JISKA (Jurnal Informatika Sunan Kalijaga)
Vol. 7, No. 1, JANUARI, 2022, Pp. 20 – 32
ISSN: 2527 – 5836 (print) | 2528 – 0074 (online)

Analisis Perbandingan Kinerja TCP Vegas dan TCP New Reno Menggunakan Antrian *Drop Tail*

Dony Fahrudy [(1)*], Bambang Sugiantoro [(2)]
Magister Informatika, Fakultas Sains dan Teknologi, UIN Sunan Kalijaga, Yogyakarta
e-mail : donnyfahrudi@gmail.com, bambang.sugiantoro@uin-suka.ac.id.
* Penulis korespondensi.
Artikel ini diajukan 5 Juli 2021, direvisi 6 September 2021, diterima 9 September 2021, dan dipublikasikan 25 Januari 2022.

Abstract

TCP was developed to deal with problems that often occur in the network, such as congestion problems. Congestion can occur when the number of packets transmitted in the network approaches the network capacity which can cause network problems. This can be overcome by implementing TCP and queue management. In this research, we will test the performance of TCP Newreno and TCP Vegas using NS-2 in the Drop Tail queue. The performance parameters used are throughput, packet drop, and congestion window with additional buffer capacity. The test results for the congestion window and packet drop parameters, TCP Vegas has better performance when the buffer gets bigger when congestion occurs with the congestion window smaller than TCP New Reno and the average packet drop is 18.33 packets compared to TCP New Reno with an average of 18.33 packets. average 41.67 packets. For throughput parameters, TCP New Reno has better performance with an average of 6.77253 Mbps than TCP Vegas with an average of 4.29693 Mbps. From testing and analysis that TCP Vegas has better performance than TCP New Reno when using Drop Tail queues.

Keywords: Drop Tail, NS-2, TCP Newreno, TCP Vegas, Packet Data Queuing

Abstrak

TCP dikembangkan untuk menangani masalah yang sering terjadi dalam jaringan, seperti masalah *congestion*. *Congestion* dapat terjadi apabila jumlah paket yang ditransmisikan dalam jaringan mendekati kapasitas jaringan yang dapat menyebabkan masalah jaringan. Hal tersebut dapat diatasi dengan menerapkan TCP dan manajemen antrian. Pada penelitian ini akan dilakukan pengujian kinerja TCP Newreno dan TCP Vegas menggunakan NS-2 pada antrian *Drop Tail*. Parameter kinerja yang digunakan adalah *throughput*, *packet drop* dan *congestion window* dengan penambahan kapasitas *buffer*. Hasil pengujian untuk parameter *congestion window* dan *packet drop*, TCP Vegas memiliki kinerja lebih baik ketika *buffer* semakin besar saat terjadi *congestion* dengan *congestion window* lebih kecil dibandingkan TCP New Reno dan rata-rata *packet drop* lebih kecil yaitu 18.33 *packet* dibandingkan TCP New Reno dengan rata-rata 41.67 *packet*. Untuk parameter *throughput*, TCP New Reno memiliki kinerja lebih baik dengan rata-rata 6.77253 Mbps dibandingkan TCP Vegas dengan rata-rata 4.29693 Mbps. Dari pengujian dan analisis bahwa TCP Vegas memiliki kinerja yang lebih baik dari TCP New Reno ketika menggunakan antrian *Drop Tail*.

Kata Kunci: Drop Tail, NS-2, TCP Newreno, TCP Vegas, Antrian Paket Data

1. PENDAHULUAN

TCP/IP (Transmission Control Protocol/Internet Protocol) merupakan standar komunikasi jaringan yang dapat digunakan untuk proses pertukaran data dari satu komputer ke komputer lain dalam jaringan Internet (Syaifuddin et al., 2016). TCP dan IP saling berhubungan, dikarenakan kedua protokol dijadikan satu nama sebab fungsinya yang saling bekerja sama dalam melakukan komunikasi data. Saat ini, telah banyak varian dari TCP, di antaranya TCP Tahoe, TCP Reno, TCP New Reno, TCP Vegas dan TCP SACK (Chaudhary & Kumar, 2017).

TCP banyak dikembangkan untuk menangani masalah yang sering terjadi dalam jaringan, seperti masalah kemacetan atau *congestion* (Kembuan, 2016). Kemacetan pada jaringan internet

pertama kali dialami pada akhir tahun 80-an, saat itu menurun drastis sehingga terjadi kolaps atau jatuh, karena pada saat itu belum adanya mekanisme yang dapat menangani hal tersebut. Pada tahun 1986 pertama terjadi *congestion collapse*, kemudian pada tahun 1988 Jacobson mengusulkan teorinya yaitu *Congestion Avoidance and Control* (Fajri, 2016). *Congestion* dapat terjadi apabila jumlah paket yang ditransmisikan sepanjang jaringan mulai mendekati kapasitas jaringan dalam menangani paket-paket, sehingga menyebabkan beberapa masalah yaitu meningkatnya *packet loss*, melambatnya transmisi bahkan dapat menyebabkan kelumpuhan pada jaringan. Oleh karena itu, dibutuhkan sebuah teknik untuk dapat mengontrol kongesti agar tetap mempertahankan jumlah paket data dalam jaringan pada saat kinerja menurun drastis (Kembuan, 2016). Untuk mengatasi hal tersebut dalam penelitian ini akan akan diterapkan TCP dan manajemen antrian yang tepat. TCP Vegas dan TCP New Reno memiliki mekanisme berbeda dalam mengatasi *congestion*, TCP Vegas memiliki sifat proaktif karena dapat menghindari *congestion* sebelum terjadi penumpukan data pada jaringan (Brakmo & Peterson, 1995), sedangkan TCP New Reno memiliki sifat reaktif karena dapat mengendalikan *congestion* ketika terjadi penumpukan data dalam jaringan (Torkey et al., 2012). Untuk manajemen antrian yang digunakan adalah *Drop Tail* yang memiliki mekanisme karakteristik sendiri dalam melayani data pada antrian.

