ISSN 1979-2867 (print)
Electrical Engineering Journal
Vol. 1 (2011) No. 2, pp. 158-172

Analisis Performansi TCP-Friendly Multicast Congestion Control (TFMCC) Pada Jaringan Broadband Wireline

Anita Supartono, FX Arie Wibowo, dan Aditya Pratomo Sarwoko

Jurusan Teknik Elektro, Universitas Kristen Maranatha, Bandung

Jl. Suria Sumantri 65, Bandung 40164, Indonesia

anita_stono@yahoo.com, fxariw@yahoo.com,
adityapratomosarwoko@yahoo.com

Abstrak: Trend baru dalam dunia komunikasi, khususnya penyebaran aplikasi–aplikasi *streaming multicast* dan *real-time audio/video*, akan menyebabkan meningkatnya trafik non-TCP pada internet. Pada aplikasi-aplikasi tersebut biasanya jarang dilakukan *congestion control* seperti halnya aplikasi-aplikasi berbasis TCP, sehingga *buffer* lokal akan penuh dengan sangat cepat. Untuk menghindari terjadinya *congestion* pada trafik non-TCP yang dikirim secara *streaming multicast*, muncul algoritma TFMCC. Untuk mengetahui performansi jaringan dilakukan simulasi TFMCC versi Jorg Widman pada jaringan yang sama. Hasil simulasi menunjukkan, grafik *throughput* TFMCC lebih stabil daripada TCP. Peningkatan nilai variabel *bandwidth* pada *bottleneck link*, akan meningkatkan *throughput* rata-rata TCP dan TFMCC serta tingkat TCP- *friendliness.* Sementara itu, presentase *packet loss* akan menurun. Jika variabel *packet size* yang ditingkatkan nilainya, besar *throughput* rata-rata untuk trafik TCP dan tingkat TCP-*friendliness* akan meningkat. Sebaliknya, akan menurunkan nilai *throughput* rata-rata TFMCC dan persentase *packet loss*. Peningkatan jumlah penerima akan menurunkan *throughput* TCP dan TFMCC. Sementara itu, presentase *packet loss* akan meningkat. Variabel jumlah penerima tidak berpengaruh signifikan terhadap tingkat TCP-*friendliness*.

Kata kunci: *Congestion Control*, TCP, TCP-*friendly*, TFMCC

Abstract: *New trends in communications, especially the distribution of multicast streaming applications and real-time audio / video, will increase non-TCP traffic on the internet. These non-TCP applications are rarely to perform congestion control as well as TCP-based applications, so the local buffer will be full very quickly. To avoid congestion on non-TCP traffic that was sent in a streaming multicast, appear TFMCC algorithm. Jörg Widman version of TFMCC simulation was used on the same network to determine the network performance. The simulation result shows that on the same network conditions, throughput of TFMCC is more stable than TCP's. Increasing the value of variable bandwidth on the bottleneck link, will increase the average throughput of TCP and TFMCC and the level of TCP-friendliness. Meanwhile, the percentage of packet loss will reduce. If variable packet size is increased, the average throughput of TCP traffic and TCP-friendliness level will increase. On the contrary, the average throughput value TFMCC and percentage of packet loss will reduce. Increasing*

the number of receiver would lose an average throughput of TCP and TFMCC. Meanwhile, an increasing number of recipients would increase the percentage of packet loss. Variable number of recipients does not significantly influence the level of TCP-friendliness.

Keywords: *Congestion Control*, TCP, TCP-*friendly*, TFMCC

I. PENDAHULUAN

Pada aplikasi-aplikasi non TCP biasanya jarang dilakukan *congestion control* seperti halnya pada aplikasi-aplikasi berbasis TCP, sehingga *buffer* lokal akan penuh dengan sangat cepat. Aplikasi-aplikasi non-TCP tidak berbagi kapasitas jaringan yang telah disediakan secara merata dengan aplikasi-aplikasi yang dibangun berdasarkan TCP, seperti *web browser*, FTP, atau *email*. Dengan keadaan yang demikian, tentu saja evolusi yang terjadi saat ini dapat menyebabkan meningkatnya *congestion* bagi trafik TCP. Karena alasan itulah, dikembangkan protokol-protokol *TCP-friendly* yang dapat bersaing dengan TCP untuk mendapatkan *bandwidth* yang sama.

Protokol *TCP-Friendly* mendefinisikan mekanisme adaptasi laju yang sesuai untuk trafik non-TCP yang kompatibel dengan mekanisme adaptasi laju dari TCP. Dari mekanisme dan aturan *congestion control* terhadap adaptasi laju tersebut dibuat aplikasi-aplikasi non-TCP menjadi *TCP-Friendly* dan menjanjikan sebuah mekanisme distribusi kapasitas jaringan yang sama dengan memperhatikan trafik TCP yang mengalir berdampingan sehingga *congestion* pada jaringan dapat dikurangi.[2]

Untuk menghindari terjadinya *congestion* pada trafik non-TCP yang dikirim secara *streaming multicast*, muncul algoritma *TCP-Friendly Multicast Congestion Control* (TFMCC) yang merupakan sebuah algoritma *congestion control* untuk aplikasi yang berdasarkan protokol non-TCP untuk transmisi *multicast.* Pada algoritma *congestion control* TFMCC, kecepatan kirim TFMCC akan diturunkan pada saat terjadi *congestion* dan sebaliknya kecepatan kirim akan dinaikkan jika tidak terjadi *congestion*, sehingga *throughput* TCP dan TFMCC pada penerima akan relatif sama.

II. TEORI PENUNJANG

II.1. Congestion Control[1]

Sebuah jaringan dikatakan mengalami *congestion* ketika terlalu banyak paket yang mencoba untuk mengakses *buffer* dari *router* yang sama, sehingga mengakibatkan sejumlah paket *drop*.

Banyak faktor yang menyebabkan *congestion*, diantaranya kecepatan paket yang datang melebihi kapasitas *outgoing link*, ketidakcukupan memori untuk menyimpan paket yang datang, trafik *bursty,* dan kurangnya kecepatan pemrosesan dari *processsor*.

Tujuan dari *congestion control* selain untuk mencegah *congestion collapse*, juga untuk memaksimalkan penggunaan *bandwidth*, menjamin stabilitas jaringan dan memastikan alokasi sumber yang merata.

Congestion dapat dicegah dengan memperbesar ukuran *buffer*, sehingga kemungkinan terjadi *congestion* dapat diminimalkan. Namun semakin besar ukuran *buffer* semakin lama *delay*-nya. Bahkan dalam kasus ekstrem bila ukuran *buffer* tak terhingga, paket yang dikirim tidak akan pernah sampai ditujuan.[1]

II.2. Klasifikasi congestion Control

Berdasarkan sifat *loop*-nya, *congestion control* dapat dibagi menjadi dua bagian, yaitu *open-loop* dan *closed-loop*.

Open-loop congestion control untuk mencegah terjadinya *congestion* dengan memastikan trafik yang dikirim dari sumbernya tidak akan menurunkan performansi jaringan tersebut.

Sementara itu, *closed-loop congestion control* akan bereaksi jika telah atau hampir terjadi *congestion* pada jaringan. Tidak seperti *open-loop* yang tidak bergantung pada *feedback* untuk mengatur trafiknya, jaringan yang menggunakan *closed-loop congestion control* harus memberi *feedback* pada sumber mengenai kondisi jaringan yang terkini. Pemberian *feedback* ini dapat secara implisit maupun eksplisit.

