# Analisis Unjuk Kerja *Perfomance-oriented Congestion Control* (PCC) Menggunakan Video *Streaming*

**Rezy Noerdyah Ayu Saputri[1], Hilal Hudan Nuha[2], Sidik Prabowo[3]**

[1,2,3]Fakultas Informatika, Universitas Telkom, Bandung
[1]rezynoerdyah@students.telkomuniversity.ac.id, [2]hilalhudan@telkomuniversity.ac.id,
[3]sidikprabowo@telkomuniversity.ac.id

**Abstrak**

**Arsitekstur *Transmission Control Protocol* (TCP) *congestion control* mengalami masalah kinerja yang tidak optimal. Sehingga membuat TCP dan berbagai variasinya mempunyai harapan kecil dalam mencapai kinerja yang tinggi. Penyebabnya adalah TCP menggunakan *hardwired mapping* yaitu suatu kejadian yang sudah diasumsikan dan tindakan yang harus dilakukan untuk mengatasi kejadian itu tanpa mengerti kondisi dari jaringan secara nyata dan kinerja yang dihasilkan. *Perfomance-oriented Congestion Control* (PCC) merupakan arsitekstur *congestion control* baru yang membuat setiap pengirim mengamati tindakan dan kinerja jaringan secara empiris, bertujuan mengambil tindakan yang menghasilkan kinerja tinggi. PCC melakukan percobaan dalam beberapa kasus, salah satu kasus yaitu video *streaming*. Proses pengujian video *streaming* mendapatkan data berupa *throughput, delay* dan *packet loss*. Data tersebut dibandingkan antara PCC dan TCP dengan tujuan untuk mengetahui hasil kinerja yang lebih baik. Hasil dari pengujian dari setiap parameter yaitu *throughput* TCP adalah 1064,841 kbps dan PCC 150,825 kbps, delay dari TCP adalah 5,326 ms dan PCC 3,843 ms dan packet loss dari TCP adalah 0,905% dan PCC 0,016%. Sehingga PCC memiliki kinerja yang baik dari parameter *delay* dan *packet loss.* Sedangkan TCP kinerjanya lebih baik dilihat dari parameter *throughput.***

**Kata kunci :** ***delay, packet loss,* PCC, TCP *congestion control, throughput,* video *streaming***

**Abstract**

**Transmission Congestion Control (TCP) congestion control architecture experienced poor perfomance for several years. It's make TCP variants has little hope for achieve high perfomance. The reason is TCP used hardwired mapping that is the assumed occurrence and the actions that be taken to resolve the event without understand condition in real network and the result of perfomance. Perfomance-oriented Congestion Control (PCC) is new congestion control architecture in which each sender continuously observes the connection between its actions and empiricially experience perfomance, enabling it to consitently adopt actions that result in high perfomance. PCC do the experiment in many cases, one of them is video streaming. In the testing video streaming obtained data in the form of throughput, delay and packet loss. The data obtained compared between PCC and TCP with the purpose of finding out better perfomance results. The test results form each parameter TCP throughput is 1064,841 kbps and PCC 150,825 kbps, delay from TCP is 5,326 ms and PCC 3,843 ms and packet loss from TCP is 0,905% and PCC 0,016%. With that result PCC have better perfomance from delay and packet loss parameter. Whereas TCP performs better seen from throughput parameters.**

**Keywords: delay, packet loss, PCC, TCP, congestion control, throughput, video streaming**

## 1. Pendahuluan

### Latar Belakang

*Transmission Congestion Control* (TCP) adalah protokol yang di layer transport yang sering digunakan *Internet Protocol*. TCP mendukung transfer lebih dari 90% dari semua trafik di internet saat ini [1]. TCP mempunyai berbagai mekanisme, salah satu mekanisme utamanya adalah *congestion control* untuk mengontrol lalu lintas jaringan agar tidak terjadi *congestion* atau kemacetan [2]. Selama penerapannya, walaupun TCP menggunakan algoritma yang berbeda, tapi variasi TCP mempunyai *hardwired mapping* yang sama, yaitu kejadian yang terdefinisikan untuk menghasilkan tindakan kontrol ditentukan. *Hardwired mapping* membuat TCP mengasumsikan kejadian di jaringan. TCP mengasumsikan bahwa *loss* mengindikasi terjadinya *congestion*. Ketika asumsi ini terbukti, mengurangi separuh ukuran *window* akan menurunkan *loss rate* dan meningkatkan kinerja.

Melainkan, asumsi ini dapat terlanggar yaitu *packet loss* dapat tidak teratur dan tidak berhubungan dengan *congestion* dan karena itu mengurangi ukuran *window* akan menyebabkan penurunan kinerja dan penurunan kelayakan. Pada kenyataannya, jaringan saat ini kompleks yaitu koneksi yang dari Kbps menjadi 100 Gbps, tidak stabilnya pengaturan *routing*, *Active Queue Management* (AQM), *software router*, pemeriksa paket dan *load balancer* [3]. Salah satu contoh menurunnya kinerja TCP adalah TCP Cubic, mengoptimasi pada tingginya koneksi *Bandwidth Delay Prroduct* (BDP), biasanya beroperasi 10x jauh dari optimal *throughput* di internet, TCP Hybla mengoptimasi *lossy* dai panjangnya koneksi RTT satelit, dalam praktiknya hanya mendapatkan 6% dari kapasitasnya [4].

Dalam menangani permasalahan pada TCP, peneliti mengusulkan *Perfomance-oriented Congestion Control* (PCC) yang merupakan arsitekstur *congestion control* terbaru yang setiap pengirim terus menerus mengamati tindakan dan kinerja jaringan secara empiris, yang bertujuan menghasilkan kinerja tinggi. PCC mengirimkan *rate* pada periode waktu yang singkat, dan mengamati hasilnya (SACK yang berisi pengiriman, *loss*, dan *latency* dari setiap paket). Kemudian menggabungkan *packet-level event* ini menghasilkan nilai utilitas *u*. Untuk membuat keputusan kontrol pada *rate*, PCC menjalankan banyak *micro-experiments* yang mencoba mengirimkan dua *rate* yang berbeda, dan berpindah ke arah hasil empiris dengan kinerja utilitas yang bagus [3]. PCC membuktikan dari berbagai kasus seperti pengiriman data, video *streaming*, koneksi satelit, dan sebagainya [3], [5]. Dalam kasus ini video *streaming* dipilih kerena menjadi salah satu jenis lalu lintas data yang paling intesif dalam pemanfaatan *bandwidth* yang besar di jaringan [6]. Salah satuh contoh pemanfaatan *bandwidth* video *streaming* dari situs *youtube* membutuhkan 562,5 MB/jam untuk resolusi 480p, 1,86 GB/jam untuk kualitas 720p, 3,04GB/jam pada 1080p dan 15,98GB untuk kualitas 4K [7]. Situs *web* juga mempunyai lebih banyak *request*, tapi skalanya tidak sebesar video dalam melakukan *request* dan bit yang ditransfer [6].

Untuk membuktikan kinerja PCC, pengujian menggunakan VLC media player yaitu yaitu pemutar multimedia dari berbagai format audio, video dan macam protokol *streaming* [8]. VLC media player digunakan sebagai penyedia layanan *streaming*. Dalam pengambilan data menggunakan Wireshark yaitu *Network Protocol Analyzer* atau ltepatnya penganalisa protokol jaringan yang lengkap. Program ini dapat merekam semua paket yang lewat serta menyeleksi dan menampilkan data tersebut dengan detail [9]. Pengambilan data dari wireshark berupa *throughput, delay* dan *packet loss.* Untuk memenuhi skenario unjuk kerja PCC yaitu ketika keadaan trafik tidak kosong, yaitu menggunakan *Distributed Internet Traffic Generator* (D-ITG). D-ITG adalah platform yang cocok untuk memproduksi trafik IPv4 dan IPv6 secara tepat mereplikasi *workload* dari aplikasi internet yang digunakan [10]. D-ITG digunakan sebagai *background traffic* saat menjalankan pengujian. Hasil kinerja dari PCC dibandingan dengan hasil kinerja TCP. Dengan perbandingan tersebut, dapat diketahui/dianalisa protokol yang mempunyai unjuk kerja atau kinerja yang lebih baik.