Mekanisme TCP Vegas dalam mengatasi *congestion* lebih kepada *packet delay*. TCP Vegas melihat varian *Round-trip time* (RTT) yang ada dalam mendeteksi *congestion* (Brakmo & Peterson, 1995), RTT merupakan banyaknya waktu yang dibutuhkan oleh suatu paket untuk melakukan perjalanan dari *host* pengirim ke *host* tujuan kemudian kembali lagi ke pengirimnya (Susandi & Pinem, 2014). Jika RTT besar maka jaringan dianggap mengalami *congestion*, sehingga *congestion window* akan dikurangi yang dapat menyebabkan pengiriman data berkurang, jika RTT kecil maka jaringan dianggap dalam keadaan normal, sehingga *congestion window* ditambah yang dapat menyebabkan pengiriman data bertambah (Brakmo & Peterson, 1995). Sedangkan TCP New Reno akan mengirimkan data secara eksponensial dalam jaringan sampai terjadi *packet loss* karena terjadi *congestion*, ketika *packet loss* terdeteksi maka *congestion window* akan dikurangi sampai setengah (Torkey et al., 2012).

Untuk manajemen antrian menggunakan mekanisme antrian *Drop Tail*. Pada antrian *Drop Tail* menggunakan manajemen *First In First Out* (FIFO), dalam hal ini data yang pertama datang akan dilayani, sedangkan apabila kapasitas *buffer* penuh, maka data yang datang akan langsung di drop (Kumar et al., 2012).

Pada penelitian sebelumnya membandingkan kinerja TCP Vegas dan TCP Newreno menggunakan ns-2 dengan penambahan kapasitas *buffer* yang berbeda, penelitian tersebut menggunakan topologi *Abilene* dengan total 31 *node*, didapatkan hasil pengujian *throughput* TCP New Reno memiliki kinerja terbaik ketika menggunakan *Drop Tail* dengan rata-rata 100749.4234kbps sedangkan 94576.815kbps untuk TCP Vegas. Untuk parameter *packet drop*, TCP Vegas memiliki kinerja lebih baik ketika menggunakan antrian *Random Early Detection* dengan rata-rata 0.0002 % dibandingkan dengan TCP New Reno memiliki nilai rata-rata 0.319 %. Dari hasil tersebut dapat disimpulkan bahwa TCP Vegas memiliki kinerja yang lebih baik dari TCP New Reno ketika menggunakan antrian *Random Early Detection* (Pamungkas et al., 2018). Sedangkan pada penelitian lain yang telah dilakukan oleh (Domański et al., 2016), dengan mengevaluasi melalui pendekatan keefektifan kontrol kemacetan TCP New Reno dan TCP Vegas terhadap Perkiraan Aliran Fluida diperoleh hasil bahwa TCP Vegas lebih baik dalam menerapkan algoritma secara adil dibandingkan dengan algoritma TCP Newreno.

Penelitian sebelumnya oleh (Pratama et al., 2015) tentang perbandingan model antrian PCQ, SFQ, RED Dan FIFO Pada Mikrotik Sebagai Upaya Optimalisasi Layanan Jaringan Pada Fakultas Teknik Universitas Tanjungpura didapat hasil bahwa metode antrian *Drop Tail* (FIFO) merupakan metode yang paling optimal untuk di implementasikan pada jaringan internet Fakultas Teknik Universitas Tanjungpura. Penelitian lain juga yang dilakukan oleh (Fajri, 2016) dapat dilihat bahwa *Drop Tail* adalah solusi antrian yang bekerja dengan baik dalam mengatasi antrian

dalam *buffer management* berdasarkan 3 karakteristik yang baik yaitu pada *Packet Dropped*, Pengiriman Ulang, dan *Buffer Usage*.

Dalam penelitian ini akan dilakukan simulasi untuk membandingkan kinerja TCP Vegas dan TCP New Reno dengan menggunakan antrian *Drop Tail* pada NS-2. Simulasi menggunakan topologi *dumbbell* yang diasumsikan sebagai jaringan kabel. Pengujian akan dilakukan dengan skema penambahan kapasitas *buffer* untuk antrian *Drop Tail*. Setelah dilakukan penelitian ini, diharapkan dapat mengetahui varian TCP mana yang memiliki kinerja terbaik ketika dalam jaringan terjadi *congestion* pada antrian *Drop Tail*. Tujuan dari penelitian ini yaitu mengetahui perbandingan kinerja TCP Vegas dan TCP New Reno pada antrian *Drop Tail* dengan melihat nilai dari parameter *throughput*, *packet drop* (*loss*) dan *congestion window*.

2. METODE PENELITIAN

Metode penelitian yang dilakukan merupakan alur dari penelitian yang akan di kerjakan terhadap simulasi jaringan. Penelitian ini dilakukan untuk membangun kerangka algoritma dari masing-masing protokol TCP Vegas dan TCP New Reno. Kerangka penelitian ini dikembangkan dan diimplementasikan pada simulasi yang dijalankan pada *network simulator* NS-2 yang akan digunakan untuk melakukan pengujian dari masing-masing algoritma protokol TCP dengan teknik simulasi topologi *Dumbbell*. Berikut rancangan simulasi jaringan pada penelitian ini.

2.1 Topologi Simulasi

Topologi yang digunakan adalah topologi *dumbbell* dengan 2 node sebagai pengirim yaitu *node* 0 dan *node* 1, 2 *node* sebagai *router* yaitu *node* 2 dan *node* 3, dan 2 *node* sebagai penerima yaitu *node* 4 dan *node* 5. Topologi ini digunakan untuk mengamati kinerja dari TCP Vegas dan TCP New Reno. Berikut rancangan topologi yang digunakan seperti terlihat pada Gambar 1.

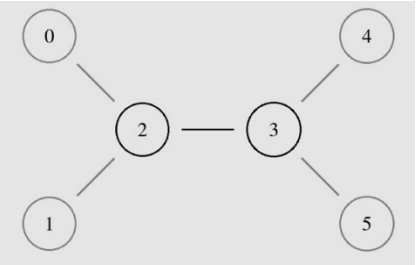
2.2 Parameter Simulasi

Pada penelitian ini ditentukan parameter-parameter jaringan yang bersifat konstan (tetap) yang akan digunakan pada simulasi TCP Vegas dan TCP New Reno menggunakan antrian *Drop Tail*. Berikut parameter-parameter simulasi jaringan yang digunakan seperti pada Tabel 1.

