Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X
Vol. 2, No. 10, Oktober 2018, hlm. 3239-3248 http://j-ptiik.ub.ac.id

Analisis Perbandingan Kinerja TCP Vegas Dan TCP New Reno Menggunakan Antrian *Random Early Detection* Dan *Droptail*

Guntur Wahyu Pamungkas[1], Widhi Yahya[2], Heru Nurwarsito[3]

Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: [1]gunturpamungkaz@gmail.com, [2]widhi.yahya@ub.ac.id, [3]heru@ub.ac.id

Abstrak

Transmission Control Protokol (TCP) adalah protokol pada lapisan *transport* yang memiliki sifat *reliable* dalam mengirimkan data. Namun pengiriman yang *reliable* tersebut dapat terganggu karena *congestion* yang disebabkan peningkatan *traffic* data. Permasalahan tersebut dapat diatasi dengan menerapkan TCP dan manajemen antrian yang tepat. Pada penelitian ini akan dilakukan simulasi dengan membandingkan TCP Vegas dan TCP NewReno serta menambahkan antrian *Random Early Detection* dan *Droptail* dengan skema penambahan *minthresh*, *maxthresh* dan kapasitas *buffer*. Setelah penelitian ini selesai akan didapatkan TCP dan manajemen antrian yang memiliki kinerja terbaik dalam mengatasi *congestion*. Hasil pengujian menunjukkan kinerja TCP Vegas lebih baik dari TCP NewReno dengan *Random Early Detection*, terutama pada *packet delivery ratio, delay* dan *packet drop*. Hasil pengujian *packet delivery ratio*, TCP Vegas memiliki kinerja terbaik ketika menggunakan *Random Early Detection* dan *Droptail* dengan rata-rata 99.9968% berbanding 99.6768% untuk TCP NewReno. Hasil pengujian *throughput*, TCP NewReno memiliki kinerja terbaik ketika menggunakan *Droptail* dengan rata-rata 100749.4234kbps berbanding 94576.815kbps untuk TCP Vegas. Hasil pengujian *delay*, TCP Vegas memiliki kinerja terbaik ketika menggunakan *Random Early Detection* dengan rata-rata 4.31ms berbanding 5.98ms untuk TCP NewReno. Dan hasil pengujian *packet drop*, TCP Vegas memiliki kinerja terbaik ketika menggunakan *Random Early Detection* dengan rata-rata 0.0002% berbanding 0.319% untuk TCP NewReno.

Kata kunci: *TCP, Vegas, New Reno, Random Early Detection, Droptail, Network Simulator 2.35*

Abstract

Transmission Control Protocol (TCP) is a protocol on transport layer that have reliable properties in sending data. But reliable transmission can be distruped due congestion which are caused by increased traffic data. The problem can be solved by applying the correcrt TCP and queue management. In this research will be done simulation by comparing TCP Vegas and TCP New Reno by adding Random Early Detection and Droptail with scheme the addition of minthresh, max thresh and buffer capacity. After this research is done will get TCP and queue management that has the best performance in over coming congestion. The test results show, TCP Vegas is better than TCP NewReno with Random Early Detection, especially in packet delivery ratio, delay and packet drop. The results of packet delivery ratio test, TCP Vegas has the best performance when using Random Early Detection and Droptail with the average 99.9968% versus 99.6768% for TCP NewReno. The results throughput test, TCP NewReno has the best performance when using Droptail with the average 100749.4234kbps versus 94576.815kbps for TCP Vegas. The results delay test, TCP Vegas has the best performance when using Random Early Detection with the average of 4.31ms versus 5.98ms for TCP NewReno. And results packet drop test, TCP Vegas has the best performance when using Random Early Detection with the average 0.0002% versus 0.319% for TCP NewReno.

Keywords: *TCP, Vegas, New Reno, Random Early Detection, Droptail, Network Simulator 2.35*

1. PENDAHULUAN

Transmission Control Protocol (TCP) yaitu salah satu protokol pada *layer transport* dengan proses pengiriman data yang aliran datanya ketika dibaca TCP tujuan tidak rusak, tidak

memiliki duplikasi dan berurutan (*reliable*) (Kurose dan Ross, 2010). Untuk mewujudkan pengiriman data yang *reliable*, TCP banyak dikembangkan untuk menangani masalah yang sering terjadi dalam jaringan yaitu *congestion*.

Congestion dapat menyebabkan beberapa masalah yaitu meningkatnya *packet loss*, melambatnya transmisi dan yang paling parah dapat menyebabkan kelumpuhan pada jaringan. Untuk mengatasi hal tersebut dalam penelitian ini akan akan diterapkan TCP dan manajemen antrian yang tepat. TCP Vegas dan TCP New Reno dipilih karena kedua TCP memiliki mekanisme berbeda dalam mengatasi *congestion*, sedangkan manajemen antrian *Random Early Detection* dan *Droptail* dipilih karena kedua manajemen antrian tersebut memiliki mekanisme yang berbeda dalam melayani data pada antrian .