**Topik dan Batasannya**

Topik dan batasan yang digunakan dalam tugas akhir ini yaitu pengujian menggunakan tiga *personal computer* (PC) sebagai *client*, satu PC sebagai server dan satu PC sebagai *traffic generator*, *switch*, dan kabel UTP CAT5E. Selama pengujian tidak membahas mengenai keamanan jaringan. Variasi algoritma yang digunakan yaitu TCP Cubic dan PCC Vivace. Untuk protokol protokol *streaming* yang digunakan adalah RTSP dan HTTP di VLC media player. Hasil dari pengujian merupakan parameter pengukuran yaitu *throughput*, *delay*, dan *packet loss* yang didapatkan dari wireshark.

**Tujuan**

Tujuan penulis yang ingin dicapai dalam tugas akhir ini adalah:

**Tabel 1 Tujuan dari Tugas Akhir**

| No | Tujuan | Pengujian/skenario |
|---|---|---|
| 1 | Mengetahui kinerja TCP dan PCC dari parameter *throughput* | 1. Video *streaming* menggunakan TCP dan PCC tanpa *background traffic*<br>2. Video *streaming* menggunakan TCP dan PCC dengan *background traffic* 10 MB, 20 MB, dan 30 MB<br>3. Video *streaming* menggunakan TCP dan PCC dengan *background traffic* 90% dari kapasitas kabel yang digunakan |
| 2 | Mengetahui kinerja TCP dan PCC dari parameter *delay* | |
| 3 | Mengetahui kinerja TCP dan PCC dari parameter *packet loss* | |

**Organisasi Tulisan**

Bagian pertama adalah dengan melakukan proses studi literatur. Proses studi literatur yaitu pengumpulan informasi berupa buku, jurnal dan internet. Bagian selanjutnya melakukan perancangan sistem, yaitu menyiapkan kebutuhan pengujian dan membuat topologi dari pengujian. Kemudian bagian selanjutnya adalah pengujian dan pengambilan data, dalam pengujian ini secara langsung data didapatkan. Bagian selanjutnya adalah Analisis hasil yaitu data yang didapatakan kemudian dikumpulkan dan dibuat menjadi grafik kemudian dianalisis. Bagian terakhir adalah pembuatan laporan hasil pengujian.

## 2. Dasar Teori

### *2.1 Transmission Control Protocol* (TCP)

TCP adalah protokol di layer transport yang menyediakan mekanisme transfer data yang *reliable*, sehingga aliran data yang dibaca TCP *receiver* tidak rusak, tanpa duplikasi, dan berurutan. Untuk menyediakan transfer data yang *reliable*, TCP menyediakan layanan *error checked* dan *flow control*. Layanan *error checking* memungkinkan deteksi beberapa kesalahan dan rekontruksi data menjadi data asli. Layanan *flow control* digunakan TCP untuk memastikan TCP *sender* tidak mengirimkan paket lebih cepat daripada yang dapat ditampung TCP *receiver*. Ketika TCP sender mengirimkan data lebih cepat daripada yang bisa ditangani TCP *receiver*, maka akan terjadi *congestion* [2].

Mekanisme *congestion control* diklasifikasikan kedalam 4 fase, yaitu *slow start*, *congestion avoidance*, *fast retransmit* dan *fast recovery*. Fase *slow start* digunakan oleh TCP *sender* untuk menyesuaikan laju aliran data ke TCP *receiver* dimana periode *slow start* baru dimulai dengan setiap *acknowledgement* (ACK) yang diterima dari TCP *receiver*. Fase *congestion avoidance* diimplementasikan ketika ukuran *congestion window* (cwnd) lebih besar daripada *threshold slow start*. *Fast retransmit* dan *fast recovery* menyediakan mekanisme untuk mempercepat koneksi. Mekanisme ini adalah metode *recovery* yang digunakan oleh TCP untuk menghindari waktu tunggu saat pengiriman ulang ketika time out terjadi setiap segmen hilang [2].

### *2.2 TCP CUBIC*

TCP CUBIC merupakan salah satu algoritma *congestion control*. TCP CUBIC adalah algoritma TCP *congestion control* yang saat ini digunakan secara default dalam kernel linux sejak versi kernel 2.6.19. Algoritma ini melakukan modifikasi pertumbuhan *window* secara linear yang terdapat dalam TCP standar menjadi fungsi cubic. Algoritma pertumbuhan *window* TCP CUBIC adalah sebagai berikut, setelah terjadi sebuah *loss event* dan pengurangan *window*, maka *Wmax* (*Window* maksimum) akan menjadi ukuran cwnd dan berkurang secara multiplikatif berdasarkan faktor β dimana β adalah sebuah konstanta penurunan dan *fast recovery*. Pertumbuhan *window* pada TCP Cubic mengikuti Persamaan berikut [11] :

$$W_{(t)} = C(t-K)^3 W_{max} \quad (1)$$

W(t) : Ukuran *congestion window*
C : Parameter Cubic
T : Waktu dari pengurangan window terakhir
K : Waktu dari pengubahan nilai dari W ke *Wmax* jika tidak terdapat loss event
*Wmax* : *Window* maksimum [11].
K dihitung dengan persamaan berikut :

$$K = \sqrt[3]{\frac{W_{max}\beta}{C}} \quad (2)$$

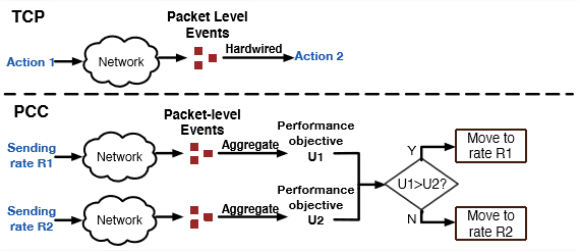
K : Waktu dari pengubahan nilai dari W ke Wmax jika tidak terdapat *loss event*
*Wmax* : Window maksimum
β : Konstanta penurunan dan *fast recovery*
C : Parameter Cubic [11]

Saat ACK diterima selama fase *congestion avoidance*, Cubic menghitung laju pertumbuhan *window* selama periode RTT selanjutnya menggunakan persamaan (1) dengan memasukan nilai W(t+RTT) sebagai kandidat baru dari *cwnd*. Bergantung pada besarnya nilai *cwnd*, CUBIC dapat berada dalam 3 kondisi/daerah. Pertama, jika *cwnd* bernilai lebih kecil dari besarnya *window* yang dapat dicapai oleh TCP standard (Reno) pada waktu *t* setelah *loss event*, maka CUBIC berada pada daerah TCP standar. Kedua, jika *cwnd* kurang dari *Wmax* maka

CUBIC berada dalam daerah *concave*, dan terakhir jika CUBIC lebih besar dari Wmax, maka CUBIC berada dalam daerah *convex* [11].

### *2.3 Perfomance-oriented Congestion Control (PCC)*

PCC yang merupakan arsitekstur *congestion control* terbaru yang dimana setiap pengirim terus menerus mengamati tindakan dan kinerja jaringan secara empiris, yang bertujuan menghasilkan kinerja tinggi. PCC menghubungkan tindakan kontrol (merubah *sending rate*) secara langsung dengan dampak pada kinerja nyata. Tujuan PCC adalah mengetahui bagaimana dalam melakukan tindakan apa dalam meningkat kan kinerja yang berdasarkan bukti eksperimen secara langsung, menghindari asumsi TCP mengenai jaringan [3].