Untuk menentukan *buffer*, peneliti menggunakan perhitungan *Bandwidth Delay Product* pada *link router node* 2 dan *link router node* 3 dengan perhitungan sebagai berikut (PERFSONAR, 2020):

Bandwidth Delay Product = *Bandwidth* * RTT (*delay*)
= 12 Mb * 10 ms
= 12 Mb * 0,01 s
= 0.12 Mbps

Buffer:
1500 *Byte* = 0.012 Mb
0.012 Mb * 8 = 0.096 Mb (<<<<< *bandwidth Delay product*)
0.12 * 10 = 0.12 Mb (===== *bandwidth Delay product*)
0.13 * 12 = 0.144 Mb (>>>>> *bandwidth Delay product*)



Gambar 1 Topologi Simulasi

Tabel 1 Parameter Simulasi

Parameter Simulasi	Nilai
Waktu Simulasi	120 detik
Jumlah Host	4
Link Node n0-n2	Bandwidth: 12Mb Delay: 10ms
Link Node n1-n2	Bandwidth: 12Mb Delay: 10ms
Link Node n2-n3	Bandwidth: 12Mb Delay: 10ms
Link Node n3-n4	Bandwidth: 12Mb Delay: 10ms
Link Node n3-n5	Bandwidth: 12Mb Delay: 10ms
Protocol Transport	TCP Newreno, TCP Vegas
Model Antrian	Drop Tail
Traffic Source	TCP vs TCP
Ukuran Buffer pada Router	8, 10, 12
TCP window size	1000

2.3 TCP New Reno

TCP New Reno menggabungkan berbagai algoritma *congestion control* untuk mengatasi kemacetan diantaranya sebagai berikut (Torkey et al., 2012).

2.3.1 Slow Start and Congestion Avoidance

Pada TCP NewReno, ketika koneksi TCP dimulai, algoritma *Slow Start* menginisialisasi jendela kemacetan (cwnd) ke satu segmen. Pengirim kemudian mulai mengirimkan paket berdasarkan ukuran jendela dan jendela kemacetan bertambah satu segmen hingga cwnd mencapai ambang *slow start* (ssthresh). Pada titik ini, NewReno memasuki fase *Congestion Avoidance* untuk memperlambat laju peningkatan cwnd. Selama penghindaran kemacetan, jendela kemacetan meningkat secara linier satu segmen setiap *round trip time* (RTT) selama kemacetan jaringan tidak terdeteksi (Torkey et al., 2012).

2.3.2 Fast Retransmit and Fast Recovery

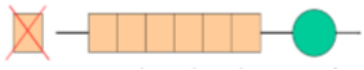
Selama penghindaran kemacetan, berakhirnya penghitung waktu transmisi ulang memberi sinyal kepada pengirim bahwa terjadi kemacetan jaringan. Sehingga pengirim harus memperlambat laju transmisinya. Jika kemacetan ditunjukkan oleh batas waktu, ssthresh diatur ke setengah dari jendela kemacetan saat ini dan jendela kemacetan diatur ke satu segmen dan pengirim masuk ke fase *slow start*. Pengirim masuk ke mode *Fast Retransmit* untuk mentransmisikan ulang paket yang hilang. Kemudian, pengirim menyetel ssthresh ke setengah dari jendela kemacetan dan masuk ke fase *Fast Recovery*. Saat memasuki *Fast Recovery*, pengirim menambah jendela kemacetan satu segmen untuk setiap ACK (*Acknowledgement*) duplikat berikutnya yang diterima. Selama Fast Recovery, TCP NewReno membedakan antara ACK sebagian dan ACK penuh. TCP New Reno berlanjut di *Fast Recovery* sampai semua paket yang beredar selama awal *Fast Recovery* telah diakui. Saat menerima ACK penuh, pengirim menyetel jendela kemacetan (cwnd) ke ssthresh, mengakhiri *Fast Recovery*, dan melanjutkan Penghindaran Kemacetan (Torkey et al., 2012).

2.4 TCP Vegas

TCP Vegas mengontrol ukuran jendelanya dengan mengamati RTT (*round-trip times*) dari paket yang telah dikirim oleh *host* pengirim sebelumnya. Jika RTT yang diamati besar, TCP Vegas mengenali bahwa jaringan mulai padat, dan membatasi ukuran jendela. Jika RTT menjadi kecil, host pengirim TCP Vegas menentukan bahwa jaringan dibebaskan dari kemacetan, dan meningkatkan ukuran jendela lagi. TCP Vegas memiliki fitur lain dalam algoritma *congestion control* yaitu mekanisme *slow-start*. Ukuran jendela bertambah setiap kali paket ACK diterima. Mekanisme *congestion control* yang digunakan oleh TCP Vegas menunjukkan bahwa jika RTT yang diamati dari paket identik, ukuran jendela tetap tidak berubah (Kurata et al., 2000).

2.5 Konfigurasi *Drop Tail*

Drop Tail adalah antrian yang menggunakan algoritma *First In First Out* (FIFO). Mekanismenya yaitu jika ada paket yang pertama kali datang akan dilayani, namun jika *buffer* pada antrian penuh, maka data yang datang akan langsung di *drop* sampai *buffer* memiliki ruang kosong untuk paket yang baru (Kumar et al., 2012). Konfigurasi *Drop Tail* di terapkan pada *node* topologi terhadap *router* dengan *bandwidth* dan *delay* yang telah ditentukan untuk menangani kondisi antrian atau *buffer* dalam jaringan. Berikut ilustrasi *Drop Tail* seperti terlihat pada Gambar 2.

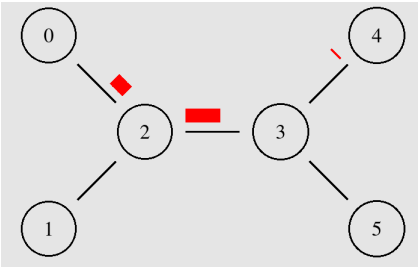


Gambar 2 Ilustrasi *Drop Tail.*

2.6 Skenario Simulasi

2.6.1 Newreno vs Newreno

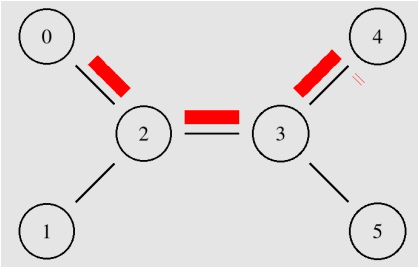
Skenario ini akan menguji congestion control pada TCP Newreno vs TCP Newreno menggunakan topologi *dumbbell* pada antrian *Drop Tail*. Simulasi dimulai dengan menjalankan *traffic* Newreno1 dari n0 menuju n4 dan traffic Newreno2 dari n1 menuju n5 sebagai *traffic* pengganggu. Simulasi ini menggunakan nilai tetap dari *bandwidth* dan *delay* dengan *buffer* = 10. *Traffic* Newreno2 akan mengganggu *flow* jaringan pada detik ke 60. Berikut rancangan topologi pada skenario Newreno vs Newreno seperti terlihat pada Gambar 3.



