View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by UPN (Universitas Pembangunan Nasional) Veteran Yogyakarta: Portal Journals

PERBANDINGAN KINERJA ALGORITMA BIC, CUBIC DAN HTCP PADA TOPOLOGI PARKINGLOT DAN MULTIHOME MENGGUNAKAN NS2

Rian Fahrizal[1)], **Wahyu Dewanto**[2)], **Sujoko Sumaryono**[3)]
[1] *Jurusan Teknik Elektro FT UNTIRTA Jln. Jend. Sudirman Km 2 Cilegon Indonesia*
[2,3] *Jurusan Teknik Elektro FT UGM Jln. Grafika 2 Yogyakarta 55281 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, dan HTCP. Algoritma-algoritma ini perlu diuji untuk mengetahui kinerjanya jika diterapkan pada jaringan dengan dua topologi yakni parkinglot dan multihome. Hasil pengujian didapatkan algoritma yang paling baik adalah BIC dengan memiliki nilai kinerja yang paling kecil pada pengujian kali ini.

***Kata Kunci** : Parkinglot, Muiltihome, BIC, CUBIC, HTC, Nilai Kinerja.*

1. PENDAHULUAN

TCP/IP merupakan protokol yang digunakan secara luas di dalam jaringan Internet. Protokol ini memiliki beberapa lapisan yakni application layer, transport layer, network layer dan physical layer. Setiap layer ini memiliki fungsi yang berbeda antara satu dengan yang lainnya. Lapisan yang bertanggung jawab di dalam melakukan pengiriman data yang juga merupakan lapisan yang penting di dalam TCP/IP ini ialah transport layer. Ada dua motode yang digunakan di dalam layer ini yakni TCP dan UDP. Sebagian besar pengiriman data di dalam jaringan komputer menggunakan metode TCP.

Di dalam pengiriman TCP yang ada digunakan algoritma adaptive increase multiplicity decrease (AIMD). Di dalam algoritma ini penambahan jumlah data yang dikirim dalam satu kali pengiriman (window) jika data berhasil dikirim dan tidak mengalami kongesti. Apabila terjadi kongesti atau terjadi error maka pengiriman window berkurang setengahnya. Algoritma ini telah digunakan sejak tahun 1980-an hingga sekarang. Seiring perkembangan waktu jaringan Internet membutuhkan bandwidth yang jauh lebih besar. Algoritma AIMD ini mengalami masalah pada jaringan komputer dengan bandwidth lebih besar dari 1 Gbps. Hasil penelitian menunjukan dibutuhkan waktu yang lama untuk mencapai nilai badwidth 1 Gbps jika propagation delay pada jaringan tersebut memiliki nilai 100 ms.

Untuk menangani masalah di dalam algoritma AIMD dibuat algoritma alternatif yang sudah diterapkan di dalam Linux kernel terbaru yakni BIC, CUBIC dan HTCP. Ketiga algoritma ini digunakan untuk menangani masalah pada jaringan Internet dengan waktu delay yang besar. Akan tetapi masih diperlukan pengujian yang baik yang dapat membuktikan bahwa ketiga algoritma ini merupakan algoritma yang terbaik yang dapat diterapkan di dalam jaringan komputer. Pengujian yang dilakukan untuk ketiga algoritma ini dilakukan dengan menggunakan topologi yang mencoba mensimulasikan jaringan Internet yang ada dengan menggunakan topologi parking lot dan multihome.

Parameter kinerja yang digunakan di dalam pengujian ini ialah menggunakan rata-rata throughput, stabilitas dan fairness (Raj Jain index). Kemudian dari nilai-nilai parameter yang telah digunakan tersebut dilakukan perhitungan yang dapat menunjukan bahwa parameter tersebut bisa dibandingkan secara keseluruhan. Perhitungan tambahan ini menunjukan kinerja dari algoritma yang dibandingkan.

2. TINJAUAN PUSTAKA

2.1 BIC-TCP

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* S_{min}). 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* S_{max}), ditambahkan nilai *window* dengan S_{max} sampai jarak menjadi kurang dari S_{max}, 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(S_{min}) 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_win_i dan min_win_i 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 *convergence*.