Ada tiga kategori Algoritma *congestion control*, yang pertama ("*the box is black*") berisikan sejumlah algoritma yang mempertimbangkan jaringan sebagai *black box,* yang diasumsikan memiliki *zero knowledge* selain *feedback* biner saat terjadi *congestion*. Pada kategori ini protokol benar-benar tidak tahu kondisi jaringan yang sebenarnya. Kategori kedua ("*the box is grey*") menggunakan pengukuran untuk memperkirakan *bandwidth* yang tersedia, level dari *contentiont* atau bahkan karakteristik sementara dari *congestion*. Pada ketegori ini, protokol hanya dapat memperkirakan kondisi jaringan berdasarkan hasil pengukurannya. Namun kategori ini memiliki kekurangan yaitu kemungkinan kesalahan estimasi kondisi jaringan sehingga menyebabkan tindakan yang keliru. Kategori ketiga yaitu ("*the box is green*") berisi bimodal *congestion control*, yang menghitung *fairshare* secara eksplisit. Bimodal *congestion control* ini memiliki kesamaan seperti *network-assisted control*, yaitu jaringan mengkomunikasikan kondisinya ke *transport layer*. Pada kategori ini, jaringan memberitahukan kondisinya yang terkini, sehingga tidak ada lagi kesalahan estimasi kondisi jaringan.

II.3. TFMCC (TCP Friendly Multicast Congestion Control)[3]

TFMCC merupakan sebuah skema *source-based, single-rate congestion control* yang dibangun berdasarkan mekanisme *unicast* pada *TCP-Friendly Rate Control* (TFRC) dan dikembangkan ke domain *multicast*. TFMCC akan stabil dan responsif pada kondisi jaringan yang jangkauannya lebar dan dalam skala penerima yang jumlahnya dapat mencapai ribuan. Untuk mendukung skalabilitas, fungsi *congestion control* ditempatkan sebanyak mungkin pada penerima. Setiap penerima secara kontinu menentukan kecepatan kirim *TCP-friendly* yang diinginkan dari pengirim ke penerima tersebut. Penerima terpilih kemudian melaporkan kecepatan tersebut ke pengirim melalui paket *feedback.*

Perbedaan tingkat respon antara TFMCC dan TCP dapat menyebabkan perbedaaan *throughput* yang signifikan dalam kasus kecepatan transmisi yang sangat rendah. Sebelum mengirimkan datanya, TFMCC membutuhkan perkiraan jumlah paket yang akan hilang untuk menghitung kecepatan kirim yang sama. Perkiraan ini tidak sepenuhnya akurat jika TFMCC hanya menerima beberapa paket dalam suatu Round-Trip Time (RTT). TFMCC tidak dapat dipergunakan bersamaan dengan TCP jika kapasitas jalur *bottleneck*-nya kurang dari 30 kbit/s.

TFMCC akan mendapatkan kecepatan kirim yang berbeda jauh dengan rata-rata kecepatan TCP jika ruang *buffer* pada jalur *bottleneck* sama sekali tidak mencukupi. Jika dibandingkan dengan TCP, TFMCC tidak mudah terpengaruh oleh ukuran *buffer* yang kecil. Hal ini disebabkan karena TFMCC memberi interval antar paket, sedangkan TCP mengirimkan

paketnya secara *back-to-back*. Sehingga TCP akan lebih banyak kehilangan paketnya jika ukuran *buffer* tidak mencukupi.

TFMCC didesain untuk aplikasi dengan ukuran paket yang tetap namun jumlah paket yang dikirim per detik berbeda-beda untuk mencegah terjadinya *congestion* pada jaringan. Beberapa aplikasi (seperti audio) membutuhkan interval antar paket yang tetap tetapi ukuran paketnya yang berbeda-beda (bukan kecepatan kirimnya) untuk mencegah *congestion*.

A. Protocol Overview[3]

Secara umum, mekanisme *congestion control* pada TFMCC adalah:

- Setiap penerima memasukkan informasi jumlah paket yang hilang dan nilai RTT paket untuk menghasilkan kecepatan kirim yang *TCP-friendly*.
- Melalui mekanisme *feedback suppression*, hanya sebagian dari penerima yang dapat mengirimkan *feedback*-nya untuk mencegah "ledakan" jumlah *feedback* yang masuk ke pengirim. Mekanisme ini memastikan bahwa penerima yang melaporkan kecepatan kirim yang diterimanya lebih rendah daripada yang diinginkannya akan mempunyai peluang yang besar untuk mengirimkan *feedback*-nya.
- Penerima yang *feedback*-nya tidak ditekan (*suppressed)*, melaporkan hasil perhitungan kecepatan kirim kepada pengirim yang disebut laporan penerima. Laporan ini berfungsi untuk menghitung kecepatan kirim yang sesuai dan memperbolehkan penerima untuk mengukur RTT-nya.
- Pengirim memilih penerima yang melaporkan kecepatan terima terendah sebagai *current limiting receiver* (CLR). Ketika *feedback* dengan kecepatan terima yang lebih rendah diterima oleh pengirim, penerima yang mengirimkan *feedback* tersebut menjadi CLR dan kecepatan kirimnya akan dikurangi sesuai dengan perhitungan penerima tersebut. Kecepatan kirim akan naik kembali, jika CLR mengirimkan pesan bahwa kecepatan terima sekarang lebih tinggi daripada yang terakhir.

Pengirim mengirimkan data streaming secara *multicast* kepada penerima dengan kecepatan yang terkontrol. Ketika *feedback* diterima, pengirim mengecek apakah diperlukan pemindahan CLR ke penerima lain dan menyesuaikan kecepatan kirimnya.

Tugas utama yang harus dilakukan pengirim TFMCC adalah

- Mengatur kecepatan kirim
- Mengontrol *feedback* dari penerima
- Membantu pengukuran RTT pada sisi penerima

B. Slowstart

TFMCC memakai mekanisme *slowstart* untuk mendapatkan *bandwidth* yang sama pada awal sesi dengan cepat. Selama *slowstart*, kecepatan kirim akan meningkat secara eksponensial. Seperti dalam mode *congestion control* normal, penerima dengan kecepatan terima terendah akan menjadi CLR. Ketika sebuah penerima tidak dapat menerima data pada kecepatan yang lebih cepat dari jalur *bandwidth*-nya, kecepatan kirim akan disesuaikan secara bertahap. *Slowstart* akan dihentikan sesaat setelah salah satu penerima mengalami kehilangan paket yang pertama kali dan kecepatan akan meningkat secara linear.

III. METODA SIMULASI

III.1. Perangkat Keras dan Lunak yang Digunakan

Perangkat keras yang digunakan adalah sebuah *notebook* dengan *operating system* Windows XP *Service Pack* 2, Cygwin versi 2.427, *patchfile* Eurane versi 111, Network Simulator 2 (NS-2) versi 2.28 *all-in-one*, skrip simulasi TFMCC versi Jorg Widmer, Notepad++ versi 5.8.1. dan Microsoft Office Excel 2007.

