JARINGAN MPTCP DENGAN MININET

Disusun Untuk Memenuhi Tugas Besar Mata Kuliah Jaringan Komputer Yang Diampu Oleh Dosen Pengampu Siti Amatullah Karimah, S.T., M.T.

Oleh:

Kurniadi Ahmad Wijaya / 1301194024 / IF-43-09

S1 INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY

2021

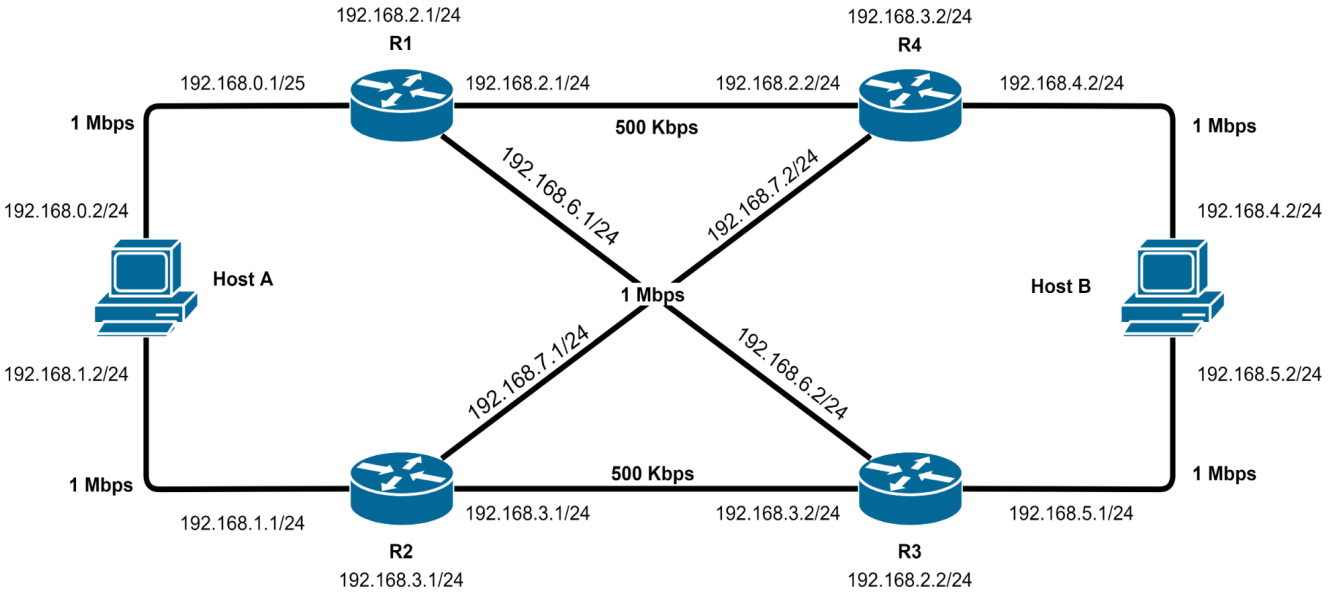
Daftar Isi

CLO 1

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

• Goal : Membandingkan QoS antara TCP dan MPTCP dari host A ke host B.
- Generate traffic dari Host A ke Host B menggunakan iPerf.
- Implementasi MPTCP level kernel pada Host A dan Host B.
- Metric QoS yang digunakan berupa Throughput, Packet Loss dan Delay.
- Analisis hasil perbandingan metric QoS antara TCP dan MPTCP.

1. Struktur Topologi



2. Uji Konektivitas Host A Ke Host B

```
mininet> hA ping hB
PING 192.168.4.2 (192.168.4.2) 56(84) bytes of data.
64 bytes from 192.168.4.2: icmp_seq=1 ttl=62 time=8.73 ms
64 bytes from 192.168.4.2: icmp_seq=2 ttl=62 time=8.74 ms
64 bytes from 192.168.4.2: icmp_seq=3 ttl=62 time=9.59 ms
64 bytes from 192.168.4.2: icmp_seq=4 ttl=62 time=8.86 ms
64 bytes from 192.168.4.2: icmp_seq=5 ttl=62 time=9.77 ms
^C
--- 192.168.4.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4029ms
rtt min/avg/max/mdev = 8.727/9.136/9.767/0.447 ms
mininet>
```

3. **Generate traffic dari Host A ke Host B menggunakan iPerf**

- **Menggunakan MPTCP**

```
mininet> iperf hA hB
*** Iperf: testing TCP bandwidth between hA and hB
*** Results: ['1.35 Mbits/sec', '1.86 Mbits/sec']
```

- **Menggunakan TCP**

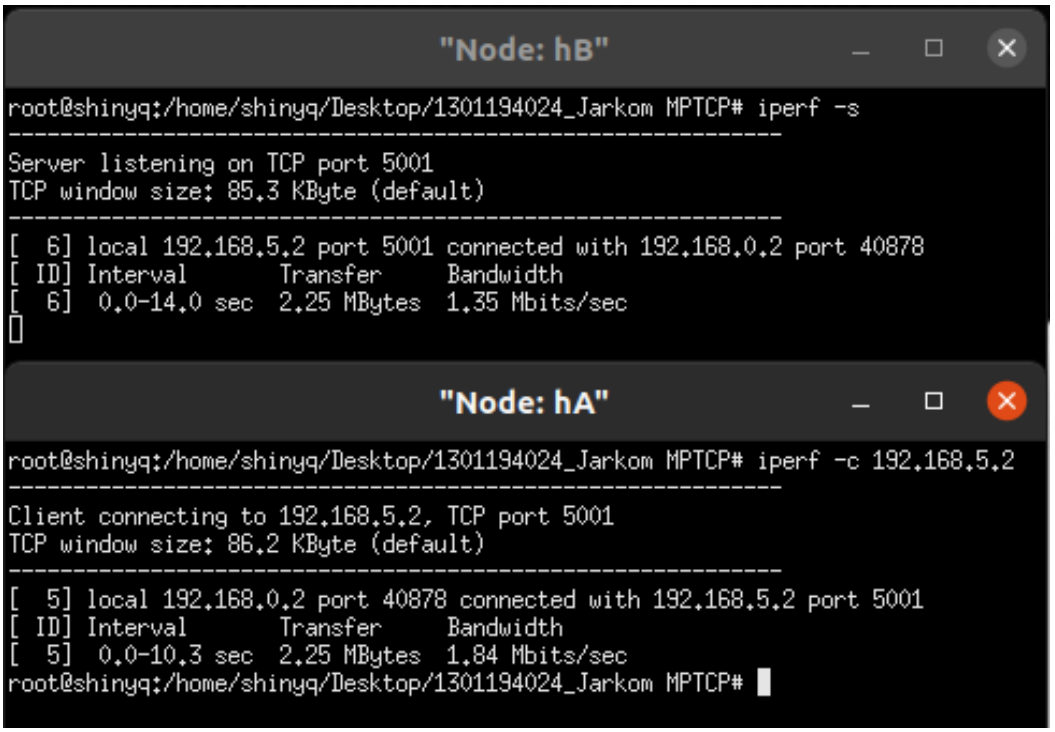
```
mininet> iperf hA hB
*** Iperf: testing TCP bandwidth between hA and hB
*** Results: ['485 Kbits/sec', '1.16 Mbits/sec']
```

4. **Perbandingan Performansi TCP Dan MPTCP**

Untuk analisis QOS yang digunakan pada perbandingan performansi ini adalah nilai bandwidth sesuai ketentuan tugas besar, delay 1ms, dan queue size 20.

```
linkopts0 = dict(bw=0.5, delay='1ms', loss=0, max_queue_size=20, use_tbf=True)
linkopts1 = dict(bw=1, delay='1ms', loss=0, max_queue_size=20, use_tbf=True)
```

- **Implementasi Pada MPTCP**



- **Implementasi Pada TCP**

Kesimpulan

Dapat kita lihat dari hasil pengetesan iperf diatas didapatkan hasil Transfer dan Bandwidth yang tinggi terhadap host A dan host B pada pengimplementasian MPTCP jika dibandingkan dengan hanya menggunakan TCP saja. Hal ini dikarenakan MPTCP dalam prosesnya membangun beberapa subflows dari sebagian atau seluruh IP pengirim ke sebagian atau ke seluruh IP penerima sehingga proses pengiriman paket dapat menjadi lebih cepat ketimbang menggunakan TCP saja.

