DESAIN DAN IMPLEMENTASI *LIVE STREAMING* TELEVISI MENGGUNAKAN *ADAPTIVE H264ENCODING*

Firza Ramadhan[1)], Agus Virgono[2)], Ida Wahidah[3)]
[1,2,3)] Fakultas Teknik Elektro dan Komunikasi, ITTelkom Bandung
Jl. Telekomunikasi 1 Dayeuh Kolot Bandung 40254 Tlp (022)-7565933
email : agv@ittelkom.ac.id

Abstrak
Teknologi Informasi yang paling luas penyebarannya adalah Televisi, dengan kemajuan teknologi sarana penyiaran Televisi tidak terbatas lagi ke TV broadcast menggunakan teknologi radio di gelombang khusus seperti saat ini, penyiaran TV telah menyebar ke sarana yang lain termasuk internet. Banyak teknologi yang bisa digunakan di internet, tetapi kandidat yang paling kuat adalah video streaming. Untuk aplikasi real-time atau live seperti kampanye atau siaran pengumuman pemerintah dll, teknologi video streaming yang digunakan adalah teknologi video streaming khusus yang disebut dengan live streaming.
Teknologi Live Streaming hampir sama dengan video streaming, hanya saja data yang digunakan langsung bersumber dari televisi atau kamera yang bersifat real time. Live Streaming memerlukan proses live encoding dan minimum buffering, sedangkan di sisi lain diharapkan delay seminimal mungkin. Masalah selanjutnya adalah keterbatasan bandwidth. Jaringan komputer yang digunakan untuk melewatkan berbagai aplikasi akan digunakan juga sebagai media streaming yang membutuhkan bitrate cukup tinggi. Proses ini akan menyebabkan beban jaringan bertambah sehingga service yang ada tidak dapat berjalan dengan baik (terganggu).
Pada penelitian ini difokuskan pada proses live streaming H264 dengan metode transmisi multicast dengan ditambahkan sebuah program adaptive streaming. Codec H264 dipilih karena performansinya yang cukup baik pada level bitrate yang lebih rendah. Sistem multicast digunakan untuk mengatasi masalah keterbatasan bandwidth yang digunakan dalam streaming. Adaptive streaming digunakan untuk menyesuaikan bitrate dengan kondisi trafik pada jaringan. Didapatkan nilai PSNR 36,58 dB untuk bitrate 500kbps dan 31,42 dB untuk bitrate 200kbps yang masih berada diatas threshold ITU 20dB dengan MOS 3,4 untuk 50 responden, sistem adaptive menyebabkan berkurangnya paket loss dari 1,53% menjadi 0,46%, bandwitdh stream unucast 1698kbps untuk multicast 558kbps.

