# PERFORMANSI VIDEO *STREAMING* PADA JARINGAN *HIGH SPEED DOWNLINK PACKET ACCESS* (HSDPA) MENGGUNAKAN IPv6

## SKRIPSI

**KONSENTRASI TEKNIK TELEKOMUNIKASI**

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik


Disusun Oleh :

**SIRTUFILLAILA**

**NIM: 0610630103 - 63**

**KEMENTERIAN PENDIDIKAN NASIONAL**

**UNIVERSITAS BRAWIJAYA**

**FAKULTAS TEKNIK**

**MALANG**

**2010**

# LEMBAR PERSETUJUAN

# PERFORMANSI VIDEO *STREAMING* PADA JARINGAN *HIGH SPEED DOWNLINK PACKET ACCESS* (HSDPA) MENGGUNAKAN IPv6

**SKRIPSI**

**KONSENTRASI TEKNIK TELEKOMUNIKASI**

Diajukan untuk memenuhi persyaratan
memperoleh gelar sarjana teknik


Disusun Oleh :

**SIRTUFILLAILA**

**NIM: 0610630103 – 63**

**Telah diperiksa dan disetujui oleh**

**Dosen Pembimbing**

| | |
|---|---|
| **Ir. Wahyu Adi Priyono, MSc.** | **Asri Wulandari, ST., MT.** |
| **NIP. 19600518 198802 1 001** | **NIP. 19750301 199903 2 001** |

# LEMBAR PENGESAHAN

# PERFORMANSI VIDEO *STREAMING* PADA JARINGAN *HIGH SPEED DOWNLINK PACKET ACCESS* (HSDPA) MENGGUNAKAN IPv6


Disusun Oleh :

**SIRTUFILLAILA**

**NIM: 0610630103 – 63**


Skripsi ini telah diuji dan dinyatakan lulus

pada tanggal **21 Desember 2010**

**Majelis Penguji :**

| | |
|---|---|
| **Ir. Endah Budi P., MT.** | **Dr. Ir. Sholeh Hadi P., MS.** |
| **NIP 19621116 198903 2 002** | **NIP. 19580728 198701 1 001** |

**Ali Mutofa, ST., MT.**
**NIP. 19710601 200003 1 001**

**Mengetahui :**
**Ketua Jurusan Teknik Elektro**

**Rudy Yuwono, ST., MSc.**
**NIP. 19710615 199802 1 003**

# PENGANTAR

Alhamdulillah, segenap puji dan syukur penulis panjatkan kepada Allah SWT yang telah melimpahkan rahmat, hidayah, dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi dengan judul "Performansi Video *Streaming* pada Jaringan *High Speed Downlink Packet Access* (HSDPA) yang Menggunakan IPv6" yang diajukan untuk memenuhi sebagian persyaratan memperoleh gelar Sarjana Teknik. Tidak lupa pula shalawat serta salam selalu penulis sampaikan kepada junjungan besar Nabi Muhammad SAW yang telah membawa kita menuju ke jalan yang terang.

Penulis ingin mengucapkan terima kasih yang sebesar-besarnya kepada berbagai pihak yang telah membantu dan mendukung dalam penyelesaian skripsi ini, yaitu :

1. Keluarga tercinta, Ibunda Sri Andayani, kakakku Yanurisma Sugianto, serta adikku Robby Firdaus dan Feby Puspita yang selalu memberikan doa, kasih sayang, semangat, dukungan dan kepercayaan yang tiada akhir hingga hari ini.
2. Bapak Ir. Wahyu Adi Priyono., MSc. dan Ibu Asri Wulandari, ST., MT. selaku dosen pembimbing skripsi yang banyak memberikan saran, konsultasi, kesabaran, dan waktu.
3. Bapak Rudy Yuwono, ST., MSc. selaku Ketua Jurusan Teknik Elektro dan Bapak M. Aziz Muslim, ST. MT., Ph.D, selaku sekretaris Jurusan Teknik Elektro.
4. Bapak dan Ibu dosen serta segenap staf dan karyawan Jurusan Teknik Elektro.
5. Sahabat-sahabat terbaikku tujuh kurcaci Sekar, Intan, Riris, Asih, Yulita, Widya, Nurul, Teteceh Eka, Ravi, Rayi, Widya, Rheny serta keluarga besar Angkatan 2006 (Ge-Force) terima kasih atas persahabatan, semangat, dan untuk segalanya.
6. Rekan-rekan asisten Laboratorium Transmisi dan Gelombang Mikro Kartika, Rizcky, dan Rendy atas nasehat, saran, serta kerjasamanya selama ini.
7. Dan untuk semua pihak yang tidak dapat penulis sebutkan satu-persatu.

Dalam penulisan skripsi ini, penulis menyadari adanya kekurangan dan ketidak sempurnaan. Oleh karena itu, penulis sangat mengharapkan kritik dan saran yang membangun untuk kelengkapan dan kesempurnaan skripsi ini. Penulis berharap semoga skripsi ini dapat bermanfaat khususnya bagi rekan-rekan mahasiswa.

Malang, Desember 2010

Penulis

# DAFTAR ISI

## DAFTAR GAMBAR

halaman

# DAFTAR TABEL

## DAFTAR LAMPIRAN

halaman

# ABSTRAK

**SIRTUFILLAILA**, Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya, Desember 2010, Performansi Video *Streaming* pada Jaringan *High Speed Downlink Packet Access* (HSDPA) yang Menggunakan IPv6, Dosen Pembimbing : Ir. Wahyu Adi Priyono., MSc. dan Asri Wulandari, ST., MT.

Video *streaming* adalah sebuah teknologi untuk memainkan file video secara langsung ataupun dengan pre-recorder dari sebuah mesin server (web server). Salah satu jaringan yang mendukung layanan video *streaming* adalah jaringan HSDPA. HSDPA (*High Speed Downlink Packet Access*) atau yang biasa dikenal dengan teknologi 3.5 G merupakan suatu teknologi telekomunikasi bergerak yang dikeluarkan oleh 3GPP *Release* 5 dan merupakan pengembangan dari WCDMA.

Pada skripsi ini, jenis CODEC yang digunakan adalah AMR WB+ untuk audio dan H.264/AVC untuk video. Sedangkan jaringan HSDPA yang dipergunakan adalah jaringan HSDPA yang merupakan *Release* 5 standard 3GPP. Terminal UE (*User Equipment*) yang digunakan adalah terminal kategori 5, dimana jenis modulasi yang digunakan adalah 16-QAM, maksimum TBS (*Transport Block Size*) sebesar 7298 bit, dan maksimum *data rate* dari UE adalah 3,6 Mbps.

Penulisan skripsi ini bertujuan untuk mengetahui bagaimana performansi video *streaming* pada jaringan HSDPA (*High Speed Downlink Packet Access*) menggunakan IPv6 yang meliputi konsumsi *bandwith*, *delay end-to-end*, probabilitas *packet loss*, dan *throughput.*

Hasil perhitungan dan analisis membuktikan bahwa penggunaan bit *rate* CODEC mempengaruhi performansi video *streaming*. Pada bit *rate* CODEC video sebesar 384 kbps dan bit *rate* CODEC audio sebesar 48 kbps diperoleh *bandwidth* aplikasi video *streaming* terbesar, yaitu 566,2879 kbps. Nilai *delay end-to-end* aplikasi video *streaming* pada jaringan HSDPA yang menggunakan IPv6 masih memenuhi standard ITU.T G.1010 untuk layanan video *streaming*, yaitu <10 s. Nilai probabilitas *packet loss* yang didapatkan juga masih dapat diterima karena tidak melebihi 1% sesuai standard ITU.T G.1010. Besar *throughput* terbesar adalah 3598,1258 kbps yang terjadi saat menggunakan *bit rate* CODEC video 256 kbps dan bit rate CODEC audio sebesar 48 kbps dengan faktor utilisasi sebesar 0,1.

**Kata Kunci**: Video *Streaming*, HSDPA, IPv6, Bit R*ate* CODEC, *Bandwidth*, *Delay End-to-end*, *Packet Loss*, *Throughput*

# BAB I
# PENDAHULUAN

## 1.1 LATAR BELAKANG

Teknologi telekomunikasi mengalami perkembangan yang pesat seiring dengan meningkatnya kebutuhan masyarakat akan sarana telekomunikasi secara *real-time* dan praktis. Selain itu, mobilitas masyarakat yang tinggi serta kebutuhan akses data yang tinggi juga mendasari kemajuan teknologi telekomunikasi saat ini.

Video *streaming* merupakan salah satu teknologi telekomunikasi yang bersifat *real time* serta dapat menyalurkan informasi berupa audio maupun video dengan menggunakan jaringan *Internet Protocol* (IP). Dengan teknologi video streaming ini, *user* tidak perlu menunggu hingga *file* selesai di-*download* secara keseluruhan untuk memainkannya. Sebaliknya, *user* dapat memainkan media dengan menunggu beberapa detik saja. Akan tetapi, ada beberapa permasalahan yang mempengaruhi performansi dari video *streaming* diantaranya adalah video berhenti berjalan atau bergerak lambat saat kita sedang menonton video sehingga kita harus menunggu beberapa waktu hingga video kembali berjalan. Selain itu, kualitas video yang dihasilkan juga seringkali buruk dimana video *streaming* yang kita amati kurang jernih dan cenderung tidak jelas.

Salah satu hal yang sangat mempengaruhi dalam performansi video *streaming* tersebut adalah *bandwidth*. Bandwidth adalah jumlah data yang dapat melalui suatu jaringan, atau bagian dari jaringan, untuk setiap waktu tertentu [Behrouz A. Forouzan, 2000: 65]. Bandwidth pada jaringan bersifat terbagi, terbatas, dan berubah terhadap waktu. Semakin besar *bandwidth* jaringan, maka kualitas video *streaming* yang dihasilkan juga akan semakin baik [Michael Gough, 2006: 16]. Selain itu, agar layanan aplikasi *video streaming* ini dapat berjalan dengan baik, lebih efisien, dan cepat dalam penyampaian informasinya maka perlu diterapkan pada suatu teknologi jaringan yang mempunyai kecepatan akses data yang tinggi untuk menghasilkan delay yang seminimal mungkin. Dengan *delay* yang seminimal mungkin diharapkan *user* dapat menikmati video *streaming* tanpa harus menunggu lama.

Kebutuhan aplikasi video *streaming* terhadap *bandwidth* dan kecepatan akses data yang tinggi dapat dipenuhi oleh jaringan HSDPA (*High Speed Downlink Packet Access*) atau yang biasa dikenal dengan teknologi 3.5G. HSDPA menggunakan *bandwidth* sebesar 5 MHz dimana sesuai dengan kebutuhan aplikasi video *streaming*

yang mengkonsumsi *bandwidth* cukup besar. HSDPA merupakan standar HSPA dengan kemampuan dari sisi kecepatan transfer *downlink*-nya dan dapat mencapai kecepatan *downlink* hingga 14.4 Mbps dengan maksimum *uplink* 384 kbps. Di lingkungan perumahan, *user* dapat men-*download* data menggunakan jaringan HSDPA dengan kecepatan 3.7 Mbps. Sedangkan seseorang yang sedang berkendara dengan kecepatan 100 km/jam dapat mengakses internet berkecepatan 1.2 Mbps, sementara user di lingkungan yang padat masih dapat menikmati video *streaming* meskipun hanya memperoleh 300 kbps. Pada HSDPA diperkenalkan beberapa fitur baru yang menjadi mekanisme kunci dalam mencapai tujuan HSDPA yaitu meningkatkan *user throughput* maksimum untuk pengiriman paket data dari sisi *downlink* dan mengurangi *delay* transmisi paket (*round trip delay*). Beberapa fitur tersebut antara lain adalah penggunaan *Adaptive Modulation and Coding* (AMC), *Hybrid Automatic Repeat reQuest* (HARQ), dan *fast scheduling*. Pengunaan teknoligi 3,5G ini diharapkan dapat mengatasi konsumsi *bandwidth* yang besar serta *delay* pengiriman paket data pada aplikasi video *streaming*.

Aplikasi video *streaming* melewatkan data yang akan dikirimkan melalui jaringan internet menggunakan protokol UDP/IP. Struktur protokol UDP/IP terdiri dari empat lapis protokol, yaitu *application layer*, *transport layer*, *network layer*, dan *physical layer*. Penggunaan IPv6 pada jaringan HSDPA terjadi pada *network layer* dimana protokol pada layer ini bertanggung jawab dalam proses pengiriman paket data ke alamat yang tepat. IPv6 yang disebut juga sebagai *Next Generation Internet Protocol* (IPng) ini direkomendasikan oleh Direktur Area IPng dari IETF (*Internet Engineering Task Force*). Salah satu keunggulan IPv6 adalah meningkatkan ukuran dan jumlah alamat yang mampu didukung oleh IPv4 dari 32 bit menjadi 128 bit. Hal ini digunakan untuk mendukung hirarki pengalamatan dan jumlah pengalamatan *node* yang lebih banyak, serta konfigurasi alamat-alamat secara otomatis yang lebih sederhana. Selain itu, adanya penyederhanaan format *header* pada IPv6. Beberapa kolom pada *header* IPv4 telah dihilangkan atau dapat dibuat sebagai *header* pilihan. Hal ini dapat digunakan untuk mengurangi biaya pemrosesan hal-hal yang umum pada penanganan paket IPv6 dan membatasi biaya *bandwidth* pada *header* IPv6. Dengan demikian, pemerosesan *header* pada paket IPv6 dapat dilakukan secara efisien. Oleh karena itu, penggunaan IPv6 ini diharapkan dapat meningkatkan performansi video *streaming* pada jaringan HSDPA.

Pada skripsi ini akan dibahas bagaimana perfomansi video *streaming* pada jaringan HSDPA yang berbasis IPv6. Pembahasan yang dilakukan meliputi analisa besarnya konsumsi *bandwidth*, probabilitas *packet loss, delay end-to-end,* dan *throughput*. Hal yang ingin dicapai dalam penulisan skripsi ini adalah dapat meningkatkan performansi video *streaming* dengan menghasilkan nilai *delay end-to-end* dan probabilitas packet loss yang seminimal mungkin serta besar *throughput* yang relatif besar.

## 1.2 RUMUSAN MASALAH

Mengacu pada permasalahan yang telah diuraikan dalam latar belakang maka rumusan masalah ditekankan pada:

1. Bagaimana penggunaan IPv6 pada jaringan HSDPA (*High Speed Downlink Packet Access*)?
2. Bagaimana performansi *video streaming* pada jaringan HSDPA (*High Speed Downlink Packet Access*) yang berbasis IPv6, meliputi konsumsi *bandwidth*, probabilitas *packet loss, delay end-to-end,* dan *throughput*?
3. Bagaimana pengaruh *bit rate* CODEC terhadap performansi *video streaming* pada jaringan HSDPA (*High Speed Downlink Packet Access*) yang berbasis IPv6.

## 1.3 RUANG LINGKUP

Berdasarkan perumusan masalah di atas, maka pembahasan dibatasi pada:

1. Performansi *video streaming* yang diamati adalah dari *server* hingga *user*.
2. Aplikasi video *streaming* menggunakan jenis CODEC AMR-WB+ untuk *audio* dan H.264/AVC untuk video.
3. Jaringan HSDPA yang dipergunakan adalah jaringan HSDPA yang merupakan Release 5 standard 3GPP.
4. Internet protokol yang dipergunakan adalah IPv6.
5. Pembahasan meliputi analisis secara perhitungan berdasarkan data sekunder yang telah ditentukan.

## 1.4 TUJUAN

Tujuan penulisan skripsi ini adalah untuk mengkaji bagaimana performansi *video streaming* pada jaringan HSDPA (*High Speed Downlink Packet Access*) yang berbasis IPv6.

## 1.5. KONTRIBUSI

Kontribusi yang dapat diberikan dengan penulisan skripsi ini, diantaranya adalah sebagai berikut:

- Memaksimalkan perfomansi video *streaming* pada jaringan HSDPA dengan menggunakan IPv6.
- Masyarakat memiliki alternatif pilihan untuk menikmati layanan video *streaming* yang lebih baik.
- Bagi Penyelenggara Layanan Telekomunikasi, dapat digunakan sebagai bahan acuan dalam meningkatkan *Quality of Service* (QoS) teknologi telekomunikasi, khususnya video *streaming* serta meningkatkan kepuasan dan kenyamanan pengguna layanan video *streaming*.

## 1.6. SISTEMATIKA PENULISAN

Sistematika penulisan dan gambaran untuk setiap bab pada skripsi ini adalah sebagai berikut: BAB I PENDAHULUAN. Pada Bab II ini memuat tentang latar belakang, rumusan masalah, batasan masalah, tujuan, metodologi, dan sistematika penulisan. BAB II TINJAUAN PUSTAKA, pada bab ini membahas tentang konsep dasar jaringan HSDPA, karakteristik HSDPA yang meliputi s*hared channel transmission, Adaptive Modulation Coding* (AMC), TTI (*Transmissions Time Interval*), *fast link adaptation, fast scheduling*, dan HARQ (*Hybrid Automatic Repeat and Request*), serta membahas tentang konfigurasi jaringan HSDPA, *protocol stack* HSDPA, dan performansi *delay end-to-end* pada Jaringan HSDPA, pengertian video s*treaming*, CODEC (COdec/DECoder), persamaan paket data aplikasi video s*treaming*, parameter-parameter performansi video s*treaming*, dan konsep dasar IPv6. BAB III METODOLOGI PENELITIAN, pada bab ini dibahas tentang jenis data dan cara pengambilan data, variabel dan cara analisis data, dan kerangka solusi permasalahan dalam mengerjakan penelitian. BAB IV PEMBAHASAN DAN HASIL, pada bab ini dilakukan analisis perhitungan terhadap performansi *video streaming* pada jaringan

HSDPA (*High Speed Downlink Packet Access*) yang berbasis IPv6, meliputi konsumsi *bandwidth*, probabilitas *packet loss*, *delay end-to-end*, dan *throughput*. BAB V PENUTUP, berisi mengenai kesimpulan yang diperoleh dari teori dan analisis perhitungan yang telah dilakukan serta pemberian saran-saran.

# BAB II
# DASAR TEORI

## 2.1 Umum

Seiring dengan berkembangnya teknologi telekomunikasi serta mobilitas masyarakat yang semakin tinggi, layanan multimedia yang praktis banyak diminati masyarakat saat ini. Salah satu layanan multimedia yang paling diminati adalah layanan video *streaming*. Video *streaming* adalah sebuah teknologi untuk memainkan file video secara langsung ataupun dengan pre-recorder dari sebuah mesin server (web server). Dengan kata lain, file video yang terletak dalam sebuah server dapat secara langsung dijalankan pada UE sesaat setelah ada permintaan dari user, sehingga proses running aplikasi yang didownload berupa waktu yang lama dapat dihindari tanpa harus melakukan proses penyimpanan terlebih dahulu. [/www.ittelkom.ac.id/library]

Salah satu jaringan yang mendukung layanan video *streaming* adalah jaringan HSDPA. HSDPA (*High Speed Downlink Packet Access*) atau yang biasa dikenal dengan teknologi 3.5 G merupakan suatu teknologi telekomunikasi bergerak yang dikeluarkan oleh 3GPP Release 5 dan merupakan pengembangan dari WCDMA. Agar aplikasi video *streaming* dapat berjalan dengan lebih baik, maka digunakan IPv6 pada jaringan HSDPA. IPv6 memiliki beberapa keunggulan dibandingkan dengan IPv4.

Pada Bab ini akan dibahas mengenai pengertian video s*treaming*, CODEC (COdec/DECoder), persamaan paket data aplikasi video s*treaming*, parameter-parameter performansi video s*treaming*, konsep-konsep dasar jaringan HSDPA, karakteristik HSDPA yang meliputi s*hared channel transmission, Adaptive Modulation Coding* (AMC), TTI (*Transmissions Time Interval*), *fast link adaptation, fast scheduling*, dan HARQ (*Hybrid Automatic Repeat and Request*), serta membahas tentang konfigurasi jaringan HSDPA, *protocol stack* HSDPA, dan performansi *delay end-to-end* pada Jaringan HSDPA, serta konsep dasar IPv6.

## 2.2 Video *Streaming*

Awalnya telekomunikasi dua arah langsung yang semula hanya berupa audio saja, kini telah berkembang menjadi audio sekaligus video yang tercakup dalam dua arah percakapan jarak jauh. *Teleconference* dan video *streaming* internet menjadi contoh generasi terbaru sistem komunikasi global.

Dalam dunia internet, *streaming* lebih mengacu kepada teknologi yang mampu mengkompresi atau menyusutkan ukuran *file* audio dan video agar mudah dikirimkan melalui jaringan internet. Pengiriman *file* audio dan video tersebut dilakukan secara ”*stream*”, alias terus-menerus. Dari sudut pandang prosesnya, *streaming* berarti sebuah teknologi pengiriman *file* dari *server* ke *client* melalui jaringan *packet-based* semisal internet. Sedangkan dari sudut pandang *user*, *streaming* adalah teknologi yang memungkinkan suatu *file* dapat segera dijalankan tanpa harus menunggu selesai di*download* seluruhnya dan terus mengalir tanpa ada interupsi.

**Tabel 2.1** Jenis-jenis Video Streaming

| No. | Jenis Video | Penjelasan |
|---|---|---|
| 1. | *Live streaming* | Terjadi ketika setiap *user* melihat video *streaming* yang sama secara bersamaan |
| 2. | *On-demand streaming* | Terjadi saat pengguna dapat meminta *prerecorded* video dialirkan kepada pengguna ketika pengguna ingin melihat video tersebut. Dalam hal ini, setiap user dapat memilih kapan saat untuk memulai dan mengakhiri *streaming* video. |
| 3. | *Download and play* | Menggunakan memori atau hard disk di dalam perangkat *user* untuk menerima file audio/ video yang kemudian dapat ditampilkan (dimainkan) oleh perangkat *user*. Salah satu keuntungan utama dari *download and play* adalah bahwa file audio/video dapat dikirimkan dengan HTTP, sehingga kemungkinan dapat melewati firewall lebih besar. Beberapa *server* tidak setuju dengan *download and play*, karena hasil salinan video tersimpan di perangkat penyimpanan setiap user sebelum video dilihat/ditampilkan. |
| 4. | *Progressive download and play* | *Progressive download and play* adalah salah satu metode yang paling populer untuk pengiriman file video/audio melalui Internet, dan digunakan oleh situs-situs seperti *YouTube*. Teknologi ini pada dasarnya sama seperti *download and play*, kecuali *file* video/audio dipecah menjadi *file-file* kecil yang dikirimkan sebagai video *playback*. *Progressive download and play* memiliki keuntungan, yaitu dapat melewati *firewall* dengan mudah ketika mentransmisikan kecepatan dan kebutuhan *storage* yang lebih kecil. |

**(Sumber**: Wes Simpson, 2008: 41

File yang dapat ditransmisikan oleh video *streaming* adalah *file* audio, video, *image*, *text*, data 3D, *software*, dan sebagainya. Tetapi streaming lebih mengacu kepada *time based* media, khususnya audio dan video, yang harus dapat dinikmati sesegera mungkin dan berdasarkan pada pewaktuan yang tepat, karena untuk dapat menikmati lagu atau film, haruslah dimainkan secara beruruan dari awal hingga akhir (*sequential*) tanpa terputus-putus (*uninterrupted*). Ada beberapa jenis video *streaming* seperti ditunjukkan pada Tabel 2.1.

Ada beberapa bentuk komunikasi pada video *streaming*, yaitu: [www.ittelkom.ac.id/library]

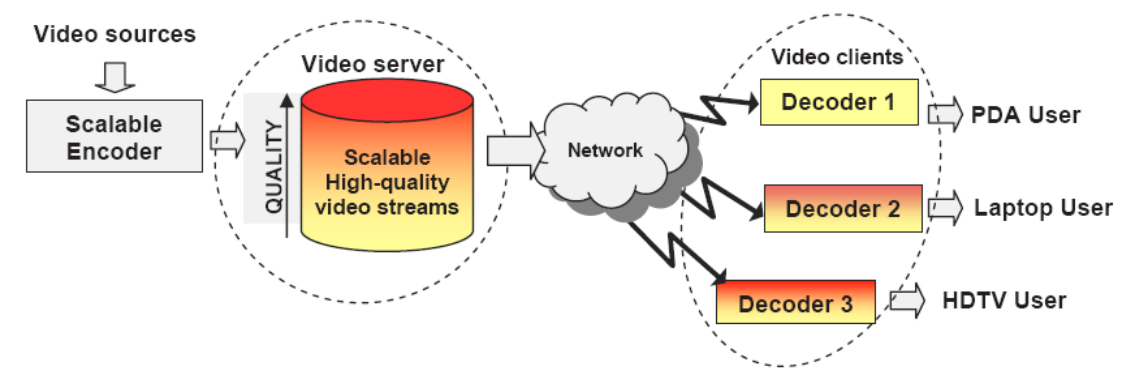
a. *Broadcast* adalah bentuk komunikasi *one to many* (yang pada dasarnya adalah *one to all*) yang paling dikenal, salah satu contohnya adalah siaran TV. Dengan menggunakan *broadcast* maka informasi yang akan dikirimkan berasal dari satu sumber atau titik kepada semua penerima yang yang tergabung dalam jaringan. Pada *broadcast* semua penerima mau tidak mau akan menerima informasi ini.
b. *Multicast*, merupakan bentuk komunikasi *one to many*, tetapi tidak seperti *broadcast*. Pada *multicast*, informasi yang dikirimkan berasal dari satu sumber atau titik kepada semua penerima yang menginginkan informasi tersebut. *Server* akan membuat *stream* satu kali kemudian *stream* ini diduplikasi dan dikirimkan ke setiap *client*. Setiap *client* akan menerima *stream* yang sama dengan *client* lainnya. Salah satu contoh penggunaan *multicast* adalah *live video*.
c. *Unicast* merupakan bentuk komunikasi *one to one* atau *point to point*. Pada *unicast*, informasi yang dikirimkan berasal dari satu sumber atau titik ke satu titik lainnya. *Server* akan mengirimkan file *streaming* ke komputer *client* berkali-kali bergantung pada banyaknya jumlah permintaan. Setiap *client* akan menerima file *streaming* yang terpisah dari client lainnya. Contoh penggunaan *unicast* adalah *video on demand*.

### 2.2.1 CODEC (*Coder/DECoder*)

Pada aplikasi multimedia baik untuk *voice*, video, maupun data, besarnya *bandwidth* tergantung dari CODEC yang digunakan. CODEC terdiri *Encoder* yang digunakan untuk melakukan *encoding* pada *file* audio/video sehingga memiliki ukuran yang lebih kecil dengan memperkecil *pixel* dan *frame* serta *Decoder* yang digunakan

untuk membaca *file* yang telah di-*encode* dan memainkannya di sisi *client*. CODEC sangat diperlukan dalam proses *streaming* video.

Pada proses *streaming* video, video dari *source* akan di-*capture* dan disimpan pada sebuah *buffer* yang berada pada memori komputer (bukan media penyimpanan seperti *harddisk*) dan kemudian di-*encode* sesuai dengan format yang diinginkan. Dalam proses *encode* ini, *server* dapat mengkompresi data sehingga ukurannya tidak terlalu besar (bersifat *optional*). Hal tersebut dilakukan karena keterbatasan *bandwitdh* jaringan. Setelah di-*encode*, data akan di-*stream* ke *user* yang lain. *User* akan melakukan *decode* data dan menampilkan hasilnya ke layar *user* untuk menikmati video *streaming*. Proses *streaming* video dapat ditunjukkan pada Gambar 2.1.




**Gambar 2.1** Proses *streaming* video melalui jaringan
(**Sumber**: Ashraf M.A. Ahmad and Ismail Khalil Ibrahim, 2009: 94)

Terdapat dua jenis CODEC yang dapat digunakan dalam video *streaming*, yaitu (Andreas Handojo, dkk, 2009: 1):

a. *Lossy* CODEC: CODEC ini akan mengurangi kualitas data dengan mengurangi ukuran data (kompresi). Pada umumnya, CODEC ini digunakan untuk menyimpan data pada media penyimpanan yang berukuran terbatas seperti CD-ROM dan DVD. *Lossy* CODEC ini biasanya digunakan untuk *streaming*, karena *bandwdith* jaringan yang terbatas. Contoh: Windows Media Video, H.264.
b. *Lossless* CODEC: Pada *lossless* CODEC ini, kualitas data yang dihasilkan tidak akan berkurang. Tetapi, ukuran data yang dihasilkan oleh *lossless* CODEC ini akan lebih besar dibandingkan dengan *lossy codec*. Pada umumnya, *lossless* CODEC ini digunakan pada video yang masih memerlukan *editing*, karena dalam proses *editing* dilakukan *encode-decode* berulang kali, sehingga jika menggunakan *lossy* CODEC,

kualitas video akan jauh menurun dibandingkan dengan video aslinya. Contoh: CorePNG, huffyuv, Apple Lossless Audio Codec.

Ada beberapa jenis audio dan video CODEC menurut ITU.T, seperti ditunjukkan pada Tabel 2.2.

**Tabel 2.2** Jenis CODEC Video dan Audio

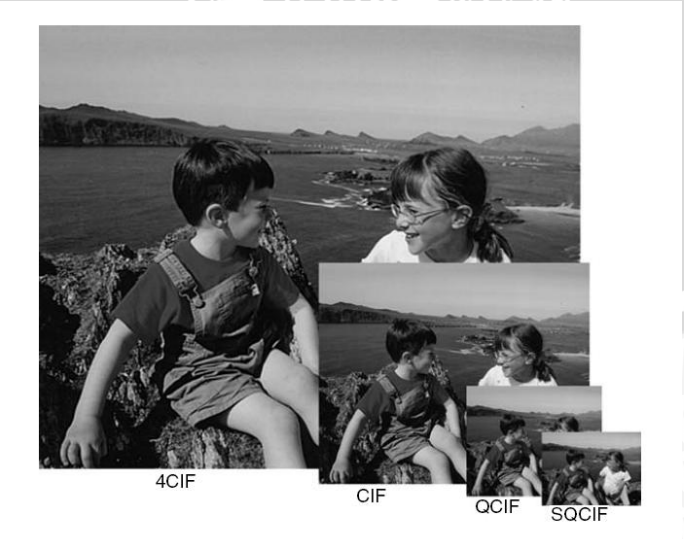
| **Audio CODEC** | **Bit Rate (kbps)** | ***Maximum Payload* (byte)** | ***Delay* CODEC (ms)** |
|---|---|---|---|
| AMR-WB | 6,6 – 23,85 | 35 | 10 – 20 |
| AMR-WB+ | 5,2 – 48 | 46 | 20 – 40 |
| HE-AAC v2 | 128 – 320 | 80 | 40 – 80 |
| **Video CODEC** | **Bit Rate (kbps)** | ***Maximum Payload* (byte)** | ***Delay* CODEC (ms)** |
| H.264/AVC | 64 – 384 | 254 | 150 – 300 |

(**Sumber**: RFC 4352,2006 dan RFC 3984, 2005)

### 2.2.2 Kualitas Video

Video dapat juga disebut sebagai gambar-gambar yang bergerak. Dalam video tampilkan sejumlah gambar atau frame dengan kecepatan tertentu yang disebut dengan istilah frame rate, yang dihitung dalam skala frame per second (fps). Seperti jenis data yang lain, data video juga dapat disimpan, diedit, ataupun dikirim melalui jaringan.

Ada beberapa format gambar yang digunakan dalam aplikasi video *streaming* diantaranya adalah format CIF, QCIF, SQCIF dan 4CIF. Perbandingan untuk setiap format gambar dapat dilihat pada Gambar 2.2.




**Gambar 2.2** Format Gambar

(**Sumber**: Iain E. G. Richardson, 2003: 20)

Pilihan resolusi frame tergantung pada aplikasi dan kapasitas yang tersedia atau kapasitas transmisi. Sebagai contoh, 4CIF sesuai untuk digunakan pada televisi standar dan DVD-video. Format CIF dan QCIF sangat populer digunakan pada aplikasi video conferencing. Sedangkan format QCIF atau SQCIF sesuai untuk digunakan pada aplikasi *mobile multimedia* dimana resolusi layar dan bit rate format ini terbatas.

