MENINGKATKAN QoS PADA *MULTIPROTOCOL LABEL SWITCHING* DENGAN METODE *DIFFERENTIATED SERVICE-AWARE TRAFFIC ENGINEERING*

SKRIPSI

KONSENTRASI TEKNIK TELEKOMUNIKASI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:

DEBBY OCTAVIA
NIM. 0410630024-63

DEPARTEMEN PENDIDIKAN NASIONAL
UNIVERSITAS BRAWIJAYA
FAKULTAS TEKNIK
MALANG
2009

LEMBAR PERSETUJUAN

MENINGKATKAN QoS PADA *MULTIPROTOCOL LABEL SWITCHING* DENGAN METODE *DIFFERENTIATED SERVICE-AWARE TRAFFIC ENGINEERING*

SKRIPSI

KONSENTRASI TEKNIK TELEKOMUNIKASI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:

DEBBY OCTAVIA
NIM. 0410630024-63

Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Rusmi Ambarwati, ST., MT.	**Ir. Endah Budi P., MT.**
NIP. 19720204 200003 2 002	**NIP. 19621116 198903 2 002**

LEMBAR PENGESAHAN

MENINGKATKAN QOS PADA *MULTIPROTOCOL LABEL SWITCHING* DENGAN METODE *DIFFERENTIATED SERVICE-AWARE TRAFFIC ENGINEERING*

Disusun oleh:

DEBBY OCTAVIA
0410630024-63

Skripsi ini telah diuji dan dinyatakan lulus pada
tanggal **23 Juni 2009**

Majelis Penguji :

Ir. Erfan A. Dahlan, MT
NIP. 19530714 198203 1 003

Ali Mustofa, ST., MT
NIP. 19710601 200003 1 001

Ir. Wahyu Adi P., MT
NIP. 19600518 198802 1 001

Mengetahui,

Ketua Jurusan Teknik Elektro

Ir. Heru Nurwarsito, M.Kom
NIP. 19650402 199002 1 001

KATA PENGANTAR

Puji syukur ke hadirat Tuhan Yang Maha Esa atas segala limpahan rahmat dan karunia-Nya, akhirnya penulis dapat menyelesaikan skripsi yang berjudul "Meningkatkan QoS pada *Multiprotocol Label Switching* dengan Metode *Differentiated Service-Aware Traffic Engineering*" ini.

Skripsi ini disusun untuk memenuhi sebagian persyaratan memperoleh gelar Sarjana Teknik dan merupakan salah satu persyaratan yang harus ditempuh dalam menyelesaikan pendidikan di Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya Malang.

Penulis ingin mengucapkan terima kasih yang sebesar-besarnya kepada Ibu Rusmi Ambarwati, ST., MT. dan Ibu Endah Budi Purnomowati, Ir., MT. sebagai dosen pembimbing atas saran, konsultasi, semangat, kesabaran dan waktu dalam pengerjaan hingga penyelesaian tugas akhir ini. Tidak terkecuali, ucapan terima kasih penulis haturkan kepada :

1. Ayah, Bunda, Adik, Nenek, Sutarsis (Mas Sis) dan seluruh keluarga tercinta yang selalu memberikan pengertian, dorongan moril, materiil dan doa restu.
2. Bapak Ir. Heru Nurwarsito, M.Kom. selaku Ketua Jurusan Teknik Elektro.
3. Bapak Rudy Yuwono, ST. MSc. selaku Sekretaris Jurusan Teknik Elektro.
4. Ibu Endah Budi Purnomowati, Ir., MT. selaku Ketua Kelompok Dosen dan Keahlian Teknik Telekomunikasi.
5. Bapak R. Arief Setyawan, ST., MT. selaku Kepala Laboratorium Dasar Pemrograman Komputer.
6. Bapak dan Ibu dosen Jurusan Teknik Elektro Universitas Brawijaya, khususnya Ibu Asri Wulandari, ST., MT.; Bapak R. Arief Setyawan, ST., MT. dan Bapak Herman Tolle, ST., MT. atas kesediaannya dalam memberikan waktu, ide, saran dan semangat.
7. Segenap staf dan karyawan Jurusan Teknik Elektro Universitas Brawijaya, khususnya mas Mulyadi laboran laboratorium Dasar Komputer dan Pemrograman, Pak Sanawi (Pak Wi) dan Mas Mulyadi karyawan perpustakaan jurusan atas dorongan dan dukungannya.
8. Rekan-rekan asisten Laboratorium Dasar Komputer dan Pemrograman, khususnya Mas Diriga, Prima, Wirawan (Wam), Indra (Rooney), Mas Hiksa, Mas Hatta, Mas Malik. Adik-adik asisten angkatan 2005 sampai 2007, Disyam

(Didit), Eka, Adityo, Putri, Norman, Krisna, Rachmania atas *sharing*, waktu, dukungan, keceriaan, semangat, bantuan, sumbangan pengetahuan dan tenaganya.

9. Rekan-rekan UPPTI Brawijaya dan TPTI Fakultas Teknik, khususnya Mas Yudho, Mas Alan, Mas Ratno, Winda, Handoko (Koko), Kharisma (Bara), Yoga, Wahyu, Aji, Riza.
10. Rekan-rekan asisten Laboratorium Sistem Informasi, khususnya Mas Marjhy, Rackhmadany (Pepeng), *sharing*, keceriaan, semangat, dukungan dan pengetahuannya.
11. Teman-teman Elektro Brawijaya, Jarot, Gammalia (Gamma), Mas Fajar Wahyu, Mbak Nuur Aisyah (Mbak Ais), Mbak Firdiana (Mbak Dina), Mbak Marcellina (Mbak Tita), Fardanto (Danto), atas *sharing*, keceriaan, semangat, dukungan, pengetahuan serta pertolongannya.
12. Bapak Abdi pada forum "Komunitas Indonesia *Open Source* (KIOS)" atas *sharing*, saran dan sumbangan pengetahuannya.
13. Teman-teman Elektro 2004, khususnya Ario, Reza, Qoriatul (Qori), Choiriyah (Onice), Ferianto (Feri), Dina, Mei, Dyah A.R. (Dany), Anissatussa'diyah (Icha), Agus, Eri, Roghib, Suciana, Junerio (Adin) atas kebersamaan & canda tawa yang akan senantiasa menjadi kenangan indah tak terlupakan.
14. Seluruh rekan-rekan mahasiswa Jurusan Teknik Elektro atas kebersamaan dan pengetahuannya.
15. Dan semua pihak yang tidak dapat penulis sebutkan satu-persatu.

Penulis menyadari bahwa skripsi ini masih jauh dari kesempurnaan. Karena itu segala kritik dan saran sangat diharapkan demi sempurnanya skripsi ini. Akhir kata, semoga skripsi ini dapat bermanfaat bagi rekan-rekan mahasiswa khususnya dan bagi seluruh pembaca pada umumnya.

Malang, 10 Juni 2009

Penulis

DAFTAR ISI

PENGANTAR i
DAFTAR ISI iii
DAFTAR TABEL vii
DAFTAR GAMBAR ix
DAFTAR LAMPIRAN xii
RINGKASAN xiii

BAB I. PENDAHULUAN
1.1. Latar Belakang 1
1.2. Rumusan Masalah 2
1.3. Ruang Lingkup 2
1.4. Tujuan 3
1.5. Sistematika Penulisan 3

BAB II. TCP/IP, *Multiprotocol Label Switching* (MPLS) dan MPLS *Traffic Engineering* (MPLS TE)
2.1. Model arsitektur TCP/IP 5
2.1.1. *Applications Layer* 5
2.1.2. *Transport Layer* 6
2.1.3. *Internetwork Layer* 7
2.1.4. *Network Interface Layer* 8
2.2. Enkapsulasi Data 9
2.3. MPLS dan Model Referensi OSI 10
2.4. Arsitektur MPLS 11
2.4.1. Label MPLS 11
2.4.2. MPLS *Encoding* 12
2.5. *Label Switch Router* 12
2.6. *Label Switch Path* 13
2.7. Distribusi Label 14
2.8. *Label Distribution Protocol* 15
2.9. *Label Forwarding Instance Base* 16
2.10. Rekayasa Trafik (*Traffic Engineering*) 16

2.11. MPLS *Traffic Engineering* (MPLS TE) 17
2.12. Operasi MPLS TE 19
2.13. *Resource Reservation Protocol* 21
2.13.1. RSVP dan Label 21
2.14. *Open Short Path First* (OSPF) 23
2.14.1. Terminologi OSPF 23
2.14.1.1. OSPF Area 23
2.14.1.2. *Neighbor* 24
2.14.1.3. Tipe Jaringan Fisik 24
2.14.1.4. *Adjacency* 24
2.14.1.5. Basis Data Keadaan Saluran (*Link State Database*) 25
2.14.1.6. *Link State Advertisement* dan *Flooding* 25

BAB III. MPLS *Differentiated Service* (MPLS DiffServ) dan MPLS *Differentiated Service-aware Traffic Engineering* (MPLS DS-TE)

3.1. *Quality of Service* 26
3.2. *Differentiated Service* dan Paket IP 26
3.2.1. *Differentiated Service Code Point* 27
3.2.2. *Per-Hop Behaviour* 28
3.2.3. *Expedited Forwarding* (EF) 29
3.2.4. *Assured Forwarding* (AF) 29
3.2.5. *Default* PHB 29
3.3. Mengimplementasikan IP QoS 29
3.3.1. Mengidentifikasi Trafik dan Kebutuhannya 30
3.3.2. Pengklasifikasian Trafik 31
3.3.3. Mendefinisikan QoS *Policy* 31
3.4. Mekanisme IP QoS 32
3.4.1. Klasifikasi dan Penandaan 32
3.4.2. Pengelolaan Kepadatan 33
3.4.3. Pencegahan Kepadatan 33
3.4.3.1. *Random Early Detection* (RED) 34
3.4.3.2. *Weighted Random Early Detection* (WRED) 35
3.4.4. *Policing* 37

3.5. *Differentiated Service* dan Paket MPLS 40
3.5.1. EXP-*inferred-class* LSP (E-LSP) 40
3.6. MPLS DiffServ *Tunneling Model* 41
3.6.1. Model Pipa Pendek (*Short Pipe Model*) 42
3.7. *Differentiated Service-Traffic Engineering* 43
3.8. Parameter Performansi Jaringan 44
3.8.1. *Throughput* 44
3.8.2. *Delay End to End* 44
3.8.3. *Delay* Proses 46
3.8.4. *Delay Antrian* 47
3.8.5. *Delay* Propagasi 50
3.8.6. *Delay* Transmisi 50

BAB IV. METODOLOGI

4.1. Studi Literatur 51
4.2. Perancangan Simulasi 51
4.3. Pengujian 51
4.4. Pengambilan Data 52
4.4.1. Data Primer 52
4.4.2. Data Sekunder 52
4.5. Analisa 53
4.6. Pengambilan Kesimpulan 53

BAB V. Perancangan, Pengujian dan Analisis Performansi Metode *DiffServ* dan DS-TE pada MPLS

5.1. Perancangan Implementasi Jaringan MPLS 55
5.1.1. Perancangan Topologi Jaringan MPLS 55
5.1.2. Perancangan *Quality of Service* 56
5.1.2.1. Identifikasi Trafik 59
5.1.2.2. Klasifikasi Trafik 59
5.1.2.3. Mendefinisikan QoS *Policy* untuk Tiap Kelas 59
5.1.3. Penerapan Rekayasa Trafik untuk Simulasi Metode DS-TE 61
5.2. Perancangan Penerapan Perangkat Keras dan Antarmuka 62
5.3. Perancangan Penerapan Perangkat Lunak 64

5.3.1. Perancangan Pembangkitan Trafik 64
5.3.2. Perancangan Simulator Jaringan MPLS 65
5.3.3. Penerapan Perangkat Lunak pada Sisi *Source* 66
5.3.4. Penerapan Perangkat Lunak pada Sisi PC MPLS 67
5.3.5. Algoritma Konfigurasi *Router*..... 68
5.3.6. Penerapan Perangkat Lunak pada Sisi *Destination* 69
5.4. Pengujian dan Analisis 72
5.5. Hasil Simulasi 78
5.6. *Throughput* 80
5.7. Analisis Perhitungan Teoretis 83
5.7.1. Analisis *Troughput* 84
5.7.2. Analisis *Delay end-to-end* Metode DiffServ 85
5.7.2.1. *Delay* Enkapsulasi 85
5.7.2.2. *Delay* Dekapsulasi 88
5.7.2.3. *Delay* Antrian 89
5.7.2.4. *Delay* Propagasi 90
5.7.2.5. *Delay* Transmisi 91
5.7.2.6. *Delay end to end* 92
5.7.3. Analisis *Delay end-to-end* Metode DS-TE 92
5.8. Perbandingan Pengujian dan Teori 93

BAB VI. PENUTUP

6.1. Kesimpulan 97
6.2. Saran 98

DAFTAR PUSTAKA

LAMPIRAN

DAFTAR TABEL

No	Judul	Halaman
Tabel 3.1	Pemetaan PHB dan DSCP yang Direkomendasikan IETF	28
Tabel 3.2	PHB AF dan *Drop Presedance Class Selector*	29
Tabel 3.3	EF *Profile*	36
Tabel 3.4	AF *Profile*	37
Tabel 5.1	Block Sizes	56
Tabel 5.2	Alokasi *Block* dan Subnet untuk Tiap Jaringan	56
Tabel 5.3	Alamat IP untuk Tiap Jaringan	56
Tabel 5.4	Alamat IP untuk Tiap Antarmuka	57
Tabel 5.5	Alamat IP untuk Tiap *Host*	57
Tabel 5.6	Klasifikasi QoS	59
Tabel 5.7	*Service Level Agreement*	60
Tabel 5.8	MPLS *Experimental Bit*	60
Tabel 5.9	QoS *policy* untuk Metode DiffServ	61
Tabel 5.10	QoS *Policy* untuk Metode DS-TE	61
Tabel 5.11	Tabel Karakteristik Paket	64
Tabel 5.12	Pengambilan Data pada P1 untuk Metode DiffServ	78
Tabel 5.13	Pengambilan Data pada P1 untuk Metode DS-TE	79
Tabel 5.14	Pengambilan Data pada P2 untuk Metode DiffServ	79
Tabel 5.15	Pengambilan Data pada P2 untuk Metode DS-TE	80
Tabel 5.16	Hasil Perhitungan Nilai *Throughput* pada P1 untuk Metode DiffServ	81
Tabel 5.17	Hasil Perhitungan Nilai *Throughput* pada P1 untuk Metode DS-TE	81
Tabel 5.18	Hasil Perhitungan Nilai *Throughput* pada P1 untuk Metode DiffServ	82
Tabel 5.19	Hasil Perhitungan Nilai *Throughput* pada P1 untuk Metode DS-TE	83
Tabel 5.20	Karakteristik Paket	84
Tabel 5.21	*Packet Loss*	84
Tabel 5.22	*Throughput* Ethernet pada P1	85

Tabel 5.23	*Throughput* Ethernet pada P2	85
Tabel 5.24	*Throughput* dengan Label MPLS	85
Tabel 5.25	*Throughput* Tanpa Label MPLS	85
Tabel 5.26	*Delay* Enkapsulasi *Source*	86
Tabel 5.27	*Delay* Enkapsulasi dengan Penyisipan Label MPLS	87
Tabel 5.28	*Delay* Enkapsulasi Tanpa Penyisipan Label MPLS	88
Tabel 5.29	*Delay* Enkapsulasi pada r3	88
Tabel 5.30	*Delay* Dekapsulasi pada *ingress router*	88
Tabel 5.31	*Delay* Dekapsulasi pada *intermediate router*	88
Tabel 5.32	*Delay* Dekapsulasi pada *egress router*	89
Tabel 5.33	*Delay* Antrian dengan Label MPLS	90
Tabel 5.34	*Delay* Antrian Tanpa Label MPLS	90
Tabel 5.35	*Delay* Antrian pada *egress router*	90
Tabel 5.36	*Delay* Transmisi pada *Source*	91
Tabel 5.37	*Delay* Transmisi dengan Label MPLS	91
Tabel 5.38	*Delay* Enkapsulasi Tanpa Penyisipan Label MPLS	91
Tabel 5.39	*Delay* Transmisi pada *egress router*	92
Tabel 5.40	*Delay end to end* Metode DiffServ	92
Tabel 5.41	*Delay end to end* Metode DS-TE Hasil Perhitungan Teori	93
Tabel 5.42	*Delay end to end* Metode DiffServ Hasil Pengujian	93
Tabel 5.43	*Delay end to end* Metode DiffServ Hasil Perhitungan Teori	93
Tabel 5.44	*Delay end to end* Metode DS-TE Hasil Pengujian	93
Tabel 5.45	*Delay end to end* Metode DS-TE Hasil Perhitungan Teori	93

DAFTAR GAMBAR

No	Judul	Halaman
Gambar 2.1	Detail Model Arsitektur TCP/IP	5
Gambar 2.2	Format Segmen TCP	6
Gambar 2.3	Format Segmen UDP	7
Gambar 2.4	IP *Header*	8
Gambar 2.5	Komunikasi Antar Komputer pada OSI *LAYER*	9
Gambar 2.6	*Frame*, *Packet* dan Segmen	10
Gambar 2.7	Proses Enkapsulasi pada Pengiriman e-*mail*	10
Gambar 2.8	OSI *Layer* dan MPLS *Layer*	11
Gambar 2.9	MPLS Label	12
Gambar 2.10	Enkapsulasi untuk Paket Berlabel	12
Gambar 2.11	Sebuah LSP yang Melalui Jaringan MPLS	14
Gambar 2.12	Sebuah Jaringan IPV4 *over* MPLS yang Menjalankan LDP	15
Gambar 2.13	*Packet Switching*	16
Gambar 2.14	*Traffic Engineering*	17
Gambar 2.15	MPLS TE *Head End Router*	18
Gambar 2.16	MPLS TE *Building Blocks*	19
Gambar 2.17	RSVP untuk TE dan Label	22
Gambar 2.18	Penerusan Paket	23
Gambar 3.1	IPV4 Datagram *Header*	27
Gambar 3.2	Oktet TOS pada IPV4	27
Gambar 3.3	DiffServ *Field*	28
Gambar 3.4	Tiga Langkah untuk Membangun QOS	30
Gambar 3.5	Mengklasifikasikan Trafik ke dalam Kelas-Kelas	31
Gambar 3.6	Mengklasifikasikan Trafik	32
Gambar 3.7	Penandaan Trafik	32
Gambar 3.8	Pengelolaan Kepadatan	33
Gambar 3.9	Pencegahan Kepadatan	34
Gambar 3.10	RED *Profile*	35
Gambar 3.11	DSCP-*Based* WRED (*Expedited Forwarding*)	36
Gambar 3.12	DSCP-Based WRED (*Assured Forwarding*)	37

Gambar 3.13	*Policing*	38
Gambar 3.14	*Single Token Bucket* (*conform*)	39
Gambar 3.15	*Single Token Bucket* (*exceed*)	39
Gambar 3.16	*Single Token Bucket Class-Based Policing*	39
Gambar 3.17	MPLS *Shim Header*	40
Gambar 3.18	Jaringan MPLS yang Menggunakan E-LSP	41
Gambar 3.19	Operasi Umum dari MPLS DiffServ *Tunneling Model*	41
Gambar 3.20	Short Pipe Model *untuk MPLS VPN*	43
Gambar 3.21	*Delay end to end* pada Jalur *source*-r1-r2-r3-*destination*	46
Gambar 3.22	*Delay end to end* pada Jalur *source*-r1-r5-r4-r3-*destination*	46
Gambar 3.23	Model Antrian M/M/1	49
Gambar 3.24	Analisis *Delay* Antrian	49
Gambar 5.1	Topologi Jaringan MPLS	55
Gambar 5.2	Pengalamatan Jaringan MPLS	57
Gambar 5.3	Alokasi *Bandwidth* Jaringan MPLS	58
Gambar 5.4	Topologi Simulasi Jaringan MPLS	64
Gambar 5.5	Diagram Alir Perancangan Simulasi Jaringan MPLS	65
Gambar 5.6	Tampilan Dynagen yang Menjalankan Lima Virtual *Router*	66
Gambar 5.7	Tampilan Dynamips *Server*	67
Gambar 5.8	Tampilan Dynagen	67
Gambar 5.9	Tampilan VMware pada PC MPLS	68
Gambar 5.10	Algoritma Konfigurasi *Router* dengan Metode DiffServ	68
Gambar 5.11	Algoritma Konfigurasi Dasar pada Metode DiffServ	69
Gambar 5.12	Algoritma Konfigurasi Kelas pada Metode DiffServ	69
Gambar 5.13	Algoritma Konfigurasi Policy pada Metode DiffServ	70
Gambar 5.14	Algoritma Konfigurasi *Router* dengan Metode DS-TE	70
Gambar 5.15	Algoritma Konfigurasi Dasar pada Metode DS-TE	71
Gambar 5.16	Algoritma Konfigurasi MPLS TE *Tunnel* pada Metode DS-TE	71
Gambar 5.17	Algoritma Konfigurasi Kelas pada Metode DS-TE	72
Gambar 5.18	Algoritma Konfigurasi *Policy* pada Metode DS-TE	72
Gambar 5.19	Hasil Keluaran Perintah “ping 192.16.10.35” pada *Destination*	73

Gambar 5.20	Hasil Keluaran Perintah "tracert 192.16.10.35" pada *Destination*	73
Gambar 5.21	Hasil Keluaran dari Perintah "show ip route" pada r1	73
Gambar 5.22	Hasil Keluaran Perintah "show ip route" pada r1	74
Gambar 5.23	Hasil Keluaran Perintah "show mpls forwarding-table" pada r1	74
Gambar 5.24	Hasil Keluaran dari Perintah "policy-map interface" pada r1	74
Gambar 5.25	Hasil Keluaran Perintah "show mpls traffic-eng tunnels" pada r1	75
Gambar 5.26	Hasil keluaran dari Perintah "show mpls traffic-eng topology" pada r1	76
Gambar 5.27	Pengiriman Data pada *Source*	76
Gambar 5.28	Penerimaan Data pada *Destination*	77
Gambar 5.29	Paket MPLS pada Antarmuka S1/0 pada r1	77
Gambar 5.30	Paket MPLS pada Antarmuka S2/0 pada r1	78
Gambar 5.31	Grafik Perbandingan *Delay end to end* Metode DiffServ dan Metode DS-TE pada P1	94
Gambar 5.32	Grafik Perbandingan *Delay end to end* Metode DiffServ dan Metode DS-TE pada P2	94
Gambar 5.33	Grafik Perbandingan *Throughput* Metode DiffServ dan Metode DS-TE pada P1	95
Gambar 5.34	Grafik Perbandingan *Throughput* Metode DiffServ dan Metode DS-TE pada P2	95
Gambar 5.35	Grafik Perbandingan *Packet Loss* pada Metode DiffServ dan Metode DS-TE pada P1 dan P2	95

DAFTAR LAMPIRAN

No	Judul	Halaman
Lampiran 1.	Konfigurasi Jaringan MPLS pada Dynagen	90
Lampiran 2.	Konfigurasi Metode DiffServ	92
Lampiran 3.	Konfigurasi Metode DS-TE	98
Lampiran 4.	Konfigurasi *Packet Generator*	105
Lampiran 5.	PHB *Class Selector*	107

RINGKASAN

Debby Octavia, Jurusan Teknik Elektro, Fakultas Teknik, Universitas Brawijaya, Juni 2009, *Meningkatkan QoS pada Multiprotocol Label Switching dengan Metode Differentiated Service-Aware Traffic Engineering*, Dosen Pembimbing: Rusmi Ambarwati, ST., MT., dan Endah Budi Purnomowati, Ir., MT.

Multiprotocol Label Switching (MPLS) adalah teknologi penyampaian paket pada jaringan *backbone* berkecepatan tinggi yang ditawarkan sebagai alternatif teknologi untuk memperbaiki kinerja jaringan IP. Suatu mekanisme untuk menyediakan QoS di internet sangatlah diperlukan untuk memenuhi kebutuhan dan keinginan *user*. *Differentiated Service* (DiffServ) merupakan salah satu arsitektur QoS yang saat ini digunakan untuk mencapai kualitas layanan yang lebih tinggi. Jaringan MPLS dapat menggunakan mekanisme rekayasa trafik (*Traffic Engineering*-TE) untuk meminimalisasi kepadatan dan meningkatkan performansi jaringan. Kedua teknik diatas, yaitu DiffServ dan TE jika dikombinasikan dapat meningkatkan QoS pada jaringan MPLS, kombinasi ini disebut MPLS DS-TE (*Differentiated Service-Aware Traffic Engineering*). MPLS DS-TE mengijinkan *per-class* TE dan menyediakan kontrol yang lebih baik untuk meminimalisasi kepadatan jaringan dan meningkatkan performansi jaringan MPLS. Untuk membuktikan performansi teknologi tersebut maka pada skripsi ini dibuat simulasi dan analisis untuk membandingkan performansi teknik DiffServ dan teknik DS-TE yang diterapkan pada jaringan MPLS. Simulasi dilakukan dengan menggunakan *software* emulator *router* untuk menciptakan jaringan MPLS, dan menggunakan *software* pembangkit trafik untuk dilewatkan dalam jaringan MPLS. Trafik yang dibangkitkan berupa trafik UDP dan TCP yang kemudian diberi nama *voice*, *mission critical*, *bulk* dan *best effort*. Parameter performansi yang dibandingkan adalah *throughput*, *packet loss*, dan *delay* pengiriman paket.

Dari hasil simulasi dan analisa disimpulkan bahwa sistem simulasi yang dirancang membuktikan bahwa penerapan metode DS-TE pada jaringan MPLS tidak dapat menjalankan *per-class* TE pada jaringan yang padat dan pada konfigurasi topologi yang telah dirancang. Melalui pengambilan data, diperoleh rata-rata *troughput* trafik UDP yang mencapai 1,0779 paket/detik, lebih kecil dibandingkan dengan *troughput* pada percobaan dengan DiffServ yang hanya mencapai 2,3761 paket/detik. Sedangkan hasil *troughput* untuk trafik TCP lainnya adalah sebaliknya. Untuk hasil pengujian *packet loss* pada penerapan metode DS-TE diperoleh *packet loss* sebesar 182 paket yang lebih besar dibandingkan pada metode DiffServ dengan 98 paket. Selain hal-hal tersebut dari simulasi metode DS-TE ini juga diperoleh kenyataan hasil pengujian *delay end to end* sebesar 158,2 detik untuk paket UDP yang lebih lama dibandingkan dengan *delay* metode DiffServ yang sebesar 112,6 detik.

Kata kunci: *Differentiated Service*, MPLS, *Differentiated Service-Aware Traffic Engineering*, *Quality of Service*.

BAB I
PENDAHULUAN

1.1. Latar Belakang

Seiring perkembangan teknologi, manusia semakin dimanjakan dengan berbagai kecanggihan teknologi internet dengan biaya yang relatif murah. Ketersediaan teknologi yang murah tersebut meningkatkan permintaan akan kebutuhan aplikasi multimedia seperti contohnya, VoIP, IPTV, *Video Streaming* dan *Video on Demand*. Hal inilah yang menjadi tantangan bagi penyedia layanan jaringan (*network service provider*) untuk menyediakan layanan yang berkualitas.

Multiprotocol Label Switching (MPLS) adalah teknologi penyampaian paket pada jaringan *backbone* berkecepatan tinggi yang menggunakan *switching node* untuk mempercepat pengiriman paket. MPLS ditawarkan sebagai alternatif teknologi baru melihat arsitektur teknologi IP, yang merupakan infrastruktur utama dalam internet, banyak menghadapi kendala dan tidak mampu memberikan QoS yang memuaskan terlebih untuk kebutuhan data *real time*.

Suatu mekanisme untuk menyediakan QoS di internet sangatlah diperlukan untuk memenuhi kebutuhan dan keinginan *user*. *Differentiated Service* (DiffServ) merupakan salah satu arsitektur QoS yang digunakan saat ini untuk mencapai kualitas layanan yang lebih tinggi. DiffServ menyediakan diferensiasi layanan, dengan membagi trafik atas kelas-kelas, dan memperlakukan setiap kelas secara berbeda. Tetapi teknik ini tidak menjamin adanya *bandwidth resource* yang cukup.

Jaringan MPLS dapat menggunakan mekanisme rekayasa trafik (*Traffic Engineering*-TE) untuk meminimalisasi kepadatan dan meningkatkan performansi jaringan. TE memodifikasi pola *routing* untuk menyediakan pemetaan aliran trafik yang efisien ke dalam *resource* jaringan.

Teknik MPLS DiffServ-*Traffic Engineering* (DS-TE) merupakan teknologi yang mengkombinasikan kelebihan-kelebihan yang dimiliki oleh DiffServ dan MPLS TE untuk meningkatkan QoS. (Scott Heinlein, 2005:7) MPLS DS-TE mengijinkan *per-class* TE dalam sebuah jaringan MPLS. DS-TE menyediakan kontrol yang lebih baik untuk meminimalisasi kepadatan jaringan dan meningkatkan performansi jaringan.

Untuk membuktikan kebenaran konsep tersebut maka pada skripsi ini dibuat simulasi dan analisis untuk membandingkan performansi teknik DiffServ dan teknik

DS-TE yang diterapkan pada jaringan MPLS. Parameter performansi yang dibandingkan adalah *throughput, packet loss,* dan *delay* pengiriman paket.

1.2. Rumusan Masalah

Berdasarkan latar belakang yang telah dipaparkan di atas, maka dapat dirumuskan permasalahan sebagai berikut:

1. Bagaimana penerapan metode *Differentiated Service* pada jaringan MPLS.
2. Bagaimanakah penerapan metode MPLS DiffServ-*Traffic Engineering* pada jaringan MPLS.
3. Bagaimana analisis *throughput, packet loss,* dan *delay* pengiriman paket pada jaringan MPLS dengan menggunakan metode DiffServ.
4. Bagaimana analisis *throughput, packet loss,* dan *delay* pengiriman paket pada jaringan MPLS dengan menggunakan metode DS-TE.

1.3. Ruang Lingkup

Berdasarkan permasalahan yang telah disebutkan, maka penulisan skripsi ini dibatasi pada hal-hal sebagai berikut:

1. Membahas konsep pengiriman paket dan pemberian label pada jaringan *Multiprotocol Label Switching*.
2. Membahas konsep teknik *Differentiated Service tunneling mode* pada jaringan MPLS.
3. Membahas konsep MPLS *Traffic Engineering* dalam jaringan MPLS.
4. Membahas konsep *Differentiated Service-Traffic Engineering* dalam jaringan MPLS.
5. Membahas analisis teknik DS-TE yang dibandingkan dengan penggunaan teknik DiffServ dalam pengiriman suatu trafik paket pada jaringan MPLS.
6. Protokol *routing* menggunakan OSPF.
7. Pensinyalan MPLS TE menggunakan RSVP.
8. Menggunakan IPv4.
9. Menggunakan emulator Dynamips dengan IOS c7200-advipservicesk9-mz[1].124-15.T
10. Aliran paket searah (dari sumber ke tujuan).
11. Menggunakan Packgen 0.2 sebagai pembangkit trafik.
12. Perbandingan performansi MPLS DiffServ dan MPLS DS-TE berdasarkan analisis *throughput, packet loss,* dan *delay*.

13. Menggunakan *Short Pipe Tunneling Mode*.
14. Asumsi ukuran paket *voice* (UDP) sebesar 40 *byte* dan paket data (TCP) sebesar 1000 *byte*.
15. Asumsi *bandwidth* trafik yang dibangkitkan oleh generator trafik untuk percobaan pertama adalah seperempat *bandwidth* kelas di saluran Serial pada metode MPLS DiffServ. Dengan durasi pengiriman trafik selama 60 detik untuk trafik *mission critical*, 50 detik untuk trafik *bulk*, 40 detik untuk trafik *best effort*, dan 30 detik untuk trafik *voice*.
16. Asumsi *bandwidth* trafik yang dibangkitkan oleh generator trafik untuk percobaan kedua adalah setengah dari *bandwidth* kelas di saluran Serial pada metode MPLS DiffServ. Dengan durasi pengiriman trafik selama 60 detik untuk trafik *mission critical*, 50 detik untuk trafik *bulk*, 40 detik untuk trafik *best effort*, dan 30 detik untuk trafik *voice*.

1.4. Tujuan

Berdasarkan rumusan masalah yang ada maka tujuan dari penulisan skripsi ini adalah menganalisis performansi penerapan metode DS-TE yang dibandingkan dengan penggunaan metode DiffServ dalam pengiriman suatu trafik paket pada jaringan MPLS terhadap *throughput*, *packet loss*, dan *delay*.

1.5. Sistematika Penulisan

Sistematika penulisan laporan skripsi ini adalah sebagai berikut:

BAB I Pendahuluan

Menjelaskan tentang latar belakang, rumusan masalah, ruang lingkup, tujuan, dan sistematika penulisan.

BAB II TCP/IP, *Multiprotocol Label Switching* (MPLS) dan MPLS *Traffic Engineering* (MPLS TE)

Menjelaskan tentang konsep MPLS dan MPLS *Traffic Engineering*.

BAB III MPLS *Differentiated Service* (MPLS DiffServ) dan MPLS *Differentiated Service-aware Traffic Engineering* (MPLS DS-TE)

Menjelaskan tentang konsep MPLS DiffServ, dan konsep MPLS DS-TE.

BAB IV Metodologi

Menjelaskan tentang tahapan perencanaan pembuatan skripsi yang telah direncanakan.

BAB V Perancangan, Pengujian dan Analisis Performansi Metode *DiffServ* dan DS-TE pada MPLS

Menjelaskan tentang penerapan teknik DS-TE yang dibandingkan dengan penggunaan teknik DiffServ dalam pengiriman suatu trafik paket pada jaringan MPLS terhadap *throughput*, *packet loss*, dan *delay*.

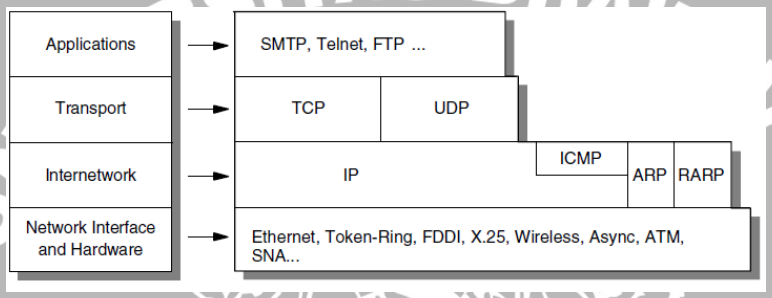
BAB VI Penutup

Memuat kesimpulan dan saran dari topik yang dianalisa dari skripsi ini.

BAB II
TCP/IP, *MULTIPROTOCOL LABEL SWITCHING* (MPLS) DAN MPLS *TRAFFIC ENGINEERING* (MPLS-TE)

2.1. Model Arsitektur TCP/IP

Protokol TCP/IP terbentuk dari 2 komponen yaitu *Transmission Control Protocol* (TCP) dan *Internet Protocol* (IP). Tujuan dari TCP/IP adalah untuk membangun suatu koneksi antar jaringan, di mana biasa disebut *internetwork*, atau *internet*, yang menyediakan pelayanan komunikasi antar jaringan yang memiliki bentuk fisik yang beragam.



Gambar 2.1. Detail Model Arsitektur TCP/IP
Sumber: *TCP/IP Tutorial and Technical Overview* (2001:8)

Aspek lain yang penting dari TCP/IP adalah membentuk suatu standarisasi dalam komunikasi. Tiap-tiap bentuk fisik suatu jaringan memiliki teknologi yang berbeda-beda, sehingga diperlukan pemrograman atau fungsi khusus untuk digunakan dalam komunikasi. TCP/IP memberikan fasilitas khusus yang bekerja diatas pemrograman atau fungsi khusus tersebut dari masing-masing fisik jaringan. Sehingga bentuk arsitektur dari fisik jaringan akan tersamarkan dari pengguna dan pembuat aplikasi jaringan. Dengan TCP/IP, pengguna tidak perlu lagi memikirkan bentuk fisik jaringan untuk melakukan sebuah komunikasi.

2.1.1. *Applications Layer*

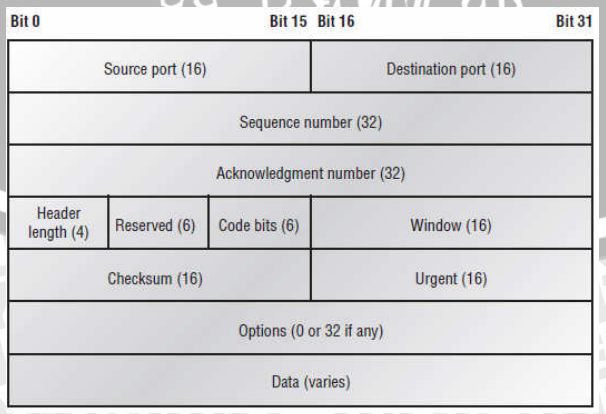
Layer aplikasi disediakan oleh program yang menggunakan TCP/IP untuk berkomunikasi. Sebuah aplikasi merupakan sebuah proses pengguna yang bekerja sama dengan proses lain yang berada pada host yang berbeda. Antarmuka yang digunakan untuk saling berkomunikasi adalah nomer *port* dan *socket*. Contoh protokol dan aplikasi antara lain Telnet, *File Transfer Protocol* (FTP), *Network File System* (NFS), *Simple*

Mail Transfer Protocol (SMTP), *Simple Network Management Protocol* (SNMP), *Domain Name Service* (DNS).

2.1.2. *Transport Layer*

Transport layer memberikan fungsi pengiriman data dari sebuah aplikasi secara *end-to-end* ke sisi *remote*. Protokol pada *layer transport* yang paling sering digunakan adalah *Transmission Control Protocol* (TCP), yang memberikan fungsi pengiriman data secara *connection oriented*, pencegahan duplikasi data, *congestion control* dan *flow control*. TCP mengambil sebuah blok besar informasi dari sebuah aplikasi dan kemudian memecahnya ke dalam segmen-segmen. Segmen-segmen tersebut diberi nomor urut agar dapat disatukan dan disusun kembali secara terurut saat sampai di tujuan. Gambar 2.2 menunjukkan format segmen TCP.

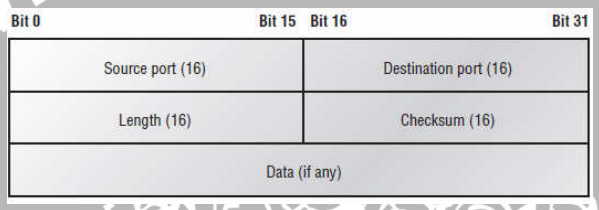
Source port merupakan nomor *port* aplikasi pada *host* yang mengirimkan data. *Destination port* adalah nomor *port host* tujuan yang meminta aplikasi. *Sequence number* adalah nomor yang digunakan TCP untuk mengurutkan data atau mentransmisikan kembali data yang hilang atau rusak. *Acknowledgement number* adalah oktet TCP berikutnya yang diharapkan. *Header length* adalah nomor 32 bit pada *header* TCP yang mengindikasikan di mana data dimulai. *Reserved* selalu di set nol. *Code bits* mengontrol fungsi yang digunakan untuk menginisialisasi dan melakukan terminasi sebuah *session*. *Window* merupakan ukuran window yang dapat diterima oleh pengirim. *Checksum* merupakan *cyclic redundancy check* (CRC) yang memeriksa *header* dan *data field*. *Urgent* akan berfungsi hanya jika *urgent pointer* pada *code bit* telah di *set*. *Options* dapat berupa 0 atau berbagai kombinasi dari 32 bit. Data diberikan oleh *layer* yang lebih tinggi, yang mengikutsertakan *header-header* layer yang lebih tinggi.