BIC-TCP menggunakan bentuk algoritma pencarian biner untuk memperbaharui *cwnd*. Nilai dari w_1 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 S_{max} dari w_1, dan memperbaharui *cwnd* lebih lambat ketika nilainya mendekati w_1. Perkalian *backoff* dari *cwnd* digunakan pada pendeteksian paket yang hilang, dengan asumsi faktor *backoff* β sebesar 0,8. Secara rinci,

$$Ack: \begin{cases} \delta = (w_1 - cwnd)/B \\ cwnd \leftarrow cwnd + \frac{f_\alpha(\delta, cwnd)}{cwnd} \end{cases} \quad (1)$$

$$Loss: \begin{cases} w_1 = \begin{cases} \frac{1+\beta}{1} x cwnd & cwnd < w_1 \\ cwnd & selainnya \end{cases} \\ w_2 = cwnd \\ cwnd \leftarrow \beta \; x \; cwnd \end{cases} \quad (2)$$

Dengan

$$f_x(\delta, cwnd)\begin{cases}\frac{\beta}{\sigma} & (\delta \leq 1, cwnd < w_1 \\ & atau\ (w_1 \leq cwnd < w_1 + B) \\ \delta & 1 < \delta \leq S_{max},\ cwnd < w_1 \\ \frac{w_1}{\beta-1} & B \leq cwnd - w_1 < S_{max}(B-1) \\ S_{max} & selainnya\end{cases} \quad (3)$$

dengan
σ = *distance*
cwnd = *congestion window(bytes)*
f(δ ,cwnd) = *fungsi penambahan window size*
δ = *pengurangan*
w_1 = *window(bytes)*
w_2 = *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.*

2.2 CUBIC

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 W_{maz} 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 recovery dan retransmit dari TCP. Kemudian setelah masuk ke algoritma congestion avoidance dari recovery, mulai ditambahkan *window* menggunakan bentuk concave pada fungsi cubic. Fungsi cubic ditentukan untuk memiliki nilai baru pada W_{max} sehingga penambahan berlanjut sampai ukuran *window* menjadi W_{max}. 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 W_{max} dengan utilisasi jaringan tertinggi dan di dalam kondisi steady, sebagian besar ukuran *window* CUBIC mendekati W_{max}, 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:

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.*

Faktor *backodd* 0,8. Pada paket yang hilang, cwnd dikurangi dengan faktor 0,8.

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.

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

$$target = W_{max} + C\left(t - \sqrt[3]{(\beta(W_{max} - 0{,}8W))}\right)^3 \quad (4)$$

dengan
W_{max} = *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 W_{max} 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.
Adaptasi fungsi cubic. Nilai dari W_{max} ditentukan tergantung dari apakah *backoff* terakhir muncul sebelum atau sesudah cwnd tercapai pada nilai W_{max} sebelumnya. Atau jika tidak W_{max} diset sama dengan 0,9W.
Algoritma ini juga mencakup kode untuk meyakinkan bahwa algoritma ini paling tidak seagresif seperti TCP yang ada.

2.3 HTCP

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

$$ACK: \quad cwnd \leftarrow cwnd + \frac{2(1-\beta)f_\alpha(\Delta)}{cwnd} \tag{5}$$

$$Loss: \quad cwnd \leftarrow g_\beta(B)\, x\, cwnd \tag{6}$$

Dengan

$$f_\alpha(\Delta) = \begin{cases} 1 & \Delta \leq \Delta_L \\ \max\left(\bar{f}_\alpha(\Delta) T_{min}, 1\right) & \Delta > \Delta_L \end{cases} \tag{7}$$

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

dengan

cwnd = *congestion window(bytes)*
β = *faktor backoff*
Δ = *delta*
$f_\alpha(\Delta)$ = *fungsi penambahan congestion window*
$g_\beta(B)$ = *fungsi pengurangan congestion window*
T_{min} = *waktu minimum(*detik*)*
T_{max} = *waktu maksimum(*detik*)*

dimana Δ_L ditentukan batas sehingga algoritma update TCP standar digunakan dimana $\Delta \leq \Delta_L$. Fungsi penambahan kuadrat untuk f_α ialah

$$\bar{f}_\alpha(\Delta) = 1 + 10(\Delta - \Delta_L) + 0{,}25(\Delta - \Delta_L)^2 \tag{9}$$