Kualitas video yang baik dapat dinilai dari tiga elemen utama, yaitu: [*Streaming White Paper*, 1998: 5]:

a. Frame Rate, jumlah gambar yang ditampilkan per detik pada video. Minimum frame yang dibutuhkan untuk mendapatkan ilusi gambar yang bergerak adalah sekitar 15 fps.
b. Color Depth, Jumlah bit pada setiap pixel yang menunjukkan informasi warna. Misalnya: 24 bit menunjukkan 16.7 juta warna, 16 bit sekitar 65,535 warna, atau 8 bit hanya 256 warna.
c. Frame Resolution, biasanya ditunjukkan dengan width dan height pada pixel. Misalnya: full screen PC mempunyai frame resolution sebesar 640x480.

Jika kita menggunakan kecepatan pengiriman kecepatan pengiriman frame per second (fps) video yang rendah, akan memakan bandwidth yang lebih rendah dibandingkan frame per second (fps) yang tinggi. Video yang cukup baik biasanya dikirim dengan kecepatan frame per second (fps) sekitar 30 fps.

## 2.3 HSDPA (*High Speed Downlink Packet Access*)

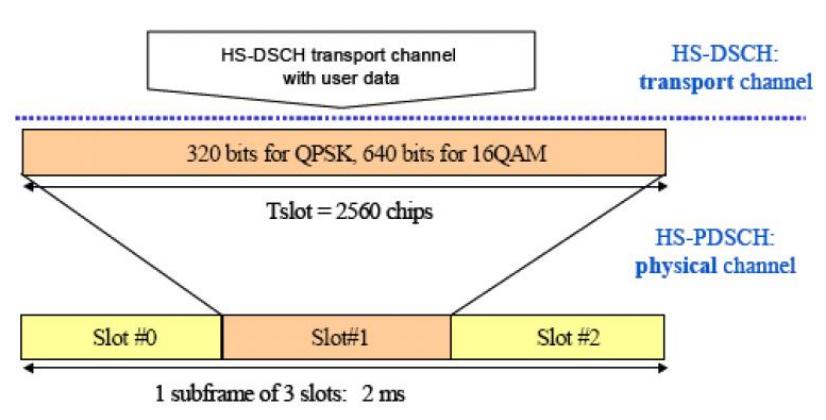
HSDPA merupakan pengembangan dari *air interface* pada jaringan WCDMA yang bertujuan meningkatkan performansi WCDMA untuk mengatasi peningkatan *user* dan perkembangan komunikasi data yang mengarah pada komunikasi multimedia. Konsep HSDPA didasarkan pada beberapa karakteristiknya, yaitu: [Ajai John, 2009: 6]

- *Shared channel transmission*
- *Adaptive Modulation Coding* (AMC)
- *Short Transmission Time Interval (TTI)*
- *Fast link adaptation*
- *Fast scheduling*
- *Fast Hybrid Automatic Repeat Request (H-ARQ)*

### 2.3.1 *Shared Channel Transmission*

Beberapa kanal baru diperkenalkan dalam jaringan HSDPA, yaitu *High-Speed Downlink Shared Channel* (HS-DSCH), *high-speed shared control channel* (HS-SCCH), dan HS-DPCCH (*High Speed-Dedicated Physical Control Channel*).

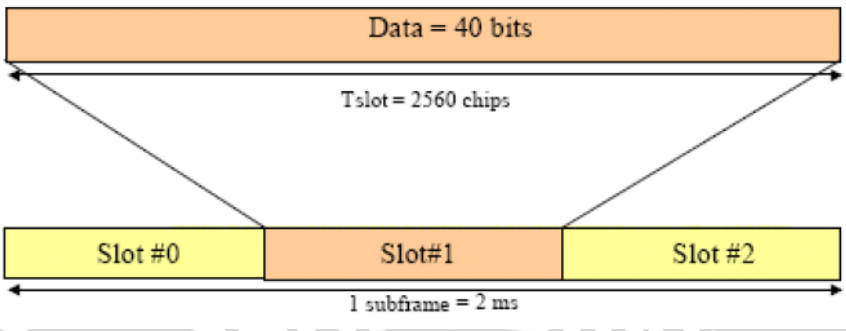
1. HS-PDSCH (*High Speed-Physical Downlink Shared Channel*), adalah kanal fisik *downlink* yang membawa data *user* HS-DSCH. HS-PDSCH memiliki *spreading factor* tetap sebesar 16 Satu *transport block* HS-DSCH ditransmisikan dengan transmission time interval (TTI) sebesar 2 ms HS-PDSCH menggunakan skema modulasi QPSK maupun 16-QAM.




**Gambar 2.3** Struktur Frame HS-PDSCH
(**Sumber**: http://www.scribd.com/doc/19864169/HSDPA-THESIS)

2. HS-SCCH (*High Speed-Shared Control Channel*), adalah kanal pensinyalan downlink yang membawa informasi untuk UE (*User Equipment*) sebelum memulai penjadwalan TTI. Kanal ini memberitahu UE bila ada data pada HS-DSCH yang dialamatkan ke UE tertentu, dan memberikan UE perubahan parameter dengan cepat yang diperlukan untuk penerimaan HS-DSCH. HS-SCCH memiliki *spreading factor* sebesar 128. HS-SCCH berisi informasi kontrol dan penjadwalan sebagai berikut: [Mohammad Assaad and Djamal Zeghlache, 2007: 83-84]
   - Identifikasi UE identification
   - HSPDSCH *channelization codes*
   - Informasi skema modulasi HSPDSCH
   - Informasi *transport block size*
   - Informasi proses HARQ

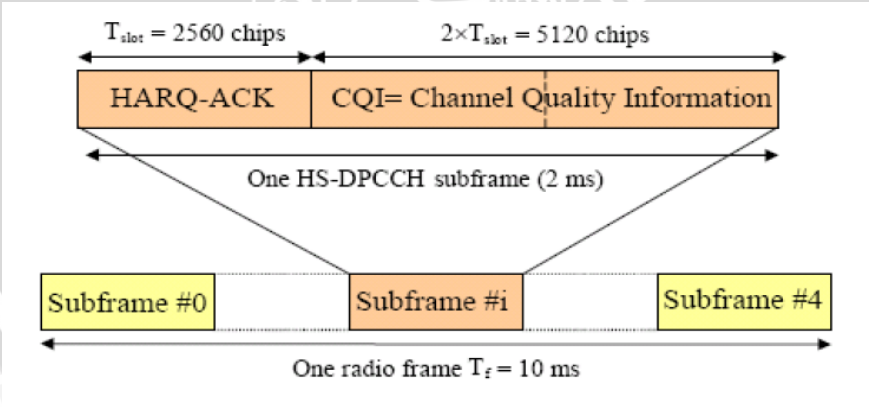
- *Redundancy* and versi *constellation*
- Indikator data HARQ yang baru




**Gambar 2.4** Struktur Frame HS-SCCH

(**Sumber**: http://www.scribd.com/doc/19864169/HSDPA-THESIS)

3. HS-DPCCH (*High Speed-Dedicated Physical Control Channel*), adalah kanal kontrol *uplink* yang membawa *feed-back* pensinyalan yang berhubungan dengan transmisi *downlink* HS-DSCH dan kontrol infromasi berupa HARQ ACK/NACK dan CQI. Kanal ini akan memberitahukan apakah hubungan transmisi pada arah *downlink* telah sukses dikodekan dan CQI (*Channel Quality Indicator*) yang digunakan telah sesuai dengan *link adaptation*.



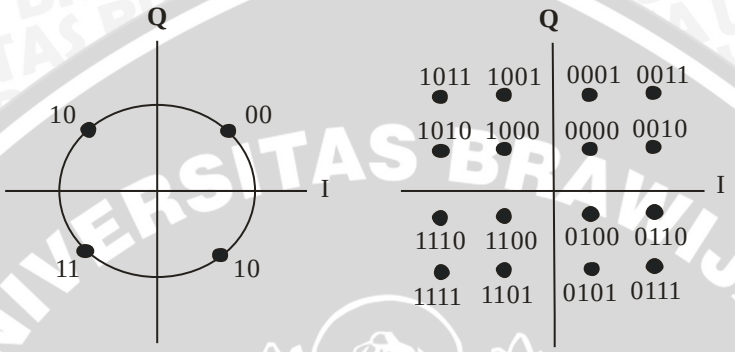

**Gambar 2.5** Struktur Frame HS-DPCCH

(**Sumber**: http://www.scribd.com/doc/19864169/HSDPA-THESIS)

### 2.3.2 *Adaptive Modulation Coding* (AMC)

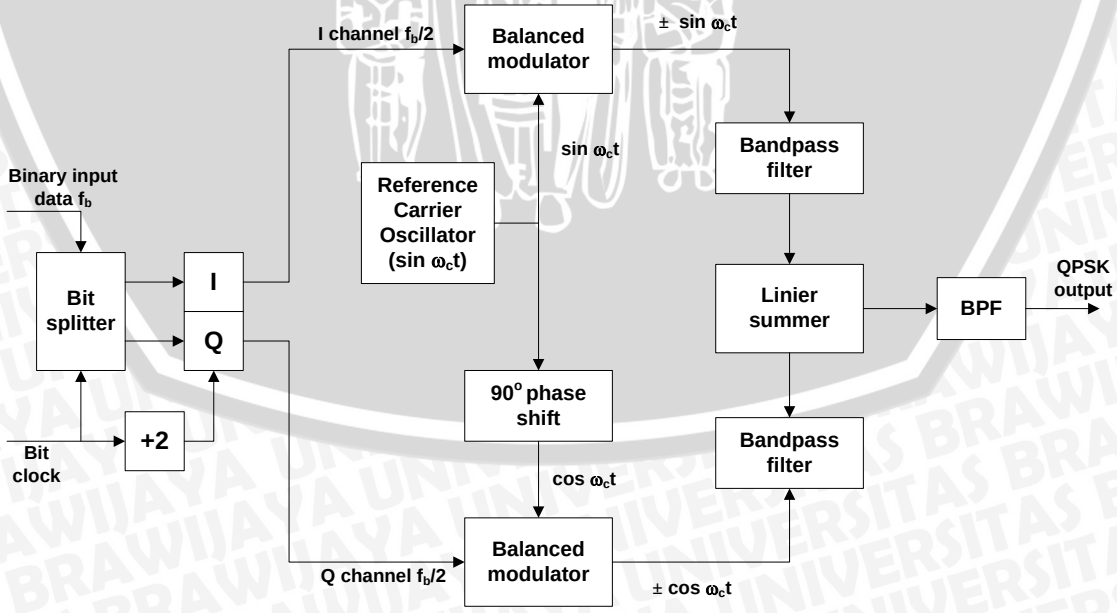
AMC merupakan teknologi utama pada HSDPA dimana feedback dari UE digunakan untuk menentukan skema *coding* dan modulasi yang akan digunakan berdasarkan CQI (*Channel Quality Indicator*) [Ajai John, 2009: 15]. Proses ini

dilakukan untuk setiap TTI dengan tujuan untuk memaksimalkan *data rate* dari UE dengan kondisi kanal yang baik. Modulasi pada HS-DSCH dilakukan secara adaptif dengan pemilihan modulasi QPSK (*Quadrature Phase Shift Keying*) atau 16 QAM (*Quadrature Amplitude Modulation*). Skema modulasi QPSK dan 16-QAM dapat ditunjukkan pada Gambar 2.6 di bawah ini




**Gambar 2.6** Skema Modulasi QPSK dan 16QAM
(**Sumber**: http://www.ittelkom.ac.id/library)

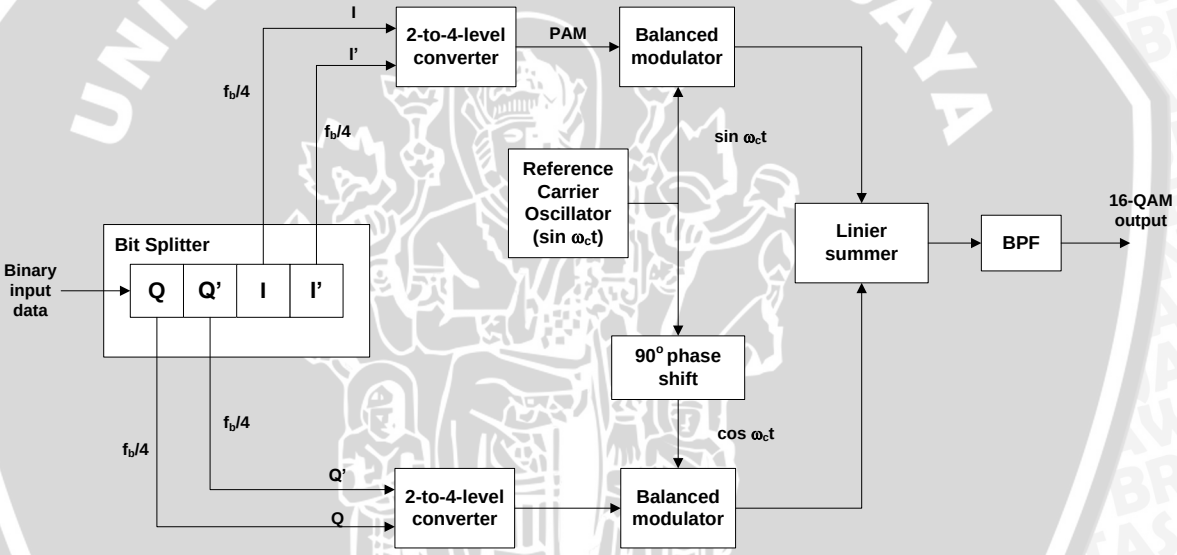
a. QPSK merupakan modulasi *M-ary* PSK (*Phase Shift Keying*) dengan beda fasa untuk masing-masing simbol sebesar 90 derajat. QPSK merepresentasikan 2 bit dalam setiap simbol. Blok diagram modulator QPSK dapat dilihat pada Gambar 2.7 di bawah ini.




**Gambar 2.7** Blok Diagram Modulator QPSK
(**Sumber:** Wayne Tomasi, 2004: 349)

Pada blok diagram di atas, sinyal informasi dalam bentuk biner dengan bit rate tertentu ($f_b$) dipisahkan melalui bit *splitter* dimana terjadi proses *serial to paralel convertion* sehingga keluaran sinyal informasi melalui kanal I dan kanal Q. Pada *balanced* modulator, sinyal informasi dimodulasi oleh sinyal carrier yang berasal dari *reference carrier oscillator*. Selanjutnya, sinyal termodulasi kedua kanal dilewatkan melalui bandpass filter dan dijumlahkan sehingga dihasilkan 4 macam keluaran QPSK.

b. QAM merupakan kombinasi jenis modulasi *M-ary* ASK (*Amplitude Shift Keying*) dan *M-ary* PSK. Pada sistem HSDPA jenis QAM yang digunakan adalah 16-QAM dimana tiap simbolnya merepresentasikan 4 bit.




**Gambar 2.8** Blok Diagram Modulator 16-QAM
(**Sumber:** Wayne Tomasi, 2004: 365)

Pada blok diagram di atas, *binary* input data merupakan sinyal informasi yang telah diubah dalam bentuk bit data. Input ini kemudian dipisahkan pada bit *splitter* (*serial to parallel convertion*) untuk menghasilkan masing-masing dua bit pada kanal Q dan kanal I. Selanjutnya, 2-to-4-level *converter* digunakan untuk membangkitkan signal PAM. Pada balance modulator kanal I, sinyal PAM dimodulasi dengan sinyal carrier yang sefasa dengan *reference carrier oscillator*. Sedangkan pada balance modulator kanal Q, sinyal PAM dimodulasi dengan sinyal carrier yang berbeda fasa $90^o$ dengan *reference carrier oscillator*. Selanjutnya,

linier summer menggabungkan keluaran sinyal termodulasi dari kanal I dan Q sehingga dihasilkan 16 macam keluaran 16-QAM.

### 2.3.3 *Transmissions Time Interval* (TTI)

HSDPA menerapkan TTI sebesar 2ms untuk sebuah *sub-frame* HS-DSCH yang terdiri dari 3 slot. Sebuah UE (*User Equipment*) dapat menggunakan semua kode kanalisasi sepanjang TTI atau beberapa UE menempati TTI yang sama dengan alokasi kode kanalisasi yang berbeda. Dengan TTI sebesar 2 ms, maka dapat mengurangi *roundtrip time* dan meningkatkan variasi kanal *tracking*.

### 2.3.4 *Fast Link Adaptation*

Adaptasi *link* pada HSDPA berdasarkan pada layer fisik CQI yang disediakan oleh terminal. Pengguna yang relatif lebih dekat dengan *base station* mentransmisikan level daya yang lebih tinggi daripada yang dibutuhkan. Adaptasi *link* mengatasi ekstra margin tersebut dengan menyeleksi parameter transmisi sehingga energi simbol yang dibutuhkan sesuai dengan daya simbol yang tersedia. Adaptasi *link* dilakukan berdasarkan informasi CQI yang juga mempertimbangkan aspek lain selain kekuatan sinyal atau C/I.

### 2.3.5 *Fast Scheduling*

Fitur *fast-scheduling* menentukan UE tertentu dalam transmisi kanal bersama yang harus diarahkan pada waktu tertentu. Tujuannya adalah untuk mentransmisikan sinyal pada pengguna dengan kondisi radio yang terbaik. Untuk setiap TTI, *scheduler* menentukan kepada pengguna mana HS-DSCH harus ditransmisikan serta menentukan modulasi dan banyaknya kode yang harus digunakan dengan menggunakan mekanisme *link adaptation*. Dengan adanya MAC-hs pada *node B* maka *delay* transmisi paket yang terjadi dapat dikurangi. Tiga cara penjadwalan dipakai dalam sistem HSDPA yaitu:

a. *Round Robin* (RR)

   Penjadwalan RR bekerja berdasarkan posisi antrian, *first in first out*. Meksipun paling sederhana dan fair, kondisi kanal yang dipakai UE tidak dijadikan pertimbangan. Sebagai konsekuensinya pengguna tetap dijadwal meskipun kondisi kanal buruk.

b. Maximum C/I

Algoritma Maximum C/I menjadwal UE berdasarkan rasio C/I sesaat. User dengan nilai C/I paling tinggi akan mendapatkan prioritas lebih tinggi. Asumsinya seluruh UE memiliki level MCS (*Modulation and Coding Scheme*) tertinggi untuk melakukan transmisi. Hal tersebut kurang fair karena menyebabkan hampir setengah pengguna sel tidak memperoleh pelayanan yang cukup.

c. *Proportional Fair* (PF)

PF merupakan bentuk kompromi antara RR dan Maximum C/I. PF bekerja berdasarkan keseimbangan antara rata-rata *Throughput* yang diperoleh dengan *data rate* sesaat. Hasilnya setiap pengguna dilayani saat kondisi kanal mendukung.

### 2.3.6 *Fast Hybrid Automatic Repeat and Request* (HARQ)

HARQ meningkatkan performansi dan menambah ketahanan terhadap *error* pada *link adaptation*. Penerima akan mengirim NACK melalui HS-DPCCH ketika mendeteksi error pada paket data yang diterima setelah 7.5 *time slot* dari akhir TTI HS-DSCH. Teknologi HARQ mengkombinasikan FEC (*Feed Error Correction*) dan ARQ untuk menyelamatkan informasi dari kegagalan transmisi sebelumnya untuk keperluan *decoding* pada UE [http://www.ittelkom.ac.id/library]

Untuk retransmisi, HARQ menggunakan TBS (*Transport Block Size*) yang sama dengan transmisi sebelumnya dimana jumlah bit informasi yang dikirimkan sama. Namun jenis modulasi, *channelization code* atau daya transmisi yang digunakan dapat berbeda. HARQ pada HSDPA dapat terdiri atas teknik berikut: [Ajai John, 2009: 14-15]

a. *Chase Combining* (CC)

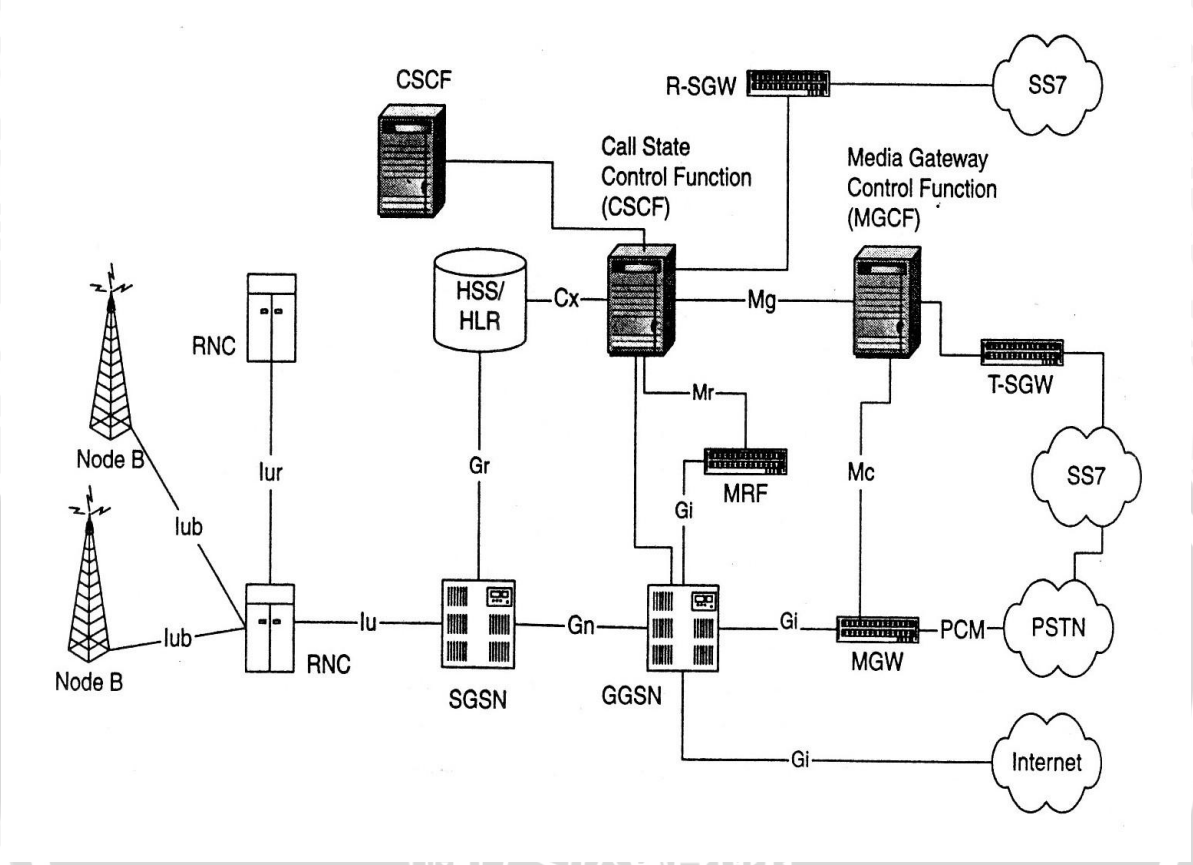
Pada CC, bit-bit yang sama akan dikirimkan pada retransmisi ketika terjadi *error* pada proses *decoding*. Paket transmisi digabungkan dengan paket sebelumnya (soft combining) untuk dapat di-*decode* selanjutnya.

b. *Incremental Redundancy* (IR)

Pada IR, bit parity tambahan dikirimkan saat terjadi *error* pada proses *decoding*. Bit *parity* tambahan ini digunakan bersama dengan bit *parity* yang asli sehingga dapat mengurangi *code rate*.

### 2.3.7 Konfigurasi Jaringan HSDPA

Berikut ini merupakan konfigurasi jaringan HSDPA yang ditunjukkan pada Gambar 2.9.




**Gambar 2.9** Arsitektur Jaringan HSDPA

(**Sumber**: Clint Smith, P.E and Daniel Collins, 2002:155)

1. *Core network*

   *Core network* terdiri dari beberapa bagian, yaitu :

   a. *Serving GPRS Support Node* (SGSN): berfungsi sama halnya seperti MSC/VLR (*Mobile Switching Centre/Visitor Location Register*), tetapi secara khusus digunakan untuk layanan *Packet Switched* (PS).

   b. *Gateway GPRS Support Node* (GGSN): berfungsi sama halnya seperti GMSC (*Gateway Mobile Service Swiching* Center), tetapi berhubungan dengan layanan-layanan PS.

2. RNC (*Radio Network Controller*)

   RNC (*Radio Network Controller*) digunakan untukm mengontrol sumber radio dimana *Node* B terhubung dengannya. RNC adalah layanan *access point* untuk semua layanan yang disediakan oleh NSS (*Network and Switchinbg Sub-System*).

Suatu RNC yang dengan beberapa Node B membentuk *Radio Network Subsystem* (RNS).

3. Node B

   Node B merupakan perangkat untuk mengkonversi aliran data antara *interface* Uu dan Iub, juga berperan dalam *radio resource management. Jadi,* Node B digunakan untuk mentransmisikan data dari *lub-interface* dan uu-*interface* atau sebaliknya serta ikut mengontrol sumber radio.

4. UE (*User Equipment*)

   UE *(User Equipment)* merupakan perangkat atau terminal pada sisi pelanggan yang berupa headset untuk mengirim dan menerima informasi. HSDPA memiliki 12 tipe UE (*User Equipment*). Perbedaaannya meliputi beberapa faktor, yaitu:

   - Kode-kode HS-PDSCH

     Merupakan jumlah maksimum kode HS-PDSCH yang dapat dialokasikan untuk UE, yaitu 5, 10, atau 15 kode. Untuk kode yang lebih rendah dari kode maksimum yang digunakan dilewatkan melalui HS-SCCH.

   - *Inter-TTI interval*

     Merupakan interval minimum yaitu 1 TTI sebesar 2 ms. Akan tetapi nilai TTI sebesar`1, 2, atau 3 diperbolehkan.

   - *Transport Block Size*

     Merupakan TBS (*Transport Block Size*) maksimum dari HS-DSCH yang dapat digunakan oleh UE pada sebuah TTI.

   - *Incremental redudancy buffer size*

     Merupakan jumlah soft bit yang dapat di-*buffer* oleh UE sepanjang proses H-ARQ aktif.

   - Modulasi

     Merupakan kemampuan sebuah UE untuk mendukung modulasi QPSK dan 16-QAM atau hanya modulasi QPSK saja.

   Kategori UE pada HSDPA ditunjukkan pada Tabel 2.3

**Tabel 2.3** Kategori *User Equipment* (UE) HSDPA

| | Parameters from 3GPP specifications | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| UE Category | Mod. scheme | Maximum number of HS-DSCH codes received | Minimum inter-TTI interval | Max. no. of bit per transport block received within an HS-DSCH TTI | Total number of soft channel bits | Maximum data rate; Mbps | Theoreti-cal max. data rate w/o channel coding; Mbps | IR HARQ possible at maximum data rate |
| 1 | 16QAM | 5 | 3 | 7298 | 19200 | 1,22 | 1,6 | No |
| 2 | 16QAM | 5 | 3 | 7298 | 28800 | 1,22 | 1,6 | Yes |
| 3 | 16QAM | 5 | 2 | 7298 | 28800 | 1,82 | 2,4 | No |
| 4 | 16QAM | 5 | 2 | 7298 | 38400 | 1,82 | 2,4 | Yes |
| 5 | 16QAM | 5 | 1 | 7298 | 57600 | 3,65 | 4,8 | No |
| 6 | 16QAM | 5 | 1 | 7298 | 67200 | 3,65 | 4,8 | Yes |
| 7 | 16QAM | 10 | 1 | 14411 | 115200 | 7,21 | 9,6 | No |
| 8 | 16QAM | 10 | 1 | 14411 | 134400 | 7,21 | 9,6 | Yes |
| 9 | 16QAM | 15 | 1 | 20251 | 172800 | 10,13 | 14,4 | No |
| 10 | 16QAM | 15 | 1 | 27952 | 172800 | 13,98 | 14,4 | No |
| 11 | QPSK | 5 | 2 | 3630 | 14400 | 0,91 | 1,2 | No |
| 12 | QPSK | 5 | 1 | 3630 | 28800 | 1,82 | 2,4 | No |

**Sumber**: HSDPA by Siemens, 2004: 17

5. *Interface*

Ada beberapa *interface* pada jaringan HSDPA, yaitu:

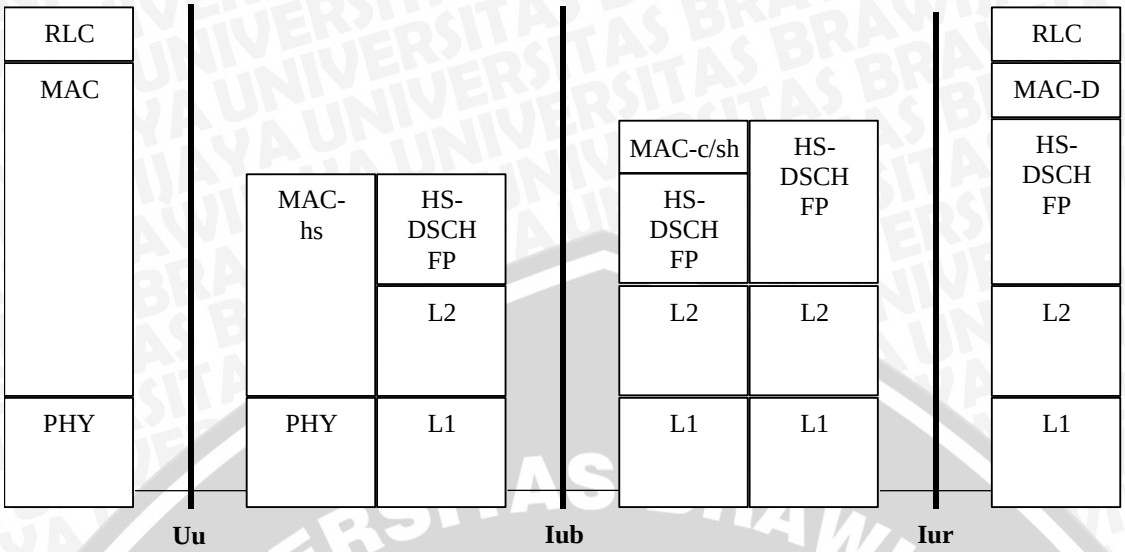
a. Uu *Interface*, sebagai terminal dari UMTS yang berhubungan dengan *radio interface* dari UTRAN dan aplikasi *user*.
b. lu *Interface*, merupakan *interface* antara *Node* B dan UE. *Interface* ini mengatur koneksi antara UE dengan UTRAN termasuk pada pembangunan, pemeliharaan, dan pemutusan koneksi.
c. lub *interface*, berfungsi menghubungkan RNC (*Radio Network Controller*) dengan *Node* B menggunakan NBAP (*Node B Application Part*) *signalling*. *Signalling* dari Node B ke RNC dan sebaliknya menggunakan protokol NBAP yang dilewatkan melalui lub-*interface*. Kode kanalisasi yang dialokasikan untuk transmisi HSDPA juga memerlukan *signalling* antara RNC ke *Node* B.
d. lur *Interface*, merupakan *interface* antar RNC. *Interface* ini mengatur *radio* dan *physical link*, serta *common transport channel resources*.
e. Gn *Interface*, merupakan *interface* antara SGSN dan GGSN, serta berfungsi untuk tunneling *payload* data *end user* dalam jaringan backbone.

f. Gi *Interface*, digunakan untuk transportasi semua *end user* data IP antara *core network* UMTS dan jaringan IP eksternal.

