

LAPORAN STUDI KASUS
PERBANDINGAN *QUALITY OF SERVICE* TCP dan MPTCP

Tugas

ditujukan untuk memenuhi salah satu tugas mata kuliah Jaringan Komputer

oleh:

Adrian Arman Fuadi (1303193048)

IT-43-01



PROGRAM STUDI TEKNOLOGI INFORMASI

FAKULTAS INFORMATIKA

UNIVERSITAS TELKOM

BANDUNG

2020

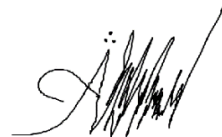
KATA PENGANTAR

Dengan mengucapkan puji syukur atas kehadiran Tuhan Yang Maha Esa, atas segala kebesaran dan limpah dan nikmat yang diberikan-Nya, sehingga penulis dapat menyelesaikan laporan studi kasus berjudul “ Laporan Perbandingan QoS TCP dan MPTCP.

Adapun penulisan laporan percobaan ini bertujuan untuk menentukan perbedaan bandwidth dan Packet Loss kedua protocol tersebut. Dalam penulisan laporan ini tentunya saya tidak terlepas dari kesulitan dan masalah pengerjaan nya, akan tetapi dengan Kerjasama dan bimbingan kak Angel masalah-masalah tersebut dapat teratasi.

Penulis sadar sepenuhnya bahwa penulisan laporan ini masih jauh dari kesempurnaan .Oleh karena itu diharapkan kritik dan saran yang membangun demi penyempurnaan selanjutnya. Semoga laporan ini memberikan manfaat kepada pembaca pada umumnya dan penulis pada khususnya . Akhir kata semoga laporan Perbandingan QoS TCP dan MPTCP dapat memperluas wawasan dan informasi pengetahuan kita semua Amin.

Bandung, 15 Juni 2021



Adrian Arman

DAFTAR ISI

| | |
|--|----|
| Kata Pengantar | i |
| Daftar isi | ii |
| BAB I PENDAHULUAN | |
| 1.1 Latar Belakang | 1 |
| 1.2 Identifikasi Rumusan Masalah..... | 1 |
| 1.3 Tujuan dan Manfaat Studi Kasus | 1 |
| BAB II TINJAUAN TEORI | |
| 2.1 Algoritma Penyortiran | 2 |
| BAB III PEMBAHASAN | |
| 3.1 Subnetting Table dan Topologi..... | 3 |
| 3.2 Script MPTCP / TCP..... | 4 |
| 3.3 Output TCP | 6 |
| 3.4 Output MPTCP | 7 |
| 3.5 Pembahasan..... | 9 |

BAB I

PENDAHULUAN

1.1 Latar Belakang

Permasalahan suatu *programmer* pada umumnya adalah ingin mencari suatu algoritma yang cepat dalam menjalankan programnya, dalam jaringan komputer terdapat protocol TCP yakni kepanjangan dari *Transmission Control Protocol* yakni sebagai protocol kendali yang berada pada layer *Transport* pada OSI layer. Setelah itu munculah MPTCP yakni kepanjangan *Multipath Transmission Control Protocol* yakni adalah upaya berkelanjutan dari kelompok kerja Multipath TCP Internet Engineering Task Force (IETF), yang bertujuan memungkinkan koneksi Transmission Control Protocol (TCP) menggunakan beberapa jalur untuk memaksimalkan penggunaan sumber daya dan meningkatkan redundansi, dengan munculnya MPTCP menimbulkan pertanyaan apakah MPTCP memiliki QoS yang lebih baik atau tidak.

1.2 Identifikasi Rumusan Masalah

Dari latar belakang diatas dapat disusun rumusan masalah sebagai berikut :

1. Dari kedua TCP tersebut manakah yang memiliki QoS terbaik?

1.3 Tujuan dan Manfaat Studi Kasus

Tujuan yang ingin dicapai dari Studi Kasus ini antara lain :

1. Mengetahui quality of service antara MPTCP dan TCP

Manfaat yang diperoleh dalam penyusunan laporan adalah sebagai berikut :

