Perbandingan Kinerja Algoritma BIC, HTCP, dan FAST dalam Jaringan Kecepatan Tinggi Dengan Waktu Tunggu Besar pada Topologi Simple Network Menggunakan NS2

Rian Fahrizal
Jurusan Teknik Elektro, Universitas Sultan Ageng Tirtayasa Cilegon
Jln. Jend. Sudirman Km 3 Cilegon Indonesia

Abstrak—*Jaringan komputer kecepatan tinggi dengan waktu tunggu yang besar merupakan bentuk jaringan yang umum di masa depan. Pada jaringan ini algoritma TCP yang umum digunakan mengalami kesulitan di dalam melakukan pengiriman data. Ada beberapa algoritma yang telah digunakan yakni BIC, CUBIC, FAST, dan HTCP. Algoritma-algoritma ini perlu diuji untuk mengetahui kinerjanya jika diterapkan pada jaringan dengan topologi yakni simple network. Dari keempat algoritma tersebut algoritma BIC memiliki nilai kinerja secara keseluruhan yang paling baik dengan nilai yang paling kecil.*

Kata Kunci *: BIC, CUBIC, FAST, HTCP, fairness, throughput, rata-rata thorughput, stabilitas, node, simple network, slow start, dan fast convergance.*

Abstract -- *High high-speed and long-distance will be a common for future computer networks. TCP may have a problem utilizing the full bandwidth. Some algorithms have been implemented to remedy this problem such as BIC, CUBIC, FAST and HTCP algorithm. These algorithms need to be tested to determine its performance when applied to network with simple network topologies. The results showed that all of the four algorithms were not worked well in this topologiy. However, the test results obtained the best algorithm is BIC.*

Keywords *: BIC, CUBIC, FAST, HTCP, fairness, average throughput, stability, multihome, dumbbell, parkinglot, and simple network..*

I. PENDAHULUAN

TCP merupakan metode pengiriman data yang banyak yang digunakan di dalam jaringan komputer pada saat ini. Protokol ini bertanggung jawab di dalam pengiriman data antara komputer di Internet. Algoritma TCP telah digunakan di dalam pengiriman data sejak tahun 1980-an. Algoritma ini telah bekerja dengan baik di dalam melakukan pengiriman data, akan tetapi bermasalah di dalam melakukan pengiriman data dengan bandwidth yang besar dengan waktu tunggu yang lama. Algoritma ini membutuhkan waktu 1,5 jam untuk mencapai bandwidth 10 Gbps dengan waktu tunggu 100ms [11].

Isu yang berkaitan dengan perilaku TCP dengan kecepatan tinggi, dengan waktu tunggu yang besar telah menarik perhatian yang luas. Algoritma pengendalian kongesti Additive Increase Multiplicative Decrease (AIMD) yang ada pada TCP dibuat untuk membagi sumber daya jaringan komputer diantara pengguna secara adil dan efisien [10]. Akan tetapi, hasil penelitian menunjukan bahwa TCP tidak menunjukan hasil yang baik dalam jaringan komputer kecepatan tinggi, dengan waktu tunda yang besar. Untuk mengatasi masalah ini, banyak algoritma yang telah diusulkan yakni High Speeed TCP (HSTCP), Scalable TCP (STCP), FAST, Binary Increase Control (BIC) TCP, CUBIC TCP, Hamilton TCP (H-TCP), dan sebagainya. Akan tetapi di dalam penggunaannya hanya algoritma BIC [11], CUBIC [4][9] dan HTCP [7] yang telah digunakan oleh sistem operasi Linux dengan kernel 2.16., dan algoritma FAST [14] telah diterapkan di dalam alat yang telah dijual secara bebas.

Keempat algoritma yang telah digunakan ini, baik digunakan pada Linux maupun yang digunakan setelah membeli peralatan seperti pada algoritma FAST, apakah merupakan algoritma yang efektif digunakan di dalam penggunaan secara umum pada jaringan kecepatan tinggi dengan waktu tunggu yang lama. Pengujian algoritma-algoritma ini menggunakan simple network. Topologi ini menguji dan harus bisa menjawab:

1. Apakah rata-rata throughput yang digunakan dibagi secara merata pada semua node yang melakukan pengiriman?
2. Apakah stabilitas throughput yang dihasilkan memiliki nilai yang kecil?
3. Apakah nilai dari throughput yang dihasilkan benar-benar adil jika dibandingkan satu dengan yang lainnya?
4. Diantara keempat algoritma yakni BIC, CUBIC, FAST dan HTCP, algoritma manakah yang paling baik dengan menggunakan pengujian topologi multihome, dumbbell, parkinglot dan simple network?

Penerapan algoritma congestion control telah dilakukan di dalam kernel Linux 2.16 (BIC, CUBIC dan HTCP) dan juga alat yang menggunakan algoritma FAST. Walaupun telah digunakan akan tetapi masih belum dilakukan pengujian apakah algoritma-algoritma ini efektif diterapkan pada jaringan komputer dengan kecepatan tinggi.

Pengujian dengan menggunakan topologi simple network ditujukan untuk mengetahui apakah algoritma yang diterapkan efektif dan juga mencari algoritma yang paling baik diterapkan di dalam jaringan komputer yang umum. Pengujian menggunakan nilai-nilai rata-rata throughput, stabilitas, dan intra protocol fairness (Raj Jain Index)[2].

Pengujian ini juga mengetahui algoritma yang paling efektif digunakan di dalam jaringan komputer secara umum. Kemudian diharapkan memberi masukan untuk perbaikan di dalam algoritma, sehingga didapatkan algoritma yang terbaik yang bisa diterapkan pada jaringan komputer kecepatan tinggi secara umum.

