RANCANG BANGUN RTP CHUNK – PACKET ENCAPSULATOR DATA AV STREAM FORMAT RTP PADA MULTI-SOURCE STREAMING SERVER

Angki[1], A Subkhan KH, ST [2]
[1]Mahasiswa Politeknik Elektronika Negeri Surabaya, Jurusan Teknik Telekomunikasi
[2] Dosen Politeknik Elektronika Negeri Surabaya Institut Teknologi Sepuluh Nopember
Kampus ITS, Surabaya 60111
e-mail : angkis@student.eepis-its.edu e-mail : subhankh@eepis-its.edu,

Abstrak

Berdasarkan pada perkembangan teknologi multimedia, penyiaran dan layanan televisi, dalam proyek akhir ini akan mendesain dan membuat RTP *chunk - packet encapsulator* yang diimplementasikan untuk *multi-source streaming server*. Ketika data *audio/video* diambil dari *server*. Data AV diolah pada *encapsulator* dengan metode *multiplexing*, kemudian dilakukan proses enkapsulasi menggunakan *protocol* RTP (Real-Time Protocol), menjadi paket *chunk* yang kemudian dikirim melalui jaringan internet. Selanjutnya akan dianalisa hasil multiplexing dan enkapsulasi serta parameter – parameter *QoS* agar menjadi sistem yang handal.

Hasil yang diharapkan dari penelitian ini didapat suatu *system multiplexing encapsulator* yang memiliki kecepatan tinggi, hemat *bandwidth* dan efisien, sehingga di masa mendatang dapat diterapkan dan diimplementasikan pada arsitektur IPTV.

Keyword : RTP, *Multiplexing*, *Encapsulator*, MSS, IPTV

1. Pendahuluan

Pengiriman sinyal gambar, suara, dan data telah menggunakan sistem transmisi *digital*. Pengembangan teknologi ini menjadi siaran televisi yang melalui jaringan internet, disebut juga dengan Teknologi IPTV. Keunggulan yang dimiliki teknologi ini adalah kualitas gambar dan warna yang dihasilkan jauh lebih bagus daripada televisi *analog*. Desain dan implementasi sistem siaran TV digital terutama ditujukan pada peningkatan kualitas gambar. Namun, masih banyak arsitektur yang distribusi dari satu *source streaming server* dan kualitas siaran yang dihasilkan teknologi ini masih sangat tergantung pada kemampuan proses *streaming* dari *server*, ketersediaan *bandwidth*, stabilitas akses internet atau jaringan IP yang digunakan.

Dalam proyek akhir ini akan merancang RTP *chunk – packet encapsulator* untuk multiplexing dan mengenkapsulasi data audio dan video , penelitian dilakukan dengan menganalisa proses multiplexing packet, *throughput* (kecepatan transfer), serta enkapsulasi *chunk – packet* dengan optimal untuk mendapatkan sistem yang diharapkan dapat bermanfaat sebagai alternatif model distribusi layanan IPTV *server*.

2. Teori Penunjang

Aplikasi Jaringan Multimedia

Dalam dunia internet terdapat banyak sekali aplikasi multimedia. Ada tiga kelas aplikasi multimedia saat ini[1]. Contoh aplikasi multimedia :

1. Streaming stored audio and video. Pada aplikasi ini *client* akan meminta atau *merequest video* dan *audio* yang tersimpan pada *server*.
2. Streaming Live Audio dan Video. Pada aplikasi ini sama seperti sistem *broadcast* radio atau televisi, bedanya pada sistem ini menggunakan jaringan internet untuk transmisinya. Aplikasi ini memungkinkan *user* dapat menerima siaran radio atau televisi dari manapun.

3. Real Time Interactive Audio and Video. Pada aplikasi ini memungkinkan *user* dapat menggunakan aplikasi *audio/video streaming* untuk berkomunikasi dengan *user* ditempat yang berbeda secara *real time*, seperti VoIP dan Video conference.