1. Mengetahui quality of Service mana yang lebih cepat

BAB II

TINJAUAN TEORI

2.1 TCP (*Transmission Control Protocol*)

Protokol Kendali Transmisi atau Transmission Control Protocol (TCP) adalah suatu protokol yang berada di lapisan transport (baik itu dalam tujuh lapis model referensi OSI atau model DARPA) yang berorientasi sambungan (*connection-oriented*) dan dapat diandalkan (*reliable*). TCP dispesifikasikan dalam RFC 793. Saat Vint Cerf dan Bob Kahn pertama menulis spesifikasi atas protokol ini tahun 1973, internet masih merupakan media yang dilarang untuk keperluan komersial. Di bulan May 1974, IEEE mempublikasikan makalah berjudul "*Protokol sebagai paket interkoneksi jaringan.*" Sebagai penulis makalah tersebut, Cerf dan Kahn menjabarkan sebuah protokol interkoneksi untuk berbagi layanan dengan memanfaatkan pertukaran-paket antara node jaringan yang terhubung. Komponen pengontrol utama atas model ini disebut sebagai "*Program pengontrol transmisi*" yang menggabungkan baik tautan koneksi terorientasi dan layanan datagram antar *host*. Program pengontrol transmisi monolitik ini kemudian dipecah ke dalam arsitektur modular yang terdiri atas *Protokol pengontrol transmisi* pada lapisan koneksi terorientasi dan *Protokol internet* pada lapisan datagram. Model inilah yang kemudian dikenal dengan istilah *TCP/IP*, meski secara formal istilah yang digunakan adalah *Paket protokol internet*.

TCP memiliki karakteristik sebagai berikut:

1. Berorientasi sambungan (*connection-oriented*): Sebelum data dapat ditransmisikan antara dua host, dua proses yang berjalan pada lapisan aplikasi harus melakukan negosiasi untuk membuat sesi koneksi terlebih dahulu. Koneksi TCP ditutup dengan menggunakan proses terminasi koneksi TCP (*TCP connection termination*).
2. *Full-duplex*: Untuk setiap host TCP, koneksi yang terjadi antara dua host terdiri atas dua buah jalur, yakni jalur keluar dan jalur masuk. Dengan menggunakan teknologi lapisan yang lebih rendah yang mendukung *full-duplex*, maka data pun dapat secara simultan diterima dan dikirim. Header TCP berisi nomor urut (*TCP sequence number*) dari data yang ditransmisikan dan sebuah *acknowledgment* dari data yang masuk.
3. Dapat diandalkan (*reliable*): Data yang dikirimkan ke sebuah koneksi TCP akan diurutkan dengan sebuah nomor urut paket dan akan mengharapkan paket *positive acknowledgment* dari penerima. Jika tidak ada paket *Acknowledgment* dari penerima,

maka segmen TCP (protocol data unit dalam protokol TCP) akan ditransmisikan ulang. Pada pihak penerima, segmen-segmen duplikat akan diabaikan dan segmen-segmen yang datang tidak sesuai dengan urutannya akan diletakkan di belakang untuk mengurutkan segmen-segmen TCP. Untuk menjamin integritas setiap segmen TCP, TCP mengimplementasikan penghitungan TCP Checksum. *Byte stream*: TCP melihat data yang dikirimkan dan diterima melalui dua jalur masuk dan jalur keluar TCP sebagai sebuah *byte stream* yang berdekatan (kontigu). Nomor urut TCP dan nomor acknowledgment dalam setiap header TCP didefinisikan juga dalam bentuk byte. Meski demikian, TCP tidak mengetahui batasan pesan-pesan di dalam byte stream TCP tersebut. Untuk melakukannya, hal ini diserahkan kepada protokol lapisan aplikasi (dalam DARPA Reference Model), yang harus menerjemahkan byte stream TCP ke dalam "bahasa" yang ia pahami.

4. Memiliki layanan *flow control*: Untuk mencegah data terlalu banyak dikirimkan pada satu waktu, yang akhirnya membuat "macet" jaringan internetwork IP, TCP mengimplementasikan layanan *flow control* yang dimiliki oleh pihak pengirim yang secara terus menerus memantau dan membatasi jumlah data yang dikirimkan pada satu waktu. Untuk mencegah pihak penerima untuk memperoleh data yang tidak dapat disangganya (*buffer*), TCP juga mengimplementasikan *flow control* dalam pihak penerima, yang mengindikasikan jumlah *buffer* yang masih tersedia dalam pihak penerima.
5. Melakukan segmentasi terhadap data yang datang dari lapisan aplikasi (dalam *DARPA Reference Model*)
6. Mengirimkan paket secara "*one-to-one*": hal ini karena memang TCP harus membuat sebuah sirkuit logis antara dua buah protokol lapisan aplikasi agar saling dapat berkomunikasi. TCP tidak menyediakan layanan pengiriman data secara *one-to-many*.
7. TCP umumnya digunakan ketika protokol lapisan aplikasi membutuhkan layanan transfer data yang bersifat andal, yang layanan tersebut tidak dimiliki oleh protokol lapisan aplikasi tersebut. Contoh dari protokol yang menggunakan TCP adalah HTTP dan FTP.