### 2.3.8 Protokol Stack HSDPA

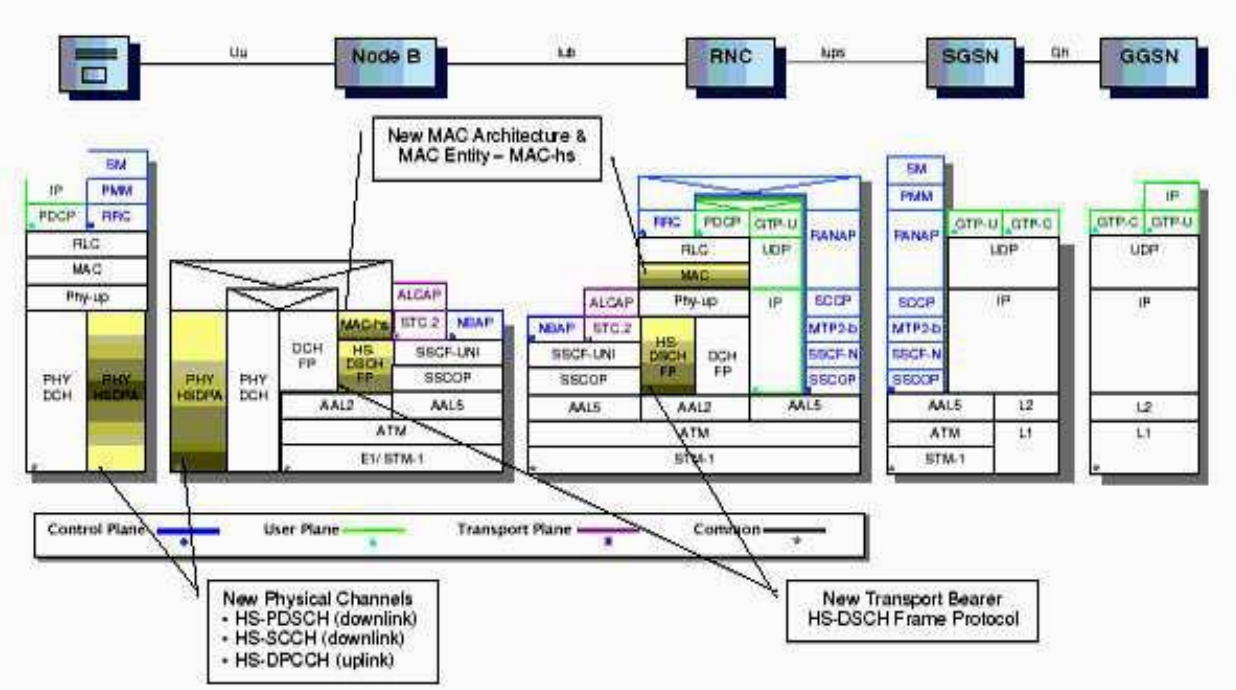
Pada HSDPA, *Node* B tidak hanya terdiri atas layer fisik. Perubahan pada *Node* B terjadi pada layer MAC (*Medium Access Control*), yaitu penambahan MAC-hs yang merupakan *entity* MAC yang menangani *transport channel* baru yang diperkenalkan HSDPA, yakni HS-DSCH. MAC-hs memiliki peran dalam fungsi retransmisi dan *schedulling* dalam menangani prioritas paket. Pada *Release*'99 (WCDMA), proses retransmisi dan *scheduling* dilakukan pada RNC, sedangkan pada HSDPA dilakukan pada *Node* B (BTS) [Y. Jay Guo, 2004: 78-79]. Oleh karena itu, waktu yang dibutuhkan untuk retransmisi lebih pendek. Dengan adanya layer MAC pada *Node* B, maka proses retransmisi dan *schedulling* dapat terjadi lebih cepat.

Fungsionalitas *Node* B berbasis *fast scheduling* merupakan fungsionalitas *layer* MAC sehingga pada HSDPA diperlukan suatu unit protokol baru pada *Node* B, yaitu MAC-hs (hs untuk *high speed*). RNC tetap mempertahankan MAC-d (d untuk *dedicated*), namun fungsi yang masih tersisa hanya sebagai *switching* kanal transport karena fungsi yang lain seperti *scheduling* dan prioritas penanganan telah dipindahkan ke MAC-hs. Layer di atas MAC yaitu layer RLC (*Radio Link Control*) dibiarkan tetap tanpa perubahan.




**Gambar 2.10** Protokol Stack UTRAN HSDPA

(**Sumber**: 3GPP TR 25.855 V2.0.0, 2001: 8




**Gambar 2.11** Protokol Stack jaringan HSDPA

**(Sumber**: PT. Indosat, Tbk KPI Surabaya dalam Linda Ekowati, 2008)

## 2.4 *Internet Protocol* versi 6 (IPv6)

Usaha untuk membangun IPv6 sebagai protokol pengganti IPv4 mulai dilaksanakan oleh *Internet Engineering Task Force* (IETF) pada awal tahun 1990-an.

IPv6 ini dirancang untuk memecahkan masalah keterbatasan ruang alamat alamat IPv4. Rekomendasi mengenai IPv6 yang terdapat dalam RFC 1752, "*The Recommendation for the IP Next Generation Protocol*" ditetapkan menjadi IETF *Draft Standard* pada 10 Agustus 1998. IP versi 6 (IPv6) adalah protokol internet versi baru yang didesain sebagai pengganti dari internet protokol versi 4 (IPv4).

Perubahan dari IPv4 ke IPv6 pada dasarnya terjadi karena beberapa hal yang dikelompokkan dalam kategori berikut [Sritrusta Sukaridhoto, 2008: 3]

- Kapasitas Perluasan Alamat

  IPv6 meningkatkan ukuran dan jumlah alamat yang mampu didukung oleh IPv4 dari 32 bit menjadi 128 bit. Peningkatan kapasitas alamat ini digunakan untuk mendukung peningkatan hirarki atau kelompok pengalamatan, peningkatan jumlah atau kapasitas alamat yang dapat dialokasikan dan diberikan pada *node*, dan mempermudah konfigurasi alamat pada *node* sehingga dapat dilakukan secara otomatis. Peningkatan skalabilitas juga dilakukan pada *routing multicast* dengan meningkatkan cakupan dan jumlah pada alamat *multicast*. IPv6 ini selain meningkatkan jumlah kapasitas alamat yang dapat dialokasikan pada *node*, juga mengenalkan jenis atau tipe alamat baru, yaitu alamat *anycast*. Tipe alamat *anycast* ini didefinisikan dan digunakan untuk mengirimkan paket ke salah satu dari kumpulan *node*.
- Penyederhanaan Format *Header*

  Beberapa kolom pada *header* IPv4 telah dihilangkan atau dapat dibuat sebagai *header* pilihan. Hal ini digunakan untuk mengurangi biaya pemrosesan hal-hal yang umum pada penanganan paket IPv6 dan membatasi biaya *bandwidth* pada *header* IPv6. Dengan demikian, pemrosesan *header* pada paket IPv6 dapat dilakukan secara efisiien.
- Peningkatan Dukungan untuk *Header* Pilihan dan *Header* Tambahan (*option and extension header*)

  Perubahan yang terjadi pada *header-header* IP yaitu dengan adanya pengkodean *header opions* (pilihan) pada IP dimaksudkan agar lebih efisien dalam penerusan paket (*packet forwarding*), agar tidak terlalu ketat dalam pembatasan panjang *header* pilihan yang terdapat dalam paket IPv6 dan sangat fleksibel dimungkinkan untuk mengenalkan *header* pilihan baru pada masa akan datang.
- Kemampuan pelabelan aliran paket

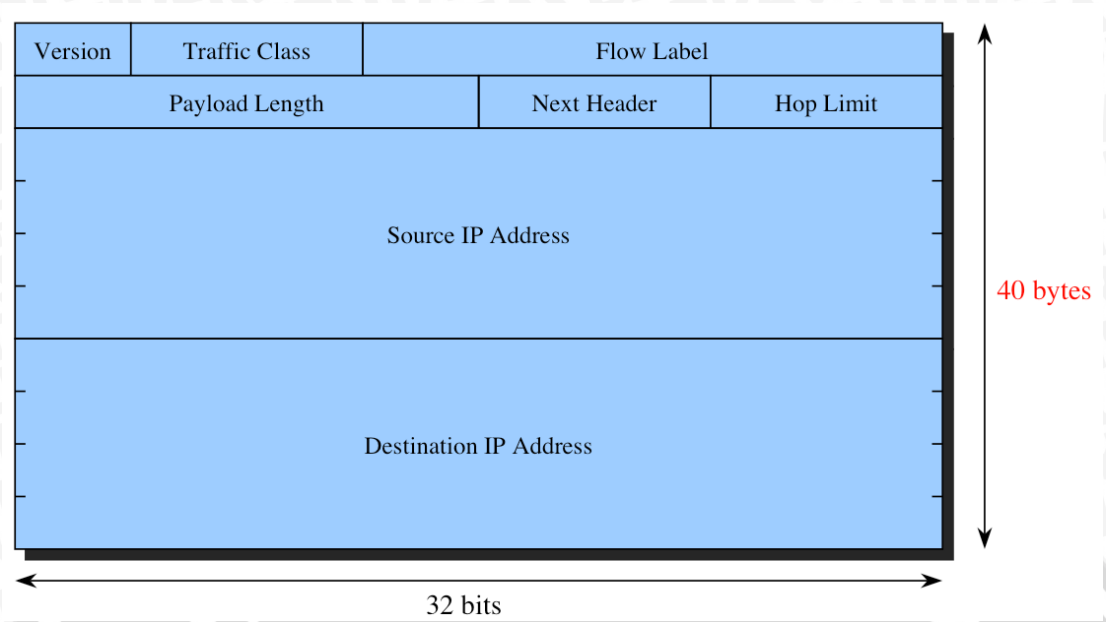
Kemampuan atau fitur baru yang ditambahkan pada IPv6 ini adalah memungkinkan pelabelan paket atau pengklasifikasian paket yang merupakan milik dari aliran trafik tertentu dan aliran trafik ini meminta penanganan spesial, seperti kualitas mutu layanan tertentu (QoS) atau layanan *real-time*.

- Autentifikasi dan Kemampuan Privasi
  Kemampuan tambahan untuk mendukung autentifikasi, integritas data dan data penting juga dispesifikasian dengan alamat IPv6.

### 2.4.1 Format *Header* IPv6

Format *header* alamat IPv6 menyederhanakan format *header* pada alamat IPv4. Setiap paket IPv6 membawa data yang terdiri dari [Silvia Hagen, 2003: 11]:

- *Version* (4 bit), berisi nomor versi protokol internet, yaitu 6.
- *Traffic Class* (8 bit), berisi kolom kelas atau klasifikasi trafik. Tersedia untuk digunakan oleh permulaan *node* atau *router forwarding* untuk mengidentifikasian dan membedakan antara kelas yang berbeda atau prioritas paket IPv6.
- *Flow Label* (20 bit), berisi label aliran dari trafi. Digunakan oleh *host* untuk melabeli paket-paket yang mana meminta penanganan khusus oleh *router* dalam suatu jaringan.
- *Payload Length* (16 bit), merupakan panjang *payload*/muatan IPv6. *Payload* ini adalah sisa paket setelah *header* IPv6 dalam bentuk oktet (di luar *header* IPv6). Catatan bahwa *header* tambahan/*extension header* yang ada dihitung sebagai bagian dari *payload* termasuk panjang *header* tambahan tersebut.
- *Next header* (8 bit), merupakan identifikasi tipe *header* yang akan ada setelah *header* IPv6. Nilai *heeder* ini menggunakan tipe *header* yang sama dengan yang ada pada protokol IPv4 (RFC 1700).
- *Hop Limit* (8 bit), nilai pada kolom ini akan dikurangi satu jika paket ini melewati *node* yang berfungsi melewatkan/mem*forward* paket (melewati *router*). Paket ini akan dibuang jika batas *hop* ini berubah menjadi nol.
- *Source Address* (128 bit), merupakan alamat IPv6 asal dari paket.
- *Destination Address* (128 bit), merupakan alamat IPv6 tujuan paket.




**Gambar 2.12** Format *header* IPv6

(**Sumber**: Rob Blokzijl, 2009: 3)

### 2.4.3 Perbandingan *Header* IPv4 dan IPv6

Pada tabel 2.4 di bawah ini memperlihatkan perbedaan antara *header* IPv4 dan *header* IPv6, yaitu:

**Tabel 2.4** Perbedaan *header* IPv6 dengan IPv4

| Perbedaan | *Field Header* IPv4 | *Field Header* IPv6 |
|---|---|---|
| *Version* | Berisi angka 4 (0100 dalam biner) | Berisi angka 6 (0110 dalam biner). |
| Internet *Header Length* | Internet *header length* terdiri dari 4 bit yang berisi panjang dari *header* paket IPv4 dalam blok. | Tidak memuat *field* Internet *Header Length* karena *header* IPv6 selalu memiliki panjang tetap 40 *byte*. |
| *Type of Service* | Penggunaan *type of service* dapat mempengaruhi cara penanganan paket IP. | digantikan oleh *field Traffic Class* yang berfungsi seperti *type of service* pada IPv4. |
| Total *Length of Datagram* | Berisi informasi jumlah total *datagram* pada layer IP dan menggunakan satuan *byte*. | digantikan oleh *field Payload Length* IPv6, yang hanya mengidentifikasikan ukuran *payload*. |
| *Options* | Penggunaan *option* dalam IPv4 menyatakan bahwa masing-masing *node intermediate* pada *path* harus memeriksa *field option* dalam *header* IPv4, meskipun *option* hanya | Dihapus dalam IPv6, digantikan oleh *extension header* IPv6 yang merupakan *header* pilihan yang tidak selalu diberikan pada paket Ipv6, diletakkan diantara *base header* IPv6 dan *header* |

| | | |
|---|---|---|
| | berkaitan dengan *node* tujuan. Sehingga menciptakan ketidakefisienan performansi *router* ketika digunakan *option*. | protokol *upper layer* (TCP/UDP). |
| *Time-to-Live* | *field* TTL digunakan untuk menunjukkan jumlah detik untuk hidup/berada dalam jaringan, sehingga mencegah paket untuk dirutekan secara berputar jika terdapat rute melingkar dalam jaringan. Tetapi, dalam implementasinya, *field* ini digunakan untuk membatasi jumlah *hop* yang diijinkan bagi paket untuk disinggahi. Pada tiap *hop*, *router* mengurangi *field* ini, dan ketika mencapai nol, paket akan dibuang dari jaringan. | *field* ini diberi nama lain *hop* limit, memiliki deskripsi yang lebih tepat sesuai implementasinya. |
| *Protocol* | *Field protocol* yang mengandung angka yang mengidentifikasikan protokol *upper layer*. | digantikan oleh *field Next Header*. |
| *Header Checksum* | *Field Header Checksum* digunakan untuk mempertahankan integritas *header* IPv4. Tetapi layer yang lebih tinggi menghitung *checksum* lagi untuk seluruh paket, sehingga membuat *field* ini digunakan secara berlebihan. | Dihilangkan dalam IPv6. Pada IPv6, jika aplikasi membutuhkan derajat integritas yang tinggi, deteksi *error* untuk seluruh paket dilaksanakan oleh *link layer* atau *layer* lain yang lebih tinggi. |
| *Source Address* | Berisi alamat pengirim paket dengan panjang alamat 32 bit | *Field* ini berperan sama seperti IPv4, menetapkan IP *address host* sumber, tetapi memiliki panjang alamat IPv6 128 bit. |
| *Destination Address* | Berisi alamat penerima paket dengan panjang alamat 32 bit | *Field* ini berperan sama seperti IPv4, menetapkan IP *address host* tujuan, tetapi memiliki panjang alamat IPv6 128 bit. |
| *Identification, Flags, Fragmen Offset* | Dalam *header* IPv4, ketiga *field* ini, seluruhnya berkaitan dengan penanganan fragmentasi dan penyusunan kembali paket-paket. Dalam IPv4, *node intermediate* dapat memfragmentasi paket ketika *Maximum Transmission Unit* | Dihapus dalam IPv6. Sedangkan pemrosesan fragmentasi dalam IPv6 dilakukan hanya pada *node* sumber, dengan menggunakan path MTU. Selanjutnya, informasi yang berkaitan dengan fragmentasi dikodekan dalam *extension header fragment*. |

| | | |
|---|---|---|
| | (MTU) pada *link outgoing* lebih kecil dibandingkan ukuran paket yang akan ditransmisikan pada *link* tersebut. | Sehingga, ketiga *field* ini tidak dibutuhkan dalam *header* IPv6. |
| *Flow Label* | Tidak ada | ***Flow Label***, satu *field* baru pada *header* IPv6 adalah *Flow Label*, yang tidak ditemukan dalam *Header* IPv4. *Field* ini digunakan untuk menandai aliran data tertentu yang membutuhkan penanganan khusus pada paket IPv6. |

## 2.5 Parameter Performansi Aplikasi Video *Streaming*

### 2.5.1 Paket Data Aplikasi *Video Streaming*

Pada aplikasi *video streaming*, paket yang ditransmisikan dibedakan atas paket audio dan paket video, dimana tiap paket tersebut mempunyai besar *payload* yang berbeda. Aplikasi video *streaming* menggunakan jenis CODEC H.264/AVC untuk video dengan *bit rate* CODEC antara 64 – 384 kbps dan AMR-WB+ untuk audio dengan *bit rate* CODEC antara 5,2 – 48 kbps. Besar *payload* tiap paket audio dan video dapat dinyatakan dengan persamaan:

$$PLa = B_{CODECA} \times f_R \quad (2\text{-}1)$$

$$PLv = B_{CODECv} \times f_R \quad (2\text{-}2)$$

dengan:

PLa = *payload* paket audio (bit)

PLv = *payload* paket video (bit)

$B_{CODECA}$ = *bit rate* CODEC *audio* (bps)

$B_{CODECv}$ = *bit rate* CODEC *video* (bps)

$f_R$ = *frame rate* (s)

Kemudian, *payload* video *streaming* yang akan di-*encode* diubah menjadi paket-paket yang disebut NALU (*Network Abstraction Layer Unit*). Besarnya tiap paket audio dan video aplikasi video *streaming* merupakan penjumlahan dari *payload* paket audio dan video dengan *header* NALU, RTP, UDP, dan IP. Sehingga, besar tiap paket audio dan video *streaming* yang dihasilkan dapat dinyatakan dengan:

$$Pa = PLa/PLa_{max} \quad (2\text{-}3)$$

$$Pa\text{-}size = PLa + Pa \times (H_{NALU} + H_{RTP} + H_{UDP} + H_{IP}) \quad (2\text{-}4)$$

$$P_v = PLv/PLv_{max} \quad (2\text{-}5)$$

$$Pv\text{-}size = PLv + Pv \times (H_{NALU} + H_{RTP} + H_{UDP} + H_{IP}) \quad (2\text{-}6)$$

dimana:

$Pa$ = jumlah paket *audio*

$PLa_{max}$ = maksimum *payload* paket audio (bit)

$Pv$ = jumlah paket *video*

$PLv_{max}$ = maksimum *payload* paket video (bit)

$H_{NALU}$ = NALU *header* (8 bit)

Paket-paket video dan audio yang telah di-*encode* akan ditransmisikan melalui RTP (*Real Time Protocol*). Besar paket video *streaming* yang akan ditransmisikan dapat diperoleh menggunakan persamaan:

$$P_{vs\text{-}size} = Pa\text{-}size + Pv\text{-}size \quad (2\text{-}7)$$

dimana:

$P_{vs\text{-}size}$ = ukuran paket aplikasi *video streaming* (byte)

$Pa\text{-}size$ = besar paket *audio* (byte)

$Pv\text{-}size$ = besar paket *video* (byte)

$H_{RTP}$ = *header* RTP (12 byte)

$H_{IP}$ = panjang *header* IP (40 byte) [RFC 2460]

$H_{UDP}$ = panjang *header* UDP (8 byte) [Stalling, 1997:619]

### 2.5.2 *Bandwidth* Aplikasi Video *Streaming*

Dalam mentransmisikan data, jika kita akan mentransmisikan data analog melalui sebuah medium, kita perlu memperhatikan *bandwidth* analog, yaitu range frekuensi yang dapat dilewati oleh suatu medium (dinyatakan dalam Hertz). Sedangkan jika kita akan mentransmisikan data digital, kita perlu memperhatikan *bandwidth* digital, yaitu maksimum bit rate yang dapat dilewati oleh suatu medium medium (bps). [Behrouz A. Forouzan, 2004: 65]

*Bandwidth* aplikasi video *streaming* dimana merupakan data digital, dinyatakan dalam bps (bit per *second*). *Bandwidth* aplikasi video *streaming* adalah jumlah *bandwidth* audio ditambah dengan video. Sehingga besarnya *bandwidth* aplikasi video *streaming* dapat dinyatakan dengan persamaan: [Cisco System, 2009: 4]

$$B_{vs} = Ba + Bv$$
$$= \frac{Pa_{size}}{PLa}.B_{CODECa} + \frac{Pv_{size}}{PLv}.B_{CODECv} \quad (2\text{-}8)$$

dimana:

Ba = *bandwidth* audio (bps)

Bv = *bandwidth* video (bps)

$B_{vs}$ = *bandwidth* aplikasi video *streaming* (bps)

### 2.5.3 *Delay End-to-End* Aplikasi Video *Streaming*

*Delay end-to-end* aplikasi video *streaming* merupakan jumlah *delay* CODEC aplikasi video *streaming* dengan *delay* jaringan dimana aplikasi tersebut berjalan. *Delay* CODEC aplikasi video *streaming* terdiri dari *delay* CODEC audio dan video untuk menghasilkan satu paket video *streaming*. Besarnya *delay* CODEC aplikasi video *streaming* dapat dihitung dengan menggunakan persamaan:

$$t_{CODEC} = t_a + t_v \quad (2\text{-}9)$$

dengan:

$t_{CODEC}$ = *delay* CODEC aplikasi *video streaming* (detik)

$t_a$ = *delay* CODEC audio (detik)

$t_v$ = *delay* CODEC *video* (detik)

Sehingga, total *delay end-to-end* penerapan aplikasi *video streaming* pada jaringan HSDPA menggunakan IPv6 dapat dihitung dengan persamaan:

$$t_{end\text{-}to\text{-}end} = t_{CODEC} + t_{net} \quad (2\text{-}10)$$

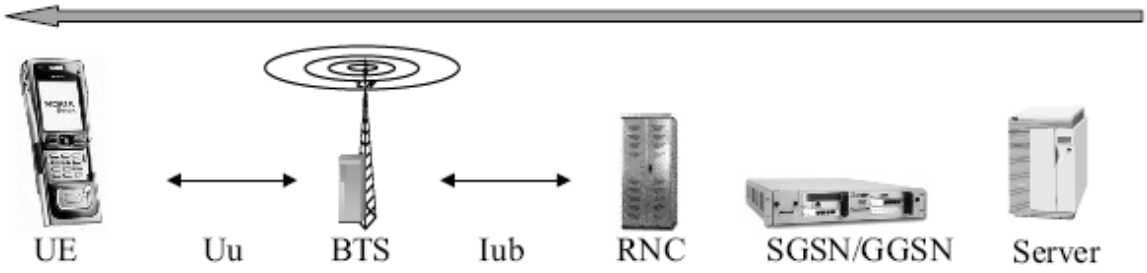
dengan:

$t_{total}$ = *delay end-to-end* untuk aplikasi *video streaming* (detik)

$t_{net}$ = delay jaringan HSDPA (detik)

#### 2.5.3.1 *Delay End-to-End* pada Jaringan HSDPA

*Delay* adalah waktu yang dibutuhkan untuk mengirimkan paket data dari sumber sampai ke tujuan. *Delay* pada jaringan HSDPA merupakan penjumlahan *delay-delay* yang ada dalam perjalanan paket dari *server* hingga ke UE (*User Equipement*). Besarnya *delay end-to-end* pada jaringan HSDPA untuk aplikasi video *streaming* dapat ditunjukkan dalam Gambar 2.13




**Gambar 2.13** *Delay End-to-End* pada HSDPA
(**Sumber**: Hari Holma, 2006: 154)

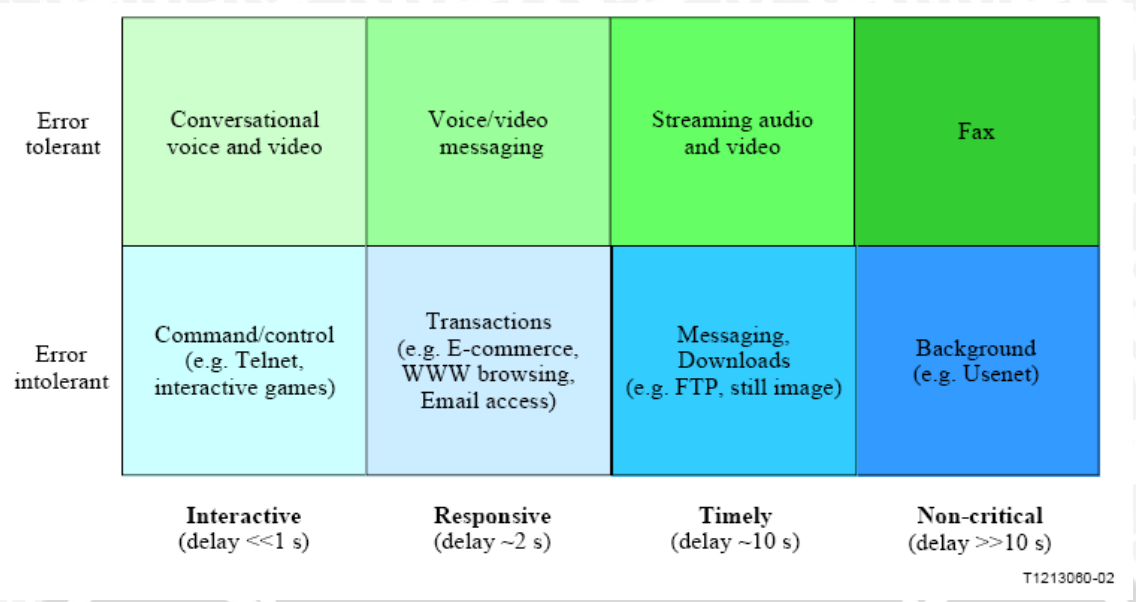
*Delay End-to-End* berdasarkan Gambar dihitung dengan menggunakan persamaan:

$$t_{net} = t_{proc} + t_t + t_p + t_w \quad (2\text{-}11)$$

dimana:

$t_{net}$ = *delay end-to-end* pada jaringan HSDPA (s)
$t_{proc}$ = *delay* proses (s)
$t_t$ = *delay* transmisi (s)
$t_p$ = *delay* propagasi (s)
$t_w$ = *delay* antrian (s)

Aplikasi video *streaming* merupakan salah satu teknologi telekomunikasi yang sangat sensitif terhadap *delay*. Sesuai dengan referensi International Telecommunications Union (ITU) G.110, batasan *delay* untuk aplikasi video *streaming* adalah < 10s seperti yang ditunjukkan pada Gambar 2.14.




**Gambar 2.14** Batasan *Delay* setiap Aplikasi Multimedia

(**Sumber**: ITU.T G.1010, 2001: 14)

### a. *Delay* Proses

*Delay* proses merupakan waktu yang dibutuhkan untuk memproses paket data dan untuk menentukan kemana data tersebut akan diteruskan. *Delay* proses pada jarngan HSDPA meliputi *delay* enkapsulasi dan *delay* dekapsulasi.

*Delay* enkapsulasi adalah waktu yang dibutuhkan untuk menambahkan keseluruhan *header* pada sebuah paket sehingga paket data tersebut dapat tepat sampai ke tujuan. Sedangkan *delay* dekapsulasi adalah waktu yang dibutuhkan untuk melepaskan keseluruhan *header* dari sebuah paket. Besarnya *delay* dekapsulasi dan *delay* enkapsulasi dapat diperoleh dengan menggunakan persamaan di bawah ini [Onno W. Purbo, et al., 2001: 24]:

$$t_{enc} = \frac{W_{frame} - L}{C} x8 \qquad (2\text{-}12)$$

$$t_{dec} = \frac{W_{frame} - L}{C} x8 \qquad (2\text{-}13)$$

dengan:

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

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

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

L = panjang paket data di *node* (byte)

C = kecepatan pemrosesan data (bps)

*Delay* enkapsulasi terjadi pada *server*, GGSN, SGSN, RNC, dan *Node* B. Besarnya *delay* enkapsulasi didapatkan dengan menggunakan persamaan:

$$t_{enc} = t_{E1} + t_{E2} + t_{E3} + t_{E4} + t_{E5} \quad (2\text{-}14)$$

dengan:

$t_{enc}$ = *delay* enkapsulasi total

$t_{E1}$ = *delay* enkapsulasi pada *server*

$t_{E2}$ = *delay* enkapsulasi pada GGSN

$t_{E3}$ = *delay* enkapsulasi pada SGSN

$t_{E4}$ = *delay* enkapsulasi pada RNC

$t_{E5}$ = *delay* enkapsulasi pada *Node* B

Sedangkan *delay* dekapsulasi terjadi di setiap *node* pada jaringan HSDPA, yaitu GGSN, SGSN, RNC, *Node* B, dan UE. Besarnya *delay* dekapsulasi dapat dihutung menggunakan persamaan:

$$t_{dec} = t_{D1} + t_{D2} + t_{D3} + t_{D4} + t_{D5} \quad (2\text{-}15)$$

dengan:

$t_{dec}$ = *delay* dekapsulasi total

$t_{D1}$ = *delay* dekapsulasi pada GGSN

$t_{D2}$ = *delay* dekapsulasi pada SGSN

$t_{D3}$ = *delay* dekapsulasi pada RNC

$t_{D4}$ = *delay* dekapsulasi pada *Node* B

$t_{D5}$ = *delay* dekapsulasi pada UE

Nilai *delay* proses pada jaringan HSDPA untuk aplikasi video *streaming* dapat diperoleh dengan menggunakan persamaan:

$$t_{proc} = t_{enc} + t_{dec} \quad (2\text{-}16)$$

dengan:

$t_{proc}$ = *delay* proses

$t_{enc}$ = *delay* enkapsulasi total

$t_{dec}$ = *delay* dekapsulasi total

Sedangkan, delay proses pada jaringan HSDPA yang berbasis IPv4 perlu ditambahkan *delay header checksum* sesuai dengan persamaan:

$$t_{checksum} = N_{packet\text{-}IP} \times T \quad (2\text{-}17)$$

dengan:

$t_{checksum}$ = *delay header checksum* (s)

$N_{packet-IP}$ = jumlah paket IP yang ditransmisikan

T = waktu pemrosesan setiap *header checksum* (s)

- ***Server***

Pada server terjadi proses enkapsulasi sebelum paket data ditransmisikan ke GGSN. Paket data video streaming diberi penambahan header RTP, UDP, IPv6 seperti dijelaskan sebelumnya. Apabila panjang paket video *streaming* melebihi *Maximum Transfer Unit* (MTU) *Ethernet* sebesar 1500 *byte*, maka paket video *streaming* akan mengalami fragmentasi menjadi beberapa buah frame sesuai dengan persamaan:

$$N_{frame\ Ethernet} = \frac{P_{VS-size}}{MTU_{ethernet}} \quad (2\text{-}18)$$

dengan:

$N_{frame\ Ethernet}$ = jumlah *frame Ethernet*

$MTU_{ethernet}$ = MTU *Ethernet* (1500 byte)

Sehingga jumlah total *frame* pada *server* yang dapat dikirimkan ke GGSN sesuai dengan persamaan:

$$W_{VS} = P_{VS\text{-}size} + [N_{frame\ Ethernet} \times (H_{ethernet} + FCS)] \quad (2\text{-}19)$$

Dalam skripsi ini, pada *server* digunakan standar *interface Gigabit Ethernet* dengan kecepatan 1 Gbps. Sehingga *delay* enkapsulasi pada *server* didapatkan dengan persamaaan berikut:

$$t_{E1} = \frac{W_{frame\ server} - P_{VS\text{-}size}}{C_{server}} x8 \quad (2\text{-}20)$$

dengan:

$t_{E1}$ = *delay* enkapsulasi pada *server* (s)

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

$C_{server}$ = kecepatan pemrosesan data pada *server* (bps)

- ***Gateway* GPRS *Support Node* (GGSN)**

Pada GGSN, paket data yang diterima dari *server* mengalami proses dekapsulasi. Kemudian, sebelum paket data aplikasi video *streaming* ditransmisikan ke SGSN, dilakukan proses enkapsulasi terlebih dahulu.

