EVALUASI KINERJA *AGGREGATE FLOW CONTROL* TERHADAP SISTEM *DIFFERENTIATED SERVICES*

PERFORMANCE EVALUATION OF AGGREGATE FLOW CONTROL ON DIFFERENTIATED SERVICES SYSTEM

Benny Winata, Veronica Windha

Fakultas Teknik Jurusan Teknik Elektro
Universitas Katolik Indonesia Atma Jaya - Jakarta

Abstrak

Differentiated Services (*Diffserv*) menyediakan cara yang sederhana untuk menangani *Quality of Service* (Qos) di dalam jaringan *Internet Protocol* (IP) dengan mekanisme klasifikasi paket dan pembedaan perlakuan, serta alokasi sumber daya jaringan untuk trafik-trafik tertentu. Masalah yang sering terjadi di *Diffserv* adalah *unfairness* dan kongesti. Untuk mengatasi masalah tersebut, maka diperlukan suatu mekanisme yang dapat mengatur aliran paket data, salah satunya adalah *Aggregate Flow Control* (AFC). Penggunaan AFC pada sistem tidak hanya mengendalikan kongesti, tetapi juga meningkatkan *fairness* dalam *throughput* sehingga dalam penelitian ini, kinerja *Diffserv* tanpa dan dengan AFC, yaitu *troughput* dan *packet loss* dievaluasi menggunakan Network Simulator 2 (NS-2). Skenario simulasi yang digunakan dalam penelitian adalah simulasi variasi jumlah *microflows*, variasi ukuran paket, variasi RTT, dan variasi kelas AF. Hasil simulasi menunjukkan bahwa *Diffserv* dengan AFC dapat meningkatkan *fairness* dalam *throughput* dan menekan *packet loss* yang terjadi di dalam jaringan. Namun, untuk simulasi variasi kelas AF, *Diffserv* dengan AFC tidak dapat menciptakan *fairness* dalam *throughput* tetapi dapat menekan *packet loss* yang terjadi di dalam jaringan.

Kata kunci: *aggregate flow control, differentiated service, fairness, network simulator, quality of service*, AF PHB.

Abstract

Differentiated Services (Diffserv) provides a simple way for addressing Quality of Service (QoS) in the network Internet Protocol (IP) with the packet classification mechanism and treatment differentiation as well as network resources allocation for certain traffics. A problem frequently occurs in Diffserv is unfairness and congestion. To overcome these problems, a mechanism is required to regulate the flow of data packets, one of which is Aggregate Flow Control (AFC). The use of AFC in the system does not only control congestion but also improve the fairness in throughput. This study therefore evaluates the performance of Diffserv with and without AFC, namely throughput and packet loss, using the Network Simulator 2 (NS-2). Simulation scenario used in this study were simulations of varied microflows amount, packet size, RTT, and AF class. Simulation result showed that Diffserv with AFC could increase fairness in throughput and decrease packet loss in network. In the case of AF class variation, Diffserv with AFC could not increase fairness in throughput but decrease the packet loss that happened in the network.

Keywords: *aggregate flow control, differentiated service, fairness, network simulator, quality of service, AF PHB.*

Tanggal Terima Naskah : 05 Agustus 2015
Tanggal Persetujuan Naskah : 21 September 2015

1. PENDAHULUAN

Perkembangan laju informasi menyebabkan meningkatnya kebutuhan penggunaan Internet. Salah satu arsitektur yang digunakan untuk mencapai *Quality of Service* (QoS) dalam teknologi Internet adalah *Differentiated Services* (*Diffserv*), yang merupakan skema implementasi QoS untuk *Internet Protocol* (IP) yang menyediakan diferensiasi layanan dengan membagi trafik menjadi beberapa kelas.

Salah satu kendala dari *Diffserv* adalah sering terjadi *unfairness* dan kongesti di antara *aggregates* pada kelas trafik yang sama. *Unfairness* adalah penggunaan *bandwidth* yang tidak merata disebabkan karena perbedaan *resources*, sedangkan kongesti adalah kemacetan dalam pengiriman data. Untuk mengatasi kendala tersebut, diperlukan suatu mekanisme yang dapat mengatur aliran paket data dan salah satunya adalah *Aggregate Flow Control* (AFC) yang merupakan mekanisme kendali kongesti yang berada pada lapisan *Transmission Control Protocol* (TCP) [1]. Penggunaan AFC pada sistem tidak hanya mengendalikan kongesti, tetapi juga meningkatkan *fairness* dalam *throughput*. Penelitian ini membandingkan kinerja Diffserv tanpa AFC dan Diffserv dengan AFC dengan melihat indikator kinerja, yaitu *throughput* dan *packet loss* pada sistem [2].

2. KONSEP DASAR

2.1 *Quality of Service*

Quality of Services (QoS) merupakan standar nilai dari kualitas suatu layanan. Sebagai tolok ukur nilai performansi dikenal adanya parameter QoS yang meliputi:

a. *Throughput*, adalah kecepatan transfer data efektif.
b. *Packet Loss*, adalah ukuran banyak paket yang hilang akibat adanya *buffer overflow* pada saat terjadi kongesti.
c. Waktu tempuh satu paket adalah waktu yang dibutuhkan untuk mentransmisikan data dari sumber ke tujuan.

2.2 Antrian

Suatu trafik terkadang tidak dilayani secara langsung, tetapi sebagian trafik harus menunggu dalam suatu *buffer*. Antrian dapat dibedakan menjadi antrian tanpa priotitas dan antrian dengan prioritas. Antrian tanpa prioritas disebut juga dengan *First In First Out* (FIFO). Pada FIFO, paket- paket yang datang ke dalam suatu *buffer* akan disimpan sesuai urutan kedatangannya. Paket yang berada pada urutan paling awal pada *buffer* akan dilayani terlebih dahulu. Selain FIFO, antrian *Round Robin* juga termasuk antrian tanpa prioritas. *Round Robin* adalah jenis antrian di mana proses *forward* paket dilakukan secara siklis untuk setiap alamat fisik [3].