2.2 MTCP (*Multipath Transmission Control Protocol*)

Multipath TCP (MPTCP) adalah upaya berkelanjutan dari kelompok kerja Multipath TCP Internet Engineering Task Force (IETF), yang bertujuan memungkinkan koneksi Transmission Control Protocol (TCP) menggunakan beberapa jalur untuk memaksimalkan penggunaan sumber daya dan meningkatkan redundansi.

Pada Januari 2013, IETF menerbitkan spesifikasi Multipath sebagai standar Eksperimental di RFC 6824. Pada Maret 2020, spesifikasi tersebut digantikan oleh spesifikasi Multipath TCP v1 di RFC 8684

Redundansi yang ditawarkan oleh Multipath TCP memungkinkan invers multiplexing sumber daya, dan dengan demikian meningkatkan throughput TCP ke jumlah semua saluran tingkat tautan yang tersedia alih-alih menggunakan satu saluran seperti yang dipersyaratkan oleh TCP biasa. Multipath TCP kompatibel dengan TCP biasa.

Multipath TCP sangat berguna dalam konteks jaringan nirkabel menggunakan Wi-Fi dan jaringan seluler adalah kasus penggunaan yang umum. Selain keuntungan dalam throughput dari multiplexing terbalik, tautan dapat ditambahkan atau dijatuhkan saat pengguna bergerak masuk atau keluar dari jangkauan tanpa mengganggu koneksi TCP ujung ke ujung.

Masalah serah terima tautan diselesaikan dengan abstraksi di lapisan transport, tanpa mekanisme khusus di tingkat jaringan atau tautan. Fungsionalitas serah terima kemudian dapat diimplementasikan di titik akhir tanpa memerlukan fungsionalitas khusus di subjaringan - sesuai dengan prinsip ujung ke ujung Internet.

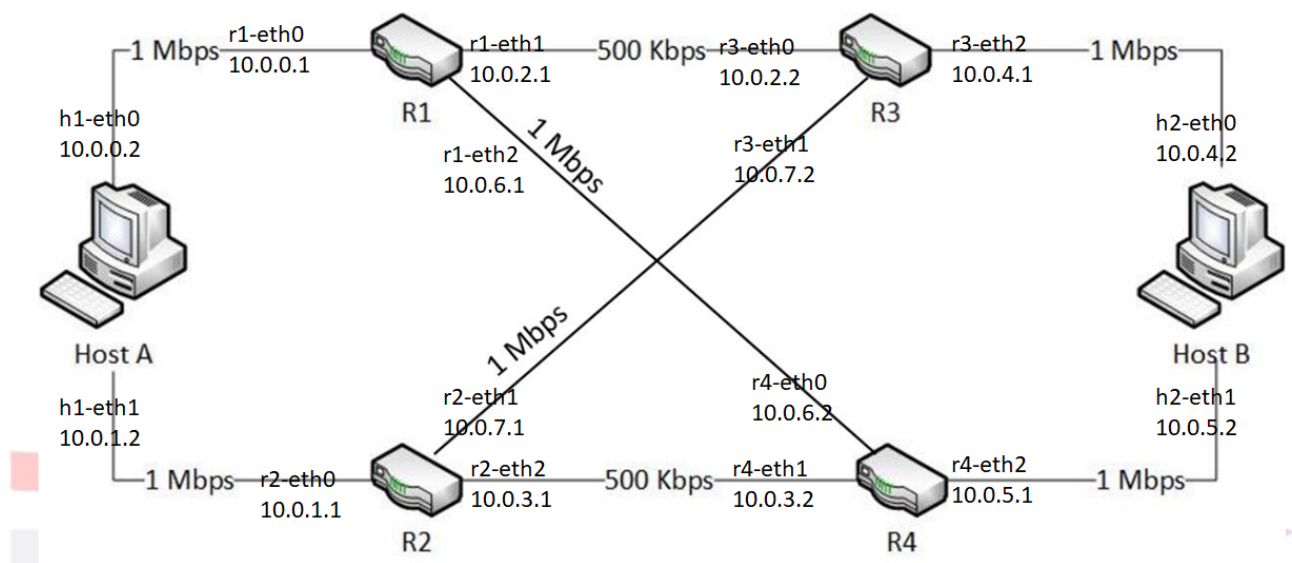
Multipath TCP juga membawa manfaat kinerja di lingkungan pusat data. Berbeda dengan ikatan saluran Ethernet menggunakan agregasi tautan 802.3ad, Multipath TCP dapat menyeimbangkan koneksi TCP tunggal di beberapa antarmuka dan mencapai throughput yang sangat tinggi.