Video Streaming

Video streaming merupakan bidang yang menarik untuk dijelajahi karena relatif baru dengan biaya yang cukup murah dengan semakin murahnya peralatan elektronik. Aplikasi dari *Video Streaming* salah satunya untuk *memonitoring* atau pemantauan kondisi ruangan, informasi video akan dikirmkan melalui saluran komunikasi, termasuk jaringan, kabel telepon, saluran ISDN atau radio. Informasi video mempunyai *bandwidth* yang lebar (sangat banyak *byte* per detik yang dikirimkan), yang oleh karenanya sangat membutuhkan teknologi kompresi video untuk mengurangi kebutuhan *bandwidth* sebelum dikirimkan melalui saluran komunikasi. Peralatan yang perlu ditambahkan hanya kamera video sederhana. Sekedar gambaran singkat, sebuah kanal video yang baik tanpa dikompresi akan mengambil *bandwidth* sekitar 9 Mbps. Dengan teknik kompresi yang sudah ada pada hari ini, kita dapat menghemat sebuah kanal video sekitar 30 Kbps. Itu berarti sebuah saluran *Internet atau LAN* yang tidak terlalu cepat sebetulnya dapat digunakan untuk menyalurkan video.

Enkapsulasi

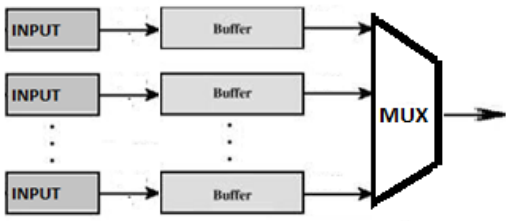
Konsep penempatan data dibalik suatu header dan trailer untuk tiap layer disebut enkapsulasi (*encapsulation*). Pada Gambar 1. terlihat pada tiap layer diberikan suatu *header* tambahan, kemudian ditambahkan lagi *header* pada layer berikutnya, sedangkan pada layer 2 selain ditambahkan *header* juga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

Gambar 1 Blok Sistem Encapsulasi

Pada pemrosesan layer 5, 6 dan 7 terkadang tidak diperlukan adanya *header* Ini dikarenakan tidak ada informasi baru yang perlu diproses. Sehingga untuk layer tersebut biasa dianggap satu proses.

Multiplexing

Multiplexer mengirim data dari beberapa input kedalam satu kanal, dimana pada setiap packet yang akan ditransferkan ditambahakan informasi *header* yang dikandung oleh paket tersebut. Seperti ditunjukan pada gambar 2.

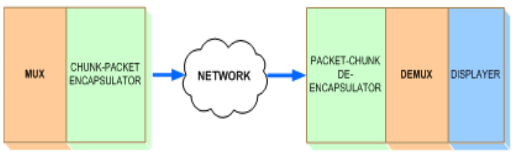


Gambar 2 Multiplexing

3. Perancangan

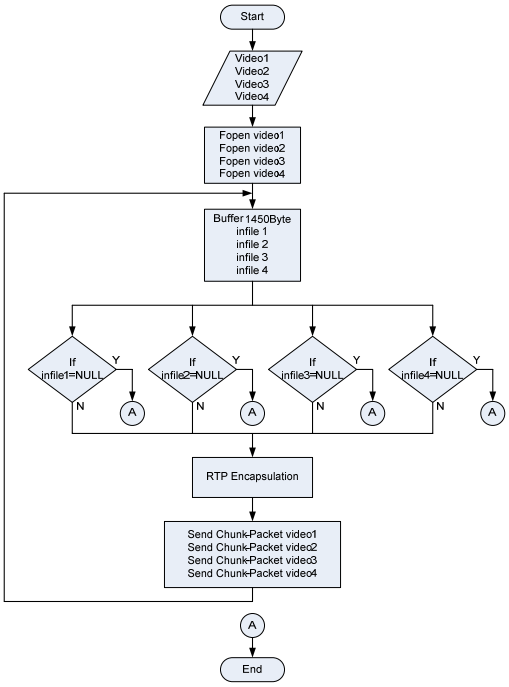
Rancangan Sistem

Sistem ini dibangun dan diimplementasikan untuk *multi-source streaming server*.