Pada proses dekapsulasi, paket data aplikasi video *streaming* didapatkan sesuai dengan persamaan:

$$W_{VS\ GGSN} = W_{frame\ server} - H_{UDP} - H_{IPv6} - H_{ethernet} - FCS \quad (2\text{-}21)$$

Dengan mengasumsikan GGSN menggunakan standar *Fast Ethernet* dengan kecepatan 100 Mbps, maka besarnya *delay* dekapsulasi pada GGSN didapatkan sesuai dengan persamaan:

$$t_{D1} = \frac{W_{frame\ server} - W_{VS\,GGSN}}{C_{GGSN}} x8 \quad (2\text{-}22)$$

dengan:

$t_{D1}$ = *delay* dekapsulasi pada GGSN (s)

$W_{VS\ GGSN}$ = panjang paket data video *streaming* di GGSN (byte)

$C_{GGSN}$ = kecpatan pemrosesan data di GGSN (bps)

Besar MSS dapat diperoleh sesuai dengan persamaan berikut:

$$MSS = MTU - H_{GTP} - H_{UDP} - H_{IP} \quad (2\text{-}23)$$

Selanjutnya, paket data yang melebihi MSS akan disegmentasi menggunakan persamaan berikut

$$N_{datagram} = \frac{W_{vs-GGSN}}{MSS} \quad (2\text{-}24)$$

Kemudian, paket data aplikasi video *streaming* dienkapsulasi dengan penambahan *header* GTP (GPRS *Tunneling Protocol*), UDP (*User Datagram Protocol*), dan IP (*Internet Protocol*) sesuai dengan persamaan berikut:

$$W_{datagram} = W_{VS\ GGSN} + N_{datagram} \times (H_{GTP} + H_{UDP} + H_{IPv6}) \quad (2\text{-}25)$$

Karena panjang datagram IP melebihi *Maximum Transfer Unit* (MTU) *Ethernet* (1500 *byte*), maka datagram IP akan mengalami fragmentasi menjadi beberapa buah frame sesuai dengan persamaan:

$$N_{frame\ Ethernet} = \frac{W_{datagram\ IP}}{MTU_{ethernet}} \quad (2\text{-}26)$$

Sehingga, jumlah total *frame* pada GGSN yang dapat dikirimkan ke SGSN sesuai dengan persamaan:

$$W_{\text{frame GGSN}} = W_{\text{datagram}} + [N_{\textit{frame}\ \text{Ethernet}} \text{ x } (H_{\text{ethernet}} + \text{FCS}) \quad (2\text{-}27)$$

Sehingga *delay* enkapsulasi pada GGSN didapatkan dengan persamaaan:

$$t_{E2} = \frac{W_{\textit{frame GGSN}} - W_{\text{VSGGSN}}}{C_{GGSN}} x8 \quad (2\text{-}28)$$

dengan:

$t_{E2}$ = *delay* enkapsulasi pada GGSN (s)

$W_{\text{frame GGSN}}$ = panjang *frame* di GGSN (byte)

$C_{GGSN}$ = kecepatan pemrosesan data di GGSN (bps)

- ***Serving GPRS Support Node* (SGSN)**

Pada SGSN, paket data yang diterima dari GGSN mengalami proses dekapsulasi. Kemudian, sebelum paket data aplikasi video *streaming* ditransmisikan ke RNC, dilakukan proses enkapsulasi terlebih dahulu.

Paket data yang diterima dari GGSN mengalami proses dekapsulasi pada SGSN sesuai dengan persamaan:

$$W_{\text{VS SGSN}} = W_{\textit{frame}\text{GGSN}} - H_{\text{GTP}} - H_{\text{UDP}} - H_{\text{IPv6}} - H_{\textit{ethernet}} - \text{FCS} \quad (2\text{-}29)$$

Dengan mengasumsikan bahwa SGSN menggunakan standar *Fast Ethernet* dengan kecepatan data 100 Mbps, maka besarnya *delay* dekapsulasi didapatkan dengan menggunakan persamaan:

$$t_{D2} = \frac{W_{\textit{frame}\text{GGSN}} - W_{\textit{VS SGSN}}}{C_{SGSN}} x8 \quad (2\text{-}30)$$

dengan:

$t_{D2}$ = *delay* dekapsulasi pada SGSN (s)

$W_{\text{VS SGSN}}$ = panjang paket data video *streaming* di SGSN (byte)

$Header_{\text{GTP}}$ = panjang *header* GTP (8 byte)

$C_{\text{GGSN}}$ = kecepatan pemrosesan data pada GGSN (bps)

Besar MSS dapat diperoleh sesuai dengan persamaan berikut:

$$\text{MSS} = \text{MTU} - H_{\text{GTP}} - H_{\text{UDP}} - H_{\text{IP}} \quad (2\text{-}31)$$

Selanjutnya, paket data yang melebihi MSS akan disegmentasi menggunakan persamaan berikut

(2-32)

$$N_{datagram} = \frac{W_{vs-GGSN}}{MSS}$$

Kemudian, paket data aplikasi video *streaming* ditambahkan dengan *header* GTP, UDP, dan IP.

$$W_{datagram} = W_{VS\ SGSN} + N_{datagram} \text{ x } (H_{GTP} + H_{UDP} + H_{IP}) \quad (2\text{-}33)$$

Setelah itu, saat memasuki lapisan AAL5, paket data tersebut dibentuk menjadi CPCS PDU (*Common Part Convergence Sublayer Protocol Data Unit*) kemudian dibentuk menjadi blok-blok PDU SAR (*Segmentation and Reassembly Sublayer*). Selanjutnya, besar paket data total pada CPCS-PDU diperoleh dengan menggunakan persamaan adalah sebagai berikut:

$$W_{CPCS\text{-}PDU\ total} = W_{datagram} + NCPCS_{\text{-}PDU} \text{ x } H_{CPCS\text{-}PDU} \quad (2\text{-}34)$$

Kemudian, *frame* CPCS-PDU dipecah menjadi blok-blok *payload* PDU SAR (*Segmentation and Reassembly Sublayer*) sebesar 48 *byte,* yang kemudian diteruskan menuju layer ATM. Sehingga banyaknya frame ATM yang terbentuk sesuai dengan persamaan berikut:

(2-35)

$$N_{frame\,ATM} = \frac{W_{frame\,CPCS-PDU}}{48byte}$$

dengan:

$W_{CPCS\text{-}PDU}$ = panjang *frame* CPCS-PDU (byte)

$H_{CPCS\text{-}PDU}$ = *header* CPCS-PDU (byte)

$N_{frame}$ ATM = jumlah *frame* ATM

Setiap *frame* ATM diberi *header* sebesar 5 *byte* sehinga panjang *frame* ATM menjadi 53 *byte.* Panjang *frame* yang siap ditransmisikan menuju GGSN merupakan panjang *frame* ATM total sesuai dengan persamaan:

$$W_{frame\ SGSN} = N_{frame\ ATM} \text{ x } W_{frame\ ATM} \quad (2\text{-}36)$$

Sehingga, *delay* enkapsulasi yang terjadi pada SGSN didapatkan dengan persamaaan:

$$t_{E3} = \frac{W_{frame\,SGSN} - W_{VSSGSN}}{C_{SGSN-RNC}} x8 \quad (2\text{-}37)$$

dengan:

$t_{E3}$ = *delay* enkapsulasi pada SGSN (s)

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

$C_{SGSN\text{-}RNC}$ = kecepatan pemrosesan data pada SGSN-RNC (bps)

- ***Radio Network Controller* (RNC)**

Pada RNC, paket data yang diterima dari SGSN mengalami proses dekapsulasi. Kemudian, sebelum paket data aplikasi video *streaming* ditransmisikan ke *Node* B, dilakukan proses enkapsulasi terlebih dahulu.

Paket data yang diterima dari SGSN mengalami proses dekapsulasi pada RNC sesuai dengan persamaan:

$$W_{VS\ RNC} = W_{frame\ SGSN} - H_{GTP} - H_{UDP} - H_{IPv6} - H_{AAL5} \quad (2\text{-}38)$$

Pada RNC, digunakan *interface* sistem transmisi ATM STM-1 (*Synchronous Transport Module*-1) yang memberikan kecepatan data sebesar 155,52 Mbps. Nilai *delay* dekapsulasi pada RNC didapatkan dengan persamaan:

$$t_{D3} = \frac{W_{frame\ SGSN} - W_{VS\ RNC}}{C_{RNC}} x8 \quad (2\text{-}39)$$

dengan:

$t_{D3}$ = *delay* dekapsulasi pada RNC (s)

$W_{VS\ RNC}$ = panjang paket data video *streaming* pada RNC (byte)

$C_{RNC}$ = kecepatan pemrosesan data pada RNC (bps)

Pada layer PDCP (*Packet Data Convergence Protocol*), paket datagram IP aplikasi video *streaming* dienkapsulasi sesuai dengan persamaan:

$$N_{datagram} = \frac{W_{vs-RNC}}{1500} \quad (2\text{-}40)$$

Besar frame total pada layer PDCP dapat dihitung sesuai dengan persamaan:

$$W_{frame\ PDCP} = W_{VS\ RNC} + H_{PDCP} \quad (2\text{-}41)$$

Selanjutnya pada layer RLC (*Radio Link Control*), frame PDCP disegmentasi menjadi RLC PDU *fixed size* sebesar 40 *byte*. Sehingga jumlah *frame* RLC sesuai dengan persamaan:

$$N_{frame\ RLC} = \frac{W_{frame\ PDCP}}{40\,byte} \quad (2\text{-}42)$$

Setiap *frame* selanjutnya diberi *header* sebesar 2 *byte*, sehingga panjang setiap *frame* RLC sebesar 42 byte. Panjang *frame* RLC total yang siap

diteruskan ke *layer* MAC-d (*Medium Acces Control-dedicated*) sesuai dengan persamaan berikut:

$$W_{frame\ RLC\ total} = N_{frame\ RLC} \times W_{frame\ RLC} \quad (2\text{-}43)$$

dengan:

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

$header_{PDCP}$ = panjang *header* PDCP (1 byte)

$N_{frame\ RLC}$ = jumlah frame pada RLC

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

$W_{frame\ RLC}$ = panjang *frame* RLC (42 byte)

Pada *layer* MAC-d, RLC PDU disegmentasi menjadi MAC-d SDU (*Service Data Unit*) sebesar 42 *byte*. Jumlah *frame* MAC-d SDU sesuai dengan persamaan:

$$N_{frame\ MAC-d} = \frac{W_{frame\ RLC total}}{42\,byte} \quad (2\text{-}44)$$

Sedangkan panjang *frame* MAC-d dihitung dengan persamaan:

$$W_{frame\ MAC\text{-}d} = H_{MAC\text{-}d} + 42\ byte \quad (2\text{-}45)$$

Kemudian, *frame* MAC-d dienkapsulasi pada HS-DSCH FP (*High Speed Downlink Shared Channel-Frame Protocol*) dengan penambahan *header* FP dan CRC (*Cyclic Redudancy Check*) sesuai dengan persamaan:

$$W_{frame\ FP} = W_{frame\ MAC\text{-}d} + H_{FP} + CRC \quad (2\text{-}46)$$

Panjang *frame* HS-DSCH FP total diproleh dengan persamaan:

$$W_{frame\ FP\ total} = N_{frame\ MAC\text{-}d} \times W_{frame\ FP} \quad (2\text{-}47)$$

dengan:

$N_{frame\ MAC-d}$ = jumlah *frame* MAC-d

$W_{frame\ MAC-d}$ = panjang *frame* MAC-d (byte)

$Header_{MAC-d}$ = panjang *header* MAC-d (3 byte)

$W_{frame\ FP}$ = panjang *frame* HSDSCH-FP (byte)

$Header_{FP}$ = panjang *header* HSDSCH-FP (7 byte)

CRC = *Cyclic Redudancy Check* (2 byte)

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

Saat memasuki lapisan AAL2, paket data tersebut dibentuk menjadi CPS (*Common Part Sublayer*) dan blok-blok PDU SAR. Banyaknya paket data pada CS PDU dapat diperoleh dengan persamaan berikut:

$$N_{CS-PDU} = \frac{W_{frame\,FP\,total}}{24} \quad (2\text{-}48)$$

Selanjutnya, besar paket data total pada CPCS-PDU diperoleh dengan menggunakan persamaan 2-50 sebagai berikut

$$W_{\text{CS-PDU total}} = W_{frame\ \text{FP total}} + (N_{\text{CPCS-PDU}} \text{ x } H_{\text{CPCS-PDU}}) \quad (2\text{-}49)$$

Kemudian, *frame* CS-PDU dipecah menjadi blok-blok *payload* PDU SAR (*Segmentation and Reassembly Sublayer*) sebesar 47 *byte*

$$N_{SAR-PDU} = \frac{W_{frame\,CS-PDU}}{47\,byte} \quad (2\text{-}50)$$

Selanjutnya, setiap blok PDU SAR ditambahkan header sebesar 1 byte sehingga besar setiap blok PDU SAR menjadi 48 byte. Besar paket data pada PDU SAR dapat diperoleh menggunakan persamaan adalah sebagai berikut:

$$W_{\text{frame PDU SAR}} = N_{\text{SAR-PDU}} \text{ x } 48 \text{ byte} \quad (2\text{-}51)$$

Pada layer ATM, *frame* AAL2 dipecah menjadi *fixed cell* ATM sebesar 48 *byte*. Sehingga banyaknya frame ATM yang terbentuk sesuai dengan persamaan:

$$N_{frame\,ATM} = \frac{W_{frame\,PDU-SAR}}{48\,byte} \quad (2\text{-}52)$$

dengan:

$W_{\text{frame AAL2}}$ = panjang *frame* AAL2 (byte)

$Header_{\text{AAL2}}$ = panjang *header* AAL2 (3 byte)

$N_{\text{frame ATM}}$ = jumlah *frame* ATM pada RNC

Setiap *frame* ATM diberi *header* sebesar 5 *byte.* Panjang *frame* yang siap ditransmisikan menuju GGSN merupakan panjang *frame* ATM total sesuai dengan persamaan:

$$W_{\text{frame RNC}} = N_{frame\ \text{ATM}} \text{ x } W_{frame\ \text{ATM}} \quad (2\text{-}53)$$

Sehingga, besar *delay* dekapsulasi yang terjadi pada RNC didapatkan dengan persamaaan berikut:

$$t_{E4} = \frac{W_{frame\ RNC} - W_{\text{VSRNC}}}{C_{RNC}} x8 \quad (2\text{-}54)$$

dengan:

$t_{E4}$ = *delay* enkapsulasi pad RNC (s)

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

$C_{RNC}$ = kecepatan pemrosesan data pada RNC (bps)

- ***Node* B**

Pada *Node* B, paket data yang diterima dari RNC mengalami proses dekapsulasi. Kemudian, sebelum paket data aplikasi video *streaming* ditransmisikan ke UE, dilakukan proses enkapsulasi terlebih dahulu.

Paket data yang diterima dari RNC mengalami proses dekapsulasi pada *Node* B sesuai dengan persamaan:

$$W_{VS\ NodeB} = W_{frame\ RNC} - H_{ATM} - H_{AAL2} - H_{FP} - CRC \quad (2\text{-}55)$$

*Node* B menggunakan *interface* STM-1 dengan kecepatan 155,52 Mbps. Maka besarnya *delay* dekapsulasi didapatkan dengan menggunakan persamaan:

$$t_{D4} = \frac{W_{frame\ RNC} - W_{VS\ Node\ B}}{C_{Node\ B}} x8 \quad (2\text{-}56)$$

dengan:

$t_{D4}$ = *delay* dekapsulasi pada *Node* B (s)

$W_{VS\ NodeB}$ = panjang paket data video *streaming* di *Node* B (byte)

$C_{Node\ B}$ = kecepatan pemrosesan data pada *Node* B (bps)

Pada layer MAC-hs, beberapa frame MAC PDU dan header MAC-hs dibentuk menjadi satu TBS (*transport block size*) sebesar 7298 bits [HSDPA by Siemens, 2004: 17]. Banyaknya TBS yang akan ditransmisikan menuju UE adalah sebagai berikut:

$$N_{TBS} = \frac{W_{VS\ Node\ B}}{7298 bit} x8 \quad (2\text{-}57)$$

Selanjutnya, setiap TBS ditambahkan header CRC sebesar 2 byte. Panjang *frame* yang siap ditransmisikan menuju UE sesuai dengan persamaan:

$$W_{frame\ Node\ B} = W_{VSNodeB} + N_{TBS} \text{ x } CRC \quad (2\text{-}58)$$

dengan:

$N_{TBS}$ = jumlah TBS

$W_{frame\ NodeB}$ = panjang *frame* MAC-hs (byte)

*CRC* = *Cyclic Redudancy Check* (2 byte)

Sehingga, *delay* enkapsulasi yang terjadi pada SGSN didapatkan dengan persamaaan berikut:

$$t_{E5} = \frac{W_{frame\ Node\ B} - W_{VS Node B}}{C_{Node\ B}} x8 \quad (2\text{-}59)$$

dengan:

$t_{E5}$ = *delay* enkapsulasi pada *Node* B (s)

$W_{frame\ Node\ B}$ = panjang *frame* pada *Node* B (byte)

$C_{Node\ B}$ = kecepatan pemrosesan data pada *Node* B (bps)

- ***User Equipment*** **(UE)**

Pada UE, paket data yang diterima dari *Node* B mengalami proses dekapsulasi. Proses dekapsulasi pada UE sesuai dengan persamaan:

$$W_{VS\ UE} = W_{frame\ Node\ B} - H_{RTP} - H_{UDP} - H_{IP} - H_{PDCP} - H_{RLC} - H_{MAC\text{-}d} - H_{MAC\text{-}hs} - CRC \quad (2\text{-}60)$$

Dengan mengasumsikan kategori UE adalah kategori 5 sesuai Tabel 2.1 dengan *data rate* maksimum sebesar 3,6 Mbps, maka besarnya *delay* dekapsulasi yang terjadi di UE didapatkan dengan persamaan:

$$t_{D5} = \frac{W_{frame\ Node\ B} - W_{VS\ UE}}{C_{UE}} x8 \quad (2\text{-}61)$$

dengan:

$t_{D5}$ = *delay* dekapsulasi pada UE (s)

$W_{VS\ UE}$ = panjang paket data video *streaming* pada UE (byte)

$C_{UE}$ = kecepatan pemrosesan data pada UE (bps)

**b. *Delay* Transmisi**

*Delay* transmisi adalah waktu yang dibutuhkan untuk meletakkan sebuah paket data ke media transmisi. Dipengaruhi ukuran paket data dan kecepatan transmisi. *Delay* transmisi dapat ditentukan dengan persamaan [Mischa Schwartz, 1987 : 132]:

$$t_T = \frac{W}{C} \quad (2\text{-}62)$$

Sehingga *delay* transmisi total ditentukan dengan persamaan:

$$t_{T\,tot} = t_{T1} + t_{T2} + t_{T3} + t_{T4} \quad (2\text{-}63)$$

dimana:

$t_T$ tot = *delay* transmisi total (s)

W = panjang *frame* pada *node* (bit)

C = kecepatan transmisi (bps)

$t_{T1}$ = *delay* transmisi pada GGSN-SGSN (s)

$t_{T2}$ = *delay* transmisi pada SGSN-RNC (s)

$t_{T3}$ = *delay* transmisi pada RNC-*Node* B(s)

$t_{T4}$ = *delay* transmisi pada *Node* B-UE (s)

**c. *Delay* Propagasi**

*Delay* propagasi adalah waktu yang dibutuhkan untuk merambatkan paket data melalui media transmisi dari SGSN ke UE. *Delay* propagasi ditentukan dengan persamaan [Forouzan, Behrouz A, 2000 : 215]:

$$t_P = \frac{N_{frame}\ x\ R}{v} \quad (2\text{-}64)$$

Sehingga *delay* propagasi total ditentukan dengan persamaan:

$$t_{p\,tot} = t_{p1} + t_{p2} + t_{p3} + t_{p4} \quad (2\text{-}65)$$

dimana:

$t_{p\,tot}$ = *delay* propagasi (s)

$N_{frame}$ = jumlah *frame* pada node

R = jarak antar *node* (m)

v = cepat rambat gelombang di cahaya = $3 \times 10^8$ (m/s)

$t_{p1}$ = *delay* propagasi pada GGSN-SGSN (s)

$t_{p2}$ = *delay* propagasi pada SGSN-RNC (s)

$t_{p3}$ = *delay* propagasi pada RNC-BTS(s)

$t_{p4}$ = *delay* propagasi pada BTS-UE (s)

Berikut ini merupakan referensi jarak antar *node* pada jaringan HSDPA ditunjukkan pada Tabel 2.5.

**Tabel 2.5** Jarak tiap *node* pada HSDPA

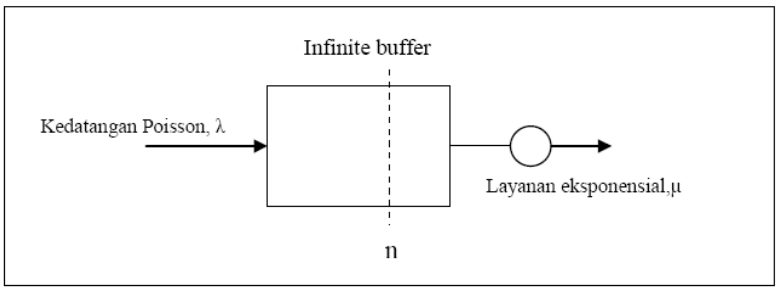
| Node | Jarak (m) |
|---|---|
| GGSN – SGSN | 2000 |
| SGSN – RNC | 3000 |
| RNC – Node B | 400 |

(**Sumber**: PT. Indosat, Tbk KPI Surabaya dalam Linda Ekowati, 2008:68)

**d. *Delay Antrian***

*Delay* antrian adalah waktu dimana paket data berada dalam antrian untuk ditransmisikan. Selama waktu ini, paket data menunggu hingga paket yang lain selesai ditransmisikan. *Delay* antrian dapat dihitung dengan menggunakan model antrian M/M/1. M pertama menunjukkan kedatangan *Poisson*, M kedua berarti distribusi waktu pelayanan eksponensial, dan 1 menunjukkan 1 menunjukkan jumlah server yang akan melayani pelanggan.

Disiplin antrian yang digunakan adalah FIFO (*First In First Out*). Bentuk model antrian M/M/1 dapat dilihat pada Gambar 2.15.




**Gambar 2.15** Model antrian M/M/1
**(Sumber**: Mischa Schwartz, 1987:31)

*Delay* antrian terjadi pada setiap *node* pada jaringan HSDPA, yaitu GGSN, SGSN, Node B, dan RNC. Besarnya *delay* antrian pada *node* ditentukan dengan persamaan [Mischa Schwartz, 1987 : 42]:

$$t_w = t_{queue} + t_{serv} \qquad (2\text{-}66)$$

dengan:

$t_w$ = *delay* antrian (s)

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

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

Perhitungan waktu rata-rata pelayanan antrian data di *node* ($t_{serv}$) dihitung dengan menggunakan persamaan [Mischa Schwartz, 1987 : 23]:

$$t_{serv} = \frac{L}{C} \quad (2\text{-}67)$$

dengan:

$\mu$ = kecepatan pelayanan *node* (bps)

L = panjang paket data di *node* (bit)

C = kecepatan transmisi pada *node* (bps)

Sedangkan waktu tunggu paket pada *router* ($t_{queue}$) dihitung dengan menggunakan persamaan [Mischa Schwartz, 1987 : 42]:

$$t_{queue} = \frac{\lambda}{\mu^2(1-\rho)} \quad (2\text{-}68)$$

Untuk performansi sistem antrian, ditunjukkan dalam bentuk $\rho$ (faktor utilisasi), yang nilainya diasumsikan berubah-ubah dengan kenaikan tertentu. Faktor utilisasi bernilai lebih besar dari 0 dan lebih kecil dari 1 dengan kenaikan sebesar 0,1. Hal ini disebabkan karena jika $\rho > 1$ berarti rata–rata kedatangan melampaui rata–rata pelayanan atau server tidak dapat menjaga rata–rata kedatangan dan panjang antrian yang bertambah tanpa batas. Besarnya faktor utilisasi dapat dihitung dengan menggunakan persamaan:

$$\rho = \frac{\lambda}{\mu} \quad \Rightarrow \quad \lambda = \mu\rho \quad (2\text{-}69)$$

Sehingga dari persamaan 2.61 dan 2.62 *delay* antrian dapat ditentukan dengan persamaan:

$$t_w = \frac{\lambda}{\mu.(\mu-\lambda)} + \frac{1}{\mu} \quad (2\text{-}70)$$

dengan:

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

$\lambda$ = kecepatan kedatangan paket pada *node* (bps)

$\rho$ = faktor utilitas ($0 < \rho < 1$)

$\mu$ = kecepatan pelayanan *node* (bps)

Sehingga *delay* antrian total ditentukan dengan persamaan:

$$t_{w\,tot} = t_{w1} + t_{w2} + t_{w3} + t_{w4} \quad (2\text{-}71)$$

dengan:

$t_{w\,tot}$ = *delay* antrian total (s)

$t_{w1}$ = *delay* antrian pada GGSN (s)

$t_{w2}$ = *delay* antrian pada SGSN (s)

$t_{w3}$ = *delay* antrian pada RNC (s)

$t_{w4}$ = *delay* antrian pada *Node* B (s)

### 2.5.4 Probabilitas *Packet Loss* pada Aplikasi *Video Streaming*

Paket *loss* adalah jumlah paket yang hilang dibandingkan dengan paket yang diterima benar. Umumnya perangkat jaringan memiliki *buffer* untuk menampung data yang diterima. Jika terjadi tabrakan yang cukup banyak, *buffer* akan penuh, dan data baru tidak dapat diterima. Paket yang hilang ini harus ditransmisi ulang, yang akan membutuhkan waktu tambahan. Prosentase *packet loss* maksimum yang diperbolehkan oleh ITU.T G.1010 untuk aplikasi video *streaming* adalah < 1%. Apabila packet loss yang dihasilkan melebihi 1%, kualitas video *streaming* akan buruk.

Probabilitas *packet loss* total penerapan aplikasi v*ideo streaming* pada suatu jaringan, ditentukan berdasarkan pada probabilitas *packet loss* pada jaringan HSDPA dan server. Probabilitas *packet loss* aplikasi *video streaming* yang berbasis protokol UDP/RTP/IP seperti pada persamaan

$$\rho_{tot} = 1 - [(1 - \rho_{network})(1 - \rho_{VS})] \qquad (2\text{-}72)$$

dimana:

$\rho_{tot}$ = probabilitas *packet loss* video *streaming*

$\rho_{network}$ = probabilitas *packet loss* pada jaringan HSDPA

$\rho_{VS}$ = probabilitas *packet loss* pada *server*

Prosentase *packet loss* ditentukan dengan Persamaan 2-72 (Wijaya Hendra A, 2005 : 34) :

$$packet\ loss(\%) = \frac{N_{packet\,loss}}{N_{paket} + N_{packet\,loss}} x100\,\% \qquad (2\text{-}73)$$

dengan :

$N_{packet\ loss}$ = jumlah paket yang hilang

$N_{paket}$ = jumlah paket yang diterima dengan benar

### 2.5.4.1 Probabilitas Packet Loss pada Server

Protokol UDP/RTP/IPv6 sendiri tidak dapat memberikan jaminan bahwa paket akan dikirimkan semua sesuai dengan permintaan. *Packet loss* yang terjadi pada server di *layer transport* dan *layer network* dapat dihitung dengan persamaan:

$$\rho_{VS} = P_{VS\text{-}size} \times \rho_b \quad (2\text{-}74)$$

dimana:

$\rho_{VS}$ = probabilitas *packet loss* pada server

$\rho_b$ = BER ($10^{-8}$)

$P_{VS\text{-}size}$ = panjang paket data video *streaming* (byte)

### 2.5.4.2 Probabilitas Packet Loss pada Jaringan HSDPA

Probabilitas packet loss pada Jaringan HSDPA dapat diperoleh dari probabilitas bit error rate (BER) di jaringan tersebut, sesuai dengan persamaan:

$$\rho_{HSDPA} = P_{VStot} \times \rho_{b\text{-}HSDPA} \quad (2\text{-}75)$$

dimana:

$\rho_{VS}$ = probabilitas *packet loss* pada jaringan HSDPA

$\rho_b$ = BER HSDPA pada saat transmisi (tanpa satuan)

$P_{VS\text{-}size}$ = panjang paket video *streaming* yang ditransmisikan pada jaringan HSDPA (byte)

BER (*bit error rate*) atau dengan sebutan lain probabilitas *error* bit merupakan nilai ukur kualitas sinyal yang diterima untuk sistem transmisi data digital. BER juga dapat didefinisikan sebagai perbandingan jumlah bit *error* terhadap total bit yang diterima. Nilai BER untuk modulasi 16QAM pada jaringan HSDPA dapat dihitung dengan persamaan [Andrea Goldsmith, 2005:167]:

$$P_b = \frac{4\left(\sqrt{M}-1\right)}{\sqrt{M}.\log_2 M}.Q.\left(\sqrt{\frac{3\frac{E_b}{N_o}\log_2 M}{M-1}}\right) \quad (2\text{-}76)$$

dengan:

$P_b$ = BER HSDPA pada saat transmisi (tanpa satuan)

M = ukuran konstelasi sinyal

$\frac{Eb}{No}$ = rasio energy bit terhadap kerapatan noise pada saat transmisi (dB)

Nilai M dipengaruhi oleh jenis modulasi yang digunakan. Untuk modulasi untuk modulasi 16-QAM dimana jumlah bit dalam 1 simbol adalah 4 bit, nilai M = $2^4$ = 16. Sedangkan, nilai Q dapat diperoleh dengan distribusi Gaussian menggunakan persamaan [John G. Proakis, 2000: 40]:

$$Q(x) = \frac{1}{2} erfc\left(\frac{x}{\sqrt{2}}\right) \quad (2\text{-}77)$$

$$erfc(x) \approx \frac{1}{\sqrt{\pi x}} . e^{-x^2} \quad (2\text{-}78)$$

dengan:

*erfc : error function complementary*

Eb/No adalah suatu parameter yang berhubungan dengan SNR yang biasanya digunakan untuk menentukan laju data digital dan sebagai ukuran mutu standar untuk kinerja sistem komunikasi digital. Hubungan SNR dengan Eb/No ditunjukkan dalam persamaan:

$$\frac{Eb}{No}(dB) = \frac{S}{N} - 10\log\frac{B}{R} \quad (2.79)$$

dengan :

PG = *processing gain* (dB)

Besarnya pengaruh redaman sinyal terhadap sinyal yang ditransmisikan dapat dinyatakan dengan perbandingan antara sinyal dengan noise (SNR) yang dinyatakan dalam persamaan berikut [E. Glatz, 1999]:

$$SNR_{(dB)} = P_r(dBm) - N_o(dBm) \quad (2\text{-}80)$$

dengan :

*SNR* = *signal to noise ratio* (dB)

$P_r$ = daya yang diterima (dBm)

$N_o$ = daya *noise* saluran transmisi (dBm)

Sedangkan untuk perhitungan daya noise dinyatakan dalam persamaan berikut [E. Glatz, 1999]:

$$N_{o(dBm)} = 10\log k \cdot T + 10\log B + NF \quad (2\text{-}81)$$

dengan :

$N_o$ = daya *noise* saluran transmisi (dBm)

*k* = konstanta Boltzman (1,38 x $10^{-23}$ J/K)

*T* = suhu *absolute* (300º K)

*NF* = *noise figure* (11,2 dB)

*B* = *bandwidth* (Hz)

Daya yang diterima oleh penerima sangat dipengaruhi oleh propagasi sinyal dari pemancar ke penerima. Sehingga daya yang diterima dapat dinyatakan dalam persamaan berikut [Robert G. Winch, 1998: 184] :

$$P_r(dBm) = P_t - FSL - L_t - L_r + G_r + G_t \quad (2\text{-}82)$$

Sedangkan nilai FSL (*Free Space Loss*) dapat diperoleh menggunakan persamaan berikut [Andrea Goldsmith, 2005: 49]

$$FSL = -20\log\left(\frac{\lambda}{4\pi d}\right) \quad (2\text{-}83)$$

dengan:

$P_r$ : daya terima *receiver* (dBm)

$P_t$ : daya pancar *transmitter* (dBm)

FSL : *free space loss* (dB)

$L_t$ : *transmitter losses* (*cable loss*) (dB)

$L_r$ : *receiver losses* (*body loss*) (dB)

$G_r$ : *gain receiver* (dBi)

$G_t$ : *gain transmitter* (dBi)

λ : Panjang gelombang (m)

*f* : Frekuensi kerja sistem (Hz)

d : jarak antara pemancar dan penerima (m)

*c* : kecepatan cahaya (3x$10^8$ m/s)

### 2.5.5 *Throughput*

*Throughput* adalah kecepatan maksimum jaringan saat tidak ada data yang hilang pada pentransmisiannya atau banyaknya data yang diterima dengan benar oleh *user*. *Throughput* yang mungkin didapat dengan memperhatikan probabilitas paket diterima dalam keadaan salah dapat dihitung dengan persamaan [Mischa Schwartz, 1987:129]

$$T = \frac{1}{t_v} = \frac{(1-\rho_{tot})}{t_l[1+(\alpha-1)\rho_{tot}]} \quad (2\text{-}84)$$

dimana:

T = *throughput* (bps)

$t_v$ = waktu rata-rata transmisi untuk mengirimkan paket yang benar (s)

$t_1$ = waktu transmisi sebuah paket data atau *frame* (s)

$\rho_{tot}$ = probabilitas frame yang salah

$\alpha$ = konstanta

Sedangkan parameter $\alpha$ dapat dihitung dengan persamaan [Mischa Schwartz, 1987:129]:

$$t_T = t_l + t_{out} \quad (2\text{-}85)$$

dan:

$$\alpha = \frac{t_T}{t_l} = 1 + \frac{t_{out}}{t_l} \quad (2\text{-}86)$$

dengan:

$t_T$ = waktu total yang dibutuhkan untuk mentransmisikan sebuah paket (s)

$t_I$ = waktu yang dibutuhkan untuk mentransmisikan sebuah paket (s)

$t_{out}$ = waktu yang dibutuhkan untuk menerima *acknowledge* atau interval waktu antara pengiriman sebuah paket dengan pengiriman paket selanjutnya/ *fixed timed out interval* (s)

Waktu transmisi *frame* ($t_l$) ditentukan dengan persamaan [Mischa Schwartz, 1987:132]:

$$t_l = \frac{PL_{frame} + H_{frame} \quad x8}{C_{trans}} \quad (2\text{-}87)$$

dengan:

$PL_{frame}$ = *payload frame* (byte)

$H_{frame}$ = *header frame* (byte)

$C_{trans}$ = kecepatan transmisi node (bps)

Sedangkan waktu yang dibutuhkan untuk menerima *acknowledge* dapat dihitung dengan menggunakan persamaan [Mischa Schwartz, 1987:129]:

$$t_{out} = 2t_p + 2t_I + t_{pro} \quad (2\text{-}88)$$

dengan:

$t_{out}$ = waktu yang dibutuhkan untuk menerima *acknowledge/fixed timed out interval* (s)

$t_p$ = *delay* propagasi untuk satu *frame* (s)

$t_{pro}$ = *delay* proses total untuk satu *frame* (s)

Nilai dari $t_{pro}$ dan $t_P$ dapat dihitung sesuai dengan persamaan:

$$t_{pro} = \frac{t_{proc}}{N_{frame}} + \frac{t_{wtotal}}{N_{frame}} \quad (2\text{-}89)$$

$$t_P = \frac{t_{ptotal}}{N_{frame}} \quad (2\text{-}90)$$

dengan:

$t_{proc}$ = *delay* proses (s)

$t_{W\ total}$ = *delay* antrian total (s)

$t_{ptotal}$ = *delay* propagasi total (s)

$N_{frame}$ = jumlah frame

# BAB III
# METODOLOGI PENELITIAN

Kajian yang digunakan dalam penelitian ini adalah kajian yang bersifat analisis, yaitu analisis terhadap performansi video *streaming* pada jaringan HSDPA (*High Speed Downlink Packet Access*) berbasis IPv6, yang mengacu pada studi literatur. Tahapan dalam penelitian ini terdiri atas: jenis data dan cara mendapatkan data, variabel dan cara analisis data yang digunakan, dan rangka keseluruhan proses penyelesaian permasalahan yang telah dirumuskan dan ditelaah dari aspek tertentu, dan sesuai dengan tujuan yang telah ditetapkan dalam bentuk *flowchart*. Analisis yang dilakukan ditinjau dari parameter *bandwidth, delay end-to-end*, probabilitas *packet loss* dan *throughput*.

