# ANALISIS *MULTIPATH TCP* DENGAN *OPENFLOW* BERBASIS *SOFTWARE DEFINED NETWORK* (SDN)

# *ANALYSIS MULTIPATH TCP USING OPENFLOW-BASED SOFTWARE DEFINED NETWORK (SDN)*

**Veby Riza[1], Sofia Naning, S.T., M.T.[2], Ridha Muldina, S.T., M.T.[3]**
[1,2,3]Prodi S1 Teknik Telekomunikasi, Fakultas Teknik Elektro, Universitas Telkom
[1]**vebbyriza@gmail.com**, [2]**sofiananing16@gmail.com**, [3]**ridhanegara@gmail.com**

**Abstrak**

*Data center* merupakan pusat lalu lintas data yang memiliki topologi jaringan berskala besar. Desain topologi jaringan *data center* harus memenuhi kriteria *high bandwidth*, *low latency*, *communication balance*. *fat-tree* adalah salah satu topologi yang sering digunakan pada *data center*. *fat-tree* diadopsi secara luas dalam membangun jaringan *data center*. *fat-tree* memiliki karakteristik *full bisection bandwidth* yang berperan penting untuk menghindari terjadinya *congestion*.

Dalam tugas akhir ini akan dilakukan penerapan konsep *multipath TCP* berbasis *Software defined network* dan menganalisa kinerja *multipath TCP* dalam mengatasi measalah pada jalur yang mengalami masalah. Dalam hal ini *multipath TCP* sangat dibutuhkan karena ketika initial *handshake*, pengirim dan penerima saling memberikan informasi tentang IP *address*. Kemudian *MTCP* akan membangun beberapa *subflow*s dari sebagian atau seluruh IP pengirim ke sebagian atau ke seluruh IP penerima, sehingga dapat dapat mengatur berapa besar *bandwidth* yang akan dialirkan pada *path* tersebut.

Hasil pengujian yang didapatkan adalah berupa parameter *one-way delay* yang telah memenuhi standar ITU-T G.114 yaitu di bawah 400 milidetik atau 150 milidetik, *retransmission* dan RTT yang lebih baik dari *single path* , serta nilai *throughput* 50% lebih besar dari komunikasi *single path* dengan rata-rata *bandwith* 43.521 Mbit/s sementara *single path* mendapat nilai rata-rata 27.247 Mbit/s. Penelitian ini juga menguji parameter *Quality of Service* dengan menambahkan *background traffic* pada sekenario trafik 40%-60% dan 20%-80% dimana *background traffic* mempengaruhi *bandwith link* dengan memperbesar *bandwith link 1* dan mengurangi bandwith link 2 karena *multipath* TCP telah mengatur trafik 40%-60% dan 20%-80% pada *link*.

Kata Kunci : *Software* Defined Network, *Multipath TCP*, *OpenFlow*, *Data center*

***Abstract***

Data center is the center of data traffic that has a large-scale network topology. The design of the data center network topology must meet the criteria of high bandwidth, low latency, communication balance. fat-tree is one of the topology that is often used in data center. fat-tree is widely adopted in building data center networks. fat-tree has the characteristics of full bisection bandwidth that plays an important role to avoid congestion.

In this final project will implement the concept of TCP-based multipath based Software defined network and analyze TCP multipath performance in overcoming the problem in the path of problem. In this case TCP multipath is needed because when the initial handshake, the sender and receiver give each other information about the IP address. Then MTCP will build some subflows from some or all sending IPs to some or all of the receiving IP, so it can be able to set how much bandwidth will be streamed on the path..

The test results obtained are a one-way delay parameter that meets ITU-T G.114 standard under 400 milliseconds or 150 milliseconds, retransmission and RTT better than single path, and throughput value 50% greater than single communication path with an average bandwidth of 43,521 Mbit / s while a single path gets an average of 27,247 Mbit / s. This study also tested the parameters of Quality of Service by adding background traffic on the traffic scenario of 40% -60% and 20% -80% where the background traffic affects bandwith links by increasing link bandwidth 1 and reducing bandwidth link 2 because multipath TCP has set 40% -60% and 20% -80% on links..

*Keywords: Software Defined Network, Multipath TCP, OpenFlow, Data Center*

## 1. Pendahuluan

*Data center* merupakan pusat lalu lintas data yang memiliki topologi jaringan berskala besar. Pada infrastruktur jaringan *data center* tradisional, setiap perangkat harus dikonfigurasi satu persatu ke masing masing perangkat. Hal ini menambah kompleksitas dari jaringan saat topologi jaringan melibatkan perangkat yang lebih banyak karena membutuhkan *reconfigure* dan *re-policy* ke masing-masing perangkat. Oleh karena itu, *Stanford University* melalui *project cleanslate* menciptakan protokol *openflow* untuk mengaplikasikan paradigma *Software Defined Networking (SDN)* pada jaringan sehingga konfigurasi jaringan menjadi lebih sederhana. Desain topologi jaringan *data center* pada SDN harus memenuhi kriteria *high bandwidth, low latency, communication balance*[1]. *fat-tree* adalah salah satu topologi yang sering digunakan pada *data center*.