Multipath TCP menyebabkan sejumlah masalah baru. Dari perspektif keamanan jaringan, perutean multipath menyebabkan fragmentasi data lintas jalur yang mengakibatkan firewall dan pemindai malware menjadi tidak efisien ketika mereka hanya melihat lalu lintas satu jalur.

BAB III PEMBAHASAN

3.1 Subnetting Table dan Topologi

| Nama | Host | Net ID | Host Range | Broadcast | Prefix | Subnet Mask |
|-------|------|----------|---------------------|------------|--------|---------------|
| Net A | 254 | 10.0.0.0 | 10.0.0.1-10.0.0.254 | 10.0.0.255 | /24 | 255.255.255.0 |
| Net B | 254 | 10.0.1.0 | 10.0.1.1-10.0.1.254 | 10.0.1.255 | /24 | 255.255.255.0 |
| Net C | 254 | 10.0.2.0 | 10.0.2.1-10.0.2.254 | 10.0.2.255 | /24 | 255.255.255.0 |
| Net D | 254 | 10.0.3.0 | 10.0.3.1-10.0.3.254 | 10.0.3.255 | /24 | 255.255.255.0 |
| Net E | 254 | 10.0.4.0 | 10.0.4.1-10.0.4.254 | 10.0.4.255 | /24 | 255.255.255.0 |
| Net F | 254 | 10.0.5.0 | 10.0.5.1-10.0.5.254 | 10.0.5.255 | /24 | 255.255.255.0 |
| Net G | 254 | 10.0.6.0 | 10.0.6.1-10.0.6.254 | 10.0.6.255 | /24 | 255.255.255.0 |
| Net H | 254 | 10.0.7.0 | 10.0.7.1-10.0.7.254 | 10.0.7.255 | /24 | 255.255.255.0 |



3.2 Script MPTCP / TCP Mininet

```
GNU nano 4.8                                jarkom.py
#!/usr/bin/env python
from mininet.net import Mininet
from mininet.cli import CLI
from mininet.link import Link, TLink, Intf
from subprocess import Popen, PIPE
from mininet.log import setLogLevel

if '__main__' == __name__:
    setLogLevel('info')
    net = Mininet(link=TLink)
    key = "net.mptcp.mptcp_enabled"
    value = 1
    p = Popen("sysctl -w %s=%s" % (key, value), shell=True, stdout=PIPE, stderr=PIPE)
    stdout, stderr = p.communicate()
    print("stdout=", stdout, "stderr=", stderr)

    h1 = net.addHost('h1')
    h2 = net.addHost('h2')
```