**Gambar 1 Struktur pembuat keputusan TCP dan PCC [3]**

Tindakan kontrol PCC adalah pemilihan *sending rate*. PCC membagi waktu menjadi periode waktu yang sama secara kontinu yang disebut *Monitor Intervals* (MI), yang panjangnya satu RTT. Setiap MI, PCC melakukan test dengan memilih *sending rate* disebut *r* dan mengirimkan data. Setelah satu RTT, pengirim akan mendapat ACK (SACK) dari penerima, seperti TCP. Namun, tidak menimbulkan respons kontrol yang sudah ditentukan. Melainkan, PCC mengumpulkan SACK menjadi suatu metrik seperti *throughput, loss rate* dan *latency*. Metrik ini adalah kombinasi nilai utilitas yaitu *u*. Tujuannya adalah "*throughput* tinggi dan *loss rate* yang rendah", seperti $u = T - L$ (dimana $T$ = *throughput* dan $L$ = *loss rate*). Fungsi utilitas yaitu [3]:

$$u_i(x_i) = T_i \,.\, Sigmoid_{\propto}(L_i - 0.05) - x_i \,.\, L_i \qquad (3)$$

$x_i$ = pengirim *i* dari *sending rate*

$L_i$ = data *loss rate* yang diamati

$T_i(x) = x_i(1 - L_i)$ *throuhput* i dari pengirim

Sigmoid(y) = $\frac{1}{1+e^{\propto y}}$ jika beberapa $\propto > 0$ yang akan dipilih selanjutnya

PCC menjalankan *micro-experiments* secara terus menerus, membandingkan utilitas dari setiap *sending rate* berbeda sehingga dapat melacak tindakan optimal selama berjalannya waktu. Secara khusus, PCC menjalankan algoritma *online learning* sama dengan *gradient ascent*. Ketika dimulai dengan *rate r*, *rate* ini dites pada $(1+\varepsilon)r$ dan *rate* $(1-\varepsilon)r$ , dan berpindah ke arah (tinggi atau *rate* rendah) yang secara empiris menghasilkan utilitas tinggi. Jika berlanjut pada arah ini selama utilitas naik, tapi jika utilitas turun maka kembali ke state pembuat keputusan untuk menguji tinggi dan rendah kedua *rate* untuk mencari yang menghasilkan utilitas tinggi [3]. Contoh dari perhitung PCC didapatkan metrik T = 100 kbps, L = 5%, $x_i$=5 maka utilitas yang dihitung adalah $u_i(5)$ = 100.Sigmoid(0,05-0,05)- 5.0,05 = 49,75. Kemudian hasil tersebut dibandingkan dengan *sending rate* selanjutnya jika ternyata lebih besar utilitas ini maka akan terus berjalan dengan dua kali lipat *rate* ini. Namun jika ternyata hasilnya lebih kecil daripada sebelumnya, maka akan menggunakan *sending rate* sebelumnya.

PCC mempunyai dua utilitas yang berbeda, utilitas yang dijelaskan s dari PCC Allegro. Sedangkan fungsi utilitas lainnya adalah PCC Vivace. Vivace memakai arsitekstur level tinggi dari PCC yaitu fungsi utilitas dan algoritma pembelajaran *rate control*. Pertama, kerangka teori pembelajaran informasi untuk utilitas penurunan yang mengabungkan pertimbangan krusial seperti meminimal *latency* dan TCP friendliness. Kedua, Vivace menggunakan optimasi *online* optimal yang terbukti (asimptotik) berdasarkan *gradient ascent* untuk mencapai utilitas tinggi dari kapasitas jaringan, bereaksi cepat pada perubahan dan konvergensi yang cepat dan stabil. Fungsi utilitas yang digunakan PCC Vivace yaitu [5] :

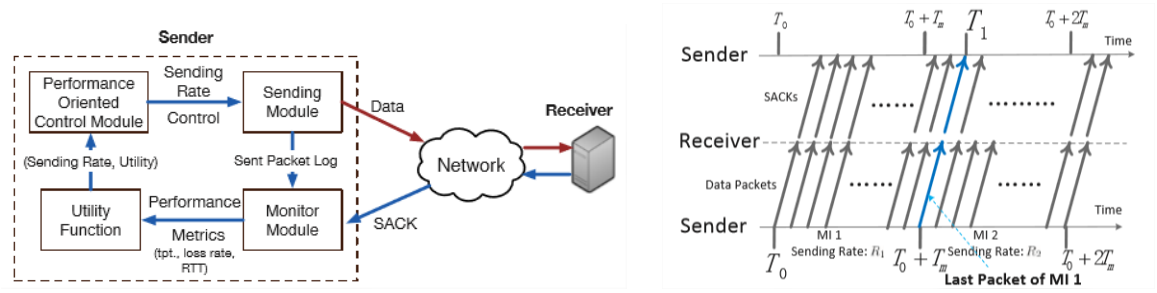
$$u\left(x_i, \frac{d(RTT_I)}{dT}, L_i\right) = x_i^t - bx_i \frac{d(RTT_i)}{dT} - cx_i \times L_I \qquad (4)$$

$x_I$ = pengirim *i* dari *sending rate*

$L_i$ = data *loss rate* yang diamati

$\frac{d(RTT_I)}{dT}$ = gradien RTT selama MI meningkan dengan *latency*

Dimana $0 < t < 1$, $b \geq 0$, $c > 0$ (parameter b,c,t adalah konstan)[5]