*Multipath TCP* merupakan metode peningkatan *throughput* dengan menggabungkan dua buah *link* dalam satu pengiriman data. Dimana dibuat dua buah *subflow* pada kedua *path* tersebut sehingga meningkatkan *bandwith* pada jaringan. Dalam metode *load balancing multipath TCP* sangat diunggulkan karena ketika *initial handshake*, pengirim dan penerima saling memberikan informasi tentang IP *address*. Kemudian *MTCP* akan membangun beberapa *subflow*s dari sebagian atau seluruh IP pengirim ke sebagian atau ke seluruh IP penerima, sehingga dapat dapat mengatur berapa besar *bandwidth* yang akan dialirkan pada *path* tersebut. Metode ini memungkinkan *administrator* jaringan pada *data center* untuk memiliki kontrol atas lalu lintas jaringan dan karena itu memiliki potensi besar untuk lebih meningkatkan kinerja jaringan dalam hal efisiensi penggunaan sumber informasi jaringan.

Pada tugas akhir ini, dilakukan penerapan layanan *multipath TCP* (*MTCP*) karena pada metode ini *administrator* jaringan pada *data center* memiliki kontrol atas lalu lintas jaringan untuk mengatur besar trafik pada dua buah *subflow* berdasarkan beban *link* agar dapat membuat *load balancing* yang baik. Penerapan ini dilakukan pada jaringan *data center* berbasis *software defined network* dan menganalisa perbandingan *Quality of Service* antara *multipath TCP* dan *singlepath TCP* dengan metode *spanning tree protocol* (STP) dari jaringan *data center* tersebut. analisis dilakukan dengan menggunakan emulasi jaringan yang terdiri dari *host*, *switch* yang saling terhubung membentuk jaringan *data center* dan perangkat *control plane* sebagai pengendali sebuah jaringan.

## 2. Dasar Teori

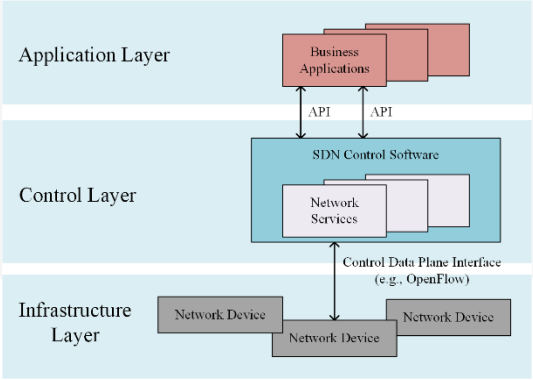
### 2.1 *Data center*

Suatu fasilitas yang digunakan untuk menempatkan sistem komputer dan komponen- komponen terkaitnya, seperti sistem telekomunikasi dan penyimpanan data. Fasilitas ini biasanya mencakup juga catu daya *redundan*, koneksi data *redundan*, pengontrol lingkungan, pencegah bahaya kebakaran, serta piranti keamanan fisik.
Terdapat 2 jenis *data center*, yaitu *data center* private dan *data center* public. *Data*

- *center private*, artinya hanya diperuntukkan bagi lembaga atau organisasi yangmendirikannya/memilikinya. *Data center* semacam ini tidak melayani atau tidak dapat diakses oleh organisasi lain atau masyarakat umum.
- *Data center public*, di mana *data center* tersebut dapat diakses dan dimanfaatkan oleh masyarakat atau organisasi lain. Umumnya lembaga riset memiliki *data center* jenis ini.

### 2.2 *Software* Defined Networking (SDN)

SDN[8] merupakan arsitektur jaringan yang muncul dengan kontrol jaringan dipisahkan dari *forwarding plane* dan secara langsung dapat diprogram. Dapat dilihat pada Gambar 2.6 bahwa intelegensi jaringan secara logikal terpusat di SDN *controller* yang dapat melihat topologi jaringan *global*. *Operator* dan *administrator* jaringan dapat mengkonfigurasi secara terprogram jaringan SDN lebih baik daripada harus melakukan konfigurasi ribuan baris pada ribuan perangkat pada jaringan konvensional.