value = 1 MPTCP / value = 0 TCP

```
GNU nano 4.8                                jarkom.py
h1 = net.addHost('h1')
h2 = net.addHost('h2')
r1 = net.addHost('r1')
r2 = net.addHost('r2')
r3 = net.addHost('r3')
r4 = net.addHost('r4')

bwlink={ 'bw':1}
bwlink2={ 'bw':0.5}

net.addLink(r1,h1,intfName1='r1-eth0',intfName2='h1-eth0',cls=TLink, **bwlink) #r1-eth0 h1-eth0
net.addLink(r1,r3,intfName1='r1-eth1',intfName2='r3-eth0',cls=TLink, **bwlink2) #r1-eth1 r2-eth0
net.addLink(r1,r4,intfName1='r1-eth2',intfName2='r4-eth0',cls=TLink, **bwlink) #r1-eth2 r2-eth0

net.addLink(h2,r3,intfName1='h2-eth0',intfName2='r3-eth2',cls=TLink, **bwlink) #r3-eth0 h1-eth1
net.addLink(r3,r2,intfName1='r3-eth1',intfName2='r2-eth1',cls=TLink, **bwlink) #r3-eth2 r2-eth1

net.addLink(r2,r4,intfName1='r2-eth2',intfName2='r4-eth1',cls=TLink, **bwlink2)
net.addLink(r2,h1,intfName1='r2-eth0',intfName2='h1-eth1',cls=TLink, **bwlink)
net.addLink(r4,h2,intfName1='r4-eth2',intfName2='h2-eth1',cls=TLink, **bwlink) #r4-eth2 h2-eth1
```

```
GNU nano 4.8                                jarkom.py
net.build()

h1.cmd("ifconfig h1-eth0 0")
h1.cmd("ifconfig h1-eth1 0")

h2.cmd("ifconfig h2-eth0 0")
h2.cmd("ifconfig h2-eth1 0")

r1.cmd("ifconfig r1-eth0 0")
r1.cmd("ifconfig r1-eth1 0")
r1.cmd("ifconfig r1-eth2 0")

r2.cmd("ifconfig r2-eth0 0")
r2.cmd("ifconfig r2-eth1 0")
r2.cmd("ifconfig r2-eth2 0")

r3.cmd("ifconfig r3-eth0 0")
r3.cmd("ifconfig r3-eth1 0")
r3.cmd("ifconfig r3-eth2 0")
```

```
adrian@adrian-VirtualBox: ~/Downloads
GNU nano 4.8 jarkom.py

r4.cmd("ifconfig r4-eth0 0")
r4.cmd("ifconfig r4-eth1 0")
r4.cmd("ifconfig r4-eth2 0")

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")

#tabel routing
h1.cmd("ifconfig h1-eth0 10.0.0.2 netmask 255.255.255.0")
h1.cmd("ifconfig h1-eth1 10.0.1.2 netmask 255.255.255.0")

h2.cmd("ifconfig h2-eth0 10.0.4.2 netmask 255.255.255.0")
h2.cmd("ifconfig h2-eth1 10.0.5.2 netmask 255.255.255.0")

r1.cmd("ifconfig r1-eth0 10.0.0.1 netmask 255.255.255.0")
r1.cmd("ifconfig r1-eth1 10.0.2.1 netmask 255.255.255.0")
r1.cmd("ifconfig r1-eth2 10.0.6.1 netmask 255.255.255.0")

^G Get Help  ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos    M-U Undo
^X Exit      ^R Read File  ^_ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line  M-E Redo
```

```
GNU nano 4.8 jarkom.py

#tabel routing
h1.cmd("ifconfig h1-eth0 10.0.0.2 netmask 255.255.255.0")
h1.cmd("ifconfig h1-eth1 10.0.1.2 netmask 255.255.255.0")

h2.cmd("ifconfig h2-eth0 10.0.4.2 netmask 255.255.255.0")
h2.cmd("ifconfig h2-eth1 10.0.5.2 netmask 255.255.255.0")

r1.cmd("ifconfig r1-eth0 10.0.0.1 netmask 255.255.255.0")
r1.cmd("ifconfig r1-eth1 10.0.2.1 netmask 255.255.255.0")
r1.cmd("ifconfig r1-eth2 10.0.6.1 netmask 255.255.255.0")

r2.cmd("ifconfig r2-eth0 10.0.1.1 netmask 255.255.255.0")
r2.cmd("ifconfig r2-eth1 10.0.7.1 netmask 255.255.255.0")
r2.cmd("ifconfig r2-eth2 10.0.3.1 netmask 255.255.255.0")

r3.cmd("ifconfig r3-eth0 10.0.2.2 netmask 255.255.255.0")
r3.cmd("ifconfig r3-eth1 10.0.7.2 netmask 255.255.255.0")
r3.cmd("ifconfig r3-eth2 10.0.4.1 netmask 255.255.255.0")

r4.cmd("ifconfig r4-eth0 10.0.6.2 netmask 255.255.255.0")

^G Get Help  ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos    M-U Undo
^X Exit      ^R Read File  ^_ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line  M-E Redo
```

```
GNU nano 4.8 jarkom.py

r4.cmd("ifconfig r4-eth0 10.0.6.2 netmask 255.255.255.0")
r4.cmd("ifconfig r4-eth1 10.0.3.2 netmask 255.255.255.0")
r4.cmd("ifconfig r4-eth2 10.0.5.1 netmask 255.255.255.0")

#routing statis
h1.cmd("ip rule add from 10.0.0.2 table 1")
h1.cmd("ip rule add from 10.0.1.2 table 2")
h1.cmd("ip route add 10.0.0.0/24 dev h1-eth0 scope link table 1")
h1.cmd("ip route add default via 10.0.0.1 dev h1-eth0 table 1")
h1.cmd("ip route add 10.0.1.0/24 dev h1-eth1 scope link table 2")
h1.cmd("ip route add default via 10.0.1.1 dev h1-eth1 table 2")
h1.cmd("ip route add default scope global nexthop via 10.0.0.1 dev h1-eth0")

h2.cmd("ip route add from 10.0.4.2 table 1")
h2.cmd("ip route add from 10.0.5.2 table 2")
h2.cmd("ip route add 10.0.4.0/24 dev h2-eth0 scope link table 1")
h2.cmd("ip route add default via 10.0.4.1 dev h2-eth0 table 1")
h2.cmd("ip route add 10.0.5.0/24 dev h2-eth1 scope link table 2")
h2.cmd("ip route add default via 10.0.5.1 dev h2-eth1 table 2")
h2.cmd("ip route add default scope global nexthop via 10.0.4.1 dev h2-eth0")

^G Get Help  ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos    M-U Undo
^X Exit      ^R Read File  ^_ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line  M-E Redo
```

```

GNU nano 4.8                                jarkom.py
r2.cmd("route add -net 10.0.0.0/24 gw 10.0.7.2")
r2.cmd("route add -net 10.0.5.0/24 gw 10.0.3.2")
r2.cmd("route add -net 10.0.4.0/24 gw 10.0.7.2")
r2.cmd("route add -net 10.0.2.0/24 gw 10.0.7.2")

r3.cmd("route add -net 10.0.5.0/24 gw 10.0.4.2")
r3.cmd("route add -net 10.0.6.0/24 gw 10.0.2.1")
r3.cmd("route add -net 10.0.0.0/24 gw 10.0.2.1")
r3.cmd("route add -net 10.0.1.0/24 gw 10.0.7.1")
r3.cmd("route add -net 10.0.3.0/24 gw 10.0.7.1")

r4.cmd("route add -net 10.0.7.0/24 gw 10.0.3.1")
r4.cmd("route add -net 10.0.4.0/24 gw 10.0.5.1")
r4.cmd("route add -net 10.0.1.0/24 gw 10.0.3.1")
r4.cmd("route add -net 10.0.0.0/24 gw 10.0.6.1")
r4.cmd("route add -net 10.0.2.0/24 gw 10.0.6.1")

CLI(net)
net.stop()

