Analisa *Packet Loss Transmission Control Protocol* (TCP) *RENO* pada Jaringan Intranet Menggunakan NS2 (*Network Simulator*)

Olivia Kembuan
Pendidikan Teknologi Informasi dan Komunikasi, FakultasTeknik , Universitas Negeri Manado
Kampus UNIMA Tondano
Email : oliviakembuan@unima.ac.id

***Abstract*— TCP is know to be the most useful transport layer protocol all around the globe. It provides a safe and connection oriented service on the network. Network congestion is one of the factors that reduces the TCP efficiency particularly, when variety of traffics are on the network. Certain input parameters, such as : number of users, network capacity, send packet size, etc, are used to control the congestion. As the network bandwidth increase, more attention is paid on the congestion control mechanism by focusing on the flow control management on the TCP. The main objective of this paper is to observe TCP Reno packet loss andcompare it with early TCP (TCP Tahoe). The result show that the TCP Reno has better lower drop rate than another version (TCP Tahoe) because using fast recovery algorithm.**

***Keywords* - TCP, congestion control, Random Early Detection**

***Intisari*— TCP dikenal sebagai protocol pada lapisan transport yang paling berguna. Protokol ini menjamin keamanan dan layanan yang bersifat koneksi dalam jaringan computer. Kongesti pada jaringan merupakan salah satu factor yang mengurangi efisisensi dari kinerja TCP, ketika terdapat bermacam-macam *traffic* pada jaringan. Beberapa input parameter seperti : jumlah user, kapasitas jaringan, ukuran paket yang dikirimkan, dan lain sebagainya, merupakan parameter yang digunakan untuk mengontrol kongesti yang terjadi. Ketika bandwidth jaringan meningkat, perhatian difokuskan pada mekanisme control kongesti dengan cara control manajemen pada TCP. Tujuan utama dari penelitian ini adalah untuk menganalisa paket loss pada TCP Reno dan perbandingannya dengan TCP sebelumnya yaitu TCP Tahoe. Hasil penelitian menunjukkan bahwa TCP Reno memiliki drop rate yang lebih kecil dibandingkan dengan TCP Tahoe karena menggunakan algoritma fas recovery.**

***Kata Kunci*— TCP, kontrol kongesti, Random Early Detection**

I. PENDAHULUAN

Dalam suatu proses pertukaran data dalam jaringan intranet, kadang terjadi kemacetan data atau yang biasa disebut juga dengan kongesti. Fenomena kongesti cukup rumit, serta merupakan subjek utama yang harus dihadapi dalam kontrol kongesti.

Menurut istilah umum, kongesti terjadi bila jumlah paket yang ditransmisikan sepanjang jaringan jaringan mulai mendekati kapasitas jaringan dalam menangani paket-paket. Bila terjadi kemacetan/kongesti kemungkinan data akan dibuang sangat besar. Karena itu dibutuhkan suatu teknik kontrol kongesti untuk tetap mempertahankan jumlah paket data dalam jaringan pada saat kinerja menurun secara drastis.

Transmission Control Protokol (TCP) merupakan protokol yang bermanfaat dalam menghindari dan menangani terjadinya suatu kongesti dalam jaringan. Teknik slow start yang terdapat pada TCP Tahoe menyebabkan kemacetan mudah terjadi pada jaringan. Hal ini disebabkan karena adanya peningkatan jumlah data yang dikirim secara drastis dalam waktu singkat pada tahap awal pengiriman data.Meskipun menggunakan TCP Tahoe sebagai protokol end user, data yang dikirim masih hilang dalam jumlah besar.TCP Reno dengan teknik fast recovery bisa mengurangi jumlah dataloss pada jaringan. Jika dikombinasikan dengan teknik Random Early Detection (RED) diharapkan dapat mengurangi banyaknya jumlah data yang dibuang pada gateway.

II. TINJAUAN PUSTAKA

Transmission Control Protocol (TCP) Tahoe

TCP pada awalnya (Early TCP) hanya memiliki mekanisme flow kontrol dengan teknik sliding window yang sederhana tanpa adanya mekanisme kontrol kongesti. Setelah kejadian-kejadian kongesti yang terjadi pada Oktober 1986 [1], tahun 1988 Van Jacobson mengembangkan TCP dengan menambahkan beberapa algoritma kontrol kongesti dan memperkenalkannya dengan nama "TCP Tahoe".