II. LANDASAN TEORI

II.1. H-TCP [3][7]

Metode pengaturan bandwidth dilakukan dengan melakukan penambahan window. Awal pengiriman data algoritma slow start dijalankan sampai mencapai nilai batas tertentu yang telah ditentukan. Setelah nilai tersebut tercapai maka algoritma ini dijalankan. Pada saat muncul kongesti nilai ambang batas menjadi setengah dari nilai sebelumnya yang digunakan pada pengiriman paket selanjutnya.

HTCP menggunakan waktu lalu Δ sejak kejadian congesti terakhir, daripada cwnd, untuk menentukan jalur pada bandwidth-waktu tunda dan parameter penambahan AIMD berbeda sebagai fungsi Δ. Parameter penambahan AIMD juga ditentukan dengan jalur waktu round-trip untuk mengurangi ketidak adilan antara aliran dengan waktu round-trip yang berbeda. Pengurangan faktor AIMD ditentukan untuk meningkatkan utilisasi jalur berdasarkan pada estimasi ketentuan queue pada jalur. Secara lengkap

$$ACK: \quad cwnd \leftarrow cwnd + \frac{2(1-\beta)f_\alpha(\Delta)}{cwnd} \qquad (2.1)$$

Loss: cwnd ← g_ β (B)x cwnd (2.2)

Dengan

$$f_\propto(\Delta) = \begin{cases} 1 & \Delta \leq \Delta_L \\ \max(\bar{f}_\propto(\Delta)T_{min}, 1) & \Delta > \Delta_L \end{cases} \qquad (2.3)$$

$$g_\beta(B) = \begin{cases} 0{,}5 & \left|\frac{B(k+1)-B(k)}{B(k)}\right| > 0 \\ \min\left(\frac{T_{min}}{T_{max}}, 0{,}8\right) & selainnya \end{cases} \qquad (2.4)$$

dengan

cwnd = congestion window(bytes)
β = faktor backoff
Δ = delta
fα(Δ) = fungsi penambahan congestion window
gβ(B) = fungsi pengurangan congestion window
Tmin = waktu minimum(detik)
Tmax = waktu maksimum(detik)

dimana ΔL ditentukan batas sehingga algoritma update TCP standar digunakan dimana Δ ≤ ΔL . Fungsi penambahan kuadrat untuk fα ialah

$$\overline{f}_\alpha(\Delta) = 1+10(\Delta - \Delta_L) + 0{,}25\,(\Delta - \Delta_L)^2 \qquad (2.5)$$

dengan

fα(Δ) = fungsi penambahan dalam rentang waktu
ΔL = Delta limit

Tmin Tmax menghitung waktu round-trip yang telah ada. β(k + 1) adalah perhitungan throughput yang dicapai maksimum pada masa congesti terakhir.

II.2. BIC-TCP [11]

Seperti pada algoritma HTCP awal pengiriman data algoritma slow start dijalankan sampai mencapai nilai batas tertentu yang telah ditentukan. Setelah nilai tersebut tercapai maka algoritma ini dijalankan. Pada saat muncul kongesti nilai ambang batas menjadi setengah dari nilai sebelumnya yang digunakan pada pengiriman paket selanjutnya.

Algoritma ini memandang pengendalian congesti merupakan sebuah masalah pencarian dimana sistem dapat memberikan jawaban ya/tidak sebagai umpan balik melalui paket yang hilang yang menunjukan bahwa nilai pengiriman lebih besar dari kapasitas jaringan. Nilai dari window minimum dapat ditentukan sebagai besarnya window yang mengalir yang tidak terdapat paket yang hilang. Apabila ukuran window maksimum diketahui, dapat diterapkan teknik pencarian biner untuk menentukan besarnya window yang dituju untuk mencapai nilai tengah maksimum dan minimum. Pada saat penambahan sampai ke nilai yang diharapkan, jika menghasilkan paket hilang, nilai window dapat ditentukan sebagai nilai maksimum baru dan megurangi besarnya window setelah paket loss dapat ditentukan sebagai minimum yang baru. Nilai antara kedua nilai ini merupakan nilai yang diharapkan.

Hubungan untuk pendekatan ini ialah bahwa ketika jaringan menghasilkan loss antara minimum baru tapi tidak terjadi terlalu dekat dengan nilai minimum yang baru, nilai yang diharapkan pasti berada diantara dua nilai. Sampai mencapai nilai yang diharapkan dan tidak menghasilkan paket hilang, maka besar nilai yang ada menjadi nilai minimum yang baru, dan nilai yang diharapkan dihitung kembali. Proses ini berulang dengan melakukan pembaharuan nilai minimum dan maksimum sampai perbedaan antara nilai-nilai tersebut mencapai nilai dibawah batas, yang disebut dengan penambahan minimum (minimum increament Smin). Teknik ini disebut dengan binary search increase.

Binary search increase menjadikan pencarian bandwidth menjadi semakin agresif ketika perbedaan antara window yang ada dengan window yang diharapkan besar, dan menjadi kurang agresif pada saat besarnya window yang ada mendekati besarnya window yang diharapkan. Teknik unik dari protokol ini ialah

fungsi penambahannya dalam bentuk logaritmik, sehingga pada saat mendekati nilai saturasi besarnya penambahan window semakin berkurang. Protokol yang lain menambahkan nilai bandwidth sampai pada saat nilai saturasi sehingga penambahan nilai saturasi adalah nilai maksimum dari keadaan tersebut. Pada umumnya, jumlah paket yang hilang berbanding lurus dengan besarnya penambahan terakhir sebelum paket hilang. Oleh karena itu binary search increase dapat mengurangi paket yang hilang. Seperti yang diketahui, keuntungan utama dari binary search ialah dapat memberikan fungsi respon yang baik, yang dapat digabungkan dengan penambahan secara linear.

