth0 root")
r1.cmdPrint("tc qdisc add dev r1-eth0 root netem delay 20ms")
```

```
mininet> hA ping hB -c 5
PING 192.168.7.2 (192.168.7.2) 56(84) bytes of data.
64 bytes from 192.168.7.2: icmp_seq=1 ttl=62 time=20.8 ms
64 bytes from 192.168.7.2: icmp_seq=2 ttl=62 time=21.0 ms
64 bytes from 192.168.7.2: icmp_seq=3 ttl=62 time=20.2 ms
64 bytes from 192.168.7.2: icmp_seq=4 ttl=62 time=20.1 ms
64 bytes from 192.168.7.2: icmp_seq=5 ttl=62 time=20.8 ms

--- 192.168.7.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4007ms
rtt min/avg/max/mdev = 20.082/20.567/20.972/0.362 ms
mininet> iperf hA hB
*** Iperf: testing TCP bandwidth between hA and hB
.*** Results: ['480 Kbits/sec', '984 Kbits/sec']
```

2. Nilai Buffer 40

```
time.sleep(1)
print("\n*** Bandwidth test")
time.sleep(1)

#iperf
hB.cmd('iperf -s &')
time.sleep(1)

hA.cmdPrint('iperf -t 60 -c 192.168.6.2 &')

#untuk tes percobaan transfer data buffer 20, 40, 60, 100
r1.cmdPrint("tc qdisc del dev r1-eth0 root")
r1.cmdPrint("tc qdisc add dev r1-eth0 root netem delay 40ms")
```

```
mininet> hA ping hB -c 5
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-40.2 sec  4.75 MBytes   992 Kbits/sec
PING 192.168.7.2 (192.168.7.2) 56(84) bytes of data.
64 bytes from 192.168.7.2: icmp_seq=1 ttl=62 time=40.2 ms
64 bytes from 192.168.7.2: icmp_seq=2 ttl=62 time=40.1 ms
64 bytes from 192.168.7.2: icmp_seq=3 ttl=62 time=40.9 ms
64 bytes from 192.168.7.2: icmp_seq=4 ttl=62 time=40.5 ms
64 bytes from 192.168.7.2: icmp_seq=5 ttl=62 time=40.1 ms

--- 192.168.7.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/mdev = 40.085/40.345/40.883/0.299 ms
mininet> iperf hA hB
*** Iperf: testing TCP bandwidth between hA and hB
.*** Results: ['480 Kbits/sec', '1.00 Mbits/sec']
```

3. Nilai Buffer 60

```
time.sleep(1)
print("\n*** Bandwidth test")
time.sleep(1)

#iperf
hB.cmd('iperf -s &')
time.sleep(1)

hA.cmdPrint('iperf -t 60 -c 192.168.6.2 &')

#untuk tes percobaan transfer data buffer 20, 40, 60, 100
r1.cmdPrint("tc qdisc del dev r1-eth0 root")
r1.cmdPrint("tc qdisc add dev r1-eth0 root netem delay 60ms")
```

```
mininet> hA ping hB -c 5
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-44.5 sec  5.25 MBytes   990 Kbits/sec
PING 192.168.7.2 (192.168.7.2) 56(84) bytes of data.
64 bytes from 192.168.7.2: icmp_seq=1 ttl=62 time=60.5 ms
64 bytes from 192.168.7.2: icmp_seq=2 ttl=62 time=60.2 ms
64 bytes from 192.168.7.2: icmp_seq=3 ttl=62 time=60.6 ms
64 bytes from 192.168.7.2: icmp_seq=4 ttl=62 time=60.1 ms
64 bytes from 192.168.7.2: icmp_seq=5 ttl=62 time=60.5 ms

--- 192.168.7.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4007ms
rtt min/avg/max/mdev = 60.094/60.368/60.617/0.204 ms
mininet> iperf hA hB
*** Iperf: testing TCP bandwidth between hA and hB
.*** Results: ['480 Kbits/sec', '986 Kbits/sec']
```

4. Nilai Buffer 100

```
time.sleep(1)
print("\n*** Bandwidth test")
time.sleep(1)

#iperf
hB.cmd('iperf -s &')
time.sleep(1)

hA.cmdPrint('iperf -t 60 -c 192.168.6.2 &')

#untuk tes percobaan transfer data buffer 20, 40, 60, 100
r1.cmdPrint("tc qdisc del dev r1-eth0 root")
r1.cmdPrint("tc qdisc add dev r1-eth0 root netem delay 100ms")
```

```
mininet> hA ping hB -c 5
PING 192.168.7.2 (192.168.7.2) 56(84) bytes of data.
64 bytes from 192.168.7.2: icmp_seq=1 ttl=62 time=101 ms
64 bytes from 192.168.7.2: icmp_seq=2 ttl=62 time=101 ms
64 bytes from 192.168.7.2: icmp_seq=3 ttl=62 time=100 ms
64 bytes from 192.168.7.2: icmp_seq=4 ttl=62 time=101 ms
64 bytes from 192.168.7.2: icmp_seq=5 ttl=62 time=101 ms

--- 192.168.7.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4007ms
rtt min/avg/max/mdev = 100.100/100.615/100.867/0.268 ms
mininet> iperf hA hB
*** Iperf: testing TCP bandwidth between hA and hB
.*** Results: ['481 Kbits/sec', '1.04 Mbits/sec']
```

5. Hasil Analisis

Berikut adalah hasil statistik tes *network* dari hA ke hB dengan nilai *buffer* (maksimal *queue*) 20 ,40, 60 dan 100.

BUFFER	Packet Transmitted	Packet Received	Packet Loss	TIME
20	5	5	0%	4007ms
40	5	5	0%	4005ms
60	5	5	0%	4007ms
100	5	5	0%	4007ms

Jika dilihat sesuai hasil statistik diatas, tidak ada perbedaan yang signifikan (ambigu) dari masing - masing *buffer*. Dari hasil tes iPerf juga masih belum pasti pengaruh dari buffer terhadap tes iPerfnya. Namun jika dilihat dari tes ping antar hostnya, menunjukan bahwa seiring bertambahnya *buffer*, maka akan terjadi perlambatan (*time*).

PENUTUP

A. Daftar Pustaka

MODUL JARINGAN KOMPUTER IF REGULER, Telkom University.

"Types of Network Topology." *GeeksforGeeks*, 16 June 2022, www.geeksforgeeks.org/types-of-network-topology.

"Introduction of a Router." *GeeksforGeeks*, 9 Nov. 2021, www.geeksforgeeks.org/introduction-of-a-router.

"What Is Transmission Control Protocol (TCP)?" *GeeksforGeeks*, 29 Nov. 2021, www.geeksforgeeks.org/what-is-transmission-control-protocol-tcp.

B. Lampiran

Dokumen Laporan lengkap :
https://docs.google.com/document/d/1p8SloftaIJoS-TludDQzcuareXqw-v_CJJXxy9hKDoU/edit?usp=sharing

Link projek Github :
https://github.com/berlianm/Simulasi-Mininet---Jaringan-Komputer

Video Demo Program :
https://youtu.be/pACEBwOdKJA