Gambar 2.2. Format Segmen TCP
Sumber: Todd Lammle (2007:75)

Protokol lainnya adalah *User Datagram Protocol* (UDP), yang memberikan fungsi pengiriman *connectionless* dan merupakan layanan *best effort*. UDP tidak mengurutkan segmen-segmen dan tidak memperdulikan urutan data yang tiba pada tujuan, serta tidak memeriksa data-data yang *error*. Karena inilah UDP disebut *unreliable protocol*. UDP banyak digunakan pada aplikasi yang membutuhkan kecepatan tinggi dan yang metoleransi kerusakan data.

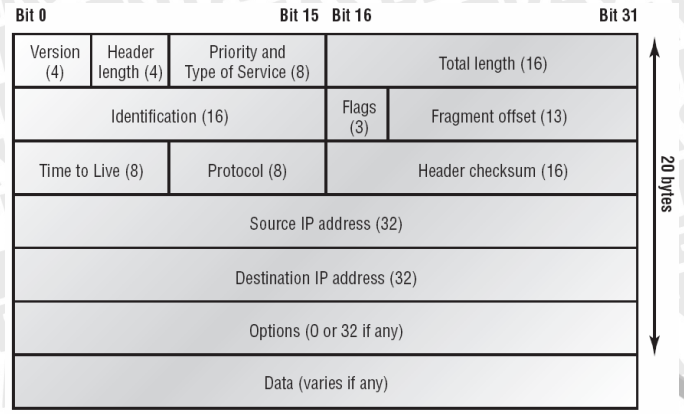
UDP memiliki panjang *header* yang lebih kecil dibandingkan dengan TCP. *Source port* merupakan nomor *port* aplikasi pada *host* yang mengirimkan data. *Destination port* merupakan nomor *port* aplikasi yang diminta oleh *host* tujuan. *Length* adalah panjang UDP *header* dan data UDP. *Checksum* merupakan *checksum* dari UDP *header* dan UDP *data field*. Data merupakan data dari *layer* yang lebih tinggi.



Gambar 2.3. Format Segmen UDP
Sumber: Todd Lammle (2007:76)

2.1.3. *Internetwork Layer*

Internetwork layer yang biasa disebut *internet layer* atau *network layer*, menyediakan "*virtual network*" pada internet. *Internet Protocol* (IP) adalah protokol yang paling penting. IP merupakan *connectionless protocol* yang tidak mengasumsikan realibilitas dari *layer* yang lebih rendah. IP tidak menyediakan realibilitas, *flow control*, atau *error recovery*. Fungsi ini harus disediakan oleh *layer* yang lebih tinggi. IP menyediakan fungsi *routing* untuk mengusahakan pengiriman pesan ke tujuannya. Sebuah unit pesan pada jaringan IP disebut dengan IP *datagram*. IP menerima segmen dari layer di atasnya dan kemudian melakukan fragmentasi terhadap segmen tersebut menjadi paket-paket apabila diperlukan. IP kemudian menyatukannya kembali ke dalam segmen pada sisi penerima. Masing-masing datagram diberikan alamat IP dari pengirim dan penerima. Masing-masing *router* yang menerima sebuah datagram akan membuat keputusan *routing* berdasarkan alamat IP tujuan paket.



Gambar 2.4. IP *Header*
Sumber: Todd Lammle (2007:84)

Gambar 2.4 menunjukkan IP *header*. *Version* merupakan nomor versi IP. *Header length* adalah panjang *header*. *Priority and type of service* (ToS) memberitahukan bagaimana datagram seharusnya ditangani. *Total length* adalah panjang paket termasuk *header* dan data. *Identification* adalah nilai paket IP yang unik. *Flags* menspesifikasikan apakah harus dilakukan fragmentasi. *Fragment offset* menyediakan fragmentasi dan *reassembly* jika paket terlalu besar untuk diletakkan ke dalam *frame* dan mengijinkan *Maximum Transmission Unit* (MTU) yang berbeda-beda di *internet*. *Time to live* di *set* ke dalam paket ketika paket dibuat. *Protocol* merupakan nomor *port* untuk protokol diatasnya yang juga mendukung *network layer protocol*, seperti ARP dan ICMP. *Header checksum* merupakan CRC hanya pada *header*. *Source* IP *address* adalah 32 bit IP address stasiun pengirim. *Destination* IP *address* adalah 32 bit IP *address* stasiun di mana paket tersebut ditujukan. *Options* digunakan untuk pengujian jaringan, *debugging*, *security*, dll. Data setelah IP *option field* akan merupakan data *layer* atas.

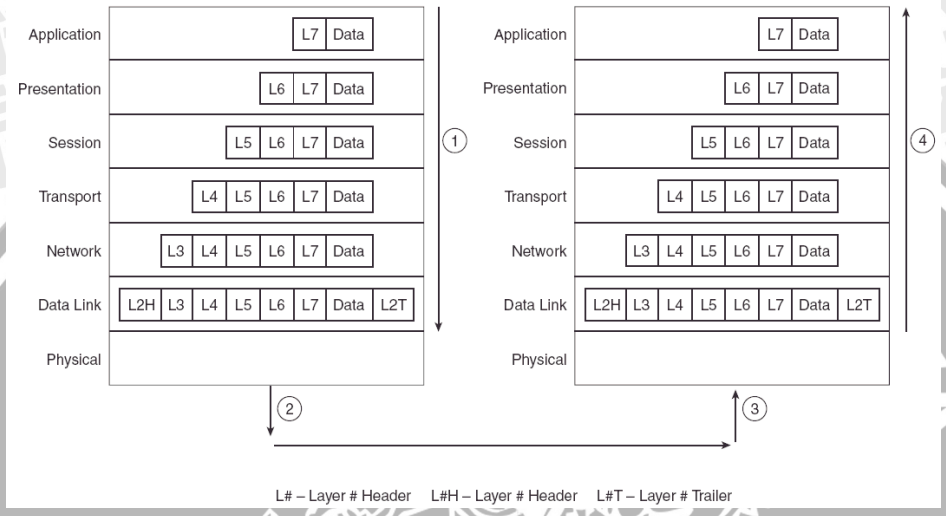
Protokol *internetwork* yang lainnya antara lain, *Internet Control Message Protocol* (ICMP), *Address Resolution Protocol* (ARP), *Reverse Address Resolution Protocol* (RARP).

2.1.4. *Network Interface Layer*

Network interface layer disebut juga *link layer* atau *data-link layer*, yang merupakan antarmuka menuju ke perangkat jaringan yang sebenarnya. Contohnya adalah IEEE 802.2, X.25, ATM, dan FDDI.

2.2. Enkapsulasi Data

Konsep penempatan data dibalik suatu *header* dan *trailer* untuk tiap *layer* disebut enkapsulasi. Pada Gambar 2.5 terlihat pada tiap *layer* diberikan suatu *header* tambahan, kemudian ditambahkan lagi *header* pada *layer* berikutnya, sedangkan pada *layer* 2 selain ditambahkan *header* juga ditambahkan *trailer*. Pada *layer* 1 tidak menggunakan *header* dan *trailer*.



Gambar 2.5. Komunikasi Antar Komputer pada OSI *Layer*
Sumber: Dhoto (2007:24)

Pada pemrosesan layer 5, 6 dan 7 terkadang tidak diperlukan adanya *header*. Ini dikarenakan tidak ada informasi baru yang perlu diproses. Sehingga untuk *layer-layer* tersebut bisa dianggap 1 proses. Sehingga langkah-langkah untuk melakukan data enkapsulasi dapat dijabarkan sebagai berikut:

1. Membuat data

   Sebuah aplikasi memiliki data untuk dikirim.
2. Memaketkan data untuk di transportasikan

   Pada *transport layer* ditambahkan *header* dan kemudian data ditempatkan dibalik header. Pada proses ini terbentuk L4PDU (Layer 4 Protocol Data Unit).
3. Menambahkan alamat tujuan *network layer* pada data

   Network layer membuat *network header*, di mana didalamnya terdapat juga alamat *network layer*, dan menempatkan L4PDU dibaliknya. Disini terbentuk L3PDU.

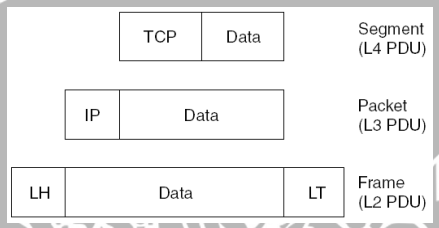
4. Menambahkan alamat tujuan *data link layer* pada data

   Data link layer membuat *header* dan menempatkan L3PDU dibaliknya, kemudian menambahkan *trailer* setelahnya. Disini terbentu L2PDU.

5. Mentransmisikan dalam bentuk bit

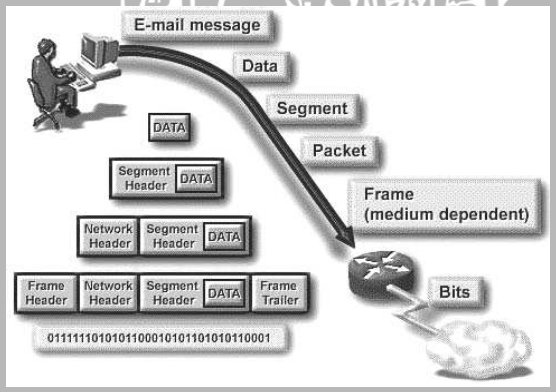
   Pada *physical layer*, dilakukan *encoding* pada sinyal kemudian dilakukan pengiriman *frame*.

Pada tiap *layer* terdapat LxPDU (Layer x *Protocol Data Unit*), di mana merupakan bentuk dari *byte* pada *header-trailer* pada data. Pada tiap-tiap *layer* juga terbentuk bentukan baru, pada layer 2 PDU termasuk *header* dan *trailer* disebut bentukan *frame*. Pada *layer* 3 disebut paket atau terkadang *datagram*. Sedangkan pada *layer* 4 disebut segmen. Sehingga dapat digambarkan pada Gambar 2.6.



Gambar 2.6. *Frame*, *Packet* dan Segmen
Sumber: Dhoto (2007:26)

Proses enkapsulasi pada contoh pengiriman e-*mail* dapat digambarkan seperti pada Gambar 2.7.



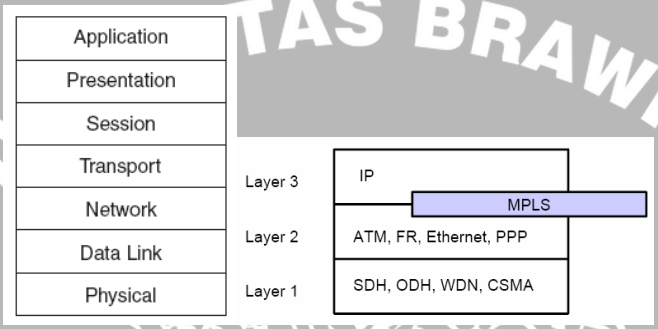
Gambar 2.7. Proses Enkapsulasi pada Pengiriman e-*mail*
Sumber: Dhoto (2007:27)

2.3. MPLS dan Model Referensi OSI

Multiprotocol Label Switching (MPLS), adalah arsitektur jaringan yang didefinisikan oleh *Internet Engineering Task Force* (IETF) untuk memadukan mekanisme pertukaran label di *layer* dua dengan *routing* di *layer* tiga untuk

mempercepat pengiriman paket. Arsitektur MPLS dirancang guna memenuhi karakteristik-karakteristik wajib dari sebuah jaringan kelas pembawa (*carrier*) berskala besar.

MPLS dapat mentransportasikan IPv4, IPv6, *Ethernet*, *High-Level Data Link Control* (HDLC), PPP, dan teknologi *layer* 2 yang lain. MPLS merupakan metode sederhana untuk melakukan *switching* terhadap *multiple protocol* dalam satu jaringan. Dibutuhkan sebuah *forwarding table* yang berisi *incoming label* untuk ditukar dengan *outgoing label* dan sebuah *next hop*. Model referensi *Open System Interconnection* (OSI) terdiri dari tujuh lapisan seperti yang ditunjukkan pada Gambar 2.8 dibawah ini.



Gambar 2.8. OSI *Layer* dan MPLS *Layer*
Sumber: Luc De Ghein (2007:28)

Lapisan (*layer*) yang paling bawah adalah lapisan pertama atau lapisan fisik, dan lapisan teratas adalah lapisan aplikasi. Di mana lapisan fisik merupakan lapisan yang berhubungan dengan pengkabelan, dan karakteristik elektris. Lapisan kedua adalah lapisan *data link* yang menangani format *frame*. Lapisan ketiga adalah lapisan jaringan (*network*) yang berhubungan dengan format paket *end to end*. Contoh protokol yang beroperasi pada lapisan ketiga ini adalah *Internet Protocol* (IP).

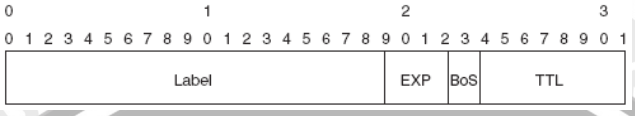
Pada lapisan-lapisan OSI tersebut, MPLS bukanlah merupakan protokol pada lapisan kedua karena enkapsulasi pada lapisan kedua masih ada dengan adanya paket yang berlabel. MPLS juga bukanlah protokol pada lapisan ketiga karena protokol pada lapisan ketiga juga masih ada. Dengan demikian, MPLS dapat diasumsikan sebagai protokol pada lapisan ke 2,5 seperti yang ditunjukkan oleh Gambar 2.9.

2.4. Arsitektur MPLS

Arsitektur MPLS terdiri dari label MPLS dan MPLS encoding membentuk struktur paket MPLS akan dijelaskan pada subsection dibawah ini.

2.4.1. Label MPLS

Satu label MPLS adalah sebuah *field* yang terdiri dari 32 bit dengan struktur tertentu. 20 bit pertama merupakan nilai label (*label value*). Nilai ini dapat berada diantara 0 dan 2^{20}-1 atau 1.048.575. Bit 20 sampai 22 merupakan *experimental* (EXP) *bit*. Bit-bit ini semata-mata digunakan untuk *Quality of Service* (QoS).



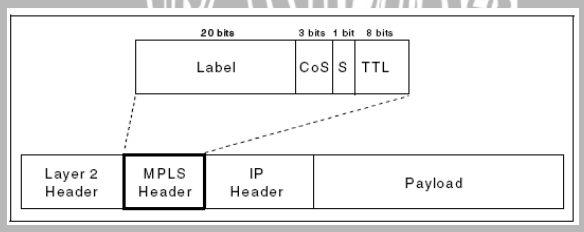
Gambar 2.9. MPLS Label
Sumber: Luc De Ghein (2007:25)

Bit ke 23 adalah bit *Bottom of Stack* (BoS). Nilainya adalah 0, kecuali ini merupakan label terbawah dari tumpukan (*stack*) label. Jika demikian maka bit ini akan bernilai 1. *Stack* adalah kumpulan dari label yang dapat ditemukan di atas paket. *Stack* dapat terdiri dari hanya 1 label atau dapat juga lebih.

Bit ke 24 sampai 31 merupakan delapan bit yang digunakan untuk *Time To Live* (TTL). TTL ini memiliki fungsi yang sama dengan TTL yang ada pada header IP. TTL akan dikurangkan dengan 1 pada tiap hop, dan fungsi utamanya adalah untuk mencegah sebuah paket menjadi terhambat dalam sebuah *routing loop*. Jika sebuah *routing loop* terjadi dan tidak ada TTL yang tersedia, maka paket akan berputar (*loop*) selamanya. Jika TTL dari label mencapai 0, maka paket akan dibuang (*discarded*).

2.4.2. MPLS *Encoding*

Label terletak didepan paket *layer* 3, yaitu sebelum *header* dari protokol transport, dan setelah *layer* 2 *header*.



Gambar 2.10. Enkapsulasi untuk Paket Berlabel
Sumber: *TCP/IP Tutorial and Technical Overview* (2001:617)

2.5. *Label Switch Router*

Label Switch Router (LSR) adalah sebuah *router* yang mendukung MPLS, mampu memahami label MPLS dan menerima serta mentransmisikan paket berlabel pada *data link*. Ada tiga macam LSR yang terdapat dalam jaringan MPLS:

a. *Ingress* LSR

Ingress LSR menerima sebuah paket yang belum dilabeli, kemudian ia menyisipkan label (*stack*) di atas paket, kemudian mengirimkannya ke *data link.*

b. *Egress* LSR

Egress LSR menerima paket berlabel, membuang label, dan mengirim paket tersebut ke *data link*. *Ingress* LSR dan *egress* LSR merupakan LSR tepi (*edge*).

c. *Intermediate* LSR

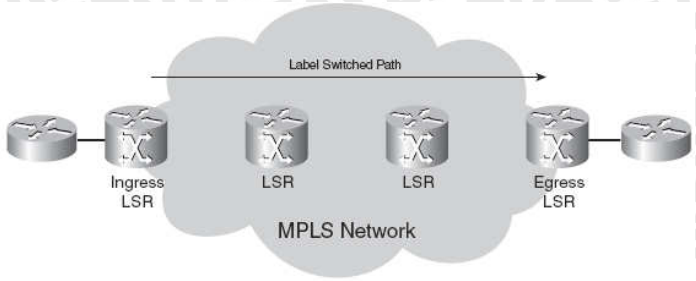
Intermediate LSR menerima paket berlabel yang datang, melakukan operasi kepadanya, melakukan *switch* terhadap paket, dan mengirimkan paket ke *data link* yang benar.

Sebuah LSR harus dapat melakukan tiga operasi, yaitu *pop*, *push* atau *swap*.

LSR harus dapat melakukan *pop* terhadap satu label atau lebih (membuang satu label atau lebih dari tumpukan label teratas) sebelum melakukan *switch* untuk mengeluarkan paket. Sebuah LSR juga harus dapat melakukan *push* terhadap satu label atau lebih ke dalam paket yang diterima. Jika paket yang diterima sudah dilabeli, maka LSR menambahkan (*push*) satu label atau lebih ke dalam *label stack* dan melakukan *switch* untuk mengeluarkan paket. Jika paket belum dilabeli, maka LSR membuat sebuah *label stack* dan menambahkannya (*push*) ke dalam paket. Sebuah LSR juga harus dapat melakukan *swap* terhadap sebuah label. Yang secara sederhana berarti bahwa ketika sebuah paket berlabel diterima, label yang berada pada tumpukan paling atas ditukar (*swapped*) dengan sebuah label baru kemudian paket tersebut di *switch* menuju *outgoing data link.*

2.6. *Label Switched Path*

Sebuah *Label Switched Path* **(**LSP) merupakan sederetan LSR yang melakukan *switch* terhadap paket berlabel melalui sebuah jaringan MPLS. Dasarnya, LSP merupakan jalur yang melalui jaringan MPLS atau bagian dari jalur yang ditempuh oleh paket. LSR pertama dari sebuah LSP adalah *ingress* LSR, sedangkan LSR terakhir dari sebuah LSP adalah *egress* LSR. Semua LSR diantara *ingress* LSR dan *egress* LSR adalah *intermediate* LSR. Pada Gambar 2.11, panah diatas mengindikasikan arah, karena sebuah LSP adalah *unidirectional*. Aliran dari paket yang berlabel dalam arah yang lain (dari kanan ke kiri) diantara *edge* LSR yang sama akan merupakan LSP yang lain.



Gambar 2.11. Sebuah LSP yang Melalui Jaringan MPLS
Sumber: Luc De Ghein (2007:30)

2.7. Distribusi Label

Ingress LSR yang bertugas menentukan label pertama pada paket. *Intermediate* LSR kemudian menukarkan label teratas dari paket berlabel yang diterima (*incoming label*) dengan label lain (*outgoing label*) dan mentransmisikan paket pada saluran *outgoing*. *Egress* LSR pada LSP membuang label LSP ini dan kemudian melewatkan paket.

Untuk melakukan hal ini, LSR yang berdekatan harus setuju terhadap label yang digunakan. Karena itulah, masing-masing intermediate LSR harus dapat mengetahui *outgoing* label mana yang dipergunakan untuk menggantikan *incoming* label. Label digunakan secara lokal oleh *router* yang berdekatan. Label tidak memiliki arti global diseluruh jaringan. Kesepakatan penggunaan label untuk prefiks tertentu antara *router* yang berdekatan ketika akan melewatkan paket memerlukan suatu bentuk komunikasi diantaranya. Bentuk komunikasi tersebut adalah suatu protokol distribusi label (*label distribution protocol*). Beberapa macam protokol distribusi label:

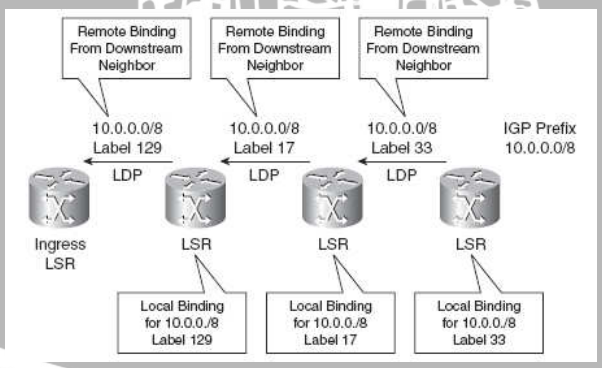
a. *Tag Distribution Protocol* (TDP)
b. *Label Distribution Protocol* (LDP)
c. *Resouce Reservation Protocol* (RSVP)

TDP yang telah mendahului LDP, merupakan protokol pertama untuk distribusi label yang dikembangkan dan diimplementasikan oleh Cisco. Kemudian IETF membentuk LDP. LDP dan TDP adalah sama dalam hal operasinya, tetapi LDP merupakan protokol yang lebih fungsional daripada TDP. Dengan kemampuan LDP yang tersebar luas, maka TDP dengan segera tergantikan oleh LDP. Distribusi label oleh RSVP hanya digunakan untuk MPLS TE saja dan akan dijelaskan pada sub bab MPLS *Traffic Engineering*.

2.8. *Label Distribution Protocol*

Setiap LSR membuat sebuah *local binding* untuk setiap prefiks IP IGP dalam tabel *routing*-nya, yang mengikat (*bind*) sebuah label pada prefiks IPv4. LSR kemudian mendistribusikan ikatan (*binding*) ini ke semua tetangga LDP-nya. Ikatan yang diterima ini kemudian menjadi *remote binding*. Tetangga LDP tersebut kemudian menyimpan *local* dan *remote binding* ini ke dalam tabel khusus, yaitu *Label Information Base* (LIB). Setiap LSR hanya memiliki satu *local binding* untuk tiap prefiks. Tetapi LSR dapat memiliki lebih dari satu *remote binding* karena LSR tersebut biasanya memiliki lebih dari satu LSR yang berdekatan.

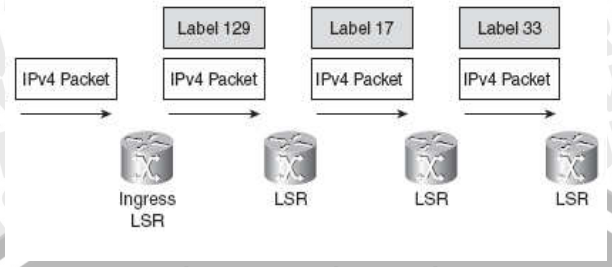
LSR harus mengambil hanya satu *remote binding* untuk sebuah prefiks, dan menggunakannya untuk menentukan *outgoing label* untuk IP prefiks tersebut. Tabel *routing* (kadang disebut dengan *Routing Instance Base* atau RIB) menentukan *next hop* tujuan untuk prefiks IPv4. LSR memilih *remote binding* yang diterima dari *downstream* LSR, yang merupakan *next hop* tujuan dari prefiks tersebut pada tabel *routing*. LSR menggunakan informasi ini untuk menginisialisasi *Label Forwarding Information Base* (LFIB)-nya di mana label dari *local binding* menjadi *incoming label* dan label dari salah satu *remote binding* yang dipilih melalui tabel *routing* menjadi *outgoing label*. Oleh karena itu, ketika sebuah LSR menerima sebuah paket berlabel, maka LSR dapat melakukan pertukaran (*swap*) pada *incoming label* yang diberikannya dengan *outgoing label* yang diberikan oleh *next-hop* LSR yang berdekatan. Gambar 2.12 menunjukkan penginformasian oleh LDP diantara LSR untuk prefiks IPv4 10.0.0.0/8.



Gambar 2.12. Sebuah Jaringan IPv4 *over* MPLS yang Menjalankan LDP
Sumber: Luc De Ghein (2007:35)

Gambar 2.13 menunjukkan paket IPv4 yang ditujukan untuk 10.0.0.0/8 memasuki jaringan MPLS pada *ingress* LSR, di mana paket tersebut diberi label 129 dan kemudian di-*switch* menuju LSR berikutnya. LSR yang kedua menukar (*swap*)

incoming label 129 dengan *outgoing label* 17 dan kemudian melewatkan paket tersebut menuju LSR ketiga. LSR ketiga menukar *incoming label* 17 dengan *outgoing label* 33 dan melewatkan paket ke LSR yang berikutnya dan seterusnya.



Gambar 2.13. *Packet Switching*
Sumber: Luc De Ghein (2007:35)

2.9. *Label Forwarding Instance Base*

Label Forwarding Instance Base (LFIB) adalah sebuah tabel yang digunakan untuk melewatkan paket berlabel. LFIB dibentuk oleh *incoming label* dan *outgoing label* untuk LSP. *Incoming label* adalah label dari *local binding* pada LSR tertentu. *Outgoing label* adalah label dari *remote binding* yang dipilih oleh LSR dari semua kemungkinan *remote binding*. Semua *remote binding* ini ditemukan pada LIB. LFIB memilih hanya satu kemungkinan *outgoing label* dari semua kemungkinan *remote binding* pada LIB dan kemudian meng-*install*-nya ke dalam LFIB. *Remote binding* yang dipilih tergantung pada jalur (*path*) terbaik yang ditemukan dalam tabel *routing*.

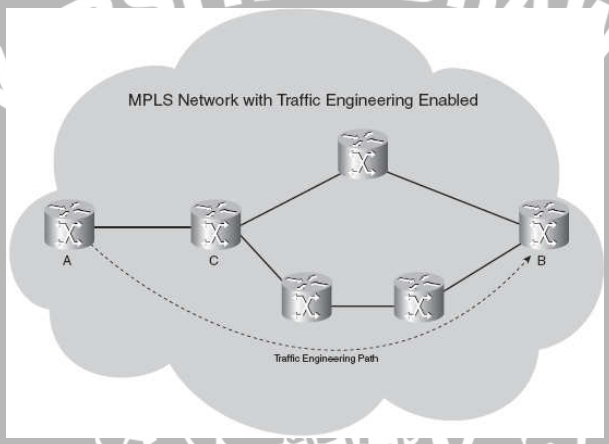
2.10. Rekayasa Trafik (*Traffic Engineering*)

Traffic engineering (TE) memanipulasi trafik agar sesuai dengan jaringan. TE pada intinya adalah memindahkan trafik sehingga trafik dari *link* yang memilki *congestion* dipindahkan ke *link* yang sedang tidak digunakan.

Ide dasar dibalik TE adalah untuk mengoptimalkan penggunaan infrastruktur jaringan, termasuk saluran-saluran (*links*) yang jarang digunakan (*underutrilized*), karena saluran-saluran tersebut merupakan jalur yang kurang dipilih. Hal ini berarti bahwa TE harus menyediakan kemungkinan untuk mengemudikan trafik melewati jaringan pada jalur yang berbeda dengan jalur yang bisanya dipilih (*preffered path*) yang selama ini menjadi pilihan karena merupakan *least-cost path* yang disediakan oleh IP *routing*. *Least-cost path* adalah jalur terpendek yang dihitung oleh *dynamic routing protocol*. Dengan implementasi dari TE dalam jaringan MPLS, trafik yang ditujukan dengan sebuah prefiks tertentu atau sebuah *quality of service* tertentu dapat dialirkan

dari A menuju B melalui sebuah jalur yang berbeda dengan jalur *least-cost*. Hasilnya adalah trafik tersebut dapat disebarkan secara merata melewati saluran yang tersedia dalam jaringan dan dapat memberdayakan saluran-saluran yang jarang terpakai dalam jaringan dengan lebih baik.

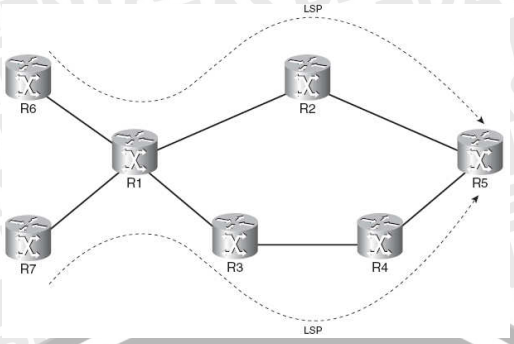
Pada Gambar 2.14, operator jaringan MPLS dengan TE dapat mengemudikan trafik dari A menuju B melalui jalur bawah, yang bukan merupakan jalur terpendek antara A dan B (4 *hop* melawan 3 *hop* pada jalur atas). Dengan demikian dapat dikirimkan trafik melalui saluran yang tidak terlalu banyak digunakan. Trafik dalam jaringan ini dapat dipandu untuk melewati jalur bawah dengan mengubah *routing protocol metric*.



Gambar 2.14. *Traffic Engineering*
Sumber: Luc De Ghein (2007:18)

2.11. MPLS *Traffic Engineering* (MPLS TE)

MPLS TE menjalankan sebuah rencana di mana *head end router* dari sebuah LSP menghitung rute yang paling efisien melalui seluruh jaringan menuju ke *tail end router* dari sebuah LSP. *Head end router* dapat melakukan hal ini apabila ia memiliki topologi jaringan. Selanjutnya, *head end router* perlu mengetahui *bandwidth* yang tersisa dari semua saluran pada jaringan. Untuk dapat membuat sebuah LSP *end to end* maka MPLS perlu diaktifkan disetiap *router*. Hal itu karena MPLS melakukan *forwarding* dengan mencocokkan sebuah *incoming label* pada LFIB dan kemudian menukarnya dengan sebuah *outgoing label*. Oleh karena itu, *head end* LSR dari sebuah LSP dapat menentukan *routing* dari paket berlabel, setelah semua LSR menyetujui label yang akan digunakan untuk sebuah LSP. Gambar 2.15 menunjukkan sebuah kemampuan *source-based routing* dari sebuah MPLS TE.



Gambar 2.15. MPLS TE *Head End Router*
Sumber: Luc De Ghein (2007:251)

Diasumsikan bahwa R6 dan R7 ingin mengirim trafik menuju R5. Jika jaringan ini hanya menjalankan IP *forwarding* saja, maka trafik ini akan mengikuti jalur R1-R2-R5 saja. Hal ini berarti *forwarding* IP *packet* dilakukan secara independen pada setiap *hop* dalam jaringan sesuai dengan IP *routing* tabelnya. R6 mungkin ingin mengirimkan trafik pada sepanjang jalur R6-R1-R2-R5, sedangkan R7 mungkin ingin melewatkan trafik disepanjang jalur R7-R1-R3-R4-R5, yang merupakan suatu hal yang tidak mungkin dilakukan pada sebuah jaringan IP saja. Jika sebuah jaringan menjalankan MPLS, maka dapat di atur dua jalur sebagai dua LSP yang berbeda jadi digunakan label-label yang berbeda. Pada *router* R1, nilai *incoming* label yang berbeda mengindikasikan apakah suatu paket berasal dari R6 sebagi *head end* atau R7 sebagai *head end*. R1 kemudian meneruskan paket melalui salah satu dari dua LSP, dan tidak meneruskan paket sesuai dengan keinginannya sendiri seperti yang dilakukan pada IP *forwarding*.

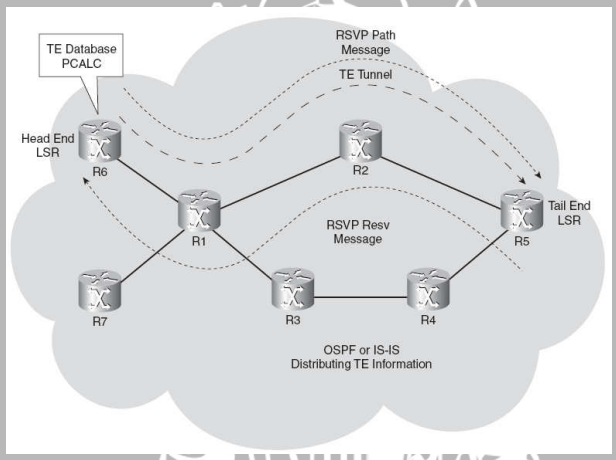
MPLS TE dapat diterapkan di jaringan apa saja yang memiliki LSR. Tetapi, karena *bandwidth* dan atribut-atribut jaringan yang lain harus diketahui juga oleh *head end* LSP, maka protokol *routing* yang dipergunakan diantara LSP *endpoint* (*head end* dan *tail end* LSR) harus merupakan *link state routing protocol*. Dengan sebuah *link state routing protocol*, masing-masing *router* membangun sebuah keadaan (*state*) dari salurannya sendiri, yang kemudian di sebarkan (*flooded*) ke semua *router* pada area yang sama. Hal ini berarti bahwa semua *router* pada area tersebut memiliki semua informasi topologi pada area tersebut. Kemudian *head end* LSR dapat menggunakan informasi tersebut untuk membuat MPLS LSP yang direkayasa trafiknya. LSP ini disebut dengan MPLS TE *tunnel*. Sebuah TE *tunnel* adalah satu arah (*unidirectional*).

2.12. Operasi MPLS TE

Berikut ini adalah hal-hal yang dibutuhkan oleh MPLS TE agar dapat bekerja:

a. *Link constraints* (seberapa banyak trafik yang dapat ditangani oleh setiap saluran).
b. Distribusi informasi TE (oleh MPLS TE yang mengaktifkan *link state routing protocol*).
c. Sebuah algoritma (*Path Calculation* atau PCALC) untuk menghitung jalur terbaik dari *head end* LSR menuju sebuah *tail end* LSR.
d. Sebuah protokol pensinyalan (*Resource Reservation Protocol* atau RSVP) untuk mensinyali TE *tunnel* disepanjang jaringan.
e. Sebuah cara untuk meneruskan trafik ke dalam TE *tunnel*.

Gambar 2.16 menunjukkan gambar TE *building block* dari Gambar 2.16. Sebuah TE *tunnel* diperpanjang dari R6 menuju R5.



Gambar 2.16. MPLS TE *Building Blocks*
Sumber: Luc De Ghein (2007:253)

Sumber daya (*resource*) ini adalah *bandwidth* saluran dan beberapa atribut saluran yang dispesifikasikan oleh operator. Atribut ini dikonfigurasi pada saluran dan kemudian di *advertise* oleh *link state protocol* (OSPF atau IS-IS). Protokol OSPF diperpanjang untuk membawa informasi ini.

Sebuah TE *database* dibangun dari informasi TE yang dikirimkan oleh *link state protocol*. Basis data ini berisi semua saluran yang mendukung MPLS TE dan karakteristik atau atributnya. Dari basis data MPLS TE ini, *path calculation* (PCALC) menghitung jalur terpendek yang masih sesuai karakteristik dengan semua *constraint* (yang paling penting adalah *bandwidth*) dari *head end* LSR menuju *tail end* LSR.

PCALC merupakan sebuah algoritma *Shortest Path First* (SPF) yang dimodifikasi untuk MPLS TE, sehingga *constraints* dapat diikutsertakan juga dalam perhitungan. PCALC mencocokkan kebutuhan *bandwidth* dan atribut TE *tunnel* dengan sebuah saluran, dan dari semua kemungkinan jalur akan diambil salah satunya. Perhitungan hanya dilakukan pada *head end* LSR.

Intermediate LSR pada sebuah LSP juga perlu mengetahui *incoming* dan *outgoing* label untuk TE *tunnel* LSP tersebut. *Intermediate* LSR hanya dapat mempelajari label jika head end router dan intermediate LSR mensinyali label dengan sebuah protokol pensinyalan. Pada IETF, dicapai sebuah konsensus yang mengembangkan RSVP sebagai protokol pensinyalan untuk MPLS TE. Maka dalam pensinyalan MPLS TE ini dipakai RSVP sebagai protokol pensinyalannya.

Dibuat perpanjangan atau ekstensi pada RSVP untuk membawa informasi label MPLS dan spesifikasi TE yang lain, seperti objek *Explicit Route* dan *Record Route*. Intinya, RSVP mencoba untuk mensinyali TE *tunnel* disepanjang jalur (dari *head end* LSR menuju *tail end* LSR) yang merupakan hasil dari perhitungan yang berdasarkan basis data TE pada *head end* LSR. Pesan RSVP PATH dikirim dari *head end* LSR menuju *tail end* LSR dan membawa sebuah permintaan (*request*) sebuah label MPLS. Pesan RSVP RESV dikirim kembali dari *tail end* LSR menuju ke *head end* LSR dengan membawa label MPLS yang dapat dipergunakan oleh LSR disepanjang TE *tunnel* LSP untuk meneruskan trafik. RSVP juga menguji apakah TE *tunnel* dengan *constraint* dapat diinisialisasi pada masing-masing simpul.

Mungkin saja dapat terjadi suatu kejadian di mana TE *tunnel* yang lain baru saja memesan sejumlah *bandwidth* pada sebuah saluran dari sebuah *intermediate* LSR, dan OSPF belum memberitahukan hal ini. Karena itulah, dimungkinkan *bandwidth* sisa pada saluran tersebut tidak mencukupi untuk inisialisasi TE *tunnel* ini, disinilah yang menjadi kebutuhan akan sebuah protokol pensinyalan untuk memastikan *bandwidth* sudah dipesan pada setiap *hop*.

Explicit Route Object (ERO) berisi daftar *hop* yang harus diikuti oleh pesan RSVP PATH untuk mensinyali TE *tunnel*. Sederetan *hop* atau jalur merupakan hasil dari perhitungan jalur pada *head end router*. Pada masing-masing *hop*, pesan PATH ini memesan *bandwidth* secara *temporary* atau sementara dan meminta sebuah label. Akhirnya, pesan PATH sampai pada *tail end* LSP, yang kemudian mengembalikan pesan RESV kembali ke *head end* LSP. Pesan RESV ini kemudian membawa label

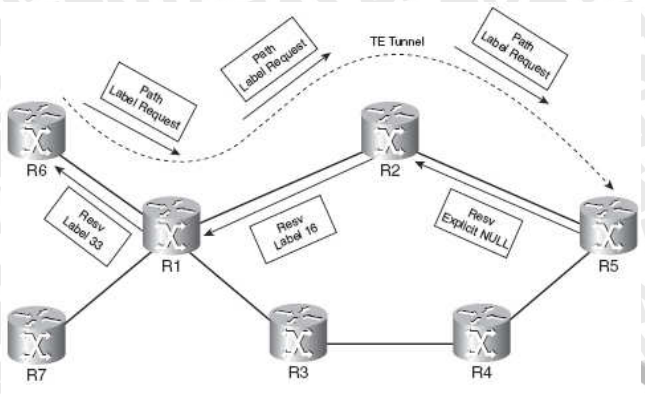
yang dapat digunakan oleh MPLS *data plane* untuk meneruskan paket MPLS TE *tunnel* disepanjang LSP.