III.2. Kondisi Awal

- Seluruh *software* dan *patch* telah ter-*install* dan berjalan dengan benar.
- Buka skrip example_bottleneck.tcl, ubah nilai *bandwidth* pada *bottleneck link* menjadi 250 *kilobytes per second*. (khusus untuk variabel *bandwidth* pada *bottleneck link*)
- Buka skrip example_bottleneck.tcl, ubah nilai *packet size* menjadi 250 *bytes*. (khusus untuk variabel *packet size*)
- Buka skrip example_bottleneck.tcl, ubah jumlah pengirim menjadi 2 *node* pengirim. Salah satu *node* merupakan pengirim trafik TCP, sedangkan *node* yang lain adalah pengirim trafik TFMCC. Ubah jumlah penerima menjadi 2 *node* penerima. (khusus untuk variabel jumlah penerima)
- Simpan skrip example_bottleneck.tcl di *folder* home/username.

III.3. Prosedur Simulasi

- Jalankan program Cygwin.
- Jalankan skrip simulasi dengan mengetik ns example_bottleneck.tcl pada jendela cygwin.
- Hasil yang diperoleh berupa 2 buah file yang masing-masing bernama out_bottleneck.tr dan out_bottleneck.nam.
- Animasi hasil simulasi dapat dilihat melalui jendela nam dengan nama file out_bottleneck.nam yang secara otomatis terbuka setelah simulasi selesai.
- Jalankan fitur perl untuk menghitung *throughput* TCP dan TFMCC pada *node* 3, 5, 7, 9
- Ulangi simulasi tersebut hingga 5 kali.
- Seluruh hasil pengukuran akan tercatat dalam bentuk file tanpa ekstensi yang dapat diakses dengan program Notepad++.
- Lakukan lagi langkah-langkah di atas untuk nilai variabel yang berbeda.
- Salin seluruh data pengukuran dari Notepad++ ke Microsoft Office Excel untuk mendapatkan grafik yang diinginkan.
- Khusus untuk mendapatkan persentase *packet loss,* kurangkan jumlah paket kirim dengan jumlah paket terima, lalu dibagi dengan jumlah paket kirim, kemudian dikali 100%.
- Khusus untuk mendapatkan tingkat TCP-friendliness, cari nilai “selisih absolut *throughput* rata-rata TCP dan TFMCC kemudian dibagi dengan *throughput* TCP” pada setiap *node*.

III.4. Hasil Simulasi Yang Diharapkan

- Hasil grafik *throughput* TFMCC lebih stabil daripada TCP.

- Semakin besar *bandwidth* pada *bottleneck link* semakin besar *throughput* TCP dan TFMCC.
- Semakin besar *packet size* semakin besar *throughput* TCP, namun *throughput* TFMCC semakin kecil.
- Untuk jumlah penerima yang sama, *throughput* rata-rata TCP dan TFMCC relatif berimbang.
- Semakin besar *bandwidth* pada *bottleneck link* semakin kecil persentase *packet loss*.
- Semakin besar *packet size* semakin kecil persentase *packet loss*.
- Semakin banyak jumlah penerima semakin besar persentase *packet loss*.
- Semakin besar *bandwidth* pada *bottleneck link* semakin besar tingkat *TCP- friendliness*.
- Tingkat *TCP-friendliness* akan meningkat seiring dengan peningkatan *packet size*.
- Jumlah penerima tidak berpengaruh terhadap tingkat *TCP- friendliness*.

IV. HASIL SIMULASI DAN ANALISIS DATA

IV.1. Throughput Berdasarkan Bandwidth Pada Bottleneck Link

Dari grafik-grafik Gambar 1, dapat dilihat kecenderungan nilai *throughput* TCP dan TFMCC akan meningkat seiring dengan kenaikan *bandwidth*. Hal ini menunjukkan bahwa semakin besar *bandwidth* pada *bottleneck link*, semakin cepat pula sebuah paket dapat tiba di sisi lain link tersebut. Sehingga pada akhirnya, jumlah bit yang diterima penerima per detik (*throughput*) juga akan meningkat.

Penjelasan ini dapat dianalogikan sebagai berikut. *Bandwidth* dianalogikan sebagai ukuran lebar sebuah jalan raya, sedangkan paket adalah kendaraan yang melintasinya. Maka semakin lebar sebuah jalan, maka semakin banyak pula kendaraan yang dapat tiba di ujung jalan tersebut per detiknya (*throughput*).

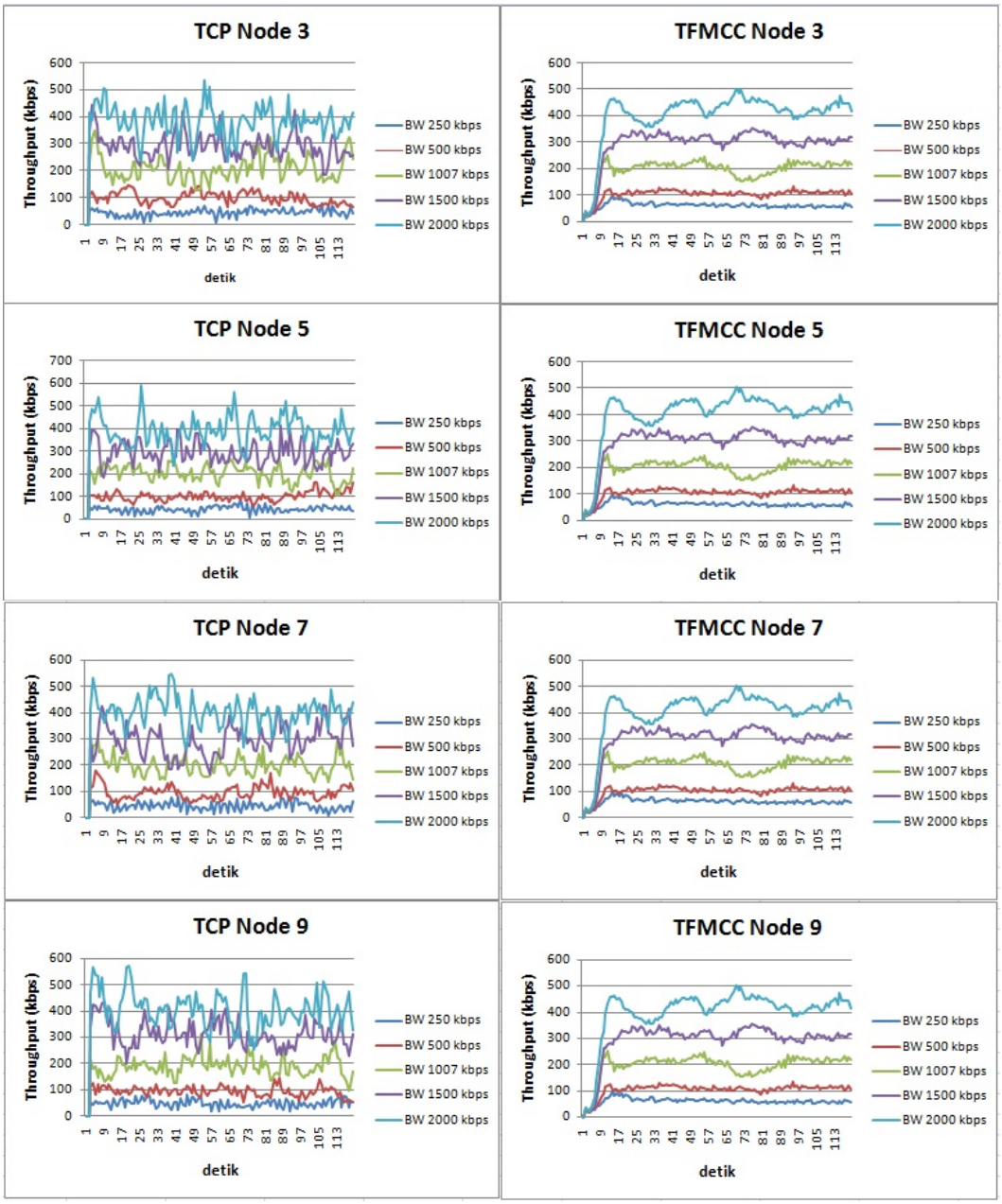
Pada lebar *bandwidth* yang sama, *throughput* TCP dan TFMCC dapat dikatakan berimbang. Jika terdapat perbedaan, nilainya tidak terlalu jauh. Perbedaan *throughput* itu disebabkan kondisi jaringan secara keseluruhan yang dinamis yang selalu berubah setiap saat.