Antrian dengan prioritas melakukan pelayanan terhadap trafik dengan pertimbangan tertentu sesuai persyaratan QoS, misalnya trafik *Assured Forwarding* (AF) yang memiliki empat kelas trafik. Contoh antrian dengan prioritas adalah *Priority Queueing, Weighted Round Robin.*

Priority Queueing adalah jenis antrian yang melayani paket sesuai kelas layanannya, paket yang memiliki tingkat prioritas lebih tinggi akan dilayani terlebih dahulu. Pada *Weighted Round Robin* pelayanan paket tidak dilakukan satu per satu untuk tiap antrian fisik; *Weighted Round Robin* bisa mengatur jumlah paket yang dilayani oleh

masing-masing alamat fisik untuk setiap siklus. Antrian trafik yang terlalu panjang akan dapat menyebabkan kongesti. Untuk menangani kongesti, maka diperlukan manajemen kongesti.

2.3 Manajemen Kongesti

Kongesti dapat terjadi ketika jumlah permintaan layanan pada suatu *server* melebihi kapasitas *server*. Jaringan yang heterogen memiliki kemungkinan adanya kongesti yang besar akibat adanya ketidaksesuaian *resource* pada tiap *node* yang membentuknya, fenomena keterbatasan ini disebut dengan *bottleneck*. Terdapat dua cara untuk menangani kongesti, yaitu *open loop congestion control* dan *closed loop congestion control*.

Pada *open loop congestion control*, penanganan kongesti dilakukan tanpa adanya umpan balik dari penerima, tetapi dilakukan dengan mengalokasikan *resource* sesuai dengan prioritas pada tiap kelas layanan, contohnya *dropping policy Random Early Detection* (RED). Pada *closed loop congestion control*, penanganan kongesti dilakukan dengan memberikan *feedback* atau informasi kepada pihak atau tempat dimana bisa diambil tindakan [4].

2.4 *Transmission Control Protocol*

TCP bertanggung jawab terhadap *reliable*, *flow control*, dan *error correction*. Tugas TCP di sisi pengirim adalah melakukan segmentasi data yang berasal dari *layer* di atasnya, sedangkan di sisi penerima tugas TCP adalah melakukan *reassembly* paket-paket menjadi data dari *layer* di bawahnya. TCP harus dapat me-*recover* data yang rusak, hilang, terduplikasi, atau sampai di tujuan dengan urutan yang salah. Oleh karena itu, digunakan penomoran pada setiap paket yang dikirimkan dan membutuhkan *acknowledgement* (ACK) dari TCP penerima atau dengan kata lain, penerima mengirimkan pesan ACK ke pihak pengirim.

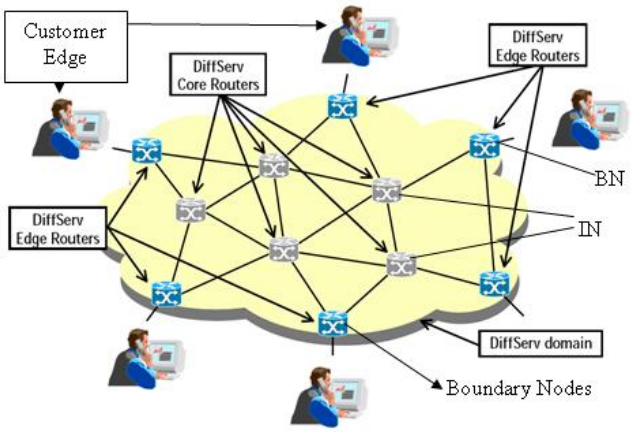
2.5 Model Layanan QoS

2.5.1 Model *Differentiated Services*

Model layanan *DiffServ* adalah skema implementasi QoS untuk IP yang menyediakan diferensiasi layanan, dengan membagi trafik menjadi beberapa kelas, dan memperlakukan setiap kelas secara berbeda [5]. Identifikasi kelas dilakukan dengan memasang semacam kode *Diffserv*, disebut *Diffserv code point* (DSCP), ke dalam paket IP. Berikut adalah bentuk arsitektur *Diffserv*.

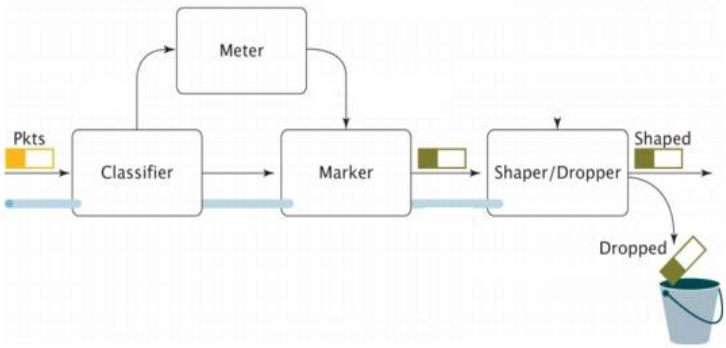
Secara garis besar, jaringan *DiffServ* dibagi atas:

a. Titik akhir, pada umumnya disebut dengan *Customer Edge* (CE).
b. Router yang berhubungan langsung dengan CE, yaitu *Boundary Nodes* (BN) yang disusun oleh *edge router*.
c. Jaringan *backbone* yang disebut *Inner Nodes* (IN) yang disusun oleh *core router*.



Gambar 1. Arsitekur *Diffserv*

Proses klasifikasi trafik terjadi pada BN sedangkan pada IN hanya meneruskan paket, sehingga pada *DiffServ*, kompleksitas layanan berada pada *edge router*.



Gambar 2. Proses klasifikasi pada *DiffServ*

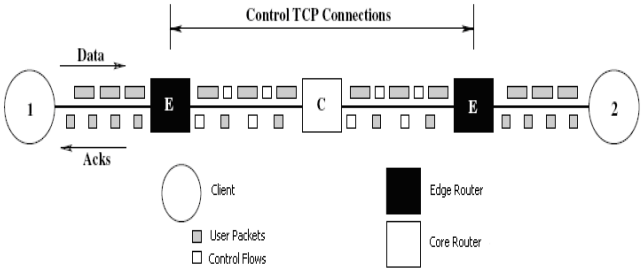
Keterangan blok diagram:

a. *Classifier* berfungsi untuk memilah paket berdasarkan *header* paket sehingga didapat informasi mengenai kelas trafik.
b. *Marker* berfungsi untuk melakukan pemberian kondisi DSCP bit.
c. *Meter* berfungsi untuk mengukur aliran trafik yang kemudian digunakan untuk menentukan trafik mana yang akan mendapat prioritas pelayanan lebih tinggi dan probabilitas *drop* yang lebih rendah.
d. *Shaper* berfungsi untuk memberikan *delay* pada trafik sesuai dengan *profile* trafik. Pada *shaper* terdapat *buffer* yang berfungsi sebagai ruang tunggu, dimana dapat dilakukan pembuangan paket jika penuh.
e. *Dropper* berfungsi untuk melakukan pembuangan paket pada saat terjadi kongesti. Hal ini dilakukan sesuai aturan RED.

