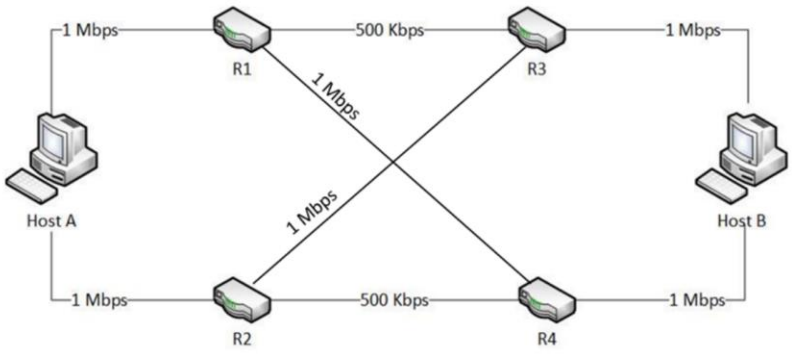
LAPORAN TUGAS BESAR
JARINGAN KOMPUTER

Oleh:
Gisella Vania Dwipayana
1301204469
IF-44-02

Program Studi S1 Informatika
Fakultas Informatika
2022

A. Deskripsi Tugas

Pada Tugas Besar Jaringan Komputer ini akan dilakukan simulasi pada mininet dengan menggunakan topologi sebagai berikut:



Gambar 12.1 Topologi untuk tugas besar

Adapun proses pengerjaan tugas besar ini dibagi menjadi beberapa bagian, meliputi:

1. CLO 1
   Pada CLO ini terdapat spesifikasi pengerjaan sebagai berikut:
   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.
2. CLO 2
   Pada CLO ini terdapat spesifikasi pengerjaan sebagai berikut:
   Goal : Mengimplementasikan mekanisme Routing pada topologi yang ada.
   - Uji konektivitas menggunakan ping.
   - Membuat table routing di semua host, dibuktikan dengan ping antar host.
   - Menganalisis routing yang digunakan menggunakan traceroute.
3. CLO 3
   Pada CLO ini terdapat spesifikasi pengerjaan sebagai berikut:
   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.
4. CLO 4
   Pada CLO ini terdapat spesifikasi pengerjaan sebagai berikut:
   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.

B. CLO 1

1. Subnetting

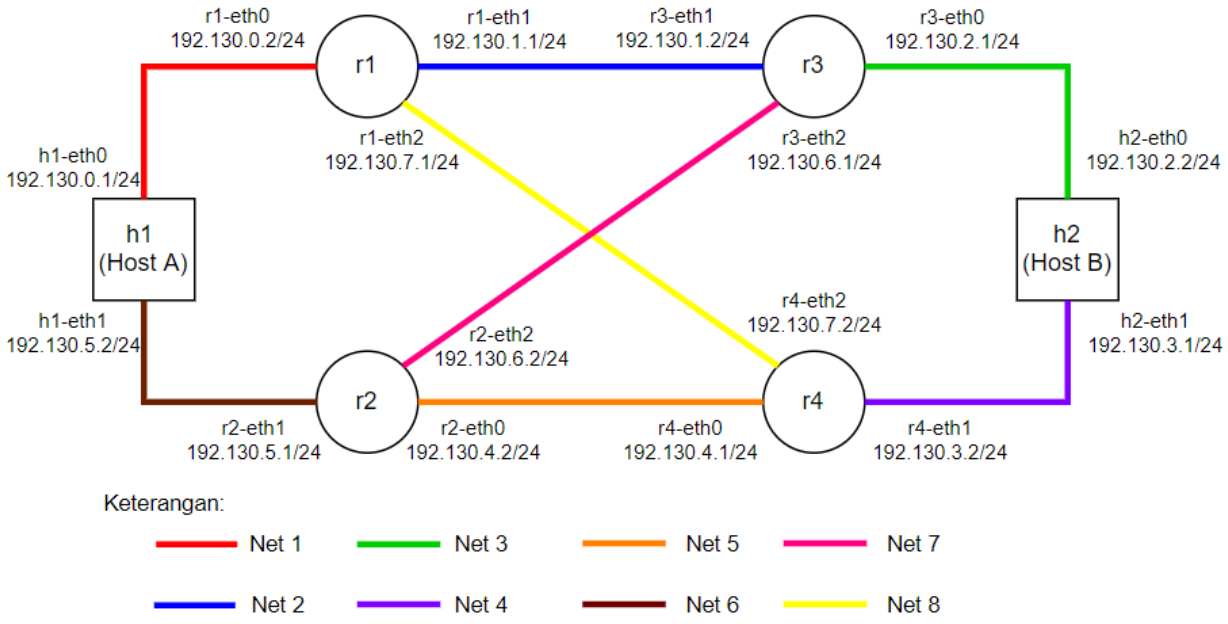
Subnetting adalah proses membagi atau memecah jaringan menjadi beberapa jaringan yang lebih kecil (subnetwork) dimana setiap subnetwork memliki alamat subnetnya masing-masing.

Adapun tabel subnetting yang saya gunakan pada tugas besar ini sebagai berikut:

IP AWAL 192.130.0.0

Subnet Name	Needed Size	Allocated Size	Network Address	Host Range	Broadcast	Prefix	Subnet Mask
Net 1	2	256	192.130.0.0	192.130.0.1 - 192.130.0.254	192.130.0.255	/24	255.255.255.0
Net 2	2	256	192.130.1.0	192.130.1.1 - 192.130.1.254	192.130.1.255	/24	255.255.255.0
Net 3	2	256	192.130.2.0	192.130.2.1 - 192.130.2.254	192.130.2.255	/24	255.255.255.0
Net 4	2	256	192.130.3.0	192.130.3.1 - 192.130.3.254	192.130.3.255	/24	255.255.255.0
Net 5	2	256	192.130.4.0	192.130.4.1 - 192.130.4.254	192.130.4.255	/24	255.255.255.0
Net 6	2	256	192.130.5.0	192.130.5.1 - 192.130.5.254	192.130.5.255	/24	255.255.255.0
Net 7	2	256	192.130.6.0	192.130.6.1 - 192.130.6.254	192.130.6.255	/24	255.255.255.0
Net 8	2	256	192.130.7.0	192.130.7.1 - 192.130.7.254	192.130.7.255	/24	255.255.255.0

Kemudian dilakukan implementasi tabel subnetting diatas pada topologi yang tertera pada soal, sebagai berikut:



2. Build Topology pada Mininet

Berikut merupakan source code yang digunakan untuk build dan run topologi pada mininet:

clo1.py

```python
# 1301204469 - Gisella Vania D
from mininet.net import Mininet
from mininet.log import setLogLevel
from mininet.cli import CLI
from mininet.link import TCLink

def runTopo():
        net = Mininet()

        # add host
        h1 = net.addHost('h1')
        h2 = net.addHost('h2')
        r1 = net.addHost('r1')
        r2 = net.addHost('r2')
        r3 = net.addHost('r3')
        r4 = net.addHost('r4')

        # add link
        net.addLink(h1, r1, intfName1='h1-eth0', intfName2='r1-eth0', cls=TCLink, bw=1)
        net.addLink(r1, r3, intfName1='r1-eth1', intfName2='r3-eth1', cls=TCLink, bw=0.5)
        net.addLink(h2, r3, intfName1='h2-eth0', intfName2='r3-eth0', cls=TCLink, bw=1)
        net.addLink(h2, r4, intfName1='h2-eth1', intfName2='r4-eth1', cls=TCLink, bw=1)
        net.addLink(r4, r2, intfName1='r4-eth0', intfName2='r2-eth0', cls=TCLink, bw=0.5)
        net.addLink(r2, h1, intfName1='r2-eth1', intfName2='h1-eth1', cls=TCLink, bw=1)
        net.addLink(r3, r2, intfName1='r3-eth2', intfName2='r2-eth2', cls=TCLink, bw=1)
        net.addLink(r1, r4, intfName1='r1-eth2', intfName2='r4-eth2', cls=TCLink, bw=1)

        net.start()

        # config IP
        h1.cmd('ifconfig h1-eth0 192.130.0.1 netmask 255.255.255.0')
        h1.cmd('ifconfig h1-eth1 192.130.5.2 netmask 255.255.255.0')

        h2.cmd('ifconfig h2-eth0 192.130.2.2 netmask 255.255.255.0')
        h2.cmd('ifconfig h2-eth1 192.130.3.1 netmask 255.255.255.0')

        r1.cmd('ifconfig r1-eth0 192.130.0.2 netmask 255.255.255.0')
```