Gambar 3 Rancangan Topologi Skenario 1

2.6.2 Vegas vs Vegas

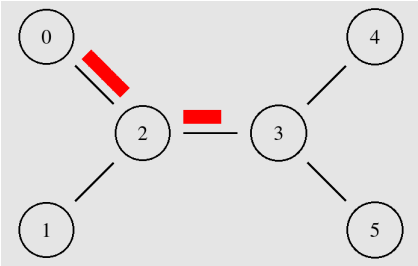
Skenario ini akan menguji *congestion control* pada TCP Vegas vs TCP Vegas menggunakan topologi *dumbbell* pada antrian *Drop Tail*. Simulasi dimulai dengan menjalankan *traffic* Vegas1 dari n0 menuju n4 dan traffic Vegas2 dari n1 menuju n5 sebagai *traffic* pengganggu. Simulasi ini menggunakan nilai tetap dari *bandwidth* dan *delay* dengan buffer = 10. *Traffic* Vegas2 akan mengganggu *flow* jaringan pada detik ke 60. Berikut rancangan topologi pada skenario Vegas vs Vegas seperti terlihat pada Gambar 4.



Gambar 4 Rancangan Topologi Skenario 2

2.6.3 Newreno vs Vegas

Skenario ini akan menguji *congestion control* pada TCP Newreno vs TCP Vegas menggunakan topologi *dumbbell* pada antrian *Drop Tail*. Simulasi dimulai dengan menjalankan *traffic* Newreno dari n0 menuju n4 dan *traffic* Vegas dari n1 menuju n5 sebagai *traffic* pengganggu. Simulasi ini menggunakan nilai tetap dari *bandwidth* dan *delay* dengan buffer yang berbeda yaitu 8, 10, dan 12. *Traffic* Vegas akan mengganggu *flow* jaringan pada detik ke 60. Berikut rancangan topologi pada skenario Newreno vs Vegas seperti terlihat pada Gambar 5.



Gambar 5 Rancangan Topologi Skenario 3

2.7 Parameter Kinerja

2.7.1 Throughput

Throughput merupakan jumlah bit data per satuan waktu yang dikirim atau ditransmisikan ke suatu tujuan melalui jaringan. Semakin besar nilai *throughput* pada TCP maka akan semakin baik. Kualitas TCP dapat terlihat melalui besarnya *throughput* yang dihasilkan. Berikut adalah rumus untuk menghitung *throughput* (Hasanul Fahmi, 2018) seperti terlihat pada Pers. (1).

$$Throughput = \frac{ukuran\ data\ dikirim}{waktu\ pengiriman\ data} \quad (1)$$

2.7.2 Congestion Window

Congestion window (cwnd) adalah satuan dari jumlah paket yang dapat dikirim oleh TCP. *Congestion window* digunakan untuk membatasi jumlah data yang dikirim dalam satu *round trip time* (RTT). Pengirim akan menambah atau mengurangi ukuran *congestion window* terhadap jaringan pada kondisi *congestion* (Taruk & Setyadi, 2016).

2.7.3 Packet Drop

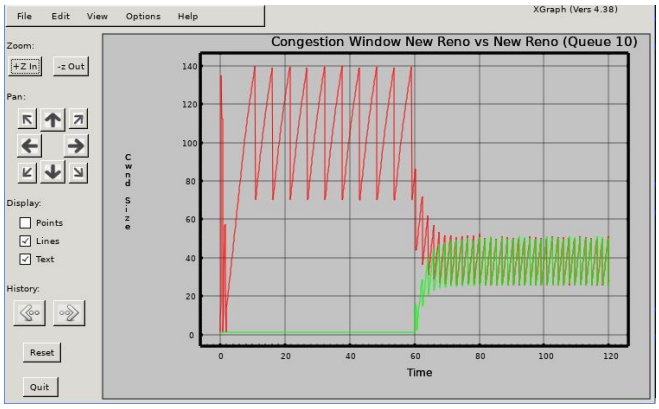
Packet drop adalah paket yang dibuang saat melewati *router*, paket tersebut dibuang dikarenakan *buffer* antrian penuh. Jumlah total paket tersebut akan dibuang dikarenakan paket hilang selama proses transmisi ke tujuan (Orueta et al., 2016). Semakin tinggi jumlah paket yang harus didrop, maka semakin rendah efektifitas penggunaan algoritma untuk paket yang memiliki *deadline* (Taruk & Ashari, 2016).

3. HASIL DAN PEMBAHASAN

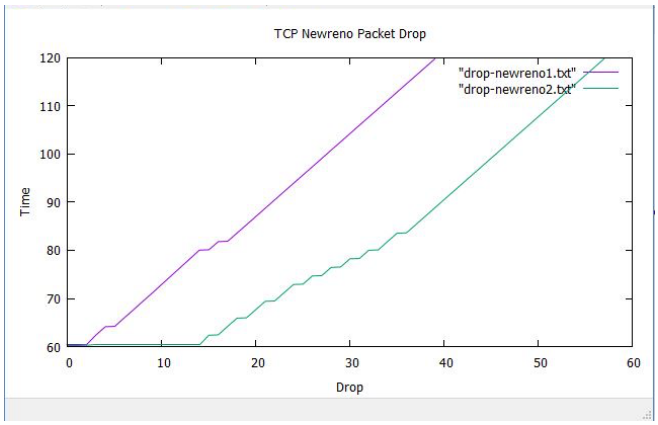
Pengambilan data terhadap pengujian simulasi seperti *congestion window*, nilai *throughput* serta jumlah dan waktu terjadinya *packet drop*, dilakukan sesuai skenario yang telah dirancang sebelumnya. Berikut hasil dari pengambilan data tersebut. Pada simulasi skenario masing-masing TCP, model antrian yang digunakan adalah *Drop Tail*. Pada jenis antrian *Drop Tail*, ketika antrian dalam keadaan penuh, maka paket yang datang akan langsung dibuang hingga terdapat antrian yang kosong.