Pada implementasinya, TCP Tahoe menambahkan beberapa algoritma baru dan memperbaharui implementasi TCP sebelumnya.Algoritma baru tersebut meliputi Slow Start, Congestion Avoidance dan Fast Restarnsmit.[1]

2.1 Slow Start

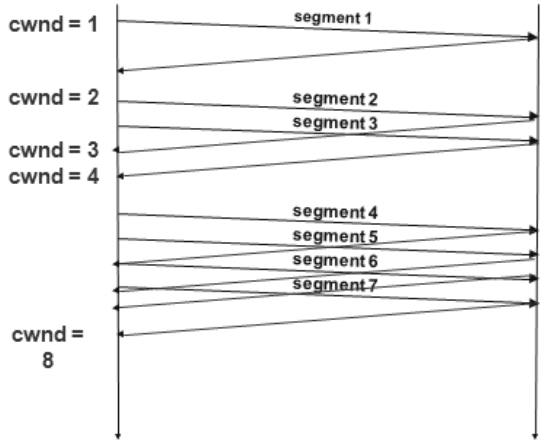
TCP merupakan end-to-end protocol yang dapat beroperasi pada berbagai jenis jaringan.TCP tidak mengetahui mengenai karakteristik suatu jaringan, karena itu protokol ini harus disesuaikan dengan keadaan suatu jaringan.TCP memiliki kemampuan untuk menghindari terjadinya kongesti pada jaringan.Pada kontrol kongesti, TCP memastikan bahwa data yang dialirkan tidak melebihi kapasitas jaringan.Slow start merupakan bagianpenting dalam suatu kontrol kongesti.

Pada TCP,receiver paket data yang dikirim akan menentukan ukuran window untuk setiap paket data. Istilah window menyatakan jumlah data (dalam byte) yang dapat dikirm oleh sender sebelum menunggu acknowledgment. Semakin besar pengiriman window yang digunakan dalam TCP, semakin banyak segmen sender TCP bisa melakukan pengiriman sebelum menerima ack. Hal ini menimbulkan masalah bila koneksi ditetapkan untuk pertama kalinya,

karena entitas TCP bebas membuang seluruh window yang ada pada internet.

Slow start mengizinkan TCP memeriksa kondisi jaringan dengan menaikkan secara perlahan data yang diinjeksikan ke dalam jaringan. Algoritma slow start menggunakan congestion window, untuk mengontrol aliran data dan menggunakan dua buah variable dalam implementasinya, yaitu cwnd dan sshtresh.Cwnd(menyatakan ukuran window) diinisialisasi ke satu segmen, biasanya 512 bytes.Sedangkan sshtresh merupakan batas dimana suatu slow start masih berlangsung. Jika cwnd meningkat melebihi sshtresh maka dianggap fase tersebut telah keluar dari prosesslow start.

Prinsip slow start sederhana, bahwa untuk setiap ACK yang diterimamenambahkan satu segmen ke cwnd. Proses slow start dapat dilihat pada Gambar1.



Gambar 1Slow Start

Pada saat awal koneksi window berukuran 1 kemudian receiver mengirimkan signal ACK yang menyatakan bahwa paket data dari pengirim telah diterima, maka pengirim meningkatkan ukuran window menjadi dua kali dari sebelumnya. Selanjutnya saat signal ACK berikutnya tiba di pengirim, maka pengirim meningkatkan ukuran window menjadi 4 (dua kali dari ukuran window sebelumnya) dan seterusnya.Dengan demikian teknik slow start meningkatkan jumlah pengiriman data dua kali lipat dari jumlah pengiriman data sebelumnya sampai.

Adapun aturan-aturan yang digunakan dalan teknik slow start yaitu, menambahkan congsetion window, cwnd pada setiap koneksi, ketika mulai mengirim data atau mengirim ulang data yang hilang, ukuran congestion window di-set menjadi satu paket. Setiap saat ACK datang, tambahkan congsetion window, cwnd dengan satu paket, ketika mengirim data, kirimkan window minimum yang terpadat pada penerima congsetion window.Proses peningkatan pengiriman data slow startakan terus berlanjut sampai mencapai sshtresh (slow start treshold) [2].