## 3.1 Jenis dan Cara Pengambilan Data

Data-data yang digunakan dalam penelitian ini berupa data sekunder yang bersumber dari buku referensi, jurnal, skripsi, internet, dan forum-forum resmi mengenai jaringan HSDPA (*High Speed Downlink Packet Access*), *video streaming*, dan IPv6.

Cara untuk mendapatkan data sekunder pada penelitian ini dilakukan dengan cara perhitungan secara teori. Perhitungan teori ini meliputi: perhitungan konsumsi *bandwidth*, probabilitas *packet loss*, *delay end-to-end*, dan *throughput*. Untuk perhitungan teori menggunakan data sekunder yang bersumber dari buku referensi, jurnal, penelitian, internet, dan forum-forum resmi mengenai jaringan HSDPA (*High Speed Downlink Packet Access*) dan Video *Streaming* Data-data yang digunakan dalam penelitian ini meliputi:

a. Spesifikasi jaringan HSDPA (*High Speed Downlink Packet Access*).
b. Spesifikasi video *streaming*

## 3.2 Variabel dan Cara Analisis Data

### 3.2.1 Variabel Data

Varibel data yang digunakan pada skripsi ini sesuai dengan parameter performansi video *streaming* yang dianalisis. Paramater performansi video streaming pada Jaringan HSDPA menggunakan IPv6 yang dianalisis adalah sebagai berikut:

1. *Bandwidth*

   *Bandwidth* adalah maksimum bit rate yang dapat dilewati oleh suatu medium. Variabel data yang digunakan pada analisis bandwidth adalah bit rate CODEC audio, bit rate CODEC video, dan frame rate.

2. *Delay*

   *Delay* adalah waktu yang dibutuhkan untuk mengirimkan data dari sumber ke tujuannya. Variabel data yang digunakan pada analisis *delay end-to-end* adalah kecepatan pemrosesan data setiap node, jarak antar node, faktor utilitas, dan jenis CODEC yang digunakan.

3. Probabilitas *Packet Loss*

   *Packet Loss* adalah hilangnya paket data yang dikirimkan dari sumber. Variabel data yang digunakan pada analisis probabilitas *packet loss* adalah bit rate CODEC audio, bit rate CODEC video, dan *downlink link budge*t pada jaringan HSDPA.

4. *Throughput*

   *Throughput* merupakan parameter yang digunakan untuk mengetahui jumlah data yang diterima dalam keadaan baik terhadap waktu total transmisi yang dibutuhkan dari sumber data ke penerima. Variabel data yang digunakan pada analisis *throughput* adalah bit rate CODEC audio, bit rate CODEC video, frame rate, *downlink link budget* pada jaringan HSDPA, dan faktor utilitas.

### 3.2.2 Cara Analisis Data

Cara analisis data yang digunakan dalam memperoleh perfomansi *video streaming* pada jaringan HSDPA (*High Speed Downlink Packet Access*) berbasis IPv6 adalah sebagai berikut:

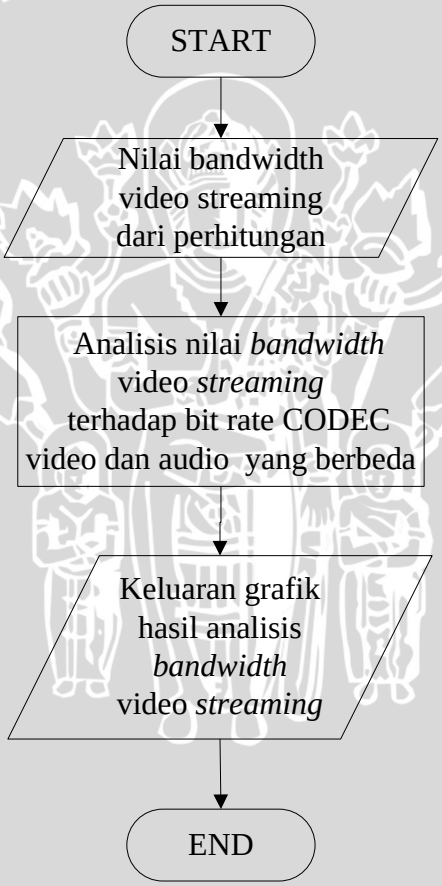
1. Analisis secara matematis yaitu mengumpulkan nilai dari parameter data-data sekunder dengan kesesuian terhadap standar yang digunakan untuk kemudian dianalisis berdasarkan pembahasan yang telah diuraikan menggunakan software matlab 7. Analisis perhitungan yang dilakukan meliputi: konsumsi *bandwidth*, probabilitas *packet loss, delay end-to-end,* dan *throughput*.
2. Analisis secara grafis yaitu melakukan simulasi hasil perhitungan ke dalam bentuk grafis sehingga dapat diketahui karakteristik sistem yang diterapkan.

Langkah-langkah analisis performansi *video streaming* pada jaringan HSDPA (*High Speed Downlink Packet Access*) yang menggunakan IPv6 adalah sebagai berikut:

1. Analisis *Bandwidth* Video Streaming

   Langkah-langkah analisis *bandwidth* video *streaming* adalah sebagai berikut:

   a. Mendapatkan nilai *bandwidth* video *streaming* dari hasil perhitungan yang dijelaskan pada subbab selanjutnya.
   b. Menganalisis besarnya nilai *bandwidth* video *streaming* terhadap *bit rate* CODEC video maupun audio yang berbeda-beda.
   c. Menampilkan grafik hasil analisis *bandwidth* video *streaming.*




**Gambar 3.1** Diagram Alir Proses Analisis *Bandwidth* Video *Streaming*

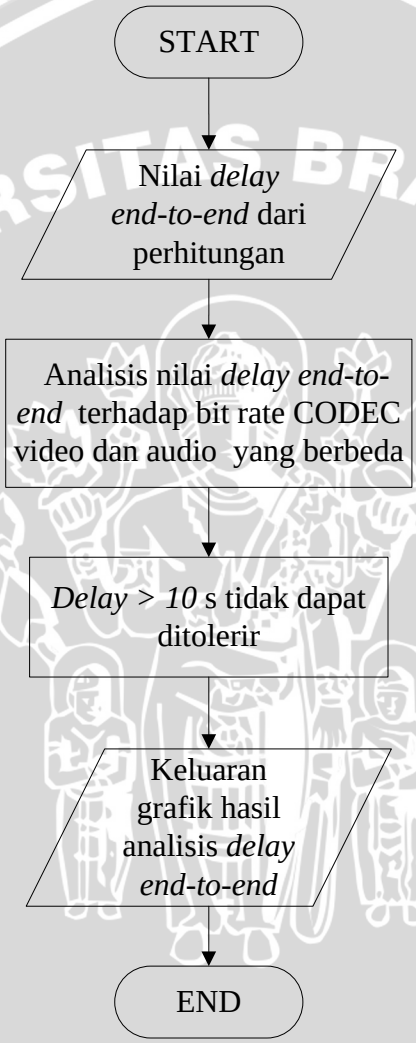
(**Sumber**: Perancangan)

2. Analisis *Delay End-to-end*

   Langkah-langkah analisis *delay end-to-end* adalah sebagai berikut:

   a. Mendapatkan nilai *delay end-to-end* dari hasil perhitungan yang dijelaskan pada subbab selanjutnya.

b. Menganalisis besarnya *delay end-to-end* terhadap *bit rate* CODEC video dan audio yang berbeda-beda.
c. Menganalisis apakah nilai *delay end-to-end* loss masih dapat ditolerir atau tidak, yaitu < 10 s (ITU.T G.1010).
d. Menampilkan grafik hasil analisis *delay end-to-end*.




**Gambar 3.2** Diagram Alir Proses Analisis *Delay End-to-end*

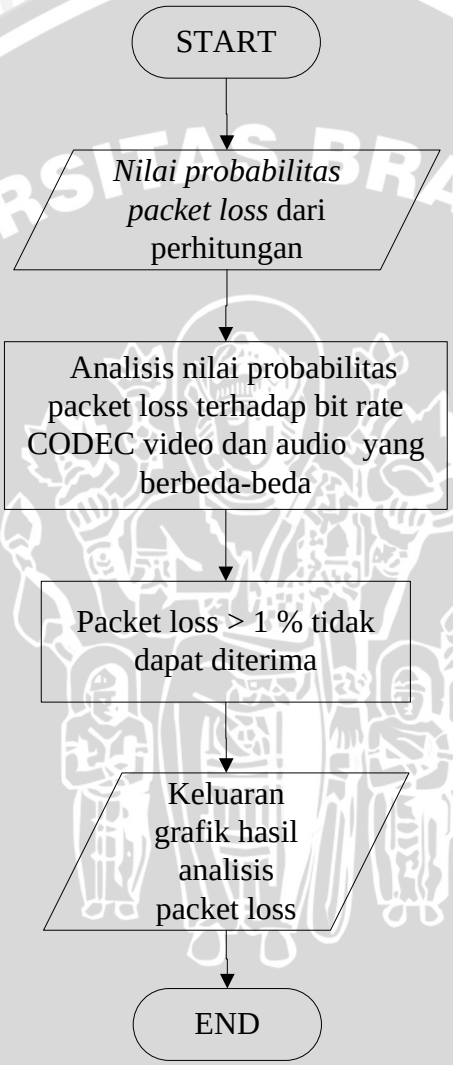
(**Sumber**: Perancangan)

3. Analisis Probabilitas *Packet Loss*

Langkah-langkah analisis probabilitas *packet loss* adalah sebagai berikut:

a. Mendapatkan nilai probabilitas *packet loss* dari hasil perhitungan yang dijelaskan pada subbab selanjutnya.

b. Menganalisis besarnya probabilitas *packet loss* terhadap *bit rate* CODEC video dan audio yang berbeda-beda.
c. Menganalisis apakah nilai probabilitas packet loss masih dapat diterima, yaitu <1% (ITU.T G.1010).
d. Menampilkan grafik hasil analisis packet loss.




**Gambar 3.3** Diagram Alir Proses Analisis Probabilitas *Packet Loss*

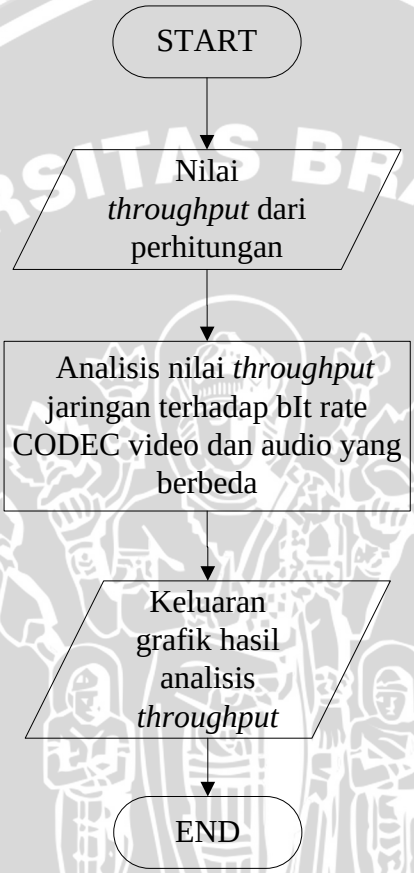
(**Sumber**: Perancangan)

4. Analisis *Throughput*

Langkah-langkah analisis *throughput* adalah sebagai berikut:

a. Mendapatkan nilai *throughput* video *streaming* dari hasil perhitungan yang dijelaskan pada subbab selanjutnya.

b. Menganalisis besarnya *throughput* video *streaming* terhadap *bit rate* CODEC video dan audio yang berbeda-beda.
c. Menganalisis apakah besarnya *throughput* antara 20-384 Kbps (RSAVY Research and 3G Americas, 2007: 52).
d. Menampilkan grafik hasil analisis *bandwidth* video *streaming*.

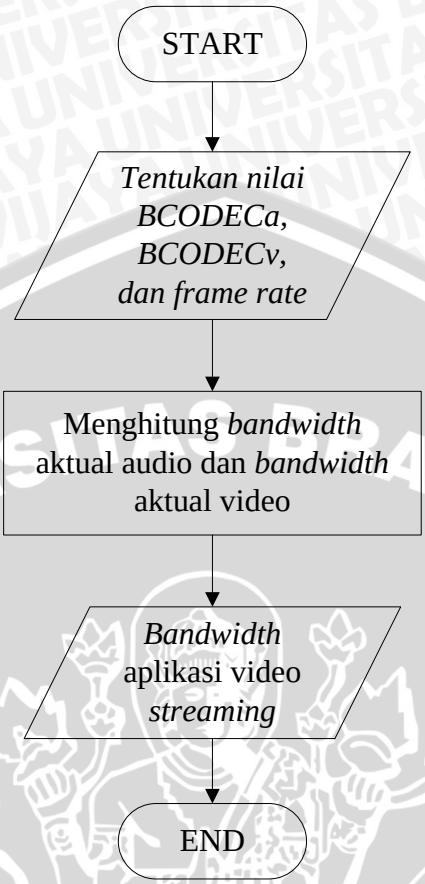



**Gambar 3.4** Diagram Alir Proses Analisis *Throughput*
(**Sumber**: Perancangan)

## 3.3 Kerangka Solusi Permasalahan

Rangka keseluruhan proses penyelesaian masalah yang telah dirumuskan dan ditelaah dari aspek tertentu, dan sesuai dengan tujuan yang telah ditetapkan dijabarkan dalam bentuk *flowchart*.

1. *Flowchart* Alir Perhitungan *Bandwidth*

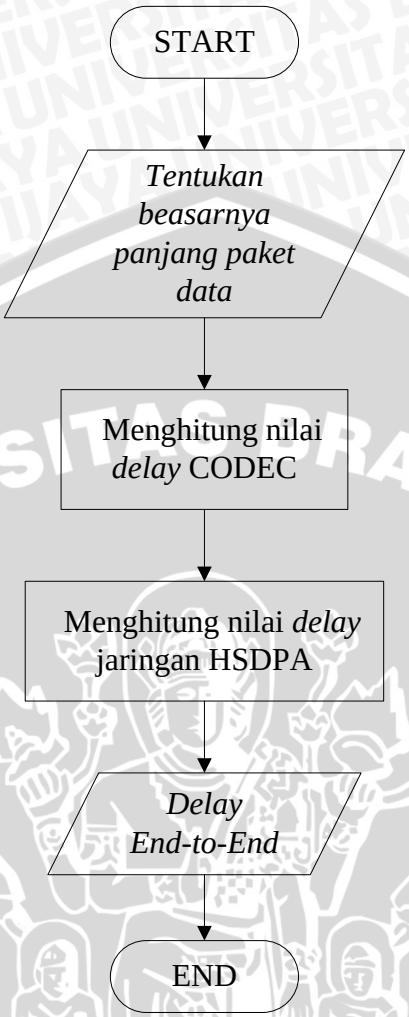



**Gambar 3.5** Diagram Alir Perhitungan *Bandwidth* Video *Streaming*
(**Sumber**: Perancangan)

Perhitungan *bandwidth* dipengaruhi oleh oleh jenis CODEC yang digunakan baik audio maupun video, *frame* rate, serta *payload* tiap paket audio maupun video. Langkah-langkah perhitungan *bandwidth* video *streaming* adalah sebagai berikut:

a. Menentukan *bit rate* CODEC video (BCODECv), *bit rate* CODEC audio (BCODECa), serta *frame rate* yang digunakan.
b. Menghitung *bandwidth* audio (Ba) dan *bandwidth* video (Bv).
c. Mendapatkan nilai *bandwidth* video *streaming*.

2. *Flowchart* Alir Perhitungan *Delay End-to-end*

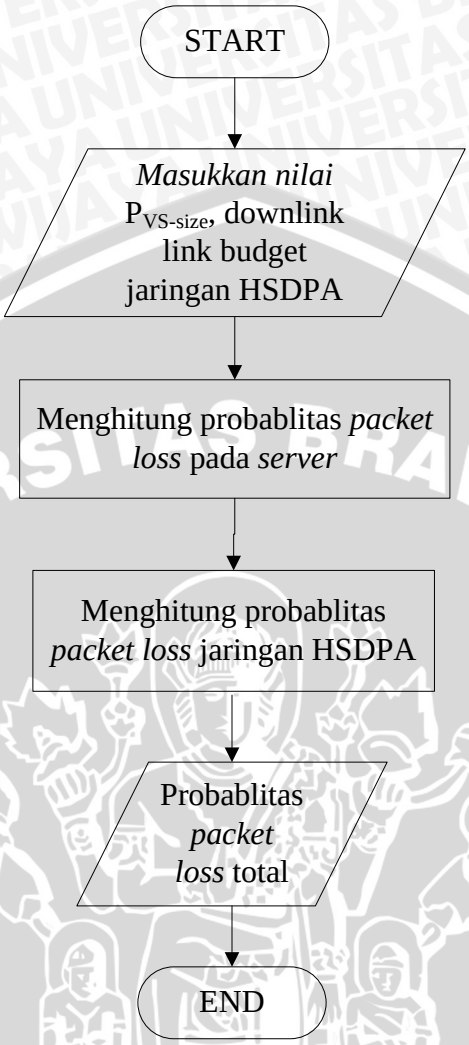



**Gambar 3.6** Diagram Alir Perhitungan *Delay End-to-End*
(**Sumber**: Perancangan)

*Delay end-to-end* merupakan keseluruhan *delay* yang ada dalam perjalanan paket data dari sumber hingga ke tujuan. Langkah-langkah perhitungan *delay end-to-end* video *streaming* pada jaringan HSDPA (*High Speed Downlink Packet Access*) menggunakan IPv6 adalah sebagai berikut:

a. Menghitung *delay* CODEC
b. Menghitung *delay* jaringan yang meliputi *delay* proses, *delay* transmisi, *delay* propagasi, dan *delay* antrian.
c. *Delay end-to-end* dapat dihitung dengan menjumlahkan *delay* CODEC dengan *delay* jaringan.

3. *Flowchart* Alir Perhitungan Probabilitas *Packet Loss*



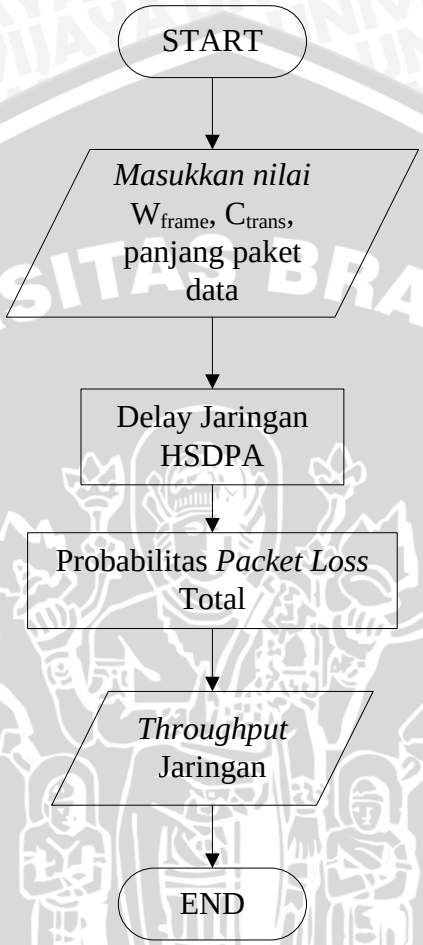

**Gambar 3.7** Diagram Alir Perhitungan Probabilitas *Packet Loss*
(**Sumber**: Perancangan)

*Packet loss* terjadi pada jaringan HSDPA, dan *server*. Perhitungan probabilitas *packet loss* dipengaruhi oleh ukuran paket data video *streaming* dan *downlink link budget* jaringan HSDPA. Langkah-langkah perhitungan probabilitas *packet loss* pada jaringan HSDPA (*High Speed Downlink Packet Access*) menggunakan IPv6 adalah sebagai berikut:

a. Menentukan panjang paket video *streaming* ($P_{VS\text{-}size}$) dan *downlink link budget* jaringan HSDPA.
b. Menghitung probablitas *packet loss* pada server
c. Menghitung probablitas *packet loss* pada jaringan HSDPA.

d. Probabilitas *packet loss* total diperoleh dengan menjumlahkan probablitas *packet loss* pada server dan probablitas *packet loss* jaringan HSDPA.

4. *Flowchart* Alir Perhitungan *Throughput*




**Gambar 3.8** Diagram Alir Perhitungan *Throughput*
(**Sumber**: Perancangan)

Perhitungan *throughput* dipengaruhi oleh *bit rate* CODEC yang digunakan, panjang frame di setiap *node*, serta kecepatan transmisi di setiap *node*. Nilai *throughput* diperlukan untuk mengetahui besarnya paket data yang diterima dari sisi penerima dengan benar setiap satuan waktu. Langkah-langkah perhitungan *throughput* adalah sebagai berikut:

a. Menentukan nilai panjang frame ($W_{frame}$) dan kecepatan transmisi ($C_{trans}$), panjang paket data.
b. Menghitung besarnya probabilitas *packet loss* total dan *delay* jaringan HSDPA
c. Mendapatkan nilai *throughput* jaringan.

# BAB IV
# HASIL DAN PEMBAHASAN

## 4.1 Umum

Pada bab ini akan dibahas mengenai analisis dan pembahasan performansi video *streaming* pada jaringan HSDPA (*High Speed Downlink Packet Access*) menggunakan IPv6 (*Internet Protocol version* 6). Analisis performansi video *streaming* yang dilakukan pada bab ini berdasarkan pada teori dan persamaan yang terdapat pada Bab II. Untuk itu ada beberapa tahap pembahasan dan analisis yang dilakukan, yaitu:

1. Menentukan parameter-parameter performansi video *streaming*, yaitu dengan menghitung nilai *bandwidth* video *streaming*, *delay end to end*, probabilitas *packet loss*, serta *throughput*.
2. Analisis dilakukan dengan cara menghitung nilai parameter performansi dengan nilai *bit rate* CODEC video dan audio yang berbeda-beda.

## 4.2 Spesifikasi Video *Streaming*

Analisis perfomansi video *streaming* dilakukan dengan mengubah nilai bit rate CODEC video maupun audio. Adapun spesifikasi video *streaming* yang digunakan dalam skripsi ini adalah sebagai berikut:

*a.* *Frame rate* video *streaming* yang dipergunakan adalah 30 fps atau 33 ms.

b. Spesifikasi CODEC audio maupun video yang dipergunakan dapat dilihat pada tabel 4.1 di bawah ini.

**Tabel 4.1** Spesifikasi CODEC

| **Audio CODEC** | **Bit Rate (kbps)** | ***Maximum Payload* (byte)** | ***Delay* CODEC (ms)** |
|---|---|---|---|
| AMR-WB+ | 5,2 – 48 | 46 | 20 – 40 |
| **Video CODEC** | **Bit Rate (kbps)** | ***Maximum Payload* (byte)** | ***Delay* CODEC (ms)** |
| H.264/AVC | 64 – 384 | 254 | 150 – 300 |

(**Sumber**: RFC 4352 and RFC 3984 RTP *Payload* Format for H.264 Video, 2005)

## 4.3 Spesifikasi HSDPA

Data spesifikasi HSDPA dipergunakan dalam perhitungan *delay* jaringan HSDPA, probabilitas *packet loss*, serta *throughput*. Untuk mempermudah proses analisis dan perhitungan, maka digunakan beberapa data sekunder, yaitu:

a. Parameter *downlink link budget* untuk jaringan HSDPA, dapat dilihat pada Tabel 4.2.

**Tabel 4.2** *Downlink Link Budget* HSDPA

| No | Parameter *Link Budget* | Nilai |
|---|---|---|
| *Transmitter* (*Node* B) | | |
| 1. | HS-DSCH Tx *Power* | 37,4 dBm |
| 2. | CPICH Tx *Power* | 33 dBm |
| 3. | *Total Tx Power* | 43 dBm |
| 4. | Tx *Antenna Gain* | 18 dBi |
| 5. | *Cable Loss* | 2 dB |
| | HS-DSCH EIRP | 53,4 dBm |
| *Receiver (User Equipment)* | | |
| 1. | UE *Noise Figure* | 8 dB |
| 2. | *Thermal Noise* | -108 dB |
| 3. | Rx *Antenna Gain* | 2 dBi |
| 4. | *Body Loss* | 0 dB |
| 5. | *Proceessing Gain* | 12 dB |
| 7. | *Orthogonality Factor* (α) | 0,6 |

(**Sumber:** PT. Indosat, Tbk KPI Surabaya, 2006: 1-23 dalam Linda Ekowati, 2008:68)

b. Terminal pengguna yang digunakan adalah terminal kategori 5, dimana jenis modulasi yang digunakan adalah 16-QAM, maksimum TBS (*Transport Block Size*) sebesar 7298 bit, dan maksimum *data rate* dari UE adalah 3,6 Mbps. (HSDPA by Siemens, 2004: 17)

c. *Physical interface* yang digunakan pada SGSN, RNC, dan *Node* B berdasarkan protokol stack HSDPA pada Gambar 2.8. Sedangkan *physical interface* yang digunakan pada GGSN adalah *Fast Ethernet*.