2.13. *Resource Reservation Protocol*

RSVP menggunakan pesan PATH dan RESV untuk melakukan pensinyalan pada suatu jalur (*path*). TE *head end router* mengirimkan pesan PATH pada *tail end router*, sementara pesan RESV mengambil jalur yang sama tetapi berlawanan arah kembali menuju *head end router*. *Head end router* pada sebuah TE *tunnel* menghitung jalur terbaik yang sebaiknya dilalui dari TE *database*, dengan mempertimbangkan *bandwidth* dan batas (*constraint*) yang lain. Sebagai alternatif, jalur didefinisikan oleh sebuah pilihan jalur yang dikonfigurasi oleh *user* pada antarmuka *tunnel*. Pada kasus lain, *head end router* mengetahui secara pasti jalur yang harus dilalui oleh TE *tunnel*. Tiap *hop* (LSR) yang harus dilewati oleh TE *tunnel* dicatat dan disimpan dalam *Explicit Route Object* (ERO), yang merupakan daftar terurut dari antarmuka alamat IP, dengan sebuah alamat IP untuk tiap LSR. Pesan PATH dikirim dari *head end router* menuju *next-hop router*. *Next-hop router* ini membuang alamat IP-nya sendiri dari ERO, kemudian memeriksa alamat IP berikutnya dan mengirimkan pesan PATH menuju *next-hop*. Hal ini berlangsung terus menerus sampai *tail end router* pada TE *tunnel* menerima pesan PATH. Ketika pesan PATH diterima, *tail end router* memberikan pesan RESV (*reservation*) disepanjang jalur yang sama dengan yang dilewati pesan PATH, tetapi dalam arah yang berlawanan. Jika *head end* menerima pesan RESV tanpa adanya *error*, maka akan dibentuk sebuah jalur dan dilakukan pensinyalan pada jalur tersebut.

2.13.1. RSVP dan Label

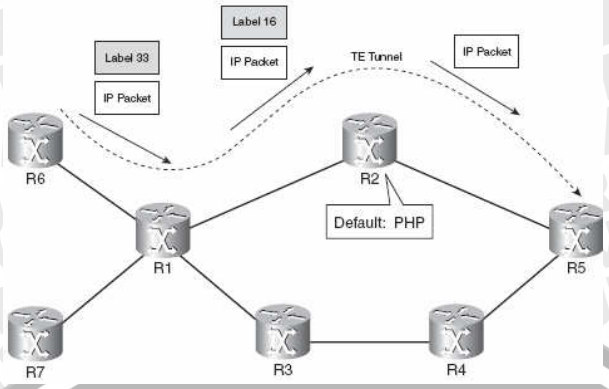
Selain mensinyali jalur untuk TE *tunnel*, RSVP juga memiliki tugas untuk membawa label MPLS sehingga paket-paket dapat ditukar labelnya (*label-switched*) disepanjang jalur TE *tunnel*. Gambar 2.17 menunjukkan pesan RSVP yang dikirim dalam pensinyalan TE *tunnel*.



Gambar 2.17. RSVP untuk TE dan Label
Sumber: Luc De Ghein (2007:280)

Pesan PATH membawa objek *Label Request*. Ketika *tail end router* menerima *Label Request* ini, ia memberikan sebuah label pada TE *tunnel* LSP dan meneruskannya menuju *upstream router* (*penultimate hop router*) dalam sebuah objek Label dalam pesan RESV. Label ini adalah *incoming label* dalam LFIB dari *tail end router*. *Upstream router* menerima label dari *tail end router* dan meletakkan label ini sebagai *outgoing label* dalam LFIB untuk TE tunnel LSP ini. *Router* memberikan sebuah label dari *global label table* pada TE tunnel LSP ini dan mengirimkannya dalam sebuah objek Label dalam pesan RESV menuju *upstream router*. Label ini menjadi *incoming label* dalam LFIB untuk TE *tunnel* LSP ini. Hal ini berlangsung terus menerus seperti ini sampai pesan RESV mencapai *head end router* dari TE tunnel LSP.

Label yang diberikan oleh *tail end router* adalah *explicit* NULL *label* dalam objek label RESV. Jadi secara *default*, paket yang dikirimkan dalam sebuah TE *tunnel* dari *penultimate hop router* menuju *tail end router* tidak memiliki label atau label yang paling atas telah dibuang (*popped off*). Jadi, terjadilah *Penultimate Hop Popping* (PHP). Gambar 2.18 menunjukkan penerusan paket (*packet forwarding*) dalam TE *tunnel*.



Gambar 2.18. Penerusan Paket
Sumber: Luc De Ghein (2007:281)

2.14. *Open Short Path First* (OSPF)

OSPF merupakan *routing protocol* berbasis *link state*, termasuk dalam *Interior Gateway Protocol* (IGP). Menggunakan algoritma Djikstra untuk menghitung *Shortest Path First* (SPF). Menggunakan *cost* sebagai *routing metric*. Setelah antar *router* bertukar informasi maka akan terbentuk *database link state* pada masing-masing *router*. *Router* dalam *broadcast domain* yang sama akan melakukan *adjacencies* untuk mendeteksi satu sama lainnya. Pendeteksian dilakukan dengan mendengarkan "*Hello Packet*".

2.14.1. OSPF Area

Jaringan OSPF terbagi ke dalam beberapa area. Area dapat bertepatan dengan *geographic* atau *administrative boundaries*. Masing-masing area diberikan 32 bit area ID. Dengan membagi-bagi jaringan, menghasilkan beberapa keuntungan sebagai berikut:

a. Dalam sebuah area, masing-masing *router* memelihara sebuah basis data topologi (*topology database*) yang identik yang menjelaskan *routing device* dan saluran-saluran dalam sebuah area. *Router* tersebut hanya sadar (*aware*) akan tujuan eksternalnya. Hal ini mengurangi ukuran basis data topologi yang dipelihara oleh tiap *router*.
b. Area membatasi kemungkinan meledaknya pertumbuhan jumlah *link state update*. Kebanyakan *Link State Advertisement* (LSA) didistribusikan hanya dalam sebuah area.
c. Area mengurangi pemrosesan CPU yang dibutuhkan untuk memelihara *topology database*. Algoritma SPF dibatasi untuk mengelola perubahan dalam suatu area.

Semua jaringan OSPF setidaknya terdiri dari sebuah area. Area dikenal sebagai area 0 atau *backbone* area. Area tambahan dapat dibuat berdasarkan pada topologi jaringan atau desain lain yang dibutuhkan.

Dalam jaringan yang terdiri dari banyak area, *backbone* secara fisik terhubung dengan semua area lain. OSPF mengharapkan semua area untuk menyiarkan informasi *routing* secara langsung ke *backbone*. *Backbone* kemudian makan mengumumkan informasi ini ke area lain.

2.14. 2. *Neighbor*

Neighbor adalah dua *router* atau lebih yang memiliki sebuah antarmuka pada sebuah jaringan yang umum.

2.14.3. Tipe Jaringan Fisik

OSPF mengelompokkan segmen jaringan ke dalam tiga tipe. Tipe komunikasi yang terjadi antara OSPF *device* yang terkoneksi dengan jaringan-jaringan ini dipengaruhi oleh tipe jaringan:

a. *Point to point*

   Jaringan ini menghubungkan dua *router* secara langsung.

b. *Multi access*

   Jaringan *multi access* mendukung hubungan untuk lebih dari dua *router*. Jaringan ini terbagi lagi menjadi dua bagian:

   - Jaringan *broadcast* memiliki kemampuan untuk secara simultan mengarahkan sebuah paket ke semua *router* yang terhubung.
   - Jaringan *non broadcast* tidak memiliki kemampuan *broadcast*. Masing-masing paket harus secara khusus dialamatkan ke tiap *router* dalam jaringan.

c. *Point to multipont*

   Jaringan ini merupakan kasus yang khusus untuk jaringan multi akses *non broadcast network*. Dalam sebuah *point to multipoint network*, sebuah *device* tidak perlu memiliki hubungan langsung ke setiap *device* lain.

2.14.4. *Adjacency*

Router yang berbagi sebuah segmen jaringan yang umum membangun sebuah hubungan pertetanggaan (*neighbor relationship*) pada suatu segmen. Sekali sebuah

router menjadi *neighbor*, sebuah hubungan *adjacency* dapat dibentuk diantara *device-device*. *Neighboring router* dapat disebut berdekatan (*adjacent*) ketika mereka telah mensinkronkan basis data topologinya. Hal ini terjadi melalui pertukaran informasi *link state*.

2.14.5. Basis Data Keadaan Saluran (*Link State Database*)

Link state database disebut juga dengan *topology database*. Berisi tentang *link state advertisement* yang mendeskripsikan jaringan OSPF dan koneksi eksternal. Tiap *router* dalam suatu area memelihara sebuah salinan yang identik dari *link state database*.

2.14.6. *Link State Advertisement* dan *Flooding*

Isi dari sebuah *Link State Advertisement* (LSA) mendeskripsikan sebuah komponen jaringan individual (yaitu *router*, segmen, atau *eksternal destination*). LSA dipertukarkan antar *router* OSPF yang berdekatan. Hal ini dilakukan untuk mensinkronkan *link state database* pada masing-masing *device*.

Ketika sebuah *router* menghasilkan atau memodifikasi LSA, *router* tersebut harus mengkomunikasikan perubahan ini ke seluruh jaringan. *Router* memulai proses ini dengan meneruskan LSA ke tiap *adjacent device*. Setelah menerima LSA, *neighbor* akan menyimpan informasi ini ke *link state database*-nya dan mengkomunikasikan LSA ke *neighbor* mereka. Penyimpanan dan penerusan ini berlangsung terus menerus sampai semua *device* menerima *update*. Proses ini disebut *reliable flooding*.

BAB III
MPLS *DIFFERENTIATED SERVICE* (MPLS DiffServ) DAN MPLS *DIFFERENTIATED SERVICE-AWARE TRAFFIC ENGINEERING* (MPLS DS-TE)

3.1. *Quality of Service*

Quality of Service (QoS), merupakan kemampuan untuk menyediakan prioritas yang berbeda pada aplikasi, pengguna, maupun aliran data yang berbeda. Jaminan QoS sangat diperlukan sehubungan dengan keterbatasan kapasitas jaringan. QoS adalah satu set teknik yang mengatur hal-hal berikut:

a. *Bandwidth* jaringan

   Trafik yang tidak penting dicegah untuk menggunakan *bandwidth* yang diperlukan oleh aplikasi yang lebih penting. Penyebab utama dari kepadatan adalah kurangnya *bandwidth*.

b. *Network Delay* (*latency*)

   Waktu yang dibutuhkan untuk memindahkan sebuah paket dari sumber ke tujuan dalam suatu saluran.

c. *Jitter*

   Merupakan variasi *delay* antar paket yang merupakan variasi kedatangan antar paket.

d. *Packet Loss*

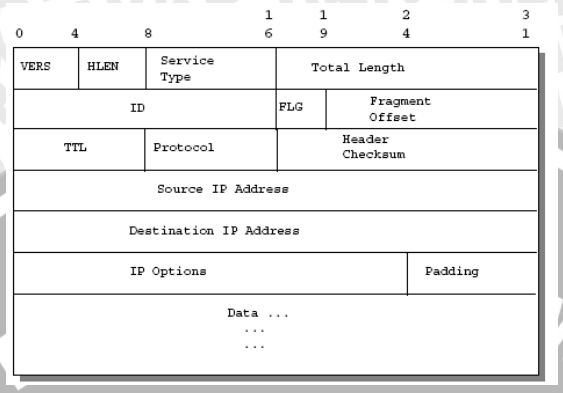
   Merupakan hilangnya paket.

3.2. *Differentiated Service* dan Paket IP

DiffServ menyediakan diferensiasi layanan, dengan membagi trafik dalam kelas-kelas, dan memperlakukan setiap kelas secara berbeda. Identifikasi kelas dilakukan dengan menambahkan kode DiffServ, yang disebut dengan *DiffServ Code Point* (DSCP), ke dalam paket IP. Dengan cara ini, klasifikasi paket melekat pada paket, dan dapat diakses tanpa perlu protokol persinyalan tambahan. Dalam hal ini DiffServ hanya beroperasi pada *layer* tiga saja dan tidak berhubungan dengan *layer* yang lebih rendah. DiffServ menggunakan bit DiffServ pada IP *header* untuk menandai (*mark*), mengantrikan (*queue*), *shape* dan menentukan *drop precedence* pada paket.

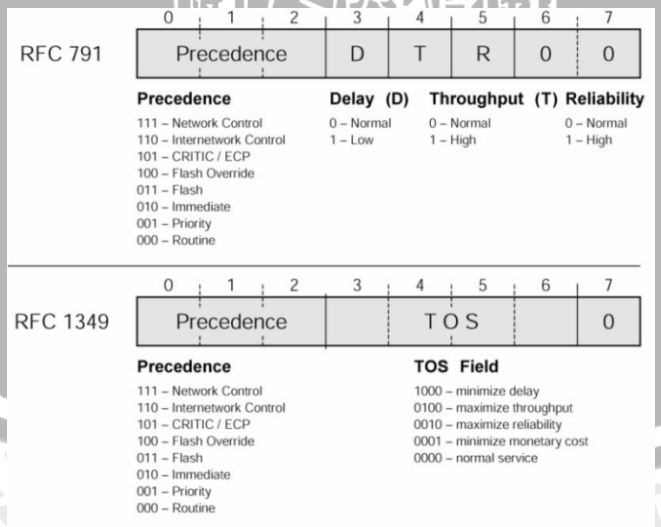
3.2.1. *Differentiated Service Code Point*

Spesifikasi protokol IP sebelumnya telah menyediakan bit-bit *header* untuk tujuan QoS, yaitu pada field *Type of Service* (ToS) atau *Service Type*. Hal ini ditunjukkan pada Gambar 3.1.



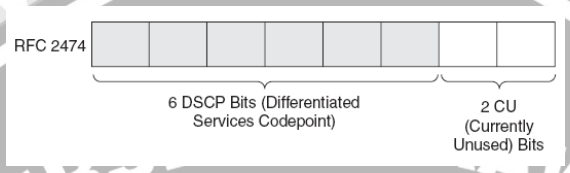
Gambar 3.1. IPv4 Datagram *Header*
Sumber: Adolfo Rodriguez (2001:91)

Spesifikasi IP versi 4 pada RFC 791 mendefinisikan *header* oktet kedua sebagai oktet *Service Type* atau ToS. Tiga bit yang berikutnya pada oktet ToS mengindikasikan *delay*, *throughput*, dan kebutuhan reliabilitas. Dua bit yang terakhir (*least significant bit*) tidak didefinisikan dan di *set* sebagai nol. Pada RFC 1349 diperkenalkan sebuah perubahan kecil dalam oktet ToS dengan mendefinisikan sebuah *field* yang memasukkan sebuah bit tambahan. Sehingga sekarang ToS terdiri dari empat bit.



Gambar 3.2. Oktet ToS pada IPv4
Sumber: Santiago Alvarez (2006)

Penggunaan *precedence bits* untuk QoS digunakan luas di berbagai jaringan diseluruh dunia. Karena bit-bit *precedence* hanya terdiri dari tiga bit, maka hanya bisa dibentuk delapan *level* layanan. Karena itulah, IETF memutuskan untuk mendedikasikan lebih banyak bit untuk QoS. Maka bit-bit *precedence* tersebut ditambahkan dengan tiga bit dari bit-bit ToS menjadi DiffServ *field*. Maka sekarang DiffServ terdiri dari enam bit yang jumlahnya lebih dari cukup untuk menyediakan *level* layanan QoS. Hal ini ditunjukkan oleh Gambar 3.3.



Gambar 3.3. DiffServ *Field*
Sumber: Santiago Alvarez (2006)

3.2.2. *Per-Hop Behaviour*

Per-Hop Behaviour (PHB) merupakan *forwarding behaviour* yang dilakukan oleh simpul terhadap suatu aliran trafik. PHB merepresentasikan sebuah deskripsi dari karakteristik *latency*, *jitter*, atau *loss* yang akan dialami trafik ketika melewati sebuah simpul DiffServ.

Simpul DiffServ memetakan paket-paket ke dalam PHB berdasarkan DSCP-nya. Tabel dibawah menunjukkan pemetaan yang direkomendasikan oleh arsitektur ini.

Tabel 3.1. Pemetaan PHB dan DSCP yang Direkomendasikan IETF

PHB	DSCP (Desimal)	DSCP (Biner)
EF	46	101110
AF43	38	100110
AF42	36	100100
AF41	34	100010
AF33	30	011110
AF32	28	011100
AF31	26	011010
AF23	22	010110
AF22	20	010100
AF21	18	010010
AF13	14	001110
AF12	12	001100
AF11	10	001010
Default	0	000000

Sumber: Santiago Alvarez (2006)

Sekelompok PHB yang merupakan bagian dari standar DiffServ yang antara lain adalah *Expedited Forwarding* (EF), *Assured Forwarding* (AF), *Class Selector* (CS), dan *Default*. *Class Selector* (CS) adalah pengecualian, karena didefinisikan sebagai

kompatibilitas dengan penggunaan *field precedence* pada oktet ToS IPv4. Tabel pemetaan paket ke dalam PHB untuk CS dapat dilihat pada Lampiran 7.

3.2.3. *Expedited Forwarding* (EF)

EF PHB meminimalisasi *latency*, *jitter*, *loss* yang mungkin terjadi pada simpul DiffServ. PHB ini berlaku sebagai penjamin transportasi trafik *real time*. Trafik yang melebihi batasan trafik yang ditentukan oleh *policy* akan dibuang.

3.2.4. *Assured Forwarding* (AF)

AF dalam grup PHB menyediakan empat *level* jaminan *forwarding* yang didukung oleh simpul DiffServ. Dengan kata lain, kelompok ini mendefinisikan bagaimana simpul DiffServ mendukung jaminan *packet loss* yang berbeda. Kelompok PHB AF memiliki nama AF1, AF2, AF3, dan AF4. Masing-masing kelompok mendukung tiga *level drop-precedence*. Semakin tinggi *drop-precedence*, maka simpul DiffServ akan langsung membuang paket jika terjadi kejenuhan terhadap *resource* yang dialokasikan (*bandwidth* dan *buffer*). Tabel 3.2 menunjukkan daftar PHB dengan masing-masing *drop precedence*-nya. Kelompok AF tidak memiliki karakterisasi *latency* atau *jitter*.

Tabel 3.2. PHB AF dan *Drop Presedance Class Selector*

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

Sumber: Santiago Alvarez (2006)

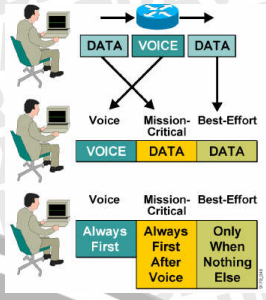
3.2.5. *Default* PHB

Domain DiffServ menyediakan *default* PHB yang menawarkan layanan *best-effort*. Dalam hal ini, domain DiffServ akan melewatkan paket sebanyak mungkin dan secepat kemampuannya. Tidak ada karakterisasi *latency*, *jitter* dan *loss*.

3.3. Mengimplementasikan IP QoS

Untuk mengimplementasikan sebuah QoS, membutuhkan tiga langkah. Hal ini ditunjukkan oleh Gambar 3.4.

1. Mengidentifikasi trafik dan kebutuhannya.
2. Membagi trafik dalam kelas-kelas (klasifikasi).
3. Mendefinisikan *policy* QoS untuk tiap kelas.



Gambar 3.4. Tiga Langkah untuk Membangun QoS
Sumber: *Implementing Cisco Quality of Service (QoS) v2.1* (2004:1-20)

Dalam sebuah jaringan yang umum, suara (*voice*) akan selalu membutuhkan *delay* yang minimum. Beberapa data yang diasosiasikan dengan aplikasi-aplikasi penting akan membutuhkan *delay* yang sangat kecil, contohnya transaksi berbasis data yang digunakan dalam pemesanan tiket pesawat terbang atau aplikasi perbankan yang *online*. Tipe-tipe data yang lainnya dapat mentoleransi *delay* yang besar, contohnya *file transfer* dan *e-mail*. Sedangkan untuk penjelajahan nonbisnis dalam jaringan, dapat mengalami *delay* atau bahkan dilarang. Selanjutnya sebuah pemetaan satu-satu antara kelas-kelas trafik dan *policy-policy* QoS perlu dilakukan.

3.3.1. Mengidentifikasi Trafik dan Kebutuhannya

Langkah pertama yang diambil untuk mengimplementasikan QoS dalam jaringan adalah mempelajari jaringan untuk menentukan tipe trafik yang berjalan dalam jaringan dan selanjutnya menentukan kebutuhan QoS untuk tipe trafik yang berbeda-beda tersebut.

Suara dan data memiliki kebutuhan QoS yang berbeda agar dapat berjalan secara efisien dalam jaringan. Berikut ini adalah kebutuhan QoS untuk masing-masing trafik suara, data:

a. Suara

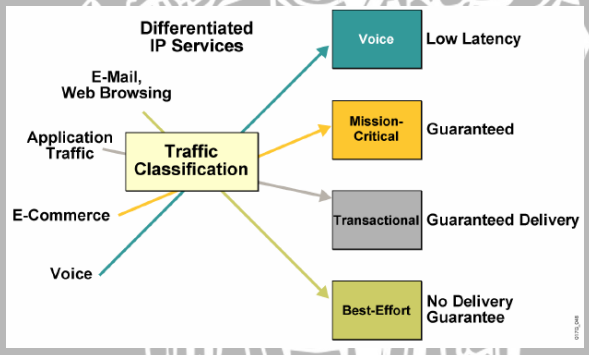
Paket-paket suara umumnya kecil (60 sampai 120 byte) dan tidak dapat mentoleransi adanya *delay* atau *drop*. Karena itulah dipergunakan UDP untuk memaketkan paket suara. Paket-paket suara dapat mentoleransi adanya *delay* dalam batas yang tidak lebih dari 150 ms dan *packet loss* yang kurang dari 1%. Sebuah panggilan suara (*voice call*) akan membutuhkan *bandwidth* prioritas yang berada dalam kisaran antara 17 sampai 106 kbps.

b. Data

Kebutuhan QoS untuk trafik data sangat bervariasi dan tergantung pada aplikasi. Aplikasi yang berbeda memiliki karakteristik trafik yang berbeda. Versi yang berbeda dari aplikasi yang sama dapat memiliki karakteristik trafik yang berbeda. Trafik data berbeda dengan trafik suara dalam hal sensitivitas *delay* dan *drop*. Hampir semua aplikasi data mentoleransi *delay* dan biasanya mentoleransi laju pembuangan yang tinggi (*high drop rates*) juga. Karena trafik data mentoleransi pembuangan, maka kemampuan retransmisi TCP menjadi penting, sebagai hasilnya banyak aplikasi data menggunakan TCP.

3.3.2. Pengklasifikasian Trafik

Setelah bagian utama dari trafik jaringan diidentifikasi dan diukur, kemudian trafik diklasifikasikan ke dalam kelas-kelas. Karena kebutuhan QoS yang ketat, trafik suara akan hampir selalu berada dalam sebuah kelas dengan sendirinya. Setelah aplikasi dengan kebutuhan yang paling penting didefinisikan, kelas-kelas trafik sisanya dapat didefinisikan. Hal ini ditunjukkan pada Gambar 3.5.



Gambar 3.5. Mengklasifikasikan Trafik ke dalam Kelas-kelas
Sumber: *Implementing Cisco Quality of Service (QoS) v2.1* (2004:1-26)

3.3.3. Mendefinisikan QoS *Policy*

Mendefinisikan *policy* QoS yang sesuai dengan kebutuhan QoS untuk masing-masing kelas. Untuk mendefinisikan QoS *policy* tersebut, melibatkan hal-hal berikut:

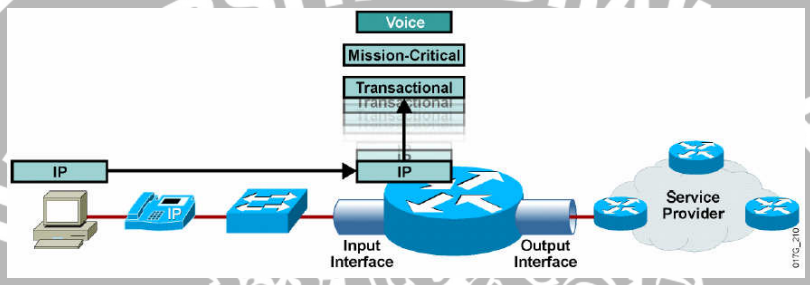
a. Mengatur *bandwidth guarantee*.
b. Memberikan prioritas pada masing-masing kelas.
c. Menggunakan mekanisme QoS.

3.4. Mekanisme IP QoS

Mekanisme IP QoS digunakan untuk mengimplementasi sebuah QoS *policy* yang terkoordinir dalam jaringan. Saat ketika paket IP memasuki jaringan, paket tersebut diklasifikasikan dan biasanya ditandai dengan identifikasi kelasnya. Dari titik tersebut, paket diproses oleh bermacam-macam mekanisme IP QoS sesuai dengan klasifikasi paketnya.

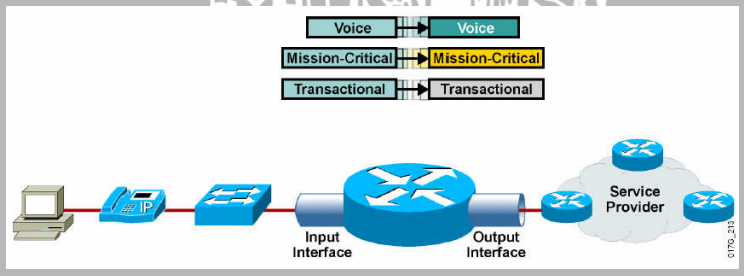
3.4.1. Klasifikasi dan Penandaan

Klasifikasi adalah mengidentifikasi dan membagi trafik ke dalam kelas-kelas yang berbeda.



Gambar 3.6. Mengklasifikasikan Trafik
Sumber: *Implementing Cisco Quality of Service (QoS) v2.1* (2004:2-29)

Penandaan (*marking*), menandai masing-masing paket atau *frame* sebagai golongan dari sebuah kelas sehingga kelas paket tersebut dapat dibedakan dari paket-paket lain dalam jaringan.

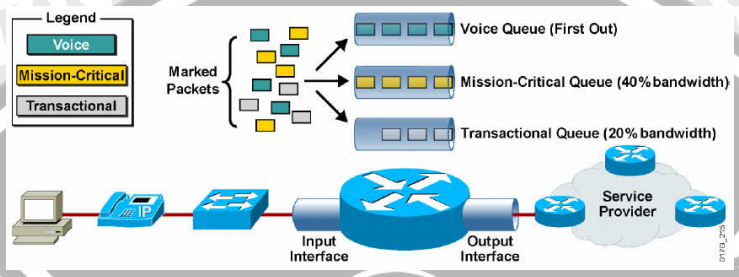


Gambar 3.7. Penandaan Trafik
Sumber: *Implementing Cisco Quality of Service (QoS) v2.1* (2004:2-30)

Mekanisme klasifikasi QoS digunakan untuk memisahkan trafik dan mengidentifikasi paket-paket sebagai golongan dari kelas layanan tertentu. Mekanisme penandaan QoS trafik digunakan untuk melabeli paket sebagai golongan dari kelas layanan tertentu.

3.4.2. Pengelolaan Kepadatan

Ketidaksesuaian kecepatan adalah penyebab terbesar terjadinya kepadatan dalam sebuah jaringan. Ketidaksesuaian kecepatan umumnya terjadi ketika trafik berpindah dari lingkungan berkecepatan tinggi menuju ke saluran berkecepatan rendah. Pengelolaan kepadatan (*congestion management*) menggunakan penandaan pada masing-masing paket untuk menentukan suatu paket akan ditempatkan pada antrian mana.



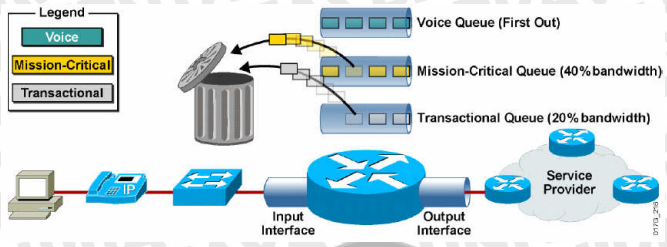
Gambar 3.8. Pengelolaan Kepadatan
Sumber: *Implementing Cisco Quality of Service (QoS) v2.1* (2004:2-31)

Pengelolaan kepadatan diimplementasikan dalam jaringan yang mendukung QoS dengan menggunakan mekanisme antrian untuk mengelola aliran keluaran trafik. Algoritma antrian di desain untuk mengelola kepadatan. Banyak algoritma di desain untuk melayani kebutuhan yang berbeda-beda. Beberapa contoh teknik untuk mengelola kepadatan atau antrian diantaranya adalah *First In First Out* (FIFO), *Low Latency Queuing* (LLQ), *Weighted Fair Queuing* (WFQ) dan *Class-based Weighted Fair Queuing* (CBWFQ).

3.4.3. Pencegahan Kepadatan

Kepadatan biasanya terjadi sebagai hasil dari kurangnya kapasitas *buffer*, titik berkumpulnya jaringan, atau sebuah saluran *wide-area* yang berkecepatan rendah, maka teknik pengelolaan kongesti diadakan untuk memastikan bahwa aplikasi tertentu dan kelas-kelas trafik diberikan bagian *bandwidth*nya masing-masing ketika kongesti terjadi. Mekanisme pencegahan kepadatan dicapai melalui pembuangan paket.

Salah satu teknik untuk mencegah kepadatan ini adalah *Weighted Random Early Detection* (WRED). WRED meningkatkan probabilitas bahwa kepadatan dicegah dengan membuang paket dengan prioritas rendah terlebih dahulu daripada paket dengan prioritas tinggi. WRED menerapkan pembuangan paket berdasarkan IP *precedence* atau penandaaan DSCP.



Gambar 3.9. Pencegahan Kepadatan
Sumber: *Implementing Cisco Quality of Service (QoS) v2.1* (2004:2-32)

3.4.3.1. *Random Early Detection* (RED)

RED adalah mekanisme pembuangan yang secara acak membuang paket sebelum sebuah antrian menjadi penuh. Strategi pembuangan secara utama berdasarkan pada rata-rata panjang antrian saat ini (*current queue depth*). Jika rata-rata panjang antrian bertambah, RED akan cenderung membuang paket yang datang.

Profile trafik RED digunakan untuk menentukan strategi pembuangan paket dan berdasarkan pada rata-rata panjang antrian. Probabilitas pembuangan paket berdasarkan pada tiga parameter yang dikonfigurasi yang terkandung dalam RED *profile*:

a. *Threshold* minimum

   Minimum *threshold* menspesifikasikan jumlah paket dalam sebuah antrian sebelum antrian tersebut mempertimbangkan pembuangan paket. Ketika rata-rata panjang antrian berada di atas *threshold* minimum, maka RED akan mulai membuang paket-paket. Kecepatan pembuangan paket akan meningkat secara linear seiring dengan peningkatan rata-rata panjang antrian, sampai rata-rata panjang antrian mencapai *threshold* maksimum.

b. *Threshold* maksimum

   Ketika rata-rata panjang antrian melebihi *threshold* maksimum, semua paket yang mencoba memasuki antrian akan dibuang.

c. *Mark probability denominator* (MPD)

   Ini adalah bagian paket yang dibuang ketika rata-rata panjang antrian berada pada *threshold* maksimum. Probabilitas pembuangan paket ketika panjang antrian mencapai threshold maksimum adalah 1/(MPD). Sebagai contoh, jika pembagi (*denominator*) adalah 512, maka satu dari setiap 512 paket akan dibuang ketika rata-rata antrian berada pada *threshold* maksimum. Peningkatan pembuangan paket secara linear dari *threshold* minimum (0 pembuangan) sampai maksimum *threshold* berdasarkan pada parameter ini dan ukuran antrian berada diantara *threshold* minimum dan maksimum.

Berdasarkan pada rata-rata panjang antrian, RED memiliki tiga mode pembuangan paket:

a. *No drop*

   Ketika rata-rata panjang antrian berada antara 0 sampai minimum *threshold*, maka tidak ada pembuangan dan paket diantrikan.

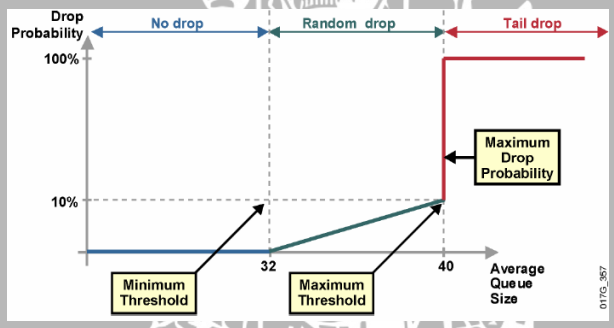
b. *Random drop*

   Ketika rata-rata panjang antrian berada antara *threshold* minimum sampai *threshold* maksimum, maka terjadi pembuangan acak yang secara linear proporsional dengan *mark probability denominator* dan rata-rata panjang paket.

c. *Full drop* (*tail drop*)

   Ketika rata-rata panjang paket berada lebih tinggi daripada *threshold* maksimum, maka *full* (*tail*) *drop* ini diterapkan pada antrian.

Gambar 3.10 berikut ini menunjukkan ketiga range dalam profil RED, yaitu *no drop*, *random drop*, dan *full drop*.



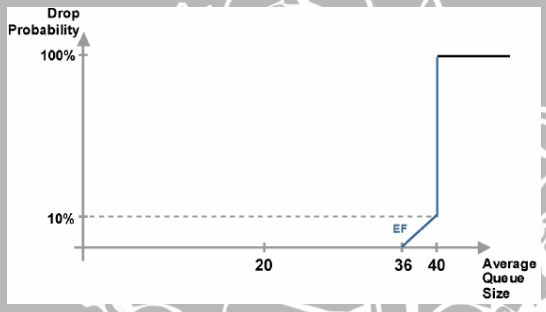
Gambar 3.10. RED *profile*
Sumber: *Implementing Cisco Quality of Service (QoS) v2.1* (2004:6-15)

3.4.3.2. *Weighted Random Early Detection* (WRED)

WRED mengkombinasikan RED dengan IP *precedence* atau DSCP dan menerapkan pembuangan paket berdasarkan IP *precedence* atau penandaaan DSCP. Seperti halnya dengan RED, WRED memantau rata-rata panjang antrian dalam *router* dan menentukan kapan akan membuang paket berdasarkan panjang antrian antarmuka. Ketika rata-rata panjang antrian melebihi minimum *threshold*, maka WRED akan mulai secara acak membuang paket (TCP dan UDP) dengan probabilitas tertentu. Jika rata-rata panjang antrian semakin meningkat dan melebihi maksimum *threshold*, maka WRED akan melakukan *tail drop* terhadap paket-paket tersebut, yang memungkinkan semua paket yang datang akan dibuang.

Ide yang ada dibalik WRED adalah untuk menjaga agar panjang antrian berada pada *level* diantara minimum dan maksimum *threshold*, dan untuk mengimplementasikan *drop policy* yang berbeda untuk berbagai kelas trafik. WRED secara selektif dapat membuang trafik dengan prioritas rendah ketika antarmuka mengalami kongesti dan dapat menyediakan karakteristik performansi yang berbeda untuk kelas layanan yang berbeda. Secara *default*, maksimum *threshold* untuk semua nilai DSCP adalah 40. Sedangkan nilai *mark probability denominator* untuk semua nilai DSCP secara *default* adalah 10. (*Implementing Cisco Quality of Service (QoS) v2.1*, 2004:6-28)

Untuk kelas trafik EF, WRED mengkonfigurasi dirinya sendiri sehingga minimum *threshold*-nya sangat tinggi, meningkatkan probabilitas *no drop* yang diterapkan pada aplikasi tersebut. Profil EF ini dapat dilihat pada Gambar 3.11 sesuai dengan profil EF *default* pada Tabel 3.3.



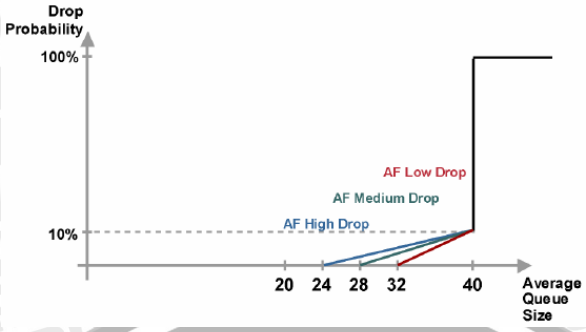
Gambar 3.11. DSCP-*Based* WRED (*Expedited Forwarding*)
Sumber: *Implementing Cisco Quality of Service (QoS) v2.1* (2004:6-30)

Tabel 3.3. EF *Profile*

DSCP (*Six Bits*)	***Default Minimum Threshold***
EF (101110)	36

Sumber: *Implementing Cisco Quality of Service (QoS) v2.1* (2004:6-30)

Ada empat kelas AF yang didefinisikan. Masing-masing kelas sebaiknya diperlakukan secara terpisah dan memiliki alokasi *bandwidth* yang berdasarkan QoS *policy*. Untuk kelas trafik AF, WRED mengkonfigurasi dirinya secara *default* untuk tiga profil yang berbeda, tergantung pada tanda bit *drop preference* DSCP (tiga bit di belakang). Hal ini dapat dilihat pada Gambar 3.12 sesuai dengan Tabel 3.4.



Gambar 3.12. DSCP-Based WRED (*Assured Forwarding*)
Sumber: *Implementing Cisco Quality of Service (QoS) v2.1* (2004:6-31)

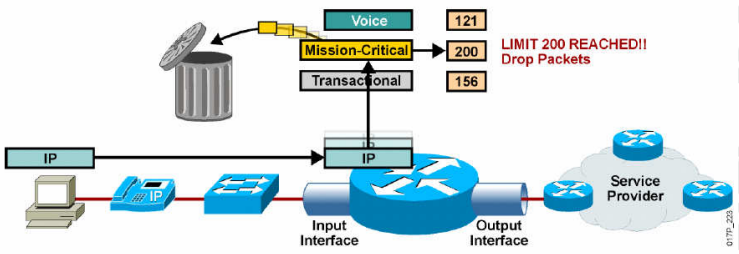
Tabel 3.4. AF *profile*

Assured Forwarding Class	***Drop Probability***	**(AF *Class*) DSCP**	***Default Minimum Threshold***
AF class 1	*Low Drop*	(AF11) 001010	32
	Medium Drop	(AF12) 001100	28
	High Drop	(AF13) 001110	24
AF class 2	*Low Drop*	(AF21) 010010	32
	Medium Drop	(AF22) 010100	28
	High Drop	(AF23) 010110	24
AF class 3	*Low Drop*	(AF31) 011010	32
	Medium Drop	(AF32) 011100	28
	High Drop	(AF33) 011110	24
AF class 4	*Low Drop*	(AF41) 100010	32
	Medium Drop	(AF42) 100100	28
	High Drop	(AF43) 100110	24

Sumber: *Implementing Cisco Quality of Service (QoS) v2.1* (2004:6-31)

3.4.4. *Policing*

Mekanisme *policing* sering digunakan untuk mengkondisikan trafik sebelum mentransmisikan trafik ke dalam sebuah jaringan atau menerima trafik dari sebuah jaringan. *Policing* adalah kemampuan untuk mengontrol *burst* dan menyesuaikan trafik untuk memastikan bahwa tipe-tipe trafik tertentu mendapatkan *bandwidth* tertentu. *Policing* akan membuang paket atau menandai paket ketika batas tertentu dicapai. Mekanisme ini biasanya digunakan untuk mengontrol aliran ke dalam sebuah perangkat jaringan dari sebuah saluran berkecepatan tinggi dengan membuang paket dengan prioritas rendah yang kelebihan.