Kondisi jaringan yang dinamis ini pula yang membuat paket trafik TFMCC yang secara teori akan diterima seluruh penerima secara bersamaan, tidak dapat terlaksana. Akan terdapat selisih waktu terima paket antara penerima yang satu dengan yang lain karena kondisi jaringan yang berbeda dari *router* ke setiap penerima.

Jika dicermati lebih mendalam lagi, baik trafik TCP maupun TFMCC, jika dikirim melalui *bandwidth* yang sempit, *throughput*-nya akan cenderung lebih stabil daripada yang dikirim melalui *bandwidth* yang lebar.

Hal ini disebabkan *congestion control* akan cenderung lebih bekerja keras saat *bandwidth* sempit agar tidak terjadi *congestion* dan *throughput* akan terjaga. Sedangkan jika *bandwidth* yang tersedia cukup bahkan berlimpah, *congestion control* tidak terlalu bekerja keras, dan akan cenderung ‘melepas’ *throughput* sesuai dengan dinamika kondisi jaringan terkini.

Pada akhirnya, *throughput*, baik TCP maupun TFMCC, pada *bandwidth* yang lebar akan lebih dinamis bahkan cenderung tidak stabil.

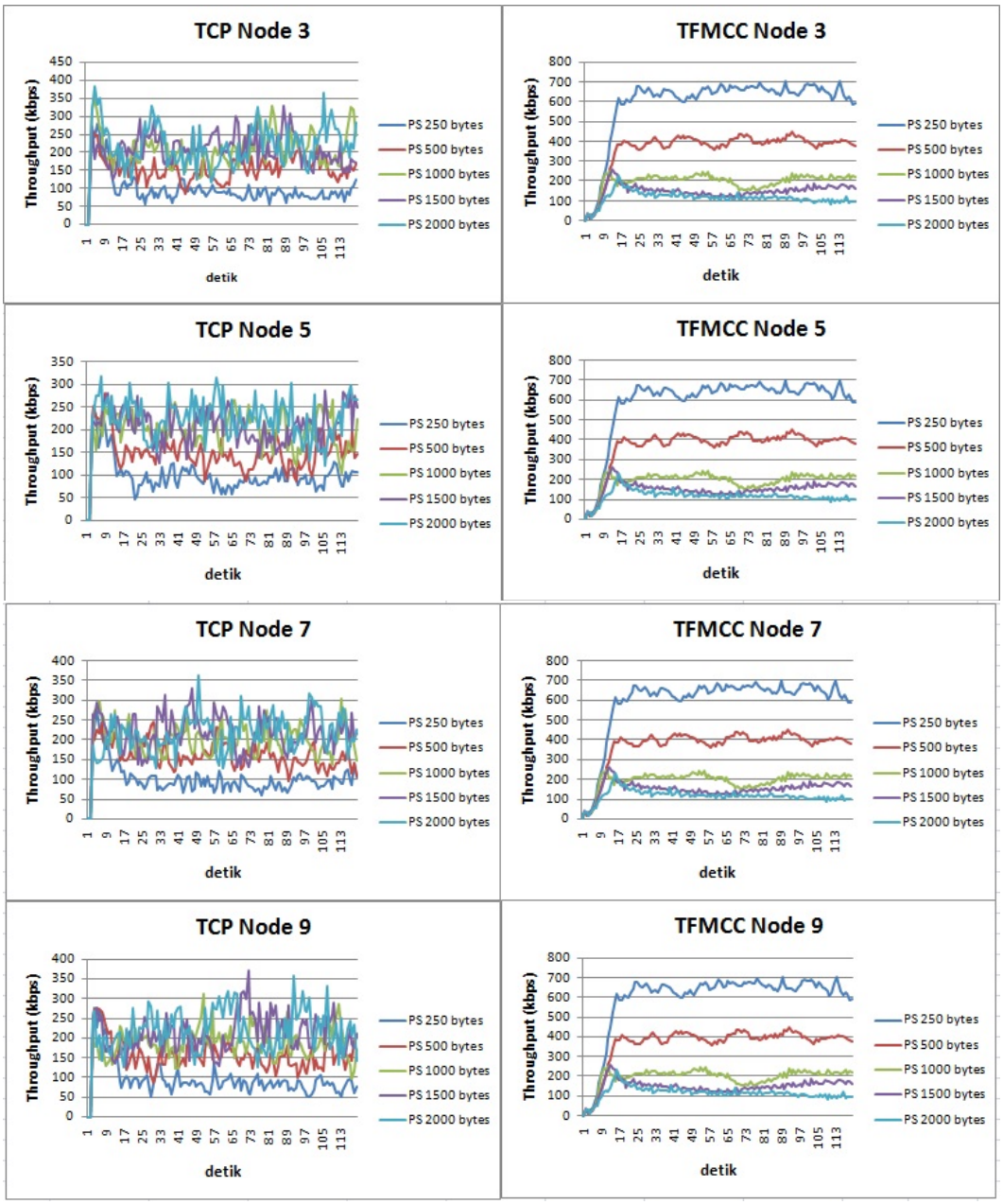


Gambar 1. Perbandingan throughput TCP dan TFMCC berdasarkan bandwidth bottleneck link

IV.2. Throughput Berdasarkan Packet Size

Pada grafik TCP, dapat dilihat perbandingan *throughput* TCP dengan *packet size* yang berbeda-beda. Semakin besar *packet size* akan cenderung semakin besar pula *throughput*-nya. Hal ini bertolak-belakang dengan grafik-grafik TFMCC yang *throughput*-nya semakin kecil seiring dengan semakin besar *packet size*-nya.

TFMCC akan mendapatkan kecepatan kirim yang berbeda jauh dengan rata-rata kecepatan TCP jika ruang *buffer* pada jalur *bottleneck* sama sekali tidak mencukupi. Secara khusus, jika dibandingkan dengan TCP, TFMCC tidak mudah terpengaruh oleh ukuran *buffer* yang kecil. Hal ini disebabkan oleh TFMCC memberi jarak waktu antar paket, sedangkan TCP mengirimkan paketnya secara berurutan (*back-to-back*). Sehingga TCP akan lebih banyak kehilangan paketnya jika ukuran *buffer* tidak mencukupi.[3]



Gambar 2. Perbandingan throughput TCP dan TFMCC berdasarkan packet size

Hal inilah yang terjadi pada PS 250 dan PS 500. Pengirim mampu mengirimkan paket data dalam jumlah banyak (jika dibandingkan dengan jumlah paket kirim PS 1000 – PS 2000) karena ukuran paketnya yang kecil. Namun ternyata *bandwidth* yang tersedia tidak cukup besar untuk melewatkan sekian banyak paket TCP dan TFMCC sekaligus. Jumlah paket yang memasuki *buffer node* 0 jauh lebih banyak daripada jumlah paket yang mampu diteruskan ke *bottleneck link*. Sesuai dengan penjelasan pada paragraf sebelumnya, hal ini menyebabkan banyak paket TCP yang *loss* di *buffer node* 0. Pada akhirnya *throughput* TCP di penerima akan jauh di bawah *throughput* TFMCC.