CLO 2

• Goal : Mengimplementasikan mekanisme Routing pada topologi yang ada

- Generate traffic dan background traffic menggunakan iPerf.
- Implementasi MPTCP level kernel pada Host A dan Host B.
- Melakukan pemutusan link R1 ke R4 untuk mensimulasikan link failure .
- Implementasikan Routing pada topologi (RIP, OSPF).
- Metric yang digunakan berupa Convergence Time, Delay.
- Analisis durasi Convergence Time dan Delay yang terjadi.

1. Konfigurasi Subnetting

Konfigurasi Static Routing MPTCP

Pada proses ini dilakukan route untuk Host A dan Host B pada konfigurasi mptcp dimana kita melakukan pemetaan untuk table 1 sebagai konfigurasi router bagian atas yaitu R1 dan R3 serta tabel 2 untuk router bawah yaitu R2 dan R4.

```
# Static Routing MPTCP
net['A'].cmd("ip rule add from 192.168.0.2 table 1")
net['A'].cmd("ip rule add from 192.168.1.2 table 2")
net['A'].cmd("ip route add 192.168.0.0/24 dev A-eth0 scope link table 1")
net['A'].cmd("ip route add default via 192.168.0.1 dev A-eth0 table 1")
net['A'].cmd("ip route add 192.168.1.0/24 dev A-eth1 scope link table 2")
net['A'].cmd("ip route add default via 192.168.1.1 dev A-eth1 table 2")
net['A'].cmd("ip route add default scope global nexthop via 192.168.0.1 dev A-eth0")

net['B'].cmd("ip rule add from 192.168.4.2 table 1")
net['B'].cmd("ip rule add from 192.168.5.2 table 2")
net['B'].cmd("ip route add 192.168.4.0/24 dev B-eth0 scope link table 1")
net['B'].cmd("ip route add default via 192.168.4.1 dev B-eth0 table 1")
net['B'].cmd("ip route add 192.168.5.0/24 dev B-eth1 scope link table 2")
net['B'].cmd("ip route add default via 192.168.5.1 dev B-eth1 table 2")
net['B'].cmd("ip route add default scope global nexthop via 192.168.4.1 dev B-eth0")
```

Konfigurasi Host Dan Router

Pada konfigurasi host dan router, konfigurasi yang dilakukan sesuai dengan topologi yang telah dibuat pada bagian CLO 1 sehingga dihasilkan konfigurasi IP Host, IP Router, Gateway serta IP ethernet seperti pada gambar berikut.

```
# Menambahkan router r1-r4
r1 = self.addNode('r1', cls=LinuxRouter, ip='192.168.2.1/24')
r2 = self.addNode('r2', cls=LinuxRouter, ip='192.168.3.1/24')
r3 = self.addNode('r3', cls=LinuxRouter, ip='192.168.2.2/24')
r4 = self.addNode('r4', cls=LinuxRouter, ip='192.168.3.2/24')

# Menambahkan host hA & hB
hA = self.addHost('hA', ip='192.168.0.2/24', defaultRoute='via 192.168.0.1')
hB = self.addHost('hB', ip='192.168.4.2/24', defaultRoute='via 192.168.4.1')
```

```
# Menambahkan links pada nodes
# Router <--> Router
# Ukuran Queue : 20, 40, 60 dan 100

linkopts0 = dict(bw=0.5, delay='1ms', loss=0, max_queue_size=20, use_tbf=True)
linkopts1 = dict(bw=1, delay='1ms', loss=0, max_queue_size=20, use_tbf=True)

self.addLink(r1, r3, cls=TCLink, **linkopts0, intfName1='r1-eth1', intfName2='r3-eth1',
             params1={'ip': '192.168.2.1/24'},
             params2={'ip': '192.168.2.2/24'})

self.addLink(r1, r4, cls=TCLink, **linkopts1, intfName1='r1-eth2', intfName2='r4-eth1',
             params1={'ip': '192.168.6.1/24'},
             params2={'ip': '192.168.6.2/24'})

self.addLink(r2, r4, cls=TCLink, **linkopts0, intfName1='r2-eth1', intfName2='r4-eth2',
             params1={'ip': '192.168.3.1/24'},
             params2={'ip': '192.168.3.2/24'})

self.addLink(r2, r3, cls=TCLink, **linkopts1, intfName1='r2-eth2', intfName2='r3-eth2',
             params1={'ip': '192.168.7.1/24'},
             params2={'ip': '192.168.7.2/24'})
```

```
# Router <--> Host

self.addLink(hA, r1, cls=TCLink, **linkopts1,  intfName2='r1-eth3',
             params1={'ip': '192.168.0.2/24'},
             params2={'ip': '192.168.0.1/24'})

self.addLink(hA, r2, cls=TCLink, **linkopts1, intfName2='r2-eth3',
             params1={'ip': '192.168.1.2/24'},
             params2={'ip': '192.168.1.1/24'})

self.addLink(hB, r3, cls=TCLink, **linkopts1, intfName2='r3-eth3',
             params1={'ip': '192.168.4.2/24'},
             params2={'ip': '192.168.4.1/24'})

self.addLink(hB, r4, cls=TCLink, **linkopts1, intfName2='r4-eth3',
             params1={'ip': '192.168.5.2/24'},
             params2={'ip': '192.168.5.1/24'})
```

Adapun hasil sukses sambungan koneksi adalah sebagai berikut:

```
*** Adding links:
(1.00Mbit 1ms delay 0.00000% loss) (1.00Mbit 1ms delay 0.00000% loss) (hA, r1)
(1.00Mbit 1ms delay 0.00000% loss) (1.00Mbit 1ms delay 0.00000% loss) (hA, r2)
(1.00Mbit 1ms delay 0.00000% loss) (1.00Mbit 1ms delay 0.00000% loss) (hB, r3)
(1.00Mbit 1ms delay 0.00000% loss) (1.00Mbit 1ms delay 0.00000% loss) (hB, r4)
(0.50Mbit 1ms delay 0.00000% loss) (0.50Mbit 1ms delay 0.00000% loss) (r1, r3)
(1.00Mbit 1ms delay 0.00000% loss) (1.00Mbit 1ms delay 0.00000% loss) (r1, r4)
(1.00Mbit 1ms delay 0.00000% loss) (1.00Mbit 1ms delay 0.00000% loss) (r2, r3)
(0.50Mbit 1ms delay 0.00000% loss) (0.50Mbit 1ms delay 0.00000% loss) (r2, r4)
```

Konfigurasi Dynamic Routing RIP

Pada bagian dynamic routing projek ini menggunakan RIP adapun konfigurasi yang dilakukan sesuai dengan library bantuan quagga adalah sebagai berikut.