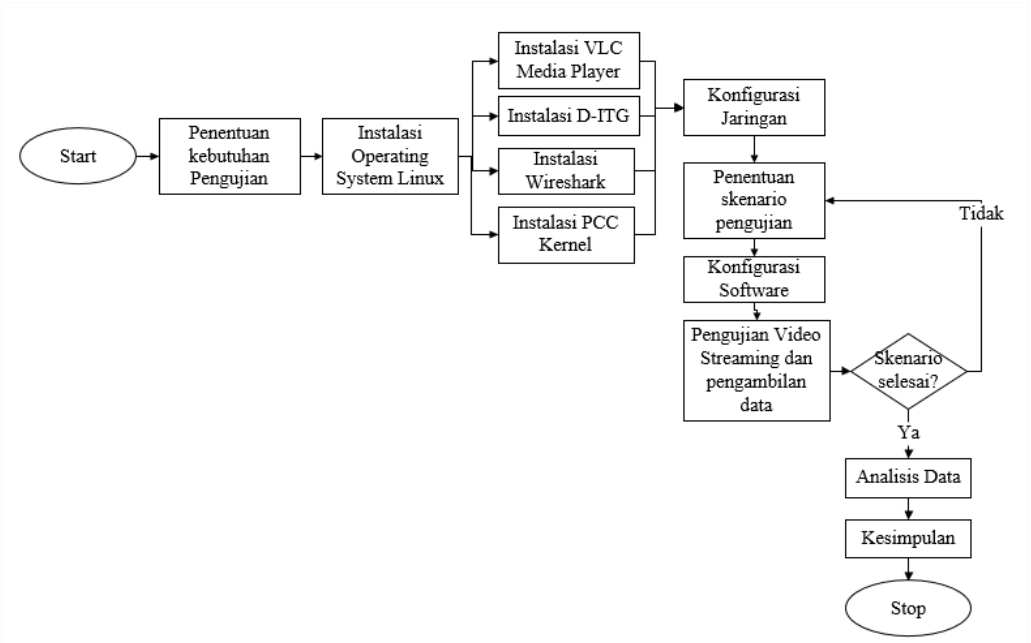



**(a)** **(b)**

**Gambar 2 (a) PCC *Prototype Design* (b) *Perfomance Monitoring Process*** [3]

Sesuai dengan gambar 2, pada PCC menggunakan $T_m$ yang disebut MI (RTT), dan akan menerima SACK yaitu *Selective acknolegdment*. Ketika paket dari MI1 selesai, maka yang awalnya $T_0$ akan menjadi $T_0+T_m$, $T_0$ disini adalah satu RTT. Terdapat state dari PCC yang berbeda dari TCP yaitu *starting state*, *decision state*, dan *rate adjusting state*. Pada *starting state*, PCC sama dengan *slow start* mendua kali lipat *sending rate* pada setiap MI. PCC akan keluar pada *state* ini jika nilai utilitas nya turun dan kembali ke *rate* sebelumnya yang mempunyai nilai utilitas yang lebih besar. Pada *decison state*, PCC menggunakan *multiple randomized controlled trials* (RCTs). PCC mengambil 4 MI yang berurutan membaginya menjadi dua pasangan, dan satu pasangan berada di *rate* yang tinggi $(1+\varepsilon)r$ disebut $U_1^+, U_2^+$ dan pasangan lain menggunakan *rate* yang rendah $(1-\varepsilon)r$ disebut $U_1^-, U_2^-$. Kemudian PCC melakukan test tiap *rate* dua kali (selama satu RTT) dan menentukan *rate* yang memiliki utilitas tinggi. Jika $U^+$ mempunyai utilitas yang tinggi maka *rate* nya menggunakan *rate* dari $U^+$ begitu sebaliknya dan akan berlanjut ke *state* selanjutnya. Tapi jika hasilnya adalah $U_1^+ > U_1^-$ dan $U_2^+ < U_2^-$, maka akan berada pada *state* ini untuk mencari *rate*. Ketika sudah keluar dari *decision state* dan berpindah ke *rate adjusting state*, pada *state* melakukan pergantian *rate* untuk mendapatkan nilai utilitas yang besar, ketika menurun maka akan kembali ke *state* sebelumnya [3].

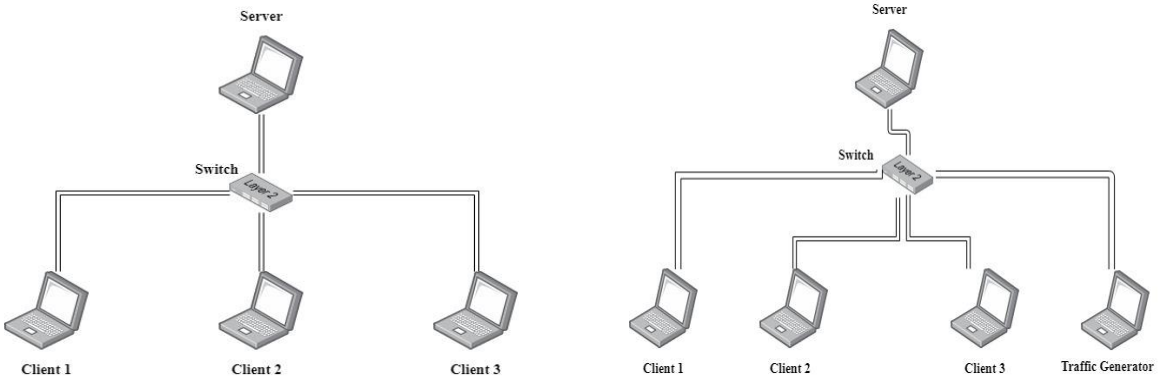
## 3. Sistem yang Dibangun




**Gambar 3 Gambaran Umum sistem yang dibangun**

Tahapan pertama adalah menentukan kebutuhan pengujian yaitu mempersiapkan perangkat keras (*hardware*) dan perangkat lunak(*software*). Perangkat yang dibutuhkan sesuai dengan kebutuhan skenario yaitu tiga *client*, satu server dan satu *traffic generator*. Sebagai penghubung menggunakan *switch* dan kabel UTP. Kemudian melakukan instalasi *operating system* Linux. Setelah itu melakukan instalasi *software* yang dibutuhkan yaitu VLC media player sebagai penyedia dan pengakses layanan *stream*, Wireshark sebagai penangkap paket yang masuk saat pengujian, *Distributed Internet Traffic Generator* (D-ITG) sebagai *background traffic*, dan PCC kernel sebagai protokol yang digunakan untuk pengujian. Untuk menginstal PCC mengakses dari link https://github.com/PCCproject/PCC-Kernel/tree/vivace. Sedangkan TCP Cubic sudah menjadi *default congestion control* di linux.

Langkah selanjutnya adalah melakukan konfigurasi jaringan yaitu saat pengujian jaringan menggunakan alamat *Internet Protocol* (IP) manual. Pengaturan alamat IP bertujuan agar setiap *Personal Computer* (PC) yang digunakan mempunyai alamat masing-masing. Pengaturan IP yaitu *client* 1 memiliki alamat IP 192.168.2.5, *client* 2 memiliki alamat IP 192.168.2.7, *client* 3 memiliki alamat IP 192.168.2.8, server memiliki alamat IP 192.168.2.2 dan *traffic generator* memiliki alamat IP 192.168.2.6. Setelah melakukan pengaturan alamat IP, tahapan selanjutnya membuat topologi jaringan. Topologi jaringan terbagi menjadi dua skenario yaitu skenario tanpa *background traffic* dan menggunakan *background traffic*.




**(a)** **(b)**

**Gambar 4 (a) Topologi Tanpa *Background Traffic* (b) Topologi Menggunakan *Background Traffic***

Tahapan selanjutnya adalah menentukan skenario pengujian, Skenario pengujian terbagi menjadi dua bagian yaitu tanpa *background traffic* dan menggunakan *background traffic* sesuai dengan tabel sebagai berikut :

**Tabel 2 Skenario Pengujian**

| No | Skenario Pengujian | Pengujian | Keterangan |
|---|---|---|---|
| 1 | Tanpa *background traffic* | Pengujian 1 | Menggunakan protokol TCP CUBIC dan RTSP sebagai protokol *streaming* |
| | | | Menggunakan protokol PCC Vivace dan HTTP sebagai protokol *streaming* |
| 2 | Menggunakan *background traffic* | Pengujian 2 | Menggunakan protokol TCP CUBIC dan RTSP sebagai protokol *streaming*. *Background traffic* yang digunakan 10 MB/s menggunakan UDP |
| | | | Menggunakan protokol PCC Vivace dan HTTP sebagai protokol *streaming*. *Background traffic* yang digunakan 10 MB/s *background traffic* menggunakan UDP |
| | | Pengujian 3 | Menggunakan protokol TCP CUBIC dan RTSP sebagai protokol *streaming*. *Background traffic* yang digunakan adalah 20 MB/s menggunakan UDP |
| | | | Mengguanakan protokol PCC Vivace dan HTTP sebagai protokol *streaming*. *Background traffic* yang digunakan adalah 20 MB/s *background traffic* menggunakan UDP |
| | | Pengujian 4 | Menggunakan protokol TCP CUBIC dan RTSP sebagai protokol *streaming*. *Background traffic* yang digunakan adalah 30 MB menggunakan UDP |
| | | | Mengguanakan protokol PCC Vivace dan HTTP sebagai protokol *streaming*. *Background traffic* yang digunakan adalah 30 MB/s *background traffic* menggunakan UDP |
| | | Pengujian 5 | Menggunakan protokol PCC Vivace dan TCP Cubic secara bersaama dengan protokol *streaming* RTSP dengan menggunakan utilitas *background traffic* 90% dari kapasitas *bandwidth* kabel yang disediakan yaitu 900 MB dalam waktu percobaan 10 detik. |

Parameter *Mega Bytes* (MB) dipilih karena kabel UTP yang digunakan mempunyai kemampuan mengirimkan data dalam 1 Gbps. Skenario pengujian sama dengan tanpa *background traffic*. Pengujian ke 5 menggunakan waktu 10 detik, karena kapasitas yang wireshark tidak bisa lama menampung *bandwidth* besar yang masuk. Hasil dari skenario adalah berupa data dari Wireshark yaitu pcapng. Kemudian parameter hasil yang digunakan adalah :

**Tabel 3 Parameter dari hasil yang ingin dicapai**

| Parameter | Satuan | Keterangan |
|---|---|---|
| *Throughput* | kilo bit per seconds (kbps) | jumlah data yang dapat dikirim dan diterima dalam jangka waktu tertentu. *Throughput* pada umumnya menggunakan *bit per seconds* (bps) untuk ukuran satuan [12] |
| *Delay* | miloseconds (ms) | waktu yang dibutuhkan data untuk menempuh jarak dari asal ke tujuan [13] |
| *Packet loss* | Persentase (%) | parameter yang menggambarkan suatu kondisi yang menunjukkan jumlah total paket yang hilang [14] |