Pada PS 1000, ukuran paket yang lebih besar menyebabkan jumlah paket yang mampu dikirimkan oleh pengirim ke *buffer node* 0 menurun. Sehingga *buffer* tidak sepenuh saat

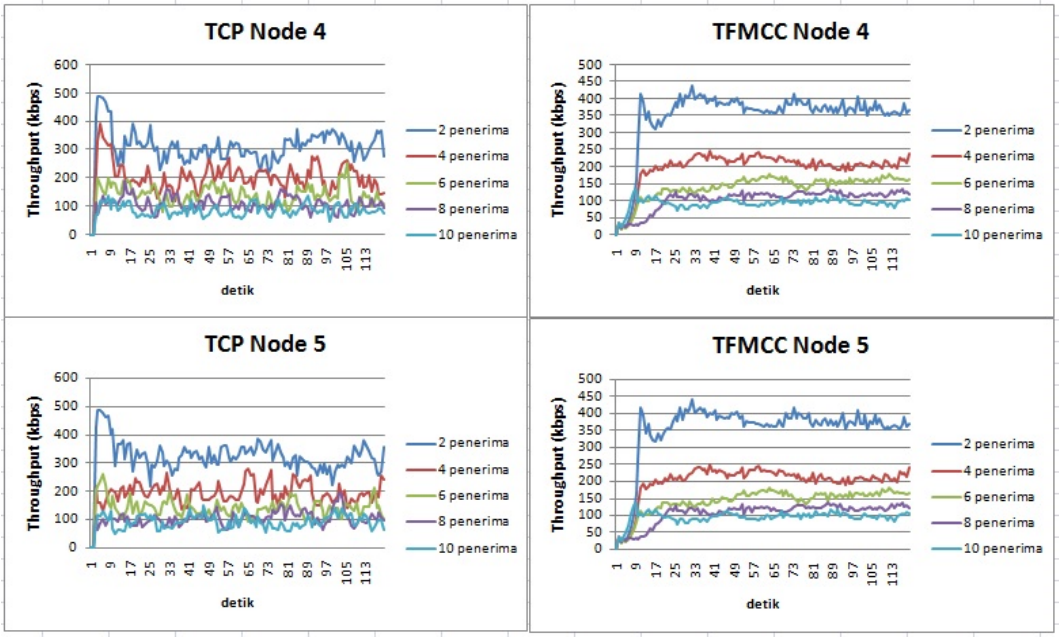
melewatkan PS 250 dan PS 500. Inilah yang menyebabkan pada PS 1000 *throughput* rata-rata TCP dan TFMCC relatif berimbang.

Seperti yang telah diketahui, TCP mengirimkan paketnya secara berurutan atau *back to back*. Selain itu, pada PS 1500 dan PS 2000, *packet size* TCP yang relatif besar akan membuat *buffer* pada *node* 0 menjadi cepat penuh. Untuk menghindari *congestion*, TFMCC "mengalah" dengan cara memberi jarak waktu antar paket yang lebih lama. Hal ini akan mengakibatkan jumlah paket kirim TFMCC per detik jauh lebih sedikit daripada jumlah paket kirim TCP per detik, sehingga *throughput* TFMCC jauh lebih rendah daripada TCP.

IV.3. Throughput Berdasarkan Jumlah Penerima

Grafik-grafik pada Gambar 3 menunjukkan bahwa semakin banyak penerima, baik TCP maupun TFMCC, *throughput*-nya akan semakin kecil. Jika perbedaan *throughput* untuk 2 penerima dan 4 penerima cukup jauh, perbedaan *throughput* untuk 6, 8, dan 10 penerima cukup sedikit, bahkan ada beberapa bagian yang saling *overlapping*.

Kasus ini dapat dijelaskan sebagai berikut. Trafik TCP merupakan trafik yang dikirim secara *unicast*. Setiap penerima yang terhubung ke *server* memerlukan tambahan *bandwidth* pada jaringan. Misalnya jika seorang penerima menerima data 100 kbps, dan ada 10 orang yang hendak menerima data, maka dibutuhkan *bandwidth* sebesar 1000 kbps. Namun, jika lebar *bandwidth* yang tersedia dalam lingkungan jaringan yang *best effort*, tidak mencukupi kebutuhan seluruh penerima maka *throughput* setiap penerima TCP akan 'dikorbankan'. Sehingga *throughput* yang didapat oleh setiap penerima lebih kecil daripada yang dibutuhkan.



Gambar 3. Perbandingan throughput TCP dan TFMCC berdasarkan jumlah penerima

Berbeda dengan trafik *unicast*, setiap penambahan penerima trafik *multicast* tidak akan memerlukan sejumlah *bandwidth* tambahan dan tidak menambah *overhead* pada *server*. Hal ini disebabkan *server multicast* hanya mengirimkan satu *stream* per sesi *multicast* untuk seluruh penerima dalam sebuah grup penerima dalam waktu yang hampir bersamaan. Dengan

penjelasan tersebut, seharusnya penambahan jumlah penerima *multicast*, tidak akan menurunkan *throughput* TFMCC.

Namun pada kasus ini, karena trafik *multicast* dikirim bersamaan dengan pengiriman trafik TCP yang *unicast*, *throughput multicast* TFMCC terkena dampak dari penurunan jumlah *bandwidth* yang tersedia pada *bottleneck link.* Hal ini menyebabkan *throughput* TFMCC juga mengalami penurunan seiring dengan penambahan jumlah penerima TFMCC.

IV.4. Persentase packet loss berdasarkan bandwidth bottleneck link

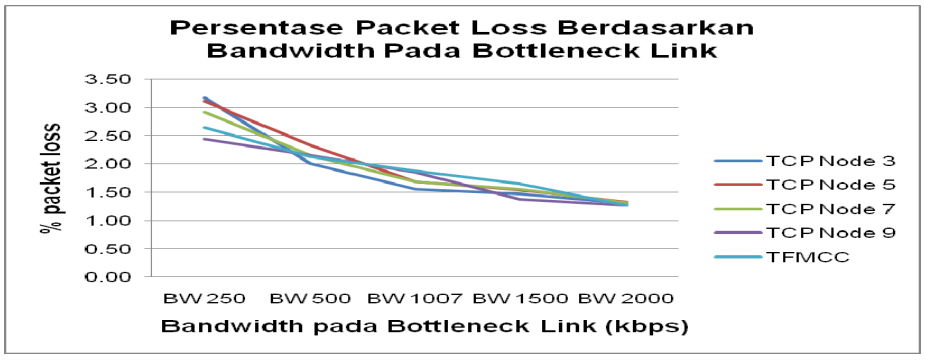
Berdasarkan Gambar 4. dapat dilihat perbedaan persentase *packet loss* untuk trafik TCP pada *node* 3, 5, 7, 9 dan trafik TFMCC berdasarkan *bandwidth* pada *bottleneck link*. Grafik tersebut menunjukkan kecenderungan penurunan persentase *packet loss* seiring dengan bertambahnya *bandwidth* pada *bottleneck link.*

Analoginya sebagai berikut. Sebuah jalan yang cukup lebar akan mampu menampung lebih banyak kendaraan daripada jalan yang sempit. Semakin sempit sebuah jalan, semakin besar kemungkinan terjadinya kemacetan total pada jalan tersebut.