- **Konfigurasi Router 1**

r1ripd.conf

config > rip > r1ripd.conf

```
1  hostname r1rip
2  password en
3
4  router rip
5   network 192.168.0.0/24
6   network 192.168.2.0/24
7   network 192.168.6.0/24
```

- **Konfigurasi Router 2**

r2ripd.conf

config > rip > r2ripd.conf

```
1  hostname r2rip
2  password en
3
4  router rip
5   network 192.168.1.0/24
6   network 192.168.3.0/24
7   network 192.168.7.0/24
```

- **Konfigurasi Router 3**

r3ripd.conf

config > rip > r3ripd.conf

```
1  hostname r3rip
2  password en
3
4  router rip
5   network 192.168.4.0/24
6   network 192.168.2.0/24
7   network 192.168.7.0/24
```

- **Konfigurasi Router 4**

r4ripd.conf

config > rip > r4ripd.conf

```
1  hostname r4rip
2  password en
3
4  router rip
5   network 192.168.5.0/24
6   network 192.168.6.0/24
7   network 192.168.3.0/24
```

2. Uji Konektivitas Router

Hasil iperf Generate Traffic

Berikut merupakan tampilan dari hasil iperf untuk traffic antara Host A (192.168.0.2) ke Host B (192.168.5.2).

```
*** Bandwidth test
*** hA : ('iperf -c 192.168.5.2 -i 1',)
------------------------------------------------------------
Client connecting to 192.168.5.2, TCP port 5001
TCP window size: 86.2 KByte (default)
------------------------------------------------------------
[  3] local 192.168.0.2 port 41784 connected with 192.168.5.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec   771 KBytes  6.32 Mbits/sec
[  3]  1.0- 2.0 sec   297 KBytes  2.43 Mbits/sec
[  3]  2.0- 3.0 sec   284 KBytes  2.33 Mbits/sec
[  3]  3.0- 4.0 sec  97.6 KBytes   800 Kbits/sec
[  3]  4.0- 5.0 sec   257 KBytes  2.10 Mbits/sec
[  3]  5.0- 6.0 sec   215 KBytes  1.76 Mbits/sec
[  3]  6.0- 7.0 sec   126 KBytes  1.03 Mbits/sec
[  3]  7.0- 8.0 sec   188 KBytes  1.54 Mbits/sec
[  3]  8.0- 9.0 sec  90.6 KBytes   743 Kbits/sec
[  3]  9.0-10.0 sec   307 KBytes  2.51 Mbits/sec
[  3]  0.0-10.3 sec  2.57 MBytes  2.10 Mbits/sec
```

Pemutusan link R1 ke R4

Untuk proses pemutusan link kita dapat menggunakan command link r1 r4 down pada mininet untuk melakukan simulasi link failure.

```
mininet> link r1 r4 down
mininet> pingall
*** Ping: testing ping reachability
hA -> hB r1 r2 r3 X
hB -> hA r1 r2 r3 X
r1 -> hA hB r2 r3 X
r2 -> hA hB r1 r3 r4
r3 -> hA hB r1 r2 X
r4 -> X X X r2 X
*** Results: 26% dropped (22/30 received)
```

3. Ketepatan analisis proses routing

Pada proses routing yang dilakukan digunakan proses perhitungan waktu convergent yaitu dengan melakukan set waktu awal melakukan perulangan saat melakukan ping terhadap seluruh node sampai seluruh node berhasil terkoneksikan. Setelah itu kita kurangkan dengan waktu proses loop selesai.

```
119     loss = 100
120     while(loss > 0):
121         loss = net.pingAll()
122
123     time_end = datetime.now() - time_start
124     print(f'Percentage Loss : {loss}')
125     print(f'Convergence Time: {time_end.total_seconds()}s')
```

Berikut merupakan hasil salah satu contoh proses penghitungan waktu convergent pada ping keseluruhan node yang sukses. Pada hasil tersebut didapatkan waktu 11.1s yang merupakan hasil wajar dalam load sistem router dikarenakan melakukan load library quagga yang dibutuhkan untuk proses routing dynamic RIP.

```
*** Connection test
*** Ping: testing ping reachability
hA -> X r1 X X 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: 10% dropped (27/30 received)
*** 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)
Percentage Loss : 0.0
Convergence Time: 11.097589s
```

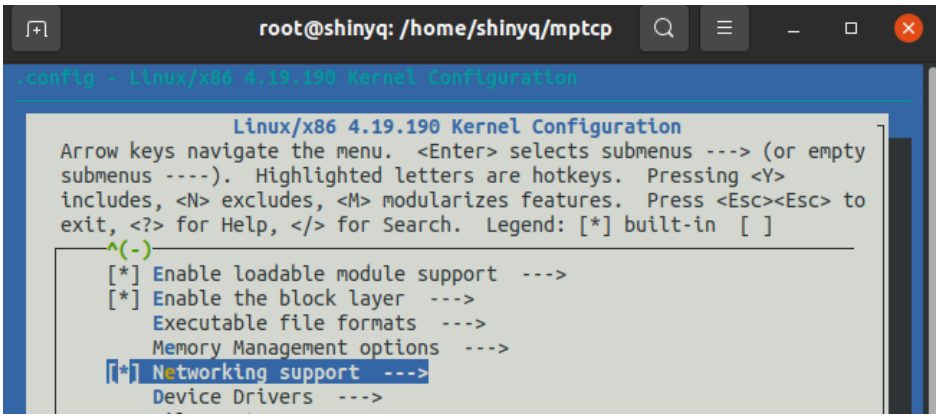
CLO 3

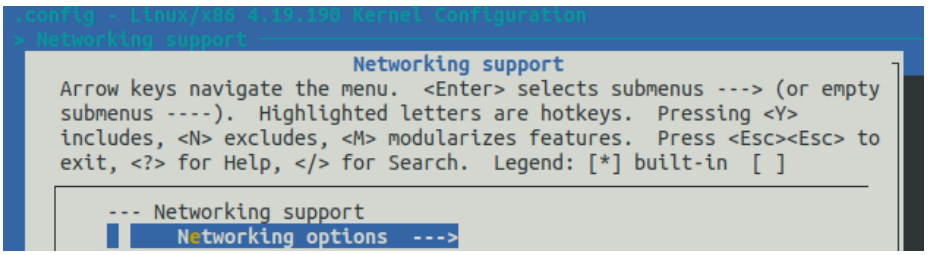
• Goal : Membuktikan bahwa MPTCP diimplementasikan dengan benar pada topologi.
- Generate traffic dan background traffic menggunakan iPerf.
- Implementasi MPTCP level kernel pada Host A dan Host B.
- Capture trafik yang mengindikasikan MPTCP menggunakan custom script atau wireshark.
- Analisis hasil capture traffic, buktikan bahwa MPTCP terimplementasi.

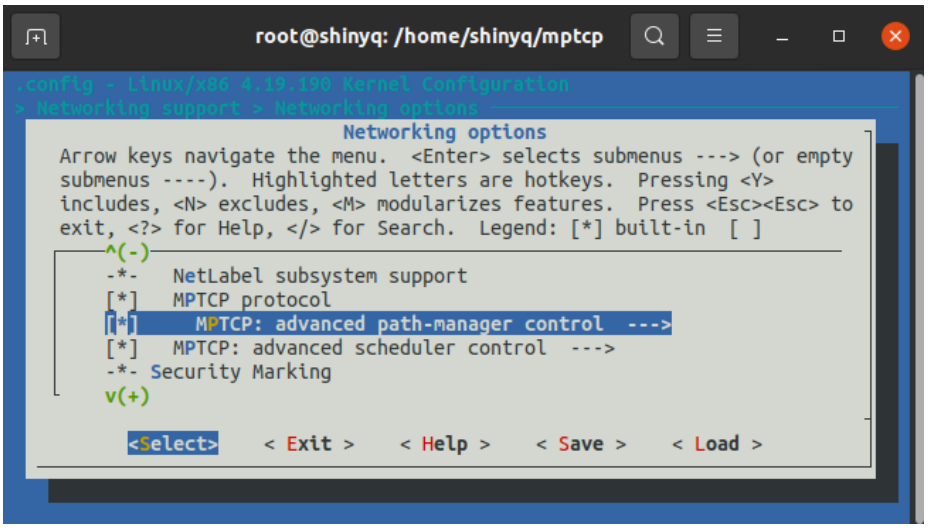
1. Ketepatan Konfigurasi Dan Verifikasi MPTCP

Proses Konfigurasi

Untuk proses konfigurasi MPTCP dilakukan instalasi kernel dengan option sebagai berikut : Networking Support > Networking Options. Centang bagian MPTCP Protocol beserta sub-subnya kemudian dilakukan save dan memulai instalasi kernel.