Gambar 3.13. *Policing*
Sumber: *Implementing Cisco Quality of Service (QoS) v2.1* (2004:2-33)

Mekanisme *traffic policing* merupakan mekanisme pengkondisian trafik yang digunakan dalam jaringan untuk mengontrol kecepatan trafik (*traffic rate*). Mekanisme *traffic policing* mengukur kecepatan trafik dan membandingkan kecepatan tersebut dengan ketentuan *traffic policing*.

Token bucket digunakan oleh perangkat jaringan untuk mengukur kecepatan trafik. *Token bucket* merupakan sebuah model matematik yang digunakan oleh *router* dan *switch* untuk mengatur aliran trafik. Model tersebut memiliki dua komponen dasar:

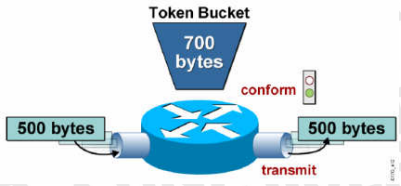
a. *Token-token*

   Masing-masing *token* merepresentasikan sebuah ijin untuk mengirimkan sejumlah bit ke dalam jaringan. *Token* diletakkan ke dalam *token bucket* pada sebuah kecepatan tertentu.

b. *Token bucket*

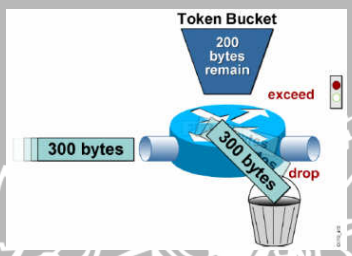
   Sebuah *token bucket* memiliki kapasitas untuk menampung sejumlah *token* yang ditentukan. Masing-masing paket yang datang, jika diteruskan, akan mengambil *token* dari *bucket*, yang merepresentasikan ukuran paket. Jika kapasitas *bucket* telah terpenuhi, maka *token-token* yang baru datang akan dibuang.

Gambar 3.14 menunjukkan sebuah implementasi *token bucket* (*single token bucket*) *traffic policing*. Bermula dengan sebuah kapasitas 700 byte, yang merupakan nilai *token* yang terakumulasi dalam *token bucket*, ketika sebuah paket 500 byte datang pada antarmuka, maka ukuran paket tersebut akan dibandingkan dengan kapasitas *token bucket* (dalam byte). Paket 500 byte tersebut sesuai dengan batas kecepatan (500 byte < 700 byte), sehingga paket dapat dilewatkan. *Token* senilai 500 byte dibawa keluar dari *token bucket*, meninggalkan 200 byte *token* untuk paket selanjutnya.



Gambar 3.14. *Single Token Bucket* (*conform*)
Sumber: *Implementing Cisco Quality of Service (QoS) v2.1* (2004:7-10)

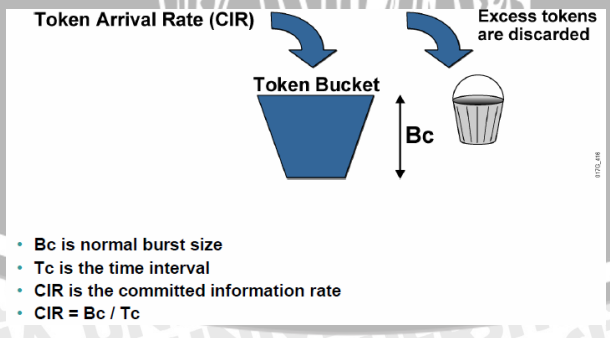
Kemudian ketika sebuah paket 300 byte yang berikutnya datang setelah paket yang pertama, dan tidak ada *token* yang ditambahkan pada *bucket* (yang dilakukan secara periodik), maka paket 300 byte tersebut melebihi batas kecepatan. Paket yang datang tersebut (300 byte) melebihi kapasitas *token* saat ini (200 byte), sehingga dilakukan tindakan *traffic policing*. Seperti ditunjukkan pada Gambar 3.15.



Gambar 3.15. *Single Token Bucket* (*exceed*)
Sumber: *Implementing Cisco Quality of Service (QoS) v2.1* (2004:7-11)

Operasi *token bucket* berdasarkan pada parameter seperti *Committed Information Rate* (CIR), *Commited Burst* (Bc), dan *Committed Time Interval* (Tc). Bc dikenal sebagai *normal burst size*. Hubungan matematis antara CIR, Bc, dan Tc adalah sebagai berikut:

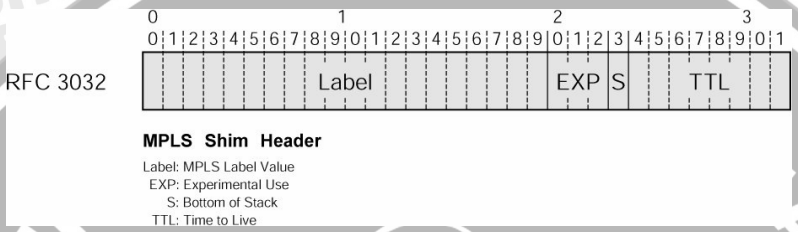
CIR (bps) = Bc (bits) / Tc (sec)



Gambar 3.16. *Single Token Bucket Class-Based Policing*
Sumber: *Implementing Cisco Quality of Service (QoS) v2.1* (2004:7-12)

3.5. *Differentiated Service* dan Paket MPLS

Seperti yang telah diketahui, ada tiga bit EXP (*experimental*) pada sebuah label MPLS yang hanya dipergunakan untuk QoS. Bit-bit ini dapat digunakan dengan cara yang sama dengan saat menggunakan tiga bit *precedence* pada IP *header*. Jika digunakan tiga bit EXP ini, maka *label switched path* (LSP) dapat disebut sebagai sebuah E-LSP, yang mengindikasikan bahwa *label switching router* (LSR) akan menggunakan bit-bit EXP untuk menjadwal paket dan untuk memutuskan *drop precedence*.



Gambar 3.17. MPLS *shim header*
Sumber: Santiago Alvarez (2006)

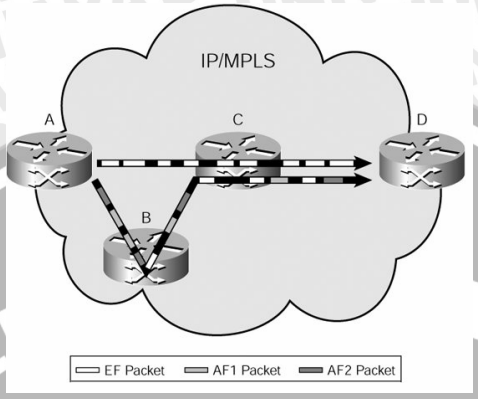
Ketika sebuah LSR melewatkan sebuah paket berlabel, LSR tersebut hanya perlu melihat pada label teratas dalam *label forwarding table*-nya (LFIB) untuk memutuskan kemana paket akan dilewatkan. Hal yang sama juga berlaku untuk perlakuan pada QoS. LSR hanya perlu melihat bit-bit EXP pada label teratas untuk menentukan bagaimana memperlakukan paket tersebut. QoS merupakan penandaan trafik, manajemen kepadatan, pencegahan kepadatan, dan pengkondisian trafik, sehingga fitur mekanisme QoS pada paket-paket IP dapat digunakan untuk mengimplementasikan QoS pada bit-bit EXP untuk paket berlabel.

3.5.1. EXP-*inferred-class* LSP (E-LSP)

E-LSP adalah sebuah jenis LSP yang dapat membawa berbagai macam kelas trafik secara simultan. LSR menggunakan *field* EXP pada *header* MPLS untuk menyimpulkan PHB (*Per Hop Behaviour*) yang dibutuhkan paket. Ukuran dari *field* ini memastikan bahwa sebuah E-LSP dapat menyalurkan paket sampai delapan kelas layanan.

Gambar 3.18 mengilustrasikan sebuah jaringan yang menggunakan E-LSP. Dalam kasus ini, ada dua E-LSP diantara simpul A dan D. Jaringan tersebut mendukung tiga kelas, yaitu EF, AF1, dan AF2. Dari atas ke bawah, E-LSP yang pertama hanya membawa trafik EF, LSP kedua membawa trafik dari tiga kelas. Berdasarkan pada

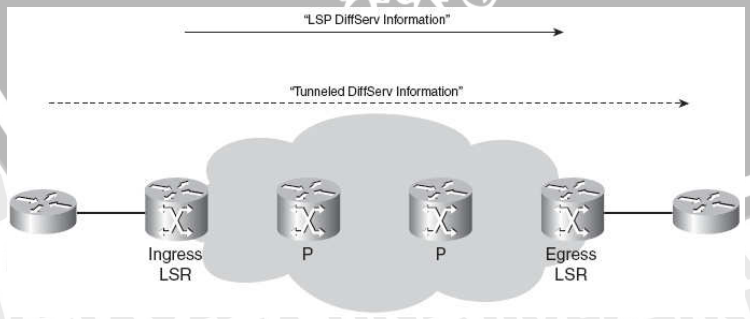
aturan E-LSP, semua simpul melakukan penentuan PHB berdasarkan pada nilai EXP yang ada pada paket. Sebuah LSR mengasosiasikan pemetaan dari EXP ke PHB sebagai label masukan (*input label*) dan pemetaan PHB ke EXP untuk label keluaran (*output label*).



Gambar 3.18. Jaringan MPLS yang Menggunakan E-LSP
Sumber: Santiago Alvarez (2006)

3.6. MPLS DiffServ *Tunneling Model*

Tunneling adalah sebuah kemampuan QoS untuk menjadi transparan dari satu ujung ke ujung lain dalam jaringan. Sebuah *tunnel* dimulai ketika ada pembebanan label. Sebuah *tunnel* berakhir ketika ada pembuangan label. Suatu keuntungan yang nyata dari model *tunneling* ini adalah jaringan MPLS dapat memiliki skema QoS yang berbeda dengan skema yang dimiliki oleh pelanggan yang terhubung dengannya, karena skema MPLS QoS independen dari skema QoS pelanggan. IETF mendefinisikan tiga model untuk melakukan *tunnel* pada informasi DiffServ. Ketiga model tersebut adalah model Pipa (*Pipe Model*), model Pipa Pendek (*Short Pipe Model*), model Seragam (*Uniform Model*). Model yang dipakai dalam skripsi ini adalah model Pipa Pendek.



Gambar 3.19. Operasi Umum dari MPLS DiffServ *Tunneling Model*
Sumber: Luc De Ghein (2007:466)

Tunneled DiffServ *information* adalah QoS dari sebuah paket berlabel atau DSCP dari paket IP yang datang menuju *ingress* LSR dari jaringan MPLS. LSP DiffServ *information* adalah QoS dari paket MPLS (nilai bit-bit EXP) yang dilewatkan pada LSP dari *ingress* LSR menuju *egress* LSR. *Tunneled* DiffServ *information* adalah informasi QoS yang harus dilewatkan secara transparan melewati jaringan MPLS, di mana LSP DiffServ *information* adalah informasi QoS yang digunakan oleh semua LSR dalam jaringan MPLS ini saat meneruskan paket berlabel.

3.6.1. Model Pipa Pendek (*Short Pipe Model*)

Dalam model pipa pendek, diterapkan aturan-aturan berikut:

a. LSP DiffServ *information* tidak perlu (tapi mungkin) diturunkan dari *tunneled* DiffServ *information* pada *ingress* LSR.

b. Pada sebuah *intermediate* LSR (*router* P), LSP DiffServ *information* dari *outgoing label* diturunkan dari LSP DiffServ *information* dari *incoming label*.

c. Pada *egress* LSR, *forwarding treatment* dari paket berdasarkan pada *tunneled* DiffServ *information*, dan LSP DiffServ *information* tidak dipropagasi ke *tunneled* DiffServ *information*.

Forwarding treatment disini berarti mengklasifikasikan paket untuk penjadwalan dan pembuangan pada antarmuka keluaran.

Tunneled DiffServ *information* merupakan bit-bit *precedence* atau DSCP dari paket IP. LSP DiffServ *information* merupakan nilai bit EXP dari label pada jaringan MPLS. *Forwarding treatment* (pengklasifikasian dan pembuangan) dari paket IP berdasarkan pada bit-bit *precedence* atau DSCP dalam IP *header*. Hal ini disebut dengan IP PHB (*per-hop behaviour*). *Forwarding treatment* terhadap paket MPLS berdasarkan pada bit-bit EXP. Hal ini disebut dengan MPLS PHB.

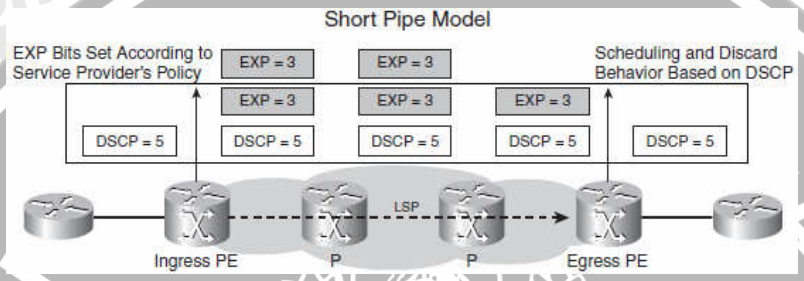
Sebagai contoh jika paket yang diterima pada *ingress* LSR merupakan paket IP, maka aturan tersebut menjadi:

a. Bit-bit EXP dapat disalin dari IP *precedence* atau di atur melalui konfigurasi pada *ingress* LSR.

b. Pada sebuah *router* P, bit-bit EXP dipropagasikan dari *incoming label* ke *outgoing label*.

c. Pada *egress* LSR, *forwarding treatment* dari paket berdasarkan pada IP PHB (IP *precedence*), dan bit-bit EXP tidak dipropagasikan ke IP *precedence*.

Model Pipa Pendek (*Short Pipe Model*) ini ditunjukkan pada Gambar 3.20 khususnya pada bagian yang diberi blok kotak. Paket-paket yang memasuki dan keluar dari MPLS VPN adalah paket-paket IP. Karena itulah *tunneled* DiffServ *information* diterjemahkan ke dalam bit-bit DSCP dalam IP *header*, dan LSP DiffServ *information* diterjemahkan dalam bit-bit EXP dalam label MPLS.



Gambar 3.20. *Short Pipe Model* untuk MPLS VPN
Sumber: Luc De Ghein (2007:469)

3.7. *Differentiated Service-Traffic Engineering*

Mekanisme DiffServ dapat digunakan untuk melengkapi mekanisme MPLS TE karena keduanya beroperasi dalam sebuah basis *aggregate* dalam semua kelas layanan DiffServ. Dalam hal ini, DiffServ dan MPLS TE mendukung keuntungannya masing-masing. Dengan memetakan trafik dari kelas trafik yang berbeda ke dalam MPLS LSP yang berbeda, DiffServ dalam jaringan MPLS dapat memenuhi kebutuhan khusus bagi tiap kelas pada jalur yang terpendek maupun pada jalur yang panjang. Strategi TE ini disebut DiffServ-*aware Traffic Engineering* (DS-TE).

DS-TE mengijinkan untuk melakukan *advertise* terhadap lebih dari sebuah *pool* dari sumber (*resource*) yang tersedia untuk sebuah saluran, yaitu sebuah *global pool* dan *subpool-subpool*. Sebuah *subpool* adalah sebagian kecil dari *bandwidth* saluran yang tersedia untuk sebuah tujuan tertentu.

Penggunaan dari *subpool* DS-TE ini direkomendasikan sebagai alokasi *bandwidth* untuk antrian dengan kebutuhan *low-latency* pada sebuah saluran. Kemampuan ini mengijinkan pembangunan TE-LSP yang secara spesifik memesan *subpool bandwidth* yang hanya membawa trafik LLQ, dan sebagai akibatnya hal ini seakan-akan seperti membangun sebuah jaringan kedua diatas jaringan yang telah

dimiliki sebelumnya. Hal ini akan memberikan utilisasi *resource* yang lebih baik dalam jaringan.

3.8. Parameter Performansi Jaringan

Performansi jaringan merupakan kumpulan dari berbagai besaran teknis yang merujuk pada tingkat kecepatan dan keandalan penyampaian berbagai jenis beban data di dalam suatu sistem komunikasi. Performansi merupakan kumpulan berbagai besaran teknis.

3.8.1. *Throughput*

Throughput adalah kecepatan maksimum jaringan saat tidak ada data yang hilang pada petransmisiannya. *Throughput* merupakan parameter yang digunakan untuk mengetahui jumlah data yang diterima dalam keadaan baik terhadap waktu total transmisi. Pengiriman data antara *client* dan *server* dengan menggunakan TCP akan berusaha secara seksama untuk dapat mengirimkan data dengan cara memeriksa kesalahan, mengirim data ulang (retransmisi) dan dengan mengirimkan *error* ke lapisan atasnya apabila TCP tidak berhasil melakukan komunikasi. Dalam proses ini *throughput* dari sistem dengan memperhatikan probabilitas paket diterima dalam keadaan salah (ρ) adalah [Schwartz, 1987:129] :

$$\lambda = \frac{(1-\rho)}{t_I[1+(\alpha-1)\rho]} \qquad (3\text{-}1)$$

with

λ = *throughput* (paket/detik)

ρ = probabilitas frame error pada node IPv6 jaringan TCP/IP

α = konstanta = $1 + (t_{out}/t_I)$

$$t_{out} = t_{prop} + 2t_I \qquad (3\text{-}2)$$

t_{out} = waktu yang diperlukan untuk menerima ack (detik)

t_I = waktu yang diperlukan untuk mengirimkan sebuah frame (detik)

Perhitungan t_{out} (*fixed timeout interval*) berdasarkan protokol *Go-Back-N*.

Pada proses pentransmisian data dari pengirim ke penerima, data yang dikirimkan mempunyai karakteristik tertentu, yaitu panjang paket data pada L bits dengan total dari L' yang digunakan pada bidang sisanya (kontrol/pengaman), sehingga probabilitas bit salah (Primantoko; 2003:61)

$$\rho = 1 - (1 - \rho_b)^{L+L'} = 1 - q_b^{L+L'} \quad (3\text{-}3)$$

dengan

ρ_b = probabilitas bit salah

q_b = probabilitas bit benar $(1 - \rho_b)$

Untuk ρ_b yang sangat kecil sedemikian hingga $(L + L')\rho_b \ll 1$, maka

$$\rho = (L + L')\rho_b \ll 1 \quad (3\text{-}4)$$

Throughput didefinisikan sebagai jumlah paket yang diterima di sisi penerima dengan benar setiap satuan waktu. *Throughput* tersebut ditentukan dengan persamaan dibawah ini. (Joko Siswanto; 2007:35)

$$\lambda = \frac{1}{t_v} \quad (3\text{-}5)$$

dengan

λ = *throughput* (*paket*/s)

t_v = waktu rata-rata untuk mentransmisikan 1 *paket* yang benar (s)

Delay end to end yang diperoleh dari simulasi adalah waktu yang diperlukan untuk mentransmisikan seluruh data. Sedangkan t_v adalah rata-rata waktu transmisi dari sebuah paket, sehingga t_v diperoleh dengan membagi *delay end-to-end* dengan jumlah paket yang diterima.

$$t_v = \frac{t_{tot}}{\sum P} \quad (3\text{-}6)$$

dengan

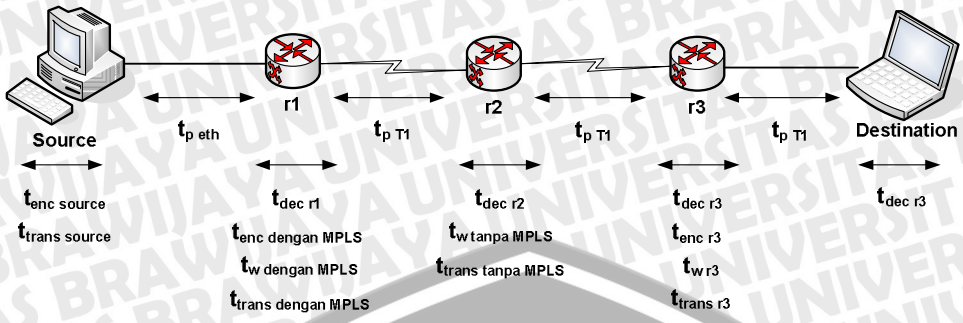
t_{tot} = *delay end-to-end* (detik)

$\sum P$ = jumlah paket yang diterima

(Evi Syarvianti; 2005:62)

3.8.2. *Delay End to End*

Delay adalah waktu yang dibutuhkan untuk mengirimkan paket data dari sumber sampai ke tujuan. Pada jaringan MPLS, delay merupakan penjumlahan dari keseluruhan waktu yang dibutuhkan dalam perjalanan paket dari sumber ke tujuan. *Delay end-to-end* terjadi antara dua *end-point* atau *end-user* yang dihitung dari sumber (*Source*) ke tujuan (*Destination*). Komponen-komponennya ditunjukkan dalam Gambar 3.21 untuk jalur *source*-r1-r2-r3-*destination*.

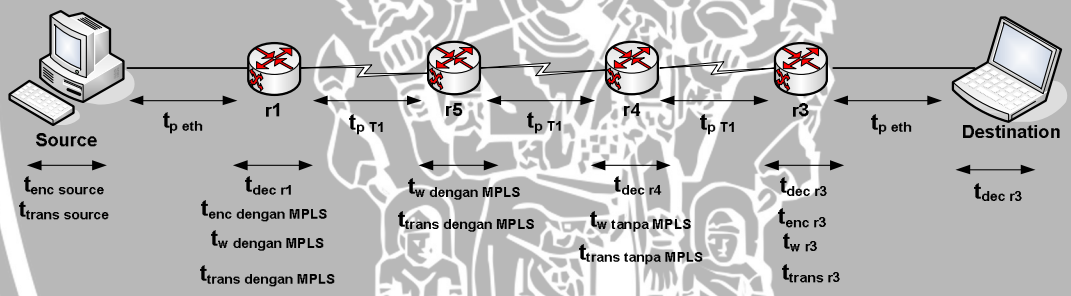


Gambar 3.21. *Delay end to end* pada Jalur *source*-r1-r2-r3-*destination*
Sumber: Analisis

Dan analisisnya yang dapat dihitung dengan persamaan :

$$t_{end\ to\ end}=t_{enc\ source}+(t_{trans\ source}+t_{p\ eth}+t_{dec\ r1}+t_{enc\ dg\ MPLS}+t_{w\ dg\ MPLS})+(t_{trans\ dg\ MPLS}+t_{p\ T1}+t_{dec\ r2}+t_{w\ tanpa\ MPLS})+(t_{trans\ tanpa\ MPLS}+t_{p\ T1}+t_{dec\ r3}+t_{enc\ r3}+t_{w\ r3})+(t_{trans\ r3}+t_{p\ T1}+t_{dec\ r3}) \quad (3\text{-}7)$$

Sedangkan komponen *delay end to end* untuk jalur *source*-r1-r5-r4-r3-*destination* ditunjukkan oleh Gambar 3.22 berikut ini.



Gambar 3.22. *Delay end to end* pada Jalur *source*-r1-r5-r4-r3-*destination*
Sumber: Analisis

Yang analisisnya yang dapat dihitung dengan persamaan sebagai berikut

$$t_{end\ to\ end}=t_{enc\ source}+(t_{trans\ source}+t_{p\ eth}+t_{dec\ Source}+t_{enc\ dg\ MPLS}+t_{w\ dg\ MPLS})+(t_{trans\ dg\ MPLS}+t_{p\ T1}+t_{w\ dg\ MPLS})+(t_{trans\ dg\ MPLS}+t_{p\ T1}+t_{dec\ r4}+t_{w\ tanpa\ MPLS})+(t_{trans\ tanpa\ MPLS}+t_{p\ T1}+t_{dec\ r3}+t_{enc\ r3}+t_{w\ r3})+(t_{trans\ r3}+t_{p\ eth}+t_{dec\ r3}) \quad (3\text{-}8)$$

dengan

$t_{end\ to\ end}$ = *delay end to end* (s)

t_{enc} = *delay* enkapsulasi (s)

t_{trans} = *delay* transmisi (s)

t_p = *delay* propagasi (s)

t_w = *delay* antrian (s)

t_{dec} = *delay* dekapsulasi (s)

3.8.3. *Delay* Proses

Adalah waktu yang dibutuhkan untuk memproses paket data dalam bentuk biner dan untuk menentukan ke mana data tersebut akan diteruskan. *Delay* proses meliputi enkapsulasi dan dekapsulasi. Apabila sumber ingin mengirim data ke tujuan, maka proses yang terjadi adalah data aplikasi akan dikirimkan ke *transport layer*. *Transport layer* yang digunakan pada analisis ini adalah TCP dan UDP.

a. Perhitungan *Delay* Enkapsulasi pada Paket UDP

Pada paket UDP pada *transport layer*, panjang segmennya adalah

$$W_{segmen} = W_{data} + Header_{UDP} \quad (3\text{-}9)$$

dengan

W_{segmen} = panjang segmen UDP (*byte*)

W_{data} = panjang data yang dikirimkan (*byte*)

$Header_{UDP}$ = panjang *header* UDP (*byte*)

Setelah melalui *transport layer*, segmen melalui *internet layer* dengan panjang datagram adalah

$$W_{datagram} = W_{segmen} + Header_{ip} \quad (3\text{-}10)$$

dengan

$W_{datagram}$ = panjang datagram IP (*byte*)

$Header_{ip}$ = panjang *header* IPv4 (*byte*)

Setelah itu paket melalui *network interface layer*, baru kemudian dikirim dengan panjang *frame*,

$$W_{frame} = W_{datagram} + Header_{Ethernet\,II} \quad (3\text{-}11)$$

dengan

W_{frame} = panjang frame *Ethernet (byte*)

$Header_{Ethernet\,II}$ = panjang *header Ethernet* (*byte*)

Sehingga jumlah total *frame Ethernet* yang dikirimkan dari sumber ke tujuan adalah:

$$W_{frame\ total} = jumlah\ paket\ udp\ x\ (W_{frame}) \quad (3\text{-}12)$$

di mana

$W_{frame\ total}$ = panjang total frame setelah enkapsulasi (*byte*)

Maka, besarnya *delay* enkapsulasi adalah:

$$t_{enc} = \frac{W_{frame\ total}}{c_{proc}} x8 \quad (3\text{-}13)$$

dengan

t_{enc} = *delay* enkapsulasi (detik)

$W_{frame\ total}$ = panjang total frame setelah enkapsulasi (*byte*)

c_{proc} = kecepatan pemrosesan data (bps)

(Maya Ari Hardini; 2006:31)

b. Perhitungan *Delay* Enkapsulasi pada Paket TCP

Sedangkan untuk paket TCP, perhitungan *delay*-nya adalah sebagai berikut:

Pada *transport layer*, panjang segmen paket TCP adalah

$$W_{segmen} = W_{data} + (Header_{TCP} + option) \quad (3\text{-}14)$$

Setelah melalui *transport layer*, segmen melalui *internet layer* dengan panjang datagram

$$W_{datagram} = W_{segmen} + Header_{ip} \quad (3\text{-}15)$$

Paket selanjutnya ditambah dengan Header MPLS. Setelah itu paket melalui *network interface layer*, baru kemudian dikirimkan dengan panjang *frame* adalah

$$W_{frame} = W_{datagram} + Header_{Ethernet\ II} \quad (3\text{-}16)$$

Sehingga jumlah total frame yang ditransmisikan dari sumber ke tujuan adalah

$$W_{frame\ total} = jumlah\ paket\ tcp\ x\ (W_{frame}) \quad (3\text{-}17)$$

Jadi, *delay* enkapsulasi paket TCP adalah

$$t_{enc} = \frac{W_{frame}}{c_{proc}} x8 \quad (3\text{-}18)$$

Sedangkan *delay* dekapsulasi dinyatakan dengan:

$$t_{dec} = \frac{W_{frame}}{c_{proc}} x8 \quad (3\text{-}19)$$

dengan

t_{enc} = *delay* dekapsulasi (detik)

$W_{frame\ total}$ = panjang total frame (*byte*)

C_{proc} = kecepatan pemrosesan data (bps)

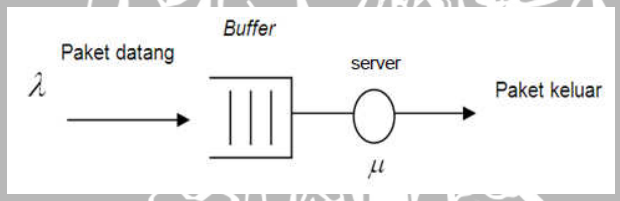
3.8.4. *Delay Antrian*

Delay antrian dapat didefinisikan sebagai banyaknya waktu yang dibutuhkan dari masuknya paket data sampai keluarnya paket data dari *router*. *Delay* antrian dapat dihitung dengan menggunakan model antrian M/M/1. M pertama menunjukkan distribusi kedatangan *Poisson* yang berarti acak, M kedua berarti distribusi waktu pelayanan eksponensial, angka 1 menunjukkan bahwa jumlah *server* adalah tunggal. Disiplin antrian yang digunakan dalam skripsi ini yaitu FIFO (*First In First Out*). Parameter pada model tersebut adalah :

a. Kapasitas link adalah C bps dan panjang paket data adalah L bit. Besarnya kapasitas link akan menentukan kecepatan pelayanan, yaitu:

$$\mu = \frac{C}{L} \quad \text{(paket/detik)} \quad \text{(3-20)}$$

b. Interval waktu untuk permintaan (*request*) merupakan distribusi Poisson dengan kecepatan data adalah λ (paket/detik).



Gambar 3.23. Model Antrian M/M/1
Sumber : Firdiana Inanda (2008:63)

Besarnya *delay* antrian yang terjadi pada *server* yaitu:

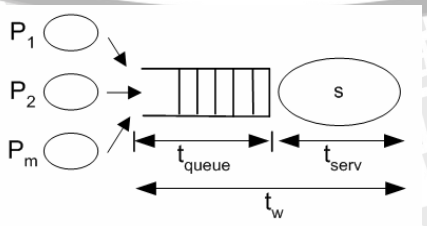
$$t_w = t_{queue} + t_{serv} \quad \text{(3-21)}$$

dengan

t_w = *delay* antrian pada *server* (s)

t_{queue} = waktu tunggu paket pada *server* (s)

t_{serv} = waktu rata-rata pelayanan *server* (s)



Gambar 3.24. Analisis *Delay* Antrian
Sumber : Joko Siswanto (2007:57)

Sementara itu,

$$t_{serv} = \frac{1}{\mu} \quad (3\text{-}22)$$

dengan

t_{serv} = waktu rata-rata pelayanan *server* (s)

μ = kecepatan pelayanan *server* (paket/s)

Nilai kecepatan pelayanan *server* diperoleh dengan persamaan (Joko Siswanto, 2007:7):

$$\mu = \frac{C}{L} \quad (3\text{-}23)$$

dengan

μ = kecepatan pelayanan *server* (paket/s)

C = kapasitas kanal (bps)

L = panjang paket (*bit*/paket)

Performansi sistem antrian ditunjukkan dalam bentuk utilitas :

$$\rho = \frac{\lambda}{\mu} \rightarrow \lambda = \mu\rho \quad (3\text{-}24)$$

dengan

ρ = utilitas ($0 < \rho < 1$)

λ = kecepatan kedatangan paket pada *server* (paket/s)

μ = kecepatan pelayanan *server* (paket/s)

Dengan menggunakan teori Little diperoleh nilai *delay* antrian sebagai berikut:

$$t_w = \frac{1}{\mu(1-\rho)} \quad (3\text{-}25)$$

dengan

t_w = *delay* antrian (s)

μ = kecepatan pelayanan *server* (paket/s)

ρ = utilitas ($0 < \rho < 1$)

Dari persamaan di atas maka waktu tunggu paket dirumuskan :

$$t_{queue} = t_w - t_{serv} = \frac{\lambda}{\mu(\mu-\lambda)} \quad (3\text{-}26)$$

dengan

t_{queue} = waktu tunggu paket pada *server* (s)

t_w = *delay* antrian (s)

t_{serv} = waktu rata-rata pelayanan *server* (s)

λ = kecepatan kedatangan paket pada *server* (paket/s)

μ = kecepatan pelayanan *server* (paket/s)

Dengan demikian, maka *delay* antrian dapat dituliskan sebagai :

$$t_w = \frac{\lambda}{\mu(\mu-\lambda)} + \frac{1}{\mu} \quad (3\text{-}27)$$

dengan

t_w = *delay* antrian (s)

λ = kecepatan kedatangan paket pada *server* (paket/s)

μ = kecepatan pelayanan *server* (paket/s)

3.8.5. *Delay* Propagasi

Delay propagasi adalah waktu yang dibutuhkan sebuah sinyal untuk merambat dari sumber ke tujuannya. Besar *delay* propagasi dihitung sebagai berikut (Firdiana Inanda; 2008:69) :

$$t_p = \frac{S}{V} \quad (3\text{-}28)$$

dengan

t_p = *delay* propagasi (detik)

S = jarak antara sumber dan tujuan (meter)

V = kecepatan propagasi informasi pada media/saluran transmisi (meter/detik)

Untuk *guided transmission*, V= $0{,}67.3.10^8$ m/s

3.8.6. *Delay* Transmisi

Adalah waktu yang dibutuhkan untuk meletakkan semua data dalam bentuk biner pada medium, dipengaruhi oleh ukuran paket dan kapasitas media transmisi. Besarnya *delay* transmisi (Maya Ari Hardini; 2006:34) :

$$t_{trans} = N_{frame} x \frac{(l+l')x8}{C_{trans}} \quad (3\text{-}29)$$

dengan

t_{trans} = *delay* transmisi (detik)

N_{frame} = jumlah *frame Ethernet*

l = panjang paket data (*byte*)

l' = panjang header paket (*byte*)

C_{trans} = kapasitas saluran transmisi (bps)

BAB IV
METODOLOGI

Kajian yang dilakukan dalam skripsi ini adalah kajian yang bersifat analisis, yaitu tentang *Meningkatkan QoS pada MPLS dengan Metode DS-TE* dalam bentuk konsep yang mengacu pada studi kepustakaan. Metodologi yang digunakan dalam penulisan skripsi ini didasarkan kepada:

4.1. Studi Literatur

Pada tahap awal dilakukan studi literatur, yaitu melakukan kajian pustaka untuk memahami teori yang berkaitan dengan MPLS, DiffServ, MPLS *Traffic Engineering*, dan MPLS DS-TE, konfigurasi *router* Cisco, perhitungan parameter-parameter performansi sistem serta teori-teori pengantar yang diperlukan untuk menunjang analisisnya.

4.2. Perancangan Simulasi

Dari literatur yang dipelajari maka perancangan simulasi dibagi menjadi tiga bagian yaitu:

a. Perancangan implementasi jaringan MPLS dengan menerapkan metode DiffServ dan DS-TE meliputi perancangan topologi, perancangan QoS yang akan diterapkan pada jaringan MPLS dan perancangan penerapan rekayasa trafik untuk simulasi metode DS-TE.
b. Perancangan penerapan perangkat keras sistem meliputi komputer sebagai *source host*, *destination host*, dan jaringan MPLS. Dan perancangan penerapan *interface* yang meliputi *Network Interface Card* (NIC)/*Local Area Connection* (LAN) Card sebagai *interface* komputer dengan media transmisi *Unshielded Twisted Pairs* (UTP) serta kabel UTP CAT 5.
c. Perancangan penerapan perangkat lunak sistem meliputi emulator jaringan MPLS dan pembangkit trafik.

4.3. Pengujian

Pengujian ini bertujuan untuk memastikan bahwa sistem dapat berjalan sesuai dengan perancangan. Bagian dari proses pengujian ini adalah sebagai berikut:

1. Pengujian interkoneksi *host* sumber (*Source*) dan *host* tujuan (*Destination*).
2. Pengujian terhadap komponen konfigurasi *router*.
3. Pengujian pengiriman dan penerimaan trafik data pada *host* sumber dan *host* tujuan.

Pengujian jalur *tunnel* yang dilalui trafik UDP dan TCP pada simulasi jaringan MPLS dengan metode DS-TE.

4.4. Pengambilan Data

Data-data yang akan digunakan adalah data primer dan data sekunder. Data primer adalah data yang diperoleh dari pengamatan secara langsung, sedangkan data sekunder adalah data yang diperoleh dari berbagai buku teks, jurnal-jurnal, *download* dari internet.

4.4.1. Data Primer

Data primer yang diperoleh dari pengamatan secara langsung adalah:

a. Waktu pengiriman paket.
b. Waktu kedatangan paket.
c. Jumlah paket yang dikirim oleh *host* sumber.
d. Jumlah paket yang diterima di *host* tujuan.
e. Jumlah paket yang hilang (*packet loss*).

4.4.2. Data Sekunder

Data sekunder yang diperlukan dalam penulisan skripsi ini adalah :

1. Panjang header IPv4 sebesar 20 *byte* (IP Infusion; 2004:7)
2. Panjang label MPLS adalah 4 *byte* (Carne; 2004:16)
3. Panjang *header* TCP sebesar 20 *byte*/paket (Todd Lammle; 2007:75)
4. Panjang *header* UDP sebesar 8 *byte*/paket (Todd Lammle; 2007:76)
5. Panjang *header* Cisco HDLC sebesar 4 *byte*/paket. (http://www.ethereal.com/lists/ethereal-users/200012/msg00030.html)
6. Panjang *header Ethernet* II sebesar 14 *byte*/paket (Joko siswanto; 2007:45)
7. Maksimum panjang kabel T1 adalah 200 meter (Appendix C Link Encryptor User's Guide; 115)
8. Maksimum panjang kabel UTP CAT 5 adalah 200 meter (David Groth; 2001:174)

9. Ukuran paket TCP sebesar 1000 *byte*. (http://mailman.isi.edu/pipermail/ns-users/2005-May/049958.html)
10. Ukuran paket UDP untuk voice sebesar 40 *byte*. (http://searchunifiedcommunications.techtarget.com/?asrc=TAB_searchUnifiedCommunications)

4.5. Analisa

Teori-teori dan data-data yang diperoleh tersebut kemudian dianalisa, hasilnya untuk digunakan sebagai bahan pengambilan simpulan. Analisa dilakukan secara matematis dengan menggunakan persamaan-persamaan yang ada pada bagian tinjauan pustaka dan parameter-parameter aplikasi pada bagian perancangan sistem serta hasil yang didapat dari proses pengujian. Analisa dilakukan pada:

a. *Delay end to end*.
b. *Throughput*.
c. *Packet loss*.

4.6. Pengambilan Kesimpulan

Kesimpulan diambil berdasarkan hasil yang diperoleh dari proses analisa.