Jika pada jalan tersebut terjadi macet total, tentu seorang polisi lalu lintas akan “menolak” kendaraan lain yang akan memasuki jalan tersebut.

Semakin sempit jalan, semakin besar pula kemungkinan sebuah kendaraan akan “ditolak” masuk. Kendaraan yang “tertolak” inilah yang disebut *packet loss*.

Dari data hasil simulasi dapat dilihat bahwa semakin besar *bandwidth* pada *bottleneck link* jumlah paket kirim dan paket terima akan semakin banyak. Namun penambahan jumlah paket kirim dan terima ini justru berdampak pada penurunan persentase *packet loss*.



Gambar 4. Perbandingan packet loss berdasarkan bandwidth bottleneck link

Hal ini disebabkan dengan semakin besar *bandwidth* pada *bottleneck link*, jumlah paket yang keluar dari *buffer* dapat menyamai atau setidaknya mendekati jumlah paket yang masuk. Sehingga waktu antre setiap paket yang memasuki *buffer node* 0 (nol) akan semakin singkat. Hal ini akan memperkecil kemungkinan paket tersebut dibuang karena *buffer node* 0 benar-benar penuh.

Selain itu, waktu mengantre yang singkat juga dapat menurunkan kemungkinan tercapainya kondisi *time out* pada pengiriman sebuah paket. Kondisi *time out* terjadi jika sebuah paket yang dikirimkan belum diterima oleh penerima hingga batas waktu yang telah ditentukan oleh pengirim. Jika kondisi ini terjadi, *node* pengirim akan mengirim ulang paket tersebut. Tentu saja hal ini akan menambah panjang daftar paket yang antre pada *buffer node* 0 atau mungkin memperbanyak jumlah paket yang dibuang. Sehingga pada akhirnya, persentase

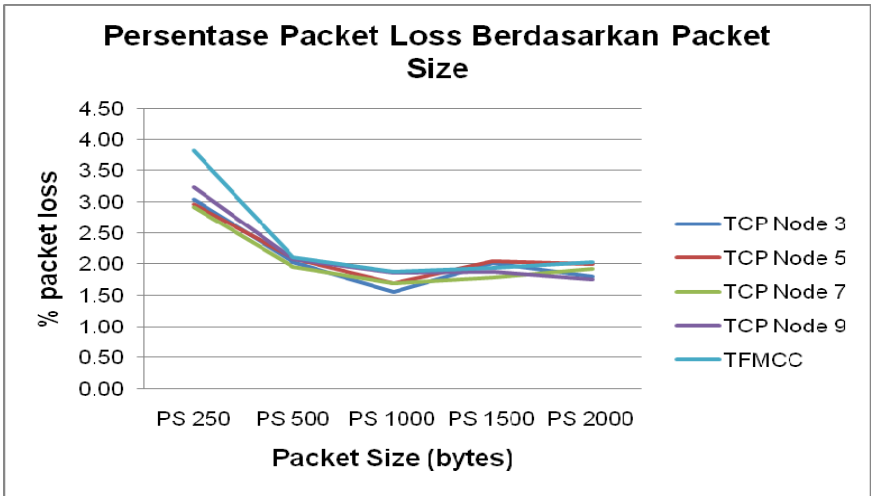
packet loss akan meningkat signifikan. Oleh karena itu, penambahan lebar *bandwidth* pada *bottleneck link* dapat menurunkan persentase *packet loss*.

IV.5. Persentase packet loss berdasarkan packet size

Berdasarkan Gambar 5. dapat dilihat perbedaan persentase *packet loss* untuk trafik TCP pada *node* 3, 5, 7, 9 dan TFMCC berdasarkan *packet size*. Grafik tersebut menunjukkan kecenderungan penurunan persentase *packet loss* seiring dengan bertambahnya *packet size*, namun cenderung stabil setelah melewati nilai tertentu, yaitu ps1000.

Pada PS 250 dan PS 500, karena ukuran paketnya yang kecil, pengirim mampu mengirimkan paket data dalam jumlah banyak (jika dibandingkan dengan jumlah paket kirim PS 1000 – PS 2000). Namun ternyata *bandwidth* yang tersedia tidak cukup besar untuk melewatkan sekian banyak paket TCP dan TFMCC sekaligus.

Jumlah paket yang memasuki *buffer node* 0 jauh lebih banyak daripada jumlah paket yang mampu diteruskan ke *bottleneck link*. Hal ini menyebabkan banyak paket TCP dan TFMCC yang hilang pada *buffer node* 0. Pada akhirnya persentase *packet loss* pada PS 250 dan PS 500 akan cukup besar.



Gambar 5. Perbandingan packet loss berdasarkan packet size

Pada PS 1000 – PS 2000, ukuran paket yang lebih besar menyebabkan jumlah paket yang mampu dikirimkan oleh pengirim ke *buffer node* 0 menurun. Sehingga *buffer* tidak sepenuh saat melewatkan PS 250 dan PS 500. *Buffer* punya jeda waktu yang cukup untuk melewatkan sebuah paket sebelum menerima paket selanjutnya. Hal inilah yang menyebabkan pada *packet size* ini persentase *packet loss* TCP dan TFMCC relatif stabil.

Analoginya, sebuah *buffer* adalah seorang penangkap bola dan ukuran paket sebagai ukuran bolanya. Pada BW 250, ukuran paketnya relatif kecil dan ringan, sehingga mudah dilempar dan diterima. Namun karena terlalu mudah dilempar, maka pelempar akan melempar terlalu banyak bola sehingga akan menyulitkan penangkap bola. Akibatnya, akan terdapat sejumlah bola yang tidak dapat ditangkap dengan benar dan terjatuh. Bola yang terjatuh itulah yang disebut dengan *packet drop* atau *packet loss*.

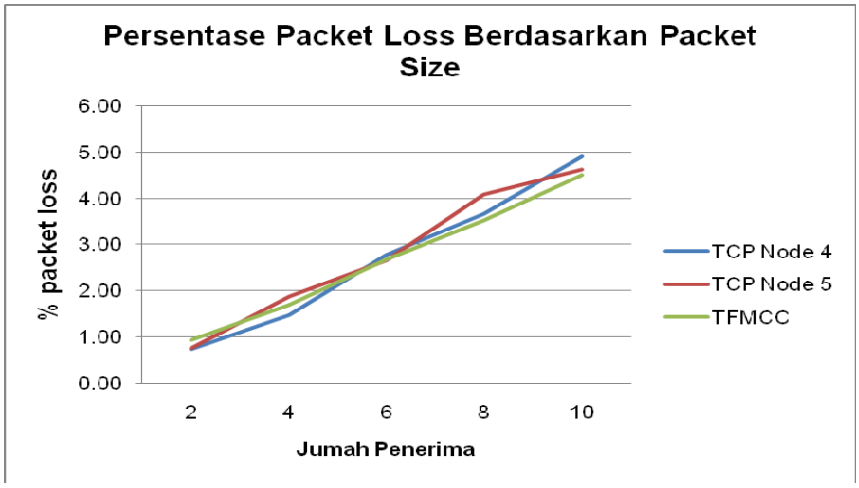
Semakin besar ukuran sebuah bola, bola tersebut akan semakin sulit untuk dilempar. Sehingga pelempar bola tidak akan mampu melemparkan bola sebanyak melemparkan bola

kecil. Akibatnya penangkap bola juga tidak akan terlalu sulit menangkap bola dan kemungkinan bola tersebut terjatuh juga cenderung semakin kecil.