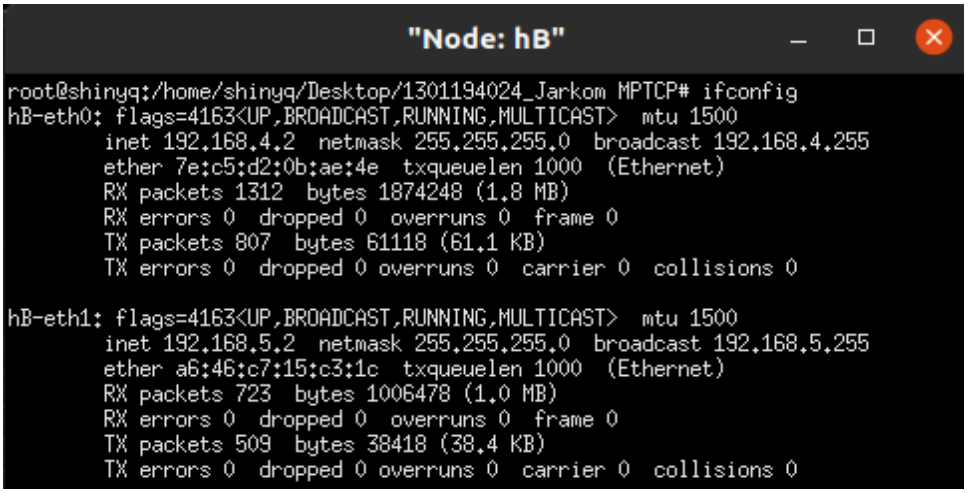
Verifikasi Hasil Konfigurasi MPTCP

Untuk melakukan pengecekan apakah MPTCP sudah terinstal pada sistem operasi ubuntu dapat menggunakan command sysctl -a | grep mptcp.

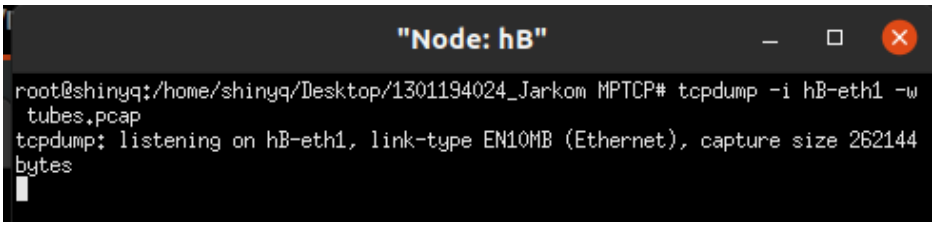
```
root@shinyq:/home/shinyq# sysctl -a | grep mptcp
net.mptcp.mptcp_binder_gateways =
net.mptcp.mptcp_checksum = 1
net.mptcp.mptcp_debug = 0
net.mptcp.mptcp_enabled = 1
net.mptcp.mptcp_path_manager = fullmesh
net.mptcp.mptcp_scheduler = roundrobin
net.mptcp.mptcp_syn_retries = 3
net.mptcp.mptcp_version = 0
```

2. Capture Packet Dan Analisis Penggunaan MPTCP

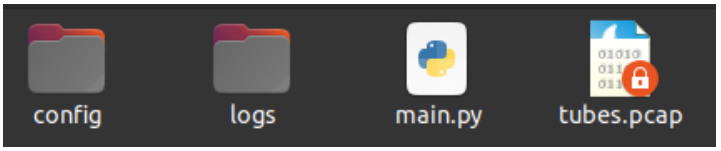
Untuk melakukan capture data kita dapat menggunakan 2 tools yaitu tcpdump ataupun wireshark sebagai bentuk analisis data yang lebih jelas adapun langkah pertama yang harus kita lakukan adalah mengecek ip serta interface untuk salah satu node yang akan kita ping.

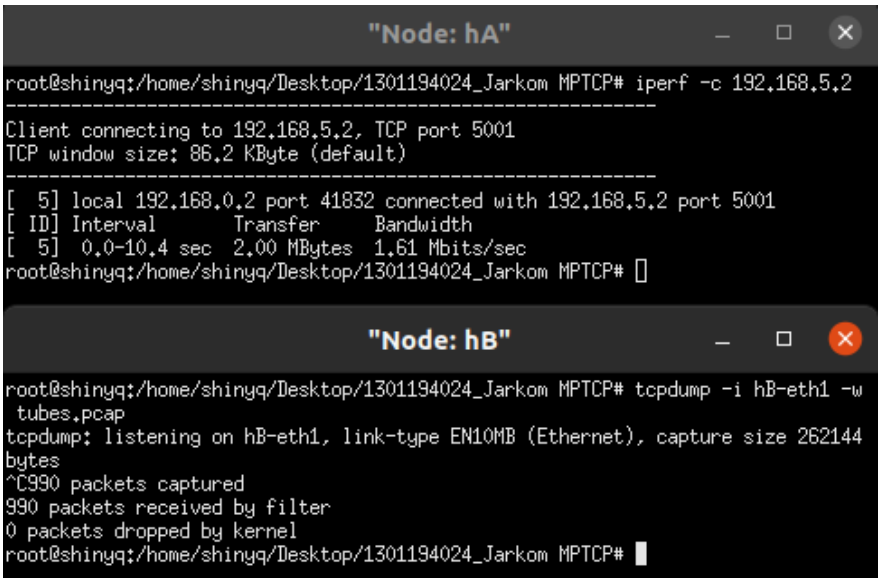


Kemudian kita jalankan iperf -s& pada node yang akan di traffic (hB). Setelah itu kita dapat langsung melakukan tcpdump pada node yang akan di traffic (hB) dengan menggunakan command tcpdump -i <int> -w <namafile>.pcap sehingga tcpdump akan melakukan listening terhadap paket yang masuk atau keluar.