Python Tab Width: 8 Ln 67, Col 9 INS

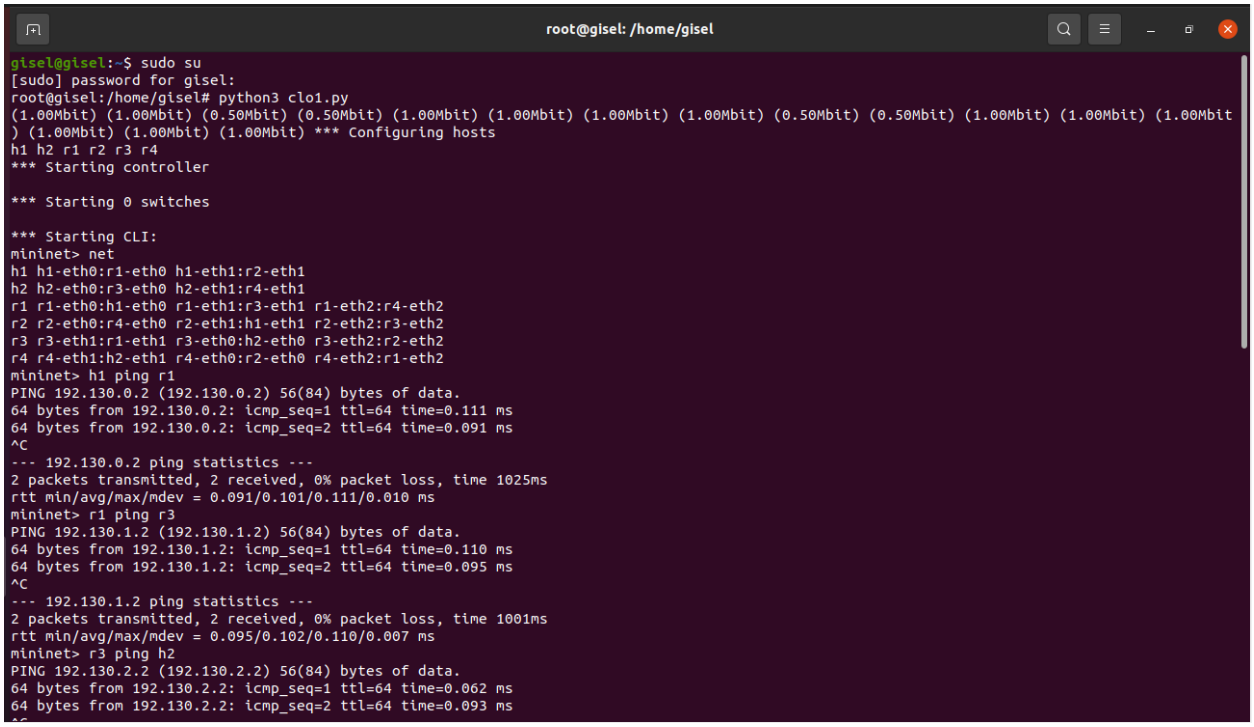
clo1.py

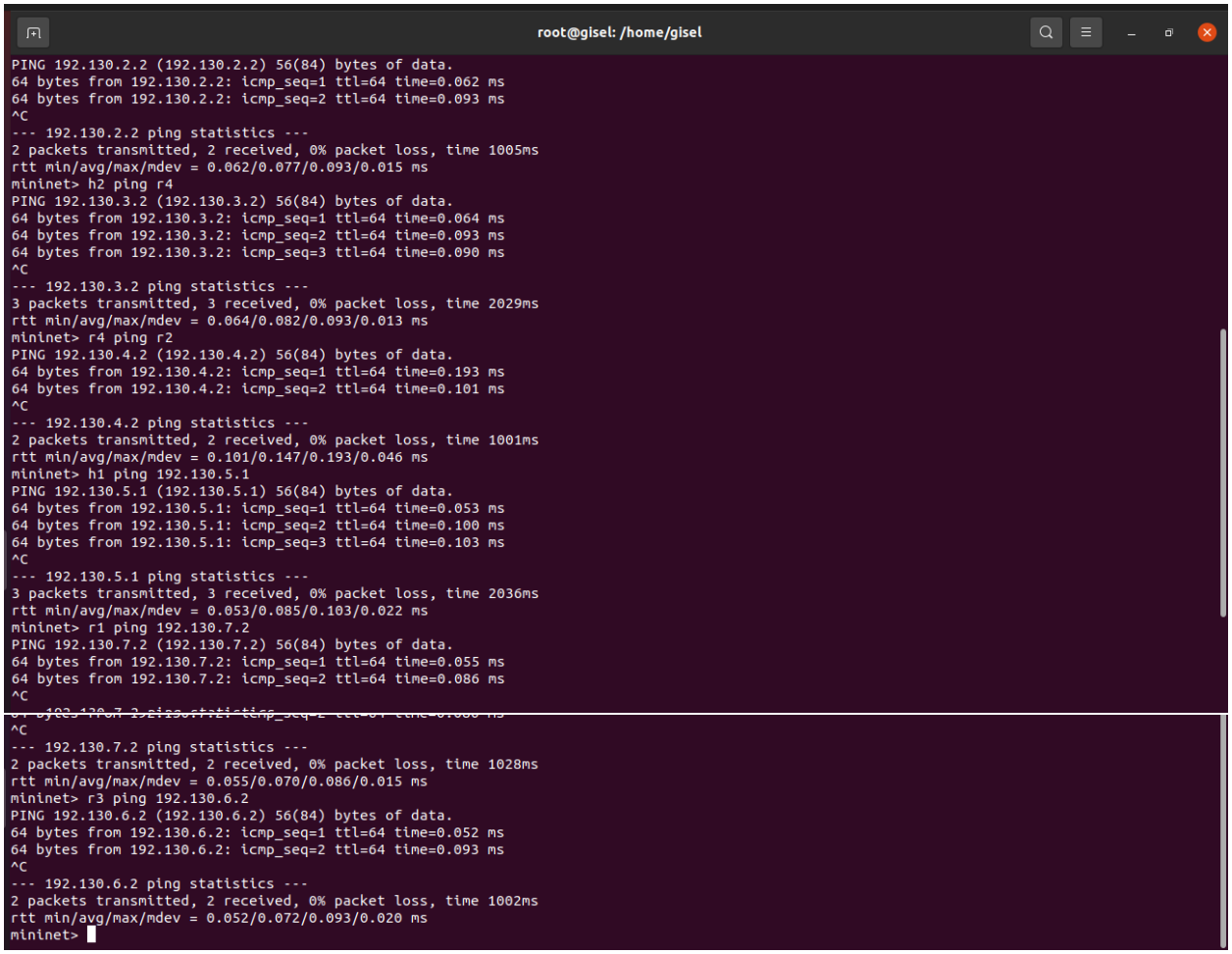
```python
        h1.cmd('ifconfig h1-eth1 192.130.5.2 netmask 255.255.255.0')

        h2.cmd('ifconfig h2-eth0 192.130.2.2 netmask 255.255.255.0')
        h2.cmd('ifconfig h2-eth1 192.130.3.1 netmask 255.255.255.0')

        r1.cmd('ifconfig r1-eth0 192.130.0.2 netmask 255.255.255.0')
        r1.cmd('ifconfig r1-eth1 192.130.1.1 netmask 255.255.255.0')
        r1.cmd('ifconfig r1-eth2 192.130.7.1 netmask 255.255.255.0')

        r2.cmd('ifconfig r2-eth0 192.130.4.2 netmask 255.255.255.0')
        r2.cmd('ifconfig r2-eth1 192.130.5.1 netmask 255.255.255.0')
        r2.cmd('ifconfig r2-eth2 192.130.6.2 netmask 255.255.255.0')

        r3.cmd('ifconfig r3-eth0 192.130.2.1 netmask 255.255.255.0')
        r3.cmd('ifconfig r3-eth1 192.130.1.2 netmask 255.255.255.0')
        r3.cmd('ifconfig r3-eth2 192.130.6.1 netmask 255.255.255.0')

        r4.cmd('ifconfig r4-eth0 192.130.4.1 netmask 255.255.255.0')
        r4.cmd('ifconfig r4-eth1 192.130.3.2 netmask 255.255.255.0')
        r4.cmd('ifconfig r4-eth2 192.130.7.2 netmask 255.255.255.0')

        # config router
        r1.cmd('sysctl net.ipv4.ip_forward=1')
        r2.cmd('sysctl net.ipv4.ip_forward=1')
        r3.cmd('sysctl net.ipv4.ip_forward=1')
        r4.cmd('sysctl net.ipv4.ip_forward=1')

        CLI(net)
        net.stop()

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

Python Tab Width: 8 Ln 67, Col 9 INS

Berikut ini merupakan koneksi antar device yang terbentuk:

3. Uji Konektivitas antar host yang berada pada subnet yang sama
   Berikut ini merupakan hasil uji konektivitas antar host yang berada pada subnet yang sama:





Dapat terlihat bahwa tidak terdapat packet loss dari hasil uji konektivitas antar host yang berada pada subnet yang sama dan data dapat terkirim dengan sempurna.

C. CLO 2

Pada bagian ini, dilanjutkan hasil pengerjaan CLO 1 dengan melakukan routing. Hal ini dilakukan agar setiap host atau node yang berada pada subnet berbeda dapat terhubung.

1. Routing

   Berikut merupakan source code yang digunakan untuk konfigurasi routing di semua host dan router pada mininet termasuk membuat tabel routing pada semua host:

```
clo2.py
Open  Save

65        # static routing
66        h1.cmd('ip rule add from 192.130.0.1 table 1')
67        h1.cmd('ip rule add from 192.130.5.2 table 2')
68        h1.cmd('ip route add 192.130.0.0/24 dev h1-eth0 scope link table 1')
69        h1.cmd('ip route add default via 192.130.0.2 dev h1-eth0 table 1')
70        h1.cmd('ip route add 192.130.5.0/24 dev h1-eth1 scope link table 2')
71        h1.cmd('ip route add default via 192.130.5.1 dev h1-eth1 table 2')
72        h1.cmd('ip route add default scope global nexthop via 192.130.0.2 dev h1-eth0')
73        h1.cmd('ip route add default scope global nexthop via 192.130.5.1 dev h1-eth1')
74
75        h2.cmd('ip rule add from 192.130.2.2 table 1')
76        h2.cmd('ip rule add from 192.130.3.1 table 2')
77        h2.cmd('ip route add 192.130.2.0/24 dev h2-eth0 scope link table 1')
78        h2.cmd('ip route add default via 192.130.2.1 dev h2-eth0 table 1')
79        h2.cmd('ip route add 192.130.3.0/24 dev h2-eth1 scope link table 2')
80        h2.cmd('ip route add default via 192.130.3.2 dev h2-eth1 table 2')
81        h2.cmd('ip route add default scope global nexthop via 192.130.2.1 dev h2-eth0')
82        h2.cmd('ip route add default scope global nexthop via 192.130.3.2 dev h2-eth1')
83
84        r1.cmd('route add -net 192.130.2.0/24 gw 192.130.1.2')
85        r1.cmd('route add -net 192.130.3.0/24 gw 192.130.7.2')
86        r1.cmd('route add -net 192.130.4.0/24 gw 192.130.7.2')
87        r1.cmd('route add -net 192.130.5.0/24 gw 192.130.1.2')
88        r1.cmd('route add -net 192.130.5.0/24 gw 192.130.7.2')
89        r1.cmd('route add -net 192.130.6.0/24 gw 192.130.1.2')
90
91        r2.cmd('route add -net 192.130.0.0/24 gw 192.130.4.1')
92        r2.cmd('route add -net 192.130.0.0/24 gw 192.130.6.1')
93        r2.cmd('route add -net 192.130.1.0/24 gw 192.130.6.1')
94        r2.cmd('route add -net 192.130.2.0/24 gw 192.130.6.1')
95        r2.cmd('route add -net 192.130.3.0/24 gw 192.130.4.1')
96        r2.cmd('route add -net 192.130.7.0/24 gw 192.130.4.1')
97
98        r3.cmd('route add -net 192.130.0.0/24 gw 192.130.1.1')
99        r3.cmd('route add -net 192.130.3.0/24 gw 192.130.6.2')
100       r3.cmd('route add -net 192.130.3.0/24 gw 192.130.1.1')
101       r3.cmd('route add -net 192.130.4.0/24 gw 192.130.6.2')