## 4.4 Analisis *Bandwidth* Video *Streaming*

Aplikasi video *streaming* menggunakan jenis CODEC H.264/AVC untuk video dengan *bit rate* CODEC antara 64 – 384 kbps dan AMR-WB+ untuk audio dengan *bit rate* CODEC antara 5,2 – 48 kbps dengan *frame rate* 33 ms. Analisis *bandwidth* video *streaming* dilakukan dengan mengubah-ubah bit rate CODEC video sedangkan bit rate CODEC audio tetap, mengubah-ubah bit rate CODEC audio sedangkan bit rate CODEC video tetap. Hal ini dilakukan untuk mengetahui pengaruh perubahan bit rate CODEC terhadap performansi video *streaming.*

Dengan menggunakan persamaan 2-1 hingga 2-2, maka besar *payload* aplikasi video *streaming* dapat diperoleh sebagai berikut:

$PLa = B_{CODECA} \times f_R = (48.10^3) \times (33.10^{-3})$

$= 1584$ bit

$PLv = B_{CODECv} \times f_R = (384.10^3) \times (33.10^{-3})$

$= 12672$ bit

Jumlah paket video dan audio yang akan di-*encode* dipengaruhi oleh besar *payload* maksimum dari video maupun audio, sepeti yang telah ditunjukkan pada tabel 2.2. Sehingga, jumlah paket video dan audio dapat diperoleh dengan menggunakan persamaan 2-3 dan 2-5 adalah sebagai berikut:

$Pa = PLa/PLa_{max} = 1584/640 = 2{,}475 \approx 3$ paket

$P_v = PLv/PLv_{max} = 12672/2032 = 6{,}2362 \approx 7$ paket

*Payload* video *streaming* yang akan di-*encode* diubah menjadi paket-paket yang disebut NALU (*Network Abstraction Layer Unit*). Besarnya tiap paket audio dan video aplikasi video *streaming* merupakan penjumlahan dari *payload* paket audio dan video dengan *header* NALU, RTP, UDP, dan IPv6. Dengan menggunakan persamaan 2-4 dan 2-6, maka besar paket audio dan video aplikasi video *streaming* yang dihasilkan dapat dihitung sebagai berikut:

$Pa\text{-}size = PLa + Pa \times (H_{NALU} + H_{RTP} + H_{UDP} + H_{IP})$

$= 1584 + 3 \times (8 + 96 + 64 + 320)$

$= 3048$ bit

$Pv\text{-}size = PLv + Pv \times (H_{NALU} + H_{RTP} + H_{UDP} + H_{IP})$

$= 12672 + 7 \times (8 + 96 + 64 + 320)$

$= 16088$ bit

Besarnya paket data aplikasi video *streaming* yang akan ditransmisikan berdasarkan persamaan 2-7 adalah sebagai berikut:

$P_{vs\text{-}size} = Pa\text{-}size + Pv\text{-}size$

$= 3048 + 16088$

$= 19136$ bit $= 2392$ byte $= 2{,}3359$ kB

Sehingga besarnya *bandwidth* dari video *streaming* yang dinyatakan dengan persamaan 2-8 dapat dihitung:

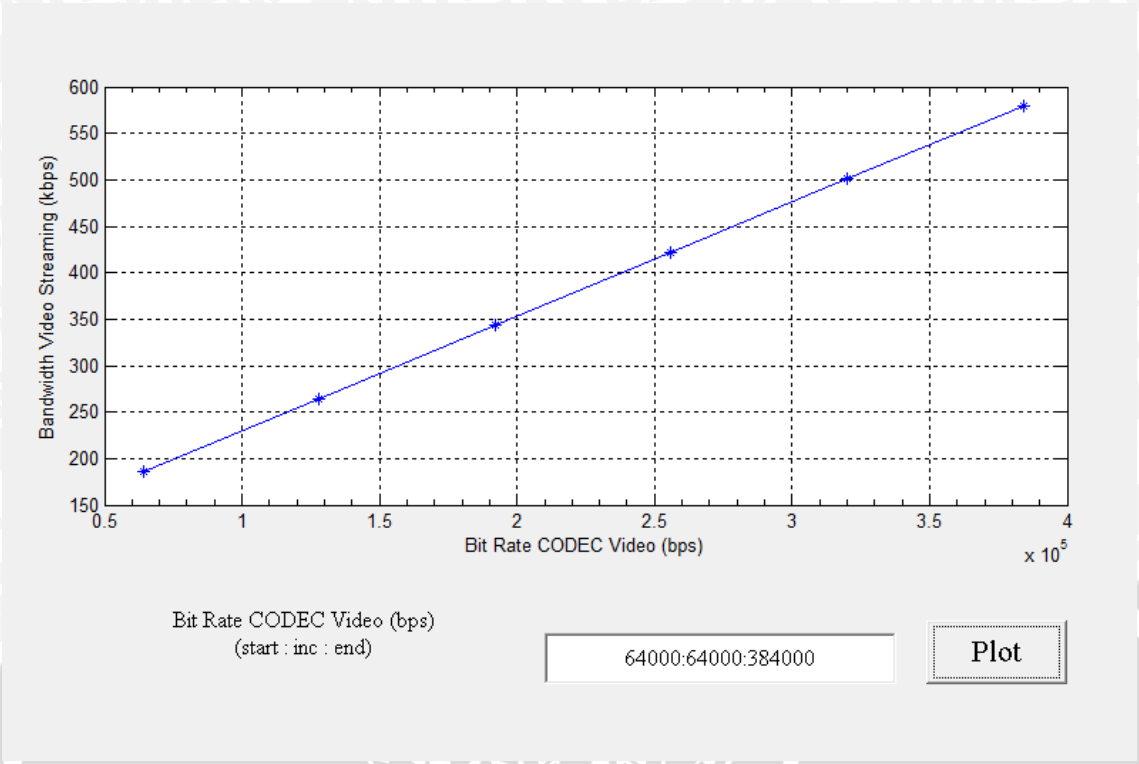
$$B_{VS} = \frac{Pa_{size}}{PLa}.B_{CODECa} + \frac{Pv_{size}}{PLv}.B_{CODECv}$$
$$= \frac{3048}{1584}.48.10^3 + \frac{16088}{12672}.384.10^3$$
$$= 92363,63636 + 487515,1515$$
$$= 579878,7879\,bps$$
$$= 579,8788\,kbps$$

Jadi, *bandwidth* aplikasi video *streaming* dengan bit rate CODEC video sebesar 384 kbps adalah 287,8344 kbps. Hasil analisis penggunaan *bandwidth* aplikasi video *streaming* dengan *bit rate* CODEC video yang berubah-ubah sebesar 64 kbps, 128 kbps, 192 kbps, 256 kbps, 320 kbps, dan 384 kbps serta bit rate CODEC audio yang tetap sebesar 48 kbps dapat ditunjukkan pada tabel 4.3 di bawah ini. Hubungan antara bandwidth aplikasi video *streaming* terhadap bit rate CODEC video dapat dilihat pada Gambar 4.1

**Tabel 4.3** Hasil analisis *bandwidth* aplikasi video *streaming* dengan bit rate CODEC video yang berubah-ubah

| No. | ***Bit Rate*** **CODEC Video (kbps)** | ***Bit Rate*** **CODEC Audio (kbps)** | ***Bandwidth*** **Aplikasi Video *Streaming* (kbps)** |
|---|---|---|---|
| 1 | 64 | 48 | 185,9394 |
| 2 | 128 | 48 | 264,7273 |
| 3 | 192 | 48 | 343,5152 |
| 4 | 256 | 48 | 422,3030 |
| 5 | 320 | 48 | 501,0909 |
| 6 | 384 | 48 | 579,8788 |

(**Sumber**: Hasil Perhitungan)



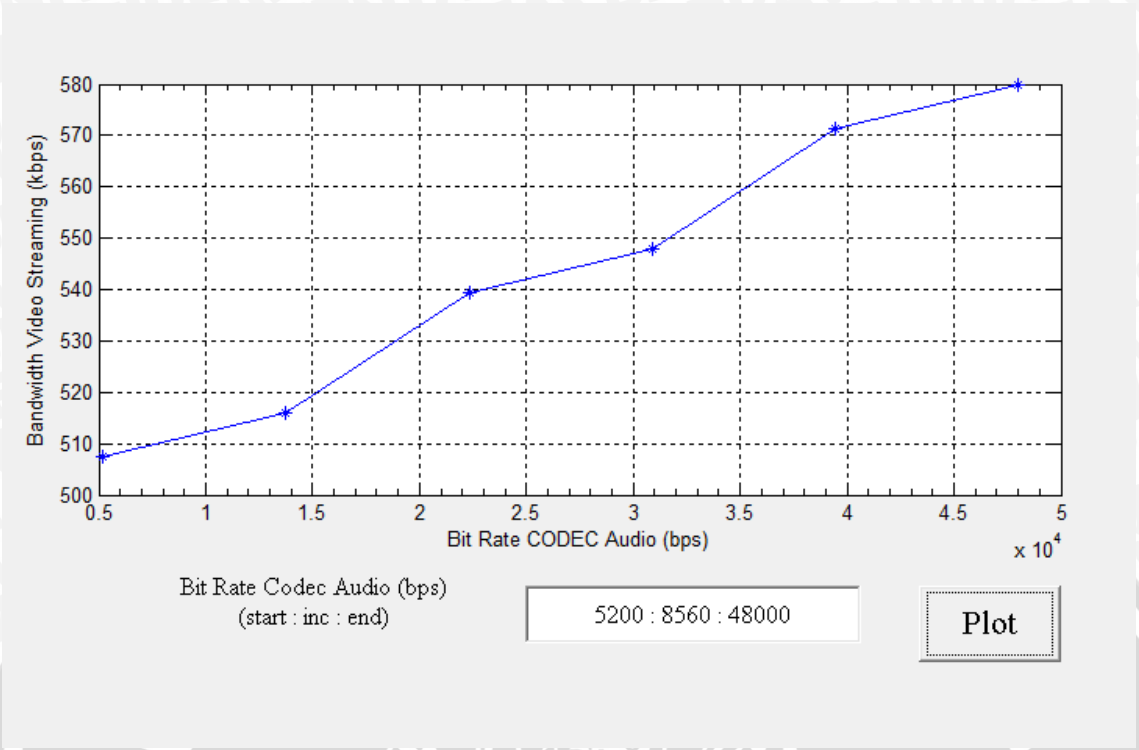

**Gambar** 4.1 Hubungan *bandwidth* aplikasi video *streaming* terhadap bit rate CODEC video yang berubah-ubah
**(Sumber**: Hasil Perhitungan)

Dengan menggunakan cara yang sama, hasil analisis penggunaan *bandwidth* aplikasi video *streaming* dengan *bit rate* CODEC video yang berubah-ubah sebesar 5,2 kbps, 13,76 kbps, 22,32 kbps, 30,88 kbps, 39,44 kbps, dan 48 kbps serta bit rate CODEC video yang tetap sebesar 384 kbps dapat ditunjukkan pada tabel 4.4 di bawah ini.

**Tabel 4.4** Hasil analisis *bandwidth* aplikasi video *streaming* dengan bit rate CODEC audio yang berubah-ubah

| No. | ***Bit Rate*** **CODEC Video (kbps)** | ***Bit Rate*** **CODEC Audio (kbps)** | ***Bandwidth*** **Aplikasi Video *Streaming* (kbps)** |
|---|---|---|---|
| 1 | 384 | 5,2 | 507,5030 |
| 2 | 384 | 13,76 | 516,0630 |
| 3 | 384 | 22,32 | 539,4109 |
| 4 | 384 | 30,88 | 547,9709 |
| 5 | 384 | 39,44 | 571,3188 |
| 6 | 384 | 48 | 579,8788 |

(**Sumber**: Hasil Perhitungan)




**Gambar** 4.2 Hubungan *bandwidth* aplikasi video *streaming* terhadap bit rate CODEC audio yang berubah-ubah
**(Sumber**: Hasil Perhitungan)

Berdasarkan hasil analisis perhitungan *bandwidth* video streaming di atas dapat kita ketahui bahwa:

1. Besarnya bandwidth aplikasi video *streaming* dipengaruhi oleh *frame rate* dan bit rate CODEC audio maupun video. *Frame rate* yang lebih rendah, akan memakan bandwidth yang lebih rendah dibandingkan frame rate yang lebih tinggi.
2. *Bandwidth* aplikasi video *streaming* paling besar adalah 566,2879 kbps dengan menggunakan *bit rate* CODEC video sebesar 384 kbps dan bit rate CODEC audio sebesar 48 kbps. Sedangkan *bandwidth* aplikasi video *streaming* paling kecil sebesar 185,9394 kbps saat *bit rate* CODEC video sebesar 64 kbps dan bit rate CODEC audio sebesar 48 kbps

## 4.5 Analisis *Delay End-to-End* Aplikasi Video *Streaming*

Model hubungan yang digunakan dalam analisis *delay end-to-end* aplikasi video *streaming* pada jaringan HSDPA yang menggunakan IPv6 adalah model hubungan antara server yang terhubung ke jaringan internet hingga ke UE (*User Equipment*).

Analisis perhitungan *delay end-to-end* aplikasi video *streaming* meliputi *delay* CODEC dan *delay* pada jaringan HSDPA.

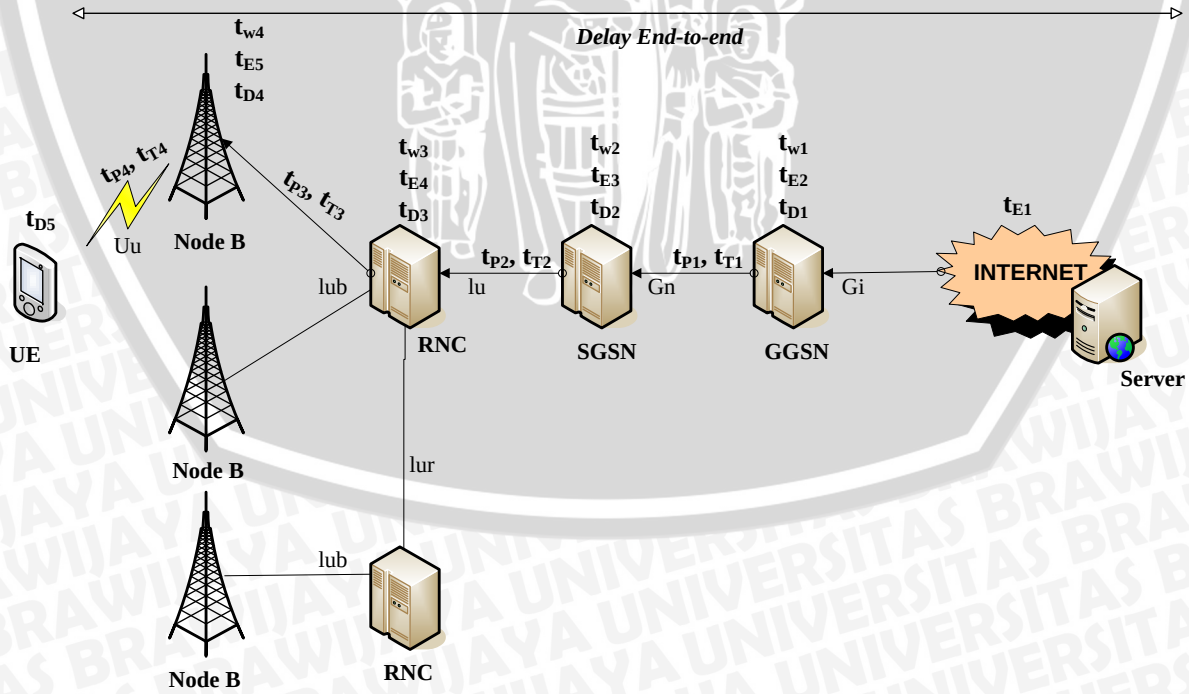
### 4.5.1 *Delay* CODEC

*Delay* CODEC aplikasi *video streaming* terdiri dari *delay* CODEC *audio* dan *video*. Aplikasi video *streaming* ini menggunakan jenis CODEC AMR-WB+ untuk audio dengan *delay* CODEC sebesar 20-40 ms. Sedangkan jenis CODEC yang digunakan untuk video adalah H.264/AVC dengan *delay* CODEC sebesar 150-300 ms seperti yang telah ditunjukkan pada Tabel 4.1. Sehingga besarnya *delay* CODEC pada aplikasi video *streaming* sesuai dengan persamaan 2-9 adalah:

$$t_{CODEC} = t_a + t_v = 40 \text{ ms} + 300 \text{ ms} = 340 \text{ ms}$$

### 4.5.2 *Delay* Jaringan *High Speed Downlink Packet Access* (HSDPA)

Analisis perhitungan *delay* pada jaringan HSDPA meliputi *delay* proses, *delay* transmisi, *delay* propagasi, dan *delay* antrian. Model hubungan perhitungan *delay* jaringan HSDPA yang menggunakan IPv6 adalah mulai dari *server* yang terhubung ke jaringan internet hingga UE. Model hubungan tersebut dapat dilihat pada Gambar 4.2.

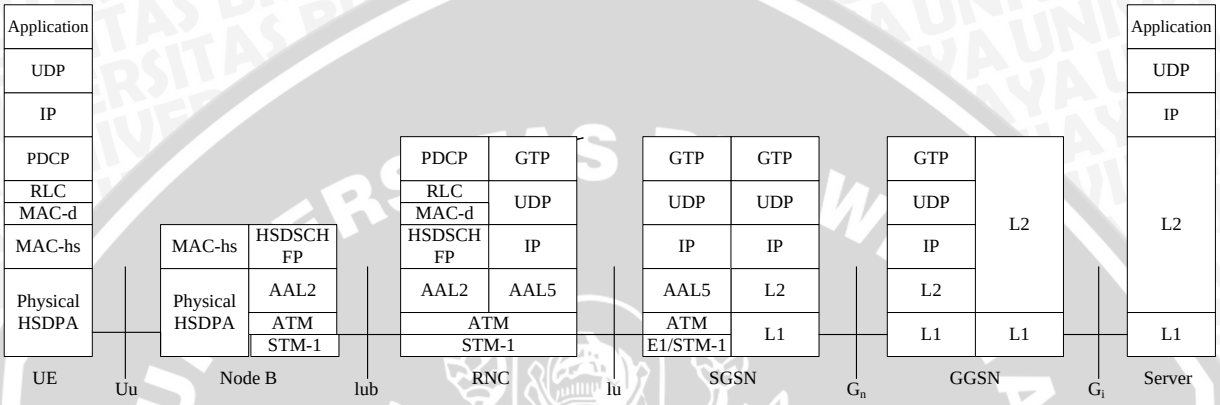



**Gambar 4.3**. Analisis *delay end-to-end* pada jaringan HSDPA

(**Sumber**: Hasil Analisis)

#### 4.5.2.1 *Delay* Proses

*Delay* proses pada jaringan HSDPA yang menggunakan IPv6 meliputi *delay* enkapsulasi dan *delay* dekapsulasi. Perhitungan *delay* proses pada jaringan HSDPA berlaku untuk masing-masing *layer-layer* seperti yang ditunjukkan pada Gambar 4.3.



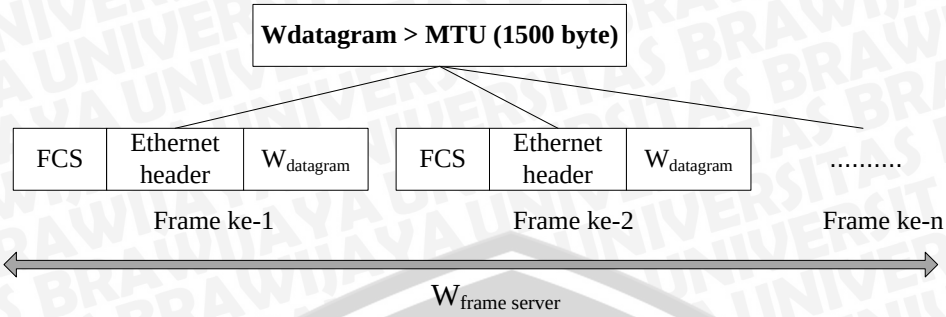

**Gambar 4.4** Analisis *delay* enkapsulasi dan dekapsulasi pada Jaringan HSDPA

(**Sumber**: Hasil Analisis)

- **Server**

Pada server terjadi proses enkapsulasi sebelum paket data ditransmisikan ke GGSN yang telah dihitung pada bab sebelumnya. Pada skripsi ini, standard protokol yang digunakan adalah Ethernet dengan *Maximum Transfer Unit* (MTU) *Ethernet* sebesar 1500 *byte* (RFC 895). Oleh karena itu, besar paket video *streaming* tidak boleh melebihi MTU sebesar 1500 byte.

Apabila panjang paket video *streaming* melebihi *Maximum Transfer Unit* (MTU) *Ethernet* sebesar 1500 *byte* (RFC 895), maka paket video *streaming* akan mengalami fragmentasi menjadi beberapa buah frame baru ditambahkan dengan *header Ethernet* dan FCS. Jika *datagram* IP tidak melebihi MTU *Ethernet*, maka *datagram* IP akan langsung ditambahkan dengan *header Ethernet* dan FCS. Format segmentasi datagram IP dapat terlihat pada Gambar 4.5 di bawah ini.




**Gambar 4.5** Format segmentasi *datagram* IP

(**Sumber**: Taufan Riza, 2002:41)

Sehingga, banyaknya frame ethernet yang ditransmisikan menuju GGSN sesuai dengan persamaan 2-18 sebagai berikut:

$$N_{frame\ Ethernet} = \frac{W_{datagram}}{MTU_{ethernet}}$$
$$= \frac{2392\ byte}{1500\ byte}$$
$$= 1,594666667\ frame$$

Terdapat 1 buah frame Ethernet berisi data 1500 byte dan 1 buah frame Ethernet berisi data sebesar (0,594666667 x 1500 *byte* = 892 *byte*), sehingga terdapat 2 buah frame Ethernet. Selanjutnya, jumlah total *frame* pada jaringan internet yang dapat dikirimkan ke GGSN sesuai dengan persamaan 2-19

$$W_{frame\ server} = P_{VS\text{-}size} + [N_{frame\ Ethernet} \text{ x } (H_{ethernet} + FCS)]$$
$$= 2392 + [2 \text{ x } (14 + 4)]$$
$$= 2428\ byte$$

Dalam skripsi ini, pada *sever* yang terhubung ke jaringan internet digunakan standar *interface Gigabit Ethernet* dengan kecepatan 1 Gbps. Sehingga *delay* dekapsulasi pada *server* didapatkan dengan persamaaan 2-20, yaitu:

$$t_{E1} = \frac{W_{frame\ server} - P_{vs-size}}{C_{int}}$$
$$= \frac{2428 - 2392}{10^9 bps} x8$$
$$= 2,88.10^{-7}\ s$$

Hasil analisis *delay* enkapsulasi pada *server* untuk *bit rate* video CODEC 256 kbps, 320 kbps, dan 384 kbps ditunjukkan pada Tabel 4.5.

**Tabel 4.5** Hasil Analisis *delay* enkapsulasi pada Server

| No. | *Bit rate* CODEC Video | *Delay* Enkapsulasi ($t_{E1}$) |
|---|---|---|
| 1 | 256 kbps | $2{,}88.10^{-7}$ |
| 2 | 320 kbps | $2{,}88.10^{-7}$ |
| 3 | 384 kbps | $2{,}88.10^{-7}$ |

(**Sumber**: Hasil Perhitungan)

- ***Gateway* GPRS *Support Node* (GGSN)**

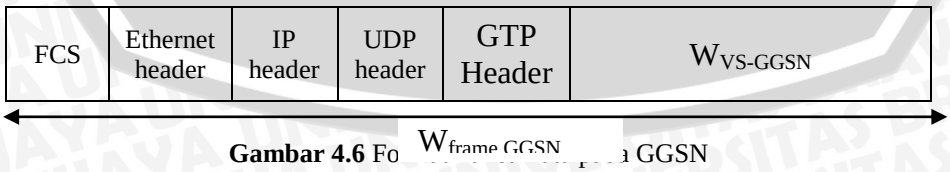
Pada GGSN, paket data yang diterima mengalami proses dekapsulasi. Dari proses dekapsulasi didapatkan paket data aplikasi video *streaming* sesuai dengan persamaan 2-21, yaitu:

$$W_{VS\ GGSN} = W_{frame\ server} - H_{ethernet} - FCS$$
$$= 2428 - 2\ x14 - 2\ x\ 4$$
$$= 2392\ byte$$

Dengan mengasumsikan GGSN menggunakan standar *Fast Ethernet* dengan kecepatan 100 Mbps (Ajay R. Mishra, 467), maka besarnya *delay* dekapsulasi pada GGSN didapatkan sesuai dengan persamaan 2-22 berikut:

$$t_{D1} = \frac{W_{frame\ server} - W_{VS\ GGSN}}{C_{GGSN}} x8$$
$$= \frac{2428 - 2392}{10^8 bps} x8$$
$$= 2{,}88.10^{-6} s$$

Format paket data GGSN dapat ditunjukkan pada Gambar 4.6 dimana paket data dari server yang telah didekapsulasi diberi penambahan header GTP, UDP , IPv6, Ethernet, dan FCS.




**Gambar 4.6** Fo[illegible] GGSN

(**Sumber**: Harri Holma and Antti Toskala, 2004:142)

Besar MSS dapat diperoleh sesuai dengan persamaan 2-23 sebagai berikut:

$$MSS = MTU - H_{GTP} - H_{UDP} - H_{IP}$$
$$= 1500 \text{ byte} - 8 \text{ byte} - 8 \text{ byte} - 40 \text{ byte} = 1444 \text{ byte}$$

Selanjutnya, paket data yang melebihi MSS akan disegmentasi menggunakan persamaan 2-24 berikut:

$$N_{datagram} = \frac{W_{vs-GGSN}}{MSS}$$
$$= \frac{2392\ byte}{1444\ byte}$$
$$= 1,656509695\ buah$$

Terdapat 1 buah datagram IP berisi data 1444 byte dan 1 buah datagram IP berisi data sebesar (0,656509695 x 1444 *byte* = 948 *byte*), sehingga terdapat 2 buah datagram IP. Kemudian, paket data aplikasi video *streaming* dienkapsulasi dengan penambahan *header* GTP, UDP, dan IP (3GPP TS 29.060 Rel.5) sesuai dengan persamaan 2-25 berikut:

$$W_{datagram} = W_{VS\ GGSN} + N_{datagram} \text{ x } (H_{GTP} + H_{UDP} + H_{IP})$$
$$= 2392 + 2 \text{ x } (8 \text{ byte} + 8 \text{ byte} + 40 \text{ byte})$$
$$= 2504 \text{ byte}$$

Karena panjang datagram IP di GGSN melebihi *Maximum Transfer Unit* (MTU) *Ethernet* (1500 *byte*), maka datagram IP akan mengalami fragmentasi menjadi beberapa buah frame sesuai dengan persamaan 2-26 di bawah ini:

$$N_{frame\ Ethernet} = \frac{W_{datagram}}{MTU_{ethernet}}$$
$$= \frac{2504\ byte}{1500\ byte}$$
$$= 1,669333333\ buah$$

Terdapat 1 buah frame Ethernet berisi data 1500 byte dan 1 buah frame Ethernet berisi data sebesar (0,669333333 x 1500 *byte* = 1004 *byte*), sehingga terdapat 2 buah frame Ethernet. Sehingga jumlah total *frame* pada GGSN yang dapat dikirimkan ke SGSN sesuai dengan persamaan 2-27 berikut:

$$W_{frame\ GGSN} = W_{datagram} + [N_{frame\ Ethernet} \text{ x } (H_{ethernet} + FCS)]$$
$$= 2504 + [2 \text{ x } (14 + 4)]$$
$$= 2540\ byte$$

Sehingga *delay* enkapsulasi pada GGSN didapatkan dengan persamaaan 2-28, yaitu:

$$t_{E2} = \frac{W_{frame\ GGSN} - W_{VS\ GGSN}}{C_{GGSN}} x8$$
$$= \frac{2540 - 2392}{10^8 bps} x8$$
$$= 1{,}184.10^{-5} s$$

Hasil analisis *delay* enkapsulasi dan *delay* dekapsulasi pada GGSN untuk *bit rate* video CODEC 256 kbps, 320 kbps, dan 384 kbps ditunjukkan pada tabel 4.6.

**Tabel 4.6** Hasil Analisis *Delay* Enkasulasi dan Dekapsulasi pada GGSN

| **No.** | ***Bit Rate* CODEC Video** | ***Delay* Dekapsulasi ($t_{D1}$)** | ***Delay* Enkapsulasi ($t_{E2}$)** |
|---|---|---|---|
| 1 | 256 kbps | $2{,}88.10^{-6}$ | $1{,}184.10^{-6}$ |
| 2 | 320 kbps | $2{,}88.10^{-6}$ | $1{,}184.10^{-6}$ |
| 3 | 384 kbps | $2{,}88.10^{-6}$ | $1{,}184.10^{-6}$ |

(**Sumber**: Hasil Perhitungan)

- ***Serving GPRS Support Node* (SGSN)**

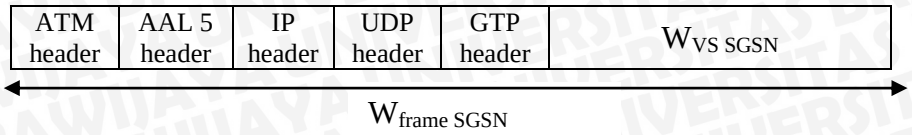
Paket data yang diterima dari GGSN mengalami proses dekapsulasi pada SGSN sesuai dengan persamaan 2-29

$W_{VS\ SGSN} = W_{frame\ GGSN} - H_{GTP} - H_{UDP} - H_{IPv6} - H_{ethernet} - FCS$
$= 2540 - 2x8 - 2x8 - 2x40 - 2x14 - 2x4$
$= 2392\ byte$

Dengan mengasumsikan bahwa SGSN menggunakan standar *Fast Ethernet* dengan kecepatan data 100 Mbps, maka besarnya *delay* dekapsulasi didapatkan dengan menggunakan persamaan 2-30

$$t_{D2} = \frac{W_{frame\ GGSN} - W_{VS\ SGSN}}{C_{SGSN}} x8$$
$$= \frac{2540 - 2392}{10^8 bps} x8$$
$$= 1{,}184.10^{-5} s$$

Format paket data SGSN dapat ditunjukkan pada Gambar 4.7 dimana paket data dari GGSN yang telah didekapsulasi diberi penambahan header GTP, UDP , IPv6, AAL5, dan ATM.




**Gambar 4.7** Format Paket Data pada SGSN
(**Sumber**: Harri Holma and Antti Toskala, 2004:149)

Besar MSS dapat diperoleh sesuai dengan persamaan 2-31 adalah sebagai berikut:

$$\begin{aligned} MSS &= MTU - H_{GTP} - H_{UDP} - H_{IP} \\ &= 1500 \text{ byte} - 8 \text{ byte} - 8 \text{ byte} - 40 \text{ byte} \\ &= 1444 \text{ byte} \end{aligned}$$

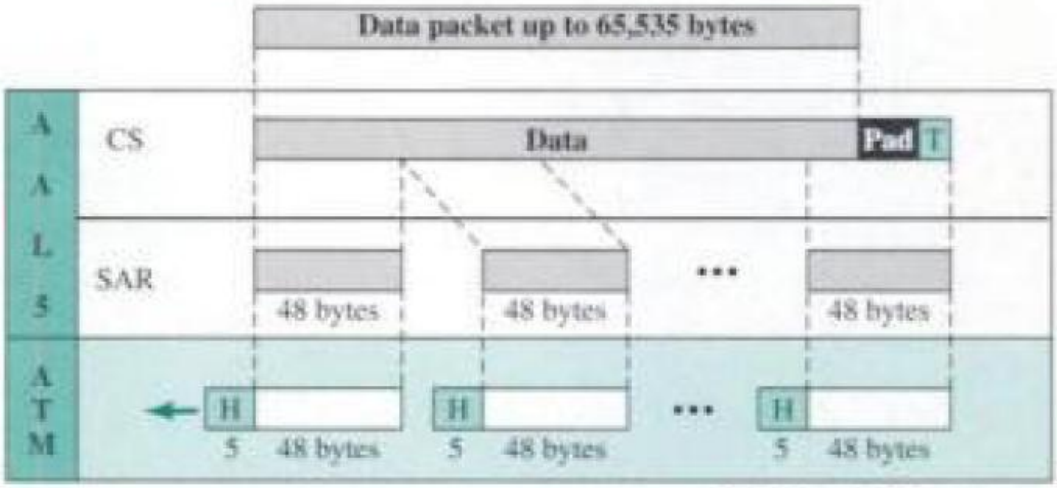
Selanjutnya, paket data yang melebihi MSS akan difragmentasi menggunakan persamaan 2-32 berikut

$$\begin{aligned} N_{datagram} &= \frac{W_{vs-SGSN}}{MSS} \\ &= \frac{2392\ byte}{1444\ byte} \\ &= 1,656509695\ buah \end{aligned}$$

Terdapat 1 buah datagram IP berisi data 1444 byte dan 1 buah datagram IP berisi data sebesar (0,656509695 x 1444 *byte* = 948 *byte*), sehingga terdapat 2 buah datagram IP. Kemudian, paket data aplikasi video *streaming* dienkapsulasi dengan *header* GTP, UDP, dan IP sesuai dengan persamaan 2-33 di bawah ini:

$$\begin{aligned} W_{datagram} &= W_{VS\ SGSN} + N_{datagram} \text{ x } (H_{GTP} + H_{UDP} + H_{IP}) \\ &= 2392 + 2\text{x } (8 \text{ byte} + 8 \text{ byte} + 40 \text{ byte}) \\ &= 2504 \text{ byte} \end{aligned}$$

Setelah itu, saat memasuki lapisan AAL5, paket data tersebut dibentuk menjadi CPCS PDU (*Common Part Convergence Sublayer Protocol Data Unit*) kemudian dibentuk menjadi blok-blok PDU SAR (*Segmentation and Reassembly Sublayer*) seperti terlihat pada Gambar 4.8.




**Gambar 4.8** Format Frame AAL5

(**Sumber**: Behrouz A. Forouzan, 2008: 459)

Terdapat 1 buah paket pada CPCS-PDU sebesar 2504 byte. Selanjutnya, besar paket data total pada CPCS-PDU diperoleh dengan menggunakan persamaan 2-34 sebagai berikut:

$$W_{CPCS\text{-}PDU\ total} = W_{datagram} + NCPCS_{\text{-}PDU} \times H_{CPCS\text{-}PDU}$$
$$= 2504 + 1 \times 8$$
$$= 2512 \text{ byte}$$

Kemudian, *frame* CPCS-PDU dipecah menjadi blok-blok *payload* PDU SAR (*Segmentation and Reassembly Sublayer*) sebesar 48 *byte*, yang kemudian diteruskan menuju layer ATM. Sehingga banyaknya frame ATM yang terbentuk sesuai dengan persamaan 2-35:

$$N_{frame\ ATM} = \frac{W_{frame\ CPCS-PDU}}{48\,byte}$$
$$= \frac{2512\ byte}{48\,byte}$$
$$= 52,33333333$$

Terdapat 52 *frame* berisi data 48 byte dan 1 *frame* berisi data sebesar (0,33333333 x 48 *byte* = 16 *byte*). *Frame* yang berisi data 24 byte tersebut akan diberi *padding bytes* sebesar (48 *byte* – 16 *byte* = 32 *byte*), sehingga terdapat 53 sel ATM. Selanjutnya setiap sel ATM diberi *header* ATM sebesar 5 byte sehingga panjang *frame* ATM menjadi 53 byte. Panjang *frame* di SGSN yang siap

ditransmisikan menuju RNC merupakan panjang *frame* ATM total, yang sesuai dengan persamaan 2-36

$$W_{\text{frame SGSN}} = W_{\textit{frame ATM total}} = N_{\textit{frame ATM}} \text{ x } W_{\textit{frame ATM}}$$
$$= 53 \text{ x } 53 \textit{ byte}$$
$$= 2809 \textit{ byte}$$

Sehingga, *delay* enkapsulasi yang terjadi pada SGSN didapatkan dengan persamaaan 2-37, yaitu:

$$t_{E3} = \frac{W_{\textit{frame SGSN}} - W_{\textit{VS SGSN}}}{C_{SGSN-RNC}} x8$$
$$= \frac{2809 - 2392}{155,52.10^{6} bps} x8$$
$$= 2,1451.10^{-5} s$$

Hasil analisis *delay* enkapsulasi dan *delay* dekapsulasi pada SGSN untuk *bit rate* video CODEC 256 kbps, 320 kbps, dan 384 kbps ditunjukkan pada tabel 4.7.