2.2 Congestion Avoidance

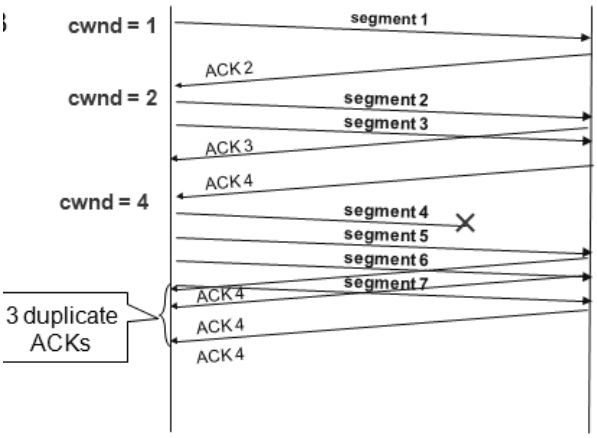
Pada tahap slow start, cwnd akan meningkat sampai mencapai sshtreshold. Pada fase ini TCP telah memasuki fase congestion avoidance, yaitu fase dimana TCP akan mengurangi laju pengiriman datanya dengan cara mengurangi jumlah cwnd untuk menghindari terjadinya kelebihan beban pada jaringan. Setelah mengurangi cwnd, TCP akan memasuki tahap flow start kembali.

Seperti halnya mekanisme slow start, dalam congestion avoidance juga terdapat aturan-aturan yaitu ketika paket hilang, ukuran congestion window (cwnd) di-set menjadi setengah ukuran window terakhir. Setiap ACK datang, tambahkan congestionwindow dengan 1/cwnd dan ketika mengirim data, kirimkan ukuran window minimal penerima dan congestionwindow[2].

2.3 Fast Retransmit

Ketika TCP menerima segmen yang tidak berurutan, maka TCP harus segera mengirim ACK untuk segmen terakhir yang telah diterima.TCPakan terus mengulang hal tersebut untuk setiap segmen yang datang sampai segmen yang hilang datang. Apabila pengirim menerima ACK lebih dari satu (duplikat ACK), maka berarti ada dua kemungkinan.Kemungkinan yang pertama ialah bahwa segmen terlambat datang (mengalami delay yang cukup panjang) sehingga segmen tiba di penerima dengan tidak berurutan.Dan kemungkinan yang kedua adalah bahwa segmen telah hilang. Dalam kasus pertama, segmen kemungkinan besar akan datang, sedangkan pada kasus kedua, datangnya lebih dari satu ACK yang sama merupakan suatu tanda bahwa segmen harus dikirim kembali (retransmited).

Jacobson merekomendasikan untuk kasus kedua, pengirim TCP harus menunggu sampai menerima tiga ACK yang sama untuk segmen yang sama. Pada saat itu, kemungkinan besar data memang telah hilang dan harus dilakukan retransmit tanpa harus menunggu batas waktu retransmit habis. Mekanisme seperti ini disebut dengan Fast retransmitted. Gambar 2 merupakan contoh suatu proses retransmit dengan mengirim ulang tiga duplikat segmen data. Suatu duplikasi ACK berarti bahwaout-of sequence segmenttelah diterima.



Gambar 2 Fast Retransmit

Transmission Control Protocol (TCP) Reno

TCP Reno adalah pengembangan dari TCP yang sudah ada (TCP Tahoe).Sejak diperkenalkan pada tahun 1988, TCP

Tahoe nampak berjalan cukup baik. Namun setelah diteliti lebih lanjut teryata TCP Tahoe dengan teknik Slow Start-nya menyebabkan banyaknya jumlah data yang terbuang, karena kongesti mudah terjadi pada tahap awal pengiriman data[3].