Python  Tab Width: 8  Ln 23, Col 50  INS

100       r3.cmd('route add -net 192.130.3.0/24 gw 192.130.1.1')
101       r3.cmd('route add -net 192.130.4.0/24 gw 192.130.6.2')
102       r3.cmd('route add -net 192.130.5.0/24 gw 192.130.6.2')
103       r3.cmd('route add -net 192.130.7.0/24 gw 192.130.1.1')
104
105       r4.cmd('route add -net 192.130.0.0/24 gw 192.130.7.1')
106       r4.cmd('route add -net 192.130.1.0/24 gw 192.130.7.1')
107       r4.cmd('route add -net 192.130.2.0/24 gw 192.130.4.2')
108       r4.cmd('route add -net 192.130.2.0/24 gw 192.130.7.1')
109       r4.cmd('route add -net 192.130.5.0/24 gw 192.130.4.2')
110       r4.cmd('route add -net 192.130.6.0/24 gw 192.130.4.2')
```

2. Uji Konektivitas

Setelah dilakukan konfigurasi routing pada setiap host dan router, dilakukan uji konektivitas untuk membuktikan apakah semua host dan router sudah saling terhubung. Berikut ini merupakan hasil uji konektivitas antar host menggunakan ping:

- Dari h1 (Host A) ke seluruh node

```
root@gisel: /home/gisel

gisel@gisel:~$ sudo su
[sudo] password for gisel:
root@gisel:/home/gisel# python3 clo2.py
(1.00Mbit) (1.00Mbit) (0.50Mbit) (0.50Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1
.00Mbit) (0.50Mbit) (0.50Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.0
0Mbit) (1.00Mbit) *** Configuring hosts
h1 h2 r1 r2 r3 r4
*** Starting controller

*** Starting 0 switches

*** Starting CLI:
mininet> h1 ping r1
PING 192.130.0.2 (192.130.0.2) 56(84) bytes of data.
64 bytes from 192.130.0.2: icmp_seq=1 ttl=64 time=0.064 ms
64 bytes from 192.130.0.2: icmp_seq=2 ttl=64 time=0.074 ms
^C
--- 192.130.0.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1025ms
rtt min/avg/max/mdev = 0.064/0.069/0.074/0.005 ms
mininet> h1 ping r2
PING 192.130.4.2 (192.130.4.2) 56(84) bytes of data.
64 bytes from 192.130.4.2: icmp_seq=1 ttl=62 time=0.127 ms
64 bytes from 192.130.4.2: icmp_seq=2 ttl=62 time=0.190 ms
^C
--- 192.130.4.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1006ms
rtt min/avg/max/mdev = 0.127/0.158/0.190/0.031 ms
mininet> h1 ping r3
PING 192.130.1.2 (192.130.1.2) 56(84) bytes of data.
64 bytes from 192.130.1.2: icmp_seq=1 ttl=63 time=0.046 ms
64 bytes from 192.130.1.2: icmp_seq=2 ttl=63 time=0.081 ms
^C
--- 192.130.1.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1011ms
rtt min/avg/max/mdev = 0.046/0.063/0.081/0.017 ms
```

```
mininet> h1 ping r4
PING 192.130.3.2 (192.130.3.2) 56(84) bytes of data.
64 bytes from 192.130.3.2: icmp_seq=1 ttl=63 time=0.050 ms
64 bytes from 192.130.3.2: icmp_seq=2 ttl=63 time=0.103 ms
^C
--- 192.130.3.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1004ms
rtt min/avg/max/mdev = 0.050/0.076/0.103/0.026 ms
mininet> h1 ping h2
PING 192.130.2.2 (192.130.2.2) 56(84) bytes of data.
64 bytes from 192.130.2.2: icmp_seq=1 ttl=62 time=0.080 ms
64 bytes from 192.130.2.2: icmp_seq=2 ttl=62 time=0.099 ms
^C
--- 192.130.2.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1023ms
rtt min/avg/max/mdev = 0.080/0.089/0.099/0.009 ms
mininet>
```

- Dari h2 (Host B) ke seluruh node

```
root@gisel: /home/gisel
gisel@gisel:~$ sudo su
[sudo] password for gisel:
root@gisel:/home/gisel# python3 clo2.py
(1.00Mbit) (1.00Mbit) (0.50Mbit) (0.50Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1
.00Mbit) (0.50Mbit) (0.50Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.0
0Mbit) (1.00Mbit) *** Configuring hosts
h1 h2 r1 r2 r3 r4
*** Starting controller

*** Starting 0 switches

*** Starting CLI:
mininet> h2 ping r1
PING 192.130.0.2 (192.130.0.2) 56(84) bytes of data.
64 bytes from 192.130.0.2: icmp_seq=1 ttl=63 time=0.100 ms
64 bytes from 192.130.0.2: icmp_seq=2 ttl=63 time=0.067 ms
^C
--- 192.130.0.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1029ms
rtt min/avg/max/mdev = 0.067/0.083/0.100/0.016 ms
mininet> h2 ping r2
PING 192.130.4.2 (192.130.4.2) 56(84) bytes of data.
64 bytes from 192.130.4.2: icmp_seq=1 ttl=63 time=0.069 ms
64 bytes from 192.130.4.2: icmp_seq=2 ttl=63 time=0.091 ms
^C
--- 192.130.4.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1022ms
rtt min/avg/max/mdev = 0.069/0.080/0.091/0.011 ms
mininet> h2 ping r3
PING 192.130.1.2 (192.130.1.2) 56(84) bytes of data.
64 bytes from 192.130.1.2: icmp_seq=1 ttl=64 time=0.044 ms
64 bytes from 192.130.1.2: icmp_seq=2 ttl=64 time=0.089 ms
^C
--- 192.130.1.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1006ms
rtt min/avg/max/mdev = 0.044/0.066/0.089/0.022 ms
mininet> h2 ping r4
```

```
mininet> h2 ping r4
PING 192.130.3.2 (192.130.3.2) 56(84) bytes of data.
64 bytes from 192.130.3.2: icmp_seq=1 ttl=64 time=0.055 ms
64 bytes from 192.130.3.2: icmp_seq=2 ttl=64 time=0.094 ms
^C
--- 192.130.3.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1025ms
rtt min/avg/max/mdev = 0.055/0.074/0.094/0.019 ms
mininet> h2 ping h1
PING 192.130.0.1 (192.130.0.1) 56(84) bytes of data.
64 bytes from 192.130.0.1: icmp_seq=1 ttl=62 time=0.168 ms
64 bytes from 192.130.0.1: icmp_seq=2 ttl=62 time=0.129 ms
^C
--- 192.130.0.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1014ms
rtt min/avg/max/mdev = 0.129/0.148/0.168/0.019 ms
mininet>
```

- Melakukan ping dari dan ke seluruh host maupun router

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 r1 r2 r3 r4
h2 -> h1 r1 r2 r3 r4
r1 -> h1 h2 r2 r3 r4
r2 -> h1 h2 r1 r3 r4
r3 -> h1 h2 r1 r2 r4
r4 -> h1 h2 r1 r2 r3
*** Results: 0% dropped (30/30 received)
mininet>
```

Dari hasil uji konektivitas dapat terlihat bahwa tidak terdapat packet loss ataupun packet yang didrop maka seluruh host dan router telah terhubung.