Keyword: *PSNR, MOS, Live Streaming, Bandwidth.*

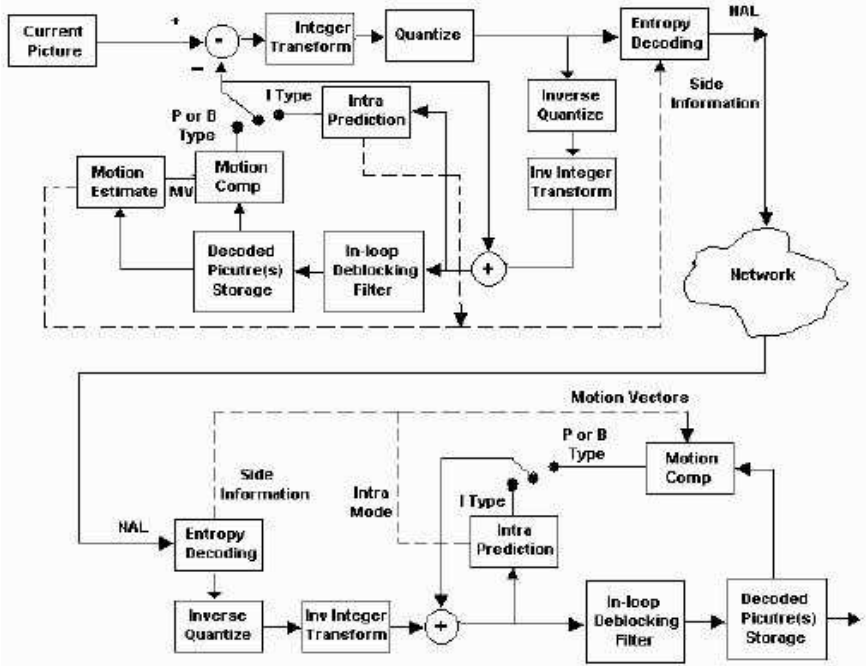
1. PENDAHULUAN

Teknologi video saat ini merupakan hal yang sangat umum dan sering dijumpai. Dunia hiburan, komunikasi, *monitoring*, dan *security* saat ini telah banyak memanfaatkan teknologi video ini. Masalah terbesar pada teknologi video ini adalah banyak menghabiskan sumber daya yang ada. Salah satu contohnya adalah aplikasi video *streaming* yang banyak menghabiskan sumber daya seperti *server*, jaringan dan *client*.
Pada aplikasi *live streaming* masalah yang ada bertambah dengan adanya proses *capturing* dan *live decoding* pada sisi *server*. Selain masalah pada *server*, masalah terbesar yang dihadapi dari teknologi ini adalah keterbatasan *bandwidth* sedangkan proses komunikasi menggunakan *digital video* ini menghabiskan *resource* yang cukup besar. Jaringan komputer yang digunakan untuk melakukan berbagai aplikasi akan digunakan juga sebagai media *streaming* yang membutuhkan *bitrate* cukup tinggi. Proses ini akan menyebabkan beban jaringan bertambah sehingga menyebabkan *service* yang diberikan tidak dapat berjalan dengan baik (terganggu).
Pada penelitian ini akan dilakukan implementasi *adapative* H264 pada aplikasi *live streaming s*ebagai salah satu solusi untuk mengatasi masalah keterbatasan *resource* jaringan. Diharapkan dengan implementasi adaptive H264 ini dapat dibangun sebuah sistem *live streaming* yang handal yang memberikan kualitas video terbaik sesuai dengan kondisi jaringan yang ada.
Penelitian ini bertujuan untuk membangun sebuah aplikasi *LiveStreaming* yang dapat melengkapi servis multimedia di dalam jaringan lokal, dan melakukan pengukuran performa dengan membandingkan sistem *adaptive* dan sistem *non-adaptiv,* membandingkan performansi IP *unicast* dengan IP *multicast* untuk trafik *real-time,* membandingkan performansi *codec* H264 dengan MPEG-4.

2. TINJAUAN PUSTAKA

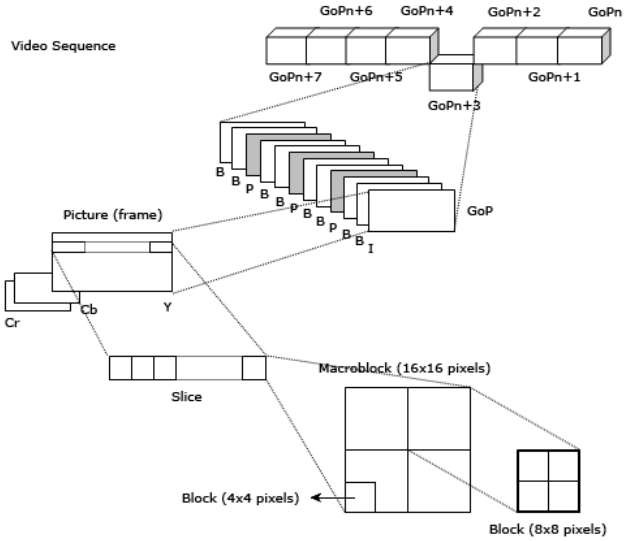
Pada teknologi video streaming digunakan codec sebagai format kompresi standar untuk transmisi, jika media video tidak dikompresi maka bitrate yang dibutuhkan menjadi sangat besar. Sebuah codec yang baik akan menyebabkan keperluan bitrate yang rendah dengan sesedikit mungkin mengurangi kualitas dari media tersebut. H264 (MPEG-4 *Part* 10) atau lebih dikenal dengan *Advanced Video Coding* (AVC) merupakan sebuah *codec* video *digital* yang dikembangkan untuk memenuhi kebutuhan akan video *digital* dengan tingkat kompresi yang

tinggi. H264 dikembangkan oleh ITU-T *Video Coding Experts Group* (VCEG) bersama-sama dengan ISO/IEC *Moving Picture Expert Group* (MPEG) yang dinamakan *Joint Video Team* (JVT).
Tujuan pengembangan H264/AVC adalah untuk membuat suatu standar video *digital* yang dapat menghasilkan kualiatas video yang baik pada bitrate yang lebih kecil dibandingkan dengan standar video *digital* sebelumnya (MPEG-2, H263, maupun MPEG-4 *Part* 2) tanpa harus melakukan perubahan yang kompleks dan dapat diimplementasikan dengan biaya yang murah. Tujuan lain dari pengembangan H264 adalah dapat digunakan dalam berbagai macam aplikasi seperti video *broadcast*, DVB *storage*, RTP/IP *packet networks*, dan ITU-T *multimedia telephony systems*.



Gambar 1. Blok diagram video encoder dan video decoder H.264.

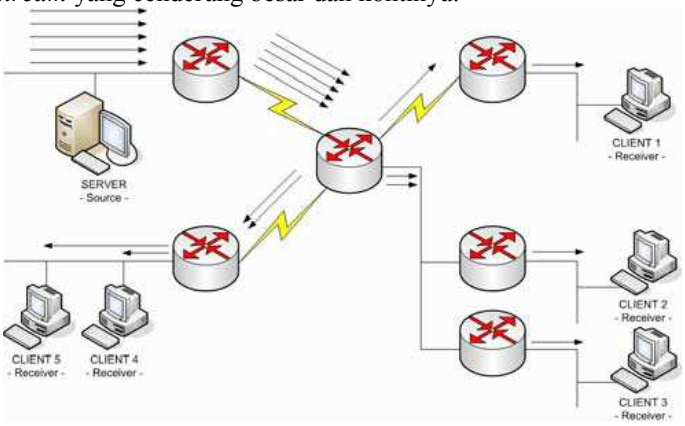
Serupa dengan MPEG-4 (karena dikembangkan berdasarkan MPEG-4) H264 memiliki beberapa bagian yaitu GOP, *slice, makroblock,* dan *block*. Hanya saja terdapat beberapa perbedaan yang merupakan penyempurnaan dari MPEG-4 yang salah satunya adalah ukuran blok yang lebih kecil yaitu 4x4.