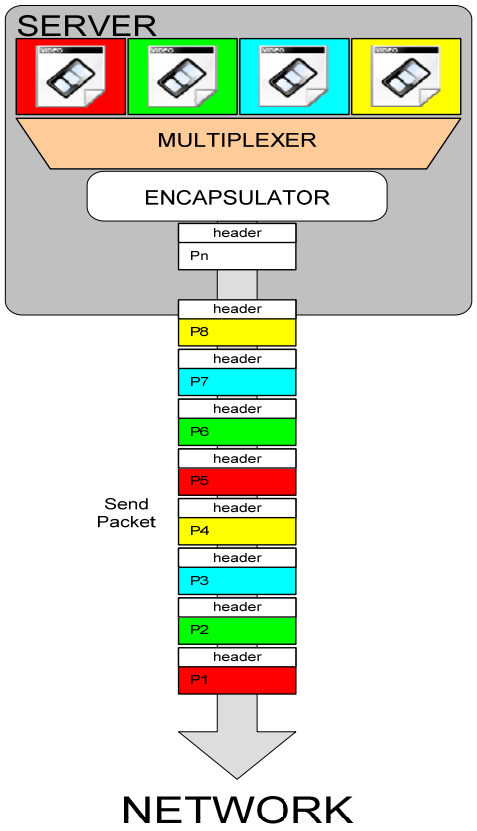
Gambar 3 Blok Diagram Sistem MSS

Prinsip kerja sistem seperti ditunjukan pada gambar 4 yaitu sistem membaca empat file video sebagai *source* yang akan di streamingkan melalui proses *read file*, kemudian data diletakan pada *buffer* sebelum melewati proses enkapsulasi menggunakan protokol rtp menjadi paket rtp, selanjutnya paket dikirimkan ke sisi *client* melalui proses *send chunk-packet*. Proses ini berlangsung berulang hingga semua paket terkirim ke *client*.



Gambar 4 *Flowchart* Sistem

Dari gambar 5, proses *multiplexing* dan enkapsulasi *packet-chunk* oleh sistem dapat dijelaskan proses pertama yaitu, sistem melakukan pengecekan file yang terdiri dari empat video, file video1, file video2, file video3 dan file video4, setelah proses pengecekan selasai akan dilakukan proses pengambilan file video secara *multiplexing*, paket yang telah terbentuk selanjutnya akan dilakukan proses enkapsulasi dengan penambahan *header* rtp menjadi paket rtp dan ditransmisikan ke *client*.



Gambar 5 Proses *multiplexing* dan enkapsulasi *paket-chunk*

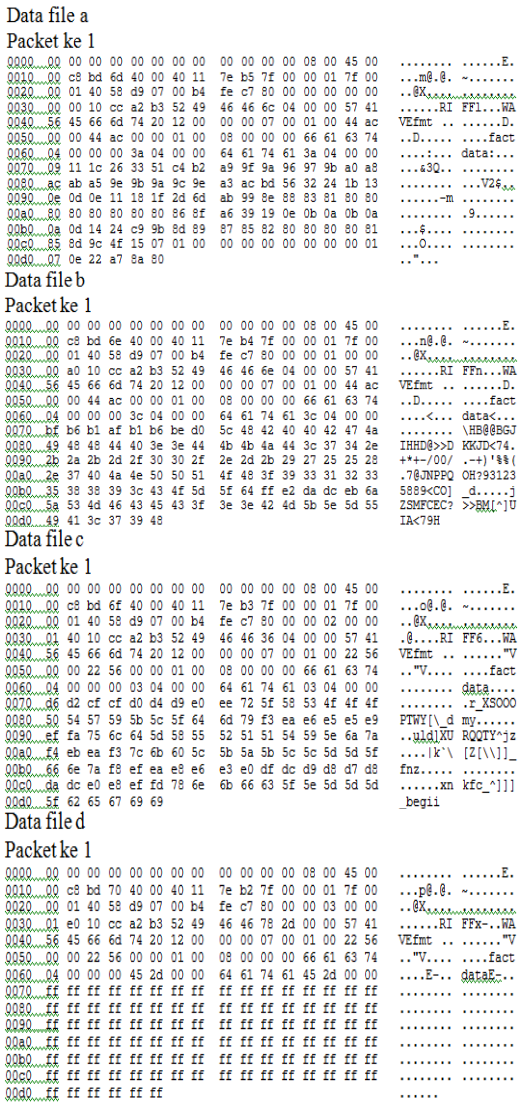
Pengujian

Hasil dari pembuatan *Chunk - Packet Encapsulator* akan dianalisa berdasarkan *multiplexing paket*, proses enkapsulasi paket rtp, nilai *throughput*, *jitter*, *delay*, *packet loss* yang efisien serta masih dalam batas toleransi kualitas aplikasi *video*.