**Tabel 4.7** Hasil Analisis *Delay* Enkasulasi dan Dekapsulasi pada SGSN

| No. | *Bit Rate* CODEC Video | *Delay* Dekapsulasi ($t_{D2}$) | *Delay* Enkapsulasi ($t_{E3}$) |
|---|---|---|---|
| 1 | 256 kbps | $1,184.10^{-5}$ | $1,6718.10^{-5}$ |
| 2 | 320 kbps | $1,184.10^{-5}$ | $1,9084.10^{-5}$ |
| 3 | 384 kbps | $1,184.10^{-5}$ | $2,1451.10^{-5}$ |

(**Sumber**: Hasil Perhitungan)

- ***Radio Network Controller* (RNC)**

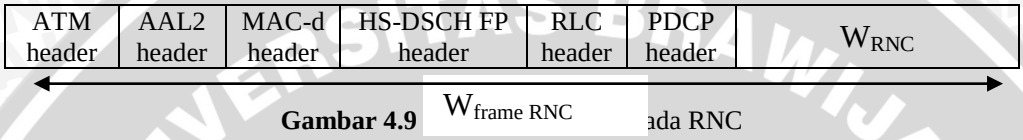
Paket data yang diterima dari SGSN mengalami proses dekapsulasi pada RNC sesuai dengan persamaan 2-38

$$W_{\text{VS RNC}} = W_{\textit{frame SGSN}} - H_{\text{GTP}} - H_{\text{UDP}} - H_{\text{IPv6}} - H_{\textit{AAL5}} - H_{\text{ATM}}$$
$$= 2809 - 2x8 - 2x8 - 2x40 - 1x8 - 53x\ 5$$
$$= 2424 \textit{ byte}$$

RNC menggunakan *interface* sistem transmisi ATM STM-1 yang memberikan kecepatan data sebesar 155,52 Mbps. Nilai *delay* dekapsulasi pada RNC didapatkan dengan persamaan 2-39

$$t_{D3} = \frac{W_{frame\,SGSN} - W_{VS\,RNC}}{C_{RNC}} x8$$
$$= \frac{2809 - 2424}{155,52.10^{6}\,bps} x8$$
$$= 1,9805.10^{-5}\,s$$

Format paket data RNC dapat ditunjukkan pada Gambar 4.9 dimana paket data dari SGSN yang telah di dekapsulasi diberi penambahan header PDCP, RLC, MAC-d, AAL2, dan ATM.




**Gambar 4.9** ada RNC

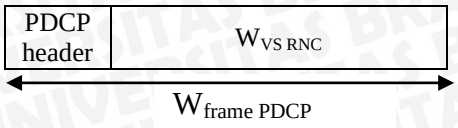
(**Sumber**: Harri Holma and Antti Toskala, 2004:156)

Pada layer PDCP (*Packet Data Convergence Protocol*), paket datagram IP aplikasi video *streaming* dienkapsulasi dengan penambahan *header* sebesar 1 *byte*, yaitu *header* PDU *Type* sebesar 3 bits dan PID (*Packet Identifier*) sebesar 5 bits (3GPP TS. 25. 323 Rel.5). Banyaknya paket IP pada RNC dapat diperoleh sebagai berikut:

$$N_{datagram} = \frac{W_{vs-RNC}}{1500}$$
$$= \frac{2424\ byte}{1500\,byte}$$
$$= 1,616\ buah$$

Terdapat 1 paket IP berisi data 1500 byte dan 1 paket IP berisi data sebesar (0,616 x 1500 *byte* = 924 *byte*). Format frame PDCP dapat ditunjukkan pada Gambar 4.10. Sehingga terdapat 2 paket IP. Besar frame total pada layer PDCP adalah sebagai berikut:

$$W_{frame\ PDCP} = W_{VS\ RNC} + H_{PDCP}$$
$$= 2424\ byte + 2 \text{ x } 1\ byte$$
$$= 2426\ byte$$




**Gambar 4.10** Format Frame PDCP
**(Sumber:** 3GPP TS. 25.323 Rel 5)

Selanjutnya pada layer RLC, frame PDCP disegmentasi menjadi RLC PDU *fixed size*. Pada Dokumen 3GPP TS.25.323. Rel.5 tentang *Radio Link Control* (RLC) *Protocol Spesifications* tidak ditegaskan besaran untuk tiap *fixed size*, tetapi pada skripsi ini besaran tiap *fixed size* mengacu pada besaran yang digunakan pada penelitian *Ericsson EuroLab Netherlands*, yaitu sebesar 40 *byte*. Sehingga jumlah *frame* RLC PDU sesuai dengan persamaan 2-42

$$N_{frame\,RLC} = \frac{W_{frame\ PDCP}}{40\,byte}$$
$$= \frac{2426\ byte}{40\,byte}$$
$$= 60,65$$

Terdapat 60 *frame* berisi data 40 byte dan 1 *frame* berisi data sebesar (0,65 x 40 *byte* = 26 *byte*). *Frame* yang berisi data 18 byte tersebut akan diberi *padding bytes* sebesar (40 *byte* – 26 *byte* = 14 *byte*) , sehingga terdapat 61 *frame* RLC. Selanjutnya setiap *frame* diberi *header* sebesar 2 byte sehingga panjang *frame* RLC menjadi 42 byte. Panjang *frame* RLC total yang siap diteruskan ke *layer* MAC-d dapat dihitung sesuai dengan persamaan 2-43

$$W_{frame\ \text{RLC total}} = N_{frame\ \text{RLC}} \text{ x } W_{frame\ \text{RLC}}$$
$$= 61 \text{ x } 42\ byte$$
$$= 2562\ byte$$

Pada *layer* MAC-d, RLC PDU disegmentasi menjadi MAC-d SDU (*Service Data Unit*) sebesar 42 *byte*. Jumlah *frame* MAC-d SDU sesuai dengan persamaan 2-44

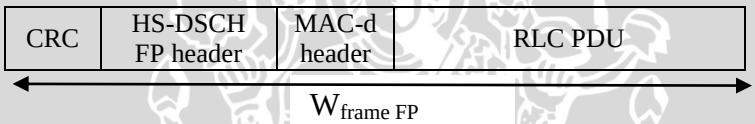
$$N_{frame\ MAC-d} = \frac{W_{frame\ RLC\,total}}{42\,byte}$$
$$= \frac{2562\,byte}{42\,byte}$$
$$= 61$$

Sedangkan panjang frame MAC-d dapat dihitung dengan menjumlahkan MAC-d SDU dengan *header* MAC-d sebesar 4 bit [Erik Dahlman et.al, 2008: 158] dengan persamaan 2-45:

$$W_{frame\ \text{MAC-d}} = H_{\text{MAC-d}} + 42\ byte$$
$$= 0{,}5\ \text{byte} + 42\ byte$$
$$= 42{,}5\ byte$$

Selanjutnya, *frame* MAC-d dienkapsulasi pada HS-DSCH FP *(Frame Protocol)* dengan menambahkan *header* HS-DSCH FP sebesar 7 byte dan CRC *(Cyclic Redundancy Check)* sebesar 2 byte. Sehingga panjang frame HS-DSCH FP sesuai dengan persamaan 2-46 adalah:

$$W_{frame\ \text{FP}} = W_{\text{frame MAC-d}} + H_{FP} + \text{CRC}$$
$$= 42{,}5\ byte + 7\ byte + 2\ byte$$
$$= 51{,}5\ byte$$

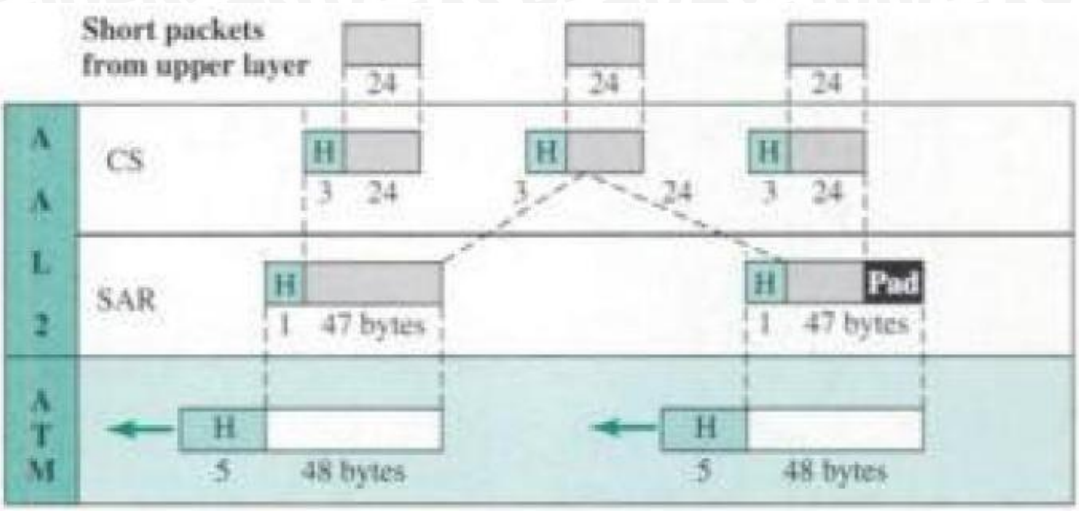



**Gambar 4.11** Format Frame FP
(**Sumber**: Anthony Lo, 2004:8)

Format frame FP ditunjukkan pada Gambar 4.11 dimana RLC-PDU diberi penambahan header MAC-d, HS-DSCH FP, dan CRC. Panjang *frame* HS-DSCH FP total diproleh dengan persamaan 2-47

$$W_{frame\ \text{FP total}} = N_{frame\ MAC\text{-d}} \times W_{frame\ \text{FP}}$$
$$= 61 \times 51{,}5\ byte$$
$$= 3141{,}5\ byte$$

Saat memasuki lapisan AAL2, paket data tersebut dibentuk menjadi CPS (*Common Part Sublayer*) dan blok-blok PDU SAR seperti terlihat pada gambar 4.12 dibawah ini.




**Gambar 4.12** Format Frame AAL2
(**Sumber**: Behrouz A. Forouzan, 2008: 457)

Banyaknya paket data pada CS PDU dapat diperoleh dengan persamaan 2-48 berikut:

$$N_{CS-PDU} = \frac{W_{frame\,FP\,total}}{24}$$
$$= \frac{3141{,}5\ byte}{24\,byte}$$
$$= 130{,}8958333\ buah$$

Terdapat 130 buah paket berisi data 24 byte dan 1 buah paket berisi data sebesar (0,8958333 x 24 *byte* = 22 *byte*), sehingga terdapat 131 buah paket CS PDU. Selanjutnya, besar paket data total pada CS-PDU diperoleh dengan menggunakan persamaan 2-49 sebagai berikut

$$W_{CS\text{-}PDU\ total} = W_{frame\ FP\ total} + (N_{CS\text{-}PDU} \times H_{CS\text{-}PDU})$$
$$= 3141{,}5 + 131 \times 3\ byte$$
$$= 3534{,}5\ byte$$

Kemudian, *frame* CS-PDU dipecah menjadi blok-blok *payload* PDU SAR (*Segmentation and Reassembly Sublayer*) sebesar 47 *byte*

$$N_{SAR-PDU} = \frac{W_{frame\,CS-PDU}}{47\,byte}$$
$$= \frac{3534{,}5\ byte}{47\,byte}$$
$$= 75{,}20212766$$

Pada PDU SAR, terdapat 75 *frame* berisi data 47 byte dan 1 *frame* berisi data sebesar (0,20212766 x 47 *byte* = 10 *byte* ). *Frame* yang berisi data 22 byte tersebut akan diberi *padding bytes* sebesar (47 *byte* – 10 *byte* = 7 *byte*), sehingga terdapat 76 blok PDU-SAR. Selanjutnya, setiap blok PDU SAR ditambahkan header sebesar 1 byte sehingga besar setiap blok PDU SAR menjadi 48 byte. Besar paket data pada PDU SAR dapat diperoleh menggunakan persamaan 2-51 sebagai berikut:

$$\begin{aligned} W_{\text{frame PDU SAR}} &= N_{\text{SAR-PDU}} \text{ x } 48 \text{ byte} \\ &= 76 \text{ x } 48 \text{ byte} \\ &= 3648 \text{ byte} \end{aligned}$$

Pada layer ATM, blok-blok PDU SAR dipecah menjadi fixed cell ATM sebesar 48 byte. Banyaknya frame ATM yang ditransmisikan menuju Node B menggunakan persamaan 2-52 sebagai berikut:

$$\begin{aligned} N_{frame\,ATM} &= \frac{W_{frame\,PDU-SAR}}{48\,byte} \\ &= \frac{3648\ byte}{48\,byte} \\ &= 76 \end{aligned}$$

Selanjutnya, setiap *payload cell* ATM diberi *header* sebesar 5 byte sehingga panjang *fixed-size cells* ATM menjadi sebesar 53 byte (Stallings). Panjang *frame* yang siap ditransmisikan menuju *Node* B merupakan panjang *frame* ATM total sesuai dengan persamaan 2-53

$$\begin{aligned} W_{\text{frame RNC}} = W_{frame\ \text{ATM total}} &= N_{frame\ \text{ATM}} \text{ x } W_{frame\ \text{ATM}} \\ &= 76 \text{ x } 53\ byte \\ &= 4028\ byte \end{aligned}$$

Sehingga, besar *delay* enkapsulasi yang terjadi pada RNC didapatkan dengan persamaaan 2-54, yaitu:

$$\begin{aligned} t_{E4} &= \frac{W_{frame\ RNC} - W_{VS\ RNC}}{C_{RNC}} x8 \\ &= \frac{4028 - 2424}{155,52.10^{6}\,bps} x8 \\ &= 8,2510.10^{-5}\,s \end{aligned}$$

Hasil analisis *delay* enkapsulasi dan *delay* dekapsulasi pada RNC untuk *bit rate* video CODEC 256 kbps, 320 kbps, dan 384 kbps ditunjukkan pada tabel 4.8.

**Tabel 4.8** Hasil Analisis *Delay* Enkasulasi dan Dekapsulasi pada RNC

| No. | *Bit Rate* CODEC Video | *Delay* Enkapsulasi ($t_{D3}$) | *Delay* Enkapsulasi ($t_{E4}$) |
|---|---|---|---|
| 1 | 256 kbps | $1{,}6204.10^{-5}$ | $5{,}9825.10^{-5}$ |
| 2 | 320 kbps | $1{,}8004.10^{-5}$ | $7{,}2531.10^{-5}$ |
| 3 | 384 kbps | $1{,}9805.10^{-5}$ | $8{,}2510.10^{-5}$ |

(**Sumber**: Hasil Perhitungan)

- ***Node* B**

Paket data yang diterima dari RNC mengalami proses dekapsulasi pada *Node* B sesuai dengan persamaan 2-55 berikut:

$$W_{VS\ NodeB} = W_{frame\ RNC} - H_{ATM} - H_{CS\text{-}PDU} - H_{SAR\text{-}PDU} - H_{FP} - CRC$$
$$= 4028 - 76x5 - 131x3 - 76x1 - 61x7 - 61x2 = 2630\ byte$$

Format paket data di *Node* B dapat ditunjukkan pada Gambar 4.15 dimana paket data dari RNC diberi penambahan header ATM, AAL2, FP, dan CRC. *Node* B menggunakan *interface* STM-1 dengan kecepatan 155,52 Mbps. Maka besarnya *delay* dekapsulasi didapatkan dengan menggunakan persamaan 2-56

$$t_{D4} = \frac{W_{frame\ RNC} - W_{VS\ Node\ B}}{C_{Node\ B}} x8$$
$$= \frac{4028 - 2630}{155,52.10^{6}\ bps} x8$$
$$= 7{,}1914.10^{-5}\ s$$

Pada layer MAC-hs, beberapa frame MAC PDU dan header MAC-hs dibentuk menjadi satu TBS (*transport block size*) sebesar 7298 bits [HSDPA by Siemens, 2004: 17]. Banyaknya TBS yang akan ditransmisikan menuju UE adalah sebagai berikut:

$$N_{TBS} = \frac{W_{VS\ Node\ B}}{7298\ bit} x8$$
$$= \frac{2630\ byte}{7298\ bit} x8$$
$$= 2{,}882981639$$

Pada layer MAC-hs, terdapat 2 *frame* berisi data 7298 bit dan 1 *frame* berisi data sebesar (0,882981639 x 7298 bit = 6444 bit). Sehingga, terdapat 3 TBS. Selanjutnya, setiap TBS ditambahkan header CRC sebesar 2 byte. Panjang *frame* yang siap ditransmisikan menuju UE sesuai dengan persamaan 2-58

$$W_{frame\ Node\ B} = W_{VSNodeB} + N_{TBS} \text{ x CRC}$$
$$= 2630 + 3 \text{ x } 2 \text{ byte}$$
$$= 2636 \text{ byte}$$

Sehingga, *delay* enkapsulasi yang terjadi pada *Node* B didapatkan dengan persamaaan 2-59, yaitu:

$$t_{E5} = \frac{W_{frame\ Node\ B} - W_{VS\ NodeB}}{C_{Node\ B}} x8$$
$$= \frac{2636 - 2630}{155,52.10^{6} bps} x8$$
$$= 3,0864.10^{-7} s$$

Hasil analisis *delay* enkapsulasi dan *delay* dekapsulasi pada Node B untuk *bit rate* video CODEC 256 kbps, 320 kbps, dan 384 kbps ditunjukkan pada tabel 4.9.

**Tabel 4.9** Hasil Analisis *Delay* Enkasulasi dan Dekapsulasi pada Node B

| No. | ***Bit Rate*** **CODEC Video** | ***Delay*** **Dekapsulasi** ($t_{D4}$) | ***Delay*** **Enkapsulasi** ($t_{E5}$) |
|---|---|---|---|
| 1 | 256 kbps | $5,2006.10^{-5}$ | $3,0864.10^{-7}$ |
| 2 | 320 kbps | $6,2500.10^{-5}$ | $3,0864.10^{-7}$ |
| 3 | 384 kbps | $7,1914.10^{-5}$ | $3,0864.10^{-7}$ |

(**Sumber**: Hasil Perhitungan)

- ***User Equipment*** **(UE)**

Paket data yang diterima dari *Node* B mengalami proses dekapsulasi pada UE sesuai dengan persamaan 2-60

$$W_{VS\ UE} = W_{frame\ Node\ B} - H_{RTP} - H_{UDP} - H_{IP} - H_{PDCP} - H_{RLC} - H_{MAC\text{-}d} - H_{MAC\text{-}hs} - CRC$$
$$= 2636 - 10x12 - 10x8 - 10x40 - 2x1 - 61x2 - 61x0,5 - 3x21/8 - 3x2$$
$$= 1867,625 \textit{ byte}$$

| CRC | MAC-hs header | MAC header | RLC header | PDCP header | IP header | UDP header | RTP header | Data |
|---|---|---|---|---|---|---|---|---|

**Gambar 4.13** Format Paket Data pada UE

**(Sumber**: Harri Holma and Antti Toskala, 2004:165)

Format paket data di UE dapat ditunjukkan pada Gambar 4.13 dimana paket data dari *Node* B dilakukan dekapsulasi. Terminal pengguna yang digunakan adalah

terminal kategori 5, sesuai Tabel 2.4 dengan *data rate* maksimum sebesar 3,6 Mbps, maka besarnya *delay* dekapsulasi yang terjadi di UE didapatkan dengan persamaan 2-61

$$t_{D5} = \frac{W_{frame\ NodeB} - W_{VSUE}}{C_{UE}} x8$$
$$= \frac{2636 - 1867,625}{3,6.10^{6} bps} x8$$
$$= 1,7075.10^{-3} s$$

Hasil analisis *delay* dekapsulasi pada UE untuk video CODEC 256 kbps, 320 kbps, dan 384 kbps ditunjukkan pada tabel 4.10.

**Tabel 4.10** Hasil Analisis *Delay* Dekapsulasi pada UE

| No. | *Bandwidth* CODEC Video | *Delay* Dekapsulasi ($t_{D5}$) |
|---|---|---|
| 1 | 256 kbps | $1,3464.10^{-3}$ |
| 2 | 320 kbps | $1,5297.10^{-3}$ |
| 3 | 384 kbps | $1,7075.10^{-3}$ |

(**Sumber**: Hasil Perhitungan)

- ***Delay* Proses Total**

Dengan menggunakan persamaan 2-14 hingga 2-16 didapatkan besarnya *delay* proses total aplikasi video streaming sebagai berikut:

$t_{Etot} = t_{E1} + t_{E2} + t_{E3} + t_{E4} + t_{E5}$
$= 2,88.10^{-7}$ s + $1,184.10^{-5}$s + $2,1451.10^{-5}$s + $8,2510.10^{-5}$ s + $3,0864.10^{-7}$s
$= 1,1640.10^{-4}$ s

$t_{Dtot} = t_{D1} + t_{D2} + t_{D3} + t_{D4} + t_{D5}$
$= 2,88.10^{-6}$s + $1,184.10^{-5}$ s + $1,9805.10^{-5}$s + $7,1914.10^{-5}$s + $1,7075.10^{-3}$s
$= 1,8139.10^{-3}$ s

$t_{proc} = t_{Etot} + t_{Dtot}$
$= 1,1640.10^{-4}$ s + $1,8139.10^{-3}$ s
$= 1,9303.10^{-3}$ s

Untuk penggunaan IPv4, ada penambahan *delay header checksum* sebesar:

$t_{checksum} = N_{packet\text{-}IP}$ x T
= 14 x 0,002 = 28 ms

Oleh karena itu, besar delay proses pada jaringan HSDPA yang menggunakan IPv4 adalah sebagai berikut:

$$t_{proc} = t_{Etot} + t_{Dtot} + t_{checksum}$$
$$= 1{,}0378.10^{-4} + 1{,}8017.10^{-3} + 2{,}8.10^{-2}$$
$$= 2{,}9905.10^{-2} \text{ s}$$

Hasil analisis *delay* proses dapat diperoleh untuk *bit rate* video CODEC 256 kbps, 320 kbps, dan 384 kbps ditunjukkan pada tabel 4.11.

**Tabel 4.11** Hasil Analisis *Delay* Proses pada Jaringan HSDPA

| No. | *Bit Rate* CODEC Video | *Delay* Proses –IPv6 (s) | *Delay* Proses –IPv4 (s) |
|---|---|---|---|
| 1 | 256 kbps | $1{,}5183.10^{-3}$ | $2{,}5499.10^{-2}$ |
| 2 | 320 kbps | $1{,}7290.10^{-3}$ | $2{,}7703.10^{-2}$ |
| 3 | 384 kbps | $1{,}9303.10^{-3}$ | $2{,}9905.10^{-2}$ |

(**Sumber**: Hasil Perhitungan)

Berdasarkan hasil analisis perhitungan *delay* proses di atas, dapat kita ketahui bahwa *delay* proses dipengaruhi oleh bit rate CODEC yang digunakan dan header-header yang digunakan dalam proses enkapsulasi maupun dekapsulasi. Penggunaan bit rate CODEC yang semakin besar mempengaruhi besar *payload* dari video *streaming*. Sehingga, semakin besar bit rate CODEC video yang digunakan maka besar *delay* proses yang dihasilkan semakin besar.

Selain itu, penggunaan IPv6 menghasilkan *delay* proses yang lebih kecil dibandingkan dengan penggunaan IPv4. Penggunaan IPv4 pada jaringan HSDPA membutuhkan penambahan *delay header checksum*, dimana pada IPv6 tidak diperlukan adanya penambahan *delay header checksum.*

### 4.5.2.2 *Delay* Transmisi

Perhitungan *delay* transmisi pada jaringan HSDPA meliputi hubungan antara GGSN-SGSN, SGSN-RNC, RNC-Node B, dan Node B-UE.

- **GGSN-SGSN**

  Hubungan antara GGSN dan SGSN menggunakan *Fast Ethernet* dengan kecepatan transmisi data sebesar 100 Mbps sesuai standard IEEE 802.3. Sehingga nilai *delay* transmisi yang terjadi sesuai dengan persamaan 2-61

$$t_{T1} = \frac{W_{frame\ GGSN}}{C_{eth}} x8$$
$$= \frac{2540}{10^8 bps} x8$$
$$= 2,032.10^{-4} s$$

- **SGSN-RNC**

Hubungan antara SGSN dan RNC menggunakan *interface* sistem tranmisi digital STM-1 yang memberikan kecepatan data sebesar 155,52 Mbps sesuai standard ITU-T G.703. Sehingga nilai *delay* transmisi yang terjadi sesuai dengan persamaan 2-62

$$t_{T2} = \frac{W_{frame\ SGSN}}{C_{ATM}} x8$$
$$= \frac{2809}{155,52.10^6 bps} x8$$
$$= 1,44496.10^{-4} s$$

- **RNC-*Node* B**

Hubungan antara RNC dan *Node* B menggunakan *interface* sistem tranmisi digital STM-1 yang memberikan kecepatan data sebesar 155,52 Mbps sesuai standard ITU-T G.703. Sehingga nilai *delay* transmisi yang terjadi sesuai dengan persamaan 2-62

$$t_{T3} = \frac{W_{frame\ RNC}}{C_{ATM}} x8$$
$$= \frac{4028}{155,52.10^6 bps} x8$$
$$= 2,0720.10^{-4} s$$

- ***Node* B-UE**

*Delay* transmisi yang terjadi antara *Node* B dan UE bergantung pada *data rate* UE serta jumlah slot. Terminal pengguna diasumsikan memiliki *data rate* maksimum sebesar 3,6 Mbps. Jumlah slot ditentukan oleh *chip rate* HSDPA yaitu sebesar 3,84 Mcps. *Chip rate* sebesar 3,84 Mcps selanjutnya dibagi menjadi radio *frame* 2 ms. Satu slot terdiri dari 2560 *chip* sehingga banyaknya slot adalah:

$$n = \frac{(cr \times rf)}{2560\ chip}$$
$$= \frac{3,84.10^6\ cps \times 2.10^{-3} s}{2560 chip} \quad = \frac{7680\ chip}{2560\ chip} = 3\ slot$$

Jumlah *slot* pada *air interface* HSDPA adalah sebanyak 3 slot. Sehingga nilai *delay* transmisi yang terjadi sesuai dengan persamaan 2-61

$$t_{T4} = \frac{W_{frame\ Node\ B}}{n\,x\,C_{UE}}\,x8$$
$$= \frac{2636}{3\,x\,(3,6.10^{6})bps}\,x8$$
$$= 1,9526.10^{-3}\,s$$

- ***Delay* Transmisi Total**

*Delay* transmisi total yang terjadi sesuai dengan persamaan 2-63

$t_{Ttot} = t_{T1} + t_{T2} + t_{T3} + t_{T4}$
$= 2,032.10^{-4} + 1,44496.10^{-4} + 2,0720.10^{-4} + 1,9526.10^{-3}$
$= 2,5075.10^{-3}$ s

Hasil analisis nilai *delay* transmisi untuk *bit rate* video CODEC 256 kbps, 320 kbps, dan 384 kbps dapat ditunjukkan pada Tabel 4.12

**Tabel 4.12** Hasil Analisis *Delay* Transmisi pada Jaringan HSDPA yang Berbasis IPv6

| No. | ***Bit Rate* CODEC Video** | ***Delay* Transmisi (s)** |
|---|---|---|
| 1 | 256 kbps | $1,8223.10^{-3}$ |
| 2 | 320 kbps | $2,1781.10^{-3}$ |
| 3 | 384 kbps | $2,5075.10^{-3}$ |

(**Sumber**: Hasil Perhitungan)

Berdasarkan hasil analisis perhitungan analisis *delay* transmisi di atas, besar *delay* transmisi dipengaruhi oleh bit rate CODEC video dan kecepatan pemrosesan data di setiap *node*. Semakin besar bit rate CODEC video yang digunakan, mengakibatkan besar *payload* video *streaming* yang akan ditransmisikan juga semakin besar. Oleh karena itu, besar *delay* transmisi yang dihasilkan semakin besar pula. Selain itu, kecepatan pemrosesan yang besar akan melewatkan paket data yang lebih cepat sehingga *delay* transmisi yang dihasilkan akan semakin kecil.

### 4.5.2.3. *Delay* Propagasi

Analisis perhitungan *delay* propagasi meliputi: GGSN - SGSN – RNC – *Node* B – UE dimana propagasi antara GGSN hingga Node B menggunakan media *fiber optic*, sedangkan propagasi antara Node B dengan UE menggunakan media udara. Referensi

jarak antar *node* pada jaringan HSDPA yang digunakan dalam analisis perhitungan *delay* propagasi ditunjukkan pada Tabel 2.5.

- ***Delay* Propagasi dari GGSN ke SGSN**

  Hubungan antara GGSN ke SGSN menggunakan *fiber optic* dengan jarak 2000 m. Nilai *delay* propagasi antara GGSN dengan SGSN sesuai dengan persamaan 2-64

$$t_{P1} = \frac{N_{frame\ GGSN}\ x\ R}{v}$$
$$= \frac{2\,x\,2.10^{3}}{3.10^{8}}$$
$$= 1,3333.10^{-5}\,s$$

- ***Delay* Propagasi dari SGSN ke RNC**

  Hubungan antara SGSN ke RNC menggunakan *fiber optic* dengan jarak 3000 m. Nilai *delay* propagasi antara SGSN dengan RNC sesuai dengan persamaan 2-64

$$t_{P2} = \frac{N_{frame\ ATM}\ x\ R}{v}$$
$$= \frac{53\ x\ 3.10^{3}}{3.10^{8}}$$
$$= 5,3.10^{-4}\,s$$

- ***Delay* Propagasi dari RNC ke *Node* B**

  Hubungan antara RNC dengan *Node* B menggunakan *fiber optic* dengan jarak 400 m. Nilai *delay* propagasi antara RNC dengan *Node* B sesuai dengan persamaan 2-64

$$t_{P3} = \frac{N_{frame\ ATM}\ x\ R}{v}$$
$$= \frac{76\ \ x\,400}{3.10^{8}}$$
$$= 1,0133.10^{-4}\,s$$

- ***Delay* Propagasi dari *Node* B ke UE**

  Hubungan antara *Node* B dengan UE menggunakan media udara dengan jarak antara *Node* B dengan UE diasumsikan 1500 m. Nilai *delay* propagasi antara RNC dengan *Node* B sesuai dengan persamaan 2-64

$$t_{P4} = \frac{N_{TBS}\, x\, R}{v}$$
$$= \frac{3\, x\, 1500}{3.10^{8}}$$
$$= 1,5.10^{-5}\, s$$

- ***Delay* Propagasi Total**

*Delay* propagasi total yang terjadi pada jaringan HSDPA terdiri dari penjumlahan *delay* propagasi yang terjadi pada tiap *interface* jaringan sesuai dengan persamaan 2-65:

$$t_{Ptot} = t_{P1} + t_{P2} + t_{P3} + t_{P4}$$
$$= 1,3333.10^{-5} + 5,3.10^{-4} + 1,0133.10^{-4} + 1,5.10^{-5}$$
$$= 6,5966.10^{-4}\ s$$

Hasil analisis nilai *delay* propagasi untuk *bit rate* video CODEC 256 kbps, 320 kbps, dan 384 kbps dapat ditunjukkan pada Tabel 4.13

**Tabel 4.13** Hasil Analisis *Delay* Propagasi pada Jaringan HSDPA yang Berbasis IPv6

| No. | ***Bit Rate* CODEC Video** | ***Delay* Propagasi (s)** |
|---|---|---|
| 1 | 256 kbps | $4,9167.10^{-4}$ |
| 2 | 320 kbps | $5,7633.10^{-4}$ |
| 3 | 384 kbps | $6,5966.10^{-4}$ |

(**Sumber**: Hasil Perhitungan)

Besar *delay* propagasi pada jaringan HSDPA dipengaruhi oleh jarak antar node dan bit rate CODEC video. Semakin besar jarak antar node serta semakin jauh jarak *Node* B terhadap UE, maka delay propagasi yang dihasilkan akan semakin besar. Selain itu, penggunaan bit rate CODEC video mempengaruhi besar *payload* video streaming. Sehingga, jumlah *frame* yang ditransmisikan akan meningkat seiring dengan penambahan besar *payload* dari video *streaming*. Sehingga, semakin besar bit rate CODEC video yang digunakan maka besar *delay* propagasi yang dihasilkan semakin besar.