BAB V
PERANCANGAN, PENGUJIAN DAN ANALISIS PERFORMANSI SIMULASI METODE *DiffServ* DAN DS-TE PADA MPLS

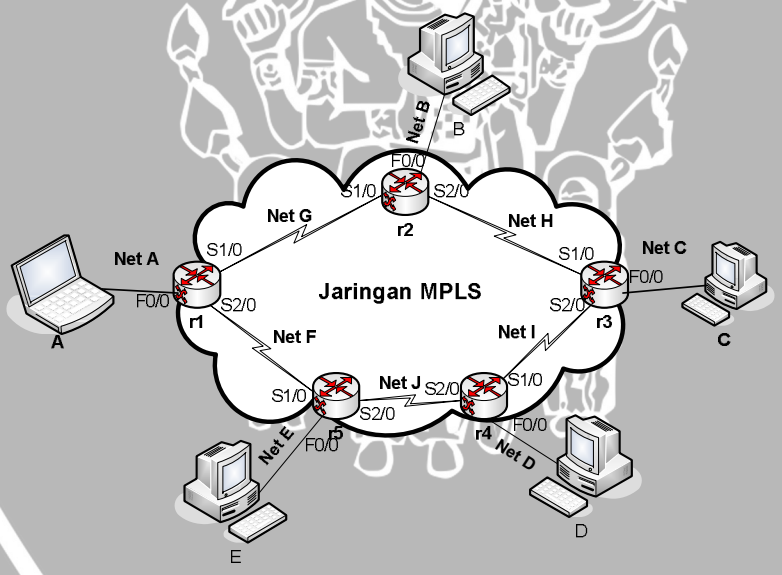
Untuk dapat melakukan simulasi, sebelumnya dilakukan perancangan simulasi yang meliputi perancangan topologi jaringan dan perancangan QoS agar diperoleh hasil yang sesuai dengan tujuan penulisan skripsi.

5.1. Perancangan Implementasi Jaringan MPLS

Perancangan implementasi jaringan MPLS ini meliputi perancangan topologi jaringan MPLS dan perancangan *Quality of Service* untuk kemudian diterapkan dalam jaringan MPLS.

5.1.1. Perancangan Topologi Jaringan MPLS

Bentuk topologi jaringan MPLS yang disimulasikan adalah seperti Gambar 5.1. Topologi jaringan MPLS dengan lima buah *router* ini diterapkan pada PC MPLS.



Gambar 5.1. Topologi Jaringan MPLS

Sumber: Perancangan

Pada Gambar 5.1 tersebut terdapat lima *router* dengan namanya masing-masing adalah r1, r2, r3, r4, dan r5. Antarmuka *router* diberi penamaan S1/0, S2/0, dan F0/0 sesuai dengan penamaan antarmuka Cisco *router*. S untuk antarmuka Serial, dan F untuk antarmuka Fast Ethernet. Tiap jaringan diberi nama Net, maka untuk jaringan A namanya adalah Net A dan seterusnya. Untuk IP jaringan, diasumsikan pemberian

sebuah alokasi IP 192.16.10.0/24 untuk jaringan kelas C. Berikut adalah hasil *subnetting* untuk jaringan MPLS tersebut dengan asumsi jumlah *host* untuk masing-masing jaringan dapat dilihat pada Tabel 5.2 sesuai dengan acuan Tabel 5.1.

Tabel 5.1. *Block Sizes*

Subnet	*Mask*	*Host*	*Block*
/25	128	126	128
/26	192	62	64
/27	224	30	32
/28	240	14	16
/29	248	6	8
/30	252	2	4

Sumber: Todd Lamle (2007:140)

Tabel 5.2. Alokasi *Block* dan Subnet untuk Tiap Jaringan

Network	*Host*	*Block*	*Subnet*	*Mask*
A	7	16	/28	240
B	5	8	/29	248
C	20	32	/27	224
D	4	8	/29	248
E	11	16	/28	240
F	2	4	/30	252
G	2	4	/30	252
H	2	4	/30	252
I	2	4	/30	252
J	2	4	/30	252

Sumber: Perancangan

Pada Tabel 5.2 tersebut tertulis jumlah *host* yang jumlahnya dialokasikan sebagai cadangan perubahan dan perkembangan banyak *host*. Untuk subnet /28 berarti alamat IP yang digunakan memiliki 28 bit *network prefix*. Hal ini berarti empat bit pertama dari delapan bit *host number* merupakan bit 1 dan yang lainnya adalah bit nol, sehingga jika delapan bit *host number* tersebut diubah nilainya dalam desimal, maka akan menjadi nilai mask yang sama dengan 240. Sedangkan jumlah *block* merupakan jumlah kombinasi bit *host number* pada kelas C yang dikurangi dengan nilai mask, yang dalam hal ini pada jaringan A adalah 256 dikurangi dengan 240.

Tabel 5.3. Alamat IP untuk Tiap Jaringan

Network	**Alamat IP**
A	192.16.10.32/28
B	192.16.10.64/29
C	192.16.10.0/27
D	192.16.10.72/29
E	192.16.10.48/28
F	192.16.10.80/30
G	192.16.10.84/30
H	192.16.10.88/30
I	192.16.10.92/30
J	192.16.10.96/30

Sumber: Perancangan

Untuk alamat IP pada masing-masing antarmuka, dapat dilihat pada Tabel 5.4 berikut. Sedangkan untuk tiap *host* diberikan alamat IP seperti yang ditunjukkan oleh Tabel 5.5. Sehingga apabila digambarkan topologinya, akan terlihat seperti pada Gambar 5.2.

Tabel 5.4. Alamat IP untuk Tiap Antarmuka

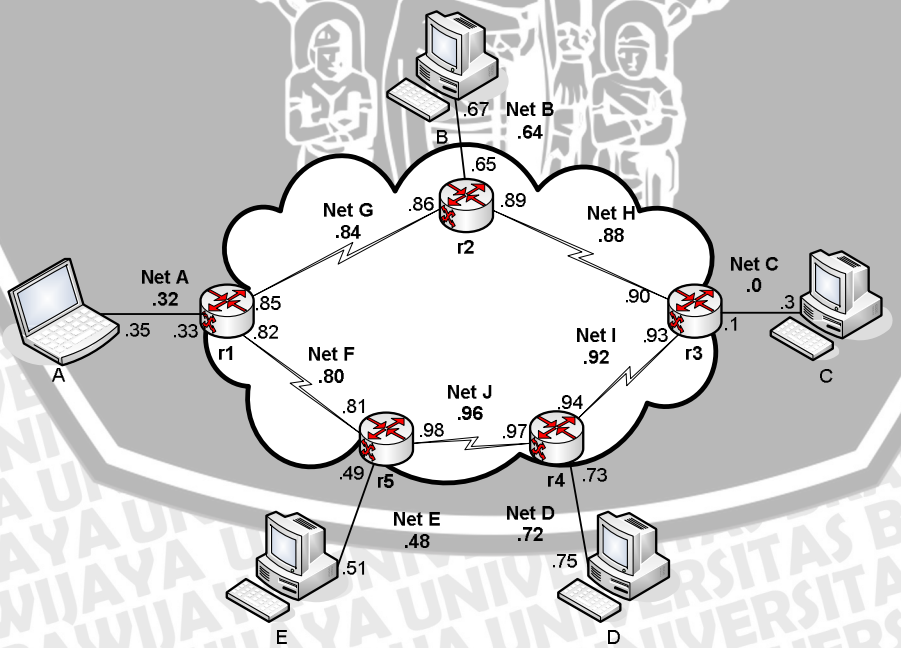
Router	**Antarmuka**	**Alamat IP**	**Mask**
r1	F0/0	192.16.10.33	255.255.255.240
	S1/0	192.16.10.85	255.255.255.252
	S2/0	192.16.10.82	255.255.255.252
r2	F0/0	192.16.10.65	255.255.255.248
	S1/0	192.16.10.86	255.255.255.252
	S2/0	192.16.10.89	255.255.255.252
r5	F0/0	192.16.10.49	255.255.255.240
	S1/0	192.16.10.81	255.255.255.252
	S2/0	192.16.10.98	255.255.255.252
r4	F0/0	192.16.10.73	255.255.255.248
	S1/0	192.16.10.94	255.255.255.252
	S2/0	192.16.10.97	255.255.255.252
r3	F0/0	192.16.10.1	255.255.255.224
	S1/0	192.16.10.90	255.255.255.252
	S2/0	192.16.10.92	255.255.255.252

Sumber: Perancangan

Tabel 5.5. Alamat IP untuk Tiap *Host*

Host	**Alamat IP**
A	192.16.10.35
B	192.16.10.67
C	192.16.10.3
D	192.16.10.75
E	192.16.10.51

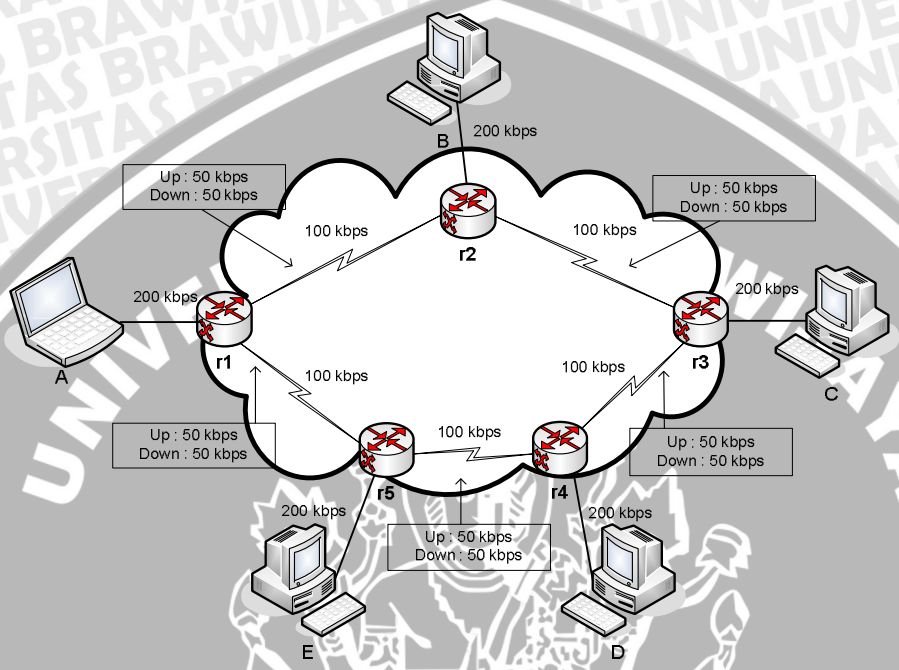
Sumber: Perancangan



Gambar 5.2. Pengalamatan Jaringan MPLS

Sumber: Perancangan

Alokasi *bandwidth* untuk saluran Serial adalah sebesar 100 kbps. *Bandwidth* tersebut dibagi ke dalam *bandwidth uplink* dan *bandwidth downlink*, yang masing-masing dialokasikan 50 kbps. Dan untuk semua saluran Fast Ethernet diberi alokasi *bandwidth* sebesar 200 kbps. Alokasi *bandwidth* ini ditunjukkan oleh Gambar 5.3 berikut.



Gambar 5.3. Alokasi *Bandwidth* Jaringan MPLS

Sumber: Perancangan

Perancangan jalur trafik dilakukan sebagai berikut:

a. A ke B
b. A ke C
c. A ke D
d. A ke E
e. B ke C
f. B ke D
g. B ke E
h. B ke A
i. C ke D
j. C ke E
k. C ke A
l. C ke B
m. D ke A
n. D ke B
o. D ke B
p. D ke C
q. E ke A
r. E ke B
s. E ke C
t. E ke D

5.1.2. Perancangan *Quality of Service*

Setelah melakukan perancangan topologi, selanjutnya dilakukan perancangan implementasi QoS dalam jaringan. Implementasian QoS meliputi hal-hal berikut:

5.1.2.1. Identifikasi Trafik

Trafik yang dipakai dalam simulasi berupa trafik UDP dan trafik TCP. Dalam hal ini trafik UDP dimisalkan sebagai trafik suara. Sedangkan trafik TCP sebagai trafik data. Karakteristik *end to end* yang dibutuhkan untuk masing-masing trafik adalah:

a. Suara dengan *latency* ≤ 150 ms dan *loss* ≤ 1% dengan jaminan *bandwidth* prioritas untuk setiap *call* sebesar 17 sampai 106 kbps.

(*Implementing Cisco Quality of Service (QoS) volume 1 version 2.1*; 2004:1-23)

b. Pada trafik data, aplikasi yang berbeda memiliki karakteristik trafik yang berbeda. Versi yang berbeda dari aplikasi yang sama dapat memiliki karakteristik trafik yang berbeda.

(*Implementing Cisco Quality of Service (QoS) volume 2 version 2.1*; 2004:9-13)

5.1.2.2. Klasifikasi Trafik

Trafik yang telah ditentukan tersebut kemudian dikelompokkan ke dalam empat kelas, yaitu

a. Kelas trafik *voice* yang diprioritaskan untuk trafik suara *real-time*, contohnya VOIP (*Voice over Internet Protocol*).
b. Kelas trafik *mission critical* didefinisikan untuk aplikasi data penting, contohnya trafik SQL.
c. Kelas trafik *bulk* didefinisikan untuk aplikasi yang membutuhkan *bandwidth* yang lebih besar dibanding aplikasi lain, contohnya FTP.
d. Kelas trafik *best effort* untuk trafik lain selain trafik yang diklasifikasikan pada poin 1 sampai 3 di atas.

(*Implementing Cisco Quality of Service (QoS) volume 2 version 2.1*; 2004:9-15)

5.1.2.3. Mendefinisikan QoS *Policy* untuk Tiap Kelas

Pada Tabel 5.6 ditunjukkan klasifikasi QoS yang direkomendasikan untuk penandaan. Penandaan pada tabel ini digunakan untuk menandai paket-paket pada PC *Source*.

Tabel 5.6. Klasifikasi QoS

Kelas	PHB
Voice	EF (*Expedited Forwarding*)
Mission critical	AF31 (*Assured Forwarding*)
Bulk data	AF11 (*Assured Forwarding*)
Best effort	0

Sumber: *Implementing Cisco Quality of Service (QoS) volume 2 version 2.1* (2004:9-20)

Untuk mengkonfigurasi jaringan MPLS dengan menerapkan metode DiffServ memerlukan definisi sebuah SLA. Definisi sebuah SLA yang disediakan oleh suatu *service provider* yang terdiri dari empat kelas layanan, yaitu *Gold, Silver, Bronze* dan *Best Effort* ditunjukkan pada Tabel 5.7.

Tabel 5.7. *Service Level Agreement*

	Gold	***Silver***	***Bronze***	***Best Effort***
Delay (40ms)	90%	75%	50%	25%
Jitter (2 ms)	90%	75%	50%	25%
Packet Loss (0.5%)	90%	75%	50%	25%

Sumber: *Implementing Cisco Quality of Service (QoS) volume 2 version 2.1* (2004:9-33)

Sehingga aplikasi-aplikasi pada Tabel 5.6 dikelompokkan lagi ke dalam empat kelas utama yang dimiliki *service provider*. Kemudian selanjutnya bit-bit EXP pada paket-paket tersebut ditandai sesuai dengan kelasnya. Hal ini ditunjukkan dalam Tabel 5.8. (*Implementing Cisco Quality of Service* (*QoS*) *v2.1*; 2004:9-23).

Tabel 5.6. MPLS *Experimental Bit*

Kelas	**Kelas**	**Experimental Bit**
Voice	*Gold*	5
Mission critical data	*Silver*	3
Bulk data	*Bronze*	1
Best effort	*Best effort*	0

Sumber: *Implementing Cisco Quality of Service (QoS) volume 1 version 2.1* (2004:4-10)

Selanjutnya ditentukan QoS *policy* untuk tiap kelas tersebut. Berikut ini adalah QoS *policy* untuk metode DiffServ dan DS-TE:

a. Kelas *gold*

-Dialokasikan *bandwidth* prioritas sebesar 20% dari *bandwidth* saluran.

-Menggunakan *Single Token Bucket Class-Based Policing*.

-Trafik yang berlebih dibuang.

b. Kelas *silver*

-Alokasi *bandwidth* sebesar 30% dari *bandwidth* saluran.

-Menggunakan *Single Token Bucket Class-Based Policing*.

-Trafik yang berlebih ditandai ulang (*remarked*) kemudian diretransmisikan.

-Menggunakan WRED (*Weighted Random Early Detection*) sebagai mekanisme pencegah kepadatan.

c. Kelas *bronze*

-Alokasi *bandwidth* sebesar 25% dari *bandwidth* saluran.

-Menggunakan *Single Token Bucket Class-Based Policing*.

-Trafik yang berlebih ditandai ulang (*remarked*) kemudian diretransmisikan.

-Menggunakan WRED.

d. Kelas *best effort*

-Dialokasikan *bandwidth* 25% dari *bandwidth* saluran.

-Trafik yang berlebih ditandai ulang (*remarked*) kemudian diretransmisikan.

(*Implementing Cisco Quality of Service (QoS) v2.1*; 2004:9-50)

Konfigurasi *policy* yang digunakan untuk metode DiffServ diterapkan pada tiap kelas tersebut meliputi juga konfigurasi *average rate*, *normal burst* dan konfigurasi *maximum burst* (*burst max*) yang merupakan *excess burst*. Konfigurasinya ditunjukkan dalam Tabel 5.9. Berikut ini adalah persamaan yang dipakai dalam perhitungan konfigurasinya:

$$bandwidth\ kelas = (persentase\ bandwidth) x (bandwidth\ saluran) \quad (5\text{-}1)$$

$$average\ rate\ =\ class\ bandwidth/2 \quad (5\text{-}2)$$

(http://eandriana.wordpress.com/2008/06/03/konsep-cir-committed-information-rate/)

$$burst\ normal = (average\ rate) x \frac{1}{8} x (1{,}5 detik) \quad (5\text{-}3)$$

$$burst\ max = burst\ normal x 2 \quad (5\text{-}4)$$

(http://www.cisconet.com/index.php/QoS/Cisco-How-to-limit-rate-on-interface.html)

Tabel 5.9. QoS *Policy* untuk Metode DiffServ

Kelas	Persentase (%)	*Bandwidth* Kelas (kb)	*Average Rate* (bit)	*Normal Burst* (byte)	*Burst Max* (byte)
Gold	20	20	10000	1875	3750
Silver	30	30	15000	2813	5625
Bronze	25	25	12500	2344	4688
Best effort	25	25	12500	2344	4688

Sumber: Perancangan

QoS *policy* untuk metode DS-TE ini sama dengan QoS *policy* untuk metode DiffServ. QoS *policy* untuk metode DS-TE ditunjukkan oleh Tabel 5.10.

Tabel 5.10. QoS *Policy* untuk Metode DS-TE

Kelas	Persentase (%)	*Bandwidth* Kelas (kb)	*Average Rate* (bit)	*Normal Burst* (byte)	*Burst Max* (byte)
Gold	20	20	10000	1875	3750
Silver	30	30	15000	2813	5625
Bronze	25	25	12500	2344	4688
Best effort	25	25	12500	2344	4688

Sumber: Perancangan

5.1.3. Penerapan Rekayasa Trafik untuk Simulasi Metode DS-TE

Untuk menerapkan metode rekayasa trafik dikonfigurasi dua buah *tunnel* sebagai jalur alternatif pada tiap pengiriman trafik dari sumber ke tujuan yang diberi

nama *tunnel* 1 dan *tunnel* 2. Tiap *tunnel* akan dialokasikan untuk *subpool tunnel* dan *global pool tunnel* dengan alokasi *bandwidth subpool* sebesar 30%. *Subpool tunnel* ini dikonfigurasi untuk membawa trafik prioritas, yang dalam hal ini ditentukan trafik *voice* yang merupakan trafik UDP sebagai trafik prioritas. Sedang *global pool tunnel* yang merupakan *tunnel* keseluruhan merupakan *tunnel* yang membawa trafik data lainnya, yaitu *mission critical*, *bulk* dan *best effort* yang merupakan trafik TCP. Sehingga jalur trafik untuk metode DS-TE dapat dituliskan sebagai berikut:

a. A ke B
b. A ke C, dengan
   tunnel 1 = r1-r2-r3
   tunnel 2 =r1-r5-r4-r3
c. A ke D, dengan
   tunnel 1 = r1-r5-r4
   tunnel 2 = r1-r2-r3-r4
d. A ke E
e. B ke D, dengan
   tunnel 1 = r2-r3-r4
   tunnel 2 = r2-r1-r5-r4
f. B ke E, dengan
   tunnel 1 = r2-r1-r5
   tunnel 2 = r2-r3-r4-r5
g. B ke A
h. B ke C, dengan
   tunnel 1 = r3-r2-r1
   tunnel 2 = r3-r4-r5-r1
i. C ke A
j. C ke B
k. C ke D
l. C ke E, dengan
   tunnel 1 = r3-r4-r5
   tunnel 2 = r3-r2-r1-r5
m. D ke A, dengan
   tunnel 1 = r4-r5-r1
   tunnel 2 = r4-r3-r2-r1
n. D ke B, dengan
   tunnel 1 = r4-r3-r2
   tunnel 2 = r4-r5-r1-r2
o. D ke C
p. D ke E
q. E ke A
r. E ke B, dengan
   tunnel 1 = r5-r1-r2
   tunnel 2 = r5-r4-r3-r2
s. E ke C, dengan
   tunnel 1 = r5-r4-r3
   tunnel 2 = r5-r1-r2-r3
t. E ke D

5.2. Perancangan Penerapan Perangkat Keras dan Antarmuka

Penerapan perangkat keras dan *interface* pada simulasi ini menggunakan lima buah komputer dan empat buah kabel UTP dengan spesifikasi berikut:

a. Laptop sebagai *Host* A
   - *Processor*: Intel Atom CPU N270, 1.60 GHz
   - RAM (*Random Access Memory*): RAM 1GB

- LAN (*Local Area Network*) *Card*: LAN Card Broadcom NetLink (TM) Fast Ethernet 100 Mbps
- Kapasitas Hard disk: 160 GB

b. Komputer 2 sebagai *Host* B
- *Processor*: Intel (R) Pentium (R) 4, 2.66GHz
- RAM (*Random Access Memory*): RAM 256 MB
- LAN (*Local Area Network*) *Card*: LAN Card Realtek RT8139 100Mbps
- Kapasitas Hard disk: 40 GB

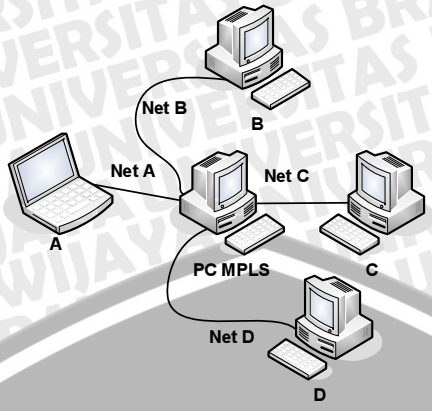
c. Komputer 3 sebagai *Host* C
- *Processor*: Intel (R) Pentium (R) 4, 2.66GHz
- RAM (*Random Access Memory*): RAM 504 MB
- LAN (*Local Area Network*) *Card*: LAN Card Realtek RT8139 100Mbps
- Kapasitas Hard disk: 40 GB

d. Komputer 4 sebagai *Host* D
- *Processor*: Intel (R) Pentium (R) 4, 2.66GHz
- RAM (*Random Access Memory*): RAM 256 MB
- LAN (*Local Area Network*) *Card*: LAN Card Realtek RT8139 100Mbps
- Kapasitas Hard disk: 40 GB

e. Komputer 5 sebagai PC MPLS
- *Processor*: Intel Pentium 4, 2.66 GHz
- RAM (*Random Access Memory*): RAM 2 GB
- Empat buah LAN (*Local Area Network*) *Card*:
  - LAN Card VIA Rhine II Fast Ethernet Adapter 100 Mbps
  - LAN Card Realtek RTL8139 100 Mbps
  - LAN Card Realtek RTL8139 100 Mbps
  - LAN Card Realtek RTL8139 100 Mbps
- Kapasitas Hard disk: 160 GB

f. Empat buah kabel UTP (*Cross*) CAT 5 yang lengkap dengan RJ-45 di kedua ujungnya.

Penerapan perangkat keras dan *interface* secara keseluruhan ditunjukkan dalam Gambar 5.4.



Gambar 5.4. Topologi Simulasi Jaringan MPLS

Sumber: Perancangan

5.3. Perancangan Penerapan Perangkat Lunak

Perangkat lunak yang diterapkan pada sistem ini adalah *software* Packgen 0.2 sebagai pembangkit trafik. Kemudian untuk simulasi jaringan MPLS digunakan *software* emulator jaringan Dynamips dan digunakan VMware sebagai emulator PC.

5.3.1. Perancangan Pembangkitan Trafik

Pembangkitan trafik untuk dialirkan ke dalam jaringan MPLS ini dilakukan oleh *software* Packgen 0.2. Untuk menjalankan pembangkit trafik dengan menggunakan Packgen maka dilakukan langkah-langkah berikut:

1. Perancangan pembangkitan ini dilakukan dengan menentukan karakteristik tiap paket yang akan dikirim. Tabel 5.11 berikut ini menunjukkan rancangan karakteristik paket yang akan dikirim oleh *host* sumber menuju *host* tujuan:

Tabel 5.11. Tabel Karakteristik Paket

Data	DSCP	Besar Paket (byte)	Port	*Bandwidth* P1(bit)	*Bandwidth* P2(bit)
Voice	EF	40	5003	2500	5000
Mission critical	AF31	1000	5000	3750	7500
Bulk	AF11	1000	5001	3125	6250
Best effort	0	1000	5002	3125	6250

Sumber: Perancangan

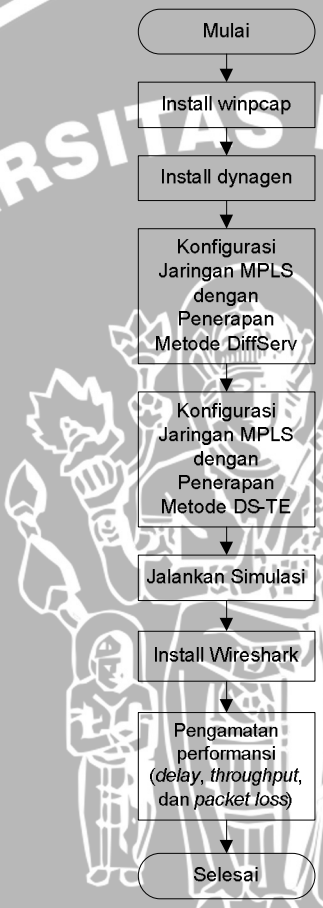
Pada tabel tersebut terdapat dua buah variasi konfigurasi *bandwidth* yang dilakukan untuk dua buah percobaan. Konfigurasi *bandwidth* tersebut dituliskan dengan *bandwidth* P1 dan P2 untuk percobaan pertama dan kedua secara berturut-turut.

2. Untuk mengirimkan paket dari sumber menuju tujuan, Packgen terlebih dahulu perlu melakukan kegiatan mendengarkan (*listen*) pada port-port yang telah

dikonfigurasi. Sehingga untuk mengirimkan paket dari *host* sumber menuju *host* tujuan pertama-tama dijalankan file "listen.yml" pada *host* tujuan terlebih dahulu, kemudian selanjutnya dijalankan file pembangkit trafik pada *host* sumber.

5.3.2. Perancangan Simulator Jaringan MPLS

Diagram alir untuk mengimplementasikan simulasi jaringan MPLS ditunjukkan pada Gambar 5.5.



Gambar 5.5. Diagram Alir Perancangan Simulasi Jaringan MPLS

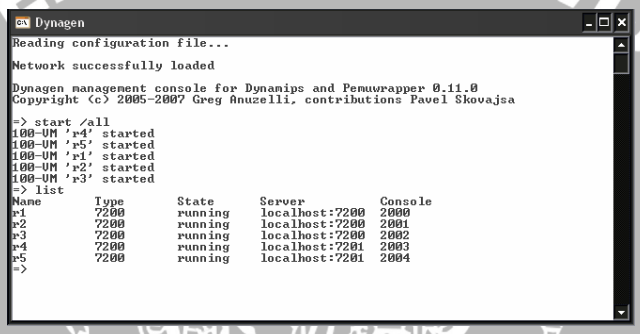
Sumber: Perancangan

Simulasi pada skripsi ini menggunakan *software* emulator jaringan Dynamips yang diinstall pada PC MPLS untuk membuat simulasi jaringan MPLS. Dari diagram alir pada Gambar 5.5, tampak bahwa pertama-tama langkah yang harus dilakukan adalah instalasi winpcap terlebih dahulu, kemudian menginstal Dynagen pada PC MPLS. Langkah berikutnya adalah mengkonfigurasi jaringan MPLS untuk simulasi yang pertama dengan penerapan metode DiffServ. Kemudian selanjutnya mengkonfigurasi jaringan MPLS dengan penerapan metode DS-TE untuk simulasi yang

kedua. Sedangkan *software* Wireshark digunakan untuk menangkap paket-paket dalam jaringan MPLS agar didapatkan data untuk menghitung performansi jaringan MPLS.

Dalam pembuatan konfigurasi jaringan MPLS dengan metode DiffServ dan DS-TE, maka dilakukan langkah-langkah berikut:

1. Membuat sebuah file jaringan Dynagen untuk menerapkan konfigurasi topologi jaringan MPLS dengan lima buah virtual *router*. Konfigurasi jaringan ini dapat dilihat pada Lampiran 1.
2. Menjalankan Dynamips *server*.
3. Menjalankan file Dynagen berekstensi ".net" yang telah dibuat pada poin 1.
4. Menjalankan lima buah virtual *router* yang telah dikonfigurasi sebelumnya pada file Dynagen tersebut. Seperti yang ditunjukkan oleh Gambar 5.6.



Gambar 5.6. Tampilan Dynagen yang Menjalankan Lima Virtual *Router*

5. Mengakses kelima virtual *router* yang telah dijalankan tersebut dengan perintah *telnet* ke tiap *router* melalui Dynagen untuk kemudian diterapkan konfigurasi DiffServ pada masing-masing *router*. Konfigurasi metode DiffServ dan DS-TE untuk tiap *router* dapat dilihat pada Lampiran 2 dan Lampiran 3.

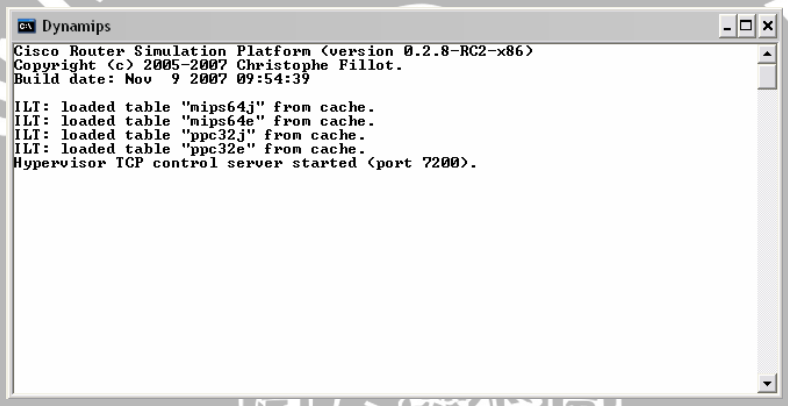
5.3.3. Penerapan Perangkat Lunak pada Sisi *Source*

Yang dimaksud pada sisi *source* ini adalah pada sisi pengirim trafik. Yang dapat merupakan *host* A sampai *host* E. Sistem operasi yang dipakai bervariasi. Semua *host* menggunakan sistem operasi Microsoft Windows XP Professional Version 2002 Service Pack 2 untuk semua *host* kecuali *host* A, *host* B dan *host* E. Host A menggunakan Microsoft Windows XP Professional Version 2002 Service Pack 3. Kemudian *host* B menggunakan Slax yang merupakan varian sistem operasi Linux yang berbasis Slackware. Untuk pembangkitan trafik, digunakan *software* Packgen 0.2 yang dibuat dalam bahasa pemrograman Ruby. Yang dalam hal ini digunakan Ruby 1.8.6. Sedangkan host E menggunakan sistem operasi Ubuntu 8.04. Selain untuk

pembangkitan trafik, *software* Packgen 0.2 digunakan untuk melakukan kegiatan mendengar (*listen*) terhadap port-port yang dikonfigurasi untuk pengiriman paket, serta untuk menampilkan perhitungan *packet loss* yang dialami oleh paket ketika sampai pada *Destination*.

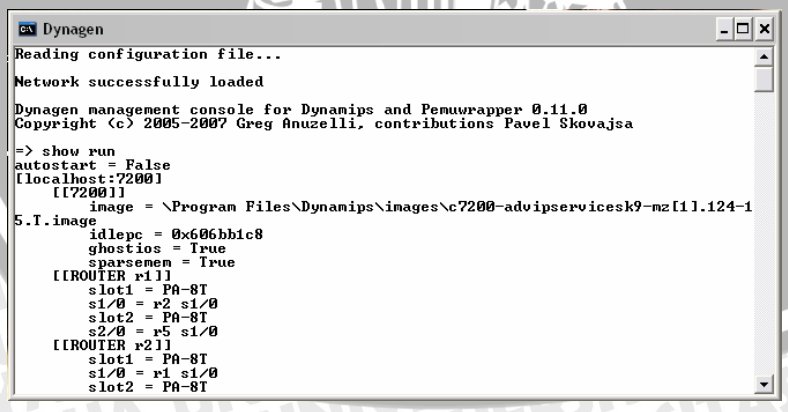
5.3.4. Penerapan Perangkat Lunak pada Sisi PC MPLS

Digunakan Microsoft Windows XP Professional Version 2002 Service Pack 2 sebagai sistem operasinya. Pada PC ini diinstal *software* emulator jaringan Dynamips yang menggunakan Dynagen versi 0.11.0 untuk simulasi jaringan MPLS. Dan diinstal pula *Software* winpcap 4.0 untuk mendukung *network interface card* agar dapat berkomunikasi dengan proses Dynamips. Pada skripsi ini digunakan simulasi untuk Cisco 7200 *series router*.



Gambar 5.7. Tampilan Dynamips *Server*

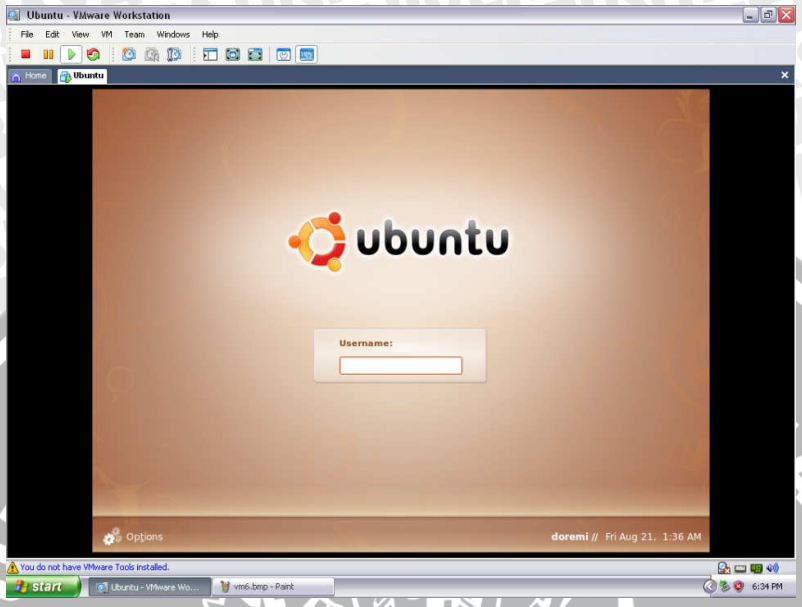
Sumber: Pengujian



Gambar 5.8. Tampilan Dynagen

Sumber: Pengujian

Selain itu pada PC MPLS ini juga diinstal dan dijalankan sebuah vmware yang merupakan emulator yang dapat digunakan untuk mengemulasi sebuah PC, yang dalam hal ini adalah *host* E. Berikut adalah tampilan dari VMware pada PC MPLS.



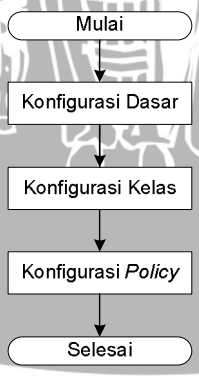
Gambar 5.9. Tampilan VMware pada PC MPLS

Sumber: Pengujian

5.3.5. Algoritma Konfigurasi *Router*

a. Metode DiffServ

Algoritma konfigurasi *router* untuk metode DiffServ memiliki tahapan seperti pada Gambar 5.10. berikut ini.

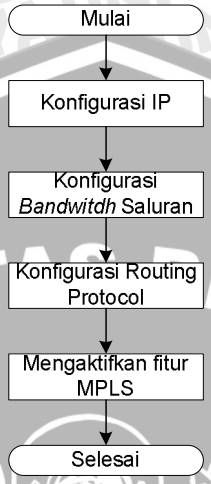


Gambar 5.10. Algoritma Konfigurasi *Router* dengan Metode DiffServ

Sumber: Perancangan

1. Konfigurasi dasar

   Konfigurasi dasar ini meliputi konfigurasi IP untuk tiap antarmuka, konfigurasi protokol *routing* yang dalam hal ini adalah OSPF, konfigurasi untuk mengaktifkan fitur MPLS dan konfigurasi *bandwidth* saluran.

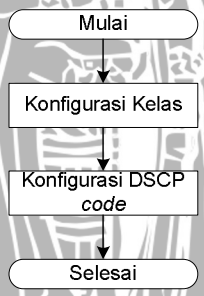


Gambar 5.11. Algoritma Konfigurasi Dasar pada Metode DiffServ

Sumber: Perancangan

2. Konfigurasi kelas

   Yang tercakup pada konfigurasi ini adalah konfigurasi kelas-kelas dan kode DSCP untuk mengklasifikasi dan membedakan trafik yang lewat.

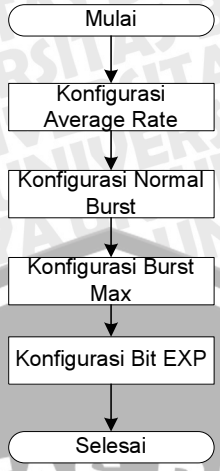


Gambar 5.12. Algoritma Konfigurasi Kelas pada Metode DiffServ

Sumber: Perancangan

3. Konfigurasi *policy*

   Pada konfigurasi ini diterapkan QoS *policy* yang telah ditentukan untuk membatasi aliran trafik yang masuk ke dalam jaringan dalam tiap kelas, serta konfigurasi EXP bit pada label MPLS.

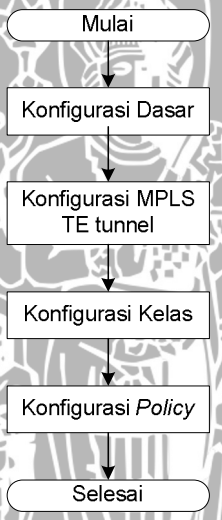


Gambar 5.13. Algoritma Konfigurasi *Policy* pada Metode DiffServ

Sumber: Perancangan

b. Metode DS-TE

Berikut ini diberikan urutan konfigurasi *router* untuk metode DS-TE.

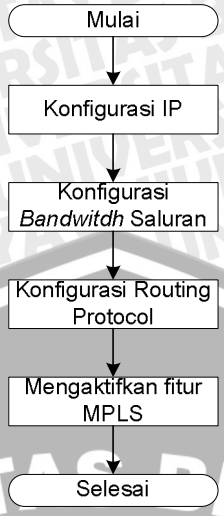


Gambar 5.14. Algoritma Konfigurasi *Router* dengan Metode DS-TE

Sumber: Perancangan

1. Konfigurasi dasar

   Konfigurasi dasar ini meliputi konfigurasi IP untuk tiap antarmuka, konfigurasi protokol *routing*, konfigurasi untuk mengaktifkan fitur MPLS dan konfigurasi *bandwidth* saluran.