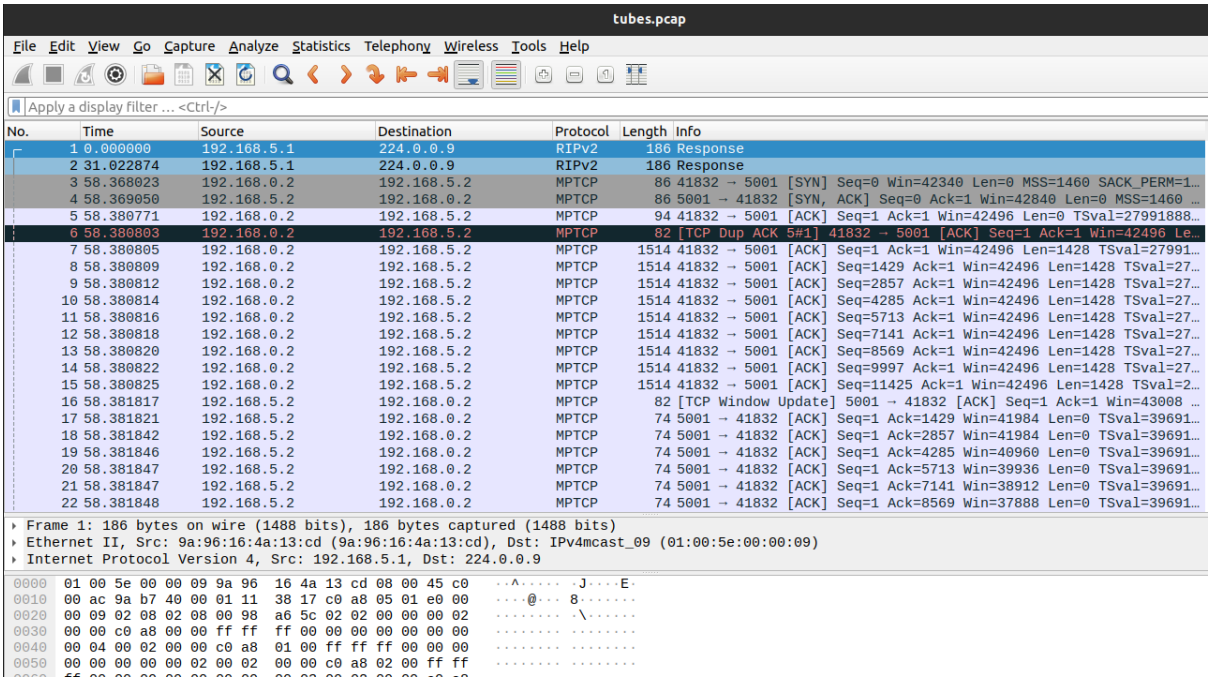


Selanjutnya kita lakukan proses running trafik iperf pada node hA sehingga didapatkanlah file .pcap yang digenerate oleh tcpdump.

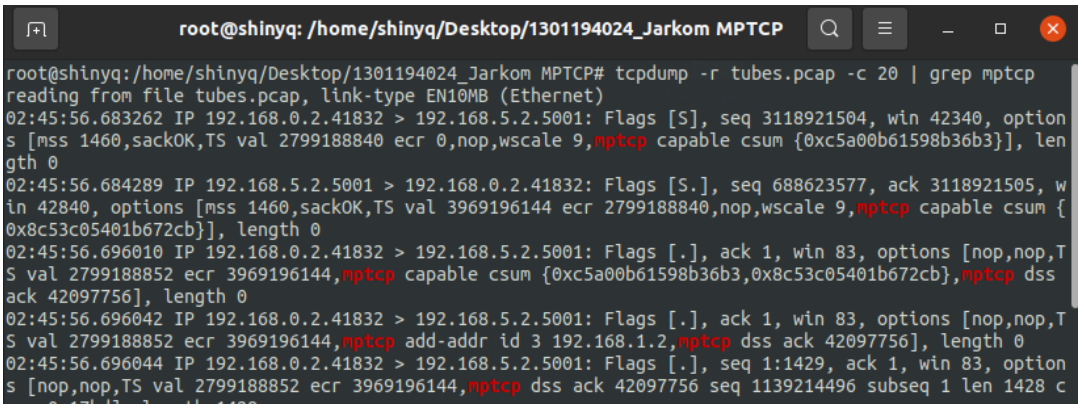




Pengecekan Pada Wireshark



Pengecekan Menggunakan Grep



CLO 4

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

• Goal : Menginspeksi penggunaan queue pada router jaringan.
- Generate traffic dan background traffic menggunakan iPerf.
- Implementasi MPTCP level kernel pada Host A dan Host B.
- Set ukuran buffer pada router : 20, 40, 60 dan 100.
- Capture pengaruh ukuran buffer terhadap delay.
- Analisis eksperimen hasil variasi ukuran buffer.

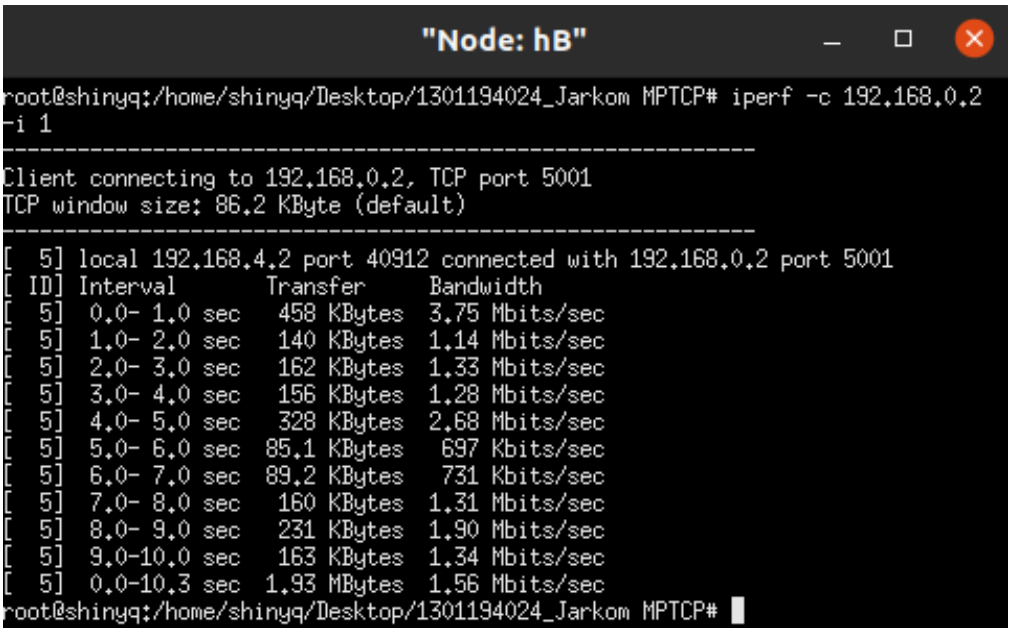
1. Konfigurasi Dan Analisis Ukuran Buffer

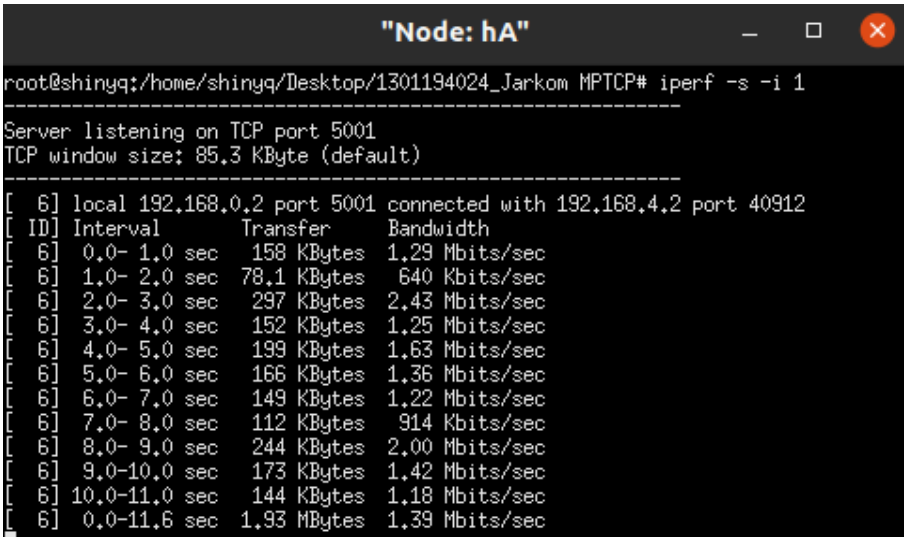
Pada program, kita dapat melakukan modifikasi terhadap ukuran buffer yang kita inginkan dengan melakukan perubahan terhadap value yang diinginkan seperti pada gambar dibawah ini.

```
# Ukuran Buffer : 20, 40, 60 dan 100

buffer = 20
# buffer = 40
# buffer = 60
# buffer = 100

linkopts0 = dict(bw=0.5, delay='1ms', loss=0, max_queue_size=buffer, use_tbf=True)
linkopts1 = dict(bw=1, delay='1ms', loss=0, max_queue_size=buffer, use_tbf=True)
```