Untuk Varian TCP, mekanisme TCP Vegas dalam mengatasi *congestion* yaitu lebih kepada *packet delay* dari pada *packet loss*. Memiliki sifat proaktif karena dapat menghindari *congestion* sebelum terjadi penumpukan data dalam jaringan. TCP Vegas dalam mendeteksi *congestion* melihat varian RTT yang ada, jika RTT besar maka jaringan dianggap mengalami *congestion*, sehingga *congestion window* dikurangi yang menyebabkan pengiriman data berkurang, jika RTT kecil maka jaringan dianggap dalam keadaan normal, sehingga *congestion window* ditambah yang menyebabkan pengiriman data bertambah (Brakmo dan Peterson, 2006). Untuk TCP New Reno memiliki sifat reaktif karena dapat mengendalikan *congestion* ketika terjadi penumpukan data dalam jaringan. TCP New Reno akan mengirimkan data secara eksponensial dalam jaringan sampai terjadi *packet loss* karena *congestion*, ketika *packet loss* terdeteksi *congestion window* akan dikurangi sampai setengah (Torkey dkk, 2012).

Untuk manajemen antrian, mekanisme *Random Early Detection* dalam melayani data yang datang yaitu dengan menentukan *min thresh* dan *max thresh*, jika data kurang dari *min thresh*, data akan dilayani, jika data diantara *min thresh* dan *max thresh*, data akan ditandai dan di *drop* secara *random* dan jika data lebih dari *max thresh*, data akan langsung di *drop* (Sungur, 2015). Untuk *Droptail* menggunakan manajemen FIFO, dimana data yang pertama datang akan dilayani, jika kapasitas *buffer* penuh, maka data akan yang datang akan langsung di *drop* (Kumar dkk, 2012).

Pada penelitian sebelumnya membandingkan kinerja TCP Reno dan TCP Vegas dengan parameter *throughput, packet loss* dan *delay* menggunakan omnet++, didapatkan hasil TCP Vegas lebih baik (Sugiri, 2016). Selanjutnya membandingkan kinerja TCP Tahoe dan TCP New Reno dengan parameter *packet loss, throughput dan delay* menggunakan omnet++, didapatkan hasil TCP New Reno lebih baik (Manibuy, 2017).

Dalam penelitian ini akan dilakukan simulasi untuk membandingkan kinerja TCP Vegas dan TCP New Reno dengan menggunakan antrian *Random Early Detetection* dan *Droptail* pada NS-2. Simulasi menggunakan topologi *Abilene* yang diasumsikan sebagai jaringan kabel. Pengujian akan dilakukan dengan skema penambahan *min thresh* dan *max thresh* untuk antrian *Random Early Detection* dan skema penambahan kapasitas *buffer* untuk antrian *Droptail*. Setelah dilakukan penelitian ini, diharapkan dapat mengetahui varian TCP dan manajemen antrian mana yang memiliki kinerja terbaik ketika dalam jaringan terjadi *congestion*. Tujuan dari penelitian ini yaitu mengetahui TCP serta manajemen antrian yang memiliki kinerja terbaik dalam mengatasi *congestion* dengan melihat nilai dari parameter *packet drop, delay , throughput* dan *packet delivery ratio.*

2. DASAR TEORI

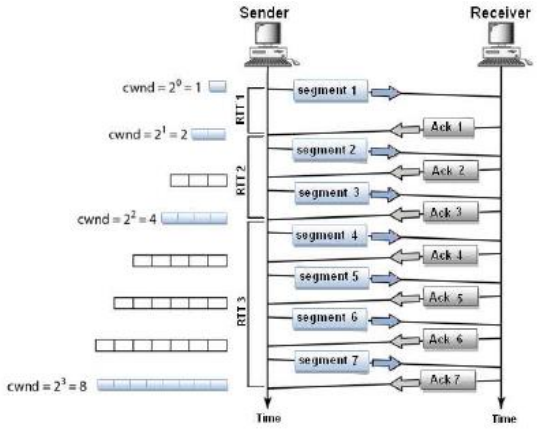
2.1. *Transmission Control Protokol* (TCP)

Transmission Control Protokol (TCP) yaitu protokol berbasis *connection-oriented* yang berada di *layer transport*. Dalam mengirimkan data TCP memiliki mekanisme yang bersifat *reliable*, hal itu membuat *traffic* data yang di baca TCP tujuan berurutan, tidak rusak dan tidak ada duplikasi (Kurose dan Ross, 2010). Sampai saat ini TCP banyak dikembangkan dalam mengatasi *congestion* untuk dapat menjamin bahwa pengiriman data *reliable*. Dalam algoritma TCP terdapat empat tahap yaitu *slow start, congestion avoidance, fast retransmit* dan *fast recovery*.

2.1.1. *Slow Start*

Tahap ini digunakan TCP sumber untuk

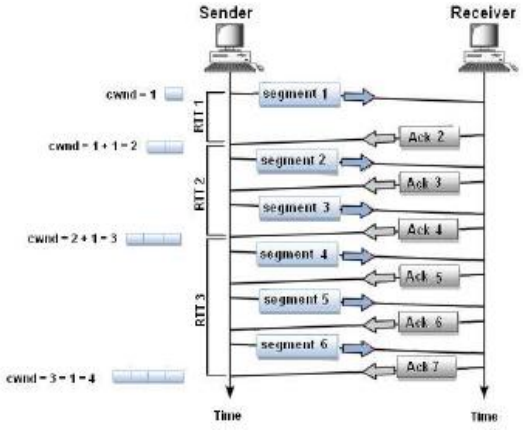
mengirimkan data ke TCP tujuan. Pengiriman data menggunakan *congestion window* dengan satu *segment size* dan dengan setiap ACK yang diterima, ukuran *congestion window* bertambah satu segmen. Alasan ini membuat *congestion window* meningkat secara eksponensial dan segmen ditambahkan ke jaringan untuk setiap RTT (Cavendish dkk, 2009). Fase *slow start* dapat dilihat pada Gambar 1, *congestion window* diatur ke satu segmen di awal dan meningkat satu segmen untuk masing-masing segmen ACK ketika berhasil diterima dan ketika terjadi *congestion* karena tidak diterimanya ACK dari TCP tujuan maka akan dilanjutkan ke fase *congestion avoidance* (Allcock dkk, 2005). .