Pada Gambar 6 dapat dilihat bahwa TCP Newreno1 ditandai dengan garis warna merah dan TCP Newreno2 ditandai dengan garis warna hijau. Pada gambar dapat terlihat bahwa gangguan yang

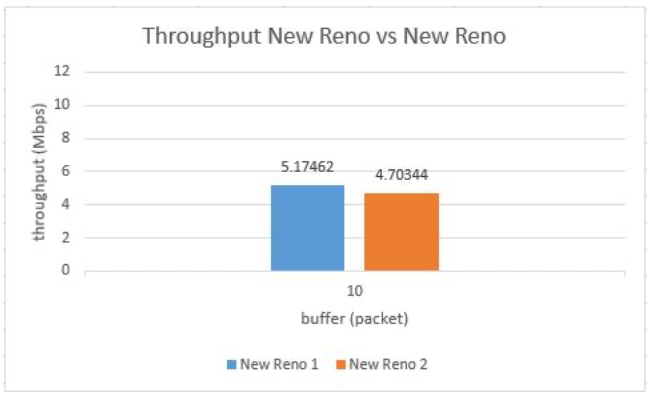
didapat pada TCP Newreno dimulai pada detik ke 60. Setiap paket yang datang dari TCP Newreno1 dan TCP Newreno2 akan ditampung di *buffer* antrian. Ketika antrian penuh maka TCP Newreno1 dan TCP Newreno2 akan dibuang bersamaan. Hal ini akan dilakukan secara berulang sehingga ukuran cwnd TCP Newreno1 sama dengan cwnd TCP Newreno2. Gambar 7 menunjukkan *packet drop* terjadi. Pada detik ke 60.321, TCP Newreno1 mengalami *timeout* karena jumlah *drop* lebih dari 2 berdekatan. Hal ini dikarenakan adanya *traffic* pengganggu TCP Newreno2 seperti yang didukung dengan *congestion window* pada Gambar 6. Selanjutnya, Gambar 8 menunjukkan bahwa *throughput* akan mengalami penurunan ketika adanya gangguan. Penuruan nilai *throughput* akan menjadi setengah dari nilai *bandwidth* yang telah diatur sebelumnya. Hal ini menunjukkan bahwa kedua TCP berbagi *bandwidth* dengan adil.



Gambar 6 *Congestion Window* Newreno vs Newreno

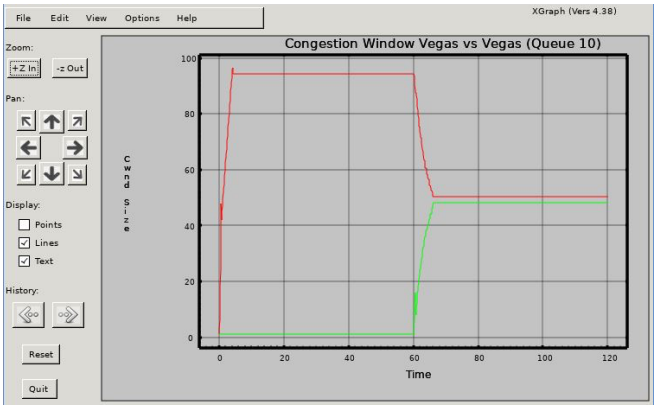


Gambar 7 *Packet Drop* Newreno vs Newreno

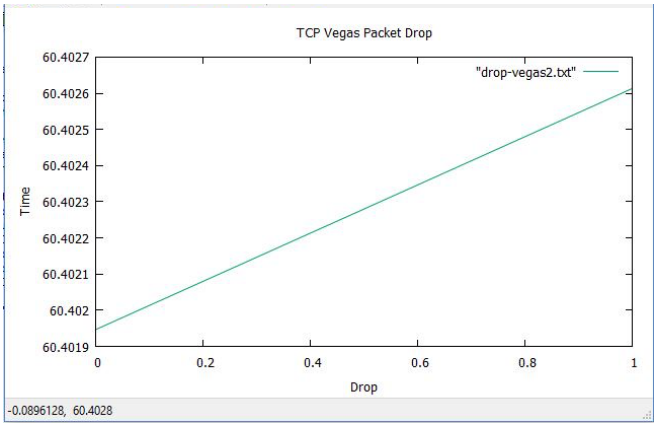


Gambar 8 *Throughput* Newreno vs Newreno

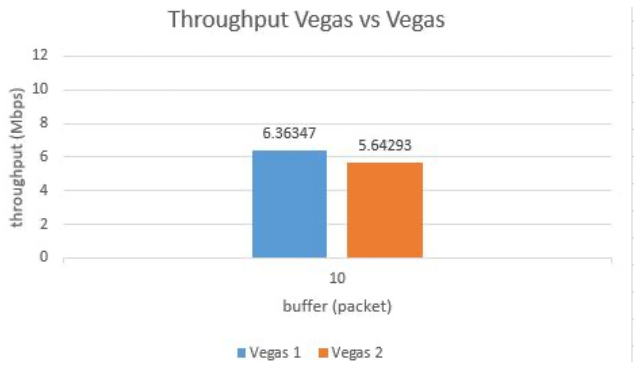
Pada Gambar 9 dapat dilihat bahwa TCP Vegas1 ditandai dengan garis warna merah dan TCP Vegas2 ditandai dengan garis warna hijau. Pada gambar menunjukkan bahwa adanya gangguan yang didapat pada TCP Vegas1 dimulai pada detik ke 60. TCP Vegas1 akan berusaha menurunkan nilai cwnd untuk menghandle adanya gangguan dari TCP Vegas2. Hal ini mengakibatkan adanya *drop* yang dialami oleh TCP Vegas2 pada detik ke 60.40194. Oleh karena itu, nilai cwnd dari TCP Vegas2 akan mendekati nilai cwnd dari TCP Vegas1, sehingga nilai dari kedua TCP seimbang. Selanjutnya, Gambar 11 menunjukkan bahwa *throughput* akan mengalami penurunan ketika adanya gangguan. Penuruan nilai *throughput* akan menjadi setengah dari *bandwidth* yang telah di atur sebelumnya. Hal ini menunjukkan bahwa kedua TCP berbagi *bandwidth* dengan adil.