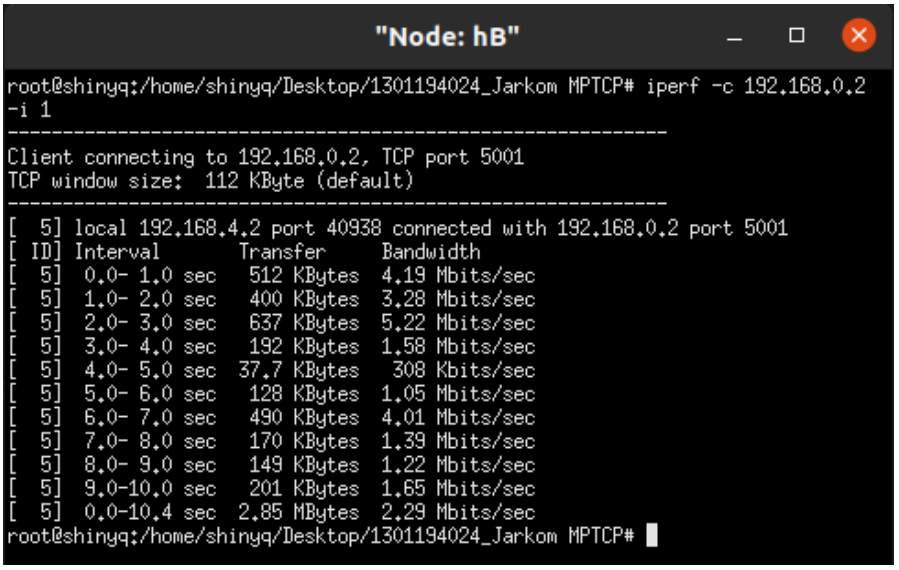
Adapun hasil dari masing-masing dari running masing - masing buffer pada pada program tersebut adalah sebagai berikut.

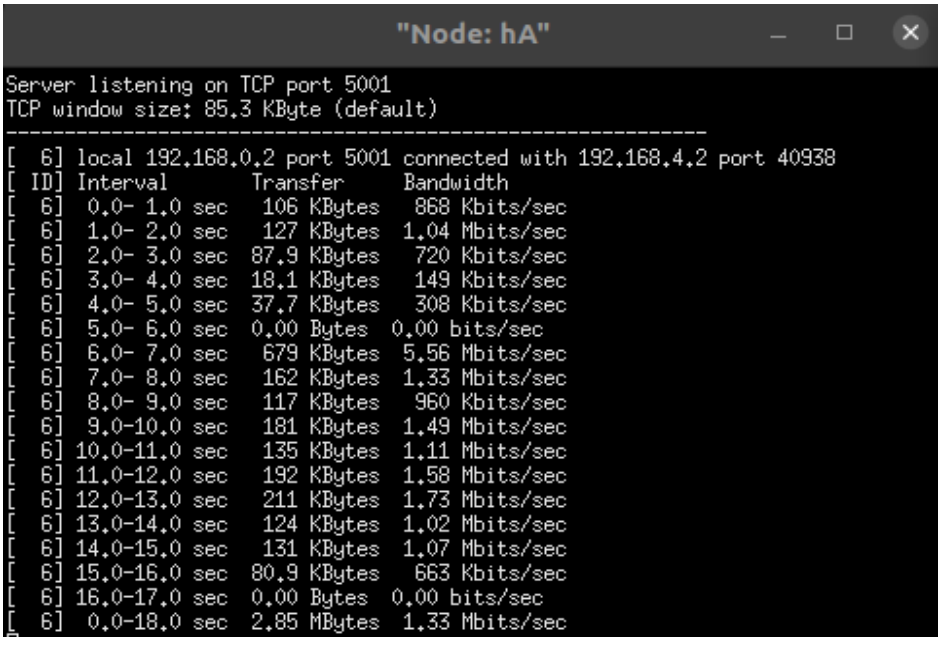
- Buffer 20



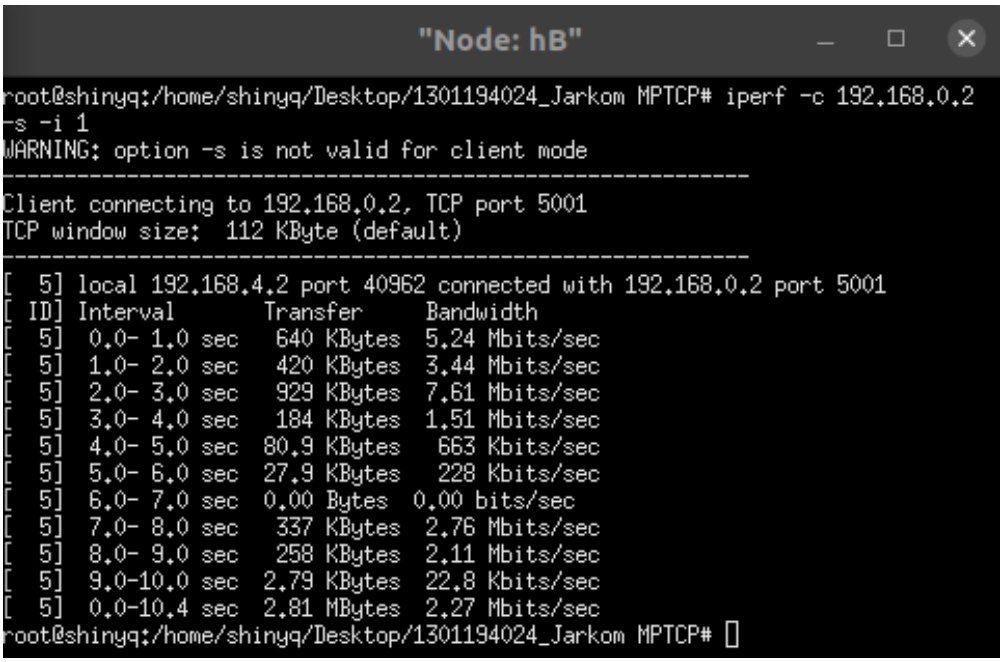

"Node: hA"
root@shinyq:/home/shinyq/Desktop/1301194024_Jarkom MPTCP# iperf -s -i 1
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
[6] local 192.168.0.2 port 5001 connected with 192.168.4.2 port 40912
[ID] Interval Transfer Bandwidth
[6] 0.0- 1.0 sec 158 KBytes 1.29 Mbits/sec
[6] 1.0- 2.0 sec 78.1 KBytes 640 Kbits/sec
[6] 2.0- 3.0 sec 297 KBytes 2.43 Mbits/sec
[6] 3.0- 4.0 sec 152 KBytes 1.25 Mbits/sec
[6] 4.0- 5.0 sec 199 KBytes 1.63 Mbits/sec
[6] 5.0- 6.0 sec 166 KBytes 1.36 Mbits/sec
[6] 6.0- 7.0 sec 149 KBytes 1.22 Mbits/sec
[6] 7.0- 8.0 sec 112 KBytes 914 Kbits/sec
[6] 8.0- 9.0 sec 244 KBytes 2.00 Mbits/sec
[6] 9.0-10.0 sec 173 KBytes 1.42 Mbits/sec
[6] 10.0-11.0 sec 144 KBytes 1.18 Mbits/sec
[6] 0.0-11.6 sec 1.93 MBytes 1.39 Mbits/sec