Tahapan selanjutnya adalah melakukan konfigurasi *software* yaitu VLC Media player, Wireshark, D-ITG dan algoritma *congestion control* yang digunakan. Dalam melakukan video *streaming*, parameter video di VLC media player sebagai berikut :

Nama : Test_video1.mp4
*Frame rate* : 29 fps
Resolusi : 1920x1080
Waktu : 03.40 menit
*Audio bit rate* : 125 kbps

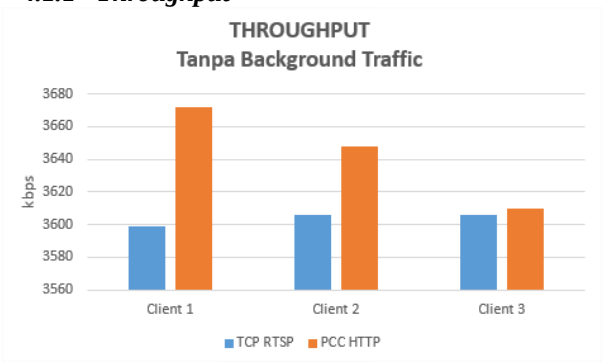
Untuk konfigurasi protokol yang digunakan pada linux adalah menggunakan perintah pada terminal seperti ini sysctl net.ipv4.tcp_available_congestion_control yang memberikan outpu *default congestion control*. Jika ingin merubahnya menggunakan sudo /sbin/sysctl -w net.ipv4.tcp_congestion_control=pcc [15].

## 4. Evaluasi

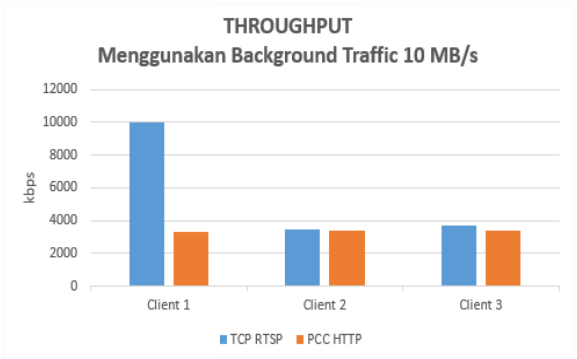
Pengujian dilakukan dengan tujuan untuk mengetahui kinerja yang lebih baik antara TCP dan PCC dengan membanding hasil dari video *streaming* berupa *throughput, delay* dan *packet loss*. Pengujian video *streaming* menggunakan VLC media player, untuk mengumpulkan data dan menganalisis menggunakan Wireshark dan sebagai *background traffic* menggunakan D-ITG. Hasil dan analisis pengujian menguunakan setiap parameter yang sudah ditetapkan sebelumnya

### 4.1 Hasil dan Analisis Pengujian

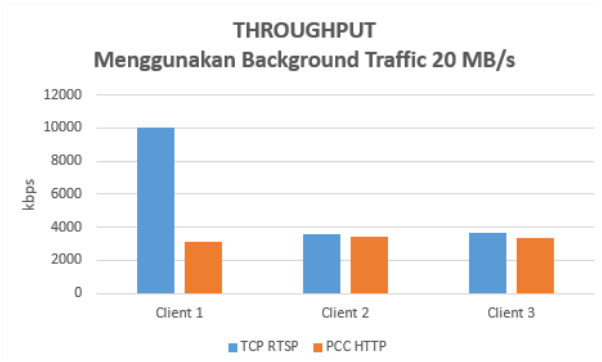
#### 4.1.1 *Throughput*



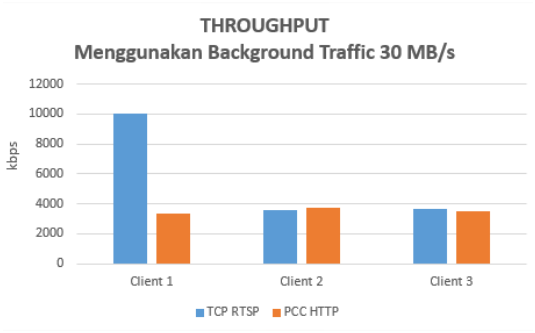

(a)



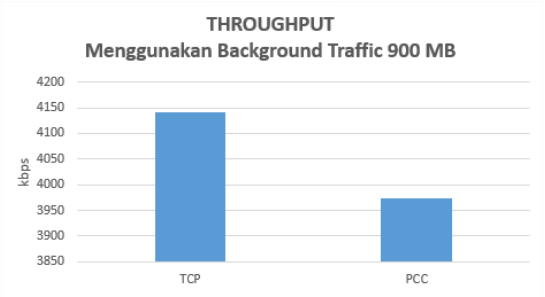

(b)




(c)




(d)



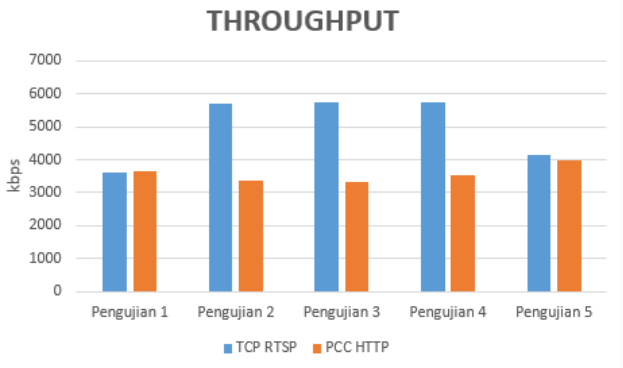

(e)

**Gambar 5 (a) Pengujian 1 : *Throughput* Tanpa *Background Traffic* (b) Pengujian 2 : *Throughput* Menggunakan *Background Traffic* 10 MB/s (c) Pengujian 3 : *Throughput* Menggunakan *Background Traffic* 20 MB/s (d) Pengujian 4 : *Throughput* Menggunakan *Background Traffic* 30 MB/s (e) Pengujian 5 : *Throughput* menggunakan *background traffic* 900 MB**

**Tabel 4 Rincian *throughput* dari setiap skenario pengujian**

| No | Skenario | Pengujian | Protokol | *Client* 1 | *Client* 2 | *Client* 3 |
|---|---|---|---|---|---|---|
| 1 | Tanpa *Background Traffic* | Pengujian 1 | TCP | 3599 kbps | 3606 kbps | 3606 kbps |
| | | | PCC | 3672 kbps | 3648 kbps | 3610 kbps |
| 2 | Menggunakan *Background Traffic* | Pengujian 2 | TCP | 10000 kbps | 3465 kbps | 3667 kbps |
| | | | PCC | 3279 kbps | 3387 kbps | 3403 kbps |
| | | Pengujian 3 | TCP | 10000 kbps | 3582 kbps | 3655 kbps |
| | | | PCC | 3110 kbps | 3457 kbps | 3385 kbps |
| | | Pengujian 4 | TCP | 10000 kbps | 3594 kbps | 3635 kbps |
| | | | PCC | 3323 kbps | 3754 kbps | 3482 kbps |
| | | Pengujian 5 | TCP | 4141 kbps | - | - |
| | | | PCC | 3974 kbps | - | - |

Setelah mendapatkan hasil dari setiap pengujian kemudian *throughput* dari setiap pengujian di rata-rata untuk melihat hasil dari setiap pengujian secara keseluruhan.