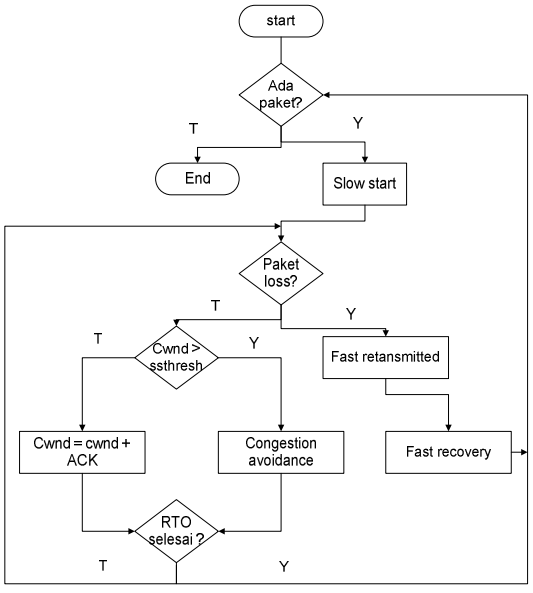
3.1 Fast Recovery

Pada tahun 1990 Jacobson melakukan sebuah modifikasi terhadap TCP Tahoe yang diberi nama "TCP Reno". Modifikasi yang dilakukan dalam TCP Reno ialah dengan menambahkan teknikFast Recovery untuk mempertahankan jumlah throughput yang dikirimkan [4].

Pada saat TCP mengirim ulang segment dengan mekanisme Fast Retarnsmit, maka dianggap ada segment yang hilang meskipun waktu (timer) bagi segmen tersebut belum habis. Oleh sebab itu, TCP harus mengambil tindakan untuk mencegah terjadinya kongesti dengan mekanisme slow start. Hal ini berarti TCPmengatur nilai ssthresh menjadi cwnd/2,cwnd =1 serta memulai proses slow start dengan nilai cwnd = ssthresh. Jacobson beralasan hal tersebut tidak perlu dilakukan, sehingga ia memperkenalkan teknik yang diebut Fast Recovery.Teknik Fast Recovery dapat dinyatakan sebagai berikut.Ketika TCP menerima ACK yang sama sebanyak 3 kali, yang akan dilakukan ialah,

1. set nilai ssthresh = cwnd/2
2. kirim ulang segment yang hilang
3. set cwnd = ssthresh + 3

Setiap saat TCP menerima ACK lain yang sama (Untuk segment yang sama) tambahkan cwnd dengan 1 dan kirimkan segment berikutnya dan ketika ACK yang selanjutnya (segment baru) datang, set cwnd = ssthresh.Gambar 3 menunjukkan mekanisme kontrol kongesti secara keseluruhan pada TCP Reno.



Gambar 3 Mekanisme Kontrol Kongesti TCP Reno

Pada tahap pengiriman data, TCPakan secara perlahan meningkatkan jumlah data yang diterima dengan teknik slow start sesuai dengan ACK yang diterima. Selama pengiriman data TCP akan terus memantau jumlah window yang dikirimkan. Jika jumlah window telah melebihi batas (ssthresh), TCP akan memasuki tahap congestion avoidance. Ketika terjadi pembuangan data TCPakan menggunakan teknik fast retransmit dan fast recovery untuk merespon ACK segmen data yang hilang yang dikirimkan penerima. tujuan pelaksanaan fast recovery adalah untuk mengisi kekosongan bandwidth akibat ditinggalkan paket yang hilang.

Random Early Detection (RED)

Random Ealry Detection atau bisa disebut Random Early Drop biasanya dipergunakan untuk gateway dengan tingkat trafik yang sangat tinggi.RED mengendalikan trafik jaringan sehingga terhindar dari kongesti pada saat trafik tinggi perdasarkan pemantauan perubahan nilai antrian minimum dan maksimum. Jika isi antrian di bawah nilai minimum maka mode drop tidak berlaku, saat antrian mulai terisi hingga melebihi nilai maksimum maka RED akan membuang paket data secara acak sehingga kongesti dapat dihindari[5].

RED bertujuan untuk mencapai hasil yaitu, RED merupakan algoritma untuk menghindari terjadinya kongesti dengan mendeteksi kongesti dan menjaga panjang antrian rata-rata dalam suatu daerah delay rendah dan throughput yang tinggi.Selama kongesti, ketika gateway membuang paket, paket pada koneksi-koneksi yang lain juga dibuang, terutama bila kongesti terjadi dalam waktu yang lama. Pada TCP/IP, pengirim paket TCP bereaksi terhadap paket yang terbuang dengan melakukan fase slow start dan jika banyak pengirim yang diperintahkan untuk melakukan hal yang sama pada saat yang bersamaan maka beban pada jaringan akan menurun secara drastis. dengan mendeteksi kongesti dan memberitahu hanya bagian tertentu yang dipilih secara acak dari pengguna, RED menghindari kongesti dan masalah sinkronisasi global [6].

RED mencegah perlakuan buruk terhadap koneksi yang bersifat burst (fairness). Hal ini penting karena bursty source akan menyebabkan kongesti, tetapi tidak berarti sumber memakai bandwidth yang sangat besar.