dengan

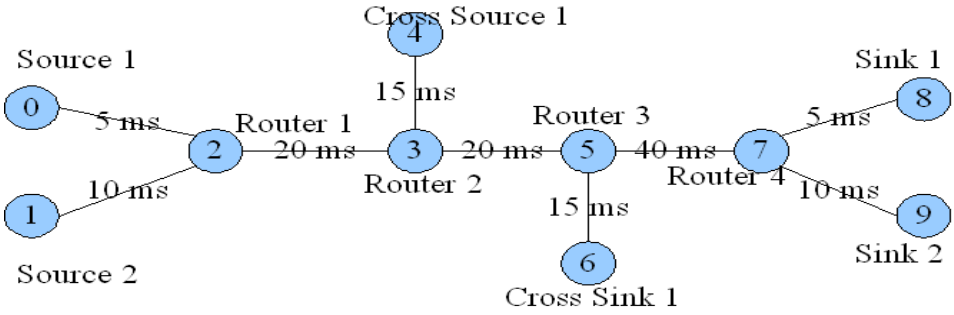
$f_\alpha(\Delta)$ = *fungsi penambahan dalam rentang waktu*
Δ_L = *Delta limit*

T_{min} T_{max} menghitung waktu round-trip yang telah ada. β(k + 1) adalah perhitungan *throughput* yang dicapai maksimum pada masa congesti terakhir.

3. METODE PENELITIAN

3.1 Topologi *Parkinglot*

Topologi *parkinglot* terlihat dari Gambar 3.5. sama dengan topologi *dumbbell* kecuali menampilkan trafik yang melintasi antara dua *router*

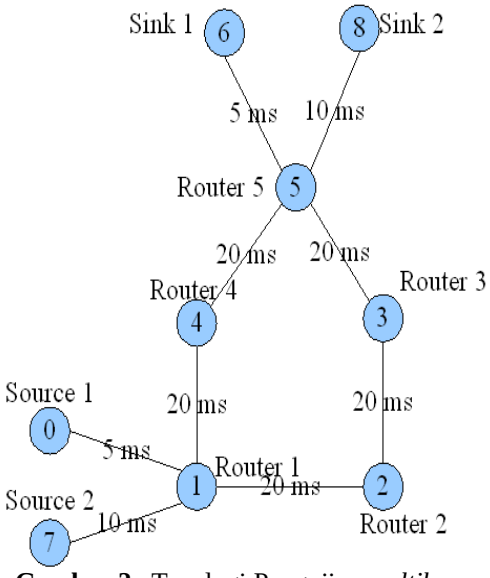


Gambar 1. Topologi *parkinglot* (Yilmaz 2002)(Wang 2007)

Pada topologi ini *node* yang melakukan pengiriman data yakni *node* 0, 1, dan 4, sedangkan *node* penerima yakni *node* 8,9 dan 6. Pada awalnya *node* 0 melakukan pengiriman selanjutnya *node* 1 melakukan pengiriman dan terakhir *node* 4 melakukan pengiriman. Pengiriman pada *node* 4 ini memiliki waktu tunggu yang paling kecil sedangkan pengiriman *node* 0 dan1 memiliki waktu tunggu pengiriman yang jauh lebih besar, dan juga memiliki jalur yang lebih panjang. *Node* 2, 3, 5, dan 7 berfungsi sebagai *node router* yang mengalirkan data dari *node* pengirim ke *node* penerima.
Program simulasi menggunakan algoritma , H-TCP, BIC dan CUBIC pada metode pengiriman data. *Node-node* yang digunakan dapat dilihat pada Gambar 1.

3.2 Topologi *multihome*

Dengan melakukan modifikasi topologi *multihome* (Arshad 2008) dan digabungkan dengan Gambar 2.13. didapatkanlah topologi pada Gambar 2. yang mensimulasikan perubahan jalur pengiriman data dan sekaligus simulasi jalur yang putus. Topologi ini digunakan untuk menguji algoritma-algoritma melewati jaringan komputer yang melewati jalur yang putus kemudian berpindah dari jalur pertama yakni melewati *node* 4, menjadi melewati *node* 2. Pada topologi ini *node* 0 dan *node* 7 sebagai *node* pengirim data sedangkan *node* 6 dan *node* 8 menjadi *node* penerima data. Aplikasi yang digunakan ialah FTP. Hasil keluaran dalam bentuk nilai *throughput*. Simulasi ini dimulai dari *node* 1 melakukan pengiriman data ke *node* 6 pada detik ke-5, kemudian pada detik ke-20 *node*-7 melakukan pengiriman data. Untuk melakukan perubahan jalur maka jalur antara *node* 1 ke *node* 4 diputus pada detik ke-50. Pada detik ke-100 simulasi ini selesai.