Gambar 9 *Congestion Window* Vegas vs Vegas

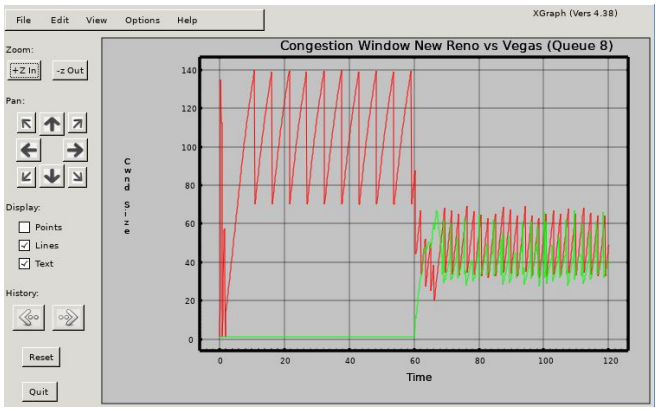


Gambar 10 *Packet Drop* Vegas vs Vegas

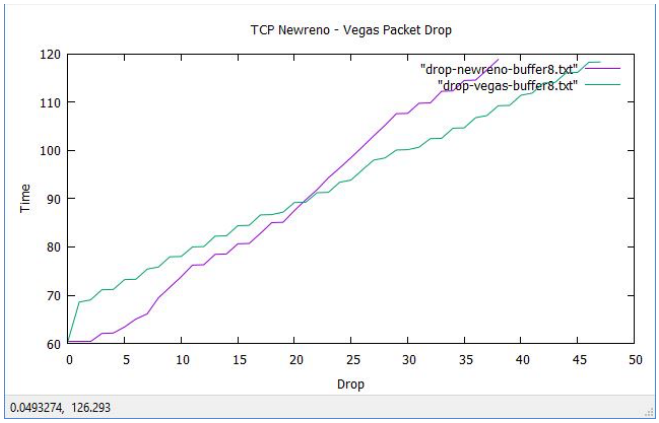


Gambar 11 *Throughput* Vegas vs Vegas

Pada Gambar 12 dapat dilihat bahwa TCP Newreno ditandai dengan garis warna merah dan TCP Vegas ditandai dengan garis warna hijau. Pada gambar menunjukkan bahwa *congestion window* dari TCP Newreno lebih kecil dari TCP Vegas. Hal ini dikarenakan cara kerja dari TCP Newreno dipengaruhi oleh adanya *packet loss* yang menyebabkan *congestion window* turun. Hal berbeda dialami oleh TCP Vegas. Pada TCP Vegas cara kerjanya dipengaruhi oleh variasi *delay* yang akan menyebabkan penurunan *congestion window*. Semakin kecil antrian *buffer* maka semakin kecil variasi *delay* yang dialami oleh TCP Vegas. Hal inilah yang menyebabkan *congestion window* TCP Vegas lebih besar dari TCP Newreno, hal tersebut didukung dengan banyaknya packet yang di *drop* pada TCP Vegas ketika jaringan mengalami *congestion* seperti pada Gambar 13.



Gambar 12 *Congestion Window* Newreno vs Vegas, *buffer* 8

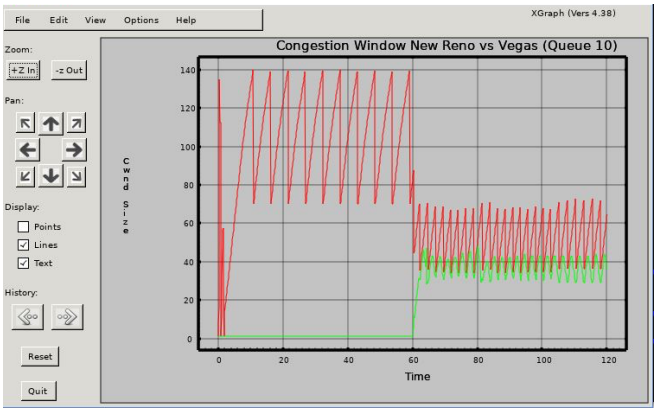


Gambar 13 *Packet Drop* Newreno vs Vegas, *buffer* 8

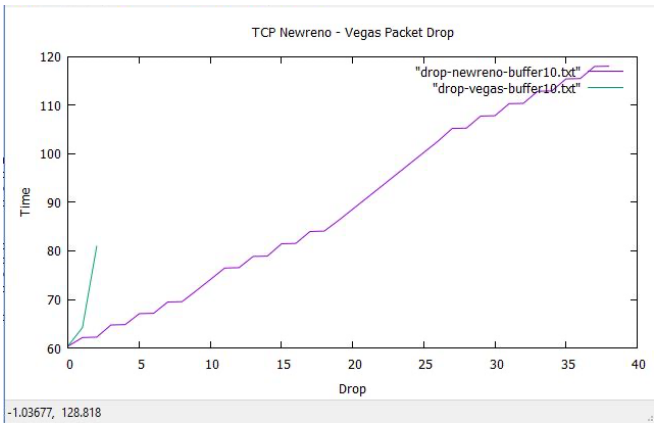
Pada Gambar 14 dapat dilihat bahwa TCP Newreno ditandai dengan garis warna merah dan TCP Vegas ditandai dengan garis warna hijau. Pada gambar menunjukkan bahwa *congestion window* dari TCP Newreno lebih besar dari TCP Vegas. Hal ini dikarenakan TCP New Reno dalam mengirimkan paket data tidak memperhatikan kondisi jaringan dengan mengirimkan paket data secara cepat, sehingga *congestion window* meningkat secara eksponential, namun ketika jaringan mengalami *congestion*, akan lebih banyak terjadi *packet drop* seperti pada Gambar 15. Hal tersebut didukung cara kerja dari TCP Newreno yang dipengaruhi oleh adanya *packet loss*. Sedangkan pada TCP Vegas semakin besar *buffer* antrian maka variasi *delay* juga semakin besar yang menyebabkan *congestion window* semakin kecil, sehingga akan lebih sedikit terjadi *packet drop* dikarenakan *congestion* yang dialami oleh jaringan lebih kecil.

Pada Gambar 16 dapat dilihat bahwa TCP Newreno ditandai dengan garis warna merah dan TCP Vegas ditandai dengan garis warna hijau. Pada gambar menunjukkan bahwa *congestion window* dari TCP Newreno lebih besar dari TCP Vegas. Hal ini dikarenakan TCP New Reno dalam

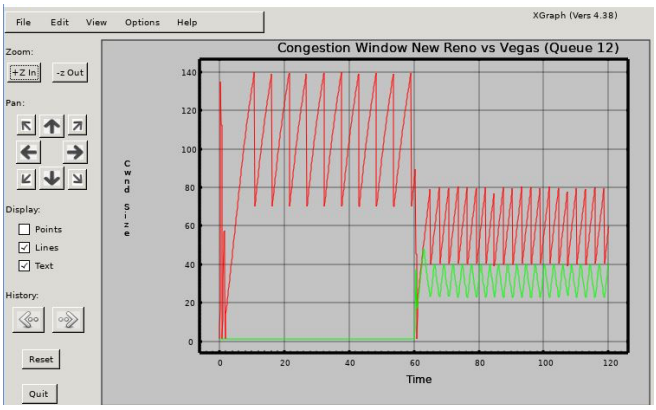
mengirimkan paket data tidak memperhatikan kondisi jaringan dengan mengirimkan paket data secara cepat, sehingga *congestion window* meningkat secara eksponential, namun ketika jaringan mengalami *congestion*, akan lebih banyak terjadi *packet drop* seperti pada Gambar 17. Hal tersebut didukung cara kerja dari TCP Newreno yang dipengaruhi oleh adanya packet loss. Sedangkan pada TCP Vegas semakin besar *buffer* antrian maka variasi *delay* juga semakin besar yang menyebabkan *congestion window* semakin kecil, sehingga akan lebih sedikit terjadi *packet drop* dikarenakan *congestion* yang dialami oleh jaringan lebih kecil.