Untuk menjamin penyatuan yang lebih cepat dan RTT yang adil, digabungkan binary search increase dengan strategi penambahan linear. Pada saat jarak dari ukuran window dengan yang diinginkan terlalu besar, penambahan ukuran window secara langsung pada nilai tengah tersebut memberikan beban berat ke dalam jaringan. Pada saat jarak dari ukuran window yang ada dengan besarnya window yang diinginkan pada binary search increase lebih besar dari langkah maksimum yang disebut dengan penambahan maksimum (maximum increment Smax), ditambahkan nilai window dengan Smax sampai jarak menjadi kurang dari Smax, pada saat dimana penambahan window secara langsung ke dalam nilai yang diinginkan. Kemudian setelah pengurangan window besar, menambahkan nilai window secara linear dan selanjutnya penambahan secara logaritmik. Penggabungan dari binary search increase dan penambahan secara penjumlahan disebut dengan binary increase.

Penggabungan dengan strategi pengurangan secara pembagian, binary increase menjadi mendekati penambahan linear dalam window yang besar. Hal ini disebabkan window yang lebih besar menghasilkan pengurangan yang lebih besar dengan pengurangan dengan perkalian, sehingga membutuhkan rentang waktu yang lebih lama. Kemudian di saat ukuran window kecil, menjadi binary search increase dengan rentang waktu penambahan yang lebih sedikit.

Dapat dilihat bahwa di dalam model loss yang telah tersinkironisasi secara penuh, binary search increase digabung dengan pengurangan secara pengali bergabung menjadi sebuah nilai throughput yang adil. Sebagai contoh ada aliran dengan ukuran window yang berbeda, tapi dengan RTT yang sama. Karena window yang berukuran lebih besar berkurang lebih banyak pada pengurangan secara perkalian (dengan factor yang tetap β), waktu untuk mencapai target lebih lama untuk window yang lebih besar. Akan tetapi, penggabungannya membutuhkan waktu yang lama. Pada penambahan binary increase, membutuhkan log(d)-log(Smin) RTT untuk mencapai window maksimum setelah pengurangan window d. karena penambahan window dalam fungsi log, semakin besar window dan semakin kecil window dapat mencapai kembali nilai maksimumnya dengan sangat cepat pada waktu yang hampir bersamaan. Akan tetapi window yang lebih kecil mengalir dengan bandwidth yang lebih kecil dari yang lebih besar sebelum pengurangan window selanjutnya. Oleh karena itu dimodifikasi binary search increase sebagai berikut.

Dalam binary search increase, setelah pengurangan window, nilai maksimum dan minimum ditentukan. Contohnya nilai-nilai ini adalah max_wini dan min_wini untuk aliran i (i=1,2). Apabila nilai maksimum baru lebih kecil dari nilai sebelumnya, window ini memiliki tren penurunan. Kemudian diatur ulang nilai maksimum baru menjadi sama seperti nilai window yang diinginkan baru, dan selanjutnya diatur lagi nilai tujuan. Kemudian diterapkan penambahan binary increase normal. Bentuk strategi ini disebut fast convergence.

BIC-TCP menggunakan bentuk algoritma pencarian biner untuk memperbaharui cwnd. Nilai dari w1 dipelihara yang menentukan nilai setengah antara nilai cwnd sebelum dan setelah kejadian kehilangan terakhir. Aturan pembaharuan cwnd mencari dengan cepat penambahan cwnd pada saat hal ini mencapai jarak tertentu Smax dari w1, dan memperbaharui cwnd lebih lambat ketika nilainya mendekati w1. Perkalian backoff dari cwnd digunakan pada pendeteksian paket yang hilang, dengan asumsi faktor backoff β sebesar 0,8. Secara rinci,

$$Ack: \begin{cases} \delta = (w_1 - cwnd)/B \\ cwnd \leftarrow cwnd + \frac{f_\alpha(\delta, cwnd)}{cwnd} \end{cases} \quad (2.16)$$

$$Loss: \begin{cases} w_1 = \begin{cases} \frac{1+\beta}{1} x cwnd & cwnd < w_1 \\ cwnd & selainnya \end{cases} \\ w_2 = cwnd \\ cwnd \leftarrow \beta \; x \; cwnd \end{cases} \quad (2.17)$$

Dengan

$$f_\alpha(\delta, cwnd) \begin{cases} \frac{B}{\sigma} & (\delta \leq 1, cwnd < w_1 \\ & atau \; (w_1 \leq cwnd < w_1 + B) \\ \delta & 1 < \delta \leq S_{max}, \; cwnd < w_1 \\ \frac{w_1}{B-1} & B \leq cwnd - w_1 < S_{max}(B-1) \\ S_{max} & selainnya \end{cases}$$

(2.18)

dengan

σ	= distance
cwnd	= congestion window(bytes)
f(δ ,cwnd)	= fungsi penambahan window size
δ	= pengurangan
w1	= window(bytes)
w2	= window(bytes)
β	= backoff factor

BIC-TCP juga menerapkan sebuah algoritma dimana pada saat utiliasasi rendah terdeteksi, akan menambahkan window lebih agresif. Hal ini dikendalikan dengan parameter Low-Util dan Util_Check. Untuk menentukan kompabilitas sebelumnya, hal ini menggunakan parameter pembaharu TCP standar dimana cwnd dibawah batas Low-Window.