2.5.2 Model *Differentiated Services* dengan *Aggregate Flow Control* (AFC)

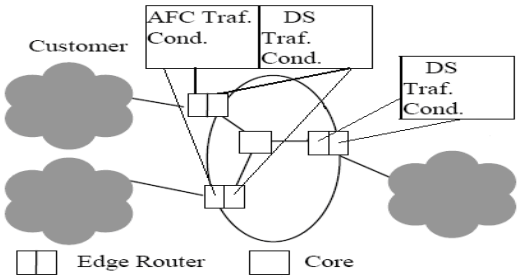
AFC adalah mekanisme kontrol *edge-to-edge* yang dikombinasikan dengan *Diffserv traffic conditioning* yang mengatur trafik *aggregate* pelanggan ke dalam *core network* untuk mengatasi kongesti di jaringan [6]. AFC bekerja dengan cara berikut:

a. *Control* TCP *Connections* terkait dengan setiap *aggregate* pelanggan dan terletak diantara dua *edge router*.
b. Pada *Control* TCP *Connections* dimasukkan *control flow* ke dalam jaringan untuk mendeteksi kongesti di sepanjang jalur *aggregate data* (baik *control flow* dan paket data harus mengikuti jalur yang sama antara *edges router*).



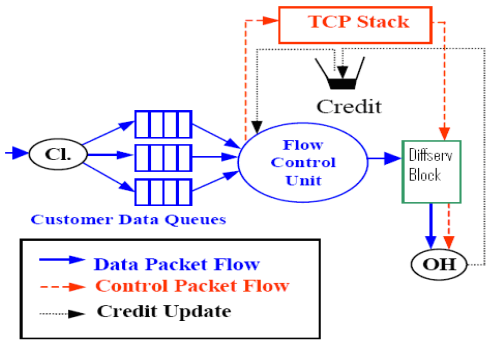
Gambar 3. Skema mekanisme kontrol *edge-to-edge*

Control flow adalah paket kontrol yang diberi *flow identifier* untuk tiap *aggregate*. Dalam *Diffserv* dengan AFC digunakan lebih dari satu *control flow* sehingga *flow identifier* untuk tiap *control flow* pada masing-masing *aggregate* akan berbeda. Mekanisme AFC merupakan mekanisme yang menggunakan *Diffserv trafic conditioning*, yaitu parameter RED dan konfigurasi AFC secara bersama di *edge router* yang berhubungan dengan sumber trafik, sedangkan di *edge router* yang berhubungan dengan tujuan trafik hanya menggunakan parameter RED [7]. Di *core router* pada *Diffserv* dengan AFC tidak terdapat perubahan atau penambahan mekanisme.



Gambar 4. *Diffserv* dengan AFC

Berikut adalah bentuk skema AFC.



Gambar 5. Skema AFC

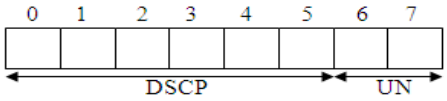
Proses klasifikasi trafik *aggregate* terjadi di *edge router*. Paket data masuk diterima oleh *Classifier* (CI), kemudian diklasifikasikan dan diantrikan sesuai dengan klasifikasinya, kemudian paket data akan diteruskan ke *Flow Control Unit* (FCU). Pada FCU digunakan *Additive Increase Multiplicative Decrease* (AIMD) untuk mengatur trafik data. Setiap terjadi kongesti, *congestion window* akan dikurangi setengahnya (*Multiplicative Decrease*). Ketika tidak terjadi kongesti, *congestion window* akan diperbesar secara bertahap (*Additive Increase*). FCU akan meneruskan paket data ketika *credit* tersedia di *bucket*. Kapasitas *bucket* akan diserap sesuai dengan ukuran paket yang

dikirim oleh FCU. AFC menggunakan *virtual maximum segment size* (vmss) untuk alur *flow control* TCP, vmss tersebut adalah jumlah data *user* (*bytes*) yang diperbolehkan untuk setiap *credit*. Untuk setiap nilai data vmss *user* (*bytes*) yang dikirim, *control flow* dihasilkan dan dikirim ke *TCP stack*. Semua paket data dan *control flow* akan diteruskan ke *Diffserv block* untuk *policing*, *marking*, dan *metering*. *Output handler* (OH) memantau semua paket keluar. Ketika OH mendeteksi *control flow*, OH akan menambah *credit* ke *bucket* sesuai vmss (*bytes*) untuk kemudian digunakan lagi pada paket data berikutnya. Pada Gambar 5 terlihat hanya satu TCP *credit bucket*, tetapi pada kenyataan setiap *aggregate* mempuyai TCP *credit bucket*-nya sendiri. Kehilangan *control flow* akan menyebabkan *congestion windows* akan berkurang sehingga akan menyebabkan berkurangnya tingkat transmisi *aggregate*. Untuk mengatasi hal tersebut, maka dalam AFC digunakan lebih dari satu *control flow* untuk setiap *aggregate.* Besarnya *credit* diinisialisasikan sebagai n+1 vmss, dimana n adalah jumlah *control flow* yang digunakan.

Trafik *aggregate* yang menyebabkan kongesti di *core router* akan mengalami *blocking* di *edge router* berdasarkan atas *dropping control flow* yang terjadi di *core router*. Apabila terjadi *dropping* dari *control flow* di *core router,* maka *core router* akan memberikan informasi berupa ACK (paket data dan *control flow*) tentang adanya kongesti ke *edge router*. Informasi tersebut akan diterima oleh OH dan kemudian OH akan mem-*blocking* paket data dan *control flow,* serta menghentikan penambahan *credit* ke *bucket,* sehingga FCU tidak dapat meneruskan paket data jika *credit* tidak tersedia di *bucket*. Hal ini menyebabkan *packet loss* yang terjadi lebih sedikit.

2.6 DSCP Bit

Pada *DiffServ* dilakukan penggantian TOS bit dengan DSCP bit.