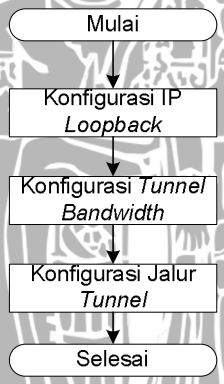


Gambar 5.15. Algoritma Konfigurasi Dasar pada Metode DS-TE

Sumber: Perancangan

2. Konfigurasi MPLS TE *tunnel*

   Disini dikonfigurasi MPLS TE *tunnel*, konfigurasi IP *loopback* sebagai identitas *router* dalam metode TE, mengkonfigurasi *bandwidth* untuk tiap *tunnel*, dan konfigurasi jalur *tunnel*.

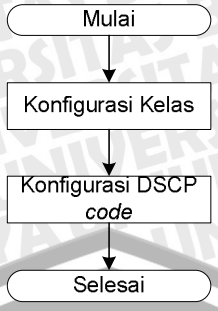


Gambar 5.16. Algoritma Konfigurasi MPLS TE *Tunnel* pada Metode DS-TE

Sumber: Perancangan

3. Konfigurasi kelas

   Yang tercakup pada konfigurasi ini adalah konfigurasi kelas-kelas dan kode DSCP untuk mengklasifikasi dan membedakan trafik yang lewat.

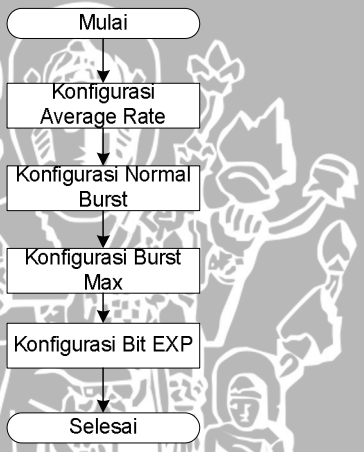


Gambar 5.17. Algoritma Konfigurasi Kelas pada Metode DS-TE

Sumber: Perancangan

4. Konfigurasi *policy*

   Pada konfigurasi ini diterapkan QoS *policy* yang telah ditentukan untuk membatasi aliran trafik yang masuk ke dalam jaringan dalam tiap kelas, serta konfigurasi EXP bit pada label MPLS.



Gambar 5.18. Algoritma Konfigurasi *Policy* pada Metode DS-TE

Sumber: Perancangan

5.4. Pengujian dan Analisis

Subbab ini berisi penjelasan prosedur pengujian dari jaringan simulasi yang telah dirancang guna mengetahui apakah jaringan simulasi dapat bekerja sesuai dengan perancangan dan konfigurasi yang diterapkan berjalan dengan baik. Sedang analisis dilakukan untuk mengetahui performansi jaringan simulasi yang dirancang.

Proses pengujian dilakukan untuk beberapa bagian konfigurasi *router* dan pengujian jaringan secara keseluruhan. Pelaksanaan pengujian sistem adalah sebagai berikut:

1. Pengujian terhadap konfigurasi IP dengan perintah "*ping*" dan "*tracert*" yang dilakukan dari *host* tujuan menuju *host* sumber.

Hasil penerapan perintah ini pada *host* C (192.16.10.3) menuju *host* A ditunjukkan dalam Gambar 5.19 dan Gambar 5.20.

```
C:\Documents and Settings\d>ping 192.16.10.35

Pinging 192.16.10.35 with 32 bytes of data:

Reply from 192.16.10.35: bytes=32 time=921ms TTL=60
Reply from 192.16.10.35: bytes=32 time=539ms TTL=60
Reply from 192.16.10.35: bytes=32 time=601ms TTL=60
Reply from 192.16.10.35: bytes=32 time=215ms TTL=60

Ping statistics for 192.16.10.35:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 215ms, Maximum = 921ms, Average = 569ms
```

Gambar 5.19. Hasil Keluaran Perintah "ping 192.16.10.35" pada Destination

Sumber: Hasil Pengujian

```
C:\Documents and Settings\d>tracert 192.16.10.35

Tracing route to 192.16.10.35 over a maximum of 30 hops

  1    121 ms     47 ms     73 ms  192.16.10.1
  2    334 ms      *       255 ms  192.16.10.89
  3    659 ms    543 ms    434 ms  192.16.10.85
  4    561 ms    405 ms    470 ms  192.16.10.35

Trace complete.
```

Gambar 5.20. Hasil Keluaran Perintah "tracert 192.16.10.35" pada Destination

Sumber: Hasil Pengujian

2. Pengujian terhadap konfigurasi *routing* dengan perintah "*show ip route*"

   Pengujian ini berhasil dilakukan. Hasil pengujian menunjukkan bahwa protokol OSPF aktif dan semua jaringan dikenali. Hasil pengujian perintah ini pada r1 dalam jaringan DiffServ ditunjukkan dalam Gambar 5.21.

```
Password:
r1#sh ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     192.16.10.0/24 is variably subnetted, 10 subnets, 4 masks
O       192.16.10.88/30 [110/2000] via 192.16.10.86, 00:03:41, Serial1/0
O       192.16.10.92/30 [110/3000] via 192.16.10.86, 00:02:41, Serial1/0
                        [110/3000] via 192.16.10.81, 00:01:16, Serial2/0
C       192.16.10.80/30 is directly connected, Serial2/0
C       192.16.10.84/30 is directly connected, Serial1/0
O       192.16.10.72/29 [110/2500] via 192.16.10.81, 00:01:16, Serial2/0
O       192.16.10.64/29 [110/1500] via 192.16.10.86, 00:04:44, Serial1/0
O       192.16.10.96/30 [110/2000] via 192.16.10.81, 00:01:16, Serial2/0
O       192.16.10.0/27 [110/2500] via 192.16.10.86, 00:03:16, Serial1/0
O       192.16.10.48/28 [110/1500] via 192.16.10.81, 00:01:16, Serial2/0
C       192.16.10.32/28 is directly connected, FastEthernet0/0
r1#_
```

Gambar 5.21. Hasil Keluaran dari Perintah "show ip route" pada r1

Sumber: Hasil Pengujian

Dan hasil pengujian perintah ini pada r1 dalam jaringan DS-TE ditunjukkan dalam Gambar 5.22.

```
r1#sh ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     17.0.0.0/32 is subnetted, 5 subnets
C       17.16.10.11 is directly connected, Loopback0
O       17.16.10.22 [110/1001] via 192.16.10.86, 00:41:00, Serial1/0
O       17.16.10.44 [110/2001] via 192.16.10.81, 00:41:10, Serial2/0
O       17.16.10.33 [110/2001] via 192.16.10.86, 00:41:00, Serial1/0
O       17.16.10.55 [110/1001] via 192.16.10.81, 00:41:10, Serial2/0
     192.16.10.0/24 is variably subnetted, 7 subnets, 3 masks
O       192.16.10.88/30 [110/2000] via 192.16.10.86, 00:41:00, Serial1/0
O       192.16.10.92/30 [110/3000] via 192.16.10.86, 00:41:00, Serial1/0
                        [110/3000] via 192.16.10.81, 00:41:10, Serial2/0
C       192.16.10.80/30 is directly connected, Serial2/0
C       192.16.10.84/30 is directly connected, Serial1/0
O       192.16.10.96/30 [110/2000] via 192.16.10.81, 00:41:10, Serial2/0
S       192.16.10.0/27 is directly connected, Tunnel2
C       192.16.10.32/28 is directly connected, FastEthernet0/0
```

Gambar 5.22. Hasil Keluaran Perintah “show ip route” pada r1

Sumber: Hasil Pengujian

3. Pengujian terhadap konfigurasi MPLS dengan ”*show mpls forwarding-table*”

   Pengujian ini berhasil dilakukan. Label yang diberikan untuk tiap prefiks tercatat dalam tabel forwarding MPLS. Hasil pengujian perintah ini pada r3 ditunjukkan dalam Gambar 5.23.

```
r3#sh mpls forwarding-table
Local  Outgoing    Prefix           Bytes tag  Outgoing    Next Hop
tag    tag or VC   or Tunnel Id     switched   interface
16     16          192.16.10.32/28  52         Se1/0       point2point
17     Pop tag     192.16.10.64/29  0          Se1/0       point2point
18     Pop tag     192.16.10.84/30  0          Se1/0       point2point
19     Pop tag     192.16.10.72/29  0          Se2/0       point2point
20     21          192.16.10.48/28  0          Se2/0       point2point
21     22          192.16.10.80/30  0          Se2/0       point2point
       21          192.16.10.80/30  0          Se1/0       point2point
22     Pop tag     192.16.10.96/30  0          Se2/0       point2point
```

Gambar 5.23. Hasil Keluaran Perintah “show mpls forwarding-table” pada r1

Sumber: Hasil Pengujian

4. Pengujian terhadap konfigurasi MPLS DiffServ dengan perintah ”*show policy-map interface*”

   Pengujian ini berhasil dilakukan. Hasilnya dapat diketahui konfigurasi yang diterapkan untuk tiap kelas (*gold*, *silver* dan *bronze*). Hasil pengujian perintah ini untuk kelas gold dan silver pada r1 ditunjukkan dalam Gambar 5.24.

```
Service-policy output: PE-P

  Class-map: gold (match-all)
    0 packets, 0 bytes
    5 minute offered rate 0 bps, drop rate 0 bps
    Match: mpls experimental topmost 5
    Queueing
      Strict Priority
      Output Queue: Conversation 40
      Bandwidth 20 (%)
      Bandwidth 20 (kbps) Burst 500 (Bytes)
      (pkts matched/bytes matched) 1/44
      (total drops/bytes drops) 0/0

  Class-map: silver (match-all)
    0 packets, 0 bytes
    5 minute offered rate 0 bps, drop rate 0 bps
    Match: mpls experimental topmost 3
    Queueing
      Output Queue: Conversation 41
      Bandwidth 25 (%)
      Bandwidth 25 (kbps)
      (pkts matched/bytes matched) 0/0
      (depth/total drops/no-buffer drops) 0/0/0
       exponential weight: 9
       mean queue depth: 0
```

Gambar 5.24. Hasil Keluaran dari Perintah “policy-map interface” pada r1

Sumber: Hasil Pengujian

5. Pengujian terhadap konfigurasi MPLS TE dengan perintah ”*show mpls traffic-eng tunnels*”.

   Pengujian ini berhasil dilakukan. Dengan bukti bahwa kedua *tunnel* yang dikonfigurasi aktif dan terhubung. Hasil pengujian untuk kedua tunnel yang bersumber pada r1 menuju r3 ini ditunjukkan dalam Gambar 5.25.

```
Name: r1_t1                               (Tunnel1) Destination: 17.16.10.33
  Status:
    Admin: up         Oper: up     Path: valid       Signalling: connected

    path option 1, type dynamic (Basis for Setup, path weight 2000)

  Config Parameters:
    Bandwidth: 75       kbps (Sub)  Priority: 0  0   Affinity: 0x0/0xFFFF
    Metric Type: TE (default)
    AutoRoute:  disabled  LockDown: disabled  Loadshare: 75       bw-based
    auto-bw: disabled

  InLabel  :  -
  OutLabel : Serial1/0, 26
  RSVP Signalling Info:
       Src 17.16.10.11, Dst 17.16.10.33, Tun_Id 1, Tun_Instance 109
    RSVP Path Info:
      My Address: 17.16.10.11
      Explicit Route: 192.16.10.86 192.16.10.90 17.16.10.33
      Record Route:  NONE
      Tspec: ave rate=75 kbits, burst=1000 bytes, peak rate=75 kbits
    RSVP Resv Info:
      Record Route:  NONE
      Fspec: ave rate=75 kbits, burst=1000 bytes, peak rate=75 kbits
  History:
    Tunnel:
      Time since created: 1 hours, 57 minutes
      Time since path change: 30 minutes, 51 seconds
    Current LSP:
      Uptime: 30 minutes, 51 seconds
    Prior LSP:
      ID: path option 1 [100]
      Removal Trigger: path verification failed

Name: r1_t2                               (Tunnel2) Destination: 17.16.10.33
  Status:
    Admin: up         Oper: up     Path: valid       Signalling: connected

    path option 1, type explicit bottom (Basis for Setup, path weight 3000)

  Config Parameters:
    Bandwidth: 75       kbps (Global)  Priority: 0  0   Affinity: 0x0/0xFFFF
    Metric Type: TE (default)
    AutoRoute:  disabled  LockDown: disabled  Loadshare: 75       bw-based
    auto-bw: disabled

  InLabel  :  -
  OutLabel : Serial2/0, 16
  RSVP Signalling Info:
       Src 17.16.10.11, Dst 17.16.10.33, Tun_Id 2, Tun_Instance 13
    RSVP Path Info:
      My Address: 17.16.10.11
      Explicit Route: 192.16.10.81 192.16.10.97 192.16.10.93 17.16.10.33
      Record Route:  NONE
      Tspec: ave rate=75 kbits, burst=1000 bytes, peak rate=75 kbits
    RSVP Resv Info:
      Record Route:  NONE
      Fspec: ave rate=75 kbits, burst=1000 bytes, peak rate=75 kbits
  History:
    Tunnel:
      Time since created: 1 hours, 57 minutes
      Time since path change: 1 hours, 55 minutes
    Current LSP:
      Uptime: 1 hours, 55 minutes
```

Gambar 5.25. Hasil Keluaran Perintah “show mpls traffic-eng tunnels” pada r1

Sumber: Hasil Pengujian

6. Pengujian terhadap konfigurasi MPLS DS-TE ”*show mpls traffic-eng topology*”

   Pengujian ini berhasil dilakukan. Perintah ini menampilkan konfigurasi *bandwidth* untuk *subpool* dan *global pool*. Hasil pengujian untuk topology pada r1 ditunjukkan dalam Gambar 5.26.

```
r1#sh mpls traffic-eng topology
My_System_id: 17.16.10.1, Globl Link Generation 111
Signalling error holddown: 10 sec

IGP Id: 17.16.10.1, MPLS TE Id:17.16.10.11 Router Node
      link[0 ]:Nbr IGP Id: 17.16.10.5, gen:108
          frag_id 1, Intf Address:192.16.10.82, Nbr Intf Address:192.16.10.81
          TE metric:1000, IGP metric:1000, attribute_flags:0x0
          physical_bw: 100 (kbps), max_reservable_bw_global: 100 (kbps)
          max_reservable_bw_sub: 30 (kbps)

                                Global Pool       Sub Pool
                Total Allocated Reservable        Reservable
                BW (kbps)       BW (kbps)         BW (kbps)
                --------------- ----------        ----------
        bw[0]:          75              25                25
        bw[1]:           0              25                25
        bw[2]:           0              25                25
        bw[3]:           0              25                25
        bw[4]:           0              25                25
        bw[5]:           0              25                25
        bw[6]:           0              25                25
        bw[7]:           0              25                25

      link[1 ]:Nbr IGP Id: 17.16.10.2, gen:111
          frag_id 0, Intf Address:192.16.10.85, Nbr Intf Address:192.16.10.86
          TE metric:1000, IGP metric:1000, attribute_flags:0x0
          physical_bw: 100 (kbps), max_reservable_bw_global: 100 (kbps)
          max_reservable_bw_sub: 75 (kbps)

                                Global Pool       Sub Pool
                Total Allocated Reservable        Reservable
                BW (kbps)       BW (kbps)         BW (kbps)
                --------------- ----------        ----------
        bw[0]:          75              25                 0
        bw[1]:           0              25                 0
        bw[2]:           0              25                 0
        bw[3]:           0              25                 0
        bw[4]:           0              25                 0
        bw[5]:           0              25                 0
        bw[6]:           0              25                 0
        bw[7]:           0              25                 0
```

Gambar 5.26. Hasil keluaran dari Perintah “show mpls traffic-eng topology” pada r1

Sumber: Hasil Pengujian

7. Pengujian keseluruhan jaringan

   Pengujian dilakukan dengan membangkitkan data pada *Source* untuk kemudian dikirimkan ke *Destination*. Pengujian ini berhasil dilakukan. *Destination* berhasil menerima semua paket yang dikirim dari *Source*. Pada Gambar 5.27 berikut ini ditunjukkan hasil pengiriman paket yang dilakukan oleh *Source* dan pada Gambar 5.28 ditunjukkan hasil penerimaan paket pada *Destination*.

```
root@slax:/mnt/hda5/c/packgen-0.2/examples# packgen -i reverse00-.yml
Sun May 03 12:33:44 +0700 2009:
    Listening on udp port 5003 for 62.0 seconds.
Sun May 03 12:33:44 +0700 2009:
    Listening on tcp port 5000 for 62.0 seconds.
Sun May 03 12:33:44 +0700 2009:
    Listening on tcp port 5001 for 62.0 seconds.
Sun May 03 12:33:44 +0700 2009:
    Listening on tcp port 5002 for 62.0 seconds.
Sun May 03 12:33:45 +0700 2009:
    Sending tcp packets of 1000B to 192.16.10.3:5000 (mission critical) at 12.50Kb/s.
Sun May 03 12:33:55 +0700 2009:
    Sending tcp packets of 1000B to 192.16.10.3:5001 (bulk) at 15.00Kb/s.
Sun May 03 12:34:05 +0700 2009:
    Sending tcp packets of 1000B to 192.16.10.3:5002 (best effort) at 12.50Kb/s.
Sun May 03 12:34:15 +0700 2009:
    Sending udp packets of 40B to 192.16.10.3:5003 (voice) at 10.00Kb/s.
Sun May 03 12:34:45 +0700 2009:
    Sent 937 udp packets (voice).
Sun May 03 12:34:46 +0700 2009:
    Not listening on port 5002 anymore.
Sun May 03 12:34:46 +0700 2009:
    Not listening on port 5001 anymore.
Sun May 03 12:34:46 +0700 2009:
    Not listening on port 5000 anymore.
Sun May 03 12:34:46 +0700 2009:
    Not listening on port 5003 anymore.
Sun May 03 12:34:48 +0700 2009:
    Sent 54 tcp packets (mission critical).
Sun May 03 12:34:48 +0700 2009:
    Sent 44 tcp packets (bulk).
Sun May 03 12:34:49 +0700 2009:
    Sent 16 tcp packets (best effort).
root@slax:/mnt/hda5/c/packgen-0.2/examples#
```

Gambar 5.27. Pengiriman Data pada *Source*

Sumber: Hasil Pengujian

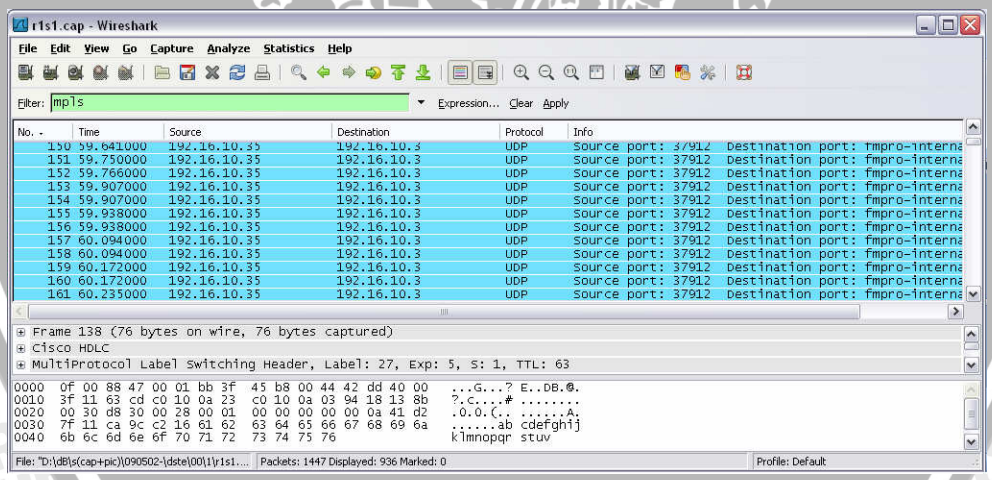
```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\d>d:

D:\>cd packgen-0.2\bin

D:\packgen-0.2\bin>ruby packgen -i ..\examples\listenr.yml
Sun May 03 12:34:58 +0700 2009:
    Listening on udp port 5003.
Sun May 03 12:34:58 +0700 2009:
    Listening on tcp port 5000.
Sun May 03 12:34:58 +0700 2009:
    Listening on tcp port 5001.
Sun May 03 12:34:58 +0700 2009:
    Listening on tcp port 5002.
Sun May 03 12:36:05 +0700 2009:
    voice => Packets received: 398 Packets lost: 539 (57.5240 %)
Sun May 03 12:36:05 +0700 2009:
    mission critical => Packets received: 54 Packets lost: 0 (0.0000 %)
Sun May 03 12:37:03 +0700 2009:
    bulk => Packets received: 44 Packets lost: 0 (0.0000 %)
Sun May 03 12:37:55 +0700 2009:
    best effort => Packets received: 16 Packets lost: 0 (0.0000 %)
```

Gambar 5.28. Penerimaan Data pada *Destination*

Sumber: Hasil Pengujian

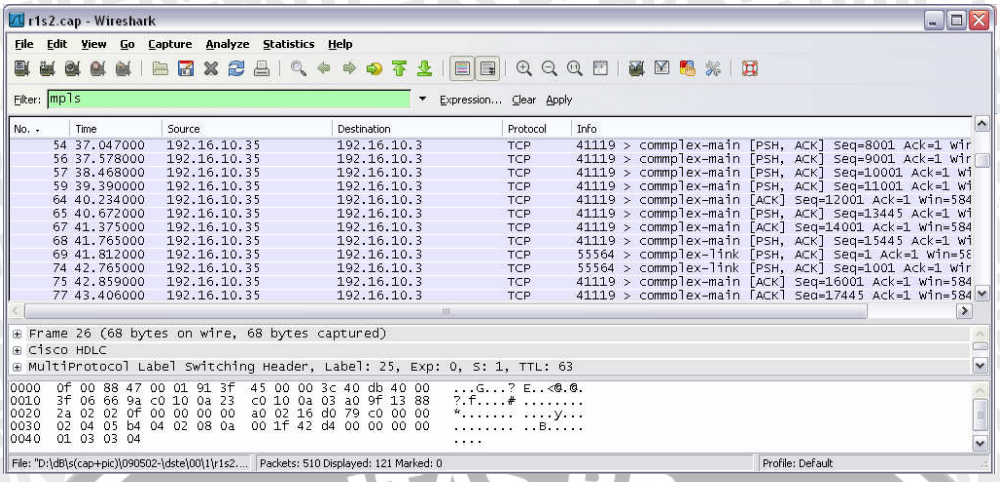
8. Pengujian jalur *tunnel* yang dilalui trafik UDP dan TCP pada simulasi jaringan MPLS dengan metode DS-TE.

   Pengujian dilakukan dengan menangkap (*capture*) paket-paket yang lewat pada antarmuka S1/0 dan S2/0 pada r1. Pengujian yang dilakukan berhasil dengan bukti bahwa paket MPLS dengan protokol transport UDP yang dikonfigurasi pada Packgen keluar melalui antarmuka S1/0, dan paket MPLS dengan protokol transport TCP yang dikonfigurasi pada Packgen keluar melalui antarmuka S2/0. Hal ini ditunjukkan pada Gambar 5.29 dan 5.30.



Gambar 5.29. Paket MPLS pada Antarmuka S1/0 pada r1

Sumber: Hasil Pengujian



Gambar 5.30. Paket MPLS pada Antarmuka S2/0 pada r1

Sumber: Hasil Pengujian

5.5. Hasil Simulasi

Hasil pengamatan yang diperoleh berupa *delay end-to-end*, jumlah paket yang dikirim, jumlah *packet loss* serta jumlah data yang diterima. Pengambilan data pada jaringan MPLS dengan penerapan metode DiffServ yang dibandingkan dengan metode DS-TE dilakukan untuk percobaan P1 dan P2. Berikut adalah tabel pengiriman data dengan metode DiffServ dan metode DS-TE.

Tabel 5.12 Pengambilan Data pada P1 untuk Metode DiffServ

Jalur	**Kelas**	***Delay end to end* (detik) $t_{end\ to\ end}$**	**Paket yang dikirim (paket) ΣP**	***Packet loss* (paket) α**
A-C	EF	110	234	98
A-B	AF31	140	16	0
A-D	AF11	188	9	0
A-E	BE	230	6	0
B-D	EF	119	234	101
B-C	AF31	149	15	0
B-E	AF11	140	8	0
B-A	BE	161	7	0
C-E	EF	120	234	98
C-D	AF31	149	16	0
C-A	AF11	149	10	0
C-B	BE	220	4	0
D-A	EF	93	234	93
D-E	AF31	119	16	0
D-B	AF11	109	9	0
D-C	BE	199	8	0

E-B	EF	121	234	116
E-A	AF31	154	12	0
E-C	AF11	149	8	0
E-D	BE	191	5	0

Sumber : Hasil Percobaan

Tabel 5.13 Pengambilan Data pada P1 untuk metode DS-TE

Jalur	**Kelas**	***Delay end to end* (detik)** $t_{end\ to\ end}$	**Paket yang dikirim (paket)** **ΣP**	***Packet loss* (paket)** **α**
A-C	EF	164	234	88
A-B	AF31	191	13	0
A-D	AF11	182	6	0
A-E	BE	303	5	0
B-D	EF	162	234	99
B-C	AF31	192	9	0
B-E	AF11	182	5	0
B-A	BE	363	3	0
C-E	EF	168	234	107
C-D	AF31	198	14	0
C-A	AF11	188	8	0
C-B	BE	279	4	0
D-A	EF	145	234	97
D-E	AF31	174	13	0
D-B	AF11	164	9	0
D-C	BE	159	4	0
E-B	EF	152	234	99
E-A	AF31	182	12	0
E-C	AF11	172	8	0
E-D	BE	420	5	0

Sumber : Hasil Percobaan

Tabel 5.14 Pengambilan Data pada P2 untuk metode DiffServ

Jalur	**Kelas**	***Delay end to end* (detik)** $t_{end\ to\ end}$	**Paket yang dikirim (paket)** **ΣP**	***Packet loss* (paket)** **α**
A-C	EF	115	468	200
A-B	AF31	145	26	0
A-D	AF11	135	44	0
A-E	BE	199	10	0
B-D	EF	116	468	167
B-C	AF31	146	24	0
B-E	AF11	135	41	0
B-A	BE	321	7	0
C-E	EF	119	468	188
C-D	AF31	148	18	0

C-A	AF11	138	44	0
C-B	BE	191	6	0
D-A	EF	113	468	181
D-E	AF31	143	23	0
D-B	AF11	175	38	0
D-C	BE	199	11	0
E-B	EF	143	468	173
E-A	AF31	218	22	0
E-C	AF11	133	22	0
E-D	BE	123	9	0

Sumber : Hasil Percobaan

Tabel 5.13 Pengambilan Data pada P2 untuk metode DS-TE

Jalur	**Kelas**	***Delay end to end* (detik)** $t_{end\ to\ end}$	**Paket yang dikirim (paket) ΣP**	***Packet loss* (paket) α**
A-C	EF	170	468	304
A-B	AF31	200	56	0
A-D	AF11	189	31	0
A-E	BE	208	16	0
B-D	EF	157	468	297
B-C	AF31	187	54	0
B-E	AF11	178	38	0
B-A	BE	252	16	0
C-E	EF	139	468	292
C-D	AF31	169	53	0
C-A	AF11	159	32	0
C-B	BE	150	17	0
D-A	EF	143	468	315
D-E	AF31	239	49	0
D-B	AF11	160	33	0
D-C	BE	155	22	0
E-B	EF	192	468	276
E-A	AF31	224	50	0
E-C	AF11	213	36	0
E-D	BE	359	18	0

Sumber : Hasil Percobaan

5.6. *Throughput*

Throughput dihitung dengan membagi jumlah paket yang diterima dengan *delay end to end*. Maka nilai *throughput* untuk trafik *voice* P1 pada metode DiffServ adalah sebagai berikut:

$$t_v = \frac{110}{136} = 0{,}8088\ detik$$

$$\lambda = \frac{1}{0{,}8088} = 1{,}2364\ paket/detik$$

(Evi Syarvianti; 2005:62)

Nilai *throughput* untuk percobaan P1 dan P2 terhadap metode DiffServ dan DS-TE ditunjukkan dalam Tabel 5.16 sampai 5.19 berikut ini.

Tabel 5.16 Hasil Perhitungan Nilai *Throughput* pada P1 untuk Metode DiffServ

Jalur	**Kelas**	***delay end to end* (detik)** $t_{end\ to\ end}$	**Paket yang diterima (paket) ΣP**	***throughput* (paket/detik) λ**
A-C	EF	110	136	1,2364
A-B	AF31	140	16	0,1143
A-D	AF11	188	9	0,0479
A-E	BE	230	6	0,0261
B-D	EF	119	133	1,1176
B-C	AF31	149	15	0,1007
B-E	AF11	140	8	0,0571
B-A	BE	161	7	0,0435
C-E	EF	120	136	1,1333
C-D	AF31	149	16	0,1074
C-A	AF11	149	10	0,0671
C-B	BE	220	4	0,0182
D-A	EF	93	141	1,5161
D-E	AF31	119	16	0,1345
D-B	AF11	109	9	0,0826
D-C	BE	199	8	0,0402
E-B	EF	121	118	0,9752
E-A	AF31	154	12	0,0779
E-C	AF11	149	8	0,0537
E-D	BE	191	5	0,0262

Sumber: Hasil Perhitungan

Tabel 5.17 Hasil Perhitungan Nilai *Throughput* pada P1 untuk metode DS-TE

Jalur	**Kelas**	***delay end to end* (detik)** $t_{end\ to\ end}$	**Paket yang diterima (paket) ΣP**	***throughput* (paket/detik) λ**
A-C	EF	164	146	0,8902
A-B	AF31	191	13	0,0681
A-D	AF11	182	6	0,0330
A-E	BE	303	5	0,0165
B-D	EF	162	135	0,8333
B-C	AF31	192	9	0,0469
B-E	AF11	182	5	0,0275
B-A	BE	363	3	0,0083
C-E	EF	168	127	0,7560

C-D	AF31	198	14	0,0707
C-A	AF11	188	8	0,0426
C-B	BE	279	4	0,0143
D-A	EF	145	137	0,9448
D-E	AF31	174	13	0,0747
D-B	AF11	164	9	0,0549
D-C	BE	159	4	0,0252
E-B	EF	152	135	0,8882
E-A	AF31	182	12	0,0659
E-C	AF11	172	8	0,0465
E-D	BE	420	5	0,0119

Sumber : Hasil Perhitungan

Tabel 5.18 Hasil Perhitungan Nilai *Throughput* pada P2 untuk metode DiffServ

Jalur	**Kelas**	***delay end to end* (detik)** $t_{end\ to\ end}$	**Paket yang diterima (paket) ΣP**	***throughput* (paket/detik) λ**
A-C	EF	115	268	2,3304
A-B	AF31	145	26	0,1793
A-D	AF11	135	44	0,3259
A-E	BE	199	10	0,0503
B-D	EF	116	301	2,5948
B-C	AF31	146	24	0,1644
B-E	AF11	135	41	0,3037
B-A	BE	321	7	0,0218
C-E	EF	119	280	2,3529
C-D	AF31	148	18	0,1216
C-A	AF11	138	44	0,3188
C-B	BE	191	6	0,0314
D-A	EF	113	287	2,5398
D-E	AF31	143	23	0,1608
D-B	AF11	175	38	0,2171
D-C	BE	199	11	0,0553
E-B	EF	143	295	2,0629
E-A	AF31	218	22	0,1009
E-C	AF11	133	22	0,1654
E-D	BE	123	9	0,0732

Sumber : Hasil Perhitungan

Tabel 5.19 Hasil Perhitungan Nilai *Throughput* pada P2 untuk metode DS-TE

Jalur	Kelas	*delay end to end* (detik) $t_{end\ to\ end}$	Paket yang diterima (paket) ΣP	*throughput* (paket/detik) λ
A-C	EF	170	164	0,9647
A-B	AF31	200	56	0,2800
A-D	AF11	189	31	0,1640
A-E	BE	208	16	0,0769
B-D	EF	157	171	1,0892
B-C	AF31	187	54	0,2888
B-E	AF11	178	38	0,2135
B-A	BE	252	16	0,0635
C-E	EF	139	176	1,2662
C-D	AF31	169	53	0,3136
C-A	AF11	159	32	0,2013
C-B	BE	150	17	0,1133
D-A	EF	143	153	1,0699
D-E	AF31	239	49	0,2050
D-B	AF11	160	33	0,2063
D-C	BE	155	22	0,1419
E-B	EF	192	192	1,0000
E-A	AF31	224	50	0,2232
E-C	AF11	213	36	0,1690
E-D	BE	359	18	0,0501

Sumber : Hasil Perhitungan

5.7. Analisis Perhitungan Teoretis

Sebelum melakukan perhitungan *delay*, maka perlu dihitung terlebih dahulu jumlah paket untuk tiap data. Perhitungan dilakukan dengan menggunakan Persamaan (5-5) mengacu pada karakteristik paket pada Tabel 5.11 dengan *range* waktu dan besar paket dalam bit yang ditunjukkan pada hasil perhitungan pada Tabel 5.20. Berikut ini adalah contoh perhitungan jumlah paket untuk P1.

$$jumlah\ paket = \left(\frac{bandwidth\ percobaan P1}{besar\ paket}\right) x\ range\ waktu \quad (5\text{-}5)$$

$$= \frac{2500}{320} x30 = 234{,}375\ \text{paket}$$

Dengan menggunakan cara yang sama, maka didapatkan jumlah paket untuk tiap kelas pada masing-masing percobaan, yaitu P1 dan P2 seperti yang ditunjukkan oleh Tabel 5.20.

Tabel 5.20 Karakteristik Paket

Data	Range Waktu (detik)	Besar Paket (bit)	Jumlah Paket yang Dikirim	
			P1 (paket)	P2 (paket)
Voice	30	320	234,375	468,75
Mission critical	60	8000	28,125	56,25
Bulk	50	8000	19,5313	39,0625
Best effort	40	8000	15,625	31,25

Sumber : Perancangan

Sedangkan perhitungan *packet loss* dibawah ini didapatkan dari pengalian antara jumlah packet yang dikirim dan probabilitas *error* pada TCP/IP yaitu $\rho_b = 10^{-5}$.

Tabel 5.21 *Packet Loss*

Data	Jumlah Paket yang Dikirim		*Packet Loss*	
	P1 (paket)	P2 (paket)	P1 (paket)	P2 (paket)
Voice	234,375	468,75	0,00234375	0,0046875
Mission critical	28,125	56,25	0,00028125	0,0005625
Bulk	19,5313	39,0625	0,000195313	0,000390625
Best effort	15,625	31,25	0,00015625	0,0003125

Sumber : Perancangan

5.7.2. Analisis *Throughput*

Throughput didefinisikan sebagai jumlah paket yang diterima di sisi penerima dengan benar setiap satuan waktu. *Throughput* jaringan didapat tersebut ditentukan dengan persamaan dibawah ini.

Untuk perhitungan *throughput* dengan penyisipan label MPLS pada data *mission critical* adalah sebagai berikut

$$\rho = (L + L')\rho_b = (1060x8)10^{-5} = 0{,}848$$

$$t_1 = \frac{L + L'}{C} = \frac{1060x8}{15000} = 0{,}5653s$$

$$t_{out} = t_{prop} + 2t_1 = 99{,}502.10^{-8} + 2(0{,}5653) = 1{,}1307s$$

$$\alpha = 1 + \frac{t_{out}}{t_1} = 1 + \frac{1{,}1307}{0{,}5653} = 3{,}00000176$$

Dan nilai dari *throughput* didapat dari persamaan (3-1)

$$\lambda = \frac{1}{t_v} = \frac{(1-\rho)}{t_1(1 + (\alpha - 1)\rho)}$$

$$\lambda = \frac{(1 - 0{,}848)}{0{,}5653(1 + (3{,}00000176 - 1), 848)}$$

$\lambda = 1{,}3841$ paket/detik

Tabel 5.22 *Throughput* Ethernet pada P1

***Bandwidth* P1 (bit)**	**Data**	**ρ**	**t_I (detik)**	**t_{out} (setik)**	**α**	**λ (paket/detik)**
2500	*Voice*	0,00656	0,2624	0,5248	3,000003792	3,7369
3750	*Mission critical*	0,08528	2,2741	4,5483	3,000000438	0,3436
3125	*Bulk*	0,08528	2,72896	5,4579	3,000000365	0,2864
3125	*Best effort*	0,08528	2,72896	5,4579	3,000000365	0,2864

Sumber : Hasil Perhitungan

Tabel 5.23 *Throughput* Ethernet pada P2

***Bandwidth* P1 (bit)**	**Data**	**ρ**	**t_I (detik)**	**t_{out} (setik)**	**α**	**λ (paket/detik)**
5000	*Voice*	0,00082	0,0164	0,0328	3,000060672	60,8259
7500	*Mission critical*	0,01066	0,1421	0,2843	3,000007001	6,81539
6250	*Bulk*	0,01066	0,17056	0,3411	3,000005834	5,6795
6250	*Best effort*	0,01066	0,17056	0,3411	3,000005834	5,6795

Sumber : Hasil Perhitungan

Tabel 5.24 *Throughput* dengan Label MPLS

***Bandwidth* P1 (bit)**	**Data**	**ρ**	**t_I (detik)**	**t_{out} (setik)**	**α**	**λ (paket/detik)**
10000	*Voice*	0,00608	0,0608	0,1216	3,000016365	16,1509
15000	*Mission critical*	0,0848	0,5653	1,1307	3,00000176	1,3841
12500	*Bulk*	0,0848	0,6784	1,3568	3,000001467	1,1534
12500	*Best effort*	0,0848	0,6784	1,3568	3,000001467	1,1534

Sumber : Hasil Perhitungan

Tabel 5.25 *Throughput* Tanpa Label MPLS

***Bandwidth* P1 (bit)**	**Data**	**ρ**	**t_I (detik)**	**t_{out} (setik)**	**α**	**λ (paket/detik)**
10000	*Voice*	0,00576	0,0576	0,1152	3,000017275	17,0645
15000	*Mission critical*	0,08448	0,5632	1,1264	3,000001767	1,3906
12500	*Bulk*	0,08448	0,67584	1,3517	3,000001472	1,1588
12500	*Best effort*	0,08448	0,67584	1,3517	3,000001472	1,1588

Sumber : Hasil Perhitungan

Pada tabel *throughput* hasil perhitungan ini dapat disimpulkan bahwa *throughput* hasil perhitungan lebih besar daripada *throughput* yang didapatkan dari hasil percobaan.

5.7.2. Analisis *Delay end-to-end* Metode DiffServ

Delay end-to-end dihitung dari *Source* ke *Destination*. Analisa *delay end to end* dari paket tersebut adalah sebagai berikut:

5.7.2.1. *Delay* Enkapsulasi

a. *Delay* Enkapsulasi pada *Source*

Paket *voice* pada *transport layer*, panjang segmen adalah

$$W_{segmen} = W_{data} + Header_{UDP}$$

$$= 40 + 8 = 48\ byte$$

Setelah melalui *transport layer*, segmen melalui *internet layer* dengan panjang datagram adalah

$$W_{datagram} = W_{segmen} + Header_{ip}$$
$$= 48 + 20 = 68\ byte$$

Setelah itu paket melalui *network interface layer*, baru kemudian dikirim dengan panjang *frame*,

$$W_{frame} = W_{datagram} + Header_{Ethernet\,II}$$
$$= 68 + 14 = 82\ byte$$

C_{proc} paket *voice* adalah 5 kbps yang merupakan *bandwidth* pembangkitan trafik *voice* pada percobaan kedua (P2). (Maya Ari Hardini; 2006:60)

Jumlah paket *voice* untuk P2 adalah 468,75 paket. Sehingga jumlah total *frame* adalah

$$W_{frame\,total} = jumlah\ paket\ udp\ x\ (W_{frame})$$
$$= 468{,}75 x 82 = 38437{,}5\ byte$$

$$t_{enc} = \frac{W_{frame\,total}}{C_{proc}} x8$$
$$= \frac{38437{,}5}{5000} x8$$
$$= 61{,}5\ detik$$

Perhitungan *delay* enkapsulasi pada masing-masing percobaan ditunjukkan pada Tabel 5.15.