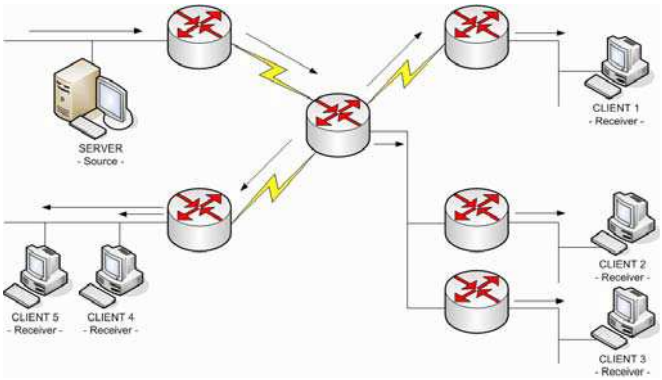
Gambar 2. Struktur Video H.264.

Layanan *adaptive streaming* dapat diimplementasikan dengan menggunakan protokol TCP untuk pengiriman video *streaming* dari *server* ke *receiver*. *Player* pada client akan menampilkan *frame-frame* dengan *rate* yang

tidak melebihi video yang di-*encode* pada server. *Client* memiliki *buffer* yang digunakan untuk menetapkan jumlah *frame* video yang ditampung sebelum ditampilkan oleh *player*. Hal ini bertujuan untuk menghaluskan *delay-jitter* pada interval kedatangan *inter-frame* dari *stream* video.
Layanan *adaptive streaming* melakukan pengawasan QoS yang didukung oleh *server streaming*. *Server streaming* secara dinamis akan melakukan adaptasi untuk *stream* video yang ditransmisikan agar selalu sesuai dengan ketersediaan *bandwidth*. Pada prinsipnya layanan *adaptive* ini akan memonitor *link-rate* setiap saat secara dinamis untuk kemudian menentukan besar *bitrate* video yang akan dikirimkan.
Pada penelitian kali ini sistem *adaptive* tidak menggunakan protokol TCP sebagai *rate control*. Paket video yang digunakan adalah paket RTP (UDP), dan *rate control* dilakukan dengan mengirimkan paket ICMP secara periodik. Informasi yang diterima *server* adalah berupa *reply* dimana terdapat informasi waktu, yang nantinya akan diolah sehingga didapatkan informasi *bandwidth* yang tersedia pada saat itu. Sistem ini dibangun dengan tujuan agar kualitas video tetap terjaga dengan menggunakan paket RTP dan *server* tidak terlalu terbebani untuk melakukan *rate control* secara dinamis setiap saat seperti halnya jika menggunakan paket TCP.
Transmisi *multicast* merupakan teknologi *bandwidth-conserving* yang dapat mengurangi trafik jaringan dengan mengirimkan sebuah *stream* tunggal secara simultan ke beberapa *client* atau *host* yang tergabung pada *multicast group*.
Setiap *terminal* yang menginginkan transmisi tersebut akan menggabungkan diri pada *multicast group* dimana setiap *multicast group* hanya akan mentransmisikan satu *stream* data. *Router* yang telah menggabungkan diri sebagai *group multicast* tidak akan membuang atau *drop* paket yang melaluinya.
Pada prinsipnya, IP *multicast* mengirimkan sumber trafik ke banyak *client* atau penerima tanpa penambahan *stream* baru. Paket *multicast* akan digandakan pada jaringan oleh *router* dengan menggunakan fasilitas *Protocol Independent Multicast* (PIM) atau protocol *multicast* lain. Untuk aplikasi yang membutuhkan *bandwidth* besar seperti *streaming*, *multicast* dapat menjadi satu solusi untuk mengatasi keterbatasan *bandwidth* dan ketersediaan jaringan terhadap paket *stream* yang cenderung besar dan kontinyu.



Gambar 3. Transmisi Unicast



Gambar 4. Transmisi Multicast

3. METODA PENELITIAN

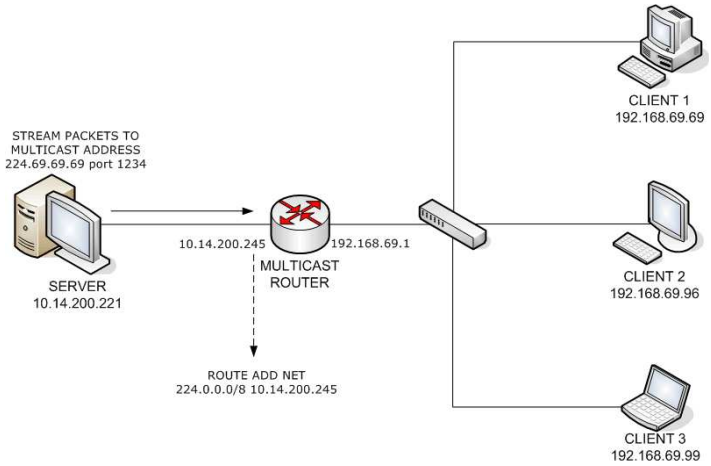
Untuk menganalisa performansi sistem *live-streaming* yang dibuat, maka perlu dilakukan suatu proses desain yang kemudian diimplementasikan secara real di laboratorium. Langkah-langkah yang dilakukan untuk menimplementasikan sistem ini, yaitu :

1. Perancangan profil dan *bitrate* video H264 yang akan digunakan.
2. Konfigurasi alamat IPv4 *multicast* pada sistem
3. Konfigurasi server untuk mensupport *multicast data stream*

4. Konfigurasi *multicast-enable-router*
5. Pembuatan program *adaptive* pada aplikasi H264 *video streaming*

3.1 SKEMA KERJA SISTEM

Server 10.14.200.221 merupakan *server* video *streaming* yang terletak pada *subnet* 1. Server mengirimkan data video secara *multicast* dengan menggunakan alamat multicast 224.69.69.69. Kemudian data tersebut akan didengar oleh *multicast-enabled-router* dan akan diteruskan apabila terdapat *client* pada subnet 2 yang ingin bergabung dengan alamat *multicast*. *Client* yang ingin bergabung dengan alamat *group multicast* akan mengirimkan IGMP pada *router*.