Gambar 2. Arsitektur *software defined network*[8]

Karakteristik SDN[16] sebagai berikut:

a. *Direct Programmable* : kontrol jaringan dapat diprogram secara langsung karena terpisah dari fungsi *forwarding*.
b. *Agile* : kontrol abstrak dari *forwarding* memungkinkan *administrator* secara dinamis mengatur trafik jaringan untuk memenuhi kebutuhan yang terus berubah.
c. *Programmatically configured* : SDN memungkinkan manager jaringan mengkonfigurasi, mengelola, mengamankan, dan mengoptimalkan sumber daya jaringan dengan sangat cepat secara dinamis, program SDN yang dinamis dimana manager jaringan dapat menulis program sendiri karena tidak bergantung pada perangkat lunak bersifat *proprietary*.
d. *Open standards-based and vendor-neutral* : SDN diimplementasikan secara terbuka.
   Dari penjelasan tentang arsitektur SDN, jelas bahwa SDN bersifat *interoperability* karena tidak tergantung pada perangkat *propriretary* dan bersifat fleksibel.

### 2.3 *Ryu Controller*

*Ryu* merupakan sebuah jenis kontroler yang banyak digunakan pada SDN. *Ryu* mendukung protokol *OpenFlow*. Selain itu, *Ryu* mendukung jenis *OpenFlow* 1.0, *OpenFlow* 1.2, *OpenFlow* 1.3 dan *OpenFlow* 1.4. *Ryu* secara resmi mendukung *Window*s, MAC OS, dan Linux. Dalam pengkodean nya, sebuah program yang ada pada kontroler *Ryu* ditulis dengan menggunakan bahasa pemrograman *Python.* Kontroler *Ryu* yang digunakan dalam tugas akhir ini dapat memanfaatkan fitur yang dimiliki oleh *OpenFlow* 1.3[17].

### 2.4 *Dijkstra Shortest Path Tree*

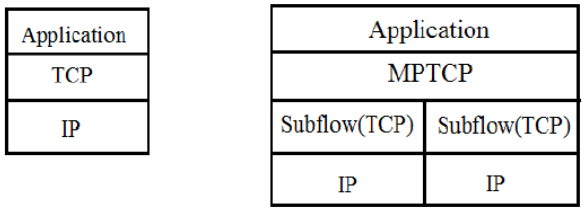
Algoritma *Dijkstra*[19] digunakan untuk menentukan pemilihan jalur terbaik berdasarkan *cost* dari sebuah *graph*. Dalam jaringan, *Dijkstra* digunakan untuk memilih jalur terbaik dari sumber ke tujuan. *Dijkstra* SPT merupakan algoritma yang sekarang digunakan oleh beberapa protokol *routing*, tidak hanya digunakan untuk transmisi *unicast* namun juga digunakan dalam komunikasi *multicast* yaitu dengan menghitung jalur terbaik dari sumber multicast ke semua tujuan yang tergabung dalam grup. *Dijkstra* SPT menggunakan beban *link* sebagai parameter untuk penentuan jalur terbaik[20].

```
Dijkstra's Algorithm
Input : Graph G = (V,E)
(∀x ≠ s) dist[x] = +∞   //initialize dist[]
dist[s] = 0
S = Ø
Q = V   //Keyed by dist[]
while Q ≠ Ø do
    u = extract_min (Q)
    S = S U {u}
    foreach vertex v ϵ Adj(u) do
        dist[v] = min(dist[v],dist[u] + w(u,v))
            //"Relax" operation
```

Gambar 3. Algoritma *Dijkstra* SPT[21]

### 2.5 *Multipath TCP*

Pada *multipath TCP*, pengirim maupun penerima memiliki beberapa *interface*. Ketika initial *handshake*, pengirim dan penerima saling memberikan informasi tentang IP *address*. Kemudian *MTCP* akan membangun beberapa *subflow*s dari sebagian atau seluruh IP pengirim ke sebagian atau ke seluruh IP penerima. Pada setiap *subflow* akan digunakan *window based congestion control* dan komunikasi yang *reliable* akan digunakan pada *subflow-subflow* tersebut. Ketika sebuah paket rusak atau hilang pada salah satu *subflow*, paket tersebut dapat dikirim ke *subflow* yang lain. Ketika suatu paket dibuang atau menerima ECN, ukuran *window* akan dikurangi. Pengurangan ukuran *window* ini harus sesuai dengan jumlah seluruh *window* pada pengirim, bukan hanya pada satu *window*. Ketika terjadi kesalahan pada salah satu *path* yang dikarenakan oleh instabilitas pada jaringan, trafik pada *path* tersebut tidak langsung dipindahkan ke jalur yang lain, tetapi *window* pada jalur yang lain akan meningkat secara bertahap terhadap RTT.




Gambar 4. Perbandingan *TCP* Stack *Protocol* dan *MTCP* Stack *Protocol*[7]

**2.6 Parameter *Quality of Service***

Pada penelitian tugas akhir ini, terdapat beberapa parameter yang telah diukur, parameter-parameter tersebut adalah sebagai berikut:

**2.6.1 *Round trip time***

*Round-trip Time* adalah banyaknya waktu yang dibutuhkan oleh suatu paket untuk melakukan perjalanan dari pengirim ke penerima dan kembali lagi kepada pengirim.