3. Menganalisis routing yang digunakan menggunakan traceroute

Traceroute adalah salah satu cara pemecahan masalah jaringan dengan melacak rute yang dilewati paket untuk mencapai tujuan kemudian menampilkannya, sehingga pengguna dapat memecahkan masalah jaringan dengan melihat perangkat mana yang menyebabkan masalah yang terjadi. Berikut merupakan penggunaan traceroute dan analisa routing yang digunakan:

- Traceroute dari host A menuju ke seluruh node

```
mininet> h1 traceroute r1
traceroute to 192.130.0.2 (192.130.0.2), 30 hops max, 60 byte packets
 1  192.130.0.2 (192.130.0.2)  0.266 ms  0.189 ms  0.178 ms
mininet> h1 traceroute r2
traceroute to 192.130.4.2 (192.130.4.2), 30 hops max, 60 byte packets
 1  192.130.0.2 (192.130.0.2)  0.378 ms  0.329 ms  0.316 ms
 2  192.130.7.2 (192.130.7.2)  0.304 ms  0.261 ms  0.245 ms
 3  192.130.4.2 (192.130.4.2)  0.231 ms  0.174 ms  0.156 ms
mininet> h1 traceroute r3
traceroute to 192.130.1.2 (192.130.1.2), 30 hops max, 60 byte packets
 1  192.130.0.2 (192.130.0.2)  0.249 ms  0.205 ms  0.194 ms
 2  192.130.1.2 (192.130.1.2)  0.183 ms  0.161 ms  0.148 ms
mininet> h1 traceroute r4
traceroute to 192.130.3.2 (192.130.3.2), 30 hops max, 60 byte packets
 1  192.130.0.2 (192.130.0.2)  0.311 ms  0.260 ms  0.246 ms
 2  192.130.3.2 (192.130.3.2)  0.233 ms  0.208 ms  0.190 ms
mininet> h1 traceroute h2
traceroute to 192.130.2.2 (192.130.2.2), 30 hops max, 60 byte packets
 1  192.130.0.2 (192.130.0.2)  0.072 ms  0.007 ms  0.006 ms
 2  192.130.1.2 (192.130.1.2)  0.020 ms  0.011 ms  0.010 ms
 3  192.130.2.2 (192.130.2.2)  0.045 ms  0.014 ms  0.014 ms
```

- Traceroute dari host B menuju ke seluruh node

```
mininet> h2 traceroute r1
traceroute to 192.130.0.2 (192.130.0.2), 30 hops max, 60 byte packets
 1  192.130.2.1 (192.130.2.1)  0.041 ms  0.009 ms  0.007 ms
 2  192.130.0.2 (192.130.0.2)  0.023 ms  0.014 ms  0.012 ms
mininet> h2 traceroute r2
traceroute to 192.130.4.2 (192.130.4.2), 30 hops max, 60 byte packets
 1  192.130.2.1 (192.130.2.1)  0.246 ms  0.202 ms  0.190 ms
 2  192.130.4.2 (192.130.4.2)  0.180 ms  0.161 ms  0.145 ms
mininet> h2 traceroute r3
traceroute to 192.130.1.2 (192.130.1.2), 30 hops max, 60 byte packets
 1  192.130.1.2 (192.130.1.2)  0.310 ms  0.258 ms  0.243 ms
mininet> h2 traceroute r4
traceroute to 192.130.3.2 (192.130.3.2), 30 hops max, 60 byte packets
 1  192.130.3.2 (192.130.3.2)  0.206 ms  0.167 ms  0.156 ms
mininet> h2 traceroute h1
traceroute to 192.130.0.1 (192.130.0.1), 30 hops max, 60 byte packets
 1  192.130.2.1 (192.130.2.1)  0.396 ms  0.344 ms  0.330 ms
 2  192.130.1.1 (192.130.1.1)  0.317 ms  0.292 ms  0.274 ms
 3  192.130.0.1 (192.130.0.1)  0.260 ms  0.230 ms  0.210 ms
mininet>
```

- Traceroute r1 ke r2

```
mininet> r1 traceroute r2
traceroute to 192.130.4.2 (192.130.4.2), 30 hops max, 60 byte packets
 1  192.130.7.2 (192.130.7.2)  0.070 ms  0.008 ms  0.007 ms
 2  192.130.4.2 (192.130.4.2)  0.046 ms  0.009 ms  0.008 ms
mininet>
```

- Traceroute r3 ke r4

```
mininet> r3 traceroute r4
traceroute to 192.130.3.2 (192.130.3.2), 30 hops max, 60 byte packets
 1  192.130.1.1 (192.130.1.1)  0.049 ms  0.009 ms  0.008 ms
 2  192.130.3.2 (192.130.3.2)  0.023 ms  0.013 ms  0.013 ms
mininet>
```

Hasil h1 traceroute h2 menunjukkan bahwa untuk menuju ke 192.130.2.2, h1 (192.130.0.1) melalui router dengan IP 192.130.0.2 (r1) dan 192.130.1.2 (r3). Begitu pula untuk membaca hasil traceroute yang lainnya.

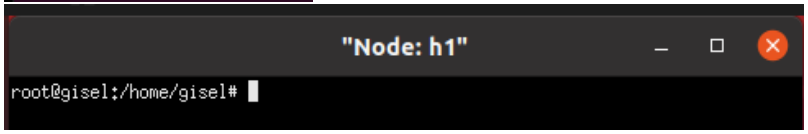
D. CLO 3

TCP (Transmission Control Protocol) adalah salah satu protokol utama dari internet protocol suite. TCP terletak pada transport layer dan digunakan untuk menyediakan layanan pengiriman yang reliable atau andal. TCP adalah connection-oriented protocol untuk komunikasi yang membantu pertukaran pesan antara perangkat yang berbeda melalui jaringan. TCP memiliki fitur-fitur seperti sistem penomoran segment, flow control, error control, congestion control. Selain itu, dalam pembangunan koneksi, TCP menggunakan mekanisme three-way-handshake. Fitur-fitur tersebut juga membedakan antara TCP dan UDP dimana UDP bersifat connection-less, unreliable, tidak terdapat pengurutan data, tidak memiliki flow control, dan tidak memiliki congestion control.

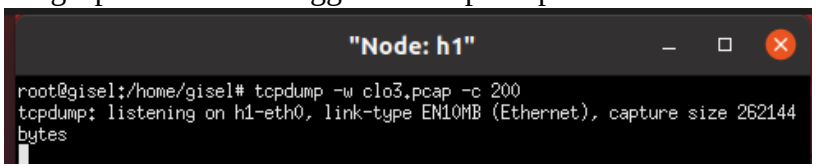
Pada bagian ini akan dibuktikan bahwa TCP telah diimplementasikan dengan benar pada topologi. Hal ini dilakukan generate traffic TCP menggunakan iPerf dan mengcapture traffic menggunakan tcpdump. Adapun langkah-langkah untuk melakukannya sebagai berikut:

1) Melakukan pembukaan terminal untuk node h1 (host A) seperti berikut:

```
*** Starting CLI:
mininet> xterm h1
```



2) Mengcapture traffic menggunakan tcpdump

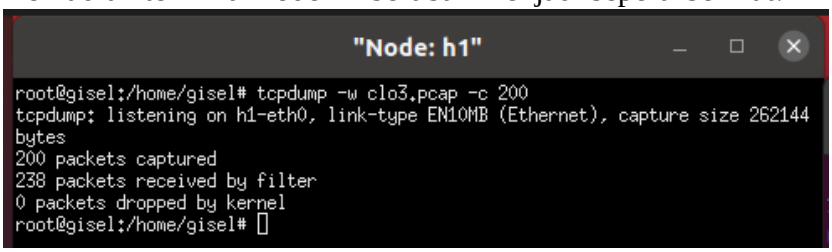


Perintah diatas berarti akan mengcapture paket sejumlah 200 dan menyimpannya dalam bentuk file yaitu pada clo3.pcap.