Gambar 5. Skema jaringan yang digunakan dalam pengukuran

Pada penelitian ini, *server* berfungsi sebagai penerima gambar (sinyal) video yang berasal dar TV untuk kemudian melakukan pengolahan sinyal menjadi video *digital* dengan *codec* H264. Setelah format video berubah, dilakukan suatu kompresi berdasarkan profil yang telah ditentukan sebelumnya, lalu digabung dengan *file audio* (proses *muxing*). Setelah digabung, file tersebut akan dikirim jika ada user melakukan *request*.
Perangkat lunak yang digunakan pada *server* adalah *Video Lan Client* yang difungsikan sebagai *streaming server* sekaligus proses *muxing* video dengan audio. Sedangkan proses *encoding* video H264 dan audio mp3 dilakukan oleh *software FFmpeg*, dimana untuk melakukan *encoding* video H264 *Ffmpeg* menggunakan *library* dari X264.

3.2 KONFIGURASI ROUTER

Router yang digunakan memakai komputer (*PC Router*). *Router* yang akan diimplementasikan menggunakan OS FreeBSD 6.1 dengan 2 buah e*thernet card* yang nantinya akan melayani 2 *subnet*. Konfigurasi secara umum untuk *static routing* dapat dilakukan dengan menambahkan *tag* sebagai berikut pada file /etc/rc/conf :

```
root@streaming# vi /etc/rc.conf
---------------------- cut here ----------------------
gateway_enable="YES"
ifconfig_sis0="inet 10.14.200.245 netmask 255.255.255.0"
ifconfig_xl0="inet 192.168.69.1 netmask 255.255.255.0"
---------------------- cut here ----------------------
```

Konfigurasi pada bagian ini akan memberikan fungsionalitas *multicast routing* pada *PC router* yang telah disiapkan. Sistem *multicast* akan menggunakan *mrouted* yang dapat diaktifkan secara manual dengan cara mengaktifkan modul *mroute* pada *kernel* FreeBSD.
Untuk mengaktifkan modul *multicast routing*, diperlukan sebuah opsi tambahan yang harus ditambahkan pada *kernel*.
Hal ini dapat dilakukan dengan cara :

```
root@streaming# vi /usr/src/sys/i386/conf/STREAMING
```

tambahkan *script* ini pada baris paling akhir :

```
# multicast-enable-router
options MROUTING
```

compile ulang kernel, *reboot*, lalu edit file /etc/mrouted.conf :

```
phyint sis0 rate_limit 0
phyint xl0 rate_limit 0
```

tambahkan alamat *multicast* pada *routing table* :

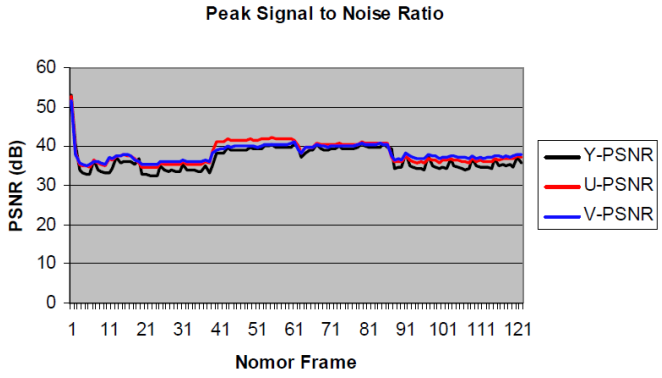
```
root@streaming# route add –net 224.0.0.0/4 10.14.200.245
```

bagian di atas menunjukkan bahwa alamat multicast akan dilewatkan oleh *interface* sis0 yang memiliki *IP address* 10.14.200.245, jadi setiap paket multicast yang dikirim ke alamat multicast tertentu dapat di-*forward* oleh *interface* tersebut.
jalankan *mrouted* :

```
root@streaming# mrouted –c /etc/mrouted.conf –d
```