**2.6.2 *Throughput***

Besar data yang diproses yang dapat melewati jaringan disebut *throughput*. Parameter ini dapat digunakan untuk melakukan pengujian terhadap kualitas suatu *link*. Satuan waktu yang biasa digunakan pada *throughput* adalah *bit per second* (bps).

***2.6.3 Retransmission***

Retransmisi adalah parameter yang menunjukkan banyaknya paket yang dikirimkan ulang karena telah terjadi sesuatu, salah satunya adalah karena terjadi *congestion*.

**2.6.4 *Delay***

*Delay* adalah waktu yang dibutuhkan oleh sebuah paket data terhitung dari saat pengiriman oleh *transmitter* sampai saat diterima oleh *receiver*. *Delay* yang diukur yaitu *one-way delay*.

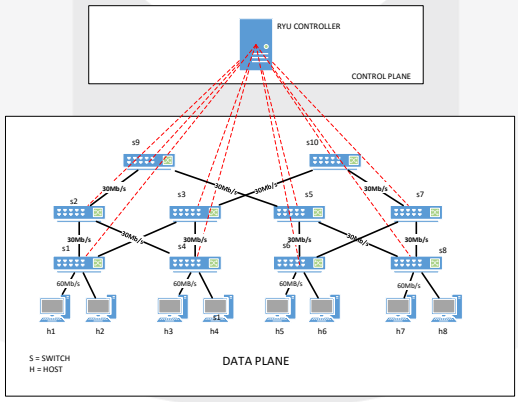
**2.7 *Iperf dan Wireshark***

*Iperf*[23] merupakan alat pengujian jaringan yang dapat menggenerate *data stream TCP* atau *udp*. *Tool* ini digunakan untuk mengukur *jitter*, *throughput*, dan *packet loss*. *Wireshark*[24] adalah sebuah *network packet analyzer*. *Network* packet analyzer akan mencoba menangkap paket – paket yang ada dalam jaringan dan menampilkan informasi yang didapat tentang paket tersebut. Kita bisa mengasumsikan sebuah *network packet analyzer* sebagai alat untuk mengukur keadaan yang terjadi dalam jaringan kabel, seperti *voltmeter* yang digunakan untuk mengetahui keadaan di dalam kabel listrik. *Wireshark* digunakan sebagai alat untuk memecahkan masalah yang ada dalam jaringan, kemanan jaringan, atau mempelajari protokol jaringan yang ada.

## 3. Perancangan Sistem dan Hasil Analisis

### 3.1 Topologi Jaringan

Gambar berikut adalah desain sistem yang akan dibuat dengan menggunakan satu *Ryu controller*, 10 *OpenFlow switch* dan 8 *host*.