II.3. FAST-TCP [5][6][14]

II.3.1 . Arsitektur

Pembagian mekanisme pengendalian congesti TCP menjadi empat komponen. Keempat komponen berfungsi secara sendiri-sendiri sehingga dapat dibuat secara terpisah dan ditingkatkan secara terpisah seperti yang tetihat pada Gambar 1.

Komponen pengendalian data (data control) menentukan paket apa yang dikirimkan, pengendalian window (window control) menentukan berapa besar paket yang dikirim dan pengendalian Burstiness (Burstiness control) menentukan kapan pengiriman paket-paket ini dilakukan. Penentuan ditentukan berdasarkan pada informasi yang ditentukan pada komponen estimation.

Secara spesifik komponen estimasi menghitung dua bagian informasi umpan balik untuk setiap paket data mengirim waktu tunggu multibit queueing dan satu-bit indikasi loss-or-no-loss dengan digunakan dengan komponen tiga lainnya. Pengendalian data memilih paket selanjutnya yang dikirim dari tiga kandidat paket data tersimpan: paket baru, paket yang hilang, dan mengirim paket yang belum di acknowledge. Pengaturan window mengatur pengiriman paket pada sekala waktu RTT, dimana pengandalian Burstiness bekerja pada rentang waktu yang sedikit. Pengendalian Burstiness menghaluskan pengiriman paket data seperti aliran fluida untuk mengetahui bandwidth yang tersedia. Dibuat dua buah mekanisme, pertama memberikan clocking mandiri di dalam pengiriman paket data individu dan yang kedua menambahkan ukuran window secara halus dalam aliran yang lebih kecil. Pengurangan Burstiness membatasi jumlah paket yang dapat dikirim ketika sebuah ack diterima congestion window dengan jumlah yang banyak. Window pacing menentukan bagaimana menaikan congestion window pada saat waktu idle koneksi ke nilai yang diinginkan dengan komponen pengendalian window. Hal ini dilakukan dengan mengurangi Burstiness dengan jumlah yang sesuai jadwal.

Data Control	*Window* Control	*Burstiness* Control
Estimation		
TCP Protocol Processing		

Gambar 1. Arsitektur FAST TCP

II.3.2. Algoritma pengendalian window

FAST bekerja berdasarkan queuing delay dan paket loss. Pada kondisi normal, FAST secara teratur mengupdate congestion window berdasarkan rata-rata RTT dan queuing delay yang dihasilkan dari komponen estimasi dengan rumus:

$$Setiap\ RTT: \quad cwnd \leftarrow \frac{\left[cwnd+\frac{T_{min}}{\bar{T}}cwnd+f_{\alpha}(B)\right]}{2} \qquad (2.19)$$

$$Setiap\ Loss: \quad cwnd \leftarrow 0{,}5\ x\ cwnd \qquad (2.20)$$

dengan

cwnd	= congestion window(bytes)
Tmin	= waktu minimum(detik)
T	= waktu rata-rata (detik)
fαβ	= fungsi pengurangan

Dimana Tmin dan T adalah nilai latency minimum dan rerata yang diamati dari aliran data berurutan. Fungsi fα(β) tergantung dari perhitungan throughput β yang dicapai aliran: biasanya, fα(β) diberi nilai 8, 20 dan 200 untuk mencapai throughputs untuk kurang dari 10Mbit/sec, kurang dari 100Mbit/sec dan lebih besar dari 1Gbit/sec secara berurutan. FAST-TCP juga termasuk menggunakan tingkat pacing. Perhatikan bahwa tingkat pacing adalah fungsi yang berubah dan dilihat dimana hal ini merupakan bagian dari algoritma pengendalian congesti.

II.3.4. CUBIC-TCP [4][8][9]

Seperti yang dijelaskan pada algoritma HTCP dan BIC awal pengiriman data algoritma slow start dijalankan sampai mencapai nilai batas tertentu yang telah ditentukan. Setelah nilai tersebut tercapai maka algoritma ini dijalankan. Pada saat muncul kongesti nilai ambang batas menjadi setengah dari nilai sebelumnya yang digunakan pada pengiriman paket selanjutnya.

Seperti namanya fungsi penambahan window yang digunakan ialah fungsi cubic yang memiliki kesamaan dengan fungsi penambahan BIC-TCP. CUBIC menggunakan fungsi cubic untuk waktu yang berlalu dari kejadian congesti yang terakhir. Walaupun sebagian besar algoritma alternative menggunakan fungsi penambahan convex dimana setelah kejadian paket hilang, penambahan window selalu bertambah, CUBIC menggunakan bentuk concave dan convex dari fungsi cubic untuk penambahan window.

Penjelasan lebih detail dari fungsi ini ialah setelah pengurangan window karena kejadian paket yang hilang, didaftarkan Wmaz menjadi nilai window dengan munculnya kejadian window hilang dan menggunakan pengurangan dengan perkalian dari congestion window dengan menggunakan factor β dengan β adalah konstanta pengurangan window dan menggunakan algoritma fast recovery dan retransmit dari TCP. Kemudian setelah masuk ke algoritma congestion avoidance dari fast recovery, mulai ditambahkan window menggunakan bentuk concave pada fungsi cubic. Fungsi cubic ditentukan untuk memiliki nilai baru pada Wmax sehingga penambahan berlanjut sampai ukuran window menjadi Wmax.kemudian fungsi cubic berubah menjadi bentuk convex dan penambahan window convex dimulai. Bentuk penentuan window (concave dan selanjutnya convex) meningkatkan stabilitas protokol dan jaringan ketika mengatur utilisasi jaringan kecepatan tinggi. Hal ini disebabkan ukuran window hamper mendekati konstan, membentuk nilai baru dekat dengan Wmax dengan utilisasi jaringan tertinggi dan di dalam kondisi steady, sebagian besar ukuran window CUBIC mendekati Wmax, sehingga mempromosikan stabilitas utilisasi jaringan kecepatan tinggi dan protokol. Protokol dengan fungsi penambahan convex menjadi memiliki penambahan window terbesar sekitar nilai saturasi, dengan paket yang hilang banyak.