Gambar 6. Konfigurasi DSCP bit

Kondisi DSCP bit, memberikan informasi mengenai prioritas trafik yang dibentuk oleh bit yang ditempatkan pada bit ke-0 hingga bit ke-2, sedangkan informasi mengenai kondisi *drop* ditempatkan pada bit ke-3 hingga bit ke-5. Bit ke-6 dan ke-7 belum digunakan (*unused*) hingga saat ini.

2.7 *Per Hop Behaviour* (PHB)

Pengklasifikasian paket yang datang menyebabkan adanya perbedaan *Behaviour Aggregat*, di mana untuk setiap agregat diset sebuah nilai DSCP [7]. Penjadwalan dilakukan berdasarkan nilai yang terdapat pada DSCP yang disebut sebagai *per hop behaviour* (PHB).

Sistem PHB yang sudah distandarisasi, diantaranya:

a. *Assured Forwarding* (AF)
   AF mendefinisikan empat kelas *forwarding* yang bersifat *independent* ketika melakukan pengiriman paket, yaitu AF1, AF2, AF3, AF4.

Tabel 1. Konfigurasi bit AF

Drop Precedence	AF1	AF2	AF3	AF4
Low Drop Precedence	AF11	AF21	AF31	AF41
Medium Drop Precedence	AF12	AF22	AF32	AF42
High Drop Precedence	AF13	AF23	AF33	AF43

b. *Expedited Forwarding* (EF)
   EF memiliki karakteristik *delay, jitter*, dan *loss probability* yang rendah; serta adanya jaminan *bandwidth* sehingga EF sesuai untuk aplikasi *real time*, seperti *video conference*.
c. *Best Effort* (BE)
   Kelas layanan ini memperlakukan trafik seperti apa adanya pada *DiffServ*. Ketika berada di *buffer*, semua trafik diperlakukan sama tanpa ada jaminan QoS apapun.

2.8 *Policer Time Slide Window Three Color Marking*

Policer Time Slide Window Three Color Marking (TSW3CM) merupakan *policer* yang berfungsi untuk memberi *color marking* (*green marking, yellow marking*, atau *red marking*) pada setiap paket yang datang, yang menandakan kondisi *drop precedence* pada trafik dengan AF PHB kemudian memberikan DSCP bit sesuai dengan kondisi paket yang masuk, memetakannya ke dalam antrian fisik dan *virtual*-nya. TSW3CM bekerja berdasarkan parameter *Commited Information Rate* (CIR), *Peak Information Rate* (PIR), dan rata-rata aliran trafik pada saat paket datang. *Policer* ini mengatur beberapa proses, yaitu:

a. *Classifying*
   Classifying adalah membedakan paket yang masuk berdasarkan tingkat prioritasnya.
b. *Measuring*
   Measuring adalah menentukan atau mengukur rata-rata aliran trafik.
c. *Marking*
   Marking adalah proses pemberian DSCP bit sesuai tingkat prioritas dan *drop precedence.*

2.9 *Random Early Detection*

Random Early Detection (RED) merupakan bentuk *congestion control* pada *DiffServ*. RED terletak pada *dropper* dan berfungsi untuk mencegah terjadinya *packet loss* secara *burst* dengan cara men-*drop* secara acak paket-paket yang datang sebelum *buffer* penuh [8]. RED akan bekerja dengan ketentuan sebagai berikut:

a. Jika rata-rata antrian di bawah nilai minimum *threshold*, maka paket akan diteruskan pada *buffer* untuk menunggu layanan.
b. Jika minimum *threshold*<rata-rata antrian *buffer*<maksimum *threshold*, maka paket akan mengalami *dropping* dengan probabilitas sebesar P_{drop} dan probabilitas paket diteruskan pada *buffer* untuk menunggu layanan sebesar 1-P_{drop}. Nilai *probabilitas dropping* ini dapat diketahui berdasarkan persamaan berikut:

$$P_{drop} = P_{max}\ Q_{avg} - Min_{th}\quad Max_{th} - Min_{th}\ \text{.....(1)}$$

   Keterangan:
   Q_{avg} = Rata-rata panjang antrian pada *buffer*
   Min_{th} = Batas minimum *threshold buffer* RED
   Max_{th} = Batas maksimum *threshold buffer* RED
   P_{max} = Probabilitas *drop* maksimum suatu paket untuk kondisi $Min_{th}<Q_{avg}<Max_{th}$
c. Jika rata-rata antrian mendekati *threshold,* maka probabilitas paket mengalami *drop* akan semakin besar.
d. Jika rata-rata antrian>maksimum *threshold*, maka paket akan langsung di *drop*.

Pada RED, rata-rata panjang antrian pada *buffer* (Q_{avg}) diukur dengan menggunakan *Low Pass Filter*. *Low Pass Filter* yang digunakan bersifat *Exponential Weighted Moving Average* (EWMA). Untuk setiap paket yang datang, nilai Q_{avg} ditentukan dengan persamaan berikut:

$$Q_{avg} = (1 - W_q)Q_{avg} + qW_q \quad \text{.....(2)}$$

Keterangan:
Q_{avg} = Rata-rata panjang antrian pada *buffer* dan bernilai awal 0
W_q = Konstanta waktu pada LPF.
q = Panjang antrian pada saat pengukuran.

Pengukuran rata-rata panjang antrian *buffer* dilakukan setiap kedatangan paket, sehingga nilai Q_{avg} pada saat pengukuran dipengaruhi oleh nilai Q_{avg} pada pengukuran sebelumnya.

3. PERANCANGAN SIMULASI

Simulasi model layanan *Diffserv* dengan dan tanpa AFC dilakukan dengan menggunakan *software Network Simulator* 2 versi 2.26 yang beroperasi pada sistem operasi *Windows* XP [9]. Setiap simulasi untuk *Diffserv* tanpa AFC dan *Diffserv* dengan AFC akan menggunakan konfigurasi RED di *edge router* seperti pada Tabel 2.

Tabel 2. Konfigurasi RED dan kapasitas *buffer* di *edge router*

AFclass	High			Med			Low			Kapasitas buffer
	Min	Max	Pd	Min	Max	Pd	Min	Max	Pd	
AF1	5	8	0.1	4	7	0.15	3	6	0.2	100
AF2	10	15	0.1	7	15	0.15	5	10	0.2	110
AF3	15	25	0.1	10	25	0.15	10	20	0.2	120

Parameter konfigurasi di *core router* untuk *Diffserv* dengan AFC dan *Diffserv* tanpa AFC dapat dilihat pada Tabel 3 sesuai dengan *thesis Aggregate Flow Control*: *Improving Assurances for Differentiated Services Network* [1,5].