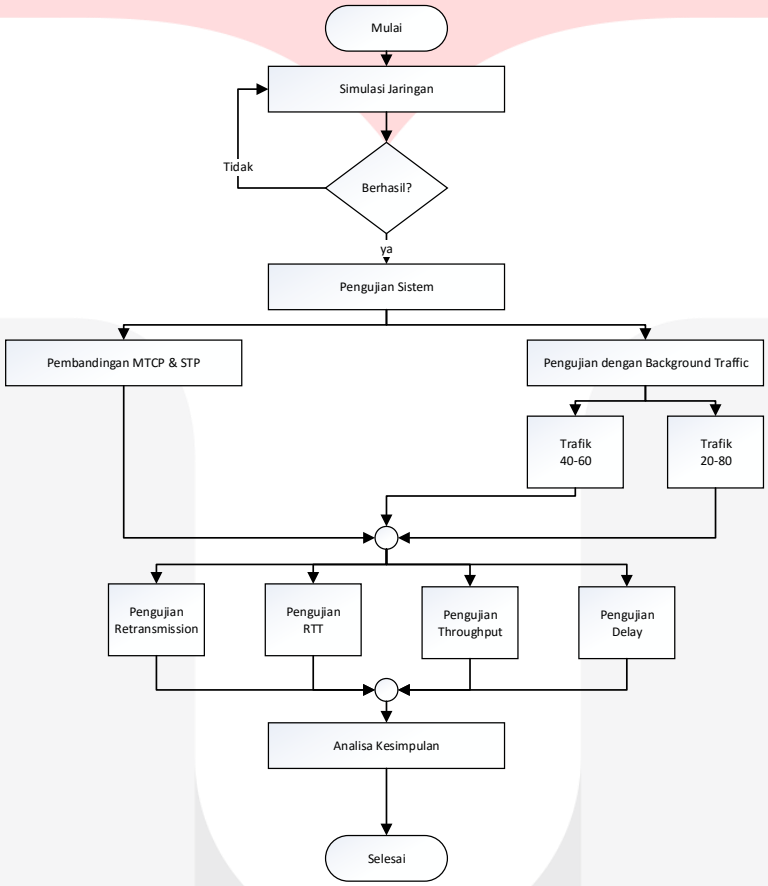



Gambar 5. Topologi Jaringan

Desain system pada Tuga Akhir diimplementasikan dengan *multipath TCP* yang terdiri dari dua buah PC. PC pertama terdiri darari *controller* dan *OpenFlow* dimana nantinya *multipath TCP* dan *ryu controller* akan dikonfigurasi pada pc pertama. sedangkan pada PC kedua terdiri dari *mininet* dengan topologi *fat-tree*. Topologi *fat-tree* dalam *mininet* tersebut berisi 10 buah *switch* yang saling terhubung serta 8 buah host yang akan saling bertukar informasi. Pada topologi tersebut nantinya akan dikonfigurasi *Multipath TCP* dari *controller* melalui *protocol OpenFlow* dan dianalisa performansinya.

### 3.2 Skenario Pengujian

Skenario pengujian pada tugas akhir ini yaitu dengan membandingkan *multipath TCP* dan *single path TCP* serta mengatur trafik yang dialirkan pada *path* yang berbeda dengan parameter yang diukur berupa *Retransmission*, *throughput*, *Delay* dan *RTT* untuk melihat performansi jaringan *MTCP dan STCP* .




Gambar 6. *Flowchart* pengambilan data pengujian

Pada pengujian ini, akan dilihat perbandingan antara *single path TCP* dan *multipath TCP* untuk melihat performansi *QoS* dengan parameter berupa *Retransmission*, *throughput*, *Delay* dan *RTT*. Dikarenakan belum ada paper yang membahas pembagian presentase trafik pada multipath TCP maka penulis membagi trafik sebesar 50%-50%, 40%-60% dan 20%-80%. pembagian trafik ini ditentukan dengan besarnya beban link yang terdapat pada jaringan *data center*. Beban link ditentukan dengan melihat *cost* atau *congestion* jaringan terbesar menggunakan algoritma *dijkstra* setelah itu akan didapat jalur terbaik pada tujuan untuk membuat besarnya presentase *subflow* pada beban *link* tersebut. Pengujian ini dilakukan dengan beberapa sekenario:

**Sekenario 1**

Pada sekenario ini pengujian akan dilakukan dengan mengatur trafik *multipath TCP* pada masing masing *path* yaitu sebesar 50%. Selanjutnya akan dilakukan pengujian *Quality of Service* dan membandingkannya dengan *single path TCP* (STP) untuk melihat perbandingan *Retransmission*, *throughput*, *Delay* dan *RTT* pada kedua mekanisme tersebut. Pengujian dilakukan dengan mengirimkan file selamat 30 detik memenuhi jaringan menggunakan *iperf* dan mengukur QoS tersebut menggunakan *wireshark*. Pengujian dilakukan sebanyak 30 kali.

**Sekenario 2**

Pada sekenario ini pengujian akan dilakukan dengan mengatur trafik *multipath TCP* pada masing masing *path* yaitu sebesar 40% dan 60%. Selanjutnya ditambahkan *background traffic*k selama 30 detik pada jaringan tersebut, lalu dilakukan pengujian *Quality of Service* dan membandingkannya dengan saat tanpa *background traffic* dan STP untuk melihat perbandingannya. Pengujian dilakukan dengan mengirimkan file selamat 30 detik memenuhi jaringan menggunakan *iperf* dan mengukur QoS tersebut menggunakan *wireshark* tanpa *background* traffik. Pengujian dilakukan sebanyak 30 kali.

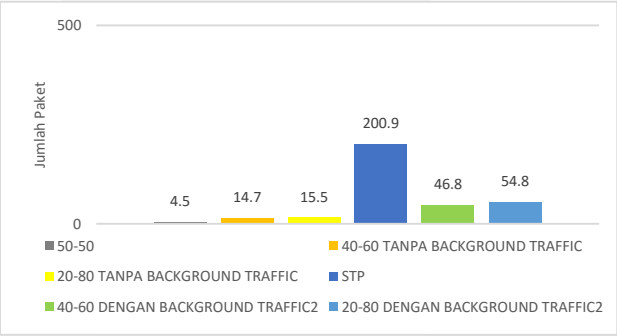
**Sekenario 3**

Pada sekenario ini pengujian akan dilakukan dengan mengatur trafik *multipath TCP* pada masing masing *path* yaitu sebesar 20% dan 60%. Selanjutnya ditambahkan *background traffic*k selama 30 detik pada jaringan tersebut, lalu dilakukan pengujian *Quality of Service* dan membandingkannya dengan saat tanpa *background traffic* dan STP untuk melihat perbandingan*nya*. Pengujian dilakukan dengan mengirimkan file selamat 30 detik memenuhi jaringan menggunakan *iperf* dan mengukur QoS tersebut menggunakan *wireshark* tanpa *background* traffik. Pengujian dilakukan sebanyak 30 kali.