Gambar 1. Fase *Slow Start*

2.1.2. *Congestion Avoidance*

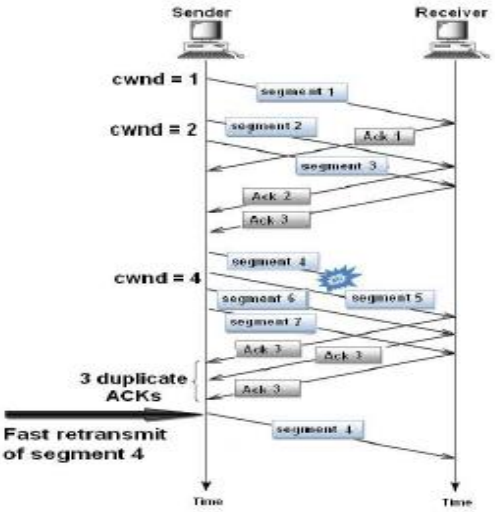
Setelah nilai *threshold* sama dengan *ssthresh*, laju pengiriman data akan diperlambat menjadi *linear*. Saat tingkat transmisi TCP sumber melebihi kapasitas *link*, *packet loss* terjadi. Setelah itu ukuran *congestion window* akan berkurang setengah dan akan kembali ke fase *slow start* lagi. *Congestion avoidance* diterapkan disaat ukuran *congestion window* lebih dari *ssthresh*. Tujuan *congestion avoidance* yaitu menghindari *congestion* dan menjaga *throughput* tetap tinggi. Jika TCP sumber mendeteksi segmen hilang, hal itu menandakan *congestion* pada jaringan. Jika *congestion* terjadi, *congestion window* akan turun yang menyebabkan tingkat pengiriman data juga berkurang dan TCP akan kembali ke fase *slow start*. Fase *congetion avoidance* dapat dilihat pada Gambar 2.



Gambar 2. Fase *Congestion Avoidance*

2.1.2. *Fast Retrasnmit* Dan *Fast Recovery*

Fast retransmit dan *fast recovery* memiliki mekanisme untuk mempercepat dalam mengambil koneksi. Dalam *fast retransmit*, TCP menduplikasi *acknowledgement* untuk mengirim ulang segmen yang hilang. Sedangkan dalam *fast recovery*, ketika terjadi *packet loss*, TCP menjaga tingkat transmisi pengiriman data tanpa harus kembali ke fase *slow start*. *Fast retransmit* dan *fast recovery* dilakukan untuk menyelesaikan masalah terkait *packet loss* tanpa menunggu RTO Masalah itu berdasar cara pengiriman. kembali *unacknowledgement segment* setelah diterimanya tiga duplikat *acknowledgement*, ketika itu terjadi, ukuran *congestion window* akan ke satu segmen dan kembali ke *fase slow* start. Tujuan dari menerima tiga duplikat *acknowledgement* yaitu untuk mengrim kembali segmen yang hilang akibat *congestion* (Ho dkk, 2005). Fase *fast retransmit* dapat dilihat pada Gambar 3.



Gambar 3. Fase *fast retransmit*

2.2. TCP Vegas

TCP Vegas dapat mendeteksi *packet loss* lebih awal, sehingga bisa melakukan *retransmit* sebelum mendapat 3 duplikat *acknowledgement* atau habis waktu RTO. Untuk mendeteksi *congestion*, TCP Vegas menggunakan varian RTT. Untuk menghitung varian RTT adalah sebagai berikut:

1. Menentukan *base* RTT (RTT terkecil selama koneksi).
2. Menghitung *expected rate* (transmisi *rate* yang diharapkan). Dengan cara:
   expectedRate = congestion window / baseRTT. *Congestion window* yang digunakan adalah saat belum mendapat *acknowledgement*.
3. Menghitung *actual rate* (transmisi *rate* yang sebenarnya). Dengan cara:
   actualRate = congestion window / RTT. (RTT yang sebenarnya).
4. Menghitung *diff* (perbedaan). Dengan cara:
   diff = expectedRate – actualRate
5. Menetukan batas *alfa* (data yang sedikit) dan beta (data yang banyak).
   Jika *diff<alfa*, maka *congestion window* +1.
   Jika *diff>beta*, maka *congestion window* -1.

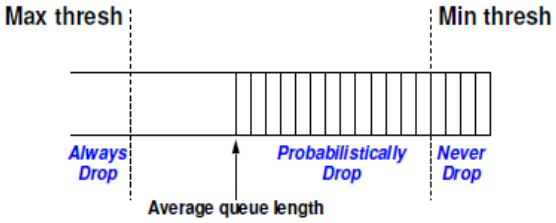
2.3. TCP New Reno

TCP New Reno dalam mengatasi congestion melalui fase-fase sebagi berikut:

1. *Slow start*, fase mengirimkan data dalam jaringan secara eskponensial sampai terdeteksi *packet loss* karena tidak diterimanya *acknowledgement* dan *congestion window* sama dengan *ssthresh.*
2. *Congestion avoidance*, fase ketika *congestion window* sama dengan *ssthresh*, kemudian tingkat pengiriman berubah menjadi *linear* untuk menghindari *congestion.*
3. *Fast retransmit*, fase dimana dilakukan proses pengiriman ulang paket yang hilang dengan diterimanya tiga duplikat *acknowledgement.*
4. Fast recovery, fase dimana menjaga *throughput* tetap tinggi setelah terjadi *packet loss* dengan mengganti fase *slow start* menjadi fase *congestion avoidance*.