Untuk mencapai tujuan tersebut, RED menggunakan dua ambang batas Th^{min} dan Th^{max} serta menggunakan weighted moving average (Wq) sebagai formula untuk menghitung panjang antrian rata-rata. Kedua ambang batas dipergunakan untuk membentuk tiga zona.Jika panjang antrian rata-rata berada di bawah nilai ambang batasa minimal (Th^{min}), algoritma berada pada daerahnormal operation dan semua paket dapat diterima. Sebaliknya, jika berada di atas nilai ambang batas maksimal (Th^{max}), RED berada dalam daerah kontrol kongesti dan semua paket yang datang akan dibuang. Jika panjang antrian rata-rata berada diantara kedua ambang batas, RED berada dalam daerah penghindaran kongesti dan paket dibuang dengan kemungkinan probabilitas Pa. Secara umum RED melakukan langkah-langkah berikut setiap paket datang [7] :

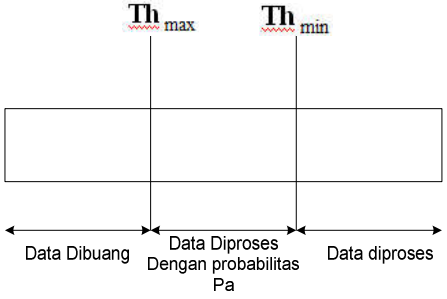
Hitung panjang antrian rata-rata, Q^{avg}

if $Q^{avg} < Th^{min}$
antrikan paket

else if $Th^{min} \leq Q^{avg} < Th^{max}$
hitung probabilitas Pa
dengan probabilitas Pa
buang paket
else with 1-Pa
antrikan paket

else if $Q^{avg} \geq Th^{max}$
buang paket

Pada Gambar 4, digambarkan pembagian daerah kongesti pada teknik Random Early Detection (RED).



Gambar 4 Pembagian daerah kongesti pada RED

Rumus untuk menghitung probabilitas jumlah data yang dibuang bila panjang antrian rata-rata (Q_{avg}) berada di antara nilai Th^{min} dan Th^{max} adalah :

$$P_b = P_{\max} \frac{(Q_{avg} - Th_{\min})}{Th_{\max} - Th_{\min}} \quad (1)$$

$$P_a = \frac{P_b}{1 - counter.P_b} \quad (2)$$

Dimana :

Q_{avg} = panjang antrian rata-rata
Wq = queue weight
P_b = probabilitas sementara pembuangan paket
P_a = probabilitas akhir pembuangan paket
$P_{\max}$ = nilai maksimum untuk P_b
q = panjang antrian
Th^{max} = ambang batas maksimal
Th^{min} = ambang batas minimal
counter= bilangan yang bernilai minimal 0 dan bertambah satu setiap kali jumlah data di buffer bertambah

Dari rumus 1 dan2, kita dapat melakukan observasi yaitu, Wq menunjukkan seberapa cepat algoritma akan merespon perubahan panjang antrian. Jka nilai Wq terlalu besar maka tidak akan ada proses seleksi terhadap paket yang dibuang untuk mengurangi kongesti, sebaliknya jika Wq terlalu kecil maka kongesti akan lambat untuk diatasi. Pemilihan Wq yang tepat tergantung pada Th^{min} dan jumlah burstiness (jumlah antrian paket data) yang diinginkan[2]. Bila burstiness yang diinginkan dalam jumlah paket data bernilai L, maka Q_{avg} dapat dihitung dengan persamaan :

$$Q_{avg} = L + 1 + \frac{(1 - W_q)^{L+1} - 1}{W_q} \quad (3)$$

Ambang batas harus ditentukan secara benar.Nilai optimal untuk Th^{max} dan Th^{min} tergantung pada besar antrian yang diinginkan.Dan besar antrian yang diinginkan tergantung jenis trafik yang dilalui network.Sebagai contoh, jika trafik secara keseluruhan bursty, Th^{min} harus cukup besar untuk menjaga high link utilization.Secara keseluruhan, problem terbesar RED adalah karakteristik trafik jaringan harus diketahui agar dapat diketahui panjangn antrian rata-rata optimum. Selain itu, nilai optimum untuk Th^{max} harus sama dengan delay rata-rata maksimum pada gateway. Karena tiap aplikasi memiliki kebutuhan delay yang berbeda, nilai ini tergantung pada jenis aplikasi yang berjalan pada jaringan.