**Gambar 6 Rata-rata *throughput* dari setiap pengujian**

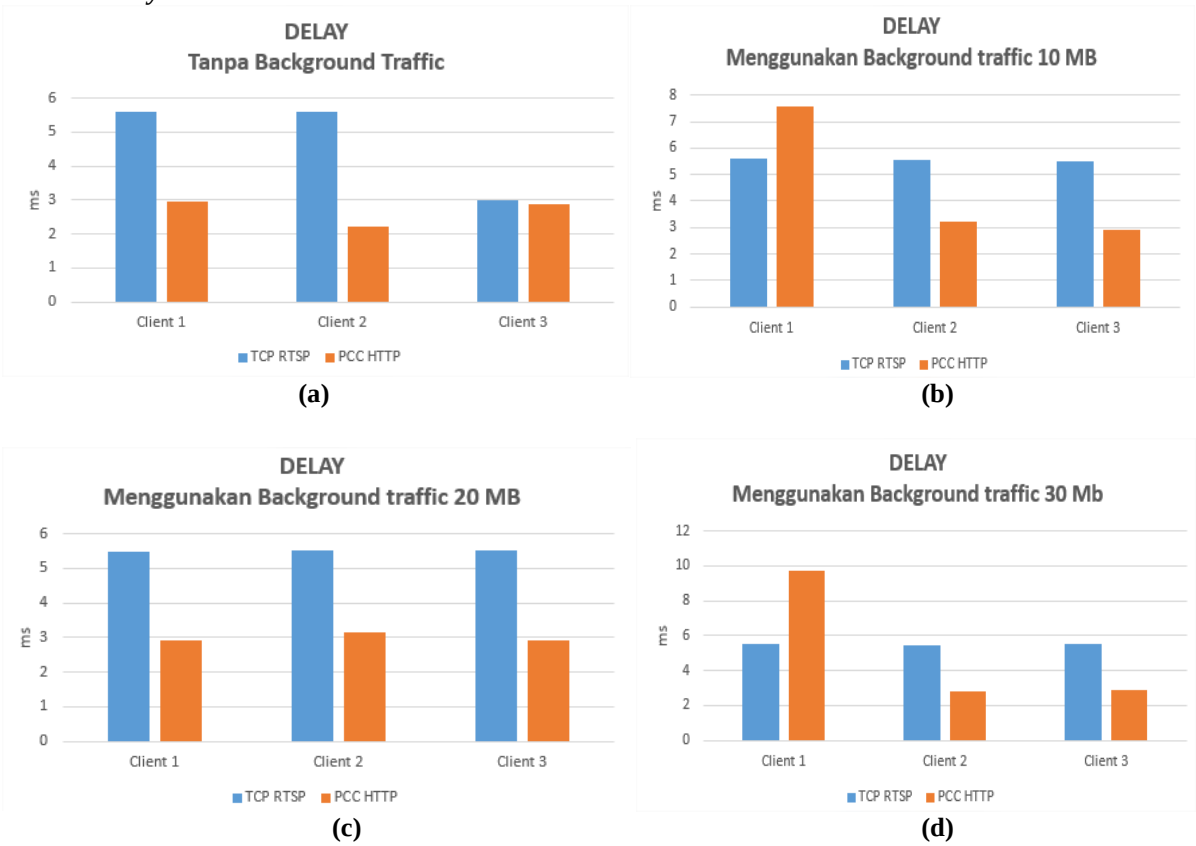
**Tabel 5 Rincian rata-rata *throughput* setiap pengujian**

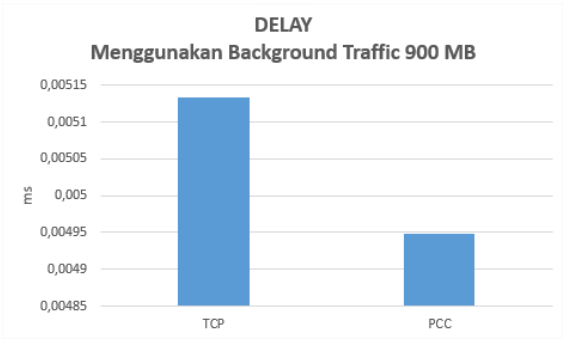
| No | Protokol | Pengujian 1 | Pengujian 2 | Pengujian 3 | Pengujian 4 | Pengujian 5 |
|---|---|---|---|---|---|---|
| 1 | TCP | 3603,667 kbps | 5710,667 kbps | 5745,667 kbps | 5743 kbps | 4141 kbps |
| 2 | PCC | 3643,334 kbps | 3356,334 kbps | 3317,334 kbps | 3519,667 kbps | 3974 kbps |

Berdasarkan dengan gambar 6 dan tabel 5, dari rata-rata *throughput* tanpa menggunakan *background traffic*, PCC lebih baik menghasilkan *throughput* daripada TCP yaitu 3643,444 kbps. Hal ini dikarenakan *packet loss* yang dihasilkan lebih kecil daripada TCP. Sedangkan TCP mempunyai *througput* yang lebih kecil yaitu 3603,667 kbps dari PCC dikarenakan saat melakukan pertumbuhan *window*, TCP Cubic bergantung pada besarnya nilai cwnd dan *packet loss* yang didapatkan TCP lebih besar dari PCC.

Untuk pengujian menggunakan *background traffic*, TCP lebih baik menghasilkan *throughput* dari PCC yaitu 5733,111 kbps dan 3397,111 kbps. Hal ini disebabkan karena terjadinya *event loss*, *Wmax* akan disetel menjadi ukuran *cwnd* dan bergantung pada besarnya *cwnd*, membuat Cubic akan berada di daerah yang sudah ditentukan dan *packet loss* yang dihasilkan tidak banyak saat tanpa *background traffic*. Sedangkan cara kerja PCC yaitu terdapat fungsi utilitas yang mepengaruhi *sending rate* yang dikirim. PCC menggunakan modul monitor yang mengingat paket yang dikirim setiap MI. Kemudian modul monitor yang menghasilkan metriks pada setiap paket yang diterima dan kemudian metriks itu menjadi fungsi utilitas. Fungsi utilitas yang dihasilkan dan dibandingkan setiap MI mempengaruhi dari *throughput* yang dihasilkan dan fungsi utilitas yang dihasilkan juga dipengaruhi oleh *loss rate* dan RTT yang dihasilkan, sehingga membuat nilai *throughput* lebih rendah dari yang TCP. Pada pengujian *background traffic* 90%, terlihat bahwa TCP lebih unggul dibandingkan PCC karena disebabkan dalam waktu pengujian 10 detik dan perubahan *window* pada TCP yang menyebabkan *throughput* yang dihasilkan lebih besar dari PCC.

### 4.1.2 *Delay*





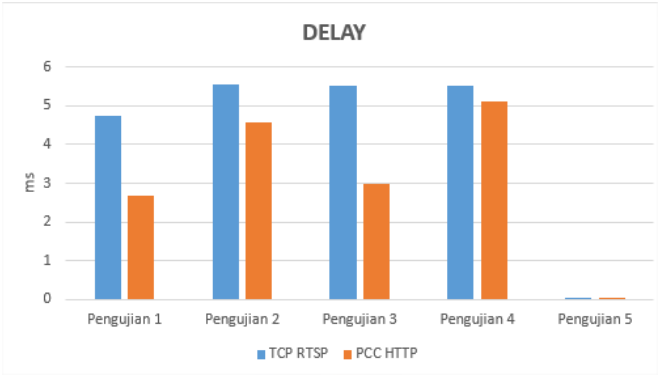

**(e)**

**Gambar 7 (a) Pengujian 1 : *Delay* Tanpa *Background Traffic* (b) Pengujian 2 : *Delay* Menggunakan *Background Traffic* 10 MB/s (c) Pengujian 3 : *Delay* Menggunakan *Background Traffic* 20 MB/s (d) Pengujian 4 : *Delay* Menggunakan *Background Traffic* 30 MB/s (e) Pengujian 5 : *Delay* menggunakan *background traffic* 900 MB**

**Tabel 6 Rincian *delay* dari setiap skenario pengujian**

| No | Skenario | Pengujian 1 | Protokol | *Client* 1 | *Client* 2 | *Client* 3 |
|---|---|---|---|---|---|---|
| 1 | Tanpa *Background Traffic* | Pengujian 1 | TCP | 5,594 ms | 5,585 ms | 3,006 ms |
| | | | PCC | 2,946 ms | 2,205 ms | 2,891 ms |
| 2 | Menggunakan *Background Traffic* | Pengujian 2 | TCP | 5,594 ms | 5,550 ms | 5,507 ms |
| | | | PCC | 7,593 ms | 3,210 ms | 2,924 ms |
| | | Pengujian 3 | TCP | 5,491 ms | 5,521 ms | 5,527 ms |
| | | | PCC | 2,924 ms | 3,149 ms | 2,924 ms |
| | | Pengujian 4 | TCP | 5,523 ms | 5,461 ms | 5,554 ms |
| | | | PCC | 9,716 ms | 2,782 ms | 2,851 ms |
| | | Pengujian 5 | TCP | 0,005 ms | - | - |
| | | | PCC | 0,004 ms | - | - |

Setelah mendapatkan hasil dari setiap pengujian kemudian *delay* dari setiap pengujian di rata-rata untuk melihat hasil dari setiap pengujian secara keseluruhan.