Tabel 5.26 *Delay* Enkapsulasi *Source*

Data	*Bandwidth* P1 (bit)	*Bandwidth* P2 (bit)	W_{frame}	*Delay* Enkapsulasi	
				P1 (detik)	P2 (detik)
Voice	2500	5000	82	61,5	61,5
Mission critical	3750	7500	1066	63,96	63,96
Bulk	3125	6250	1066	53,3	53,3
Best effort	3125	6250	1066	42,64	42,64

Sumber: Hasil Perhitungan

b. *Delay* Enkapsulasi dengan Penyisipan Label MPLS

Paket *mission critical* yang akan dikirim diberi *header* pada masing-masing *layer* TCP/IP. Pada *transport layer*, panjang segmen adalah

$$W_{segmen} = W_{data} + (Header_{TCP} + option)$$
$$= 1000 + (20 + 12) = 1032\ byte$$

Setelah melalui *transport layer*, segmen melalui *internet layer* dengan panjang datagram adalah

$$W_{datagram} = W_{segmen} + Header_{ip}$$
$$= 1032 + 20 = 1052\ byte$$

Paket selanjutnya ditambah dengan Header MPLS. Setelah itu paket melalui *network interface layer*, baru kemudian dikirimkan dengan panjang *frame* adalah

$$W_{frame} = W_{datagram} + Header_{MPLS} + Header_{HDLC}$$
$$= 1052 + 4 + 4 = 1060\ byte$$

C_{proc} adalah kecepatan proses sumber data (bps). (Maya Ari Hardini, 2006:33)

Sumber data adalah *router* r1. Sehingga C_{proc} sama dengan alokasi *bandwidth* kelas *mission critical*, yaitu 30% dari *bandwidth* saluran.

$$C_{proc} = 30\ kbps$$

Jumlah paket *mission critical* untuk P2 adalah 56,25. Sehingga jumlah total frame yang ditransmisikan dari sumber ke tujuan adalah

$$W_{frame\,total} = jumlah\,paket\,tcp\,x\,(W_{frame})$$
$$= 56{,}25x1060 = 59625\ byte$$

Sehingga *delay* enkapsulasi paket *mission critical* adalah

$$t_{enc} = \frac{W_{frame}}{C_{proc}} x8$$
$$= \frac{59625}{15000} x8$$
$$= \frac{477000}{15000}$$
$$= 31{,}8\ detik$$

Dengan cara yang sama, maka hasil perhitungan *delay* enkapsulasi dengan penyisipan label MPLS paket UDP dengan $W_{frame\,total} = 76\ byte$ dan paket TCP dengan $W_{frame\,total} = 1060\ byte$ adalah

Tabel 5.27 *Delay* Enkapsulasi dengan Penyisipan Label MPLS

Data	*Bandwidth* Kelas (kb)	W_{frame} (byte)	*Delay* Enkapsulasi	
			P1 (detik)	P2 (detik)
Voice	10	76	14,25	28,5
Mission critical	15	1060	15,9	31,8
Bulk	12,5	1060	13,25	26,5
Best effort	12,5	1060	10,6	21,2

Sumber: Hasil Perhitungan

c. *Delay* Enkapsulasi Tanpa Penyisipan Label MPLS

Perhitungan *delay* enkapsulasi tanpa penyisipan label MPLS dengan $W_{frame\ total} = 72\ byte$ untuk paket UDP dan $W_{frame\ total} = 1056\ byte$ pada paket TCP untuk masing-masing percobaan ditunjukkan pada Tabel 5.17.

Tabel 5.28 *Delay* Enkapsulasi Tanpa Penyisipan Label MPLS

Data	*Bandwidth* Kelas (kb)	W_{frame} (byte)	*Delay* Enkapsulasi	
			P1 (detik)	P2 (detik)
Voice	10	72	13,5	27
Mission critical	15	1056	15,84	31,68
Bulk	12,5	1056	13,2	26,4
Best effort	12,5	1056	10,56	21,12

Sumber : Hasil Perhitungan

d. *Delay* Enkapsulasi pada *Egress Router*

Perhitungan *delay* enkapsulasi untuk paket UDP dengan $W_{frame\ total} = 82\ byte$ dan paket TCP dengan $W_{frame\ total} = 1066\ byte$, serta *bandwidth* saluran 200 kbps pada masing-masing percobaan ditunjukkan pada Tabel 5.29.

Tabel 5.29 *Delay* Enkapsulasi pada *egress router*

Data	*Bandwidth* Kelas (kb)	W_{frame} (byte)	*Delay* Enkapsulasi	
			P1 (detik)	P2 (detik)
Voice	40	82	3,8438	7,6875
Mission critical	60	1066	3,9975	7,995
Bulk	50	1066	3,3312	6,6625
Best effort	50	1066	2,665	5,33

Sumber : Hasil Perhitungan

5.7.2.2. *Delay* Dekapsulasi

Nilai *delay* dekapsulasi pada *host* tujuan diasumsikan sama dengan *delay* enkapsulasi pada *egress router*. *Delay* dekapsulasi diasumsikan dihitung dengan nilai C_{proc} yang sama dengan nilai C_{proc} pada *delay* enkapsulasi. *Delay* dekapsulasi pada *ingress router* dengan $W_{frame\ dekapsulasi} = 34\ byte$,pada *intermediate router* dengan $W_{frame\ dekapsulasi} = 4\ byte$, dan pada *egress router* dengan $W_{frame\ dekapsulasi} = 24\ byte$ dapat dilihat pada tabel berikut ini. Nilai *delay* dekapsulasi yang lain diasumsikan sama dengan *delay* enkapsulasi.

Tabel 5.30 *Delay* Dekapsulasi pada *ingress router*

Data	*Bandwidth* Kelas (kb)	W_{frame} (byte)	*Delay* Dekapsulasi	
			P1 (detik)	P2 (detik)
Voice	10	34	6,375	12,75
Mission critical	15	34	0,51	1,02
Bulk	12,5	34	0,425	0,85
Best effort	12,5	34	0,34	0,68

Sumber : Hasil Perhitungan

Tabel 5.31 *Delay* Dekapsulasi pada *intermediate router*

Data	*Bandwidth* Kelas (kb)	W_{frame} (byte)	*Delay* Dekapsulasi	
			P1 (detik)	P2 (detik)
Voice	10	4	0,75	1,5
Mission critical	15	4	0,06	0,12
Bulk	12,5	4	0,05	0,1
Best effort	12,5	4	0,04	0,08

Sumber : Hasil Perhitungan

Tabel 5.32 *Delay* Dekapsulasi pada *egress router*

Data	*Bandwidth* Kelas (kb)	W_{frame} (byte)	*Delay* Dekapsulasi	
			P1 (detik)	P2 (detik)
Voice	10	24	4,5	9
Mission critical	15	24	0,36	0,72
Bulk	12,5	24	0,3	0,6
Best effort	12,5	24	0,24	0,48

Sumber : Hasil Perhitungan

5.7.2.3. *Delay* Antrian

Pada analisis ini digunakan model antrian M/M/1 yang merupakan sistem dengan satu server dan paket yang datang dari sumber merupakan paket data yang acak. (Firdiana Inanda; 2008:89)

a. *Delay* Antrian dengan Penyisipan Label MPLS

Total waktu antrian paket *voice* pada *router* dengan kapasitas 10 kbps berdasarkan persamaan adalah

$$t_w = t_{queue} + t_{serv}$$

dengan $t_{serv} = \frac{1}{\mu}$, di mana $\mu = \frac{C}{\bar{b}}$

$$\mu = \frac{10000}{(68+4+4)x8}$$

$$= \frac{10000}{608} = 16{,}4473\ paket/detik$$

sehingga $t_{serv} = \frac{1}{\mu} = \frac{1}{16{,}4473} = 0{,}0608\ detik$

dan dengan mengasumsikan bahwa *traffic load* ρ = 0,9 (Kukuh Wira Kurniawan; 2004:48), maka kecepatan kedatangan paket adalah

$$\lambda = \rho . \mu$$

$$= 0{,}9x16{,}4473$$

$$= 14{,}8026\ paket/s$$

$$t_w = \frac{\lambda}{\mu(\mu - \lambda)} + \frac{1}{\mu}$$

$$= \frac{14{,}8026}{16{,}4473(16{,}4473 - 14{,}8026)} + 0{,}0608$$

$$= 0{,}608 \; detik$$

Untuk total waktu antrian paket TCP dan UDP ditunjukkan pada Tabel 5.33.

Tabel 5.33 *Delay* Antrian dengan Label MPLS

Data	*Bandwidth* Kelas (kb)	*Delay* Antrian (detik)
Voice	10	0,608
Mission critical	15	5,6533
Bulk	12,5	6,784
Best effort	12,5	6,784

Sumber : Hasil Pengujian

b. *Delay* Antrian Tanpa Penyisipan Label MPLS

Total waktu antrian tanpa penyisipan label MPLS dengan ditunjukkan oleh tabel berikut.

Tabel 5.34 *Delay* Antrian Tanpa Label MPLS

Data	*Bandwidth* Kelas (kb)	*Delay* Antrian (detik)
Voice	10	0,576
Mission critical	15	5,632
Bulk	12,5	6,7584
Best effort	12,5	6,7584

Sumber : Hasil Pengujian

d. *Delay* Antrian pada r3

Total waktu antrian pada *egress router* ditunjukkan oleh tabel berikut.

Tabel 5.35 *Delay* Antrian pada *egress router*

Data	*Bandwidth* Kelas (kb)	*Delay* Antrian (detik)
Voice	40	0,164
Mission critical	60	1,4213
Bulk	50	1,7056
Best effort	50	1,7056

Sumber : Hasil Perhitungan

5.7.2.4. *Delay* Propagasi

Delay propagasi adalah waktu yang dibutuhkan untuk merambatkan sinyal yang berisi data dari sumber ke tujuan. Dengan mengasumsikan panjang kabel T1 antar *router* adalah panjang maksimumnya, yaitu 200 meter. Maka *delay* propagasi adalah

$$t_p = \frac{S}{V}$$

$$= \frac{200}{0{,}67x3x10^8} = 99{,}502.10^{-8} detik$$

Sedang hasil perhitungan *delay* propagasi saluran ethernet dengan kabel UTP CAT 5 dan jarak maksimum 200 meter adalah sama dengan *delay* propagasi kabel T1 diatas.

5.7.2.5. *Delay* Transmisi

Delay transmisi adalah waktu yang dibutuhkan untuk meletakkan sebuah paket ke media transmisi.

a. *Delay* Transmisi pada *host* sumber

Berikut ini adalah perhitungan *delay* transmisi untuk paket *voice* pada P2.

$$t_{trans} = N_{frame} x \frac{(l + l')x8}{C_{trans}}$$

$$= 468{,}75x\frac{(68 + 14)x8}{5000}$$

$$= \frac{468{,}75x656}{5000} = 61{,}5\ detik$$

(Maya Ari Hardini; 2006:63)

Hasil perhitungan *delay* transmisi pada saluran Ethernet ditunjukkan oleh Tabel 5.36.

Tabel 5.36 *Delay* Transmisi pada *Source*

Data	*Bandwidth* P1 (bit)	*Bandwidth* P2 (bit)	*Delay* Transmisi	
			P1 (detik)	P2 (detik)
Voice	2500	5000	61,5	61,5
Mission critical	3750	7500	63,96	63,96
Bulk	3125	6250	53,3	53,3
Best effort	3125	6250	42,64	42,64

Sumber : Hasil Perhitungan

b. *Delay* Transmisi dengan Penyisipan Label MPLS

Hasil perhitungan *delay* transmisi dengan penyisipan label MPLS ditunjukkan oleh Tabel 5.37 dibawah ini.

Tabel 5.37 *Delay* Transmisi dengan Label MPLS

Data	*Bandwidth* Kelas (kb)	*Delay* Transmisi	
		P1 (detik)	P2 (detik)
Voice	10	14,25	28,5
Mission critical	15	15,9	31,8
Bulk	12,5	13,25	26,5
Best effort	12,5	10,6	21,2

Sumber : Hasil Perhitungan

c. *Delay* Transmisi Tanpa Penyisipan Label MPLS

Hasil perhitungan *delay* transmisi tanpa penyisipan label MPLS ditunjukkan oleh Tabel 5.38.

Tabel 5.38 *Delay* Enkapsulasi Tanpa Penyisipan Label MPLS

Data	*Bandwidth* Kelas (kb)	*Delay* Transmisi	
		P1 (detik)	P2 (detik)
Voice	10	13,5	27
Mission critical	15	15,84	31,68
Bulk	12,5	13,2	26,4
Best effort	12,5	10,56	21,12

Sumber : Hasil Perhitungan

d. *Delay* Transmisi pada *egress router*

Perhitungan *delay* transmisi pada masing-masing percobaan ditunjukkan pada Tabel 5.39.

Tabel 5.39 *Delay* Transmisi pada *egress router*

Data	*Bandwidth* Kelas (kb)	*Delay* Transmisi	
		P1 (detik)	P2 (detik)
Voice	40	3,8438	7,6875
Mission critical	60	3,9975	7,995
Bulk	50	3,3312	6,6625
Best effort	50	2,665	5,33

Sumber : Hasil Perhitungan

5.7.2.6. *Delay end to end*

Dari perhitungan nilai *delay* di atas, maka perhitungan nilai *delay end to end* pada DiffServ dihitung berdasarkan jalur *source*-r1-r2-r3-*destination*, perhitungannya sesuai dengan Persamaan (3-3), yaitu

$$t_{end\ to\ end} = t_{enc\ source} + \left(t_{trans\ source} + t_{prop} + t_{dec\ ingress\ router} + t_{enc\ dg\ MPLS} + t_{wdg\ MPLS}\right) + \left(t_{trans\ dg\ MPLS} + t_{pT1} + t_{dec\ intermediate\ router} + t_{w\ tanpa\ MPLS}\right) + \left(t_{trans\ tanpa\ MPLS} + t_{pT1} + t_{dec\ egress\ router} + t_{enc\ egress\ router} + t_{w\ egress\ router}\right) + \left(t_{trans\ egress\ router} + t_{pT1} + t_{dec\ egress\ router}\right)$$

Hasil perhitungan *delay end to end* metode DiffServ untuk masing-masing kelas adalah sebagai berikut.

Tabel 5.40 *Delay end to end* Metode DiffServ

Data	*Delay end to end*	
	P1 (detik)	P2 (detik)
Voice	190,1606	255,9730
Mission critical	197,5516	254,4766
Bulk	169,2854	216,7230
Best effort	138,4780	176,4280

Sumber : Hasil Perhitungan

5.7.3. Analisis *Delay end-to-end* Metode DS-TE

Dari perhitungan nilai *delay* di atas, maka rumus perhitungan nilai *delay end to end* untuk paket *voice* sama dengan rumus *delay end to end* pada DiffServ pada Persamaan (3-3). Sedang perhitungan untuk trafik *mission critical*, *bulk* dan *best effort* adalah seperti pada Persamaan (3-4). Dan hasil perhitungannya ditunjukkan oleh Tabel berikut.

Tabel 5.41 *Delay end to end* Metode DS-TE Hasil Perhitungan Teori

Data	***Delay end to end* Hasil Perhitungan Teori**	
	P1 (detik)	**P2 (detik)**
Voice	190,1606	255,9730
Mission critical	223,0424	291,9299
Bulk	192,6006	250,0070
Best effort	158,4870	204,4120

Sumber : Hasil Perhitungan

5.8. Perbandingan Pengujian dan Teori

Perbandingan perhitungan *delay end to end* antara pengujian dengan teori untuk metode DiffServ adalah sebagai berikut.

Tabel 5.42 *Delay end to end* Metode DiffServ Hasil Pengujian

Data	**Paket yang dikirim**		***Delay end to end* Hasil Pengujian**	
	P1 (paket)	**P2 (paket)**	**P1 (detik)**	**P2 (detik)**
Voice	234	468	112,6	121,2
Mission critical	15	22,6	142,2	160
Bulk	8,8	37,8	147	143,2
Best effort	6	8,6	200,2	206,6

Sumber : Hasil Pengujian

Tabel 5.43 *Delay end to end* Metode DiffServ Hasil Perhitungan Teori

Data	**Paket yang dikirim**		***Delay end to end* Hasil Perhitungan Teori**	
	P1 (paket)	**P2 (paket)**	**P1 (detik)**	**P2 (detik)**
Voice	234,375	468,75	190,1606	255,9730
Mission critical	28,125	56,25	223,0424	299,8049
Bulk	19,5313	39,0625	192,6006	256,5695
Best effort	15,625	31,25	158,4870	209,6620

Sumber : Hasil Perhitungan

Sedangkan untuk metode DS-TE ditunjukkan oleh tabel berikut

Tabel 5.44 *Delay end to end* Metode DS-TE Hasil Pengujian

Data	**Paket yang dikirim**		***Delay end to end* Hasil Pengujian**	
	P1 (paket)	**P2 (paket)**	**P1 (detik)**	**P2 (detik)**
Voice	234	468	158,2	160,2

Mission critical	12,2	52,4	187,4	203,8
Bulk	7,2	34	177,6	179,8
Best effort	4,2	17,8	304,8	224,8

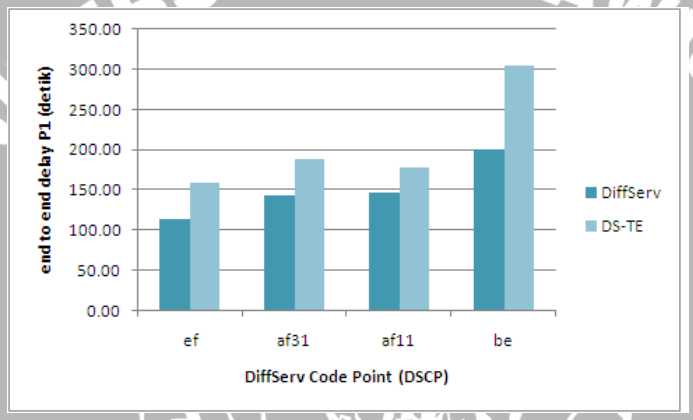
Sumber : Hasil Pengujian

Tabel 5.45 *Delay end to end* Metode DS-TE Hasil Perhitungan Teori

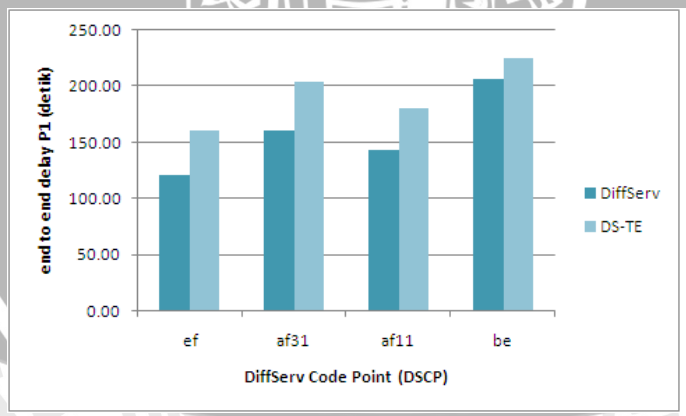
Data	Paket yang dikirim		*Delay end to end* Hasil Perhitungan Teori	
	P1 (paket)	P2 (paket)	P1 (detik)	P2 (detik)
Voice	234,375	468,75	190,1606	255,9730
Mission critical	28,125	56,25	223,0424	291,9299
Bulk	19,5313	39,0625	192,6006	250,0070
Best effort	15,625	31,25	158,4870	204,4120

Sumber : Hasil Perhitungan

Berikut ini adalah perbandingan grafik *delay end to end* dari hasil pengujian antara metode DiffServ dan DS-TE pada percobaan pertama dan kedua.

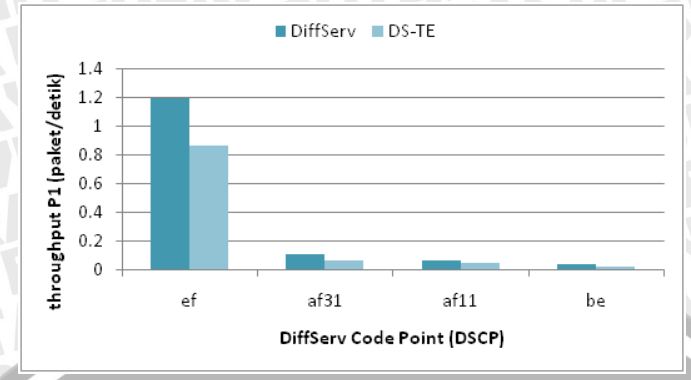


Gambar 5.31. Grafik Perbandingan *Delay end to end*
Metode DiffServ dan Metode DS-TE pada P1
Sumber: Hasil Pengujian

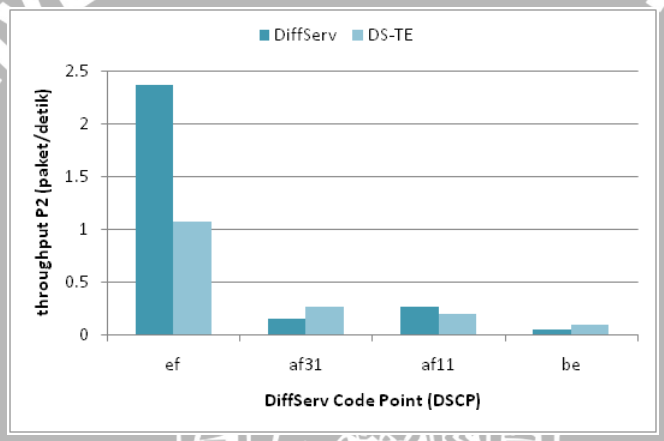


Gambar 5.32. Grafik Perbandingan *Delay end to end*
Metode DiffServ dan Metode DS-TE pada P2
Sumber: Hasil Pengujian

Perbandingan *troughput* ditunjukkan oleh grafik berikut

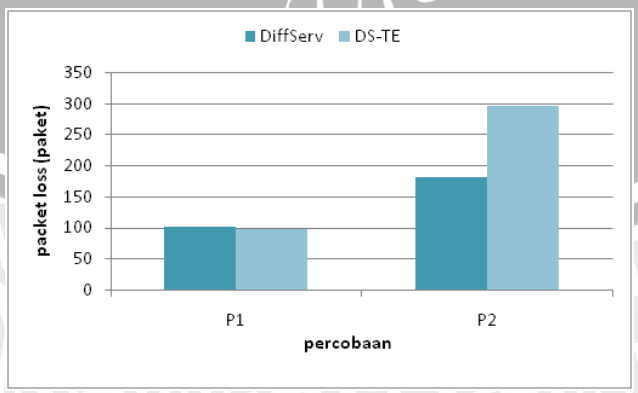


Gambar 5.33. Grafik Perbandingan *Throughput*
Metode DiffServ dan Metode DS-TE pada P1
Sumber: Hasil Pengujian



Gambar 5.34. Grafik Perbandingan *Throughput*
Metode DiffServ dan Metode DS-TE pada P2
Sumber: Hasil Pengujian

Sedangkan perbandingan *paket loss* untuk trafik *voice* pada percobaan pertama dan kedua pada metode DiffServ dan DS-TE, ditunjukkan oleh grafik dibawah ini



Gambar 5.35. Grafik Perbandingan *Packet Loss* pada

Metode DiffServ dan Metode DS-TE pada P1 dan P2

Sumber: Hasil Pengujian

Pada grafik perbandingan *delay end to end* pada Gambar 3.1 dan Gambar 3.2 terlihat bahwa *delay end to end* pada trafik data jauh lebih besar daripada trafik yang lain. Untuk hasil pengujian *throughput* untuk metode DS-TE dan DiffServ, dapat diketahui bahwa *throughput* untuk trafik *voice* pada metode DS-TE lebih besar daripada *throughput* pada metode DiffServ. Dan *packet loss* yang terjadi pada trafik *voice* untuk metode DS-TE lebih besar daripada metode DiffServ.

BAB VI
PENUTUP

6.1. Kesimpulan

Berdasarkan hasil yang diperoleh dari proses analisa, disimpulkan bahwa:

1. Dalam menerapkan metode *Differentiated Service* pada jaringan MPLS, maka diperlukan tahap-tahap sebagai berikut:
   a. Mengidentifikasi trafik yang akan dilewatkan dalam jaringan MPLS dan mengidentifikasi kebutuhan QoS untuk tiap trafik.
   b. Membagi trafik dalam kelas-kelas (klasifikasi) sesuai dengan kebutuhan tiap trafik.
   c. Mendefinisikan QoS *policy* untuk tiap kelas yang sesuai dengan kebutuhan masing-masing kelas. Dengan menentukan mekanisme QoS yang akan diterapkan, selain itu juga menentukan alokasi *bandwidth* untuk tiap kelas dan menentukan MPLS *experimental* bit yang akan diberikan.
2. Dalam menerapkan teknik DiffServ-*Traffic Engineering* pada jaringan MPLS diperlukan tiga tahap sebagai berikut:
   a. Mengidentifikasi trafik dan kebutuhan QoS trafik tersebut.
   b. Membagi trafik dalam kelas-kelas (klasifikasi) sesuai dengan kebutuhannya.
   c. Mendefinisikan QoS *policy* untuk masing-masing kelas. Dengan menentukan mekanisme QoS, alokasi *bandwidth* dan MPLS *experimental* bit.

   Selanjutnya dilakukan perancangan jumlah *tunnel* dan jalur yang akan dilewati *tunnel* untuk membawa masing-masing trafik.
3. Nilai *delay end to end* untuk metode DiffServ dan DS-TE dipengaruhi oleh jumlah hop yang dilewati dan dipengaruhi oleh *Per Hop Behaviour* (PHB) yang dijalankan oleh *router* dalam meneruskan paket. Pengaruh jumlah hop ini ditunjukkan pada hasil pengujian, di mana *delay end to end* untuk semua kelas trafik pada metode DS-TE relatif lebih tinggi daripada *delay end to end* pada metode DiffServ. Semakin banyak jumlah hop yang dilalui, *delay* juga semakin besar. Sedangkan pengaruh PHB terlihat dari hasil perbandingan antara *delay end to end* dari hasil pengujian dan *delay end to end* yang didapat dari hasil perhitungan teori.
4. Dari hasil pengujian *throughput* untuk metode DS-TE dan DiffServ, dapat diketahui bahwa *throughput* untuk trafik *voice* pada metode DS-TE lebih besar daripada *throughput* pada metode DiffServ. Hal ini terjadi karena pengaruh implementasi

DiffServ PHB yang memprioritaskan trafik ini, dan pengaruh alokasi *bandwidth sub-pool* yang diberikan untuk trafik *voice* ini serta jumlah hop yang dilalui. Semakin besar alokasi *bandwidth* dan semakin sedikit jumlah hop maka *troughput* akan semakin besar.

5. *Packet loss* yang terjadi pada trafik *voice* untuk metode DS-TE lebih besar daripada metode DiffServ. *Paket loss* ini dipengaruhi oleh mekanisme *policing* yang diterapkan pada *router*, konfigurasi *bandwidth* yang diterapkan pada saluran, dan kepadatan trafik pada jaringan.

6.2. Saran

Beberapa hal yang disarankan untuk pengembangan dan perbaikan terhadap pemodelan sistem yang dirancang antara lain:

1. Membuat *tunnel* tersendiri dengan jalur yang berbeda untuk masing-masing trafik pada penerapan metode DS-TE, agar kebutuhan tiap kelas trafik dapat terpenuhi.
2. Analisis QoS pada MPLS ini dapat dikembangkan dengan menerapkan mekanisme antrian yang sesuai dengan kebutuhan trafik sebagai pengelola kepadatan.
3. Menggunakan komputer yang terpisah untuk menerapkan kelima *router* dengan menggunakan emulator Dynamips. Yaitu dengan menjalankan maksimal dua buah *router* dalam sebuah komputer, agar CPU komputer tidak bekerja terlalu keras.

DAFTAR PUSTAKA

Alvarez, S. 2006. *QoS for IP/MPLS Networks*. Indianapolis: Cisco Press.

Anjali, T. 2004. *DiffServ/MPLS Network Design dan Management*. Thesis. Georgia: Georgia Institute of Technology.

Anonim. *Link Encryptor User's Guide-Part Number: 81068-00J, Appendix C Cable Assemblies*.

Anonim. *MPLS Traffic Engineering–DiffServ Aware (DS-TE), Release Number* 12.2(14)S. San Jose: Cisco Systems, Inc.

Anonim. *Multiprotocol Label Switching (MPLS) Traffic Engineering, Release Number* 12.0(5)S. San Jose: Cisco Systems, Inc.

Anonim. *QoS Quick Reference Sheets*.

Anonim. 2004. *Implementing Cisco Quality of Service (QoS) volume 1 version 2.1*. Cisco Systems.

Anonim. 2004. *Implementing Cisco Quality of Service (QoS) volume 2 version 2.1*. Cisco Systems.

Anonim. 2004. *Quality of Service and MPLS Methodologies*. IP Infusion.

Anonim. 2006. IOS *Quality of Service Solution Configuration Guide*. Cisco Systems, Inc.

Anonim. 2006. *MPLS DiffServ Tunneling Modes, Cisco IOS Release: Multiple releases*. San Jose: Cisco Systems, Inc.

Anonim. 2009. *SLAX*. Wikimedia Foundation, Inc http://en.wikipedia.org/wiki/SLAX. (diakses tanggal 12 mei 2009).

Anuzelli, G. Dynamips/Dynagen *Tutorial Documentation Revision* 1.11.7.

Audin, G. 2007. *VoIP Bandwidth Fundamentals*. http://searchunifiedcommunications.techtarget.com/?asrc=TAB_searchUnifiedCommunications. (diakses tanggal 5 mei 2009)

Carne, B. E. 2004. *A Proffesional's Guide to Data Communication in a TCP/IP World*. Norwood: Artech House, Inc.

Chris. 2007. *How to Limit Rate on Interface*. http://www.cisconet.com/index.php/QoS/Cisco-How-to-limit-rate-on-interface.html. (diakses tanggal 22 maret 2009).

De Ghein, L. 2007. *MPLS Fundamentals*. Indianapolis: Cisco Press.

De Mey, Olivier. 2005. *How to Set HTTP Packet Size to 500 Bytes*. http://mailman.isi.edu/pipermail/ns-users/2005-May/049958.html. (diakses tanggal 5 mei 2009)

Dhoto. 2007. *Jaringan Komputer*. Politeknik Elektronika Negeri Surabaya-Institut Teknologi Sepuluh Nopember (PENS-ITS). Surabaya.

Groth, D., McBee, J., Barnett, D. 2001. *Cabling: The Complete Guide to Network Wiring*. Alameda: Sybex Inc.

Hardini, M. A. 2006. *Analisis Penerapan Metode Tunnel Broker pada Interkoneksi Jaringan Ipv4 dengan Ipv6*. Skripsi Tidak Diterbitkan. Malang: Universitas Brawijaya.

Harris, G. 2000. *CISCO HDLC encapsulation*. Ethreal, Inc. http://www.ethereal.com/lists/ethereal-users/200012/msg00030.html. (diakses tanggal 2 mei 2009).

Hijryati, N. 2003. *Integrasi Multiprotocol Label Switching (MPLS) pada TCP/IP*. Skripsi Tidak Diterbitkan. Malang: Universitas Brawijaya.

Heinlein, S. 2005. *Delivering Predictable IP Communications: Quality of Service for Voice over IP*. Sunnyvale: Juniper Networks, Inc.

Inanda, F. 2008. *Rekayasa Trafik MPLS (Multi Protocol Label Switching) Menggunakan Logika FUZZY*. Skripsi Tidak Diterbitkan. Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya. Malang.

Kana, W. A. 2004. *Keamanan dan Layanan VPN dalam Arsitektur Jaringan MPLS*. Institut Teknologi Bandung.

Kurniawan, K. W. 2004. *Penerapan Multi Protocol Label Switching (MPLS) pada Virtual Private Network (VPN)*. Skripsi Tidak Diterbitkan. Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya. Malang.

Lammle, T. 2007. *Cisco Certified Network Associate Study Guide, Sixth Edition*. Indianapolis: Wiley Publishing, Inc.

Osborne, E., Simha, A. 2002. *Traffic Engineering with MPLS*. Indianapolis: Cisco Press.

Primantoko, B. P. 2003. *Perencanaan VoMPLS (Voice over Multiprotocol Label Switching) Melalui Jaringan ATM (Asynchronous Transfer Mode)*. Skripsi Tidak Diterbitkan. Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya. Malang.

Rodrigues, A., Gatrell, J., Karas, J., Peschke, R. 2001. *TCP/IP Tutorial and Technical Overview*. IBM Corporation.

Siswanto, J. 2007. *Perancangan TV over IP (Internet Protocol) Menggunakan Jaringan Speedy*. Skripsi Tidak Diterbitkan. Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya. Malang.

Syarvianti, E. 2005. *Penerapan IPv6 pada Jaringan TCP/IP*. Skripsi Tidak Diterbitkan. Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya. Malang.

Taufik, E. 2008. *Konsep CIR (Committed Information Rate)*. http://eandriana.wordpress.com/2008/06/03/konsep-cir-committed-information-rate/. (diakses tanggal 23 maret 2009).

Wibowo, B. P. 2008. *Cisco Router Simulator with Dynamips/Dynagen*. http://bayupw.blogspot.com/2008/02/cisco-router-simulator-with-dynamips.html. (diakses tanggal 19 april 2008)

Xiao, Xipeng. *Internet QoS: The Big Picture*, Michigan State University.