2.4. *Random Early Detection*

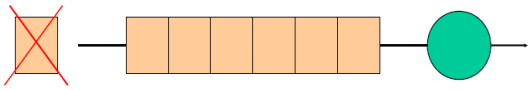
Random Early Detetction merupakan mekanisme antrian yang dapat melakukan *packet drop* sebelum *buffer* penuh dengan cara menentukan parameter *min thresh* dan *max thresh.* Jika data < *min thresh*, maka data dilayani. Jika data diantara *min thresh* dan *max thresh*, maka data di tandai dan di *drop* secara *random*. Jika data > *max thresh*, maka data akan langsung di *drop*. Ilustrasi *Random Early Detection* dapat dilihat pada Gambar 4.



Gambar 4. Ilustrasi Antrian *Random Early Detection*

2.5. *Droptail*

Droptail merupakan antrian yang menggunakan algoritma FIFO. Mekanismenya yaitu jika ada paket yang datang pertama 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. Ilustrasi *Droptail* dapat dilihat pada Gambar 5.



Gambar 5. Ilustrasi Antrian *Droptail*

3. METODOLOGI

Metodologi akan membahas tentang alur dari penelitian yang akan dikerjakan. Diagram alir penelitian dapat dilihat pada Gambar 6.

Gambar 6. Diagram Alir Penelitian

4. PERANCANGAN

4.1. Analisa Kebutuhan

Kebutuhan dalam penelitian ini dibagi menjadi 2 yaitu kebutuhan simulasi dan kebutuhan jaringan. Untuk kebutuhan simulasi adalah sebagi berikut:

1. Kebutuhan *Hardware*
   - Laptop Lenovo G40-70
   - *CPU* : Intel I3-4010U, *up to* 1.70GHz
   - *RAM* : 4 GB
   - *Hardisk* : 500 GB
2. Kebutuhan *Software*
   - Ubuntu 16.04 LTS 64 *bit*
   - *Network Simulator* 2.35
   - *Network Animator*

Untuk kebutuhan jaringan adalah sebagai berikut:

- Kebutuhan *Router*
- Kebutuhan *Node* Sumber
- Kebutuhan *Node* Tujuan
- Kebutuhan *Link*
- Kebutuhan *Buffer*

4.2. Perancangan Simulasi

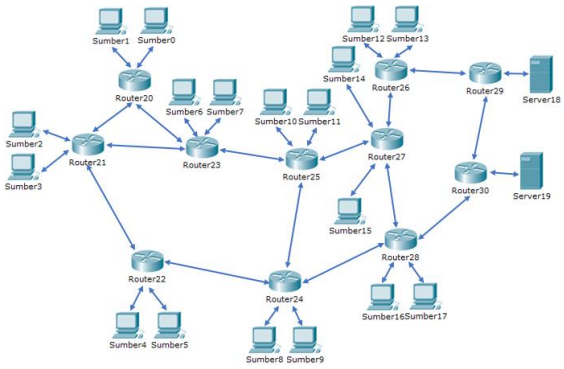
Perancangan simulasi ini akan menjelaskan parameter yang akan digunakan. Parameter pada penelitian ini digunakan sebagai nilai yang berfungsi untuk melakukan proses komputasi saat simulasi berjalan. Parameter simulasi dapat dilihat pada Tabel 1.

Tabel 1. Parameter Simulasi

Parameter Uji	Nilai			
Varian TCP	TCP Vegas	TCP Vegas	TCP New Reno	TCP New Reno
Antrian pada *router*	*Droptail*	*Random Early Detection*	*Droptail*	*Random Early Detection*
Buffer Size	20, 30, 40, 50 dan 60 paket	60 paket	20, 30, 40, 50 dan 60 paket	60 paket
Nilai min thresh & max thresh	-	1. *Min thresh* : 20, 25, 30, 35 dan 40 paket *Max thresh* : 50 paket 2. *Min Thresh :* 30, 35, 40, 45 dan 50 paket *Max Thresh* : 20 paket	-	1. *Min thresh* : 20, 25, 30, 35 dan 40 paket *Max thresh* : 50 paket 2. *Min Thresh :* 30, 35, 40, 45 dan 50 paket *Max Thresh* : 20 paket
Jumlah *node*	31 *node* (18 *node* sumber, 11 *router*, 2 *node* tujuan)			
Waktu simulasi	300 detik			
Traffic source	FTP			
Ukuran Paket	1024 *byte*			
Delay	2 ms			
Bandwidth	10 Mbps			

4.3. Perancangan Topologi

Dalam penelitian ini akan digunakan *Network Simulator* 2.35 untuk melakukan simulasi, topologi yang digunakan yaitu *Abilene*, *node-node* dalam topologi akan dihubungkan secara *wired* (kabel). Dalam topologi terdapat 18 *node* sumber, 11 *router*, 2 *node* tujuan, 10 Mbps *bandwidth* dan 2 ms *delay* dalam *link* dan penambahan *buffer*, *min thresh* dan *max thresh* pada antrian. Gambaran umum dapat dilihat pada Gambar 7.