**Gambar 8 Rata-rata *delay* dari setiap pengujian**

**Tabel 7 Rincian rata-rata *delay* dari setiap pengujian**

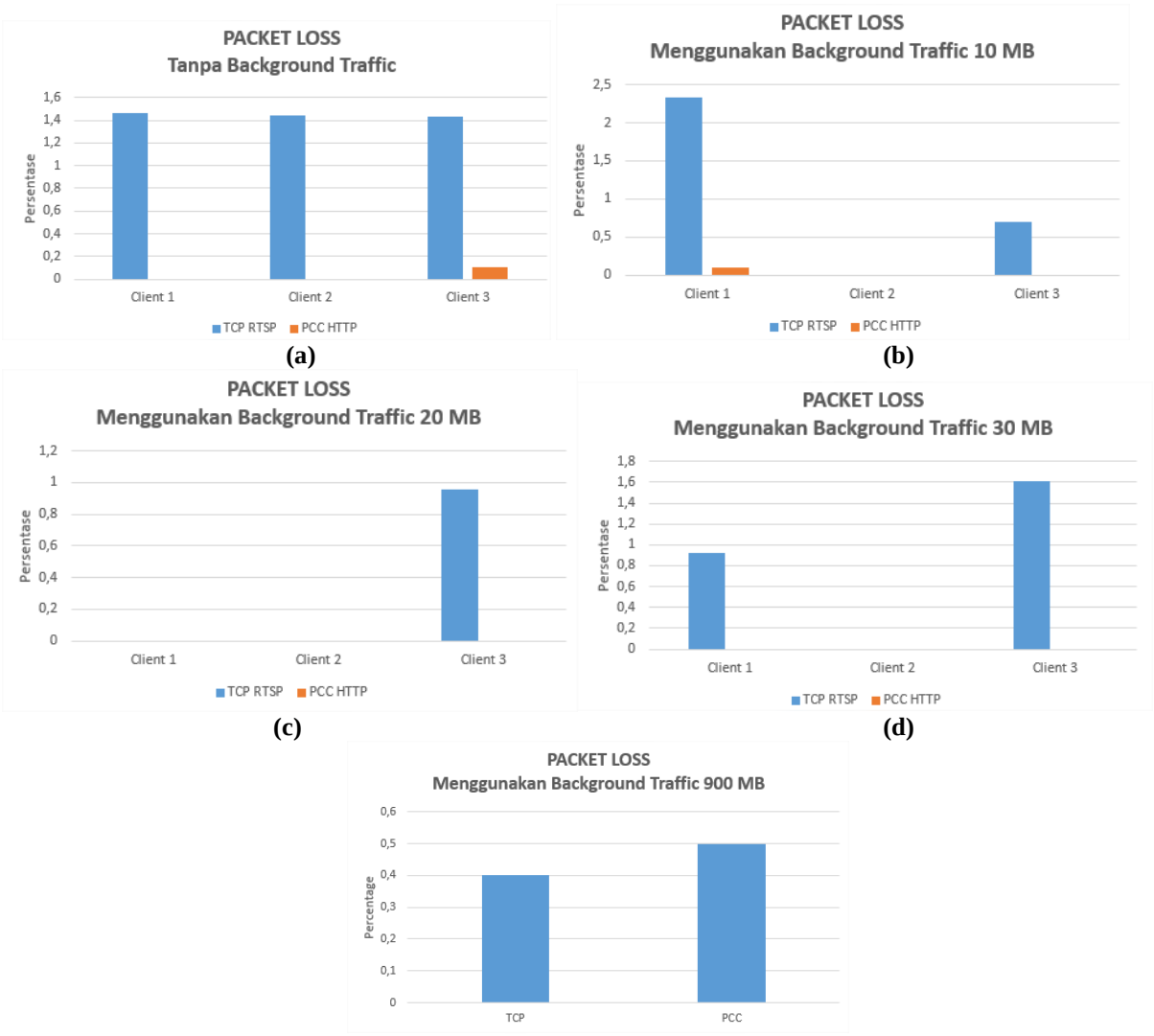
| No | Protokol | Pengujian 1 | Pengujian 2 | Pengujian 3 | Pengujian 4 |
|---|---|---|---|---|---|
| 1 | TCP | 4,728 ms | 5,551 ms | 5,513 ms | 5,513 ms |
| 2 | PCC | 2,681 ms | 4,576 ms | 2,999 ms | 5,116 ms |

Berdasarkan dengan gambar 8 dan tabel 7, diketahui bahwa saat pengujian 1 tanpa menggunakan *background traffic*, PCC memberikan hasil yang lebih baik yaitu 2,681 ms dari *delay* TCP yaitu 4,728 ms. Hal ini disebabkan karena *packet loss* yang diterima lebih kecil dari TCP dan terdapat modul monitor yang menghasilkan metriks

yang digunakan dalam fungsi utilitasnya. Sedangkan TCP disebabkan karena terjadi pertumbuhan laju cwnd membuat paket yang dikirim banyak membuat antrian menjadi penuh.

Pada skenario menggunakan *background traffic*, PCC mendapatkan hasil yang lebih baik yaitu 9,280 ms sedangkan TCP mendapatkan 12,391 ms. Hal ini disebabkan karena *packet loss* yang didapat kecil dan terdapat modul monitor yang menghasilkan metriks yang digunakan. Sedangkan TCP karena pada saat adanya *background traffic*, delay yang didapat lebih besar dibandingkan tanpa *background traffic*, dikarena karena antrian paket lebih besar saat kondisi yang tidak sibuk. Penyebab lainnya adalah perubahan *cwnd* ketika terjadinya *packet loss*. ketik

### 4.1.3 *Packet Loss*



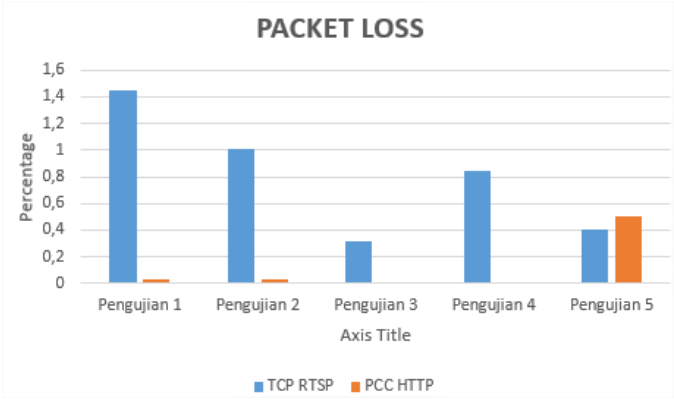

**Gambar 9 (a) Pengujian 1 : *Packet loss* Tanpa *Background Traffic* (b) Pengujian 2 : *Packet loss* Menggunakan *Background Traffic* 10 MB/s (c) Pengujian 3 : *Packet loss* Menggunakan *Background Traffic* 20 MB/s (d) Pengujian 4 : *Packet loss* Menggunakan *Background Traffic* 30 MB/s (e) Pengujian 5 : *Packet loss* menggunakan *background traffic* 900 MB**

**Tabel 8 Rincian *delay* dari setiap skenario pengujian**

| No | Skenario | Pengujian | Protokol | *Client* 1 | *Client* 2 | *Client* 3 |
|---|---|---|---|---|---|---|
| 1 | Tanpa *Background Traffic* | Pengujian 1 | TCP | 1,46 % | 1,44 % | 1,43 % |
| | | | PCC | 0 % | 0 % | 0,1 % |
| 2 | Menggunakan *Background Traffic* | Pengujian 2 | TCP | 2,34 % | 0 % | 0,7 % |
| | | | PCC | 0,1 % | 0 % | 0 % |
| | | Pengujian 3 | TCP | 0 % | 0 % | 0,96 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | PCC | 0 % | 0 % | 0 % |
| | | Pengujian 4 | TCP | 0,92 % | 0 % | 1,61 % |
| | | | PCC | 0 % | 0 % | 0 % |
| | | Pengujian 4 | TCP | 0,4 % | - | - |
| | | | PCC | 0,5 % | - | - |

Setelah mendapatkan hasil dari setiap pengujian kemudian *packet loss* dari setiap pengujian di rata-rata untuk melihat hasil dari setiap pengujian secara keseluruhan.