4. Analisa dan Pengujian

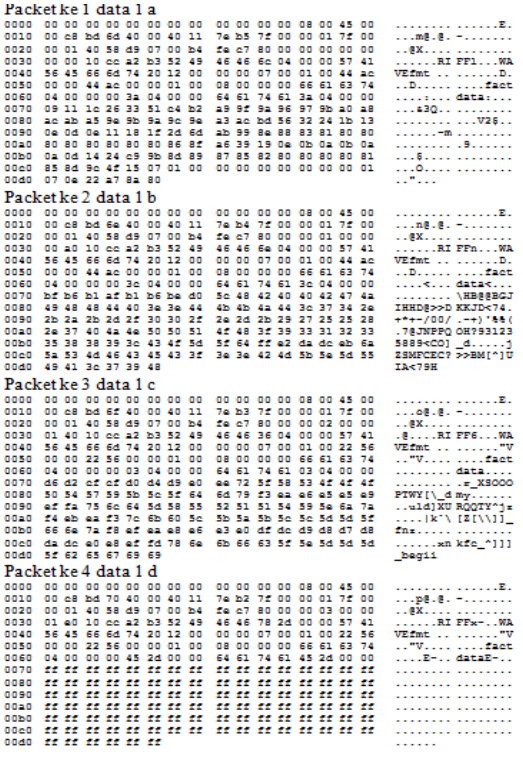
4.1 Pengujian multiplexing

Pada pengujian system awal ini, yang akan kita uji adalah kesesuaian jalannya multiplexing paket saat pengiriman menggunakan tool wireshark, ujicoba dilakukan dengan 4 sample file yaitu file a, file b, file c, file d sebagai source yang akan dikirim. Data file a, b, c,d seperti ditunjukan pada gambar 6.



Gambar 6 Bentuk data file a,b,c,d

Saat pengiriman paket secara multiplexing, didapatkan hasil serperti yang ditunjukan pada gambar 7.



Gambar 7 Urutan pengiriman paket secara *multiplexing*

Dari data hasil capture seperti pada gambar 7 terlihat bahwa paket data yang dikirim oleh system berturut-turut dimulai dari paket ke satu berisi data file a, paket ke dua berisi data file b, paket ke tiga berisi data file c, paket ke empat berisi data file d dan seterusnya hingga semua paket selesai terkirim, dapat dianalisa bahwa sistem telah berhasil melakukan pengiriman paket secara *multiplexing*.

4.2 Analisa MTU dan Penambahan RTP header

Pengujian ini bertujuan untuk mengetahui berapa besar ukuran maksimum MTU (Max Transmission Unit) data/payload yang dapat dipaketkan dengan penambahan rtp header sebesar 12 Bytes terhadap kondisi video dan Fps (Frame per second) yang dikirimkan oleh sistem dengan merubah ukuran buffer data pada program.

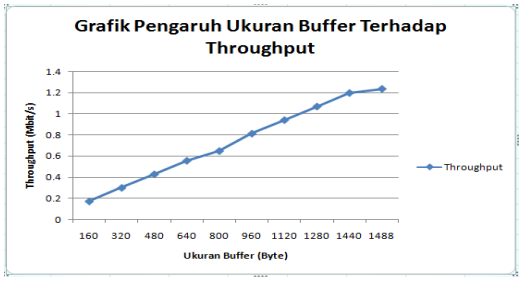
Tabel 1 Pengaruh Ukuran Data, RTP header, MTU Terhadap Kondisi Video dan Fps

Ukuran Buffer	Ukuran Data (Byte)	RTP header (Byte)	MTU (Byte)	Kondisi Video	Fps
160	160	12	172	Baik	25
320	320	12	332	Baik	25
480	480	12	492	Baik	25
640	640	12	652	Baik	25
800	800	12	812	Baik	25
960	960	12	972	Baik	25
1120	1120	12	1132	Baik	25
1280	1280	12	1292	Baik	25
1440	1440	12	1452	Baik	25
1450	1450	12	1462	Baik	25
1488	1488	12	1500	Baik	25
1489	1489	12	1501	Rusak	-

Dari data tabel 1 dapat dianalisa bahwa ukuran *payload* maksimum yang dapat ditampung sebesar 1488 *byte* dengan penambahan RTP *header* 12 bytes total MTU menjadi 1500 *Byte*, video yang diterima oleh *client* dalam kondisi baik. Sebaliknya saat MTU melebihi 1500 *Byte* data video tidak dapat dimainkan atau dalam kondisi rusak, ini dikarenakan terjadi fragmentasi yang tidak diinginkan pada layer yang lebih rendah.