Gambar 7. Rancangan Topologi *Abilene*

5. PENGUJIAN DAN ANALISIS HASIL

5.1. Pengujian

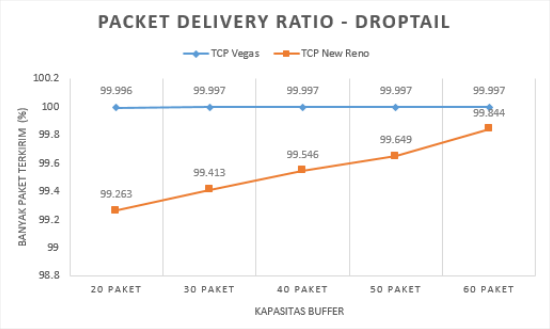
Dalam penelitian ini akan dilakukan pengujian dengan menguji kinerja dari TCP Vegas dan TCP New Reno menggunakan antrian *Random Early Detection* dan *Droptail* pada *Network Simulator* 2. Pertama, pengujian akan dilakukan pada TCP Vegas dan TCP NewReno secara bergantian menggunakan antrian *Droptail* dengan skema penambahan kapasitas *buffer*. Kedua, pengujian akan dilakukan pada TCP Vegas dan TCP NewReno secara bergantian menggunakan antrian *Random Early Detection* dengan skema penambahan *min thresh* dan skema penambahan *max thresh*. Parameter simulasiyang akan digunakan dapat dilihat pada tabel 1. Kedua pengujian tersebut akan menghasilkan *file* berisi data mentah dengan format *.tr*. Kemudian *file* tersebut akan dioleh dan dianalisis menggunakan *awk script*. Parameter hasil yang untuk mengetahui perbandingan kinerja dari varian TCP dan manajemen antrian digunakan pada penelitian ini adalah sebagai berikut: *packet delivery ratio*, *throughput*, *delay* dan *packet drop.*

5.2. Analisis Hasil

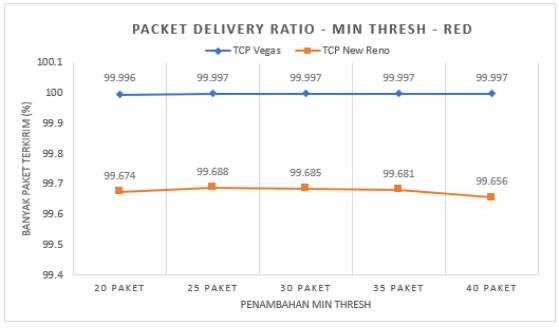
5.2.1 *Packet Delivery Ratio*

Packet delivery ratio adalah perbandingan jumlah paket yang berhasil sampai ke tujuan dengan jumlah paket yang dikirimkan oleh sumber.

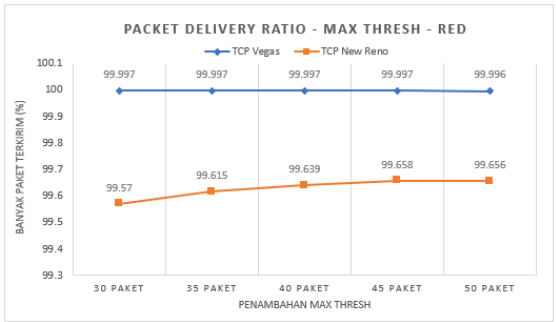
$$Packet\ Delivery\ Ratio = \frac{paket\_diterima}{paket\_dikirim} x\ 100\%$$



Gambar 8. *Packet Delivery Ratio* Dengan Kapasitas *Buffer*



Gambar 9. *Packet Delivery Ratio* Dengan *Min Thresh*



Gambar 10. *Packet Delivery Ratio* Dengan *Max Thresh*

Pembahasan dari hasil pengujian *packet delivery ratio* yang ditampilkan dalam grafik diatas adalah sebagai berikut:

1. TCP Vegas

Dari grafik pada Gambar 8, 9 dan 10 menunjukkan grafik yang datar dengan nilai rata-rata 99.9968 %.

Hal ini karena TCP Vegas dalam mengirimkan data melihat kondisi jaringan dengan menghitung varian RTT, jika RTT besar TCP Vegas menganggap jaringan mengalami *congestion*, sehingga akan mengurangi pengiriman paket data untuk menghindari *packet drop*, jika RTT kecil TCP Vegas mengganggap jaringan dalam kondisi baik, sehingga akan mengirimkan paket data lebih banyak. Alasan tingginya *packet delivery ratio* TCP Vegas ketika penambahan kapasitas *buffer*, *min* dan *max thresh* karena meskipun kapasitas *buffer*, *min* dan *max thresh* bertambah, TCP Vegas mampu mempertahankan pengiriman paket data secara konstan melalui varian RTT.

2. TCP NewReno

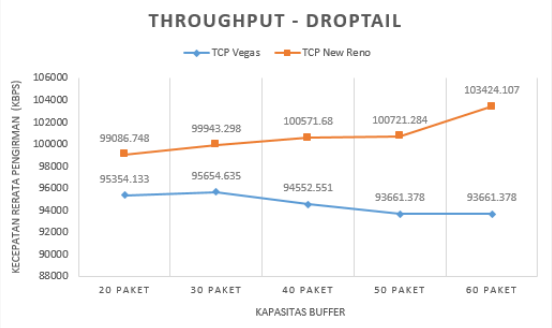
Dari grafik pada Gambar 8 dan 10, menunjukkan grafik yang naik dengan nilai rata-rata 99.543 % dan 99.631%.. Untuk grafik pada gambar 9, menunjukkan grafik yang datar dengan nilai rata-rata 99.676 %.