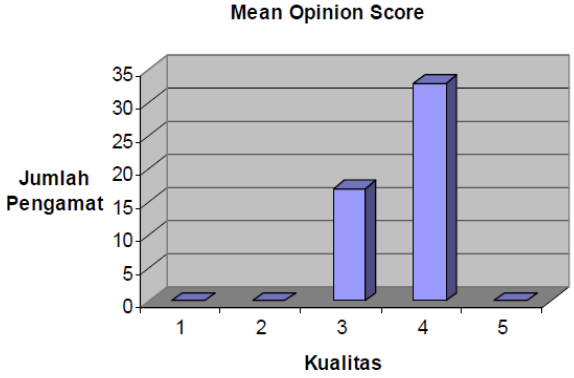
4 HASIL DAN PEMBAHASAN

Untuk mengevaluasi secara tepat dilakukan pengamatan video dengan *bitrate* sebesar 500 kbps. Parameter yang akan diukur meliputi PSNR, MOS, *packet loss*, *delay* paket, *bandwidth*, dan *jitter*.
Sedangkan untuk pengukuran program *adaptive* yang dibuat sebagai solusi dari perubahan *bandwidth* secara signifikan akan dilakukan secara terpisah. Hal yang akan diamati adalah delay pergantian *bitrate*, PSNR, dan MOS.
Ada dua buah skenario yang digunakan dalam pengamatan dan pengukuran kali ini. Skenario pertama adalah sebagai berikut pada sistem ini, dikirimkan *stream* video H264 dengan *bitrate* 500 kbps, *frame rate* 25 fps, resolusi 320x240, *bandwidth* jaringan tanpa batasan, tanpa trafik pengganggu, *server - router - client* disinkronisasi menggunakan NTP.
Nilai PSNR dinyatakan dalam satuan dB dimana sebuah video dikatakan memiliki kualitas baik jika nilai PSNR video tersebut berada di atas 20dB. Parameter PSNR memiliki tiga komponen utama yaitu Y-PSNR yang digunakan untuk menyatakan level *luminance* dari sebuah gambar, U-PSNR dan V-PSNR untuk tingkat komponen warna pada sebuah gambar.
Proses perhitungan PSNR pada penelitian kali ini adalah dengan cara membandingkan dua buah video, dimana video asli merupakan hasil *grab* gambar dari *TV-Card* dan video kedua merupakan video yang ditampilkan di sisi *client*.



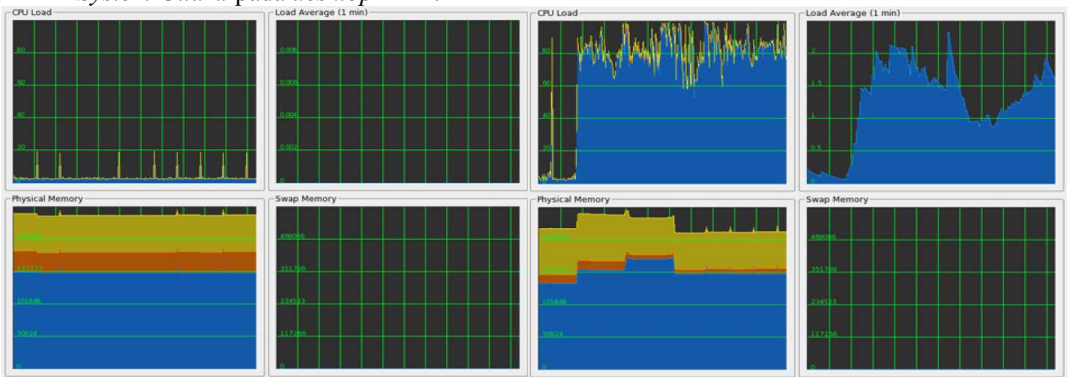
Gambar 6. *Peak Signal to Noise Ratio* (PSNR)

Dari hasil pengukuran terlihat bahwa file video H264 (*baseline profile*) dengan *bitrate* 500kbps dan *framerate* 25 fps memiliki nilai PSNR rata-rata sebesar 36,58 dB. Nilai ini masih berada di atas *threshold* 20dB.
Pengukuran MOS dilakukan dengan mensurvey 50 orang akan kualitas relatif dari video dan didapatkan hasilnya seperti pada gambar 7 dibawah ini dengan MOS = 3,68



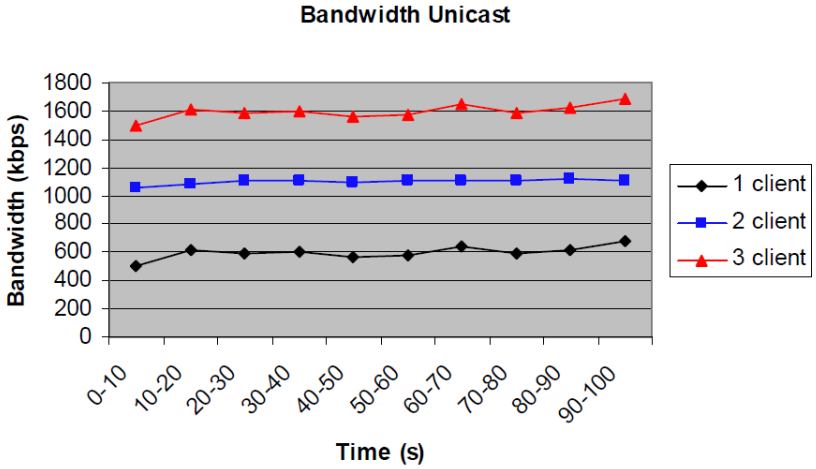
Gambar 7. Mean Opinion Score

Pengamatan beban memori dan CPU di server dilakukan untuk mengukur seberapa besar proses *encoding* H264 yang dilakukan oleh *server* sehingga dapat menghasilkan file video yang siap untuk ditransmisikan menuju *client*. Pengkodean video H264 yang dilakukan pada penelitian kali ini menggunakan *software* X264 yang membutuhkan kemampuan MMX dan SSE2 pada CPU.
Pengamatan dilakukan dengan menggunakan perintah "*top*" pada *konsole* atau juga dapat diamati melalui aplikasi *KDE System Guard* pada *desktop KDE*.