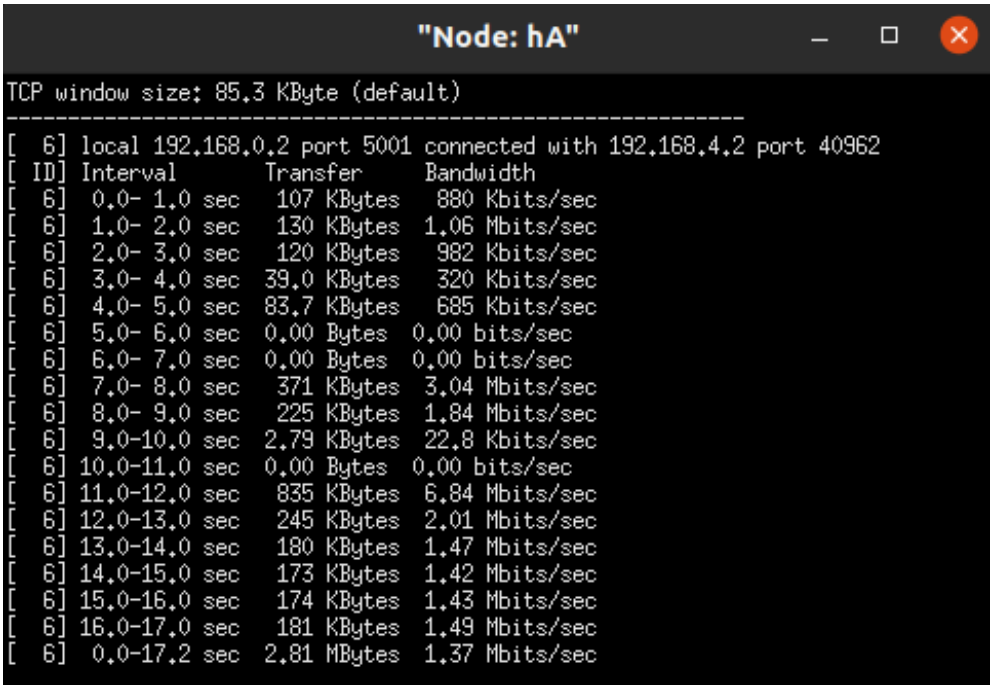
- **Buffer 40**


"Node: hB"
root@shinyq:/home/shinyq/Desktop/1301194024_Jarkom MPTCP# iperf -c 192.168.0.2
-i 1
Client connecting to 192.168.0.2, TCP port 5001
TCP window size: 112 KByte (default)
[5] local 192.168.4.2 port 40938 connected with 192.168.0.2 port 5001
[ID] Interval Transfer Bandwidth
[5] 0.0- 1.0 sec 512 KBytes 4.19 Mbits/sec
[5] 1.0- 2.0 sec 400 KBytes 3.28 Mbits/sec
[5] 2.0- 3.0 sec 637 KBytes 5.22 Mbits/sec
[5] 3.0- 4.0 sec 192 KBytes 1.58 Mbits/sec
[5] 4.0- 5.0 sec 37.7 KBytes 308 Kbits/sec
[5] 5.0- 6.0 sec 128 KBytes 1.05 Mbits/sec
[5] 6.0- 7.0 sec 490 KBytes 4.01 Mbits/sec
[5] 7.0- 8.0 sec 170 KBytes 1.39 Mbits/sec
[5] 8.0- 9.0 sec 149 KBytes 1.22 Mbits/sec
[5] 9.0-10.0 sec 201 KBytes 1.65 Mbits/sec
[5] 0.0-10.4 sec 2.85 MBytes 2.29 Mbits/sec
root@shinyq:/home/shinyq/Desktop/1301194024_Jarkom MPTCP#

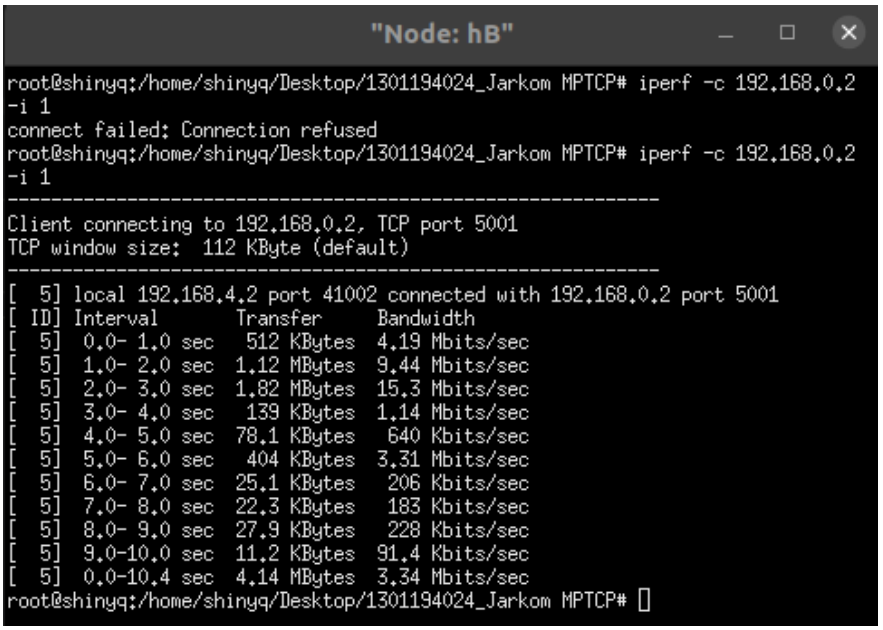

"Node: hA"
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
[6] local 192.168.0.2 port 5001 connected with 192.168.4.2 port 40938
[ID] Interval Transfer Bandwidth
[6] 0.0- 1.0 sec 106 KBytes 868 Kbits/sec
[6] 1.0- 2.0 sec 127 KBytes 1.04 Mbits/sec
[6] 2.0- 3.0 sec 87.9 KBytes 720 Kbits/sec
[6] 3.0- 4.0 sec 18.1 KBytes 149 Kbits/sec
[6] 4.0- 5.0 sec 37.7 KBytes 308 Kbits/sec
[6] 5.0- 6.0 sec 0.00 Bytes 0.00 bits/sec
[6] 6.0- 7.0 sec 679 KBytes 5.56 Mbits/sec
[6] 7.0- 8.0 sec 162 KBytes 1.33 Mbits/sec
[6] 8.0- 9.0 sec 117 KBytes 960 Kbits/sec
[6] 9.0-10.0 sec 181 KBytes 1.49 Mbits/sec
[6] 10.0-11.0 sec 135 KBytes 1.11 Mbits/sec
[6] 11.0-12.0 sec 192 KBytes 1.58 Mbits/sec
[6] 12.0-13.0 sec 211 KBytes 1.73 Mbits/sec
[6] 13.0-14.0 sec 124 KBytes 1.02 Mbits/sec
[6] 14.0-15.0 sec 131 KBytes 1.07 Mbits/sec
[6] 15.0-16.0 sec 80.9 KBytes 663 Kbits/sec
[6] 16.0-17.0 sec 0.00 Bytes 0.00 bits/sec
[6] 0.0-18.0 sec 2.85 MBytes 1.33 Mbits/sec