Hal ini karena TCP NewReno 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 terjadi *packet drop* yang menyebabkan *congestion window* berkurang setengah dan paket data yang dikirimkan pun ikut berkurang. Alasan meningkatnya *packet delivery ratio* TCP New Reno ketika penambahan kapasitas *buffer*, *min* dan *max thresh* karena semakin banyak kapasitas *buffer*, *min* dan *max thresh*, semakin banyak pula data yang dilayani.

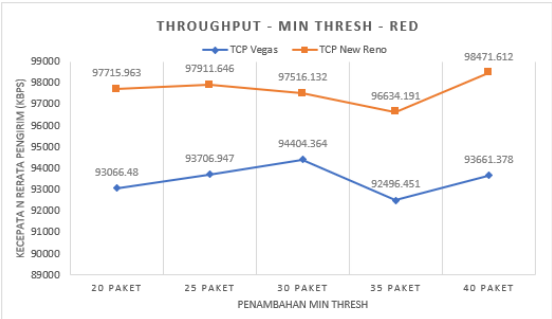
5.2.1 *Throughput*

Throughput adalah banyaknya paket yang diterima dalam kurun waktu yang sudah ditentukan.

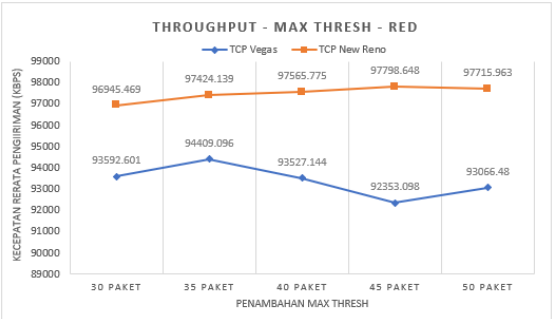
$$Throughput = \frac{paket\_diterima}{jumlah\_waktu\_pengiriman} x\ ukuran\ paket$$



Gambar 11. *Throughput* Dengan Kapasitas *Buffer*



Gambar 12. *Throughput* Dengan *Min Thresh*



Gambar 13. *Throughput* Dengan *Max Thresh*

Pembahasan dari hasil pengujian *throughput* yang ditampilkan dalam grafik diatas adalah sebagai berikut:

1. TCP Vegas

Dari grafik pada Gambar 11 dan 13, menunjukkan grafik yang fluktuatif dengan nilai rata-rata 94576.8 kbps dan 93397.6 kbps. Untuk grafik pada Gambar 10, menunjukkan grafik fluktuatif dengan nilai rata-rata 93467.1 kbps.

Hal ini karena TCP Vegas dalam mengirimkan data melihat kondisi jaringan dengan menghitung varian RTT, jika RTT besar maka TCP Vegas akan mengurangi data yang dikirim, jika RTT kecil maka TCP Vegas akan mengirimkan data lebih banyak. Alasan kenapa TCP Vegas mengalami penurunan ketika penambahan kapasitas *buffer*, *min* dan *max thresh* karena semakin besar kapasitas *buffer*, *min* dan *max thresh* maka data yang ditampung lebih banyak sehingga nilai RTT menjadi besar, ketika nilai RTT besar TCP Vegas menganggap jaringan dalam keadaan *congestion* sehingga data yang dikirim semakin sedikit.

2. TCP NewReno

Dari grafik pada Gambar 11, 12 dan 13, menunjukkan grafik yang naik dengan nilai rata-rata 100749.4 kbps, 97649.9 kbps dan 97489.9 kbps.

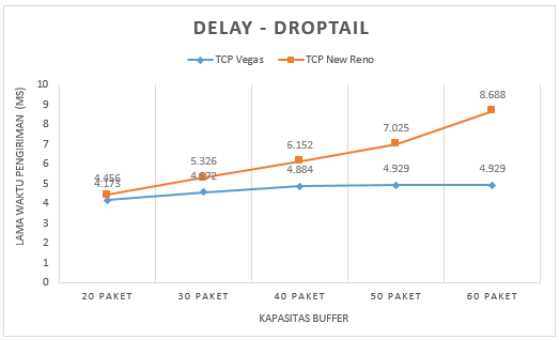
Hal ini karena TCP New Reno dalam mengirimkan data tidak memperhatikan kondisi jaringan dengan mengirimkan data secara cepat, sehingga *congestion window* meningkat secara eksponential, namun ketika jaringan mengalami *congestion*, akan terjadi *packet drop*. Alasan meningkatnya *throughput* TCP New Reno ketika penambahan kapasitas *buffer*, *min* dan *max thresh* karena semakin banyak kapasitas *buffer*, *min* dan *max thresh*, semakin banyak pula data yang dilayani dan meskipun *packet drop* terjadi TCP New Reno akan melanjutkan ke fase *fast recovery* untuk menjaga *throughput* tetap tinggi.