Gambar 14 *Congestion Window* Newreno vs Vegas, buffer 10

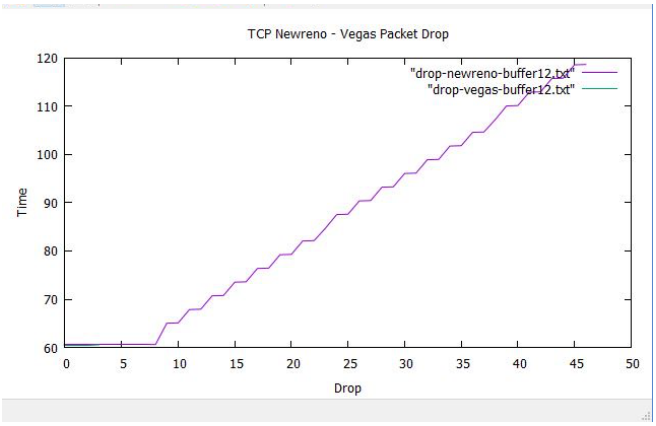


Gambar 15 *Packet Drop* Newreno vs Vegas, buffer 10

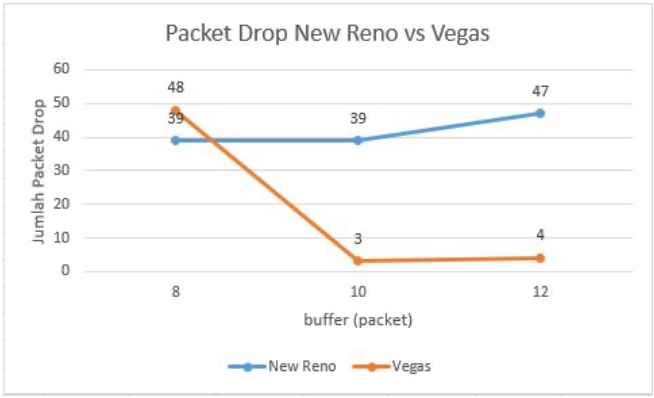


Gambar 16 *Congestion Window* Newreno vs Vegas, buffer 12

Berdasarkan *graph packet drop* pada Gambar 13, 15 dan 17, pada Gambar 18 merupakan jumlah *packet drop* pada masing-masing TCP untuk *buffer* 8, 10 dan 12 yang menujukkan bahwa *packet drop* TCP Newreno terlihat mengalami kenaikan ketika antrian *buffer* semakin besar dengan rata-rata lebih besar yaitu 41.67 packet, hal tersebut dikarenakan TCP New Reno dalam mengirimkan data tidak memperhatikan kondisi jaringan dengan mengirimkan data secara cepat, sehingga *congestion window* meningkat secara eksponential yang menyebabkan terjadinya *packet drop*. Sedangkan pada TCP Vegas terlihat *packet drop* mengalami penurunan drastis ketika antrian *buffer* semakin besar dengan rata-rata lebih kecil yaitu 18.33 packet, hal tersebut dikarenakan TCP Vegas dalam mengirimkan data akan melihat kondisi jaringan terlebih dahulu dengan menghitung varian RTT, ketika antrian *buffer* semakin besar dan paket data yang dapat ditampung semakin banyak, maka nilai RTT menjadi besar dan membuat TCP Vegas mengurangi jumlah data yang dikirim, sehingga besar data yang dapat di tampung pada *buffer* tidak terlalu banyak yang akan membuat *packet drop* menjadi kecil.



Gambar 17 *Packet Drop* Newreno vs Vegas, buffer 12

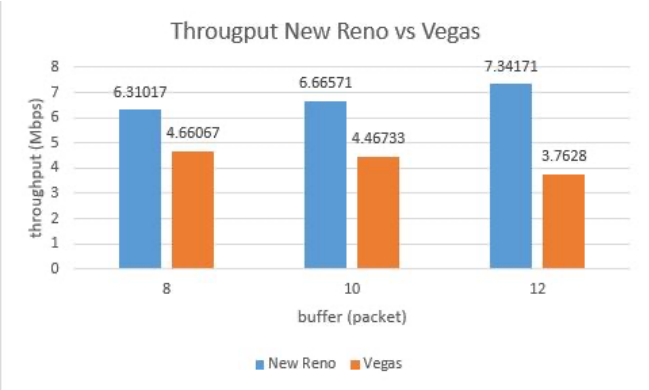


Gambar 18 *Packet Drop* Newreno vs Vegas

Gambar 19 menunjukkan bahwa nilai *throughput* TCP Newreno lebih tinggi daripada TCP Vegas baik ketika antrian *buffer* semakin besar dengan rata-rata lebih besar yaitu 6.77253 Mbps. Semakin tinggi *buffer* maka akan semakin tinggi nilai *throughput* TCP Newreno. TCP Newreno akan mengirim paket sebanyak mungkin sehingga semakin banyak data yang dilayani dan meski demikian *packet drop* terjadi, TCP New Reno akan melanjutkan ke fase *fast recovery* untuk menjaga *throughput* tetap tinggi.

Nilai *throughput* dari TCP Vegas lebih kecil dari TCP Newreno ketika *buffer* antrian besar dengan rata-rata lebih kecil yaitu 4.29693 Mbps. Hal ini dikarenakan TCP Vegas dalam mengirimkan data akan melihat kondisi jaringan berdasarkan varian RTT, jika RTT besar maka TCP Vegas akan mengurangi data yang dikirim, jika RTT kecil maka TCP Vegas akan mengirimkan data lebih

banyak. TCP Vegas mengalami penurunan ketika penambahan kapasitas *buffer*, hal tersebut disebabkan semakin besar kapasitas *buffer* maka data yang ditampung lebih banyak sehingga nilai RTT menjadi besar, saat nilai RTT besar TCP Vegas menganggap jaringan dalam keadaan *congestion* sehingga data yang dikirim semakin sedikit yang menyebabkan nilai *throughput* nya menjadi kecil.