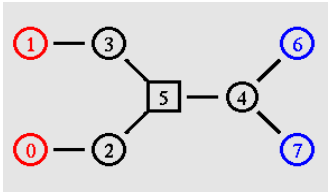
Tabel 3. Konfigurasi RED dan kapasitas *buffer* di *core router*

AFclass	High			Med			Low			Kapasitas buffer
	Min	Max	Pd	Min	Max	Pd	Min	Max	Pd	
AF1	50	80	0.02	40	70	0.06	30	60	0.12	100
AF2	100	150	0.02	70	150	0.06	50	100	0.12	180
AF3	200	240	0.02	160	200	0.06	50	160	0.12	250

3.1 *Differentiated Services* tanpa *Aggregate Flow Control* (AFC)

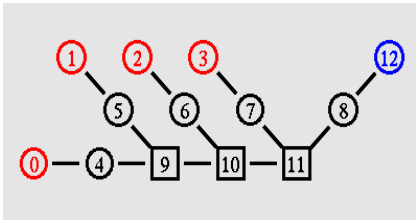
Simulasi untuk model layanan *Diffserv* tanpa AFC dilakukan dengan ketentuan sebagai berikut:

a. Topologi yang akan disimulasikan ada dua, yaitu topologi jaringan *single bottleneck* dan topologi jaringan *multiple bottleneck*, seperti yang terlihat pada Gambar 7 dan 8.



Gambar 7. Topologi jaringan *single bottleneck*

Keterangan untuk Gambar 7:
Node 0 = *source node* 1
Node 1 = *source node* 2
Node 2, 3, 4 = *edge router*
Node 5 = *core router*
Node 6 = *node* tujuan *source node* 1
Node 7 = *node* tujuan *source node* 2



Gambar 8. Topologi jaringan *multiple bottleneck*

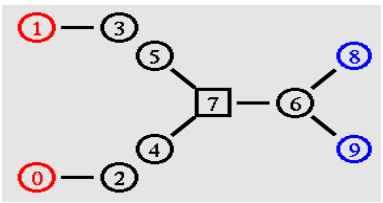
Keterangan untuk Gambar 8:
Node 0 = *source node* 1
Node 1 = *source node* 2
Node 2 = *source node* 3
Node 3 = *source node* 4
Node 4, 5, 6, 7, 8 = *edge router*
Node 9, 10, 11 = *core router*
Node 12 = *node* tujuan dari semua *node source*

b. Simulasi ini menggunakan protokol *transport* TCP.
c. Besarnya kapasitas *link* antara masing-masing sumber trafik dengan *edge router* adalah 5 Mbps.
d. Kapasitas *link* antara *edge router* dengan *core router* adalah sebesar 5 Mbps.
e. Kapasitas *link* antara *core router* dengan *edge router* adalah 5 Mbps.
f. Besarnya kapasitas *link* antara *edge router* dengan masing-masing tujuan trafik adalah 5 Mbps.

3.2 *Differentiated Service* dengan *Aggregate Flow Control* (AFC)

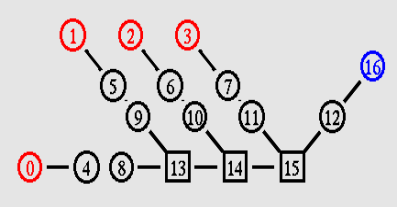
Simulasi untuk model layanan *Diffserv* dengan AFC dilakukan dengan ketentuan sebagai berikut:

a. Topologi yang akan disimulasikan ada dua, yaitu topologi jaringan *single bottleneck* dan topologi jaringan *multiple bottleneck*, seperti yang terlihat pada Gambar 9 dan 10.



Gambar 9. Topologi jaringan *single bottleneck*

Keterangan untuk Gambar 9:
Node 0 = *source node* 1
Node 1 = *source node* 2
Node 2, 3 = *node* AFC
Node 4, 5, 6 = *edge router*
Node 7 = *core router*
Node 8 = *node* tujuan *source node* 1
Node 9 = *node* tujuan *source node* 2



Gambar 10. Topologi jaringan *multiple bottleneck*

Keterangan untuk Gambar 10:
Node 0 = *source node* 1
Node 1 = *source node* 2
Node 2 = *source node* 3
Node 3 = *source node* 4
Node 4, 5, 6, 7 = *node* AFC
Node 8, 9, 10, 11, 12 = *edge router*
Node 13, 14, 15 = *core router*
Node 16 = *Node* tujuan dari semua *source node*

b. Simulasi ini menggunakan protokol *transport* TCP.
c. Besarnya kapasitas *link* antara masing-masing sumber trafik dengan *edge router* adalah 5 Mbps.
d. Kapasitas *link* antara *edge router* dengan *core router* adalah sebesar 5 Mbps.
e. Kapasitas *link* antara *core router* dengan *edge router* adalah 5 Mbps.
f. Besarnya kapasitas *link* antara *edge router* dengan masing-masing tujuan trafik adalah 5 Mbps.
g. Kapasitas *link* antara *node* AFC dengan *edge router* adalah 500 Mbps.
h. Waktu propagasi anatara *node* AFC dengan *edge router* adalah 0,1 ms.

3.3 SKENARIO SIMULASI

Simulasi untuk *Diffserv* tanpa AFC dan *Diffserv* dengan AFC terdiri atas empat skenario, yaitu:

a. Skenario simulasi variasi jumlah *microflows*
Tujuan dari skenario simulasi variasi jumlah *microflows* adalah untuk mengetahui kinerja *Diffserv* tanpa AFC dan *Diffserv* dengan AFC jika setiap *aggregate* memiliki jumlah *microflows* yang berbeda pada kelas trafik yang sama dengan melihat indikator kinerja, yaitu *throughput* dan *packet loss*.

Tabel 4. Parameter yang digunakan untuk simulasi variasi *microflows*

Agg	*microflows*	*Packetsize*	RTT	CIR	PIR	Kelas AF
Agg 1	10 TCP	1500 *bytes*	10 ms	0.3Mbps	1Mbps	AF1
Agg 2	25 TCP	1500 *bytes*	10 ms	0.3Mbps	1Mbps	AF1

b. Skenario simulasi variasi ukuran paket
Tujuan dari skenario simulasi variasi ukuran paket adalah untuk mengetahui kinerja *Diffserv* tanpa AFC dan *Diffserv* dengan AFC jika setiap *aggregate* memiliki ukuran paket yang berbeda pada kelas trafik yang sama dengan melihat indikator kinerja, yaitu *throughput* dan *packet loss*.