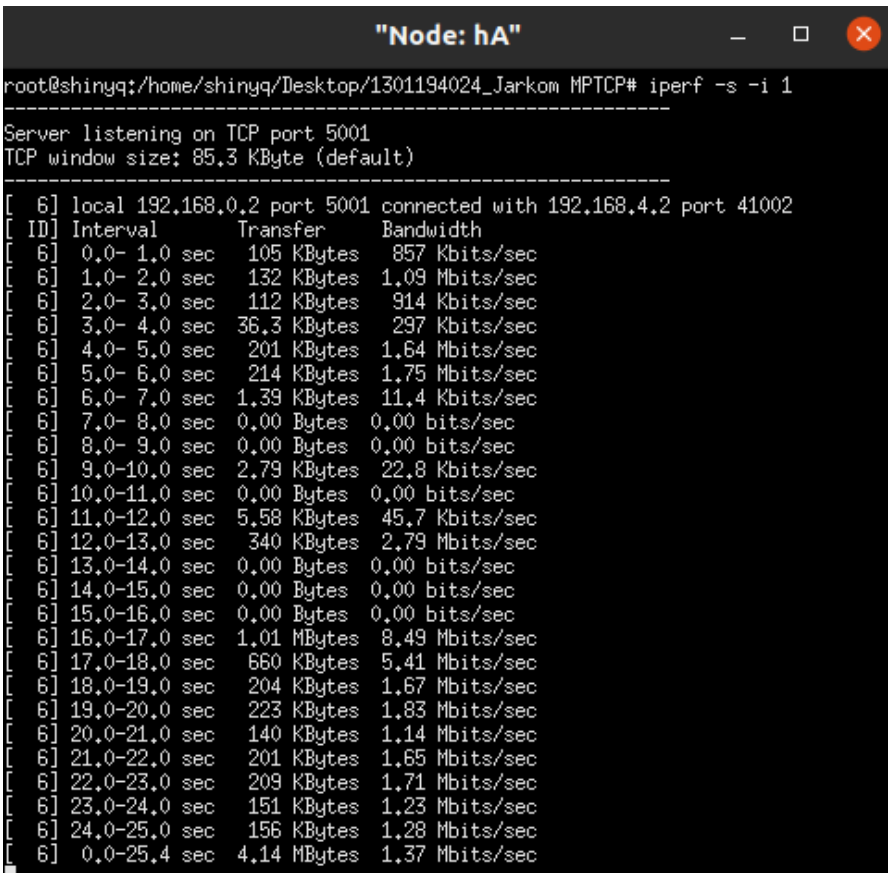
- **Buffer 60**


"Node: hB"
root@shinyq:/home/shinyq/Desktop/1301194024_Jarkom MPTCP# iperf -c 192.168.0.2
-s -i 1
WARNING: option -s is not valid for client mode
Client connecting to 192.168.0.2, TCP port 5001
TCP window size: 112 KByte (default)
[5] local 192.168.4.2 port 40962 connected with 192.168.0.2 port 5001
[ID] Interval Transfer Bandwidth
[5] 0.0- 1.0 sec 640 KBytes 5.24 Mbits/sec
[5] 1.0- 2.0 sec 420 KBytes 3.44 Mbits/sec
[5] 2.0- 3.0 sec 929 KBytes 7.61 Mbits/sec
[5] 3.0- 4.0 sec 184 KBytes 1.51 Mbits/sec
[5] 4.0- 5.0 sec 80.9 KBytes 663 Kbits/sec
[5] 5.0- 6.0 sec 27.9 KBytes 228 Kbits/sec
[5] 6.0- 7.0 sec 0.00 Bytes 0.00 bits/sec
[5] 7.0- 8.0 sec 337 KBytes 2.76 Mbits/sec
[5] 8.0- 9.0 sec 258 KBytes 2.11 Mbits/sec
[5] 9.0-10.0 sec 2.79 KBytes 22.8 Kbits/sec
[5] 0.0-10.4 sec 2.81 MBytes 2.27 Mbits/sec
root@shinyq:/home/shinyq/Desktop/1301194024_Jarkom MPTCP#


"Node: hA"
TCP window size: 85.3 KByte (default)
[6] local 192.168.0.2 port 5001 connected with 192.168.4.2 port 40962
[ID] Interval Transfer Bandwidth
[6] 0.0- 1.0 sec 107 KBytes 880 Kbits/sec
[6] 1.0- 2.0 sec 130 KBytes 1.06 Mbits/sec
[6] 2.0- 3.0 sec 120 KBytes 982 Kbits/sec
[6] 3.0- 4.0 sec 39.0 KBytes 320 Kbits/sec
[6] 4.0- 5.0 sec 83.7 KBytes 685 Kbits/sec
[6] 5.0- 6.0 sec 0.00 Bytes 0.00 bits/sec
[6] 6.0- 7.0 sec 0.00 Bytes 0.00 bits/sec
[6] 7.0- 8.0 sec 371 KBytes 3.04 Mbits/sec
[6] 8.0- 9.0 sec 225 KBytes 1.84 Mbits/sec
[6] 9.0-10.0 sec 2.79 KBytes 22.8 Kbits/sec
[6] 10.0-11.0 sec 0.00 Bytes 0.00 bits/sec
[6] 11.0-12.0 sec 835 KBytes 6.84 Mbits/sec
[6] 12.0-13.0 sec 245 KBytes 2.01 Mbits/sec
[6] 13.0-14.0 sec 180 KBytes 1.47 Mbits/sec
[6] 14.0-15.0 sec 173 KBytes 1.42 Mbits/sec
[6] 15.0-16.0 sec 174 KBytes 1.43 Mbits/sec
[6] 16.0-17.0 sec 181 KBytes 1.49 Mbits/sec
[6] 0.0-17.2 sec 2.81 MBytes 1.37 Mbits/sec

- **Buffer 100**





Kesimpulan:

Adanya peningkatan buffer pada program meningkatkan lama proses paket dikirimkan (interval). Hal ini terjadi karena adanya delay yang disebabkan oleh paket yang masuk ke dalam buffer semakin banyak. Banyaknya paket yang masuk kedalam buffer berarti juga meningkatkan kapasitas bandwith yang ada. Hal ini dapat kita lihat dari Interval dan hasil bandwith yang ada pada percobaan diatas yang semakin meningkat setiap menaikkan nilai buffer.

2. Test Capture Queue Length

- **Buffer 20**

```
mininet> r1 tc -s qdisc show dev r1-eth1
qdisc tbf 5: root refcnt 2 rate 500Kbit burst 15000b lat 240.0ms
 Sent 896096 bytes 638 pkt (dropped 37, overlimits 1201 requeues 0)
 backlog 0b 0p requeues 0
qdisc netem 10: parent 5:1 limit 20 delay 1.0ms
 Sent 896096 bytes 638 pkt (dropped 37, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
```

- **Buffer 40**

```
mininet> r1 tc -s qdisc show dev r1-eth1
qdisc tbf 5: root refcnt 2 rate 500Kbit burst 15000b lat 240.0ms
 Sent 1392730 bytes 967 pkt (dropped 106, overlimits 1914 requeues 0)
 backlog 0b 0p requeues 0
qdisc netem 10: parent 5:1 limit 40 delay 1.0ms
 Sent 1392730 bytes 967 pkt (dropped 106, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
```

- **Buffer 60**

```
mininet> r1 tc -s qdisc show dev r1-eth1
qdisc tbf 5: root refcnt 2 rate 500Kbit burst 15000b lat 240.0ms
 Sent 1270054 bytes 885 pkt (dropped 92, overlimits 1711 requeues 0)
 backlog 0b 0p requeues 0
qdisc netem 10: parent 5:1 limit 60 delay 1.0ms
 Sent 1270054 bytes 885 pkt (dropped 92, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
```

- **Buffer 100**

```
mininet> r1 tc -s qdisc show dev r1-eth1
qdisc tbf 5: root refcnt 2 rate 500Kbit burst 15000b lat 240.0ms
 Sent 1364036 bytes 1766 pkt (dropped 147, overlimits 1817 requeues 0)
 backlog 0b 0p requeues 0
qdisc netem 10: parent 5:1 limit 100 delay 1.0ms
 Sent 1364036 bytes 1766 pkt (dropped 147, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
```

Daftar Pustaka

Iceteam. (2017). *How To Install MPTCP*. Diakses pada 30 Mei 2021. https://multipath-tcp.org/pmwiki.php/Users/HowToInstallMPTCP

Chih-Heng Ke. (2017). *MPTCP Test With Mininet*. Diakses pada 30 Mei 2021. http://csie.nqu.edu.tw/smallko/sdn/mptcp-test.htm

R. Veby, N. Sofia, M. Ridha. (2018). ANALISIS MULTIPATH TCP DENGAN OPENFLOW BERBASIS SOFTWARE DEFINED NETWORK (SDN). Diakses pada 6 juni 2021.

qyang18. (2017). *Mininet Quagga*. Diakses pada 6 juni 2021. https://github.com/qyang18/Mininet-Quagga