4.3 Analisa Throughput

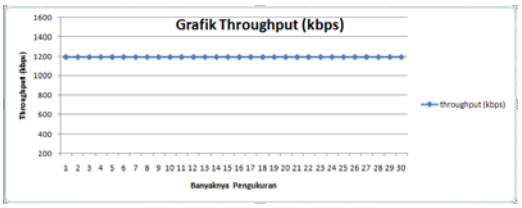
4.3.1 Analisa Pengaruh Ukuran Buffer Terhadap Throughput



Gambar 8 Grafik pengaruh ukuran *buffer* terhadap *throughput*

Berdasarkan grafik pada gambar 8 dapat di analisa bahwa semakin besar ukuran *buffer*, maka nilai *throughput* juga akan semakin besar, Dengan catatan pemilihan ukuran *buffer* setelah ditambahkan *header* rtp, tidak boleh melebihi ukuran MTU sebesar 1500 *Byte*. Jika melebihi ukuran MTU maka akan terjadi fragmentasi yang tidak diinginkan untuk layer yang lebih rendah.

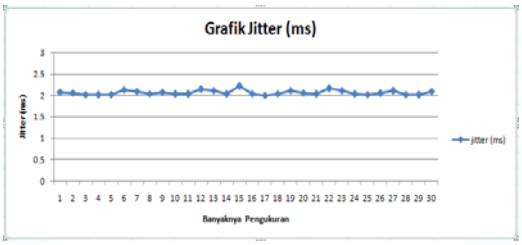
4.3.2 Analisa Rata-rata Throughput



Gambar 9 Grafik analisa throughput

Berdasarkan grafik pada gambar 9 dapat dianalisa bahwa *throughput* yang dihasilkan oleh sistem rata-rata sebesar 1193.408178 kbps. Besarnya nilai *throughput* tergantung oleh besarnya ukuran data dan jumlah paket yang dikirim perdetik, semakin besar ukuran data dan jumlah paket yang dikirim perdetik maka nilai *throughput* akan semakin besar.

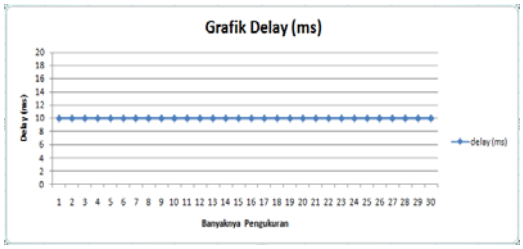
4.4 Analisa Rata-rata Jitter



Gambar 10 Grafik analisa *jitter*

Berdasarkan grafik pada gambar 10 dapat dianalisa bahwa rata-rata *jitter* saat sistem melakukan transmisi paket sebesar 2.068166587 ms. Hal ini disebabkan karena pada port 9999 terjadi pengiriman empat paket, dengan masing-masing ukuran paket sebesar 1450 byte yang dikirimkan secara berurutan, sehingga pada port 9999 ini terjadi pembebanan empat kali lebih besar.

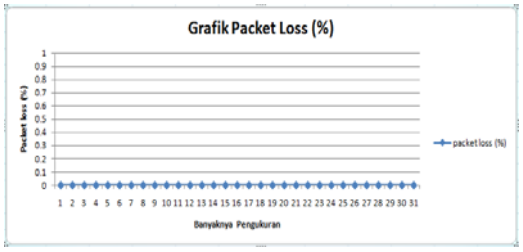
4.5 Analisa Rata-rata Delay



Gambar 11 Grafik analisa delay

Berdasarkan grafik pada gambar 11 Nilai delay ini merupakan rata-rata selisih waktu antara pengiriman paket pertama dengan paket berikutnya kesisi client dalam durasi waktu selama 2 menit. Pangukuran ini dilakukan menggunakan software wireshark. Dari grafik menujukan bahwa rata-rata *delay* saat sistem melakukan transmisi paket sebesar 9.999297801 ms. Nilai ini sangat baik sesuai *standard* ITU-T G.1010(< 10s).