Tabel 5. Parameter yang digunakan untuk simulasi variasi ukuran paket

Agg	*microflows*	*Packetsize*	RTT	CIR	PIR	Kelas AF
Agg 1	20 TCP	500 *bytes*	10 ms	0.4Mbps	1Mbps	AF2
Agg 2	20 TCP	1500 *bytes*	10 ms	0.4Mbps	1Mbps	AF2

c. Skenario simulasi variasi besarnya *Round Trip Time* (RTT)
Tujuan dari skenario simulasi variasi besarnya RTT adalah untuk mengetahui kinerja *Diffserv* tanpa AFC dan *Diffserv* dengan AFC jika setiap *aggregate* memiliki nilai RTT yang berbeda pada kelas trafik yang sama dengan melihat indikator kinerja, yaitu *throughput* dan *packet loss*.

Tabel 6. Parameter yang digunakan untuk simulasi variasi RTT

Agg	*microflows*	*Packetsize*	RTT	CIR	PIR	Kelas AF
Agg 1	20 TCP	1500 *bytes*	10ms	0.5Mbps	1Mbps	AF3
Agg 2	20 TCP	1500 *bytes*	100ms	0.5Mbps	1Mbps	AF3
Agg 3	20 TCP	1500 *bytes*	200ms	0.5Mbps	1Mbps	AF3
Agg 4	20 TCP	1500 *bytes*	300ms	0.5Mbps	1Mbps	AF3

d. Skenario simulasi variasi kelas AF
Tujuan dari skenario simulasi variasi kelas AF adalah untuk mengetahui kinerja *Diffserv* tanpa AFC dan *Diffserv* dengan AFC jika setiap *aggregate* memiliki kelas trafik AF yang berbeda dengan melihat indikator kinerja, yaitu *throughput* dan *packet loss*.

Tabel 7. Parameter yang digunakan untuk simulasi variasi kelas AF

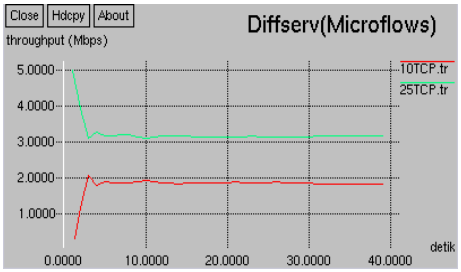
Agg	*microflows*	*Packetsize*	RTT	CIR	PIR	Kelas AF
Agg 1	10 TCP	1500 *bytes*	10 ms	0.3Mbps	1Mbps	AF1
Agg 2	10 TCP	1500 *bytes*	10 ms	0.4Mbps	1Mbps	AF2
Agg 3	10 TCP	1500 *bytes*	10 ms	0.4Mbps	1Mbps	AF2
Agg 4	10 TCP	1500 *bytes*	10 ms	0.5Mbps	1Mbps	AF3

4. HASIL SIMULASI

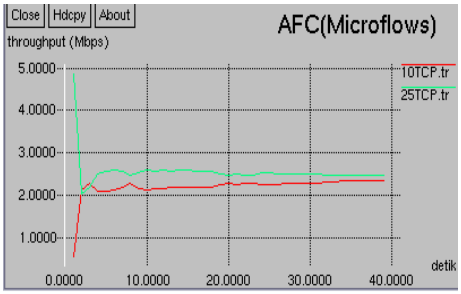
4.1 *Throughput*

4.1.1 *Throughput* untuk Simulasi Variasi Jumlah *Microflows*

Throughput untuk *Diffserv* tanpa dan dengan AFC untuk simulasi variasi jumlah *microflows* dapat dilihat pada Gambar 11 dan 12.



Gambar 11. *Diffserv* tanpa AFC

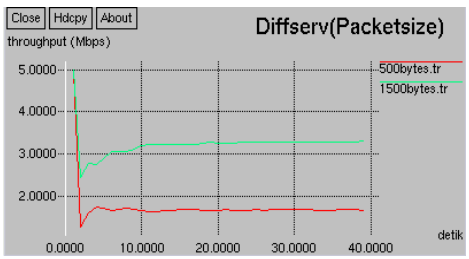


Gambar 12. *Diffserv* dengan AFC

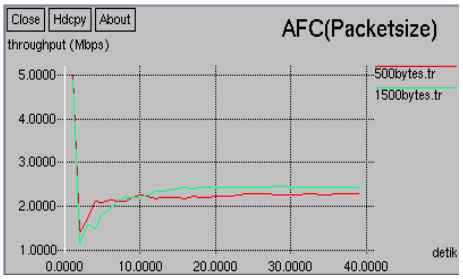
Hasil simulasi untuk *Diffserv* tanpa AFC dan *Diffserv* dengan AFC yang dilakukan pada kelas AF yang sama, yaitu AF1 menunjukkan bahwa pada *Diffserv* tanpa AFC, faktor jumlah *microflows* memengaruhi *throughput* yang diperoleh setiap *aggregate*, dimana *aggregate* yang mempunyai jumlah *microflows* yang besar maka *throughput* yang diperoleh juga besar. Pada *Diffserv* dengan AFC, faktor jumlah *microflows* tidak mempengaruhi *throughput* yang diperoleh setiap *aggregate*, dimana setiap *aggregate* memperoleh *throughput* yang hampir sama besarnya walaupun jumlah *microflows* yang dimiliki setiap *aggregate* berbeda jumlahnya. *Fairness* dalam *throughput* ini terjadi karena tiap *aggregate* menggunakan jumlah *control flow* yang sama tanpa memperhatikan jumlah *microflows* pada tiap *aggregate*.

4.1.2 *Throughput* untuk Simulasi Variasi Ukuran Paket

Throughput untuk *Diffserv* tanpa dan dengan AFC untuk simulasi variasi ukuran paket dapat dilihat pada Gambar 13 dan 14.