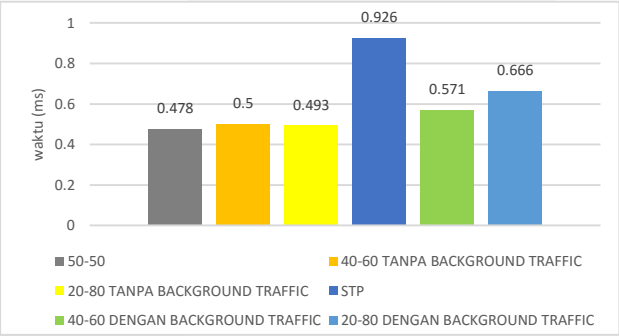
### 3.3 Hasil dan analisis Keseluruhan

Tabel 1. *Tabel Hasil Keseluruhan Pengujian*

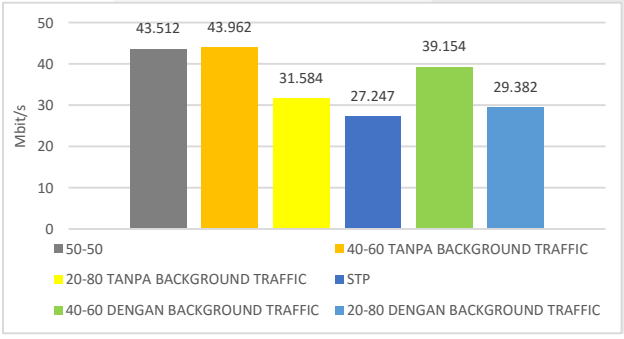
| | Sekenario 50-50 | Sekenario 40-60 | | Sekenario 20-80 | | STP |
|---|---|---|---|---|---|---|
| | MTCP | MTCP Tanpa *background traffic* | MTCP dengan *background traffic* | MTCP Tanpa *background traffic* | MTCP dengan *background traffic* | |
| Throughput (Mbit/s) | 43.521 | 43.045 | 39.154 | 31.584 | 29.382 | 27.247 |
| Delay (ms) | 0.269 | 0.318 | 0.330 | 0.287 | 0.334 | 0.889 |
| RTT (ms) | 0.478 | 0.500 | 0.571 | 0.493 | 0.666 | 0.926 |
| Retransmissi (Paket) | 4.5 | 14.7 | 46.8 | 15.3 | 54.8 | 200.900 |
| Beban Link 1 (Mbit/s) | 21.835 | 18.617 | 18.663 | 6.553 | 6.173 | - |
| Beban Link 2 (Mbit/s) | 22.031 | 25.556 | 21.153 | 25.411 | 23.445 | - |