```

3.3 Output TCP

Test Pingall

```

root@adnan-virtualbox:/home/adnan/downloads
64 bytes from 10.0.6.2: icmp_seq=5 ttl=64 time=0.123 ms
^C
--- 10.0.6.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4091ms
rtt min/avg/max/mdev = 0.116/0.121/0.128/0.005 ms
mininet> r3 ping r4
PING 10.0.6.2 (10.0.6.2) 56(84) bytes of data:
64 bytes from 10.0.6.2: icmp_seq=1 ttl=63 time=0.059 ms
64 bytes from 10.0.6.2: icmp_seq=2 ttl=63 time=0.122 ms
64 bytes from 10.0.6.2: icmp_seq=3 ttl=63 time=0.127 ms
^C
--- 10.0.6.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2050ms
rtt min/avg/max/mdev = 0.059/0.102/0.127/0.030 ms
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>

```

Trace Route h1 – h2 dan h2 – h1

```

mininet> h1 traceroute h2
traceroute to 10.0.4.2 (10.0.4.2), 30 hops max, 60 byte packets
 1 10.0.0.1 (10.0.0.1) 0.052 ms 0.008 ms 0.008 ms
 2 10.0.2.2 (10.0.2.2) 0.023 ms 0.013 ms 0.011 ms
 3 10.0.4.2 (10.0.4.2) 0.026 ms 0.016 ms 0.015 ms
mininet> h2 traceroute h1
traceroute to 10.0.0.2 (10.0.0.2), 30 hops max, 60 byte packets
 1 10.0.4.1 (10.0.4.1) 0.051 ms 0.008 ms 0.006 ms
 2 10.0.2.1 (10.0.2.1) 0.021 ms 0.012 ms 0.010 ms
 3 10.0.0.2 (10.0.0.2) 0.027 ms 0.016 ms 0.015 ms
mininet>