Nilai probabilitas akhir (Pa) meningkat bersamaan dengan besar antrian rata-rata dan bertambahnya counter. Untuk menghindari masalah sinkronisasi global, RED menandai atau membuang paket secara acak.

III. METODOLOGI PENELITIAN

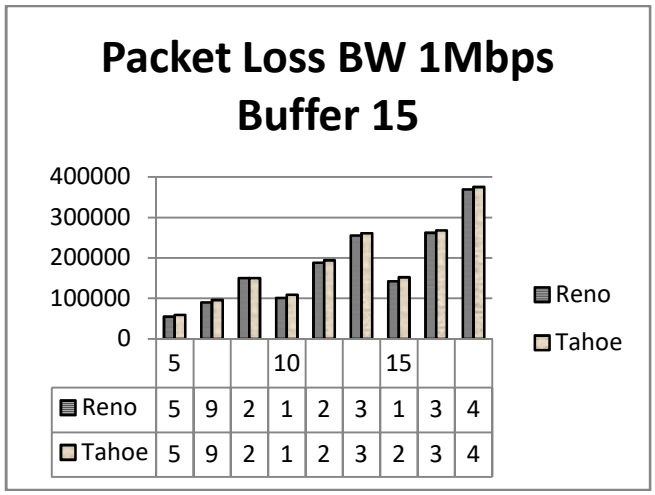
Metodologi penelitian yang digunakan adalah penelitian simulasi. Alat simulasi yang digunakan adalah *Ns2 network simulator*. Simulasi yang dilakukan memiliki empat skenario. Skenario pertama mensimulasikan keadaan jaringan dengan besar bandwidth 1Mbps pada link antara node pengirim dan router dengan ukuran buffer maksimal 15 packet pada router. Skenario kedua mensimulasikan keadaan jaringan dengan bandwidth 5Mbps pada link antara node pengirim dan router dengan ukuran buffer maksimal ditingkatkan menjadi 30 packet.

Skenario ketiga mensimulasikan keadaan jaringan dimana banwith pada link antara node pengirim dan router ditingkatkan menjadi 5Mbps, sedangkan buffer pada router maksimal 15 packet.Dan skenario keempat mensimulasikan keadaan jaringan dimana bandwidth pada link antara node pengirim dan router 5Mbps dengan ukuran maksimal buffer ditingkatkan menjadi 30 packet. Besar bandwidth pada link

antara node penerima dan router tetap 256kbps dengan delay 1ms.Perbandingan kinerja antara penggunaan TCP Reno dengan teknik RED dan TCP Tahoe menunjukkan performa penggunaan TCP Reno dalam suatu jaringan baik dalam segi throughput yang berhasil dikirimkan maupun jumlah pembuangan data pada router.

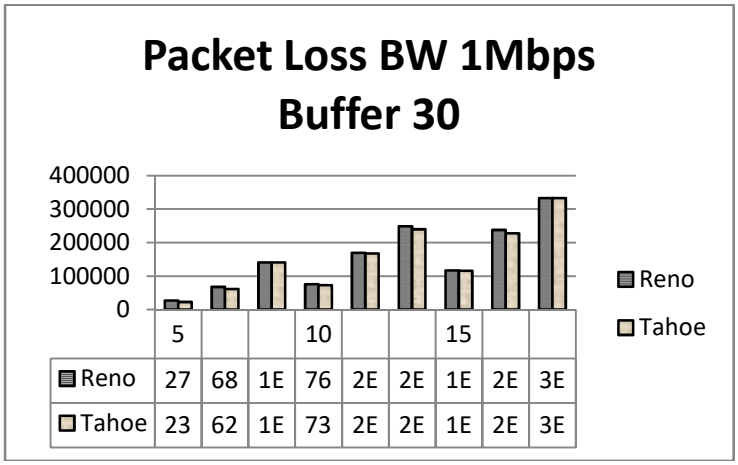
IV. HASIL DAN PEMBAHASAN

Pada skenario pertama saat bandwtih yang digunakan antara pengirim dan router 1Mbps dan buffer pada router maksimal 15 packet, penggunaan TCP Reno dengan teknik RED menunjukkan rata-rata pembuangan data yang lebih sedikit dibandingkan penggunaan TCP Tahoe. Dengan rata-rata selisih jumlah packet loss5250 byte. Perbandingan hasil simulasi antara TCP Reno dan TCP Tahoe dengan bandwidth 1Mbps dan buffer maksimal 15 packet dapat dilihat pada Gambar 5.