Kesimpulannya adalah mengirimkan trafik dengan *packet size* yang sedikit lebih besar dapat mengurangi persentase *packet loss* walaupun jumlah paket kirim dan paket terima per detiknya tidak sebanyak pengiriman dengan *packet size* kecil. Kondisi seperti ini memerlukan penelitian lebih lanjut untuk mendapatkan titik optimal antara jumlah minimal paket terima per detik yang diinginkan dengan persentase *packet loss* maksimal yang diizinkan.

IV.6. Persentase packet loss berdasarkan jumlah penerima

Berdasarkan Gambar 6. dapat dilihat perbedaan persentase *packet loss* TCP pada *node* 4 dan 5 serta TFMCC. Dapat disimpulkan semakin banyak jumlah penerima akan semakin besar pula persentase *packet loss*-nya.



Gambar 6. Perbandingan packet loss berdasarkan jumlah penerima

Hal ini dapat terjadi karena semakin banyak penerima trafik TCP yang *unicast*, akan semakin banyak pula paket yang dikirimkan oleh *node* pengirim, sehingga menyebabkan *buffer* pada *node* 0 menjadi cepat penuh. Jika sebuah *buffer* sudah benar-benar penuh, *buffer* tersebut akan menolak paket yang akan masuk, sehingga terjadi *packet loss*. Jumlah *packet loss* ini akan semakin banyak seiring dengan meningkatnya jumlah penerima.

S*erver multicast* hanya mengirimkan 1 *stream* per sesi *multicast*, penambahan penerima TFMCC seharusnya tidak berdampak pada peningkatan *packet loss* pada trafik TFMCC. Namun karena pada simulasi ini trafik TCP dan TFMCC melalui *buffer* dan *bottleneck link* yang sama, persentase *packet loss* pada trafik TFMCC mengalami peningkatan sebagai dampak dari peningkatan persentase *packet loss* trafik TCP.

IV.7. TCP-friendliness berdasarkan bandwidth bottleneck link

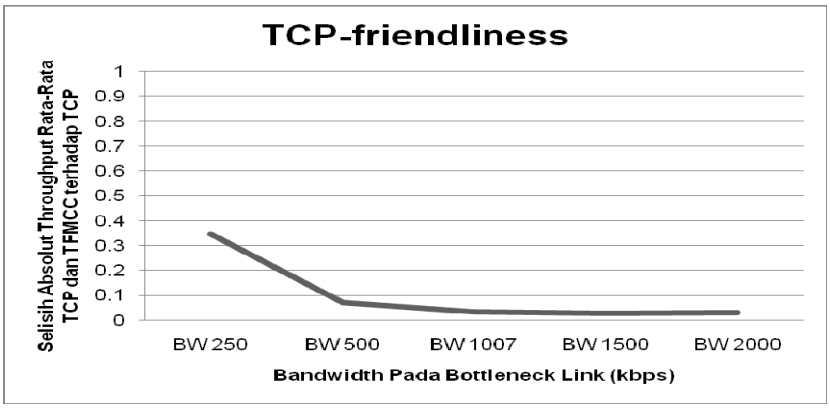
Sebuah algoritma non-TCP dapat dikatakan *TCP-friendly* jika *throughput*-nya dalam jangka panjang sama dengan *throughput* TCP dalam kondisi jaringan yang sama.[4]

Semakin kecil (mendekati nol) nilai “selisih absolut *throughput* rata-rata TCP dan TFMCC terhadap TCP” berarti tingkat *TCP-friendliness* trafik TFMCC tersebut semakin besar.

Nilai nol menandakan *throughput* rata-rata TCP sama dengan *throughput* rata-rata TFMCC. Nilai satu menandakan *throughput* rata-rata TCP dua kali lebih besar atau lebih kecil

daripada *throughput* rata-rata TFMCC.

Pada Gambar 7,. nilai "selisih absolut *throughput* rata-rata TCP dan TFMCC terhadap TCP" akan menurun seiring dengan bertambahnya *bandwidth* pada *bottleneck link*. Semakin besar *bandwidth* pada *bottleneck link*, tingkat *TCP-friendliness* trafik TFMCC tersebut semakin besar.



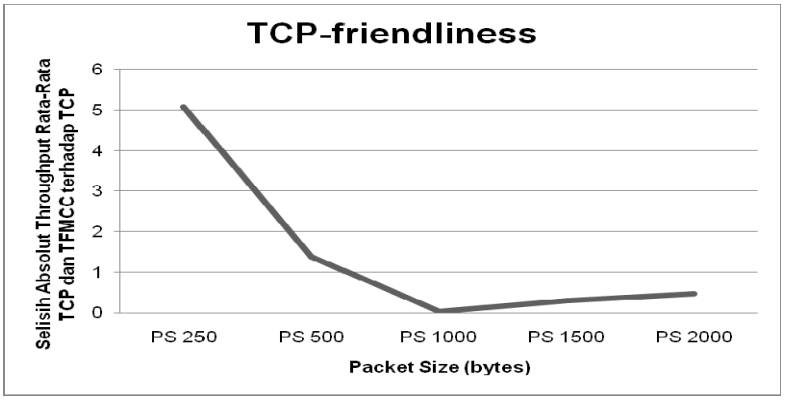
Gambar 7. Perbandingan TCP-friendliness berdasarkan bandwidth bottleneck link

Jika *bandwidth* pada *bottleneck link* semakin kecil, *throughput* trafik TCP akan 'tertekan' oleh *throughput* trafik TFMCC. Bahkan pada kasus ekstrem, yakni pada *bandwidth* yang sangat sempit, trafik TCP dapat mengalami kematian alias *throughput*-nya nol. Hal ini disebabkan TCP tidak mempunyai cukup *bandwidth* untuk menghindari dari 'tekanan' trafik TFMCC yang kuat. Sebaliknya, jika *bandwidth* yang tersedia cukup lebar, TCP akan mampu menghindari 'tekanan' TFMCC. Ini ditandai dengan *throughput* rata-rata TCP dan TFMCC yang relatif sama. Pada kondisi yang demikian, trafik TFMCC dapat dikatakan *TCP-friendly*.

IV.8. TCP-friendliness berdasarkan packet size

Semakin kecil nilai "selisih absolut *throughput* rata-rata TCP dan TFMCC terhadap TCP" berarti tingkat *TCP-friendliness* trafik TFMCC tersebut semakin besar.

Pada Gambar 8. dapat dilihat nilai "selisih absolut *throughput* rata-rata TCP dan TFMCC terhadap TCP" akan menurun dratis seiring dengan bertambahnya *packet size*. Dengan demikian, semakin besar *packet size*, tingkat *TCP-friendliness* trafik TFMCC tersebut semakin besar.