Gambar 2 : Topologi Pengujian *multihome*

Dengan melakukan modifikasi topologi *multihome* (Arshad 2008) dan digabungkan dengan Gambar 2.13. didapatkanlah topologi pada Gambar 2. yang mensimulasikan perubahan jalur pengiriman data dan sekaligus simulasi jalur yang putus. Topologi ini digunakan untuk menguji algoritma-algoritma melewati jaringan komputer yang melewati jalur yang putus kemudian berpindah dari jalur pertama yakni melewati *node* 4, menjadi melewati *node* 2. Pada topologi ini *node* 0 dan *node* 7 sebagai *node* pengirim data sedangkan *node* 6 dan *node* 8 menjadi *node* penerima data. Aplikasi yang digunakan ialah FTP. Hasil keluaran dalam bentuk nilai *throughput*. Simulasi ini dimulai dari *node* 1 melakukan pengiriman data ke *node* 6 pada detik ke-5, kemudian

pada detik ke-20 *node*-7 melakukan pengiriman data. Untuk melakukan perubahan jalur maka jalur antara *node* 1 ke *node* 4 diputus pada detik ke-50. Pada detik ke-100 simulasi ini selesai.
Pada *node* ini *node* 1, 2, 3, 4, dan 5 berfungsi sebagai *router* yang bertugas untuk melewatkan data dari *node* sumber ke *node* tujuan. Sedangkan *node* 0 dan *node* 7 berfungsi sebagai komputer yang melakukan pengiriman data, dimana *node* 6 dan *node* 8 berfungsi sebagai komputer yang menerima data. Semua jalur yang digunakan menggunakan *bandwidth* 1 Gbps.
Program simulasi menggunakan algoritma , H-TCP, BIC dan CUBIC pada metode pengiriman data. Dua aliran data mengalir dari *node* 0 ke *node* 6 dan pada *node* 7 ke *node* 8 pada Gambar 3.2. Pada detik ke 5 data mengalir dari *node* 0 ke *node* 6 dan pada detik ke 25 data mengalir dari *node* 7 ke *node* 8, dan pada detik ke 50 link antara *node* 1 dan *node* 4 terputus, sehingga aliran data berubah menjadi melalui link *node* 1 dan *node* 2.

4. HASIL DAN PEMBAHASAN

4.1. Analisis Topologi Parkinglot

Pada topologi ini semua algoritma berhasil melakukan pengiriman data. Algoritma merupakan algoritma yang paling baik di dalam topologi dengan nilai *throughput* yang sama antara ketiga *node*-nya di akhir simulasi. Algoritma BIC memiliki nilai *throughput* yang mendekati sama pada akhir simulasi walaupun nilai tersebut berosilasi dengan besar. Sedangkan pada algoritma CUBIC dan HTCP nilai akhir *throughput* tidak sama antara ketiga *node*-nya walaupun sampai akhir simulasi walaupun nilai ketiga *node*-nya saling mendekati. Algoritma CUBIC memiliki nilai *throughput* yang selalu berosilasi dengan sangat besar pada simulasi kali ini.
Analisis kinerja algoritma pada pengujian topologi *parkinglot* dapat ditunjukan pada Tabel 1.

Tabel 1. Hasil Simulasi topologi *parkinglot*

Algoritma	Y	r	z	u
BIC	0.161919	0.270972	0.081519	0.514409
CUBIC	0.44485	0.272659	0.208175	0.925684
HTCP	0.336323	0.153779	0.133771	0.623873

Berdasarkan Tabel 1. nilai kinerja yang paling baik ditunjukan oleh algoritma dengan memiliki nilai 0,39 sedangkan untuk algoritma BIC memiliki kinerja yang lebih baik jika dibandingkan dengan algoritma HTCP dan CUBIC dengan nilai kinerja 0,5. Algoritma HTCP lebih baik dari algoritma CUBIC dengan nilai 0,6. Algoritma CUBIC memiliki nilai kinerja yang paling buruk jika dibandingkan dengan algoritma yang lainnya dengan nilai 0,9.