4.6 Analisa Packet Loss



Gambar 12 Grafik analisa packet loss

Berdasarkan grafik pada gambar 12 dapat dijelaskan bahwa rata-rata *packet loss* saat sistem melakukan transmisi paket sebesar 0 % atau tidak adanya paket yang hilang. nilai ini sangat baik sesuai *standard* ITU-T (< 1%). Nilai *packet loss* 0% ini bisa terjadi karena dalam komunikasi UDP satu arah (*one way*), sangat kecil kemungkinan terjadinya *collision* dan antrian paket yang mengakibatkan ada paket yang hilang.

5. PENUTUP

5.1 Kesimpulan

Dari hasil pengujian, pengambilan data dan analisa sistem serta *Quality of Service* (QoS), maka dapat diambil beberapa kesimpulan, yaitu :

1. Proses *multiplexing* yang dilakukan oleh sistem, dengan cara mendeteksi empat file video, kemudian mengambil datanya secara berurutan dari file video 1, file video 2, file video 3, dan file video 4 untuk dienkapsulasi menjadi paket-paket rtp yang dikirim ke *client*.
2. Dengan ukuran data sebesar 1450 Byte didapatkan nilai rata-rata *throughput* untuk 30 kali pengiriman sebesar 1193.408 Kbps.
3. Rata-rata *jitter* yang didapatkan sebesar 2.068 ms. Nilai disebabkan karena terjadi pembebanan empat kali lebih besar pada port 9999. Pada standart ITU-T G.1010 tidak ada ketetapan nilai *jitter* untuk aplikasi video *one way*.
4. *Delay* rata-rata untuk 30 kali pengiriman sebesar 9.999 ms, nilai ini sangat baik sesuai *standard* ITU-T G.1010(< 10s).
5. *Packet loss* yang didapatkan sebesar 0%, nilai ini juga sangat baik sesuai *standard* ITU-T (< 1%).
6. Ukuran *Buffer* untuk data *video* maksimum yang dapat digunakan adalah sebesar 1488 *Byte*. Setelah ditambah *header* rtp sebesar 12 Byte, ukuran paket tidak boleh melebihi ukuran MTU sebesar 1500 *Byte*, jika melebihi ukuran MTU maka akan terjadi fragmentasi yang tidak diinginkan, sehingga data video tidak dapat dimainkan oleh *client*.

5.2 Saran

Pada proyek akhir ini terdapat beberapa kekurangan sehingga terdapat beberapa kekurangan sehingga perlu dilakukan pengembangan. Beberapa saran untuk pengembangan proyek akhir ini adalah :

1. Sistem ini dapat dikembangkan dengan menambah beberapa *streaming server* yang berdiri sendiri untuk menyediakan *source file video* yang akan dikirim.
2. Sistem ini masih memiliki kekurangan, karena belum cukup baik untuk menangani *flowcontrol packet*.

Daftar Pustaka

[1]. James F. Kurose, Keith W. Ross, ”Computer networking : a top-down approach 4th ed”, 2008.

[2]. Ben Haj Amara Wiem, Genevois Vincent, Lamoriniere Cedric, Sliva Johann,” Multi-source streaming mechanism implementation on a Video LAN (VLC) platform”.

[3]. Testing MPEG based IP video QoE/QoS”,http://www. shenick.com, Shenick Network Systems.

[4]. Gerard O’driscoll, “NEXT GENERATION IPTV SERVICES AND TECHNOLOGIES”, 2007.

[5]. Pablo Rodrıguez-Bocca, Hector Cancela, Gerardo Rubino, “Video Quality Assurance in Multi-Source Streaming Techniques”, Universidad de la Republica, Montevideo, Uruguay, Universidad dela Republica, Montevideo, Uruguay, Campus universitaire de Beaulieu, Rennes, France.

[6]. Learn linux in 24 hours http://fileserver.eepisits.edu/ebook/pdf/ , 2000

[7]. Sritusta Sukaridhoto. “Buku Jaringan Komputer”, PENS-ITS, Surabaya, 2007.

[8]. Darwin Streaming Server, http://dss.macosforge.org/

[9]. Rizal Aulia Firmansyah,” Aplikasi Video Switch Berbasis Web pada Eepis i-Studio”, Laporan Proyek Akhir PENS-ITS, Surabaya, 2008

[10]. ITU-T Recommendation G.1010 (2001), *End-user multimedia Qos Categories*.

[11]. “oRTP Reference Manual” http://www.grogy.com/local_doc/share/doc/ortp/ortpapi.html

[12]. “RTP, Real-Time Transport Protokol” http://www.networksorcery.com/enp/protocol/rtp.htm