**Gambar 10 Rata-rata *packet loss* dari setiap pengujian**

**Tabel 9 Rincian rata-rata *packet loss* dari setuap pengujian**

| No | Protokol | Pengujian 1 | Pengujian 2 | Pengujian 3 | Pengujian 4 |
|---|---|---|---|---|---|
| 1 | TCP | 1,433 % | 1,013 % | 0,32 % | 0,843 % |
| 2 | PCC | 0,033 % | 0,033 % | 0 % | 0 % |

Dari gambar 10 dan tabel 9, diketahui dari pengujian 1 tanpa *background traffic* PCC mendapat *packet loss* yang lebih baik yaitu 0,033% dari TCP yaitu 1,433%. Hal ini disebabkan karena saat terjadi *packet loss*, PCC tidak keluar dari *starting state* nya melainkan PCC memonitor hasil utilitas. mekanisme PCC akan keluar dari *starting state* ketika fungsi utilitasnya menurun. Sedangkan ketika TCP Cubic mengalami *packet loss*, maka akan ada perubahan pada *cwnd* yang membuat parameter yang dihasilkan mengalami penurunan kinerja salahs atunya *packet loss.*

Pada pengujian dengan menggunakan *background traffic*, PCC memberikan hasil yang lebih kecil dan baik yaitu 0,033 % dan TCP yaitu 1,612 %. Hal ini disebabkan selama ketika terjadinya paket yang hilang SACK akan langsung mengirimkan paket yang benar-benar hilang. Sedangkan TCP Cubic, ketika terjadi *loss event* membuat perubahan pada *cwnd*. Sehingga ketika *event* ini terjadi TCP Cubic menggunakan mekanisme nya yang berfokus pada perubahan *cwnd*, membuat parameter lain yang dihasilkan mengalami penurunan kinerja salah satunya *packet loss*.

## 5. Kesimpulan

Kesimpulan dari hasil pengujian juga analisis yang dilakukan berdasarkan skenario yang sudah dibuat adalah sebagai berikut :

a. Hasil pengujian TCP dan PCC dengan parameter *throughput* memberikan hasil bahwa pada saat keaadan tanpa *background traffic*. *throughput* yang dihasillkan PCC lebih besar daripada TCP yaitu 3643,334 kbps dan 3603,667 kbps. Ketika skenario menggunakan *background traffic*, *throughput* TCP lebih besar dari PCC yaitu 5733,111 kbps dan 3397,111 kbps. Sedangkan berdasarkan dari rata-rata semua pengujian yaitu TCP 1064,842 kbps dan PCC 150,825 kbps. Pengujian dengan utilitas *background traffic* 90%, menghasilkan TCP 4141 kbps dan PCC 3974 kbps. Sehingga kinerja TCP dalam menghasilkan *throughput* lebih baik dari PCC.

b. Hasil pengujian dengan parameter *delay* pada skenario tanpa *background traffic* yaitu *delay* dari PCC lebih kecil dari TCP yaitu 2,681 ms dan 4,728 ms, begitu juga pada skenario menggunakan *backgrund traffic* yaitu PCC 4,231 ms dan TCP 5,525 ms. Berdasarkan dari rata-rata semua pengujian yaitu TCP adalah 5,326 ms dan PCC adalah 3,843 ms. Pengujian dengan utilitas *background traffic* 90%,

menghasilkan TCP 0,005 ms dan PCC 0,004 ms. Sehingga dalam parameter *delay,* PCC lebih baik kinerja daripada TCP.

c. Hasil pengujian TCP dan PCC dengan parameter *packet loss* memberikan hasil bahwa pada saat keadaan tanpa *background traffic, packet loss* yang dihasillkan PCC lebih kecil daripada TCP yaitu 0,033 % dan 1,433 %. Ketika skenario menggunakan *background traffic, packet loss* TCP lebih besar dari PCC yaitu 0,725 % dan 0,111 %. Berdasarkan dari rata-rata semua pengujian yaitu TCP 0,905% dan PCC 0,016 %. Pengujian dengan utilitas *background traffic* 90%, menghasilkan TCP 0,4% dan PCC 0,5% Sehingga kinerja PCC dalam menghasilkan *packet loss* lebih baik daripada TCP, namun ketika utilitas *background traffic* 90% TCP menghasilkan *packet loss* yang lebih baik.

**Daftar Pustaka**

[1] T. by Geoff Huston, “TCP Performance - The Internet Protocol Journal - Volume 3, No. 2.” [Online]. Available: https://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-5/ipj-archive/article09186a00800c8417.html.

[2] D. Westwood, “Analisa Kinerja Algoritma TCP Congestion Control Cubic , Reno , Vegas,” vol. 2, no. 3, pp. 1099–1108, 2018.

[3] M. Dong, Q. Li, D. Zarchy, and I. Nsdi, “PCC : Re-architecting Congestion Control for Consistent High Performance This paper is included in the Proceedings of the,” 2015.

[4] M. Dong, “PCC: Performance-oriented Congestion Control Dramatically higher performance data delivery with a flexible transport architecture,” 2014. [Online]. Available: https://modong.github.io/pcc-page/.

[5] N. Jay *et al.*, “A PCC-Vivace Kernel Module for Congestion Control.”

[6] S. Becker and S. Becker, “EXPERIMENTS ON VIDEO STREAMING OVER COMPUTER NETWORKS by,” 2012.

[7] D. Price, “How Much Data Does Streaming Video Use?,” 2018. [Online]. Available: https://www.makeuseof.com/tag/how-much-data-does-streaming-video-use/.

[8] “VLC Media Player.” [Online]. Available: https://wiki.videolan.org/VLC_media_player/.

[9] Wireshark, “Wireshark User ’ s Guide.”

[10] D. Manual, A. Botta, W. De Donato, A. Dainotti, S. Avallone, and A. Pescap, “D-ITG 2.8.1 Manual,” pp. 1–35, 2013.

[11] A. Hoc, A. Firnanda, and T. Y. Arif, “Jurnal Rekayasa Elektrika,” vol. 13, no. 36, 2017.

[12] T. KEARY, “Throughput vs Bandwidth: Understanding the Difference Plus Tools,” 2018. [Online]. Available: https://www.comparitech.com/net-admin/throughput-vs-bandwidth/.

[13] A. I. Diwi, R. M. Rumani, and I. Wahidah, “Analisis Kualitas Layanan Video Live Streaming pada Jaringan Lokal Universitas Telkom Quality of Service Analysis for Live Streaming Video Services on Telkom University Local Network,” pp. 207–216, 2014.

[14] P. Studi, T. Informatika, J. Teknik, E. Fakultas, and T. Universitas, “ANALISIS QOS ( QUALITY OF SERVICE ) PADA JARINGAN INTERNET ( STUDI KASUS : FAKULTAS TEKNIK UNIVERSITAS TANJUNGPURA ).”

[15] L. Jie, “Performance Evaluation of Different TCP Congestion Control Schemes in 4G System,” 2013.