Algoritma ini menggabungkan ide dasar dari High-Speed TCP dan H-TCP. Disebut dengan penambahan cwnd sebagai fungsi dari waktu karena pemberitahuan terakhir dari congesti, dan besarnya window pada

pemberitahuan terakhir congesti. Bentuk dari algoritma ini dapat disimpulkan sebagai berikut:

1. Perubahan slow start. Perubahannya ditentukan pada awal. Sekali cwnd meningkat di atas ssthresh, CUBIC keluar dengan menggunakan algoritma slow start normal dan perubahan menggunakan penambahan eksponensial yang agresif dimana cwnd ditambahkan sebesar satu paket untuk setiap 50 ack yang diterima atau sama dengan dua kali cwnd mendekati setiap 35 waktu round-trip.
2. Faktor backodd 0,8. Pada paket yang hilang, cwnd dikurangi dengan faktor 0,8.
3. Clamp pada laju penambahan maksimum. Laju penambahan pada operasi AIMD dibatasi paling tidak 20*delay_min paket setiap RTT, dimana delay_min adalah perkiraan round-trip propagation waktu tunda dari aliran data. Merubah dari paket per RTT menjadi paket per detik, clamp ini kira-kira sama dengan laju penambahan 20 paket/detik tidak tergantung RTT.
4. Fungsi penambahan cubic. Subyek pada clamp ini, laju penambahan adalah paket target-cwnd per RTT. Perhatikan bahwa efek dari penambahan ini ialah menyesuaikan cwnd menjadi sama dengan target pada arah dalam RTT tunggal. Nilai dari target dihitung dari

$$target = W_{max} + C(t - \sqrt[3]{(\beta(W_{max} - 0{,}8W))})^3$$

dengan

Wmax = window maksimum(bytes)
W = window(bytes)
t = waktu (detik)
C = cubic
β = faktor pengurangan

dimana t adalah waktu berlalu sejak backoff terkahir (mendekati nilai delay_min ditambahkan ke dalam nilai ini) dan Wmax dihubungkan dengan cwnd pada backoff terkahir dan ditandai origin_point pada kode. W adalah nilai cwnd sebelum backoff terakhir, sehingga 0,8W adalah nilai cwnd nilai sebelum backoff muncul.

5. Adaptasi fungsi cubic. Nilai dari Wmax ditentukan tergantung dari apakah backoff terakhir muncul sebelum atau sesudah cwnd tercapai pada nilai Wmax sebelumnya. Atau jika tidak Wmax diset sama dengan 0,9W.

Algoritma ini juga mencakup kode untuk meyakinkan bahwa algoritma ini paling tidak seagresif seperti TCP yang ada.

II.4. Perhitungan Kinerja

Hasil dari simulasi yang dihasilkan menghasilkan nilai rata-rata throughput, stabilitas dan fairness. Nilai-nilai ini menunjukan kinerja dari simulasi, akan tetapi masih merupakan data yang terlampau banyak dan agak sulit untuk mengambil kesimpulan. Oleh karena itu untuk memudahkan di dalam analisis dilakukan penyederhanaan hasil perhitungan dengan cara menggunakan parameter simpangan rata-rata throughput terhadap nilai ideal yang diinginkan. Dengan menggunakan rumus:

$$y = \left|\frac{\bar{x} - Nilai\ ideal}{Nilai\ ideal}\right| \quad (2.22)$$

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \quad (2.23)$$

Dengan

y = kinerja untuk rata-rata throughput.
$\bar{x}$ = rata-rata throughput
Nilai ideal = kapasitas bandwidth / jumlah node yang digunakan
$\bar{y}$ = rata-rata kinerja rata-rata throughput

Kemudian di dalam penyerdahanaan perhitungan kinerja dengan menggunakan parameter fairness dilakukan penyederhanaan dengan menggunakan rumus, sebagai berikut:

$$z = 1 - \overline{fairness} \quad (2.24)$$

Dengan

z = kinerja fairness

$\overline{(fairness)}$ = rata-rata fairness

Parameter nilasi stabilitas perlu dilakukan perubahan untuk dapat melihat kinerja dengan menggunakan rumus, sebagai berikut

$$r = \frac{\bar{s}}{Nilai\ ideal} \quad (2.25)$$

Dengan

r = nilai kinerja stabilitas

Perhitungan ini dirangkum ke dalam satu paremeter kinerja algoritma dengan menggunakan rumus.

$$u = y + z + r \quad (2.26)$$

Dengan

u = nilai kinerja algoritma

Perhitungan kinerja algoritma ini ditunjukan dengan nilai tertentu, yang semakin kecil nilai tersebut, maka semakin baik kinerja algoritma suatu algoritma. Kemudian untuk nilai yang lebih besar menunjukan bahwa algoritma tersebut memiliki kinerja yang lebih buruk.