Gambar 13. *Diffserv* tanpa AFC

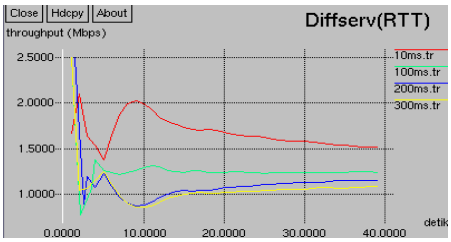


Gambar 14. *Diffserv* dengan AFC

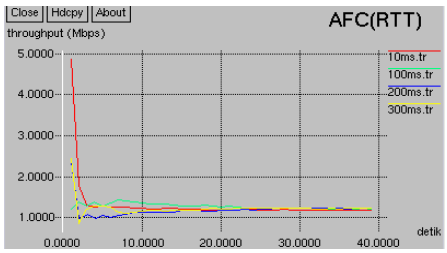
Hasil simulasi untuk *Diffserv* tanpa AFC dan *Diffserv* dengan AFC yang dilakukan pada kelas AF yang sama, yaitu AF2 menunjukkan bahwa pada *Diffserv* tanpa AFC, faktor ukuran paket memengaruhi *throughput* yang diperoleh setiap *aggregate*, dimana *aggregate* yang mempunyai ukuran paket yang besar maka *throughput* yang diperoleh juga besar. Pada *Diffserv* dengan AFC, faktor ukuran paket tidak mempengaruhi *throughput* yang diperoleh setiap *aggregate*, dimana setiap *aggregate* memperoleh *throughput* yang hampir sama besarnya walaupun ukuran paket yang dimiliki setiap *aggregate* berbeda besarnya. *Fairness* dalam *throughput* ini terjadi karena skema AFC secara efektif memberlakukan *congestion window* ke setiap *aggregate* yang tidak bergantung kepada ukuran paket.

4.1.3 *Throughput* untuk Simulasi Variasi *Round Trip Time* (RTT)

Throughput untuk *Diffserv* tanpa dan dengan AFC untuk simulasi variasi RTT dapat dilihat pada Gambar 15 dan 16.



Gambar 15. *Diffserv* tanpa AFC



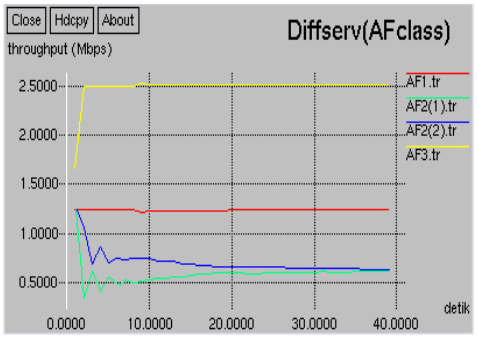
Gambar 16. *Diffserv* dengan AFC

Hasil simulasi untuk *Diffserv* tanpa AFC dan *Diffserv* dengan AFC yang dilakukan pada kelas AF yang sama, yaitu AF3 menunjukkan bahwa pada *Diffserv* tanpa AFC, faktor nilai RTT mempengaruhi *throughput* yang diperoleh setiap *aggregate*, dimana *aggregate* yang mempunyai nilai RTT yang kecil maka *throughput* yang diperoleh besar. Pada *Diffserv* dengan AFC, faktor nilai RTT tidak memengaruhi *throughput* yang diperoleh setiap *aggregate*, dimana setiap *aggregate* memperoleh *throughput* yang hampir sama besarnya walaupun nilai RTT yang dimiliki setiap *aggregate* berbeda besarnya. Trafik *aggregate* yang mempunyai nilai RTT paling kecil

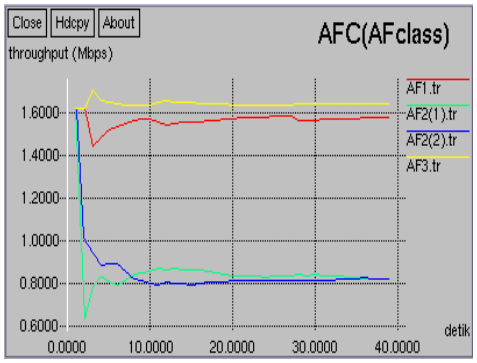
akan mempunyai kesempatan besar untuk membuat kongesti, tetapi masalah ini dapat diatasi oleh skema AFC dengan mem-*blocking* trafik *aggregate* tersebut di *egde router*, sehingga hasil *throughput* untuk tiap *aggregate* dapat mencapai *fair sharing*.

4.1.4 *Throughput* untuk Simulasi Variasi Kelas AF

Throughput untuk *Diffserv* tanpa dan dengan AFC untuk simulasi variasi kelas AF dapat dilihat pada Gambar 17 dan 18.



Gambar 17. *Diffserv* tanpa AFC



Gambar 1.8 *Diffserv* dengan AFC

Hasil simulasi untuk *Diffserv* tanpa AFC dan *Diffserv* dengan AFC yang dilakukan pada kelas AF yang bervariasi menunjukkan bahwa pada *Diffserv* tanpa AFC, *aggregate* yang mempunyai kelas trafik dengan prioritas tinggi akan memperoleh *throughput* yang lebih besar. Pada *Diffserv* dengan AFC, *throughput* yang didapat setiap *aggregate* tidak mencapai *fairness* karena perbedaan kelas trafik. Di sini terlihat bahwa skema AFC tidak dapat menciptakan *fairness* dalam *throughput* untuk tiap *aggregate* yang mempunyai kelas trafik yang berbeda-beda.

4.2 *Packet Loss*

4.2.1 *Packet Loss* untuk Simulasi Variasi Jumlah *Microflows*

Hasil *packet loss* untuk simulasi variasi jumlah *microflows* dapat dilihat pada Tabel 8. Total *packet loss* pada *Diffserv* dengan AFC untuk semua *aggregate* di *core router* mengalami penurunan sekitar ±57,6% (974 paket) dari *Diffserv* tanpa AFC, sedangkan total *packet loss* di jaringan mengalami penurunan sekitar ±61,4 % (1.139 paket) dari *Diffserv* tanpa AFC. *Diffserv* dengan AFC mem-*blocking* trafik *aggregate* yang menyebabkan kongesti di *core router* sehingga kongesti tidak terjadi dan *packet loss* yang terjadi juga sedikit.

Tabel 8. *Packet loss* untuk simulasi variasi jumlah *microflows*

Simulasi	Skema	Jumlah *microflows*	*Packet loss*	
			Core router	Jaringan
Microflows	Diffserv tanpa AFC	10 TCP	542	542
		25 TCP	1148	1313
Total			1690	1855
Microflows	Diffserv dengan AFC	10 TCP	375	375
		25 TCP	341	341
Total			716	716

4.2.2 *Packet Loss* untuk simulasi variasi ukuran paket

Hasil *packet loss* untuk simulasi variasi ukuran paket dapat dilihat pada Tabel 9.