Gambar 19 *Throughput* Newreno vs Vegas

4. KESIMPULAN

Dari hasil analisa dan pengujian yang dilakukan, dapat disimpulkan bahwa penerapan TCP Vegas dan TCP New Reno pada antrian *Drop Tail* dengan parameter uji yang berbeda telah berhasil dilakukan. Penerapan dilakukan dengan cara melakukan instalasi NS-2, setelah instalasi NS-2 berhasil dilanjutkan dengan konfigurasi *script* simulasi. Pengujian dilakukan dengan skema penambahan kapasitas *buffer* untuk antrian *Drop Tail*. Kemudian melakukan uji coba pada TCP Vegas dan TCP New Reno dengan melihat nilai dari parameter *throughput*, *packet drop* dan *congestion window*.

Hasil yang didapat dari pengujian yaitu untuk parameter *congestion window*, TCP Vegas memiliki kinerja lebih baik ketika *buffer* semakin besar. Pada antrian lebih kecil yaitu *buffer* 8, *congestion window* pada TCP Vegas lebih besar dibandingkan TCP New Reno namun tidak ada perbedaan signifikan, sedangkan pada *buffer* 10 dan 12, *congestion window* pada TCP Vegas lebih kecil dibandingkan TCP New Reno. Untuk parameter *packet drop*, TCP Vegas memiliki kinerja lebih baik ketika *buffer* semakin besar dengan rata-rata lebih kecil yaitu 18.33 *packet* dibandingkan dengan TCP New Reno memiliki nilai rata-rata yang lebih besar yaitu 41.67 *packet*. Untuk parameter *throughput*, TCP New Reno memiliki kinerja lebih baik dengan rata-rata lebih besar yaitu 6.77253 Mbps dibandingkan dengan TCP Vegas dengan rata-rata yang lebih kecil yaitu 4.29693 Mbps. Oleh karena itu, dari hasil pengujian dan analisis yang telah dilakukan pada TCP Vegas dan TCP Newreno dapat disimpulkan bahwa TCP Vegas memiliki kinerja yang lebih baik dari TCP New Reno ketika menggunakan antrian *Drop Tail* yaitu pada parameter kinerja *congestion window* dan *packet drop*.

Adapun saran yang dapat diberikan untuk pengembangan penelitian selanjutnya adalah membandingkan varian TCP yang lainnya yang memiliki algoritma *congestion window* yang sama. Serta melakukan pengujian dengan mekanisme antrian lainnya.

DAFTAR PUSTAKA

Brakmo, L. S., & Peterson, L. L. (1995). TCP Vegas: end to end congestion avoidance on a global Internet. *IEEE Journal on Selected Areas in Communications*, *13*(8), 1465–1480. https://doi.org/10.1109/49.464716

Chaudhary, P., & Kumar, S. (2017). A Review of Comparative Analysis of TCP Variants for Congestion Control in Network. *International Journal of Computer Applications*, *160*(8), 28–

34. https://doi.org/10.5120/ijca2017913087

Domański, A., Domańska, J., Pagano, M., & Czachórski, T. (2016). The Fluid Flow Approximation of the TCP Vegas and Reno Congestion Control Mechanism. In *Communications in Computer and Information Science* (Vol. 659, pp. 193–200). https://doi.org/10.1007/978-3-319-47217-1_21

Fajri, M. (2016). Simulasi Antrian Paket Data Jaringan dengan Mekanisme Drop Tail. *Jurnal Ilmiah FIFO*, *8*(2), 151. https://doi.org/10.22441/fifo.v8i2.1310

Hasanul Fahmi. (2018). Analisis Qos (Quality of Service) Pengukuran Delay, Jitter, Packet Lost Dan Throughput Untuk Mendapatkan Kualitas Kerja Radio Streaming Yang Baik. *Jurnal Teknologi Informasi Dan Komunikasi*, *7*(2), 98–105.

Kembuan, O. (2016). Analisa Packet Loss Transmission Control Protocol (TCP) RENO pada Jaringan Intranet Menggunakan NS2 (Network Simulator). *Engineering Education Journal (E2J-UNIMA)*, *4*(3), 24.

Kumar, A., Sharma, A. K., & Singh, A. (2012). Comparison and Analysis of Drop Tail and RED Queuing Methodology in PIM-DM Multicasting Network. *International Journal of Computer Science and Information Technologies*, *3*(2), 3816–3820.

Kurata, K., Hasegawa, G., & Murata, M. (2000). Fairness comparisons between TCP Reno and TCP Vegas for future deployment of TCP Vegas. *INET Conferences*, 1–20.

Orueta, G. D., Ruiz, E. S. C., Alonso, N. O., & Gil, M. C. (2016). Quality of Service. In *Ad Hoc Mobile Wireless Networks* (pp. 200–221). CRC Press. https://doi.org/10.1201/b13094-7

Pamungkas, G. W., Yahya, W., & Nurwarsito, H. (2018). Analisis Perbandingan Kinerja TCP Vegas Dan TCP New Reno Menggunakan Antrian Random Early Detection Dan Droptail. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer (J-PTIIK)*, *2*(10), 3239–3248.

PERFSONAR. (2020). *Lab 6 : Bandwidth-delay Product and TCP Buffer Size*.

Pratama, T., Irwansyah, M. A., & Yulianti. (2015). Perbandingan Metode PCQ, SFQ, RED Dan FIFO Pada Mikrotik Sebagai Upaya Optimalisasi Layanan Jaringan Pada Fakultas Teknik Universitas Tanjungpura. *Jurnal Sistem Dan Teknologi Informasi*, *3*(3), 298–303.

Susandi, H., & Pinem, M. (2014). Analisis Kualitas Layanan Data pada Jaringan Telekomunikasi Berbasis CDMA EVDO Rev . A. *Jurnal Singuda Ensikom*, *6*(2), 93–98.

Syaifuddin, M., Andika, B., & Ginting, R. I. (2016). Analisis Celah Keamanan Protocol TCP/IP. *Jurnal Ilmiah Sains Dan Teknologi (SAINTIKOM)*, *16*(2), 130–135.

Taruk, M., & Ashari, A. (2016). Analisis Throughput Varian TCP Pada Model Jaringan WiMAX. *IJCCS (Indonesian Journal of Computing and Cybernetics Systems)*, *10*(2), 115. https://doi.org/10.22146/ijccs.15529

Taruk, M., & Setyadi, H. J. (2016). Analisis Mekanisme Penanganan Kemacetan (Congestion Control) pada Algoritma Varian. *Konferensi Nasional Ilmu Komputer (KONIK)*, 1–4.

Torkey, H., Attiya, G., & Z. Morsi, I. (2012). Modified Fast Recovery Algorithm for Performance Enhancement of TCP-NewReno. *International Journal of Computer Applications*, *40*(12), 30–35. https://doi.org/10.5120/5018-7351