III. METODE PENELITIAN

III.1. Bahan Penelitian

Bahan penelitian yang digunakan di dalam pelaksanaan penelitian ini ialah:

1. Network Simulator ns2 Ver 2.34 dengan patch fast-tcp-ns2-v1_1c.
2. Gnuplot Ver 4.2 patchlevel 5.

3. Microsoft Excel 2007 untuk melakukan pengolahan data.
4. Text editor Gedit 2.28.
5. Linux OS ver 10.10. Maverick.

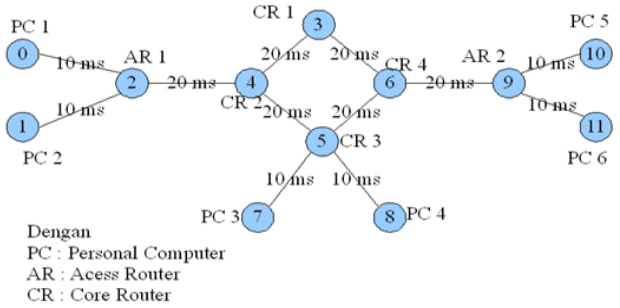
III.2. Alat Penelitian

Alat Penelitian yang akan digunakan didalam pelaksanaan penelitian adalah sebuah unit komputer dengan spesifikasi sebagai berikut:

1. Processor AMD Turion II X2 M520 dengan clock speed 2.3 Ghz, dan L2 cache 1 MB.
2. Memory RAM 2 GB.
3. Hard Disk Drive 320 GB.

III.3. Jalan Penelitian

Topologi simple network dapat dilihat pada Gambar 1. Pada konfigurasi ini, router inti menunjukan backbone jaringan dengan router perantara bertanggung jawab untuk node pengirim dan penerima terhubung dengan jaringan.



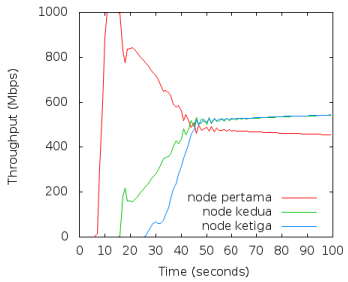
Gambar 1. Topologi Simple network [12]

Pada topologi ini node 0,1, dan 7 melakukan pengiriman data sedangkan node 8, 10, dan 11 menerima data yang dikirim. Topologi ini merupakan penyederhanaan topologi menggunakan transit dan stub [15].

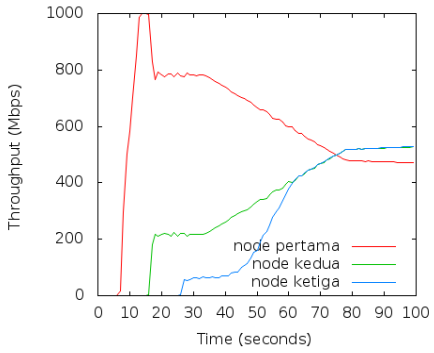
Program simulasi menggunakan algoritma FAST, H-TCP, BIC dan CUBIC pada metode pengiriman data. Hubungan antara node dapat dilihat pada Gambar 5.

III. ANALISIS HASIL SIMULASI

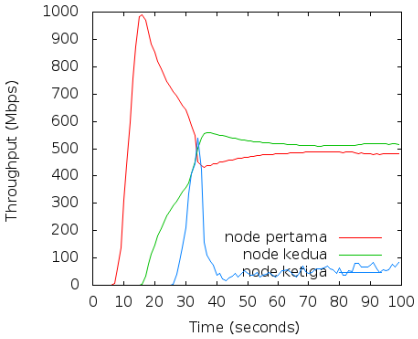
Hasil dari pengujian kali ini dapat dilihat pada Gambar 2 yang terdiri dari hasil throughput .



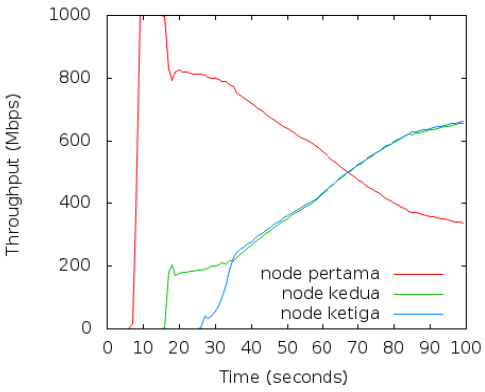
(a) Algoritma BIC



(b) Algoritma CUBIC



(c) Algoritma FAST



(d) Algoritma HTCP

Gambar 2. Nilai throughput simulasi topologi simple network

Seperti yang terlihat pada Gambar 2 (a) pada algoritma BIC selalu menggunakan algoritma slow start untuk memulai pengiriman data pada node pertama. Pada saat node kedua melakukan pengiriman digunakan algoritma fast convergance selama beberapa detik kemudian menggunakan algoritma BIC. Pada node ketiga yang melakukan pengiriman menggunakan algoritma BIC. Pada node kedua dan ketiga nilainya mendekati sama pada detik ke-40 sampai simulasi selesai. Akan tetapi pada node pertama nilai throughputnya semakin turun pada saat simulasi berakhir.

Seperti yang terlihat pada Gambar 2 (b) dalam algoritma CUBIC node pertama menggunakan algoritma slow start pada saat mencapai bandwidth > 400 Mbps kemudian menggunakan algoritma CUBIC untuk

mencapai nilai maksimumnya. Pada saat node kedua melakukan pengiriman kedua node menggunakan fast convergance selanjutnya menggunakan algoritma CUBIC untuk mencapai nilai yang cenderung stabil. Node ketiga menggunakan algoritma slow start di awal pengiriman selanjutnya menggunakan algoritma cubic sampai dicapai nilai yang cenderung stabil yang memiliki nilai hampir sama dengan nilai throughput node kedua. Dari hasil terakhir dari simulasi bahwa node kedua dan ketiga memiliki throughput yang lebih besar dari nilai throughput node pertama.