5.2.1 *Delay*

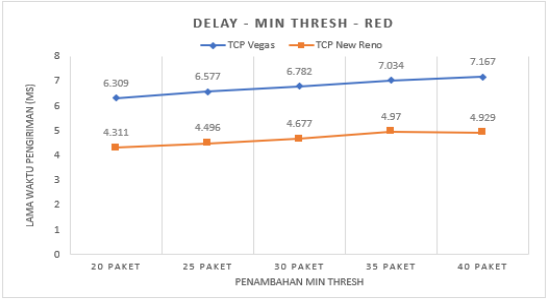
Delay adalah waktu yang dibutuhkan paket dari sumber ke tujuan saat dikirimkan.

$$Delay = \frac{waktu\_paket\_[i]\_diterima - waktu\_paket\_[i]\_dikirim}{jumlah\_paket\_dikirim}$$

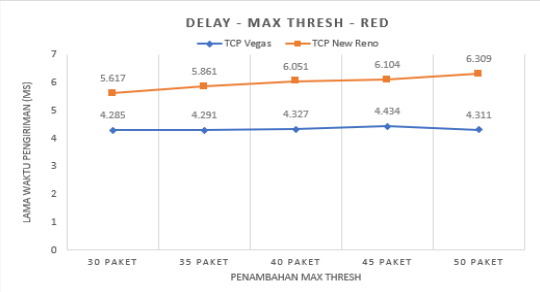
Keterangan : i = nomor paket yang berhasil diterima.



Gambar 14. *Delay* Dengan Kapasitas *Buffer*



Gambar 15. *Delay* Dengan *Min Thresh*



Gambar 16. *Delay* Dengan *Max Thresh*

Pembahasan dari hasil pengujian *delay* yang ditampilkan dalam grafik diatas adalah sebagai berikut:

1. TCP Vegas

Dari grafik pada Gambar 14, 15 dan 16, menunjukkan grafik yang datar dengan nilai rata-rata 4.69 ms, 4,67 ms dan 4.31 ms.

Hal ini karena TCP Vegas dalam mengirimkan data akan melihat kondisi jaringan terlebih dahulu dengan menghitung varian RTT. Alasan meningkatnya *delay* TCP Vegas yaitu ketika kapasitas *buffer*, *min* dan *max thresh* semakin besar dan paket data yang dapat ditampung semakin banyak, nilai RTT menjadi besar yang membuat TCP Vegas mengurangi jumlah data yang dikirim, sehingga besar data yang dapat di tampung pada kapasitas *buffer*, *min* dan *max thresh* tidak terlalu banyak dan hal tersebut membuat waktu tunggu paket untuk dilayani semakin kecil.

2. TCP NewReno

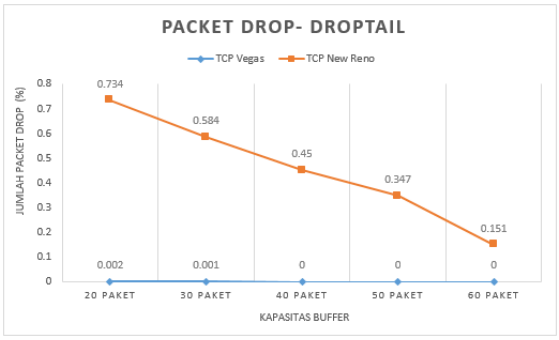
Dari grafik pada Gambar 14, 15 dan 16, menunjukkan grafik yang naik dengan nilai rata-rata 6.32 ms, 6.77 ms dan 5.98 ms.

Hal ini karena 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 terjadi *packet drop* yang menyebabkan *congestion window* berkurang setengah dan paket data yang dikirimkan pun ikut berkurang. Alasan meningkatnya *delay* TCP New Reno yaitu dengan kapasitas *buffer*, *min* dan *max thresh* yang kecil maka *delay* pun kecil namun dengan meningkatnya kapasitas *buffer*, *min* dan *max thresh* maka semakin banyak pula paket data yang bisa di tampung dan hal ini menyebabkan waktu untuk melayani paket data semakin besar.

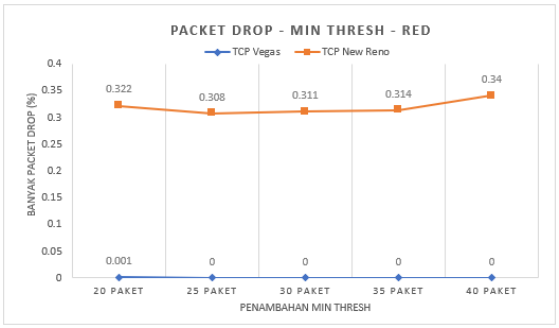
5.2.1 *Packet Drop*

Packet drop adalah paket yang hilang ketika proses pengiriman dari sumber ke tujuan terjadi.

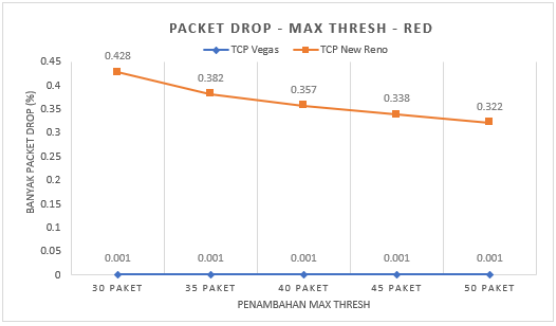
$$Packet\ Drop = \frac{paket\_dikirim\ -paket\_diterima}{paket\_dikirim} x\ 100\%$$



Gambar 17. *Packet Drop* Dengan Kapasitas *Buffer*



Gambar 18. *Packet Drop* Dengan *Min Thresh*



Gambar 19. *Packet Drop* Dengan *Max Thresh*

Pembahasan dari hasil pengujian *packet drop* yang ditampilkan dalam grafik diatas adalah sebagai berikut:

1. TCP Vegas

Dari grafik pada Gambar 17, 18 dan 19, menunjukkan grafik yang datar dengan nilai rata-rata 0.0006 %, 0.0002 % dan 0.001 %.