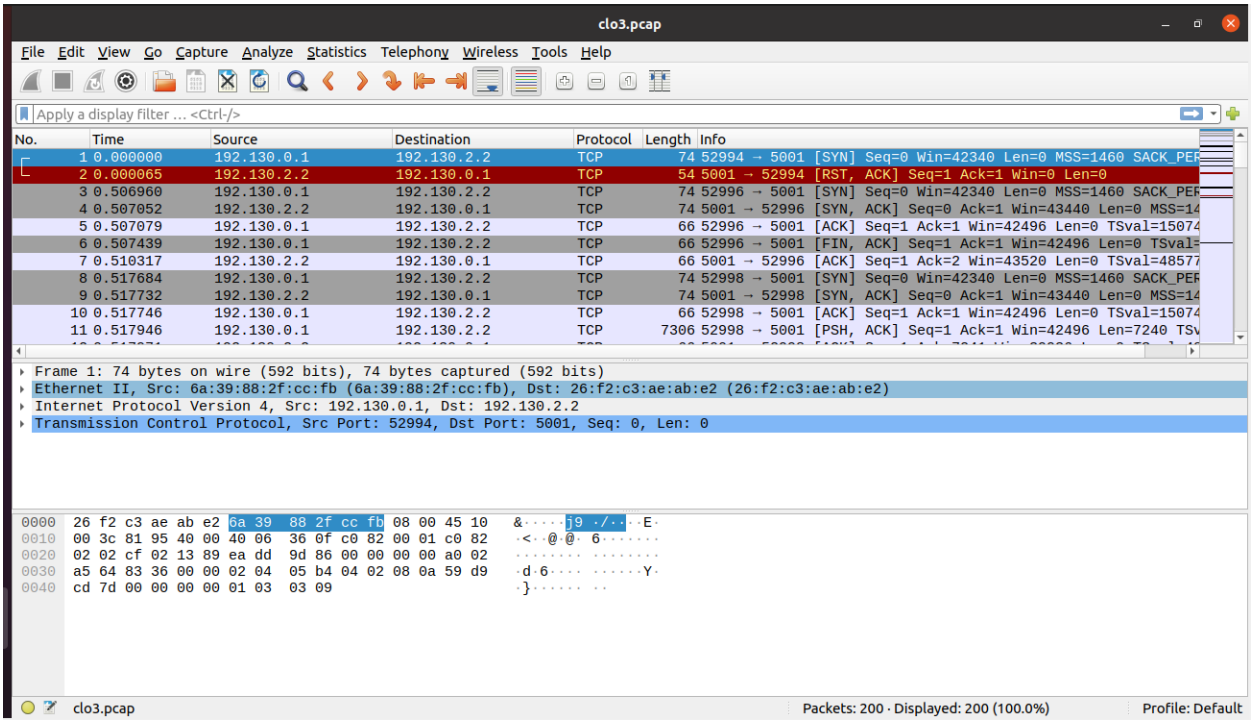
3) Menjalankan iPerf

```
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
.*** Results: ['478 Kbits/sec', '1.16 Mbits/sec']
mininet>
```

Kemudian terminal node h1 berubah menjadi seperti berikut:



Hasil capture traffic yang telah dilakukan sebagai berikut:



Dari hasil tersebut terlihat bahwa protokol yang digunakan adalah TCP. Selain itu, terlihat proses pembangunan koneksi pada TCP menggunakan three-way-handshake pada paket nomor 3, 4, dan 5 dimana

- Paket 3: 192.130.0.1 (h1 atau host A) mengirim message SYN ke 192.130.2.2 (h2 atau host B)
- Paket 4: 192.130.2.2 (h2 atau host B) mengirim message SYN-ACK ke 192.130.0.1 (h1 atau host A)
- Paket 5: 192.130.0.1 (h1 atau host A) mengirim message ACK ke 192.130.2.2 (h2 atau host B)

Dengan demikian, telah terbukti bahwa TCP telah diimplementasikan dengan benar pada topologi.

E. CLO 4

Network scheduler atau packet scheduler adalah program arbiter yang mengelola urutan paket jaringan dalam antrian dikirim dan diterima dari interface jaringan yang merupakan circular data buffer. Packet scheduling bertujuan untuk meningkatkan tingkat keadilan (fairness) dan meningkatkan efisiensi penggunaan bandwidth.

Pada bagian ini akan dilakukan analisa pengaruh ukuran buffer dengan melakukan perubahan ukuran buffer, dimana ukuran buffer yang digunakan adalah 20, 40, 60, dan 100. Pertama-tama dilakukan generate traffic menggunakan iPerf dengan menambahkan source code berikut ini:

```
# menjalankan iPerf di background process
h2.cmd('iperf -s &')
h1.cmd('iperf -t 30 -c 192.130.2.2 &')
```

Kemudian untuk menggunakan queue HTB dan mengeset ukuran buffer, ditambahkan source code “max_queue_size = <ukuran buffer yang ingin digunakan>” dan “use_htb = True” pada setiap link. Adapun source code dan hasil run untuk setiap ukuran buffer (20, 40, 60, dan 100) sebagai berikut:

1. Ukuran buffer = 20

   Source code yang diubah:

```
# add link
net.addLink(h1, r1, max_queue_size=20, use_htb=True, intfName1='h1-eth0', intfName2='r1-eth0', cls=TCLink, bw=1)
net.addLink(r1, r3, max_queue_size=20, use_htb=True, intfName1='r1-eth1', intfName2='r3-eth1', cls=TCLink, bw=0.5)
net.addLink(h2, r3, max_queue_size=20, use_htb=True, intfName1='h2-eth0', intfName2='r3-eth0', cls=TCLink, bw=1)
net.addLink(h2, r4, max_queue_size=20, use_htb=True, intfName1='h2-eth1', intfName2='r4-eth1', cls=TCLink, bw=1)
net.addLink(r4, r2, max_queue_size=20, use_htb=True, intfName1='r4-eth0', intfName2='r2-eth0', cls=TCLink, bw=0.5)
net.addLink(r2, h1, max_queue_size=20, use_htb=True, intfName1='r2-eth1', intfName2='h1-eth1', cls=TCLink, bw=1)
net.addLink(r3, r2, max_queue_size=20, use_htb=True, intfName1='r3-eth2', intfName2='r2-eth2', cls=TCLink, bw=1)
net.addLink(r1, r4, max_queue_size=20, use_htb=True, intfName1='r1-eth2', intfName2='r4-eth2', cls=TCLink, bw=1)
```