Gambar 8. Perbandingan TCP-friendliness berdasarkan packet size

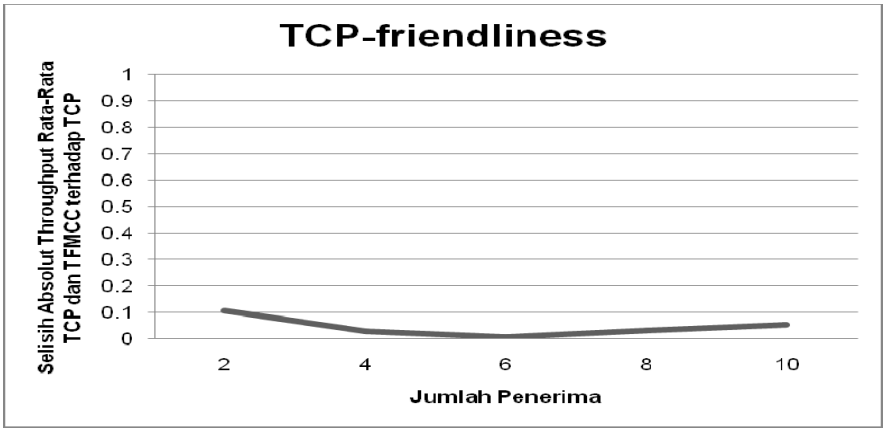
Jika melihat data *throughput* berdasarkan *packet size*, pada *packet size* 250 dan 500 *bytes*, grafik *throughput* TFMCC akan 'menekan' grafik *throughput* TCP.

Pada PS 250 dan PS 500, pengirim mampu mengirimkan paket data dalam jumlah banyak (jika dibandingkan dengan jumlah paket kirim PS 1000 – PS 2000) karena ukuran paketnya yang kecil. Namun ternyata *bandwidth* yang tersedia tidak cukup besar untuk melewatkan sekian banyak paket TCP dan TFMCC sekaligus.

Jumlah paket yang memasuki *buffer node* 0 jauh lebih banyak daripada jumlah paket yang mampu diteruskan ke *bottleneck link*. Sesuai dengan karakteristik TCP yang mengirimkan paket datanya secara *back-to-back*, TCP akan lebih banyak kehilangan paket pada *buffer node* 0 daripada TFMCC. Hal tersebut akan membuat *throughput* TCP di penerima akan jauh di bawah *throughput* TFMCC. Perbedaan *throughput* yang sangat jauh itulah yang menyebabkan tingkat *TCP-friendliness* trafik TFMCC dengan PS 250 dan 500 sangat rendah.

Pada PS 1000 - 2000, ukuran paket yang lebih besar menyebabkan jumlah paket yang mampu dikirimkan oleh pengirim ke *buffer node* 0 menurun. Sehingga *buffer* tidak sepenuh saat melewatkan PS 250 dan PS 500. Inilah yang menyebabkan pada PS 1000 - 2000 *throughput* rata-rata TCP dan TFMCC relatif berimbang, sehingga pada akhirnya tingkat TCP-friendliness pun meningkat.

IV.9. Friendliness berdasarkan jumlah penerima



Gambar 9. Perbandingan TCP-friendliness berdasarkan jumlah penerima

Gambar 9 menunjukkan tingkat *TCP-friendliness* berdasarkan jumlah penerima yang berbeda-beda, mulai dari 2 hingga 10 penerima. Nilai "selisih absolut rata-rata *throughput* TCP dan TFMCC terhadap TCP" tidak berubah signifikan, sehingga tingkat *TCP-friendliness* tidak berubah signifikan seiring dengan bertambahnya jumlah penerima. Hal ini dapat terjadi karena setiap penambahan *node* penerima, *throughput* TCP dan TFMCC sama-sama menurun dengan tingkat penurunan yang relatif sama. Penurunan *throughput* yang relatif sama ini yang menyebabkan tingkat *TCP-friendliness* trafik TFMCC relatif tidak berubah. Kalaupun terjadi perubahan, hal ini lebih disebabkan oleh kondisi jaringan yang dinamis. Dengan demikian dapat disimpulkan bahwa *TCP-friendliness* trafik TFMCC terhadap trafik TCP tidak bergantung pada jumlah penerima. Jumlah penerima hanya akan berdampak pada besarnya *throughput* dan persentase *packet loss*.

V. PENUTUP

V.1. Simpulan

- *Congestion control* pada TFMCC mempunyai peran yang besar terhadap performansi TFMCC pada jaringan internet.
- Pada kondisi jaringan yang sama, grafik *throughput* TFMCC lebih stabil (tidak fluktuatif; *range*-nya lebih kecil) daripada TCP.
- Peningkatan nilai variabel *bandwidth* pada *bottleneck link*, akan meningkatkan *throughput* rata-rata TCP dan TFMCC serta tingkat *TCP- friendliness*. Pada simulasi ini, kedua parameter *throughput* akan mencapai nilai tertingginya saat melalui *bandwidth* sebesar 2000 kbps, sedangkan *TCP-friendliness* saat lebar *bandwidth* sebesar 1500 kbps. Sementara itu, peningkatan *bandwidth* pada *bottleneck link* akan menurunkan persentase *packet loss*, dengan nilai persentase *packet loss* terendah pada saat *bandwidth* pada *bottleneck link* bernilai 2000 kbps.
- Jika variabel *packet size* yang ditingkatkan nilainya, besar *throughput* rata-rata untuk trafik TCP dan tingkat *TCP- friendliness* akan meningkat, dengan nilai tertinggi masing-masing pada *packet size* 2000 dan 1000 *bytes*. Sebaliknya, peningkatan nilai variabel *packet size*, akan menurunkan nilai *throughput* rata-rata TFMCC dan persentase *packet loss*. Kedua parameter tersebut akan mencapai nilai tertingginya masing-masing pada *packet size* 250 dan 1000 *bytes*.
- Peningkatan jumlah penerima akan menurunkan *throughput* TCP dan TFMCC. *Throughput* rata-rata tertinggi akan didapat saat terdapat 2 *node* penerima. Sementara itu, peningkatan jumlah penerima akan menaikkan persentase *packet loss.* Parameter ini mencapai nilai terendahnya pada 2 *node* penerima. Variabel jumlah penerima tidak berpengaruh signifikan terhadap tingkat *TCP- friendliness*, walaupun tingkat *TCP-friendliness* dapat mencapai nilai optimalnya pada 6 penerima.

V.2. Saran

- Diperlukan simulasi atau penelitian lebih lanjut untuk menemukan nilai optimal antara *bandwidth* pada *bottleneck link*, *packet size* dan jumlah penerima.
- Diperlukan simulasi atau penelitian lebih lanjut yang memasukkan *error*, QoE (*Quality of Experience*), *jitter*, dan *delay/latency* sebagai parameternya.

DAFTAR PUSTAKA

[1] L. Garcia, Alberto, I. Widjaja, *Communication Network: Fundamental Concepts and Key Architectures*. New Delhi: McGraw-Hill, 2002.

[2] http://digilib.itb.ac.id/gdl.php?mod=browse&op=read&id=jbptitbpp-gdl-herusukoco-25077, 5 Agustus 2010.

[3] http://inst.eecs.berkeley.edu/~cs294-9/fa03/slides/TCPFriendly.ppt, 13 Januari 2011.

[4] http://www.faqs.org/rfcs/rfc4654.html, 2 Agustus 2010.