Gambar 8. Beban CPU dan memory sebelum dan setelah proses streaming

Gambar 8. sebelah kiri menunjukkan beban *CPU* dan *load memory* pada saat sebelum proses *streaming* berlangsung. Pada gambar tersebut terlihat bahwa penggunaan *CPU* dan *memory* masih belum terlalu besar karena belum ada proses *encoding* yang berjalan., sedangkan pada Gambar 8 kanant menunjukkan beban *CPU* dan *memory* pada saat proses *streaming* berlangsung. Dapat dilihat terjadi perubahan secara signifikan penggunaan *resource CPU* dan *memory* pada saat proses *streaming* berlangsung. Hal ini membuktikan bahwa proses *live streaming* H264 membutuhkan *resource* yang sangat besar untuk digunakan pada proses *live encoding* video H264 dan penggunaan *buffer memory* sebagai media penyimpanan sementara pada *server* sebelum ditransmisikan.
Pengukuran *bandwidth* dilakukan pada sisi *server*, yang bertujuan untuk melihat trafik *streaming* yang dikirimkan oleh *server* pada suatu sesi *streaming*. Pada sistem *unicast*, penambahan jumlah *client* berarti menambah jumlah *stream* atau aliran data yang dikirim dari *server* ke *client*. Hal ini mengakibatkan bertambahnya *bandwidth* yang digunakan.

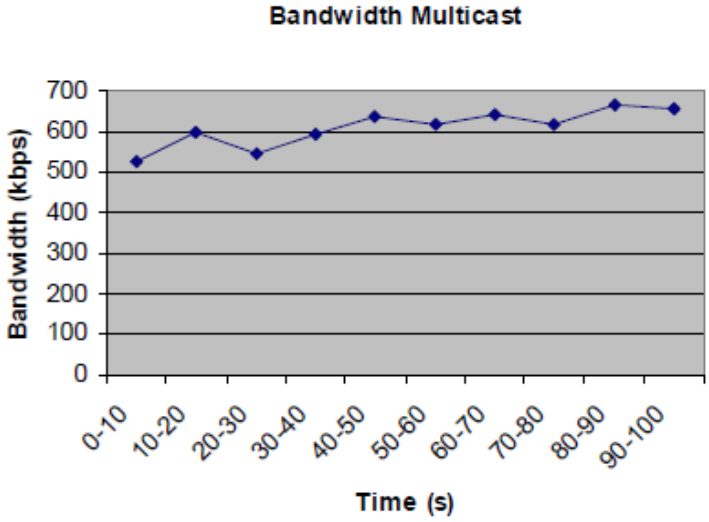


Gambar 9. Penggunaan bandwidth untuk sistem non-adaptive streaming unicast

Gambar 9. ini menunjukkan bahwa penambahan *client* akan menyebabkan peningkatan *bandwidth* secara simultan. Hal ini disebabkan oleh koneksi *unicast* yang bersifat *point-to-point* antara *server* dan *client*, sehingga satu *host* mendapatkan satu *stream* data. Pada transmisi *unicast*, besarnya *bandwidth* yang digunakan adalah besarnya *stream* tunggal dikalikan jumlah *client* yang melakukan *request* data *stream*.
Penggunaan *bandwidth* rata-rata :

1 *client* : 598,32 kbps
2 *client* : 1152,16 kbps (1,152 Mbps)
3 *client* : 1698,21 kbps (1,698 Mbps)

Pada transmisi *multicast*, penambahan *client* yang melakukan *request* data *streaming* tidak berpengaruh pada penggunaan *bandwidth*. Hal ini dikarenakan *server* hanya mengirim satu buah *stream* data ke satu alamat *multicast* tertentu, yang nantinya proses penyalinan data dilakukan oleh *multicast-enable-router* untuk selanjutnya dibagikan kepada *host/client* yang bergabung ke *group multicast* tersebut.



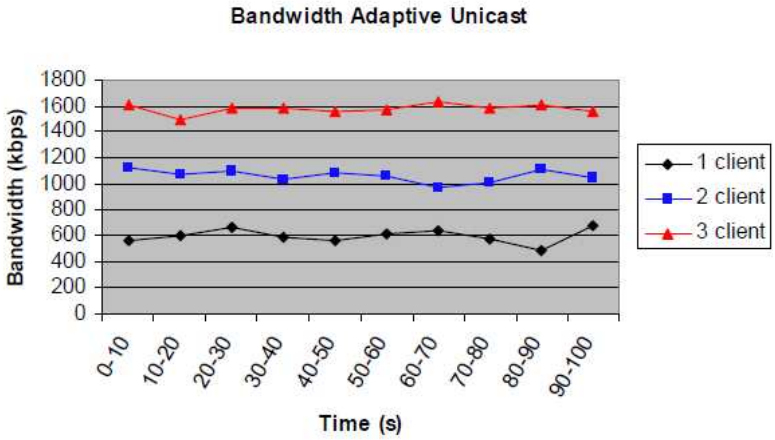
Gambar 10. Penggunaan bandwidth untuk sistem non-adaptive streaming multicast

Bandwidth rata-rata yang digunakan untuk aplikasi *streaming* dengan transmisi *multicast* dengan 3 *client* adalah 610,24 kbps

Pada bagian ini akan dilakukan analisa pengaruh penggunaan program *adaptive* yang bertujuan untuk melakukan perubahan *bitrate* secara otomatis apabila terjadi penurunan *bandwidth* akibat aktivitas jaringan yang tidak dapat diprediksikan.