Pada Gambar 2 (d) node pertama menggunakan algoritma slow start pada saat mulai melakukan pengiriman data, kemudian menggunakan algoritma HTCP sampai > 600 Mbps pada throughputnya kemudian menggunakan fast convergance pada saat node kedua mulai melakukan pengiriman node kedua juga menggunakan algoritma fast convergence. Selanjutnya node pertama mengalami tren penurunan dengan menggunakan algoritma HTCP akan tetapi node kedua turun selama beberapa saat kemudian cenderung meningkat dengan tajam bahkan pada simulasi terakhir memiliki nilai throughput yang lebih besar dari nilai throughput node pertama. Node ketiga langsung menggunakan algoritma HTCP di dalam pengiriman datanya selanjutnya cenderung landai dan mendekati nilai pengiriman node kedua. Pada saat detik ke 100 memiliki nilai yang mendekati nilai throughput node kedua dan lebih besar dari node pertama.

Pada penerapan algoritma FAST pada topologi simple network. Node pertama mulai melakukan pengiriman data dan berhasil mencapai titik maksimum kemudian mulai turun setelah node kedua mulai melakukan pengiriman. Node kedua mulai melakukan pengiriman naik sampai > 500 Mbps pada throughputnya kemudian mengalami tren penurunan dan cenderung stabil mendekati 500 Mbps.

Hasil simulasi ini menjukan bahwa algoritma FAST mengalami kegagalan pada topologi ini. Hal ini ditunjukan dengan kecilnya nilai throughput pada node ketiga. Algoritma yang paling baik di dalam topologi ini ialah algoritma BIC dan algoritma ini juga sesuai dengan nilai throughput pada penelitian sebelumnya dimana nilai throughput terbagi antara node pertama dan kedua serta node pertama dan node ketiga dan mendekati dengan penelitian sebelumnya (Wei 2006).

Pengujian pada simulasi ini menunjukan bahwa keempat algoritma memiliki nilai yang selalu berubah-ubah hingga akhir simulasi. Akan tetapi algoritma BIC memiliki nilai akhir throughput yang tidak terlalu besar perbedaannya antara node-node-nya. Kemudian algoritma CUBIC membutuhkan waktu yang lama untuk mencapai nilai throughput yang mendekati stabil. Sedangkan algoritma HTCP tidak mencapai nilai yang stabil atau mendekati stabil hingga akhir simulasi. Pada algoritma FAST terdapat kegagalan di dalam pengiriman pada node ketiga.

Dari hasil pengujian diambil nilai rata-rata throughput yang ditunjukan dalam Tabel 1. Berikut ini.

Tabel 1. Nilai rata-rata throughput topologi simple network

Algoritma	*Node* 1 $\bar{x}$(Mbps)	*Node* 2 $\bar{x}$(Mbps)	*Node* 3 $\bar{x}$(Mbps)
BIC	448.4048	549.7024	440.7162
CUBIC	405.0986	593.3803	383.8852
FAST	462.7857	535.2024	82.36486
HTCP	410.7262	550.1757	433.4595

Tabel 1. menunjukan algoritma yang memiliki rata-rata throughput yang mendekati sama antara ketiga node ialah algoritma BIC dengan memiliki nilai node pertama (node 1) 448,4 Mbps, node kedua (node 2) 549,7 Mbps dan node ketiga 440,7 Mbps. Sedangkan untuk algoritma FAST hanya node pertama dan kedua yang memiliki nilai rata-rata throughput yang mendekati sama yakni 462,8 Mbps untuk node pertama dan 535,2 Mbps untuk node kedua. Akan tetapi untuk node ketiga memiliki rata-rata throughput 100,3 Mbps. Kemudian untuk algoritma HTCP memiliki nilai rata-rata throughput yang mendekati sama antara ketiga node-nya yakni 410,7 Mbps, 550,2 Mbps, dan 433,5 Mbps untuk node pertama, node kedua dan node ketiga. Algoritma CUBIC memiliki nilai yang cukup berbeda antara node-nya.

Kemudian hasil pengujian nilai stabilitas pada pengujian kali ini dapat ditunjukan pada table 2. Berikut ini.

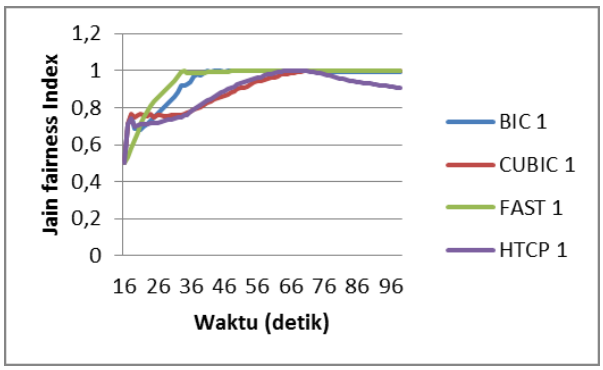
Tabel 2. Nilai stabilitas algoritma pengujian

Algoritma	Node 1 S_x	Node 2 S_x	Node 3 S_x	$\overline{S_x}$
BIC	133.6715	134.6867	170.1265	146.1616
CUBIC	111.6394	112.1566	165.7361	129.844
FAST	129.9823	130.5624	100.3157	120.2868
HTCP	175.3104	157.3839	224.3752	185.6898

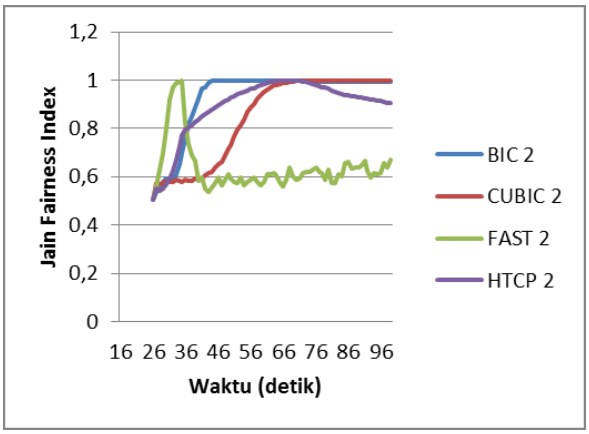
Dari hasil pengujian ini nilai stabilitas yang paling kecil ditunjukan oleh algoritma FAST dengan memiliki nilai 120,2868, akan tetapi pada rata-rata throughput-nya jauh lebih kecil. Oleh karena itu dapat dikatakan

algoritma CUBIC lebih baik untuk nilai stabilitasnya yang memiliki nilai 129,844.