Hal ini karena TCP Vegas dalam mengirimkan data akan melihat kondisi jaringan terlebih dahulu dengan menghitung varian RTT.

Alasan menurunnya *packet drop* TCP Vegas yaitu ketika kapasitas *buffer*, *min* dan *max thresh* 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*, *min* dan *max thresh* tidak terlalu banyak dan hal tersebut membuat *packet drop* menjadi kecil.

2. TCP NewReno

Dari grafik pada Gambar 17 dan 19, menunjukkan grafik yang turun dengan nilai rata-rata 0.4532 %, 0.319 % dan 0.3754 %. Untuk grafik pada Gambar 18, menunjukkan grafik yang datar dengan nilai rata-rata.

Hal ini karena 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 terjadi *packet drop.* Alasan menurunnya *packet drop* TCP New Reno yaitu dengan kapasitas *buffer*, *min* dan *max thresh* yang kecil maka p*acket drop* menjadi banyak namun dengan meningkatnya kapasitas *buffer*, *min* dan *max thresh* maka semakin sedikit paket data yang di *drop*.

6. KESIMPULAN

Dari hasil yang telah di dapat dalam penelitian ini dapat disimpulkan sebagai berikut:

1. Penerapan TCP Vegas dan TCP New Reno serta antrian *Random Early Detection* dan *Droptail* dengan parameter uji yang berbeda berhasil dilakukan. Penerapan dilakukan dengan cara melakukan instalasi NS-2, setelah instalasi NS-2 berhasil dilanjutkan dengan konfigurasi *script* simulasi. Dalam konfigurasi *script* simulasi ini pembuatan lingkungan simulasi dan penerapan TCP serta manajemen antrian dimulai. Dimulai dari pembuatan topologi, menentukan TCP yang digunakan, menentukan antrian yang digunakan, menentukan *buffer size*, menentukan aliran data, besar paket yang dikirim, *traffic source* yang digunakan dan lama simulasi.
2. Hasil yang didapat dari pengujian yaitu untuk parameter *packet delivery ratio*, TCP Vegas memiliki kinerja lebih baik ketika menggunakan antrian *Random Early Detection* maupun *Droptail* dengan rata-rata yaitu 99.9968 % dibandingkan dengan dari TCP New Reno dengan rata-rata 99.6768 %. Untuk parameter *throughput*, TCP New Reno memiliki kinerja lebih baik ketika menggunakan antrian *Droptail* dengan rata-rata 100749.4234 kbps dibandingkan dengan TCP Vegas dengan rata-rata yaitu 94576.815 kbps. Untuk parameter *delay*, TCP Vegas memiliki kinerja lebih baik ketika menggunakan antrian *Random Early Detection* dengan rata-rata yaitu 4.31 ms dibandingkan dengan TCP New Reno dengan rata-rata yaitu 5.98 ms. Dan 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.*

7. DAFTAR PUSTAKA

Allcock, W., Hegde, S. & Kettimuthu, R. 2005. *Restricted Slow-Start for TCP*. IEEE International Conference in Cluster Computing. 1-2.

Brakno, L. S., and Peterson, L. L. 2006. *TCP Vegas: End to End Congestion Avoidance on a Global Internet.* IEEE Press Piscataway, NJ, USA.

Cavendish, D., Kumazoe, K., Tsuru, M., Oie, Y. & Gerla, M. 2009. *CapStart: An Adaptive TCP Slow Start for High Speed Networks.* International Conference on INTERNET. 15-20.

Kumar, A., Sharma, A. K., and Singh, A. 2012. *Comparison and Anlysis of Drop Tail and RED Queuing Methodology in PIM-DM Multicasting Network*. India: National Institute of Technology.

Kurose, J. F. & Rose, K. W. 2010. Computer Networking: A Top-Down Approach. Prentice Hall, New Jearsey.

Ho, C. Y., Chan, Y. C. & Chen, Y. C. 2005. *An Enhanced Slow-Start Mechanism for TCP Vegas.* Journal of Communications and Networks. 405-411 Vol. 401.

Manibuy, Kukuh. R. A. 2017. Analisis Perbandingan Unjuk Kerja TCP Tahoe Dan New Reno Pada Jaringan *Wireless* dan *Wired.* Yogyakarta : Universitas

Sanata Dharma.

Satria, R. R. 2013. Analisis Perbandingan Kinerja Protokol TCP New Reno dan Westwood+ Pada Jaringan WiMAX. Bandung : *Universitas Widyatama*.

Sugiri, Theo. M. B. 2016. Analisis Perbandingan Unjuk Kerja TCP Reno Dan Vegas Pada Jaringan *Wired*. Yogyakarta : Universitas Sanata Dharma.

Sungur, A. 2015. TCP-Random Early Detection (RED) Mechanism For Congestion Control. New York : *Rochester Institute of Technology*.

Torkey, H., Attiya, G. & Morsi, I. Z. 2010. Modified Fast Recovery Algoritm for Performance Enhancement of TCP-NewReno. *International Journal of Computer Traffic sources*.