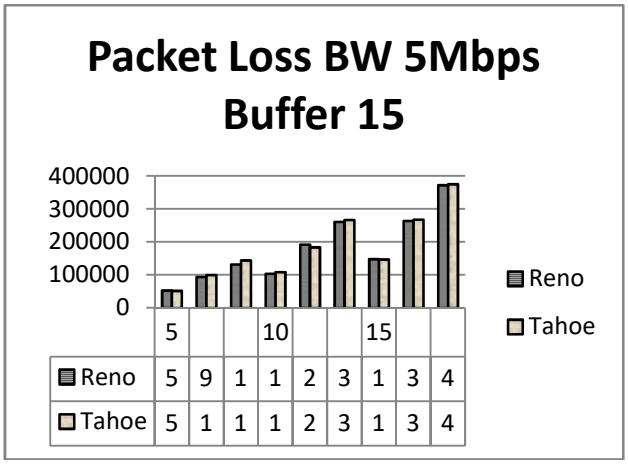
Gambar 5 Grafik perbandingan Packet loss, Bandwidth 1Mbps buffer 15 packet

Pada skenario kedua ukuran buffer maksimal pada router dicoba untuk dinaikkan menjadi 30 packet sedangkan ukuran bandwidth antara pengirim dan router tetap 1Mbps. Dari Gambar 6 dapat dilihat bahwa TCP Tahoe memiliki rata-rata pembuangan data yang lebih sedikit dibandingkan TCP Reno. Hal ini terjadi karena meskipun terjadi peningkatan pangiriman jumlah packet data dengan teknik slow start pada TCP Tahoe, tapi kapasitas buffer masih memungkinkan data untuk diantrikan sehingga terjadi penurunan drastis jumlah packet loss menggunakan TCP Tahoe. Jumlah packet loss pada TCP Reno juga mengalami penurunan diakibatkan peningkatan ukuran buffer, tapi tidak sebesar yang terjadi pada TCP Tahoe.



Gambar 6 Grafik perbandingan Packet loss, Bandwidth 1Mbps buffer 30 packet

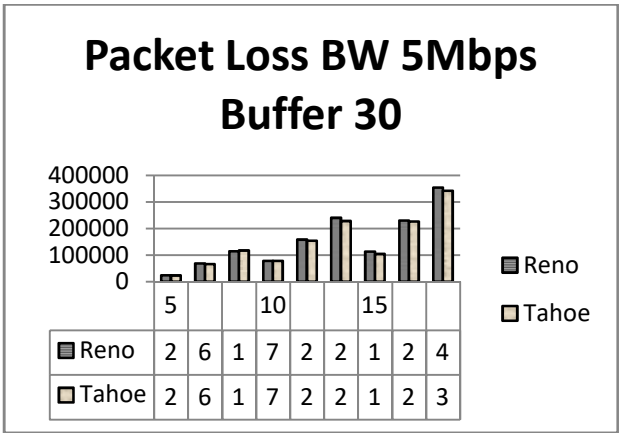
Pada skenario ketiga bandwidth jaringan dinaikkan menjadi 5Mbps pada masing-masing link antara node pngirim dan router.Sedangkan buffer minimal pada router tetap 5 packet dan max_thresh 3 kali ukuran minimal.Grafik perbandingan pengukuran packet loss menggunakan TCP Reno dengan teknik RED dan TCP Tahoe dengan teknik yang sama dapat dilihat pada Gambar 7.



Gambar 7 Grafik perbandingan Packet loss, Bandwidth 5Mbps buffer 15 packet

Dari Gambar 7 dapat dilihat bahwa pada trafik jaringan yang padat dengan ukuran buffer yang kecil yaitu 15 packet maksimal, TCP Reno memiliki packet loss yang relative kecil dibandingkan dengan TCP Tahoe. Ini disebabkan oleh fase fast recovery yang dimiliki TCP Reno mencegah peningkatan pengiriman data yang drastic ketika terjadi pembuangan data pada router.