Pembuatan program ini bertujuan untuk meminimalisir terjadinya *packet loss* yang besar apabila terjadi penurunan *bandwidth* secara signifikan. Dalam aplikasi streaming, *packet loss* akan sangat berpengaruh pada kualitas video.

Semakin besar *packet loss* berarti semakin banyak *dropped frame* dari video yang menyebabkan kualitas video pada *client* akan buruk. Hal yang umum terjadi adalah video akan mengalami *jerky motion* apabila *bitrate* video tersebut tidak cukup dilewatkan pada *bandwidth* yang tersedia.



Gambar 11. Penggunaan bandwidth untuk sistem adaptive streaming unicast

Penggunaan *bandwidth* rata-rata :

1 *client* : 595,76 kbps
2 *client* : 1062,93 kbps (1,062 Mbps)
3 *client* : 1577,18 kbps (1,577 Mbps)

Bandwidth rata-rata yang digunakan untuk aplikasi *streaming* dengan transmisi *multicast* dengan 3 *client* adalah 558,04 kbps.

Pada pengujian program *adaptive* kali ini, digunakan *skenario* yang berbeda dengan kondisi sebelumnya. Percobaan ini dilakukan dengan kondisi sebagai berikut :

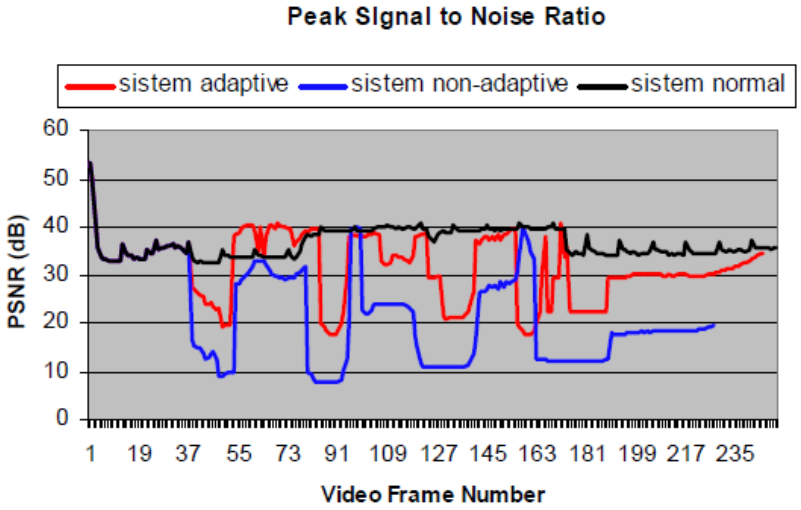
1. Melakukan pembatasan *bandwidth* pada *server-router*
2. Memberikan *background traffic* berupa *stream* video lain
3. Hanya dilakukan pada sistem transmisi *multicast*

Tujuan dilakukan pembatasan *bandwidth* adalah sebagai pemodelan kondisi ekstrim dari sebuah jaringan. Sedangkan penambahan *background traffic* berupa stream video lainnya adalah untuk memberikan trafik pengganggu sehingga jaringan tersebut dapat dikatakan cukup padat. *Bitrate* video *streaming* yang ditambahkan adalah sebesar 500 kbps sedangkan *bitrate live streaming* yang diuji coba kali ini adalah 500 kbps untuk kondisi trafik kosong dan 200 kbps apabila terjadi penurunan *bandwidth* pada jaringan tersebut.

Pada saat awal proses *streaming* berlangsung, *server* mengirimkan video dengan *bitrate* 500 kbps. Pada pertengahan proses *streaming* terjadi penurunan ketersediaan *bandwidth* pada jaringan akibat dari adanya *stream* video lain, sehingga secara otomatis *server* akan menurunkan *bitrate* video yang dikirimkan menuju ke *client* menjadi 200 kbps.

Proses perubahan *bitrate* yang dilakukan oleh *server* ini membutuhkan waktu kurang lebih selama 1 detik. Selama waktu ini, video yang ditampilkan pada *client* akan terputus dikarenakan ada proses perubahan *bitrate* oleh *server*. Akan tetapi terputusnya video ini terbukti tidak terlalu mengganggu *client* apabila dibandingkan dengan penggunaan *bitrate* yang statis. Apabila *bitrate* yang dikirim oleh *server* melebihi besarnya ketersediaan *bandwidth* jaringan pada saat itu maka dapat dipastikan akan banyak sekali *packet loss*. Hal ini menyebabkan video akan rusak (patah-patah) selama *bandwidth* pada jaringan masih belum cukup untuk melewatkan *bitrate* video tersebut.

Pada sistem *non-adaptive* terlihat lebih banyak terjadi *dropped frame* dibandingkan dengan sistem *adaptive*. *Dropped frame* ini disebabkan karena banyaknya paket yang hilang pada jaringan (*packet loss*). Sehingga informasi yang dikirimkan tidak semuanya dapat diterima oleh *client*.