Pada topologi ini terdapat dua nilai indeks Jain yang digunakan untuk menghitung nilai fairness pada node pertama dan node kedua, serta node pertama dan node ketiga. Pada Gambar 3. nilai fairness antara node pertama dan node kedua. Dari gambar juga dapat dilihat algoritma FAST memiliki nilai yang paling tinggi. Nilai fairness dari algoritma FAST ini mendekati nilai 1 yang merupakan nilai tertinggi dalam menghitung nilai fairness.



Gambar 3. Nilai fairness 1 topologi simple network



Gambar 4. Nilai fairness 2 topologi simple network

Dari Gambar 3 dan Gambar 4 dapat dilihat bahwa algoritma BIC memiliki nilai yang paling baik yang dapat dilihat dengan nilai fairness yang mendekati 1 dan paling cepat mencapai nilai yang mendekati 1.
Hasil analisis kinerja algoritma pada pengujian topologi simple network dapat ditunjukan pada Tabel 3.

Tabel 3. Hasil perhitungan kinerja algoritma topologi simple network

Algoritma	y	r	z	u
BIC	0.107054	0.292323	0.064604	0.463982
CUBIC	0.202931	0.259688	0.126967	0.589586
FAST	0.326701	0.240574	0.205244	0.772519
HTCP	0.137327	0.37138	0.111944	0.62065

Berdasarkan Tabel 5. menunjukan bahwa algoritma BIC memiliki kinerja yang paling baik jika dibandingkan dengan algoritma yang lainnya yakni memiliki nilai 0,46. Kemudian algoritma yang terbaik kedua ialah algoritma CUBIC dengan nilai 0,59. Algoritma HTCP memiliki kinerja yang lebih baik jika dibandingkan dengan FAST, algoritma HTCP memiliki nilai kinerja 0,62. Algoritma FAST memiliki kinerja yang paling buruk pada topologi ini dengan memiliki nilai 0,77.

V. KESIMPULAN

Pada pengujian kali ini dapat dilihat bahwa algoritma yang paling baik ditunjukan oleh algoritma BIC dengan memiliki nilai kinerja keseluruhan sebesar 0,463982.

UCAPAN TERIMA KASIH

Terima kasih diberikan kepada Bapak Wahyu Dewanto dan Bapak Sujoko Sumaryono yang telah meluangkan waktu untuk membantu di dalam penelitian kali ini.

REFERENSI

[1] Arshad M.J., Mian M.S., Issues of Multihoming Implementation Using FAST TCP: Simulation Based Analysis, IJCSNS, 2008.
[2] Chiu D.M., Jain R., Analyis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks, Computer Networks and ISDN Sistems, 1989
[3] Even B., An Experimental Investigation of TCP Performance in High Bandwidth-Delay Product Path, Thesis, Hamilton Institute, Maynooth, Ireland, 2007.
[4] Ha S., Rhee I., CUBIC: A New TCP-Friendly High-Speed TCP Variant, International Workshop on Protocols for FAST and Long Distance Networks, 2005.
[5] Jin C., Wei D. X., Low S. H, FAST TCP: Motivation, Architecture, Algorithms, Performance, IEEE Infocom, Hongkong, 2004.
[6] Jin C., Wei D. X., Low S. H, Bunn J., C. H. D., Doyle J. C., Newman H., Ravot S., Sigh S., Fast TCP: From Theory to Experiments, IEEE Network, 2005.
[7] Leith D., Shorten R., "H-TCP Protocol for High-Speed Long Distance Networks", PFLDnet, 2004.
[8] Leith D., Shorten R.N., McCullagh G., “Experimental Evaluation of CUBIC-TCP, PFLDnet, 2007.
[9] Rhee I., Xu L., CUBIC: A New TCP-Friendly High-Speed TCP Variant, North Carolina State University, University of Nebraska-Licoln, USA: 2004.
[10] Tanenbaum A. S., Computer Network, Fourth Edition, Pearson Education International, Upper Saddle River, New Jersey, USA: Prentice Hall, 2003.
[11] Xu L., Khaled H., Rhee I., Binary Increase Congestion Control for Fast, Long Distance Networks, Paper, North Carolina state University, Releigh, NC, USA, 2003.
[12] Wang G., Xia Y., Harisson D., An NS2 TCP Evaluation Tool, Internet Engineering Task Force, USA, 2007.

[13] Wei D.X., Cao P., NS-2 TCP-Linux: An NS-2 TCP Implementation with Congestion Control Algorithm from Linux, ACM, 2006.
[14] Wei D.X., Jin C., Low S.H., Hegde S., FAST TCP: Motivation, Architecture, Algorithms, Performance, IEEE/ACM Transactions on Networking, 2006.
[15] Zegura W. E., Calvert L. K., Danahoo J. M., A Quantitative Comparison of Graph-based Models for Internet Topology, GT-ITM, 1997.