4.2 Analisis Hasil Topologi Multihome

Pada simulasi dengan menggunakan topologi ini terdapat link yang terputus pada detik ke-50 yang menyebabkan rute pengiriman berubah. Dari hasil simulasi ini masing-masing algoritma berbeda di dalam pengiriman data ke tujuannya.
Hasil perhitungan kinerja algoritma pada topologi *multihome* ditunjukan pada Tabel 1. berikut ini.

Tabel 2. Hasil perhitungan kinerja algoritma topologi *multihome*

Algoritma	y	R	z	u
BIC	0.12762	0.26826	0.078063	0.473943
CUBIC	0.545278	0.11354	0.2385	0.897319
HTCP	0.235342	0.174353	0.087803	0.497497

Seperti yang terlihat pada Tabel 2. dapat dilihat bahwa kinerja algoritma BIC memiliki kinerja yang paling baik yang ditunjukan dengan nilai 0,4, sedangkan algoritma HTCP memiliki kinerja yang lebih baik dari algoritma CUBIC dan dengan nilai 0,5. Algoritma yang memiliki kinerja yang paling buruk adalah dengan nilai kinerja 1,1. Kemudian algoritma CUBIC memiliki nilai kinerja 0,89.
Dari hasil simulasi ini menunjukan bahwa algoritma BIC merupakan algoritma yang terbaik dengan nilai kinerja yang paling kecil. Hal ini menunjukan bahwa algoritma BIC memiliki tingkat kehandalan di dalam jaringan computer yang lebih baik jika dibandingkan dengan algoritma HTCP dan CUBIC.

5. KESIMPULAN

Pengujian dari kedua topologi untuk algoritma BIC, CUBIC dan HTCP menunjukan bahwa algoritma yang paling baik untuk pengujian *Multihome* dan *Parkinglot* ialah algoritma BIC, sedangkan untuk algoritma HTCP cukup baik di dalam simulasi ini walaupun tidak lebih baik dari algoritma BIC.

DAFTAR PUSTAKA

Arshad M.J., Mian M.S., 2008, *Issues of Multihoming Implementation Using TCP: Simulation Based Analysis*, IJCSNS.

Chiu D.M., Jain R. , 1989, *Analyis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks*, Computer Networks and ISDN Sistems

Even B. , 2007, *An Experimental Investigation of TCP Performance in High Bandwidth-Delay Product Path*, Thesis, Hamilton Institute, Maynooth, Ireland.

Ha S., Rhee I. , 2005, *CUBIC: A New TCP-Friendly High-Speed TCP Variant*, International Workshop on Protocols for and Long Distance Networks.

Leith D., Shorten R. , 2004, *H-TCP Protocol for High-Speed Long Distance Networks*, PFLDnet.

Leith D., Shorten R.N., McCullagh G. , 2007, *Experimental Evaluation of CUBIC-TCP*, PFLDnet.

Rhee I., Xu L., 2004, *CUBIC: A New TCP-Friendly High-Speed TCP Variant*, North Carolina State University, University of Nebraska-Licoln, USA.

Tanenbaum A. S., 2003, *Computer Network, Fourth Edition*, Pearson Education International, Upper Saddle River, New Jersey, USA: Prentice Hall.

Xu L., Khaled H., Rhee I. , 2003, *Binary Increase Congestion Control for Long Distance Networks*, Paper, North Carolina state University, Releigh, NC, USA.

Wang G., Xia Y., Harisson D., 2007, *An NS2 TCP Evaluation Tool*, Internet Engineering Task Force, USA.

Wei D.X., Cao P., 2006, NS-2 TCP-Linux: An NS-2 TCP Implementation with Congestion Control Algorithm from Linux, ACM.

Wei D.X., Jin C., Low S.H., Hegde S., 2006, TCP: Motivation, Architecture, Algorithms, Performance, IEEE/ACM Transactions on Networking.