Tabel 9. *Packet loss* untuk simulasi variasi ukuran paket

Simulasi	Skema	Ukuran paket	*Packet loss*	
			Core router	Jaringan
Ukuran paket	Diffserv tanpa AFC	500 *bytes*	1016	1016
		1500 *bytes*	1076	1094
Total			2092	2110
Ukuran paket	Diffserv dengan AFC	500 *bytes*	1036	1036
		1500 *bytes*	347	347
Total			1383	1383

Total *packet loss* pada *Diffserv* dengan AFC untuk semua *aggregate* di *core router* mengalami penurunan sekitar ±33,8% (709 paket) dari *Diffserv* tanpa AFC, sedangkan total *packet loss* di jaringan mengalami penurunan sekitar ±36,4 % (727 paket) dari *Diffserv* tanpa AFC. Sama seperti simulasi variasi jumlah *microflows*, *Diffserv* dengan AFC mem-*blocking* trafik *aggregate* yang menyebabkan kongesti di *core router* sehingga kongesti tidak terjadi dan *packet loss* yang terjadi juga sedikit.

4.2.3 *Packet Loss* untuk Simulasi variasi *Round Trip Time* (RTT)

Hasil *packet loss* yang terjadi untuk simulasi variasi RTT dapat dilihat pada Tabel 10. Pada *Diffserv* dengan AFC, total *packet loss* yang terjadi untuk semua *aggregate* mengalami penurunan sekitar ±54,5% (6 paket) untuk *core router* 1, ±89,2% (182 paket) untuk *core router* 2, ±53,6% (708 paket) untuk *core router* 3 dan ±58,6% (938 paket) untuk jaringan. Hal ini terjadi karena trafik *aggregate* yang menyebabkan kongesti akan mengalami *blocking* pada *edge router*.

Tabel 10. *Packet loss* untuk simulasi variasi RTT

Simulasi	Skema	RTT	*Packet loss*			
			Core router 1	*Core router* 2	*Core router* 3	Jaringan
RTT	Diffserv tanpa AFC	10 ms	5	34	421	526
		100 ms	6	88	327	421
		200 ms	0	82	266	348
		300 ms	0	0	305	305
Total			11	204	1319	1600
RTT	Diffserv dengan AFC	10 ms	5	5	120	154
		100 ms	0	11	147	158
		200 ms	0	6	169	175
		300 ms	0	0	175	175
Total			5	22	611	662

4.2.4 *Packet Loss* untuk Simulasi Variasi Kelas AF

Hasil *packet loss* untuk simulasi variasi kelas AF dapat dilihat pada Tabel 11.

Tabel 11. *Packet loss* untuk simulasi variasi kelas AF

Simulasi	Skema	Kelas AF	*Packet loss*			
			Core router 1	*Core router* 2	*Core router* 3	Jaringan
Kelas AF	Diffserv tanpa AFC	AF1	0	0	220	220
		AF2	0	0	62	62
		AF2	0	0	118	118
		AF3	0	0	159	159
Total			0	0	559	559
Kelas AF	Diffserv dengan AFC	AF1	0	0	187	187
		AF2	0	0	50	50
		AF2	0	0	101	101
		AF3	0	0	37	37
			0	0	375	375

Pada *Diffserv* dengan AFC, *packet loss* yang terjadi lebih sedikit walaupun tidak dapat menciptakan *fairness* dalam *throughput*. Pada *Diffserv* dengan AFC, total *packet loss* yang terjadi mengalami penurunan ±32,9% untuk *core router* 3 dan jaringan. Hal ini

membuktikan bahwa skema AFC masih dapat mengendalikan kongesti di simulasi kelas AF yang berbeda walaupun dalam *throughput* tidak tercapai *fair sharing*.

5. KESIMPULAN

Dari hasil simulasi yang telah dilakukan dapat disimpulkan sebagai berikut:

1. Pada kelas trafik yang sama, *Diffserv* dengan AFC dapat menciptakan *fairness* dalam *throughput* bagi tiap *aggregate* daripada *Diffserv* tanpa AFC.
2. Pada kelas trafik yang berbeda, *Diffserv* dengan AFC tidak dapat menciptakan *fairness* dalam *throughput* bagi tiap *aggregate* daripada *Diffserv* tanpa AFC.
3. *Diffserv* dengan AFC dapat meminimalisasikan *packet loss* yang terjadi di jaringan daripada *Diffserv* tanpa AFC.

REFERENSI

[1]. Alonso, S.H., et all. 2003. *Improving Aggregate Flow Control in Differentiated Services Networks.* Spanyol: ETSE Telecomunication, University of Vigo.

[2]. Blake, S.,et.al. 1998. *RFC 2475: An Architecture for Differentiated Services*, (Online), (http://www.faqs.org/rfcs/rfc2475.html, diakses 18 Oktober 2007).

[3]. Floyd, S. and Jacobson, V. 1993. *Random Early Detection for Congestion Avoidance*, (Online), (ftp://ftp.ee.lbl.gov/papers/**early**.pdf, diakses 24 Oktober 2007).

[4]. Heinanen J., et.al. 1999. *RFC 2597: Assured Forwarding PHB Group,* (Online), (http://www.faqs.org/rfcs/rfc2597.html, diakses 19 Oktober 2007).

[5]. Nandy, B., et all. 2003. *Aggregate Flow Control: Improving Assurances for Differntiated Services*. Canada: Nortel Networks.

[6]. Nichols, K., et.al. 1999. *RFC 2859: A Time Slide Window Three Colour Marker(TSWTCM),* (Online), (http://www.faqs.org/rfcs/rfc2859.html, diakses 26 Oktober 2007).

[7]. Pieda, P., et.al. 2000. *A Network Simulator Differentiated Service Implementation,* (Online), (http://www.sop.inria.fr/mistral/personnel/Eitan.Altman/COURS-NS/DOC/DSnortel.pdf, diakses 2 November 2007)

[8]. Veiga, M.F., et all. 2003. *Stabilized Edge-to-Edge Aggregate Flow Control*, Spanyol: ETSE Telecomunication, University of Vigo.

[9]. Wirawan, A. dan Indarto, Eka. 2004. *Mudah Membangun Simulasi dengan Network Simulator-2*. Yogyakarta: Penerbit Andi