   Hasil run:

```
mininet> h1 ping h2

PING 192.130.2.2 (192.130.2.2) 56(84) bytes of data.
64 bytes from 192.130.2.2: icmp_seq=1 ttl=62 time=0.058 ms
64 bytes from 192.130.2.2: icmp_seq=2 ttl=62 time=0.149 ms
64 bytes from 192.130.2.2: icmp_seq=3 ttl=62 time=0.144 ms
64 bytes from 192.130.2.2: icmp_seq=4 ttl=62 time=0.134 ms
^C
--- 192.130.2.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3073ms
rtt min/avg/max/mdev = 0.058/0.121/0.149/0.036 ms
mininet> h1 iperf -i 2 -c 192.130.2.2
------------------------------------------------------------
Client connecting to 192.130.2.2, TCP port 5001
TCP window size:  128 KByte (default)
------------------------------------------------------------
[  3] local 192.130.0.1 port 60224 connected with 192.130.2.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 2.0 sec   544 KBytes  2.23 Mbits/sec
[  3]  2.0- 4.0 sec   106 KBytes   434 Kbits/sec
[  3]  4.0- 6.0 sec   127 KBytes   521 Kbits/sec
[  3]  6.0- 8.0 sec   127 KBytes   521 Kbits/sec
[  3]  8.0-10.0 sec   127 KBytes   521 Kbits/sec
[  3]  0.0-10.6 sec  1.01 MBytes   801 Kbits/sec
mininet>
```

2. Ukuran buffer = 40

   Source code yang diubah:

```
# add link
net.addLink(h1, r1, max_queue_size=40, use_htb=True, intfName1='h1-eth0', intfName2='r1-eth0', cls=TCLink, bw=1)
net.addLink(r1, r3, max_queue_size=40, use_htb=True, intfName1='r1-eth1', intfName2='r3-eth1', cls=TCLink, bw=0.5)
net.addLink(h2, r3, max_queue_size=40, use_htb=True, intfName1='h2-eth0', intfName2='r3-eth0', cls=TCLink, bw=1)
net.addLink(h2, r4, max_queue_size=40, use_htb=True, intfName1='h2-eth1', intfName2='r4-eth1', cls=TCLink, bw=1)
net.addLink(r4, r2, max_queue_size=40, use_htb=True, intfName1='r4-eth0', intfName2='r2-eth0', cls=TCLink, bw=0.5)
net.addLink(r2, h1, max_queue_size=40, use_htb=True, intfName1='r2-eth1', intfName2='h1-eth1', cls=TCLink, bw=1)
net.addLink(r3, r2, max_queue_size=40, use_htb=True, intfName1='r3-eth2', intfName2='r2-eth2', cls=TCLink, bw=1)
net.addLink(r1, r4, max_queue_size=40, use_htb=True, intfName1='r1-eth2', intfName2='r4-eth2', cls=TCLink, bw=1)
```

Hasil run:

```
mininet> h1 ping h2

PING 192.130.2.2 (192.130.2.2) 56(84) bytes of data.
64 bytes from 192.130.2.2: icmp_seq=1 ttl=62 time=0.060 ms
64 bytes from 192.130.2.2: icmp_seq=2 ttl=62 time=0.149 ms
64 bytes from 192.130.2.2: icmp_seq=3 ttl=62 time=0.160 ms
64 bytes from 192.130.2.2: icmp_seq=4 ttl=62 time=0.113 ms
^C
--- 192.130.2.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3074ms
rtt min/avg/max/mdev = 0.060/0.120/0.160/0.039 ms
mininet> h1 iperf -i 2 -c 192.130.2.2
------------------------------------------------------------
Client connecting to 192.130.2.2, TCP port 5001
TCP window size:  128 KByte (default)
------------------------------------------------------------
[  3] local 192.130.0.1 port 60228 connected with 192.130.2.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 2.0 sec   544 KBytes  2.23 Mbits/sec
[  3]  2.0- 4.0 sec   106 KBytes   434 Kbits/sec
[  3]  4.0- 6.0 sec   191 KBytes   782 Kbits/sec
[  3]  6.0- 8.0 sec   382 KBytes  1.56 Mbits/sec
[  3]  8.0-10.0 sec   127 KBytes   521 Kbits/sec
[  3]  0.0-10.5 sec  1.32 MBytes  1.05 Mbits/sec
mininet>
```

3. Ukuran buffer = 60

   Source code yang diubah:

```
# add link
net.addLink(h1, r1, max_queue_size=60, use_htb=True, intfName1='h1-eth0', intfName2='r1-eth0', cls=TCLink, bw=1)
net.addLink(r1, r3, max_queue_size=60, use_htb=True, intfName1='r1-eth1', intfName2='r3-eth1', cls=TCLink, bw=0.5)
net.addLink(h2, r3, max_queue_size=60, use_htb=True, intfName1='h2-eth0', intfName2='r3-eth0', cls=TCLink, bw=1)
net.addLink(h2, r4, max_queue_size=60, use_htb=True, intfName1='h2-eth1', intfName2='r4-eth1', cls=TCLink, bw=1)
net.addLink(r4, r2, max_queue_size=60, use_htb=True, intfName1='r4-eth0', intfName2='r2-eth0', cls=TCLink, bw=0.5)
net.addLink(r2, h1, max_queue_size=60, use_htb=True, intfName1='r2-eth1', intfName2='h1-eth1', cls=TCLink, bw=1)
net.addLink(r3, r2, max_queue_size=60, use_htb=True, intfName1='r3-eth2', intfName2='r2-eth2', cls=TCLink, bw=1)
net.addLink(r1, r4, max_queue_size=60, use_htb=True, intfName1='r1-eth2', intfName2='r4-eth2', cls=TCLink, bw=1)
```

Hasil run:

```
mininet> h1 ping h2
PING 192.130.2.2 (192.130.2.2) 56(84) bytes of data.
64 bytes from 192.130.2.2: icmp_seq=1 ttl=62 time=0.088 ms
64 bytes from 192.130.2.2: icmp_seq=2 ttl=62 time=0.136 ms
64 bytes from 192.130.2.2: icmp_seq=3 ttl=62 time=0.144 ms
64 bytes from 192.130.2.2: icmp_seq=4 ttl=62 time=0.143 ms
^C
--- 192.130.2.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3066ms
rtt min/avg/max/mdev = 0.088/0.127/0.144/0.023 ms
mininet> h1 iperf -i 2 -c 192.130.2.2
------------------------------------------------------------
Client connecting to 192.130.2.2, TCP port 5001
TCP window size:  128 KByte (default)
------------------------------------------------------------
[  3] local 192.130.0.1 port 60232 connected with 192.130.2.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 2.0 sec   544 KBytes  2.23 Mbits/sec
[  3]  2.0- 4.0 sec   106 KBytes   434 Kbits/sec
[  3]  4.0- 6.0 sec   191 KBytes   782 Kbits/sec
[  3]  6.0- 8.0 sec   382 KBytes  1.56 Mbits/sec
[  3]  8.0-10.0 sec   445 KBytes  1.82 Mbits/sec
[  3]  0.0-10.5 sec  1.63 MBytes  1.30 Mbits/sec
mininet>
```