Lampiran 1. Konfigurasi Jaringan MPLS pada Dynagen

Berikut ini adalah konfigurasi file jaringan MPLS yang digunakan oleh emulator Dynamips untuk melakukan simulasi.

```
autostart = False
sparsemem = True
ghostios = True
[localhost:7200]
    workingdir = C:\Program
Files\Dynamips\sample_labs\advipservicesk9-mz[1]-124-15.T(DS-TE)n2002-
-
    [[7200]]
        npe = npe-400
        image = \Program Files\Dynamips\images\c7200-advipservicesk9-
mz[1].124-15.T.image
        nvram = 192
        confreg = 0x2102
        midplane = std
        ghostios = True
        sparsemem = True
        mmap = true
        clock = 10

    [[ROUTER r1]]
        f0/0 = nio_gen_eth:\Device\NPF_{456218F9-33BB-4FBF-B7E7-
172286E8188D}
        slot1 = PA-8T
        s1/0 = r2 s1/0
        slot2 = PA-8T
        s2/0 = r5 s1/0
        model = 7200
        idlepc = 0x62578d34

    [[ROUTER r2]]
        f0/0 = nio_gen_eth:\Device\NPF_{456218F9-33BB-4FBF-B7E7-
172286E8188D}?
        slot1 = PA-8T
        s1/0 = r1 s1/0
        slot2 = PA-8T
        s2/0 = r3 s1/0
        model = 7200
        idlepc = 0x606ba350
    [[ROUTER r3]]
        f0/0 = nio_gen_eth:\Device\NPF_{C9832AC9-5181-45A8-ABD7-
693EA4C9263D}
        slot1 = PA-8T
        s1/0 = r2 s2/0
        slot2 = PA-8T
        s2/0 = r4 s1/0
        model = 7200
        idlepc = 0x606ba754
[localhost:7201]
    workingdir = C:\Program
Files\Dynamips\sample_labs\advipservicesk9-mz[1]-124-15.T(DS-TE)n2002-
-
    [[7200]]
        npe = npe-400
```

```
        image = \Program Files\Dynamips\images\c7200-advipservicesk9-
mz[1].124-15.T.image
    [[ROUTER r4]]
        f0/0 = nio_gen_eth:\Device\NPF_{456218F9-33BB-4FBF-B7E7-
172286E8188D}?
        slot1 = PA-8T
        s1/0 = r3 s2/0
        slot2 = PA-8T
        s2/0 = r5 s2/0
        model = 7200
        idlepc = 0x62578d34
    [[ROUTER r5]]
        f0/0 = nio_gen_eth:\Device\NPF_{456218F9-33BB-4FBF-B7E7-
172286E8188D}?
        slot1 = PA-8T
        s1/0 = r1 s2/0
        slot2 = PA-8T
        s2/0 = r4 s2/0
        model = 7200
        idlepc = 0x606ba710
```

Lampiran 2. Konfigurasi Metode DiffServ

Untuk mengimplementasikan metode DiffServ pada jaringan MPLS, dilakukan konfigurasi pada semua *router*. Konfigurasi yang dibutuhkan untuk tiap *router* tersebut adalah sebagai berikut:

a. Konfigurasi *Router* r1

```
!
hostname r1
!
enable secret 5 $1$T3o7$IqQ1jVzPpKkgDDXueV.z/1
!
ip cef
!
class-map match-all gold
 match mpls experimental topmost 5
class-map match-all bronze
 match mpls experimental topmost 1
class-map match-all bulk
 match ip dscp af11
class-map match-all silver
 match mpls experimental topmost 3
class-map match-all bulk1
 match ip dscp af11
class-map match-all voice1
 match ip dscp ef
class-map match-all mission-critical
 match ip dscp af31
class-map match-all voice
 match ip dscp ef
class-map match-all mission-critical1
 match ip dscp af31
!
policy-map PE-P2
 class voice1
  priority percent 20
 class mission-critical1
  bandwidth percent 30
  random-detect dscp-based
 class bulk1
  bandwidth percent 25
  random-detect dscp-based
policy-map PE-P
 class gold
  priority percent 20
 class silver
  bandwidth percent 30
  random-detect
 class bronze
  bandwidth percent 25
  random-detect
policy-map PE-C
 class voice
    police 10000 1875 3750 conform-action set-mpls-exp-imposition-
transmit 5 exceed-action drop
 class mission-critical
```

```
    police 15000 2813 5625 conform-action set-mpls-exp-imposition-
transmit 3 exceed-action set-mpls-exp-imposition-transmit 3
 class bulk
    police 12500 2344 4688 conform-action set-mpls-exp-imposition-
transmit 1 exceed-action set-mpls-exp-imposition-transmit 1
 class class-default
    police 12500 2344 4688 conform-action set-mpls-exp-imposition-
transmit 0 exceed-action drop
!
interface FastEthernet0/0
 bandwidth 200
 ip address 192.16.10.33 255.255.255.240
 duplex half
 service-policy input PE-C
 service-policy output PE-P2
!
interface Serial1/0
 description connection to r2
 bandwidth 100
 ip address 192.16.10.85 255.255.255.252
 mpls ip
 serial restart-delay 0
 service-policy output PE-P
!
interface Serial2/0
 description connection to r5
 bandwidth 100
 ip address 192.16.10.82 255.255.255.252
 mpls ip
 serial restart-delay 0
 service-policy output PE-P
!
router ospf 1
 router-id 17.16.10.1
 log-adjacency-changes
 network 192.16.10.32 0.0.0.15 area 0
 network 192.16.10.80 0.0.0.3 area 0
 network 192.16.10.84 0.0.0.3 area 0
!
```

b. Konfigurasi *Router* r2

```
!
hostname r2
!
enable secret 5 $1$Topm$9m5jIl3cFaM6R6SpuUJK2.
!
ip cef
!
class-map match-all gold
 match mpls experimental topmost 5
class-map match-all bronze
 match mpls experimental topmost 1
class-map match-all bulk
 match ip dscp af11
class-map match-all silver
 match mpls experimental topmost 3
class-map match-all bulk1
 match ip dscp af11
class-map match-all voice1
 match ip dscp ef
```

```
class-map match-all mission-critical
 match ip dscp af31
class-map match-all voice
 match ip dscp ef
class-map match-all mission-critical1
 match ip dscp af31
!
policy-map PE-P2
 class voice1
  priority percent 20
 class mission-critical1
  bandwidth percent 30
  random-detect dscp-based
 class bulk1
  bandwidth percent 25
  random-detect dscp-based
policy-map PE-P
 class gold
  priority percent 20
 class silver
  bandwidth percent 30
  random-detect
 class bronze
  bandwidth percent 25
  random-detect
policy-map PE-C
 class voice
    police 10000 1875 3750 conform-action set-mpls-exp-imposition-
transmit 5 exceed-action drop  exceed-action set-mpls-exp-imposition-
transmit 3
 class mission-critical
    police 15000 2813 5625 conform-action set-mpls-exp-imposition-
transmit 3 exceed-action set-mpls-exp-imposition-transmit 3
 class bulk
    police 12500 2344 4688 conform-action set-mpls-exp-imposition-
transmit 1 exceed-action set-mpls-exp-imposition-transmit 1
 class class-default
    police 12500 2344 4688 conform-action set-mpls-exp-imposition-
transmit 0 exceed-action drop
!
interface FastEthernet0/0
 bandwidth 200
 ip address 192.16.10.65 255.255.255.248
 duplex half
 service-policy input PE-C
 service-policy output PE-P2
!
interface Serial1/0
 description connection to r1
 bandwidth 100
 ip address 192.16.10.86 255.255.255.252
 mpls ip
 serial restart-delay 0
 service-policy output PE-P
!
interface Serial2/0
 description connection to r3
 bandwidth 100
 ip address 192.16.10.89 255.255.255.252
 mpls ip
 serial restart-delay 0
```

```
 service-policy output PE-P
!
router ospf 2
 router-id 17.16.10.2
 log-adjacency-changes
 network 192.16.10.64 0.0.0.7 area 0
 network 192.16.10.84 0.0.0.3 area 0
 network 192.16.10.88 0.0.0.3 area 0
!
```

c. Konfigurasi *Router* r3

```
!
hostname r3
!
enable secret 5 $1$p/bF$t.bHNj/1ZJvYva65woOsl.
!
ip cef
!
no ip domain lookup
!
class-map match-all gold
 match mpls experimental topmost 5
class-map match-all bronze
 match mpls experimental topmost 1
class-map match-all bulk
 match ip dscp af11
class-map match-all silver
 match mpls experimental topmost 3
class-map match-all bulk1
 match ip dscp af11
class-map match-all voice1
 match ip dscp ef
class-map match-all mission-critical
 match ip dscp af31
class-map match-all voice
 match ip dscp ef
class-map match-all mission-critical1
 match ip dscp af31
!
policy-map PE-P2
 class gold
  priority percent 20
 class silver
  bandwidth percent 30
  random-detect
 class bronze
  bandwidth percent 25
  random-detect
policy-map PE-P
 class voice1
  priority percent 20
 class mission-critical1
  bandwidth percent 30
  random-detect dscp-based
 class bulk1
  bandwidth percent 25
  random-detect dscp-based
policy-map PE-C
 class voice
```

```
    police 10000 1875 3750 conform-action set-mpls-exp-imposition-
transmit 5 exceed-action drop
 class mission-critical
    police 15000 2813 5625 conform-action set-mpls-exp-imposition-
transmit 3 exceed-action set-mpls-exp-imposition-transmit 3
 class bulk
    police 12500 2344 4688 conform-action set-mpls-exp-imposition-
transmit 1 exceed-action set-mpls-exp-imposition-transmit 1
 class class-default
    police 12500 2344 4688 conform-action set-mpls-exp-imposition-
transmit 0 exceed-action drop
!
interface FastEthernet0/0
 bandwidth 200
 ip address 192.16.10.1 255.255.255.224
 duplex half
 service-policy input PE-C
 service-policy output PE-P
!
interface Serial1/0
 description connection to r2
 bandwidth 100
 ip address 192.16.10.90 255.255.255.252
 mpls ip
 serial restart-delay 0
 service-policy output PE-P2
!
interface Serial2/0
 description connection to r4
 bandwidth 100
 ip address 192.16.10.93 255.255.255.252
 mpls ip
 serial restart-delay 0
 service-policy output PE-P2
!
router ospf 3
 router-id 17.16.10.3
 log-adjacency-changes
 network 192.16.10.0 0.0.0.31 area 0
 network 192.16.10.88 0.0.0.3 area 0
 network 192.16.10.92 0.0.0.3 area 0
!
```

d. Konfigurasi *Router* r4

```
!
hostname r4
!
enable secret 5 $1$pZS7$EubLM73BMFEKh7g8/mlXY.
!
ip cef
!
no ip domain lookup
!
class-map match-all gold
 match mpls experimental topmost 5
class-map match-all bronze
 match mpls experimental topmost 1
class-map match-all bulk
 match ip dscp af11
class-map match-all silver
```

```
 match mpls experimental topmost 3
class-map match-all bulk1
 match ip dscp af11
class-map match-all voice1
 match ip dscp ef
class-map match-all mission-critical
 match ip dscp af31
class-map match-all voice
 match ip dscp ef
class-map match-all mission-critical1
 match ip dscp af31
!
policy-map PE-P2
 class voice1
  priority percent 20
 class mission-critical1
  bandwidth percent 30
  random-detect dscp-based
 class bulk1
  bandwidth percent 25
  random-detect dscp-based
policy-map PE-P
 class gold
  priority percent 20
 class silver
  bandwidth percent 30
  random-detect
 class bronze
  bandwidth percent 25
  random-detect
policy-map PE-C
 class voice
    police 10000 1875 3750 conform-action set-mpls-exp-imposition-
transmit 5 exceed-action drop
 class mission-critical
    police 15000 2813 5625 conform-action set-mpls-exp-imposition-
transmit 3 exceed-action set-mpls-exp-imposition-transmit 3
 class bulk
    police 12500 2344 4688 conform-action set-mpls-exp-imposition-
transmit 1 exceed-action set-mpls-exp-imposition-transmit 1
 class class-default
    police 12500 2344 4688 conform-action set-mpls-exp-imposition-
transmit 0 exceed-action drop
!
interface FastEthernet0/0
 bandwidth 200
 ip address 192.16.10.73 255.255.255.248
 duplex half
 service-policy input PE-C
 service-policy output PE-P2
!
interface Serial1/0
 description connection to r3
 bandwidth 100
 ip address 192.16.10.94 255.255.255.252
 mpls ip
 serial restart-delay 0
 service-policy output PE-P
!
interface Serial2/0
 description connection to r5
```

```
 bandwidth 100
 ip address 192.16.10.97 255.255.255.252
 mpls ip
 serial restart-delay 0
 service-policy output PE-P
!
router ospf 4
 router-id 17.16.10.4
 log-adjacency-changes
 network 192.16.10.72 0.0.0.7 area 0
 network 192.16.10.92 0.0.0.3 area 0
 network 192.16.10.96 0.0.0.3 area 0
!
```

e. Konfigurasi *Router* r5

```
!
hostname r5
!
enable secret 5 $1$QKQJ$gi7TjeK3MrIbwn2ru2mE50
!
ip cef
!
no ip domain lookup
!
class-map match-all gold
 match mpls experimental topmost 5
class-map match-all bronze
 match mpls experimental topmost 1
class-map match-all bulk
 match ip dscp af11
class-map match-all silver
 match mpls experimental topmost 3
class-map match-all bulk1
 match ip dscp af11
class-map match-all voice1
 match ip dscp ef
class-map match-all mission-critical
 match ip dscp af31
class-map match-all voice
 match ip dscp ef
class-map match-all mission-critical1
 match ip dscp af31
!
policy-map PE-P2
 class voice1
  priority percent 20
 class mission-critical1
  bandwidth percent 30
  random-detect dscp-based
 class bulk1
  bandwidth percent 25
  random-detect dscp-based
policy-map PE-P
 class gold
  priority percent 20
 class silver
  bandwidth percent 30
  random-detect
 class bronze
```

```
  bandwidth percent 25
  random-detect
policy-map PE-C
 class voice
    police 10000 1875 3750 conform-action set-mpls-exp-imposition-
transmit 5 exceed-action drop
 class mission-critical
    police 15000 2813 5625 conform-action set-mpls-exp-imposition-
transmit 3 exceed-action set-mpls-exp-imposition-transmit 3
 class bulk
    police 12500 2344 4688 conform-action set-mpls-exp-imposition-
transmit 1 exceed-action set-mpls-exp-imposition-transmit 1
 class class-default
    police 12500 2344 4688 conform-action set-mpls-exp-imposition-
transmit 0 exceed-action drop
!
interface FastEthernet0/0
 bandwidth 200
 ip address 192.16.10.49 255.255.255.240
 duplex half
 service-policy input PE-C
 service-policy output PE-P2
!
interface Serial1/0
 description connection to r1
 bandwidth 100
 ip address 192.16.10.81 255.255.255.252
 mpls ip
 serial restart-delay 0
 service-policy output PE-P
!
interface Serial2/0
 description connection to r4
 bandwidth 100
 ip address 192.16.10.98 255.255.255.252
 mpls ip
 serial restart-delay 0
 service-policy output PE-P
!
router ospf 5
 router-id 17.16.10.5
 log-adjacency-changes
 network 192.16.10.48 0.0.0.15 area 0
 network 192.16.10.80 0.0.0.3 area 0
 network 192.16.10.96 0.0.0.3 area 0
!
```

Lampiran 3. Konfigurasi Metode DS-TE

Sama halnya dengan mengimplementasikan metode DiffServ pada jaringan MPLS, maka pada penerapan DS-TE pada jaringan MPLS juga dilakukan konfigurasi pada semua *router*. Konfigurasi yang diperlukan oleh tiap *router* tersebut adalah sebagai berikut:

a. Konfigurasi *Router* r1

```
!
hostname r1
!
enable secret 5 $1$T3o7$IqQ1jVzPpKkgDDXueV.z/1
!
ip cef
!
mpls traffic-eng tunnels
!
class-map match-all gold
 match mpls experimental topmost 5
class-map match-all bronze
 match mpls experimental topmost 1
class-map match-all bulk
 match ip dscp af11
class-map match-all silver
 match mpls experimental topmost 3
class-map match-all bulk1
 match ip dscp af11
class-map match-all voice1
 match ip dscp ef
class-map match-all mission-critical
 match ip dscp af31
class-map match-all voice
 match ip dscp ef
class-map match-all mission-critical1
 match ip dscp af31
!
policy-map PE-P2
 class gold
  priority percent 30
 class silver
  bandwidth percent 25
  random-detect
 class bronze
  bandwidth percent 20
  random-detect
policy-map PE-P
 class voice1
  priority percent 30
 class mission-critical1
  bandwidth percent 25
  random-detect dscp-based
 class bulk1
  bandwidth percent 20
  random-detect dscp-based
policy-map PE-C
```

```
 class voice
    police 10000 1875 3750 conform-action set-mpls-exp-imposition-
transmit 5 exceed-action drop
 class mission-critical
    police 15000 2813 5625 conform-action set-mpls-exp-imposition-
transmit 3 exceed-action set-mpls-exp-imposition-transmit 3
 class bulk
    police 12500 2344 4688 conform-action set-mpls-exp-imposition-
transmit 1 exceed-action set-mpls-exp-imposition-transmit 1
 class class-default
    police 12500 2344 4688 conform-action set-mpls-exp-imposition-
transmit 0 exceed-action drop
!
interface Loopback0
 ip address 17.16.10.11 255.255.255.255
!
interface Tunnel1
 ip unnumbered Loopback0
 tunnel destination 17.16.10.33
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng priority 0 0
 tunnel mpls traffic-eng bandwidth sub-pool 30
 tunnel mpls traffic-eng path-option 1 dynamic
 no routing dynamic
!
interface Tunnel2
 ip unnumbered Loopback0
 tunnel destination 17.16.10.33
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng priority 0 0
 tunnel mpls traffic-eng bandwidth  75
 tunnel mpls traffic-eng path-option 1 explicit name bottom
 no routing dynamic
!
interface Tunnel3
 ip unnumbered Loopback0
 tunnel destination 17.16.10.44
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng priority 0 0
 tunnel mpls traffic-eng bandwidth sub-pool 30
 tunnel mpls traffic-eng path-option 1 dynamic
 no routing dynamic
!
interface Tunnel4
 ip unnumbered Loopback0
 tunnel destination 17.16.10.44
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng priority 0 0
 tunnel mpls traffic-eng bandwidth  75
 tunnel mpls traffic-eng path-option 1 explicit name top
 no routing dynamic
!
interface FastEthernet0/0
 bandwidth 200
 ip address 192.16.10.33 255.255.255.240
 ip policy route-map foo
 duplex half
 service-policy input PE-C
 service-policy output PE-P
!
interface Serial1/0
```

```
 description connection to r2
 bandwidth 100
 ip address 192.16.10.85 255.255.255.252
 mpls ip
 mpls traffic-eng tunnels
 serial restart-delay 0
 service-policy output PE-P2
 ip rsvp bandwidth 100 sub-pool 30
!
interface Serial2/0
 description connection to r5
 bandwidth 100
 ip address 192.16.10.82 255.255.255.252
 mpls ip
 mpls traffic-eng tunnels
 serial restart-delay 0
 service-policy output PE-P2
 ip rsvp bandwidth 100 sub-pool 30
!
router ospf 1
 mpls traffic-eng router-id Loopback0
 mpls traffic-eng area 0
 router-id 17.16.10.1
 log-adjacency-changes
 redistribute connected
 network 17.16.10.11 0.0.0.0 area 0
 network 192.16.10.32 0.0.0.15 area 0
 network 192.16.10.80 0.0.0.3 area 0
 network 192.16.10.84 0.0.0.3 area 0
!
ip route 192.16.10.0 255.255.255.224 Tunnel2
ip route 192.16.10.72 255.255.255.248 Tunnel4
!
ip explicit-path name bottom enable
 next-address 192.16.10.81
 next-address 192.16.10.97
 next-address 192.16.10.93
!
ip explicit-path name top enable
 next-address 192.16.10.86
 next-address 192.16.10.90
 next-address 192.16.10.94
!
logging alarm informational
access-list 101 permit ip any host 192.16.10.3 dscp ef
access-list 102 permit ip any host 192.16.10.75 dscp ef
!
route-map foo permit 10
 match ip address 101 102
 set interface Tunnel1 Tunnel3
!
```

b. Konfigurasi *Router* r2

```
!
hostname r2
!
enable secret 5 $1$Topm$9m5jIl3cFaM6R6SpuUJK2.
!
ip cef
!
```

```
mpls traffic-eng tunnels
!
class-map match-all gold
 match mpls experimental topmost 5
class-map match-all bronze
 match mpls experimental topmost 1
class-map match-all bulk
 match ip dscp af11
class-map match-all silver
 match mpls experimental topmost 3
class-map match-all bulk1
 match ip dscp af11
class-map match-all voice1
 match ip dscp ef
class-map match-all mission-critical
 match ip dscp af31
class-map match-all voice
 match ip dscp ef
class-map match-all mission-critical1
 match ip dscp af31
!
policy-map PE-P2
 class voice1
  priority percent 30
 class mission-critical1
  bandwidth percent 25
  random-detect dscp-based
 class bulk1
  bandwidth percent 20
  random-detect dscp-based
policy-map PE-P
 class gold
  priority percent 30
 class silver
  bandwidth percent 25
  random-detect
 class bronze
  bandwidth percent 20
  random-detect
policy-map PE-C
 class voice
    police 10000 1875 3750 conform-action set-mpls-exp-imposition-
transmit 5 exceed-action drop
 class mission-critical
    police 15000 2813 5625 conform-action set-mpls-exp-imposition-
transmit 3 exceed-action set-mpls-exp-imposition-transmit 3
 class bulk
    police 12500 2344 4688 conform-action set-mpls-exp-imposition-
transmit 1 exceed-action set-mpls-exp-imposition-transmit 1
 class class-default
    police 12500 2344 4688 conform-action set-mpls-exp-imposition-
transmit 0 exceed-action drop
!
interface Loopback0
 ip address 17.16.10.22 255.255.255.255
!
interface Tunnel1
 ip unnumbered Loopback0
 tunnel destination 17.16.10.44
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng priority 0 0
```

```
 tunnel mpls traffic-eng bandwidth sub-pool 30
 tunnel mpls traffic-eng path-option 1 dynamic
 no routing dynamic
!
interface Tunnel2
 ip unnumbered Loopback0
 tunnel destination 17.16.10.44
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng priority 0 0
 tunnel mpls traffic-eng bandwidth  75
 tunnel mpls traffic-eng path-option 1 explicit name bottom
 no routing dynamic
!
interface Tunnel3
 ip unnumbered Loopback0
 tunnel destination 17.16.10.55
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng priority 0 0
 tunnel mpls traffic-eng bandwidth sub-pool 30
 tunnel mpls traffic-eng path-option 1 dynamic
 no routing dynamic
!
interface Tunnel4
 ip unnumbered Loopback0
 tunnel destination 17.16.10.55
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng priority 0 0
 tunnel mpls traffic-eng bandwidth  75
 tunnel mpls traffic-eng path-option 1 explicit name top
 no routing dynamic
!
interface FastEthernet0/0
 bandwidth 200
 ip address 192.16.10.65 255.255.255.248
 ip policy route-map fo2
 duplex half
 service-policy input PE-C
 service-policy output PE-P2
!
interface Serial1/0
 description connection to r1
 bandwidth 100
 ip address 192.16.10.86 255.255.255.252
 mpls ip
 mpls traffic-eng tunnels
 serial restart-delay 0
 service-policy output PE-P
 ip rsvp bandwidth 100 sub-pool 35
!
interface Serial2/0
 description connection to r3
 bandwidth 100
 ip address 192.16.10.89 255.255.255.252
 mpls ip
 mpls traffic-eng tunnels
 serial restart-delay 0
 service-policy output PE-P
 ip rsvp bandwidth 100 sub-pool 35
!
router ospf 2
 mpls traffic-eng router-id Loopback0
```

```
 mpls traffic-eng area 0
 router-id 17.16.10.2
 log-adjacency-changes
 redistribute connected
 network 17.16.10.22 0.0.0.0 area 0
 network 192.16.10.64 0.0.0.7 area 0
 network 192.16.10.84 0.0.0.3 area 0
 network 192.16.10.88 0.0.0.3 area 0
!
ip route 192.16.10.48 255.255.255.240 Tunnel4
ip route 192.16.10.72 255.255.255.248 Tunnel2
!
ip explicit-path name bottom enable
 next-address 192.16.10.85
 next-address 192.16.10.81
 next-address 192.16.10.97
!
ip explicit-path name top enable
 next-address 192.16.10.90
 next-address 192.16.10.94
 next-address 192.16.10.98
!
logging alarm informational
access-list 101 permit ip any host 192.16.10.75 dscp ef
access-list 102 permit ip any host 192.16.10.51 dscp ef
!
route-map fo2 permit 10
 match ip address 101 102
 set interface Tunnel1 Tunnel3
!
```

c. Konfigurasi *Router* r3

```
!
hostname r3
!
enable secret 5 $1$p/bF$t.bHNj/1ZJvYva65woOsl.
!
ip cef
!
no ip domain lookup
!
mpls traffic-eng tunnels
!
class-map match-all gold
 match mpls experimental topmost 5
class-map match-all bronze
 match mpls experimental topmost 1
class-map match-all bulk
 match ip dscp af11
class-map match-all silver
 match mpls experimental topmost 3
class-map match-all bulk1
 match ip dscp af11
class-map match-all voice1
 match ip dscp ef
class-map match-all mission-critical
 match ip dscp af31
class-map match-all voice
 match ip dscp ef
class-map match-all mission-critical1
```

```
 match ip dscp af31
!
policy-map PE-P2
 class gold
  priority percent 30
 class silver
  bandwidth percent 25
  random-detect
 class bronze
  bandwidth percent 20
  random-detect
policy-map PE-P
 class voice1
  priority percent 30
 class mission-critical1
  bandwidth percent 25
  random-detect dscp-based
 class bulk1
  bandwidth percent 20
  random-detect dscp-based
policy-map PE-C
 class voice
    police 10000 1875 3750 conform-action set-mpls-exp-imposition-
transmit 5 exceed-action drop
 class mission-critical
    police 15000 2813 5625 conform-action set-mpls-exp-imposition-
transmit 3 exceed-action set-mpls-exp-imposition-transmit 3
 class bulk
    police 12500 2344 4688 conform-action set-mpls-exp-imposition-
transmit 1 exceed-action set-mpls-exp-imposition-transmit 1
 class class-default
    police 12500 2344 4688 conform-action set-mpls-exp-imposition-
transmit 0 exceed-action drop
!
interface Loopback0
 ip address 17.16.10.33 255.255.255.255
!
interface Tunnel1
 ip unnumbered Loopback0
 tunnel destination 17.16.10.11
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng priority 0 0
 tunnel mpls traffic-eng bandwidth sub-pool 30
 tunnel mpls traffic-eng path-option 1 dynamic
 no routing dynamic
!
interface Tunnel2
 ip unnumbered Loopback0
 tunnel destination 17.16.10.11
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng priority 0 0
 tunnel mpls traffic-eng bandwidth  75
 tunnel mpls traffic-eng path-option 1 explicit name bottom
 no routing dynamic
!
interface Tunnel3
 ip unnumbered Loopback0
 tunnel destination 17.16.10.55
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng priority 0 0
 tunnel mpls traffic-eng bandwidth sub-pool 30
```

```
 tunnel mpls traffic-eng path-option 1 dynamic
 no routing dynamic
!
interface Tunnel4
 ip unnumbered Loopback0
 tunnel destination 17.16.10.55
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng priority 0 0
 tunnel mpls traffic-eng bandwidth  75
 tunnel mpls traffic-eng path-option 1 explicit name top
 no routing dynamic
!
interface FastEthernet0/0
 bandwidth 200
 ip address 192.16.10.1 255.255.255.224
 ip policy route-map fo3
 duplex half
 service-policy input PE-C
 service-policy output PE-P
!
interface Serial1/0
 description connection to r2
 bandwidth 100
 ip address 192.16.10.90 255.255.255.252
 mpls ip
 mpls traffic-eng tunnels
 serial restart-delay 0
 service-policy output PE-P2
 ip rsvp bandwidth 100 sub-pool 30
 ip rsvp resource-provider none
!
interface Serial2/0
 description connection to r4
 bandwidth 100
 ip address 192.16.10.93 255.255.255.252
 mpls ip
 mpls traffic-eng tunnels
 serial restart-delay 0
 service-policy output PE-P2
 ip rsvp bandwidth 100 sub-pool 30
!
router ospf 3
 mpls traffic-eng router-id Loopback0
 mpls traffic-eng area 0
 router-id 17.16.10.3
 log-adjacency-changes
 redistribute connected
 network 17.16.10.33 0.0.0.0 area 0
 network 192.16.10.0 0.0.0.31 area 0
 network 192.16.10.88 0.0.0.3 area 0
 network 192.16.10.92 0.0.0.3 area 0
!
ip route 192.16.10.32 255.255.255.240 Tunnel2
ip route 192.16.10.48 255.255.255.240 Tunnel4
no ip http server
no ip http secure-server
!
ip explicit-path name bottom enable
 next-address 192.16.10.94
 next-address 192.16.10.98
 next-address 192.16.10.82
```

```
!
ip explicit-path name top enable
 next-address 192.16.10.89
 next-address 192.16.10.85
 next-address 192.16.10.81
!
logging alarm informational
access-list 101 permit ip any host 192.16.10.32 dscp ef
access-list 102 permit ip any host 192.16.10.51 dscp ef
!
route-map fo3 permit 10
 match ip address 101 102
 set interface Tunnel1 Tunnel3
!
```

d. Konfigurasi *Router* r4

```
!
hostname r4
!
enable secret 5 $1$pZS7$EubLM73BMFEKh7g8/mlXY.
!
ip cef
!
no ip domain lookup
!
mpls traffic-eng tunnels
!
class-map match-all gold
 match mpls experimental topmost 5
class-map match-all bronze
 match mpls experimental topmost 1
class-map match-all bulk
 match ip dscp af11
class-map match-all silver
 match mpls experimental topmost 3
class-map match-all bulk1
 match ip dscp af11
class-map match-all voice1
 match ip dscp ef
class-map match-all mission-critical
 match ip dscp af31
class-map match-all voice
 match ip dscp ef
class-map match-all mission-critical1
 match ip dscp af31
!
policy-map PE-P2
 class gold
  priority percent 30
 class silver
  bandwidth percent 25
  random-detect
 class bronze
  bandwidth percent 20
  random-detect
policy-map PE-P
 class voice1
  priority percent 30
 class mission-critical1
  bandwidth percent 25
```

```
  random-detect dscp-based
 class bulk1
  bandwidth percent 20
  random-detect dscp-based
policy-map PE-C
 class voice
    police 10000 1875 3750 conform-action set-mpls-exp-imposition-
transmit 5 exceed-action drop
 class mission-critical
    police 15000 2813 5625 conform-action set-mpls-exp-imposition-
transmit 3 exceed-action set-mpls-exp-imposition-transmit 3
 class bulk
    police 12500 2344 4688 conform-action set-mpls-exp-imposition-
transmit 1 exceed-action set-mpls-exp-imposition-transmit 1
 class class-default
    police 12500 2344 4688 conform-action set-mpls-exp-imposition-
transmit 0 exceed-action drop
!
interface Loopback0
 ip address 17.16.10.44 255.255.255.255
!
interface Tunnel1
 ip unnumbered Loopback0
 tunnel destination 17.16.10.22
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng priority 0 0
 tunnel mpls traffic-eng bandwidth sub-pool 30
 tunnel mpls traffic-eng path-option 1 dynamic
 no routing dynamic
!
interface Tunnel2
 ip unnumbered Loopback0
 tunnel destination 17.16.10.22
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng priority 0 0
 tunnel mpls traffic-eng bandwidth  75
 tunnel mpls traffic-eng path-option 1 explicit name bottom
 no routing dynamic
!
interface Tunnel3
 ip unnumbered Loopback0
 tunnel destination 17.16.10.11
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng priority 0 0
 tunnel mpls traffic-eng bandwidth sub-pool 30
 tunnel mpls traffic-eng path-option 1 dynamic
 no routing dynamic
!
interface Tunnel4
 ip unnumbered Loopback0
 tunnel destination 17.16.10.11
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng priority 0 0
 tunnel mpls traffic-eng bandwidth  75
 tunnel mpls traffic-eng path-option 1 explicit name top
 no routing dynamic
!
interface FastEthernet0/0
 bandwidth 200
 ip address 192.16.10.73 255.255.255.248
 ip policy route-map fo4
```

```
 duplex half
 service-policy input PE-C
 service-policy output PE-P
!
interface Serial1/0
 description connection to r3
 bandwidth 100
 ip address 192.16.10.94 255.255.255.252
 mpls ip
 mpls traffic-eng tunnels
 serial restart-delay 0
 service-policy output PE-P2
 ip rsvp bandwidth 100 sub-pool 30
!
interface Serial2/0
 description connection to r5
 bandwidth 100
 ip address 192.16.10.97 255.255.255.252
 mpls ip
 mpls traffic-eng tunnels
 serial restart-delay 0
 service-policy output PE-P2
 ip rsvp bandwidth 100 sub-pool 30
!
router ospf 4
 mpls traffic-eng router-id Loopback0
 mpls traffic-eng area 0
 router-id 17.16.10.4
 log-adjacency-changes
 redistribute connected
 network 17.16.10.44 0.0.0.0 area 0
 network 192.16.10.72 0.0.0.7 area 0
 network 192.16.10.92 0.0.0.3 area 0
 network 192.16.10.96 0.0.0.3 area 0
!
ip route 192.16.10.32 255.255.255.240 Tunnel4
ip route 192.16.10.64 255.255.255.248 Tunnel2
no ip http server
no ip http secure-server
!
ip explicit-path name bottom enable
 next-address 192.16.10.98
 next-address 192.16.10.82
 next-address 192.16.10.86
!
ip explicit-path name top enable
 next-address 192.16.10.93
 next-address 192.16.10.89
 next-address 192.16.10.85
!
logging alarm informational
access-list 101 permit ip any host 192.16.10.64 dscp ef
access-list 102 permit ip any host 192.16.10.32 dscp ef
!
route-map fo4 permit 10
 match ip address 101 102
 set interface Tunnel1 Tunnel3
!
```

e. Konfigurasi router r5

```
!
hostname r5
!
enable secret 5 $1$QKQJ$gi7TjeK3MrIbwn2ru2mE50
!
ip cef
!
no ip domain lookup
!
mpls traffic-eng tunnels
!
class-map match-all gold
 match mpls experimental topmost 5
class-map match-all bronze
 match mpls experimental topmost 1
class-map match-all bulk
 match ip dscp af11
class-map match-all silver
 match mpls experimental topmost 3
class-map match-all bulk1
 match ip dscp af11
class-map match-all voice1
 match ip dscp ef
class-map match-all mission-critical
 match ip dscp af31
class-map match-all voice
 match ip dscp ef
class-map match-all mission-critical1
 match ip dscp af31
!
policy-map PE-P2
 class voice1
  priority percent 30
 class mission-critical1
  bandwidth percent 25
  random-detect dscp-based
 class bulk1
  bandwidth percent 20
  random-detect dscp-based
policy-map PE-P
 class gold
  priority percent 30
 class silver
  bandwidth percent 25
  random-detect
 class bronze
  bandwidth percent 20
  random-detect
policy-map PE-C
 class voice
    police 10000 1875 3750 conform-action set-mpls-exp-imposition-
transmit 5 exceed-action drop
 class mission-critical
    police 15000 2813 5625 conform-action set-mpls-exp-imposition-
transmit 3 exceed-action set-mpls-exp-imposition-transmit 3
 class bulk
    police 12500 2344 4688 conform-action set-mpls-exp-imposition-
transmit 1 exceed-action set-mpls-exp-imposition-transmit 1
 class class-default
```

```
    police 12500 2344 4688 conform-action set-mpls-exp-imposition-
transmit 0 exceed-action drop
!
interface Loopback0
 ip address 17.16.10.55 255.255.255.255
!
interface Tunnel1
 ip unnumbered Loopback0
 tunnel destination 17.16.10.22
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng priority 0 0
 tunnel mpls traffic-eng bandwidth sub-pool 30
 tunnel mpls traffic-eng path-option 1 dynamic
 no routing dynamic
!
interface Tunnel2
 ip unnumbered Loopback0
 tunnel destination 17.16.10.22
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng priority 0 0
 tunnel mpls traffic-eng bandwidth  75
 tunnel mpls traffic-eng path-option 1 explicit name bottom
 no routing dynamic
!
interface Tunnel3
 ip unnumbered Loopback0
 tunnel destination 17.16.10.33
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng priority 0 0
 tunnel mpls traffic-eng bandwidth sub-pool 30
 tunnel mpls traffic-eng path-option 1 dynamic
 no routing dynamic
!
interface Tunnel4
 ip unnumbered Loopback0
 tunnel destination 17.16.10.33
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng priority 0 0
 tunnel mpls traffic-eng bandwidth  75
 tunnel mpls traffic-eng path-option 1 explicit name top
 no routing dynamic
!
interface FastEthernet0/0
 bandwidth 200
 ip address 192.16.10.49 255.255.255.240
 ip policy route-map fo5
 duplex half
 service-policy input PE-C
 service-policy output PE-P2
!
interface Serial1/0
 description connection to r1
 bandwidth 100
 ip address 192.16.10.81 255.255.255.252
 mpls ip
 mpls traffic-eng tunnels
 serial restart-delay 0
 service-policy output PE-P
 ip rsvp bandwidth 100 sub-pool 30
!
interface Serial2/0
```

```
 description connection to r4
 bandwidth 100
 ip address 192.16.10.98 255.255.255.252
 mpls ip
 mpls traffic-eng tunnels
 serial restart-delay 0
 service-policy output PE-P
 ip rsvp bandwidth 100 sub-pool 30
!
router ospf 5
 mpls traffic-eng router-id Loopback0
 mpls traffic-eng area 0
 router-id 17.16.10.5
 log-adjacency-changes
 redistribute connected
 network 17.16.10.55 0.0.0.0 area 0
 network 192.16.10.48 0.0.0.15 area 0
 network 192.16.10.80 0.0.0.3 area 0
 network 192.16.10.96 0.0.0.3 area 0
!
ip route 192.16.10.0 255.255.255.224 Tunnel4
ip route 192.16.10.64 255.255.255.248 Tunnel2
no ip http server
no ip http secure-server
!
ip explicit-path name bottom enable
 next-address 192.16.10.97
 next-address 192.16.10.93
 next-address 192.16.10.89
!
ip explicit-path name top enable
 next-address 192.16.10.82
 next-address 192.16.10.86
 next-address 192.16.10.90
!
logging alarm informational
access-list 101 permit ip any host 192.16.10.64 dscp ef
access-list 102 permit ip any host 192.16.10.0 dscp ef
!
route-map fo5 permit 10
 match ip address 101 102
 set interface Tunnel1 Tunnel3
!
```

Lampiran 4. Konfigurasi *Packet Generator*

Untuk membangkitkan trafik dengan Packgen 0.2, maka perlu dilakukan konfigurasi terlebih dahulu. Berikut ini adalah konfigurasi file listen.yml yang digunakan untuk mendengar pada port-port tertentu dan konfigurasi file listen_and_send.yml yang digunakan untuk mengirimkan paket-paket pada host A. File pada poin e sampai dengan h digunakan untuk menghasilkan paket-paket yang kemudian digunakan pada percobaan pertama (P1) untuk host A. File pada poin i sampai dengan l digunakan untuk menghasilkan paket-paket yang kemudian digunakan pada percobaan kedua (P2) untuk host A. Sedangkan untuk host-host lain digunakan konfigurasi serupa namun dengan nomor port yang berbeda sesuai keinginan.

a. Konfigurasi File listenA1.yml

```
LISTEN:
  tcp:
    -
      ports: !ruby/range 5000..5000
```

b. Konfigurasi File listenA2.yml

```
LISTEN:
  tcp:
    -
      ports: !ruby/range 5001..5001
```

c. Konfigurasi File listenA3.yml

```
LISTEN:
  tcp:
    -
      ports: !ruby/range 5002..5002
```

d. Konfigurasi File listenA4.yml

```
LISTEN:
  udp:
    -
      ports: !ruby/range 5003..5003
```

e. Konfigurasi File listen_and_send_A1a.yml

```
LISTEN:
  tcp:
    -
      ports: !ruby/range 5000..5000
      from..to: !ruby/range 0.0..62.0
```

```
SEND:
 tcp:
  -
    name: mission critical
    host: 192.16.10.3:5000
    bandwidth: 7500b
    packet_size: 1000B
    dscp: af31
    from..to: !ruby/range 1.0..61.0
```

f. Konfigurasi File listen_and_send_A1b.yml

```
LISTEN:
 tcp:
  -
    ports: !ruby/range 5001..5001
    from..to: !ruby/range 0.0..62.0

SEND:
 tcp:
  -
    name: bulk
    host: 192.16.10.3:5001
    bandwidth: 6250b
    packet_size: 1000B
    dscp: af11
    from..to: !ruby/range 11.0..61.0
```

g. Konfigurasi File listen_and_send_A1c.yml

```
LISTEN:
 tcp:
  -
    ports: !ruby/range 5000..5002
    from..to: !ruby/range 0.0..62.0

SEND:
 tcp:
  -
    name: best effort
    host: 192.16.10.3:5002
    bandwidth: 6250b
    packet_size: 1000B
    from..to: !ruby/range 21.0..61.0
```

h. Konfigurasi File listen_and_send_A1d.yml

```
LISTEN:
 udp:
  -
    ports: !ruby/range 5003..5003
    from..to: !ruby/range 0.0..62.0
```

```
SEND:
  udp:
    -
      name: voice
      host: 192.16.10.3:5003
      bandwidth: 5000b
      packet_size: 40B
      dscp: ef
      from..to: !ruby/range 31.0..61.0
```

i. Konfigurasi File listen_and_send_A2a.yml

```
LISTEN:
  tcp:
    -
      ports: !ruby/range 5000..5002
      from..to: !ruby/range 0.0..62.0

SEND:
  tcp:
    -
      name: mission critical
      host: 192.16.10.3:5000
      bandwidth: 7500b
      packet_size: 1000B
      dscp: af31
      from..to: !ruby/range 1.0..61.0
```

j. Konfigurasi File listen_and_send_A2b.yml

```
LISTEN:
  tcp:
    -
      ports: !ruby/range 5000..5002
      from..to: !ruby/range 0.0..62.0

SEND:
  tcp:
    -
      name: bulk
      host: 192.16.10.3:5001
      bandwidth: 6250b
      packet_size: 1000B
      dscp: af11
      from..to: !ruby/range 11.0..61.0
```

k. Konfigurasi File listen_and_send_A2c.yml

```
LISTEN:
  tcp:
    -
      ports: !ruby/range 5000..5002
      from..to: !ruby/range 0.0..62.0
```

```
SEND:
  tcp:
    -
      name: best effort
      host: 192.16.10.3:5002
      bandwidth: 6250b
      packet_size: 1000B
      from..to: !ruby/range 21.0..61.0
```

l. Konfigurasi File listen_and_send_A2d.yml

```
LISTEN:
  udp:
    -
      ports: !ruby/range 5003..5003
      from..to: !ruby/range 0.0..62.0

SEND:
  udp:
    -
      name: voice
      host: 192.16.10.3:5003
      bandwidth: 5000b
      packet_size: 40B
      dscp: ef
      from..to: !ruby/range 31.0..61.0
```

Lampiran 5. PHB *Class Selector*

Simpul DiffServ memetakan paket-paket ke dalam PHB berdasarkan DSCP-nya. Tabel dibawah menunjukkan pemetaan *Class Selector* (CS) yang direkomendasikan oleh arsitektur ini. *Class Selector* (CS) adalah pengecualian, karena didefinisikan sebagai kompatibilitas dengan penggunaan *field precedence* pada oktet ToS IPv4.

Tabel 1. Pemetaan Antara PHB dan DSCP yang Direkomendasikan IETF

PHB	DSCP (Desimal)	DSCP (Biner)
CS7	56	111000
CS6	48	110000
CS5	40	101000
CS4	32	100000
CS3	24	011000
CS2	16	010000
CS1	8	001000

Sumber: Santiago Alvarez (2006)

PHB CS didefinisikan sebagai kompatibilitas dengan penggunaan IP *precedence* pada oktet ToS IPv4. Tidak ada karakterisasi *latency*, *jitter* atau *loss*.

Tabel 2. Hubungan antara Nilai IP *Precedence* dan CS

PHB	DSCP (Desimal)	DSCP (Biner)	*Precedence Name*	*Precedence* (Biner)	*Precedence* (Desimal)
CS7	56	111000	*Network Control*	111	7
CS6	48	110000	*Internetwork Control*	110	6
CS5	40	101000	*Critic/ECP*	101	5
CS4	32	100000	*Flash Override*	100	4
CS3	24	011000	*Flash*	011	3
CS2	16	010000	*Immediate*	010	2
CS1	8	001000	*Priority*	001	1

Sumber: Santiago Alvarez (2006)