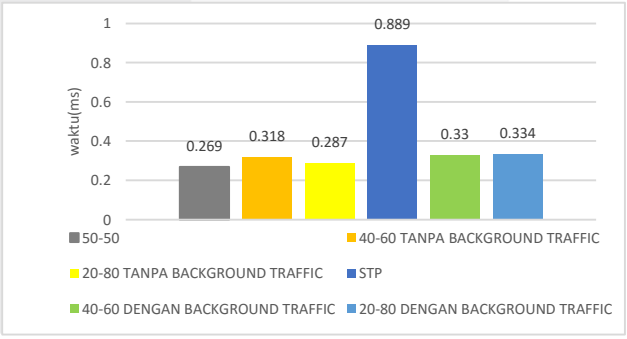

Gambar 7. *Retransmisi* seluruh pengujian



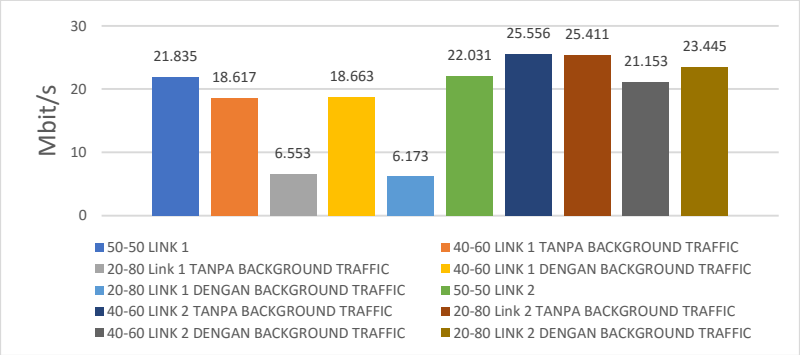

Gambar 8. *RTT* seluruh pengujian




Gambar 9. *Throughput* trafik *Multipath TCP dan STP*




Gambar 10. *Delay* seluruh pengujian




Gambar 11. *Beban Link* seluruh pengujian

Gambar 7-11 merupakan merupakan hasil pengukuran *Retransmission, RTT, Throughput, Delay* dan Beban *link* terhadap trafik *multipath TCP* pada seluruh sekenario. Perbandingan nilai retransmisi *multipath TCP* pada tanpa *background traffic*, sekenario 50-50 memiliki nilai retransmisi paling kecil yaitu hanya meretransmisikian rata-rata 4.5 paket saja. Pada *background traffic* scenario 40-60 memiliki nilai retransmisi lebih kecil yaitu dengan 46.8 paket sedangkan *STP* memiliki nilai retransmisi yang besar yaitu 200.9 paket.

Perbandingan nilai RTT pada *Multipath TCP* tanpa *background traffic* memiliki rata-rata RTT sebesar 0.49 ms sedangakan dengan *background traffic* pada sekenario 40-60 memiliki nilai yang lebih kecil yaitu 0.571 dibandingkan scenario 20-80 yang memiliki RTT 0.666 ms sementara nilai RTT *STP* memiliki rata-rata yang lebih besar yaitu 0.926 ms.

Nilai *throughput* yang dihasilkan *Multipath* TCP tanpa *background traffic* sekenario 20-80 memiliki nilai *throughput* paling kecil dengan nila maksimal rata-rata 31.5 Mbit/s karena pada link kedua bandwith link maksimal sebesar 30mb/s sehingga tidak dapat memaksimalkan seluruh bandwith jaringan. sementara dengan dialiri *background traffic* mempunyai nilai *throughput* rata-rata 43.962 Mbit/s. *STP* mendapat nilai rata-rata *throughput* maksumum 27.247 Mbit/s.

Perbandingan nilai *delay* pada tanpa *background traffic* maupun dengan *background traffic* memiliki kualitas jaringan yang lebih baik yaitu dengan rata-rata 0.31ms sedangkan *STP* memiliki nilai delay yang lebih besar yaitu 0.888 ms. Mengacu kepada standar ITU-T G.114 untuk transmisi satu arah, pada pengujian *delay* trafik *multipath TCP* sudah memenuhi standar apabila *data rate* yang dikirim hingga 40 mbit/s dengan kapasitas *link* sebesar 60 mbit/s yaitu dengan rekomendasi *delay* di bawah 400 ms atau 150 ms. Ketika jaringan mengalami kemacetan yang akan menimbulkan peningkatan *delay* yang tajam di atas 400 ms sehingga jaringan sudah tidak dalam kondisi yang baik.

Nilai beban link yang dihasilkan *Multipath* TCP tanpa *background traffic* memiliki nilai beban yang sesuai dengan sekenario. Pada saat diberi *background traffic* nilai beban link pada sekenario 40-60 hnya dapat mengalirkan beban link maksimal sebesar 2Mbit/s. hal ini membuat kedu link menjadi seimbang. sedangkapn pada scenario 20-80 *background traffic* maksimal sebesar 11Mbit/s memenuhi subflow 1 sehingga membuat beban link pada subflow 1 naik dan menaikan throughput jaringan.

## 4. Kesimpulan

Berdasarkan pengujian dan analisis yang telah dilakukan pada jaringan *data center* di SDN menggunakan *Ryu controller* dengan algoritma *djikstra multipath TCP*, dapat diambil kesimpulan sebagai berikut:

1. *Multipath* TCP terbukti dapat mengatur trafik pada dua buah subflow yang telah dibuat. Pembuktian pengaturan trafik dapat dilihat pada tabel 5.1 dimana beban link pada sublow 1 dan subflow 2 terbagi sesuai presentase yang diberikan.

**Tabel 1.** *Tabel Hasil pengaturan trafik*

| | 50-50 | 40-60 | 20-80 |
|---|---|---|---|
| subflow 1 (Mbit/s) | 21.858 | 18.167 | 6.553 |
| subflow 2 (Mbit/s) | 22.031 | 25.556 | 25.411 |
| Throughput (Mbit/s) | 43.521 | 43.045 | 31.584 |

2. Implementasi MPTCP pada jaringan data center mampu meningkatkan nilai *throughput* dari pada regular TCP dengan STP yang hanya menggunakan satu *interface*. Hal ini dapat dilihat dari hasil pengujian pada skenario 1. *Multipath* TCP dengan dua buah *subflow* memiliki nilai *throughput* sebesar 43,521 Mbps jika dibandingkan dengan regular TCP dengan STP yang hanya menggunakan satu

interfcae yaitu sebesar 27,247 Mbps. *Delay* yang didapatkan dari hasil pengukuran terhadap trafik *multipath TCP* pada SDN menggunakan topologi *data center fat-tree* adalah sebesar 0.278 hingga 0.334. Hal ini telah memenuhi standar ITU-T G.114 untuk *one-way transmission* yaitu dengan rekomendasi di bawah 150 ms.