```

Iperf h1 ke h2 dengan h2 sebagai server

```
2 packets transmitted, 2 received, 0% packet loss, time 1010ms
rtt min/avg/max/mdev = 0.074/0.101/0.120/0.027 ms
mininet>
Node: h2
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
root@adrian-VirtualBox:/home/adrian/Downloads# iperf -s -i1
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
--- 10.0.0.0 port 5001 connected with 10.0.0.2 port 38304
[ ID] Interval Transfer Bandwidth
[ 0] 0.0-1.0 sec 120 KBytes 365 Kbits/sec
[ 1] 1.0-2.0 sec 120 KBytes 365 Kbits/sec
[ 2] 2.0-3.0 sec 113 KBytes 327 Kbits/sec
[ 3] 3.0-4.0 sec 120 KBytes 365 Kbits/sec
[ 4] 4.0-5.0 sec 113 KBytes 327 Kbits/sec
[ 5] 5.0-6.0 sec 120 KBytes 365 Kbits/sec
[ 6] 6.0-7.0 sec 113 KBytes 327 Kbits/sec
[ 7] 7.0-8.0 sec 120 KBytes 365 Kbits/sec
[ 8] 8.0-9.0 sec 116 KBytes 350 Kbits/sec
[ 9] 9.0-10.0 sec 117 KBytes 361 Kbits/sec
[10] 10.0-11.0 sec 116 KBytes 350 Kbits/sec
[11] 0.0-11.0 sec 1.36 MBytes 550 Kbits/sec
h1 ->
r1 ->
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> xterm h1 h2
Node: h1
root@adrian-VirtualBox:/home/adrian/Downloads# iperf -c 10.0.0.2 -t10 -i1
Client connecting to 10.0.0.2, TCP port 5001
TCP window size: 128 KByte (default)
[ 0] local 10.0.0.2 port 38304 connected with 10.0.0.2 port 5001
[ ID] Interval Transfer Bandwidth
[ 0] 0.0-1.0 sec 311 KBytes 2.95 Mbits/sec
[ 1] 1.0-2.0 sec 127 KBytes 1.04 Mbits/sec
[ 2] 2.0-3.0 sec 106 KBytes 869 Kbits/sec
[ 3] 3.0-4.0 sec 127 KBytes 1.04 Mbits/sec
[ 4] 4.0-5.0 sec 127 KBytes 1.04 Mbits/sec
[ 5] 5.0-6.0 sec 106 KBytes 869 Kbits/sec
[ 6] 6.0-7.0 sec 127 KBytes 1.04 Mbits/sec
[ 7] 7.0-8.0 sec 106 KBytes 869 Kbits/sec
[ 8] 8.0-9.0 sec 127 KBytes 1.04 Mbits/sec
[ 9] 9.0-10.0 sec 127 KBytes 1.04 Mbits/sec
[10] 0.0-10.0 sec 1.36 MBytes 1.11 Mbits/sec
root@adrian-VirtualBox:/home/adrian/Downloads#
```

Packet Loss = 0%, avg Bandwidth = 1,137Mbits/sec

Delay = 2050ms

3.4 Output MPTCP

Test Ping

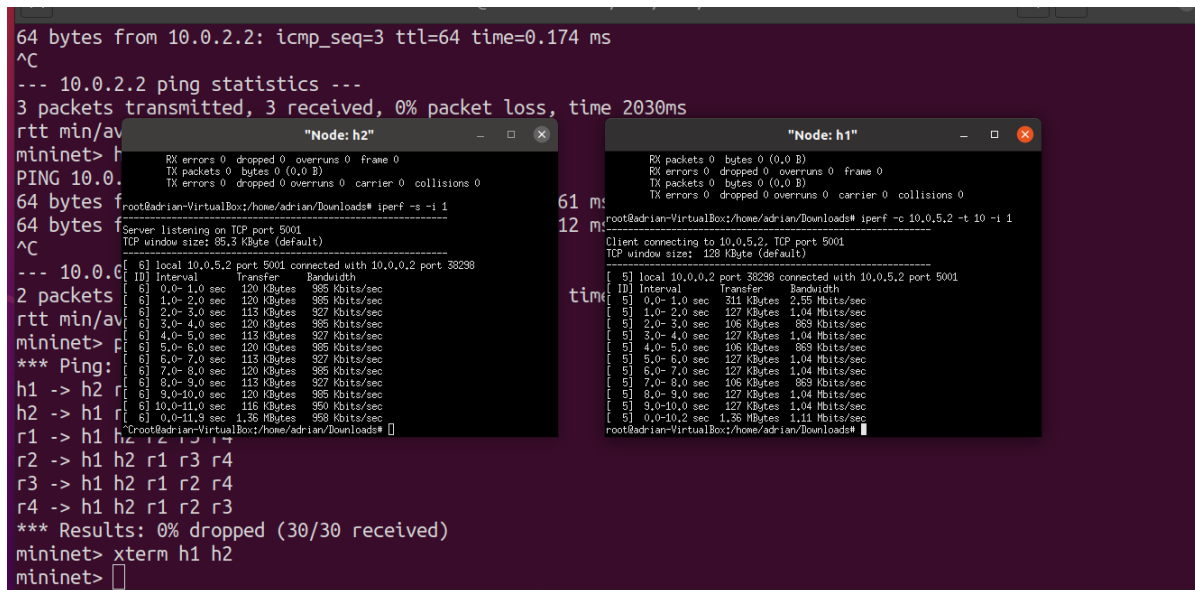
```
adrian@adrian-VirtualBox: ~/Downloads
adrian@adrian-VirtualBox:~/Downloads$ sudo python3 jarkom.py
stdout= b'' stderr= b'sysctl: cannot stat /proc/sys/net/mptcp/mptcp_enabled: No such file or directory\n'
(1.00Mbit) (1.00Mbit) (0.50Mbit) (0.50Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (0.50Mbit) (0.50Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) (1.00Mbit) *** Configuring hosts
h1 h2 r1 r2 r3 r4
*** Starting CLI:
mininet> r3 ping r4
PING 10.0.6.2 (10.0.6.2) 56(84) bytes of data.
64 bytes from 10.0.6.2: icmp_seq=1 ttl=63 time=0.124 ms
64 bytes from 10.0.6.2: icmp_seq=2 ttl=63 time=0.110 ms
64 bytes from 10.0.6.2: icmp_seq=3 ttl=63 time=0.113 ms
64 bytes from 10.0.6.2: icmp_seq=4 ttl=63 time=0.122 ms
64 bytes from 10.0.6.2: icmp_seq=5 ttl=63 time=0.115 ms
64 bytes from 10.0.6.2: icmp_seq=6 ttl=63 time=0.117 ms
64 bytes from 10.0.6.2: icmp_seq=7 ttl=63 time=0.116 ms
64 bytes from 10.0.6.2: icmp_seq=8 ttl=63 time=0.109 ms
64 bytes from 10.0.6.2: icmp_seq=9 ttl=63 time=0.110 ms
^C
--- 10.0.6.2 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8182ms
rtt min/avg/max/mdev = 0.109/0.115/0.124/0.005 ms
mininet>
```