Pada skenario keempat, ukuran bandwidth tetap 5 Mbps dengan ukuran buffer pada router ditingkatkan menjadi maksimal 30 packet. Grafik perbandingan packet loss pada TCP Reno menggunakan RED dengan TCP Tahoe dengan teknik yang sama dapat dilihat pada Gambar 8.



Gambar 8 Grafik perbandingan Packet loss, Bandwidth 5Mbps buffer 30 packet

Pada Gambar 8 dapat dilihat bahwa terjadi penurunan packet loss pada masing-masing TCP karena meningkatnya jumlah buffer pada router dan TCP Tahoe memiliki rata-rata jumlah packet loss yang lebih kecil dibandingkan TCP Reno dengan skenario yang sama. Hal ini hampir sama dengan yang terjadi pada skenario dua dimana buffer pada router yang ditingkatkan masih bisa menampung dengan baik peningkatan jumlah pengiriman data akibat slow start pada TCP Tahoe. Sehingga tidak terjadi pembuangan data yang cukup besar dibandingkan penggunaan TCP Reno.TCP Reno sendiri mengalami penurunan pembuangan data tapi tidak sebesar yang terjadi pada penggunaan TCP Tahoe.

Dari keempat skenario tersebut dapat disimpulkan bahwa semakin besar buffer, semakin kecil jumlah data yang akan dibuang pada router. Dan semakin besar ukuran data yang dikirimkan, semakin besar jumlah packet loss yang terjadi.TCP Reno memiliki karakteristik pembuangan data yang cenderung stabil jika dibandingkan dengan TCP Tahoe, yang pada jaringan yang padat dengan bufferyang kecil akan mengalami kenaikan packet loss secara drastic. Ini disebabkan teknik fast recovery pada TCP Reno yang mencegah kembalinya fase slow start saat jaringan pulih dari kongesti. Berbeda dengan TCP Tahoe yang cenderung akan segera kembali pada fase slow start sehingga memicu terjadinya kongesti pada jaringan dan menyebabkan jumlah pembuangan data yang dihasilkan akan lebih besar..

V. KESIMPULAN

Setelah melakukan simulasi dan analisa pada beberapa kondisi, maka didapat kesimpulan sebagai berikut, perbedaan packet loss antara TCP Reno dan TCP Tahoe tidak jauh berbeda karena memiliki persamaan pada beberapa fase. Seperti slow start pada tahap awal koneksi, dan mengalami congestion avoidance saat terjadi kongesti.TCP Reno cenderung lebih stabil jika dibandingkan TCP Reno yang kadang mengalami perubahan drastis. Karena fase fast recovery yang dimiliki TCP Renoakan menahan terjadinya fase slow start setelah terjadinya kongesti yang mencegah peningkatan transmisi data secara cepat.Selain faktor TCP yang digunakan, besarnya throughput juga bergantung pada ukuran data yang ditransmisikan dan besar bandwidth yang dialokasikan pada link.Jumlah data yang dibuang bergantung pada kapasitas buffer pada router.Semakin besar ukuran buffer, semakin kecil kemungkinan data dibuang.

REFERENSI

[1] V. Jacobson, 1988, Congestion avoidance and control, ACM Computer Communication Review, 18(4):314-329

[2] V. Jacobson, 1990, Modified TCP congestion avoidance algorithm, ftp://ftp.isi.edu/end2end/end2end-interest-1990.mail. Diakses tanggal 12 Oktober 2009.

[3] Brakmo, Lawrence S & Peterson, Larry L., 1995, TCP Vegas : End to end Congestion Avoidance on a Global Internet, IEEE Journal on Selected Areas in Communications, 13(8):1465-1480, Diakses tanggal 12 Oktober 2009.

[4] Stallings, William, 1998, High Speed Networks:TCP/IP and ATM Design Principles, Prentice Hall.

[5] Santosa, Budi, 2002, Manajemen Bandwidth Internet dan Intranet, Diakses tanggal 12 Oktober 2009.

[6] Labrador, Miguel A. & Banerjee, Sujata, Packet Dropping Policies for ATM and IP Networks, IEEE Communications Surveys, The Electronic Magazine of Original Peer-Reviewed Survey Articles, University of Pittsburgh, 1999

[7] Floyd, Sally & Jacobson Van, 1993, Random Early Detection Gateways for Congestion Avoidance, IEE/ACM Transaction on Networking, 1(4):397-413