3. Penambahan *background traffic* pada system *multipath* TCP dengan dua buah *subflow* dapat mengurangi nilai beban link pada *subflow* 2. nilai beban *link* pada *subflow* 1 yang meningkat karena ada *background traffic* sementara beban link pada *subflow* 2 menurun dikarenakan *background traffic* hanya memenuhi *subflow* 1, sehingga pada *subflow*1 beban trafik bertambah serta mengurangi beban trafik pada *subflow* 2 karena *multipath* TCP telah mengatur sesuai pada presentase pada *link*. Ketika beban *link* pada *subflow* 1 berkurang karena *background traffic* yang mengalir pada *subflow* 1 maka mempengaruhi presentase pada *subflow* 2 sehingga mgnurangi beban link pada *subflow* 2.

**Tabel 2.** *Tabel Pengaruh Background traffic*

| | 40-60 | | 20-80 | |
|---|---|---|---|---|
| | MTCP tanpa *background traffic* | MTCP dengan *background traffic* | MTCP tanpa *background traffic* | MTCP dengan *background traffic* |
| *subflow 1 (Mbit/s)* | 18.167 | 20.792 | 6.553 | 18.776 |
| *subflow 2 (Mbit/s)* | 25.556 | 23.028 | 25.411 | 25.902 |
| *Throughput (Mbit/s)* | 43.045 | 43.962 | 31.584 | 43.985 |
| *Background traffic* | 4Mbit/s | | 11Mbit/s | |

**Daftar Pustaka:**


[1] Y. Peng, Y. Yuan, X. Huang, W. Wu, X. Meng, and N. Supercomputer, "Research on Maintainability of Network Topology for *Data center*s," vol. 1, pp. 317–321, 2014.

[2] Z. Qian and B. Hu, "An Efficient Routing Algorithm in Fat-tree *Data center* Networks," pp. 0–5, 2016.

[3] Allied Telesis, "Multicasting White *Paper*," pp. 1–22, 2007.

[4] Anjani. M,"Implementasi Load Balancing dengan Metode ECMP (Equal Cost Multi *Path*)".Universitas Bina Darma. Palembang

[5] Nugroho. Agung,"Analisis Perbandingan Performa Algoritma Round Robin dan Least Connection untuk Load Balancing pada Software defined Network".Universitas Brawijaya. Malang

[6] Eka. Aldana,"Sistem Optimasi Pembebanan jaringan dengan Koneksi Internet Menggunakan Mikrotik".Universitas Binus. Jakarta

[7] Gumilar, Wildan , Implementasi *Multipath* TCP pada Jaringan *Wired* dan *Wireless*.Universitas Telkom.Bandung

[8] Open Network Foundation, "*Software*-Defined Networking: The New Norm for Networks," pp. 1–12, 2012.

[9] Hussein. Ali," SDN for *MTCP*: An Enhanced Architecture for Large Data Transfers in Datacenters ".American University. Beirut

[10] Anggita , Brayan L , Perancangan dan Analisis *Software defined network* pada jaringan LAN: Penerapan dan analisis metode Penjakuran *path Calculating* menggunakan algoritma djikstra, Universitas Telkom. Bandung 2014.

[11] Hyunwoo, Nam. Towards Dynamic MPTCP *Path* Control Using SDN.Columbia University. Newyork.

[12] Ryanda P, "Emulasi dan analisis keamanan jaringan virtual data center dengan memanfaatkan sflow dan *OpenFlow* untuk mendeteksi dan memitigasi syn flood attack," 2016. Universitas Telkom.Bandung.

[13] Charles E. Leiserson *Fat-trees: universal networks for hardware-efficient supercomputing*, IEEE Transactions on Computers, Vol. 34 , no. 10, Oct. 1985, pp. 892-901.

[14] Charles E. Leiserson, Zahi S. Abuhamdeh, David C. Douglas, Carl R. Feynman, Mahesh N. Ganmukhi, Jeffrey V. Hill, W. Daniel Hillis, Bradley C. Kuszmaul,

[15] L. Huawei Technologies Co., "Multicast *Paper* Technology," 2011.

[16] Open Network Foundation, "*Software*-Defined Networking (SDN) Definition," 2017. [Online]. Available: https://www.opennetworking.org/sdn-resources/sdn-definition.

[17] Murdha. Sawung A. Pengujian Performa Kontroler *Software defined network* (SDN): *Ryu* and Floodight.Institut Teknologi Bandung.Bandung

[18] V. Listiani, "ANALISIS PERFORMANSI SDN (*SOFTWARE* DEFINED NETWORK) MENGGUNAKAN PROTOKOL ROUTING OSPF (OPEN SHORTEST *PATH* FIRST)."