4. Ukuran buffer = 100

   Source code yang diubah:

```
# add link
net.addLink(h1, r1, max_queue_size=100, use_htb=True, intfName1='h1-eth0', intfName2='r1-eth0', cls=TCLink, bw=1)
net.addLink(r1, r3, max_queue_size=100, use_htb=True, intfName1='r1-eth1', intfName2='r3-eth1', cls=TCLink, bw=0.5)
net.addLink(h2, r3, max_queue_size=100, use_htb=True, intfName1='h2-eth0', intfName2='r3-eth0', cls=TCLink, bw=1)
net.addLink(h2, r4, max_queue_size=100, use_htb=True, intfName1='h2-eth1', intfName2='r4-eth1', cls=TCLink, bw=1)
net.addLink(r4, r2, max_queue_size=100, use_htb=True, intfName1='r4-eth0', intfName2='r2-eth0', cls=TCLink, bw=0.5)
net.addLink(r2, h1, max_queue_size=100, use_htb=True, intfName1='r2-eth1', intfName2='h1-eth1', cls=TCLink, bw=1)
net.addLink(r3, r2, max_queue_size=100, use_htb=True, intfName1='r3-eth2', intfName2='r2-eth2', cls=TCLink, bw=1)
net.addLink(r1, r4, max_queue_size=100, use_htb=True, intfName1='r1-eth2', intfName2='r4-eth2', cls=TCLink, bw=1)
```

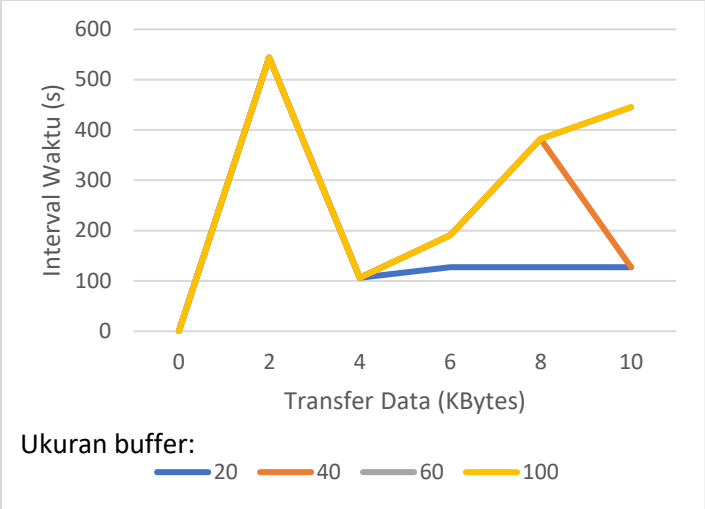
   Hasil run:

```
mininet> h1 ping h2
PING 192.130.2.2 (192.130.2.2) 56(84) bytes of data.
64 bytes from 192.130.2.2: icmp_seq=1 ttl=62 time=0.054 ms
64 bytes from 192.130.2.2: icmp_seq=2 ttl=62 time=0.059 ms
64 bytes from 192.130.2.2: icmp_seq=3 ttl=62 time=0.057 ms
64 bytes from 192.130.2.2: icmp_seq=4 ttl=62 time=0.054 ms
^C
--- 192.130.2.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3060ms
rtt min/avg/max/mdev = 0.054/0.056/0.059/0.002 ms
mininet> h1 iperf -i 2 -c 192.130.2.2
------------------------------------------------------------
Client connecting to 192.130.2.2, TCP port 5001
TCP window size:  128 KByte (default)
------------------------------------------------------------
[  3] local 192.130.0.1 port 40728 connected with 192.130.2.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 2.0 sec   544 KBytes  2.23 Mbits/sec
[  3]  2.0- 4.0 sec   106 KBytes   434 Kbits/sec
[  3]  4.0- 6.0 sec   191 KBytes   782 Kbits/sec
[  3]  6.0- 8.0 sec   382 KBytes  1.56 Mbits/sec
[  3]  8.0-10.0 sec   445 KBytes  1.82 Mbits/sec
[  3]  0.0-10.5 sec  1.63 MBytes  1.30 Mbits/sec
mininet>
```

Analisa pengaruh ukuran buffer:

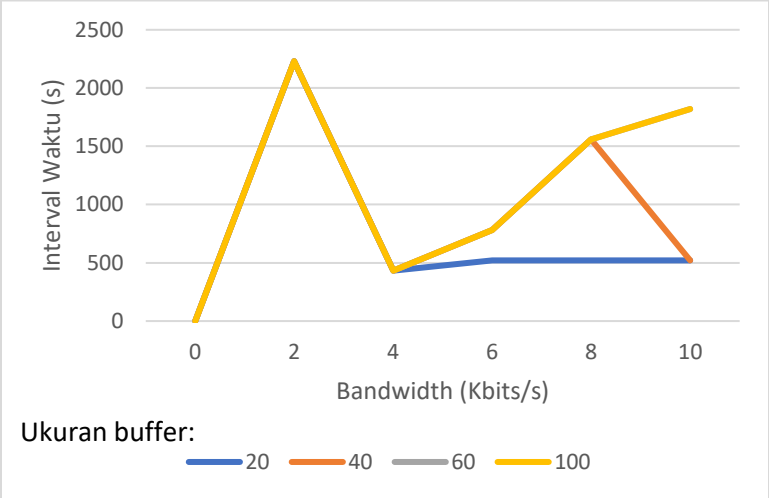
Dari hasil run terlihat bahwa pada setiap ukuran buffer memiliki transfer data dan bandwidth yang berbeda-beda. Berikut merupakan tabel dan grafik interval waktu terhadap transfer data untuk setiap ukuran buffer yang digunakan:

Interval Waktu (s)	Transfer Data (KBytes) dengan Ukuran Buffer:			
	20	40	60	100
0	0	0	0	0
2	544	544	544	544
4	106	106	106	106
6	127	191	191	191
8	127	382	382	382
10	127	127	445	445
0-10.5	1010	1320	1630	1630



Berikut merupakan tabel dan grafik interval waktu terhadap bandwidth untuk setiap ukuran buffer yang digunakan:

Interval Waktu (s)	Bandwidth (Kbits/s) dengan Ukuran Buffer:			
	20	40	60	100
0	0	0	0	0
2	2230	2230	2230	2230
4	434	434	434	434
6	521	782	782	782
8	521	1560	1560	1560
10	521	521	1820	1820
0-10.5	801	1050	1300	1300



Baik dari hasil run, tabel, maupun grafik untuk bandwidth jika dilihat untuk setiap ukuran buffer, terlihat bahwa semakin besar ukuran buffer maka cenderung semakin besar pula rata-rata bandwidthnya. Begitu pula untuk transfer data jika dilihat untuk setiap ukuran buffer, terlihat bahwa semakin besar ukuran buffer maka cenderung semakin besar pula rata-rata transfer datanya.

Link Video Demo
https://youtu.be/8ST-Okn6h78