Gambar 12. Perbandingan nilai PSNR sistem adaptive dan non-adaptive

Percobaan ini bertujuan untuk menganalisa besarnya *packet loss* yang terjadi pada sistem *non-adaptive* dan sistem *adaptive*. Pengamatan *packet loss* dilakukan pada saat terdapat background traffic. Pada sistem non-adaptive terlihat bahwa pada saat kondisi trafik penuh dan tetap dipaksakan dengan melakukan stream dengan besar *bitrate* 500 kbps terjadi cukup banyak paket yang hilang di jaringan. Hal ini berakibat pada hilangnya sebagian informasi video sehingga video yang ditampilkan pada *client* terganggu. Sedangkan pada sistem *adaptive*, apabila terjadi penurunan bandwidth jaringan maka *server* otomatis akan mengganti *bitrate codec* dari 500 kbps menjadi 200 kbps. Hal ini memang menyebabkan video terputus selama kurang lebih satu detik seperti telah dipaparkan pada sub bab sebelumnya. Akan tetapi pada sistem *adaptive* paket yang hilang jauh lebih sedikit dibandingkan dengan sistem *non-adaptive* yang berarti gambar yang ditampilkan pada *client* tidak akan terganggu seperti pada sistem *non-adaptive*.

Packet loss rata-rata selama satu menit (60 s) pada sistem *non-adaptive* adalah sebesar 1,531046 % dari total seluruh paket yang dikirim. Sedangkan *packet loss* rata-rata selama satu menit (60 s) pada sistem *adaptive* hanya sebesar 0,467387 %.

Pada sistem *adaptive* ada beberapa paket yang hilang, akan tetapi jumlahnya jauh lebih kecil dibandingkan dengan sistem *non-adaptive*. Hilangnya paket pada sistem *adaptive* ini lebih disebabkan karena kondisi jaringan yang *unpredictable* sehingga sangat mungkin terjadi *collision* dengan paket-paket pengganggu.

5 KESIMPULAN

Berdasarkan hasil pengukuran, pengamatan, dan analisis dari implementasi sistem *adaptive* H264 *live streaming* pada IPv4 *multicast*, beberapa kesimpulan yang dapat diambil, yaitu :

- Nilai PSNR *codec* H264 yang terukur untuk *bitrate* 500 kbps adalah sebesar 36,58 dB dan bernilai 31,42 dB pada *bitrate* 200 kbps. Nilai ini masih berada di atas *threshold* 20 dB (standar ITU). Sebanding dengan PSNR, nilai MOS yang didapatkan dari 50 orang responden adalah 3,6 dari skala 1-5.
- Untuk 3 *client*, rata-rata *bandwidth* stream yang dibutuhkan pada sistem transmisi *unicast* adalah sebesar 1698,21 kbps sedangkan sistem *multicast* hanya membutuhkan 558,04 kbps. Akan tetapi, *delay* transmisi rata-rata yang dihasilkan pada sistem *multicast* lebih besar yaitu 20,13 ms, dan sistem *unicast* hanya menghasilkan *delay* 14,52 ms. Nilai *jitter unicast* dan *multicast* memberikan hasil yang hampir sama yaitu 8,7 ms untuk sistem *multicast* dan 9,7 ms untuk sistem *unicast*.
- Sistem *adaptive streaming* terbukti dapat menangani masalah pada jaringan. Nilai *packet loss* pada sistem *adaptive* adalah 0,46 % sedangkan pada sitem *non-adaptive* nilai *packet loss* 1,53 %. Besarnya *packet loss* mempengaruhi nilai PSNR, nilai PSNR untuk sistem *adaptive* adalah 31,42 dB yang jauh lebih baik dibandingkan dengan sistem *non-adaptive* yaitu 19,59 dB. Begitu juga dengan hasil perhitungan MOS, sistem *adaptive* mendapatkan nilai 3-4 dan sistem *non-adaptive* mendapatkan nilai 2,7 dari skala 1-5.

6 DAFTAR PUSTAKA

[1] Azikin, Askari. 2005. *Analisis Kinerja Adaptive Streaming MPEG-4Encoding Pada Real-Time Monitoring System Via IPv6 Multicast*. Tugas Akhir Jurusan Teknik Elektro STT Telkom Bandung.
[2] Yudha, Aditya. 2004. *Implementasi dan Analisa IP Multicast Untuk Trafik Real TIme*. Tugas Akhir Jurusan Teknik Elektro STT Telkom Bandung.
[3] Richardson, Iain E G. 2003. *H264 and MPEG-4 Video Compression :Video Coding for Next-generation Multimedia*, Aberdeen: The Robert Gordon University.
[4] Ghanbari, M. 1999. *Video Coding: an introduction to standard codecs*, London: The Institution of Electrical Engineers.
[5] Matrawy, Ashraf. 2002. *Multicasting of Adaptively-encoded MPEG4 over QoS-aware IP Networks*, Canada: Broadband Networks Laboratory, Department of Systems and Computer Engineering Carleton University.
[6] Pawse, Guruprasad V. 2003. *H.264 / MPEG-4 Part 10 AVC Baseline Decoder*, India: Global Edge Software Ltd.
[7] Vatolin , Dmitriy. 2005. *MPEG-4 AVC/H.264 Video Codec Comparison*, CS MSU Graphics&Media Lab Video Group.
[8] RFC3550. *RTP: A Transport Protocol for Real-Time Applications.*

7 CURRICULUM VITAE

Firza Ramadhan, menyelesaikan studi S1 bidang Teknik Telekomunikasi ITTelkom Bandung pada 2007.
Agus Virgono, menyelesaikan studi S1 bidang Teknik Elektro Sub Teknik Komputer ITB pada 1991, menyelesaikan S2 bidang Sistem Informasi Telekomunikasi ITB pada 2001.
Ida Wahidah, menyelesaikan studi S1 bidang Teknik Elektro Sub Teknik Telekomunikasi ITB pada 1998, menyelesaikan S2 bidang Sistem Informasi Telekomunikasi ITB pada 2005.