Trace Route h1 – h2 dan h2 – h1

```
mininet> h1 traceroute h2
traceroute to 10.0.4.2 (10.0.4.2), 30 hops max, 60 byte packets
 1 10.0.0.1 (10.0.0.1) 0.052 ms 0.008 ms 0.008 ms
 2 10.0.2.2 (10.0.2.2) 0.023 ms 0.013 ms 0.011 ms
 3 10.0.4.2 (10.0.4.2) 0.026 ms 0.016 ms 0.015 ms
mininet> h2 traceroute h1
traceroute to 10.0.0.2 (10.0.0.2), 30 hops max, 60 byte packets
 1 10.0.4.1 (10.0.4.1) 0.051 ms 0.008 ms 0.006 ms
 2 10.0.2.1 (10.0.2.1) 0.021 ms 0.012 ms 0.010 ms
 3 10.0.0.2 (10.0.0.2) 0.027 ms 0.016 ms 0.015 ms
mininet>
```

Iperf h1 ke h2 dengan h2 sebagai server

```
64 bytes from 10.0.2.2: icmp_seq=3 ttl=64 time=0.174 ms
^C
--- 10.0.2.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2030ms
rtt min/avg/max = 0.174/0.174/0.174 ms
mininet> h1
PING 10.0.2.2: 64 bytes of data:
64 bytes from 10.0.2.2: icmp_seq=1 ttl=64 time=0.174 ms
64 bytes from 10.0.2.2: icmp_seq=2 ttl=64 time=0.174 ms
64 bytes from 10.0.2.2: icmp_seq=3 ttl=64 time=0.174 ms
^C
--- 10.0.2.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2030ms
rtt min/avg/max = 0.174/0.174/0.174 ms
mininet> h2
*** Ping: 61 ms
h1 -> h2 r
h2 -> h1 r
r1 -> h1 h2
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> xterm h1 h2
mininet>
```



Packet Loss = 0% dropped Avg Bandwidth = 1,137Mbits/sec

Delay = 8182ms

3.5 Pembahasan

Untuk perbandingan antara TCP dengan MPTCP bisa kita lihat dari tiga metric QoS yaitu bandwidth/throughput, delay, dan packet loss.

- Bandwidth/Throughput Untuk bandwidth, TCP dan MPTCP memiliki rata-rata bandwidth yang sama.
- Delay Untuk delay, MTCP memiliki rata-rata delay lebih tinggi daripada TCP.
- Packet Loss Untuk packet loss, kedua protocol tidak memiliki packet loss ketika pengiriman data atau tidak ada data yang hilang saat pengiriman.
- Kesimpulan Dari analisis yang telah saya lakukan, bisa disimpulkan bahwa protocol TCP lebih baik untuk pengiriman data daripada protocol MPTCP karena walaypun delay protokol TCP lebih tinggi daripada protokol MPTCP tetapi bandwidth dari protokol TCP rata-rata dua kali lebih besar daripada MPTC