#### 4.5.2.4 *Delay* Antrian

*Delay* antrian terjadi pada GGSN, SGSN, RNC, dan *Node* B dengan menggunakan model antrian M/M/1. Pada GGSN digunakan standar *Fast Ethernet*

dengan kecepatan 100 Mbps dengan panjang data 2540 *byte*. Besar laju pelayanan paket data didapatkan sesuai dengan persamaan 2-67

$$\mu_{GGSN} = \frac{C_{GGSN}}{L}$$
$$= \frac{10^8}{2540\,x\,8}$$
$$= 4921,26\ paket/s$$

Pada SGSN digunakan standart *Fast Ethernet* 100 Base-Fx yang memiliki kecepatan data 100 Mbps dengan panjang data sebesar 2809 *byte*. Besar laju pelayanan paket data didapatkan sesuai dengan persamaan 2-67

$$\mu_{SGSN} = \frac{C_{SGSN}}{L}$$
$$= \frac{10^8}{2809\ x\ 8}$$
$$= 4449,98\ paket/s$$

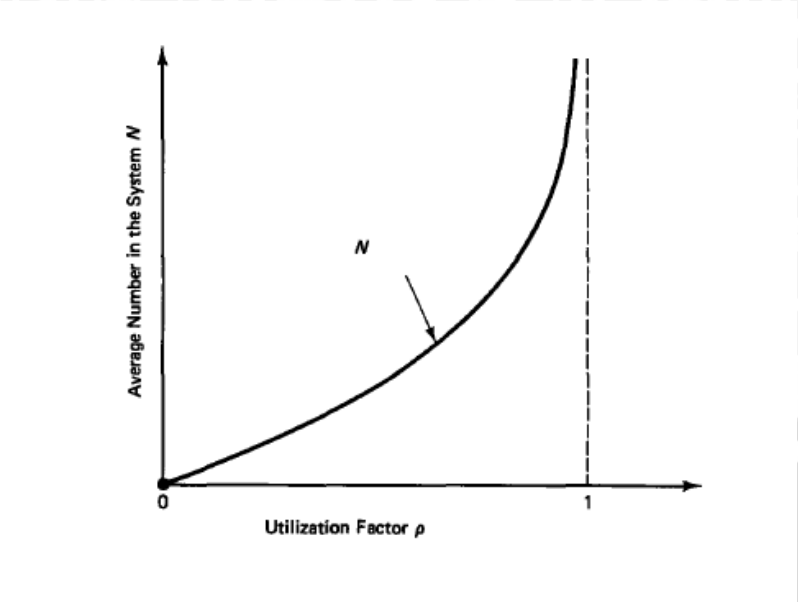
Pada RNC digunakan *interface* sistem transmisi ATM STM-1 yang memberikan kecepatan data sebesar 155,52 Mbps dengan panjang data sebesar 4028 *byte*. Besar laju pelayanan paket data didapatkan sesuai dengan persamaan 2-67

$$\mu_{RNC} = \frac{C_{RNC}}{L}$$
$$= \frac{155,52.10^6}{4028\,x\,8}$$
$$= 4826,22\ paket/s$$

Pada Node *B* digunakan *interface* sistem transmisi ATM STM-1 yang memberikan kecepatan data sebesar 155,52 Mbps dengan panjang data 2636 *byte*. Besar laju pelayanan paket data didapatkan sesuai dengan persamaan 2-67

$$\mu_{Node\ B} = \frac{C_{Node\ B}}{L}$$
$$= \frac{155,52.10^6}{2636\ x\ 8}$$
$$= 7374,81\ paket/s$$

Dalam analisis perhitungan *delay* antrian ini diasumsikan bahwa besar faktor utilisasi (ρ) berubah-ubah antara nilai 0,1 sampai dengan 0,9. Faktor utilisasi yang berubah-ubah mengindikasikan banyaknya *user* yang berada dalam antrian (Dimitri Bertsekas and Robert Gallager, 1987: 129) seperti yang ditunjukkan pada Gambar 4.14




**Gambar 4.14** Hubungan antara Faktor Utilitas terhadap Banyaknya User dalam Sistem Antrian
**(Sumber:** Dimitri Bertsekas and Robert Gallager, 1987: 129**)**

Dari masing-masing nilai ρ dapat ditentukan laju kedatangan paket (λ) sesuai dengan persamaan 2-69

$\lambda_{GGSN} = \mu_{GGSN} \times \rho_{GGSN}$
$= 4921{,}26 \times 0{,}1$
$= 492{,}126$ paket/s

$\lambda_{SGSN} = \mu_{SGSN} \times \rho_{SGSN}$
$= 4449{,}98 \times 0{,}1$
$= 444{,}998$ paket/s

$\lambda_{RNC} = \mu_{RNC} \times \rho$
$= 4826{,}22 \times 0{,}1$
$= 482{,}622$ paket/s

$\lambda_{Node\ B} = \mu_{Node\ B} \times \rho$
$= 7374{,}81 \times 0{,}1$
$= 737{,}481$ paket/s

Kemudian, dapat ditentukan *delay* antrian sesuai dengan persamaan 2-70

$$t_{w1} = \frac{\lambda_{GGSN}}{\mu_{GGSN}(\mu_{GGSN} - \lambda_{GGSN})} + \frac{1}{\mu_{GGSN}}$$
$$= \frac{492,126}{4921,26 .(4921,26 - 492,126)} + \frac{1}{4921,26}$$
$$= 2,2578.10^{-4} s$$

$$t_{w2} = \frac{\lambda_{SGSN}}{\mu_{SGSN}(\mu_{SGSN} - \lambda_{SGSN})} + \frac{1}{\mu_{SGSN}}$$
$$= \frac{444,998}{4449,98 .(4449,98 - 444,998)} + \frac{1}{4449,98}$$
$$= 2,4969.10^{-4} s$$

$$t_{w3} = \frac{\lambda_{RNC}}{\mu_{RNC}(\mu_{RNC} - \lambda_{RNC})} + \frac{1}{\mu_{RNC}}$$
$$= \frac{482,622}{4826,22 .(4826,22 - 482,622)} + \frac{1}{4826,22}$$
$$= 2,3022.10^{-4} s$$

$$t_{w4} = \frac{\lambda_{Node\,B}}{\mu_{Node\,B}(\mu_{Node\,B} - \lambda_{Node\,B})} + \frac{1}{\mu_{Node\,B}}$$
$$= \frac{737,481}{7374,81 .(7374,81 - 737,481)} + \frac{1}{7374,81}$$
$$= 1,5066.10^{-4} s$$

*Delay* antrian total yang terjadi pada jaringan HSDPA dengan faktor utilisasi 0,1 sesuai dengan persamaan 2-70 adalah sebagai berikut:

$t_{wtot} = t_{w1} + t_{w2} + t_{w3} + t_{w4}$
$= 2,2578.10^{-4} + 2,4969.10^{-4} + 2,3022.10^{-4} + 1,5066.10^{-4}$
$= 8,5635.10^{-4}$ s

Hasil analisis *delay* antrian yang terjadi pada GGSN, SGSN, RNC, dan *Node* B terhadap perubahan faktor utilisasi dan berdasarkan *bit rate* video CODEC sebesar 256 kbps, 320 kbps, dan 384 kbps ditunjukkan pada tabel. 4.14

**Tabel 4.14** Hasil Analisis *Delay* Antrian pada Jaringan HSDPA yang Berbasis IPv6

| No. | *Faktor Utilisasi* | Delay Antrian untuk *bit rate* CODEC video 256 kbps (s) | Delay Antrian untuk *bit rate* CODEC video 382 kbps (s) | Delay Antrian untuk *bit rate* CODEC video 384 kbps (s) |
|---|---|---|---|---|
| 1 | 0,1 | $6{,}2751.10^{-4}$ | $7{,}4436.10^{-4}$ | $8{,}5635.10^{-4}$ |
| 2 | 0,2 | $7{,}0595.10^{-4}$ | $8{,}3741.10^{-4}$ | $9{,}6340.10^{-4}$ |
| 3 | 0,3 | $8{,}0680.10^{-4}$ | $9{,}5704.10^{-4}$ | $1{,}1010.10^{-3}$ |
| 4 | 0,4 | $9{,}4127.10^{-4}$ | $1{,}1165.10^{-3}$ | $1{,}2845.10^{-3}$ |
| 5 | 0,5 | $1{,}1295.10^{-3}$ | $1{,}3399.10^{-3}$ | $1{,}5414.10^{-3}$ |
| 6 | 0,6 | $1{,}4119.10^{-3}$ | $1{,}6748.10^{-3}$ | $1{,}9268.10^{-3}$ |
| 7 | 0,7 | $1{,}8825.10^{-3}$ | $2{,}2331.10^{-3}$ | $2{,}5691.10^{-3}$ |
| 8 | 0,8 | $2{,}8238.10^{-3}$ | $3{,}3496.10^{-3}$ | $3{,}8536.10^{-3}$ |
| 9 | 0,9 | $5{,}6476.10^{-3}$ | $6{,}6993.10^{-3}$ | $7{,}7072.10^{-3}$ |

(**Sumber**: Hasil Perhitungan)

Besar *delay* antrian pada jaringan HSDPA dipengaruhi oleh faktor utilitas dan bit rate CODEC video. Besarnya faktor utilitas menunjukkan besar pemakaian user pada *node-node*. Sehingga, semakin besar faktor utilitas maka besar *delay* antrian yang digunakan akan semakin besar.

Selain itu, penggunaan bit rate CODEC video yang semakin besar mempengaruhi besar *payload* dari video streaming. Oleh karena itu, laju pelayanan paket video *streaming* pada setiap *node* akan semakin besar seiring dengan penambahan *payload* video *streaming*. Sehingga, semakin besar bit rate CODEC video maka besar delay antrian yang dihasilkan akan semakin besar pula.

### 4.5.2.5 *Delay End to End*

Perhitungan *delay* proses, *delay* transmisi, *delay* propagasi, dan *delay* antrian akan mendapatkan *delay* jaringan HSDPA. Sesuai dengan persamaan 2-11 didapatkan *delay* jaringan HSDPA yang menggunakan IPv6 dengan *bit rate* video CODEC sebesar 384 kbps dan faktor utilisasi sebesar 0,1, yaitu:

$$t_{net} = t_{proc} + t_{Ttot} + t_{Ptot} + t_{Wtot}$$
$$= 1{,}9303.10^{-3}\text{s} + 2{,}5075.10^{-3}\text{ s} + 6{,}5966.10^{-4}\text{s} + 8{,}5635.10^{-4}\text{ s}$$
$$= 5{,}9538.10^{-3}$$

Sehingga nilai *delay end to end* aplikasi video *streaming* pada jaringan HSDPA yang menggunakan IPv6 dengan *bit rate* video CODEC sebesar 384 kbps dan faktor utilisasi sebesar 0,1 sesuai dengan persamaan 2.10 adalah

$t_{end\ to\ end} = t_{CODEC} + t_{net}$

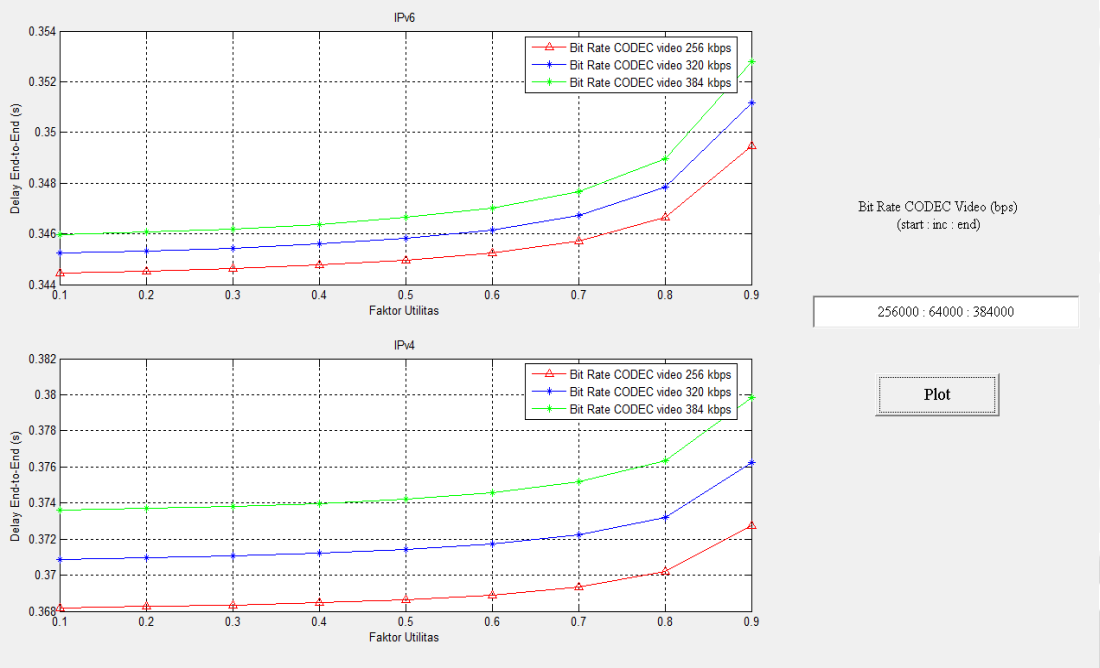
$= 3{,}4.10^{-1} + 5{,}9538.10^{-3} = 345{,}9538$ ms

Berdasarkan hasil analisis di atas, hasil perhitungan *delay end to end* untuk *bit rate* CODEC video sebesar 256 kbps, 320 kbps, dan 384 kbps serta dengan faktor utilisasi sebesar 0,1 sampai dengan 0,9 ditunjukkan pada tabel 4.15. Sedangkan hubungan antara *delay end-to-end* terhadap faktor utilitas dengan bit rate CODEC video yang berubah-ubah dapat dilihat pada Gambar 4.15.

**Tabel 4.15** Hasil Analisis *Delay End to End* Aplikasi Video *Streaming* pada Jaringan HSDPA yang Berbasis IPv6 dengan Bit Rate CODEC Video yang Berubah-ubah

| **No.** | ***Faktor Utilisasi*** | ***Delay End to End* (ms)** | | |
|---|---|---|---|---|
| | | ***Bit Rate* CODEC video 256 kbps** | ***Bit Rate* CODEC video 320 kbps** | ***Bit Rate* CODEC video 384 kbps** |
| 1 | 0,1 | 344,4598 | 345,2278 | 345,9538 |
| 2 | 0,2 | 344,5382 | 345,3208 | 346,0609 |
| 3 | 0,3 | 344,6391 | 345,4405 | 346,1985 |
| 4 | 0,4 | 344,7735 | 345,5999 | 346,3820 |
| 5 | 0,5 | 344,9618 | 345,8233 | 346,6389 |
| 6 | 0,6 | 345,2442 | 346,1582 | 347,0243 |
| 7 | 0,7 | 345,7148 | 346,7165 | 347,6666 |
| 8 | 0,8 | 346,6561 | 347,8331 | 348,9511 |
| 9 | 0,9 | 349,4799 | 351,1827 | 352,8047 |

(**Sumber**: Hasil Perhitungan)



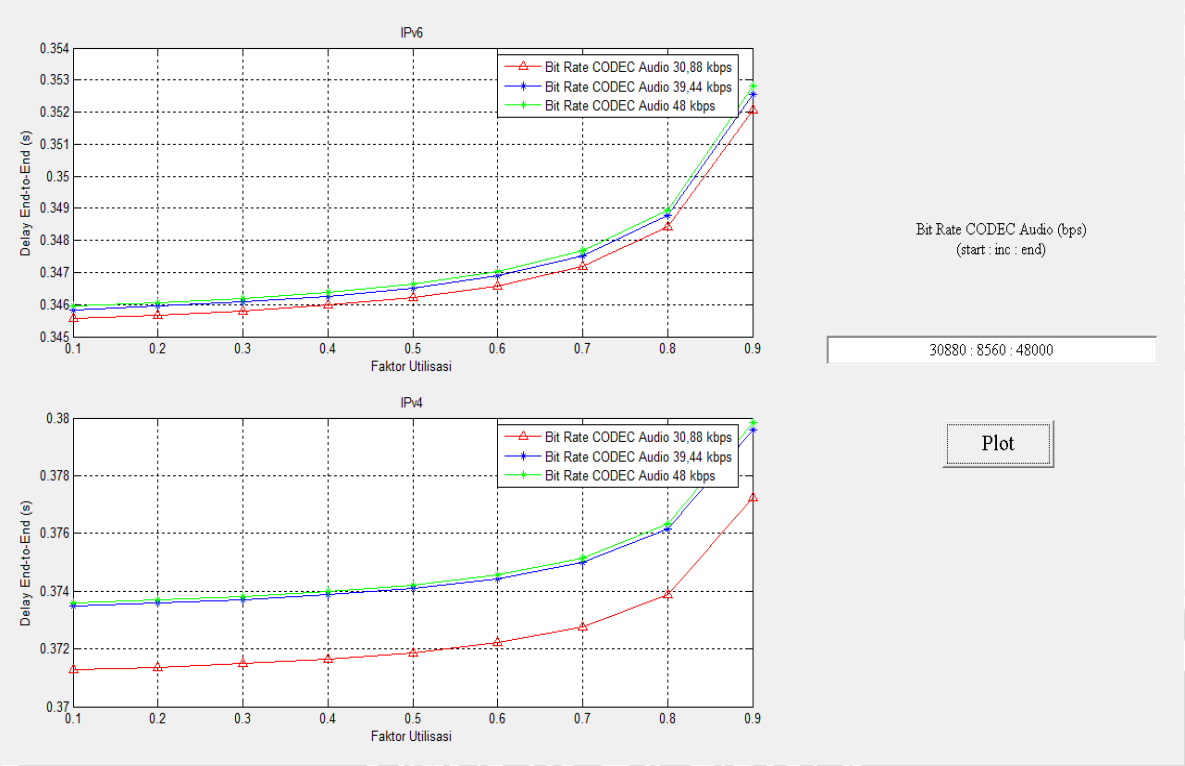

**Gambar 4.15** Hubungan *Delay End-to-End* terhadap Faktor Utilitas dengan Bit Rate CODEC Video yang Berubah-ubah
(**Sumber**: Hasil Perhitungan)

Dengan menggunakan cara yang sama, hasil analisis besar *delay end-to-end* aplikasi video *streaming* dengan *bit rate* CODEC audio yang berubah-ubah dan bit rate CODEC video yang tetap dapat ditunjukkan pada tabel 4.16. Sedangkan hubungan antara *delay end-to-end* terhadap faktor utilitas dengan bit rate CODEC audio yang berubah-ubah dapat dilihat pada Gambar 4.16

**Tabel 4.16** Hasil Analisis *Delay End to End* Aplikasi Video *Streaming* pada Jaringan HSDPA yang Berbasis IPv6 dengan bit rate CODEC audio yang berubah-ubah

| No. | *Faktor Utilisasi* | *Delay End to End* (ms) | | |
|---|---|---|---|---|
| | | ***Bit Rate* CODEC audio 30,88 kbps** | ***Bit Rate* CODEC audio 39,44 kbps** | ***Bit Rate* CODEC video 48 kbps** |
| 1 | 0,1 | 345,5776 | 345,8421 | 345,9539 |
| 2 | 0,2 | 345,6789 | 345,9469 | 346,0609 |
| 3 | 0,3 | 345,8091 | 346,0815 | 346,1985 |
| 4 | 0,4 | 345,9827 | 346,2611 | 346,3820 |
| 5 | 0,5 | 346,2258 | 346,5125 | 346,6389 |
| 6 | 0,6 | 346,5904 | 346,8896 | 347,0243 |
| 7 | 0,7 | 347,1981 | 347,5180 | 347,6666 |
| 8 | 0,8 | 348,4134 | 348,7749 | 348,9511 |
| 9 | 0,9 | 352,0594 | 352,5456 | 352,8047 |

(**Sumber**: Hasil Perhitungan)




**Gambar 4.16** Hubungan *Delay End-to-End* terhadap Faktor Utilitas dengan Bit Rate CODEC Audio yang Berubah-ubah

(**Sumber**: Hasil Perhitungan)

Dari hasil analisis perhitungan dan grafik *delay end-to-end* di atas dapat kita ketahui bahwa:

1. Analisis performansi perhitungan delay end-to-end meliputi delay CODEC dan delay pada jaringan HSDPA. *Delay* CODEC dipengaruhi oleh bit rate CODEC yang digunakan dalam aplikasi video *streaming*. Sedangkan delay jaringan HSDPA meliputi delay proses yang dipengaruhi oleh kecepatan pemrosesan pada pengirim dan penerima, *delay* transmisi yang dipengaruhi oleh kecepatan pemrosesan di setiap node, *delay* propagasi yang dipengaruhi jarak antar *node* dan media propagasi yang digunakan, serta *delay* antrian yang dipengaruhi oleh faktor utilitas pada jaringan.
2. Penggunaan *bit rate* CODEC video maupun audio pada aplikasi video *streaming* mempengaruhi konsumsi *bandwidth* dan besar *payload* dari aplikasi video *streaming*. Oleh karena itu, penggunaan *bit rate* CODEC video maupun audio yang digunakan mempengaruhi besar *delay end-to-end* jaringan HSDPA pada aplikasi video *streaming*.

3. Faktor utilisasi menunjukkan besarnya pemakaian jaringan oleh *user*. Besarnya faktor utilitas mempengaruhi waktu pemrosesan paket di setiap *node.* Hal ini mengakibatkan nilai *delay* antrian menjadi lebih besar sehingga *delay end-to-end* juga menjadi lebih besar.
4. Besar *delay end to end* yang menerapkan IPv4 lebih besar karena adanya penambahan *delay header checksum*.
5. Pada panjang paket data yang sama, semakin besar faktor utilisasi maka semakin besar pula nilai *delay end-to-end.*
6. Besarnya *delay end-to-end* aplikasi video *streaming* pada jaringan HSDPA yang menggunakan IPv6 paling kecil adalah 344,4598 ms, saat menggunakan *bit rate* CODEC video 256 kbps dan bit rate CODEC audio 48 kbps dengan faktor utilitas sebesar 0,1. Sedangkan besarnya *delay end-to-end* terbesar adalah 352,8047 ms yang terjadi pada saat menggunakan *bit rate* CODEC video 384 kbps dan bit rate CODEC audio 48 kbps dengan faktor utilisasi sebesar 0,9.
7. Besarnya nilai *delay end-to-end* aplikasi video *streaming* pada jaringan HSDPA yang menggunakan IPv6 masih memenuhi standard ITU.T G.1010 untuk layanan video *streaming*, yaitu < 10 s.

## 4.6 Analisis Perhitungan Probabilitas *Packet Loss*

Probabilitas *packet loss* untuk aplikasi video *streaming* pada jaringan HSDPA dihitung dari probabilitas *packet loss* pada *server* yang terhubung ke jaringan internet, serta probabilitas *packet loss* pada jaringan HSDPA.

### 4.6.1 Probabilitas *Packet Loss* pada Server

Probabilitas *packet loss* pada server didapatkan sesuai dengan persamaan 2.74

$$\rho_{VS} = P_{VS\text{-}size} \times \rho_b$$

$$= (19136) \times 10^{-8} = 1{,}9136.10^{-4}$$

Hasil analisis nilai *packet loss* aplikasi video *streaming* untuk *bit rate* video CODEC 256 kbps, 320 kbps, dan 384 kbps dapat ditunjukkan pada Tabel 4.17

**Tabel 4.17** Hasil Analisis Probabilitas *Packet Loss* pada Server

| No. | *Bit Rate* CODEC Video | Probabilitas *Packet Loss* Aplikasi Video *Streaming* ($\rho_{VS}$) |
|---|---|---|
| 1 | 256 kbps | $1{,}3936.10^{-4}$ |
| 2 | 320 kbps | $1{,}6536.10^{-4}$ |
| 3 | 384 kbps | $1{,}9136.10^{-4}$ |

(**Sumber**: Hasil Perhitungan)

Besar probabilitas *packet loss* video *streaming* dipengaruhi oleh bit rate CODEC video yang digunakan. Semakin besar bit rate CODEC video yang digunakan, maka besar *payload* video *streaming* yang dihasilkan akan semakin besar. Oleh karena itu, besar paket loss yang digunakan akan semakin besar pula.

### 4.6.2 Probabilitas *Packet Loss* pada Jaringan HSDPA

Probabilitas BER (Pb) pada jaringan HSDPA dengan menggunakan modulasi 16-QAM dapat dihitung dengan persamaan 2-76:

$$P_b = \frac{4\left(\sqrt{M}-1\right)}{\sqrt{M}.\log_2 M}.Q.\left(\sqrt{\frac{3\frac{E_b}{N_o}\log_2 M}{M-1}}\right)$$

Parameter yang diperlukan dalam perhitungan SNR dapat diperoleh menggunakan parameter pada tabel 4.2, yaitu:

PG = 12 dB

NF = 8 dB

$P_t$ = 43 dBm

$L_t$ = 2 dB

$L_r$ = 0 dB

$G_t$ = 18 dBi

$G_r$ = 2 dBi

Besar daya noise dapat diperoleh menggunakan persamaan 2-81 sebagai berikut:

$$N = 10\log k.T + 10\log B + NF$$

= 10 log (1,381 x $10^{-23}$ x 300) + 10 log (5 x $10^{6}$) + 8

= -203,827 + 66,989 + 8

= -128,837 dBm

Selanjutnya, nilai daya yang diterima dapat diperoleh sebagai berikut:

$$\mathrm{FSL} = 20\log\left(\frac{4\pi d}{\lambda}\right)$$

Dengan $\lambda = \frac{c}{f}$

$$= \frac{3\times10^{8}}{2{,}1\times10^{9}}$$

= 0,14 m

Maka akan diperoleh nilai *free space loss* (FSL) sebagai berikut :

$$\mathrm{FSL} = 20\log\left(\frac{4\,\mathrm{x}\,3{,}14\,\mathrm{x}\,1500}{0{,}14}\right)$$

= 102,5719 dB

Selanjutnya, nilai daya yang diterima UE adalah sebagai berikut:

$$P_r(dBm) = P_t - FSL - L_t - L_r + G_r + G_t$$

$$\mathrm{P_r\,(dBm)} = 43 - 102{,}5719 - 2 - 0 + 18 + 2$$

= - 41,5719 dBm

Sehingga, besar SNR dapat diperoleh sebagai berikut:

$$SNR = P_r - N$$

= -41,5719 – (-128,837)

= 87,2653 dB

Besar Eb/No dapat diperoleh sebagai berikut:

$$\frac{\mathrm{Eb}}{\mathrm{No}}(\mathrm{dB}) = \frac{\mathrm{S}}{\mathrm{N}} - 10\log\frac{\mathrm{B}}{\mathrm{R}}$$

$$= 87{,}2653 - 10\log\frac{5.10^{6}}{3{,}6.10^{6}}$$

$$= 85{,}8386\ \mathrm{dB}$$

Perhitungan probabilitas BER pada jaringan HSDPA dengan modulasi 16-QAM dapat diperoleh sebagai berikut:

$$P_b = \frac{4\ \sqrt{M}-1}{\sqrt{M}.\log_2 M}.Q.\left(\sqrt{3\frac{E_b}{N_o}\frac{\log_2 M}{M-1}}\right)$$

$$= \frac{4(\sqrt{16}-1)}{\sqrt{16}.\log_2 16}.\frac{1}{2}erfc\left(\sqrt{\frac{3.\frac{Eb}{No}.\log_2 16}{\frac{16-1}{2}}}\right)$$

$$= 4,3630.10^{-17}$$

Sehingga, besar probabilitas packet loss pada Jaringan HSDPA dapat dihitung sebagai berikut:

$\rho_{HSDPA} = P_{VStot}$ x $\rho_{b\text{-}HSDPA}$

$= (21088)$ x $4,3630.10^{-17}$

$= 9,2007.10^{-13}$

Besar probabilitas *packet loss* pada jaringan HSDPA dipengaruhi oleh jenis modulasi yang digunakan. HSDPA dapat menggunakan dua jenis modulasi, yaitu QPSK dan 16-QAM. Pada skripsi ini, jenis modulasi yang digunakan adalah 16-QAM dengan nilai M = $2^4$ = 16. Apabila jenis modulasi yang digunakan adalah QPSK, nilai M = $2^2$ = 4. Semakin besar ukuran konstelasi sinyal (M), maka besar *packet loss* yang dihasilkan akan semakin kecil.

### 4.6.3. Probabilitas *Packet Loss* Total

Probabilitas *packet loss* untuk aplikasi video *streaming* pada jaringan HSDPA dapat diperoleh menggunakan persamaan 2.72

$\rho_{tot}$ $= 1 - [(1 - \rho_{network})(1 - \rho_{VS})]$

$= 1 - [(1 - 9,2007.10^{-13})(1 - 1,3936.10^{-4})]$

$= 1,9136.10^{-4}$

Prosentase *packet loss* dihitung sebagai berikut:

$$\text{packet loss}(\%) = \frac{N_{\text{packet loss}}}{N_{\text{paket}} + N_{\text{packet loss}}} \text{x} 100\,\%$$

$$= \frac{5}{21088} \text{x} 100\%$$

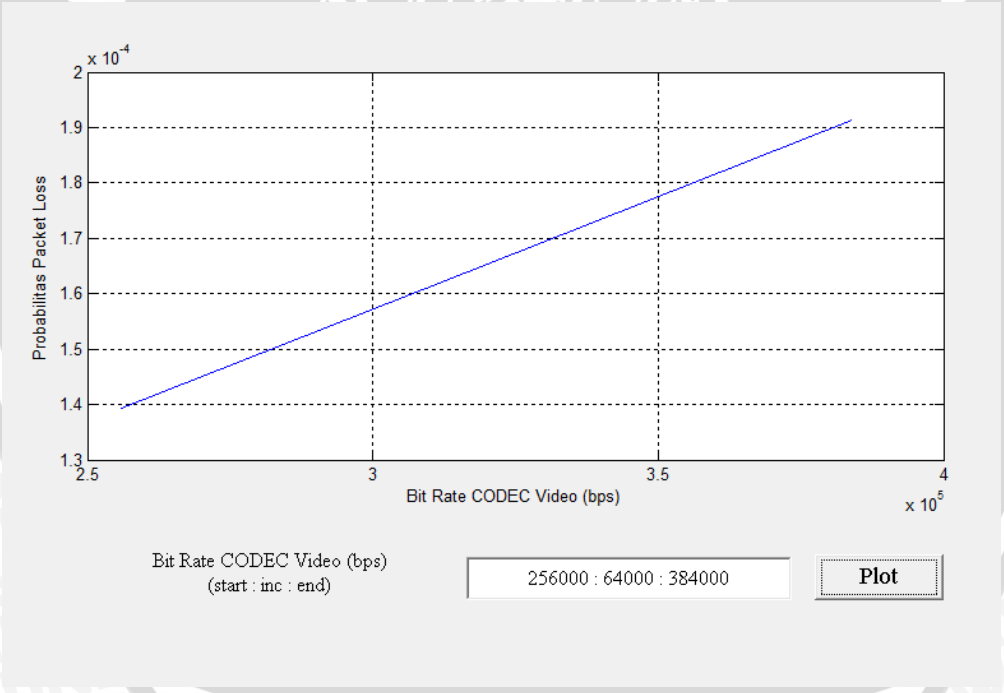
$$= 0,0237\%$$

Menggunakan cara yang sama, akan didapatkan besar probabilitas *packet loss* total untuk *bit rate* CODEC video sebesar 256 kbps, 320 kbps, dan 384 kbps yang ditunjukkan pada tabel 4.18. Sedangkan hubungan antara probabilitas packet loss dan bit rate CODEC video dapat dilihat pada Gambar 4.17

**Tabel 4.18** Hasil Analisis Probabilitas *Packet Loss* Aplikasi Video *Streaming* pada Jaringan HSDPA yang Berbasis IPv6 dengan Bit Rate CODEC Video yang Berubah-ubah

| No. | *Bit Rate* CODEC Video | Probabilitas *Packet Loss* Total |
|---|---|---|
| 1 | 256 kbps | $1{,}3936.10^{-4}$ |
| 2 | 320 kbps | $1{,}6536.10^{-4}$ |
| 3 | 384 kbps | $1{,}9136.10^{-4}$ |

(**Sumber**: Hasil Perhitungan)




**Gambar 4.17** Hubungan Probabilitas *Packet Loss* terhadap Bit Rate CODEC Video (**Sumber**: Hasil Perhitungan)

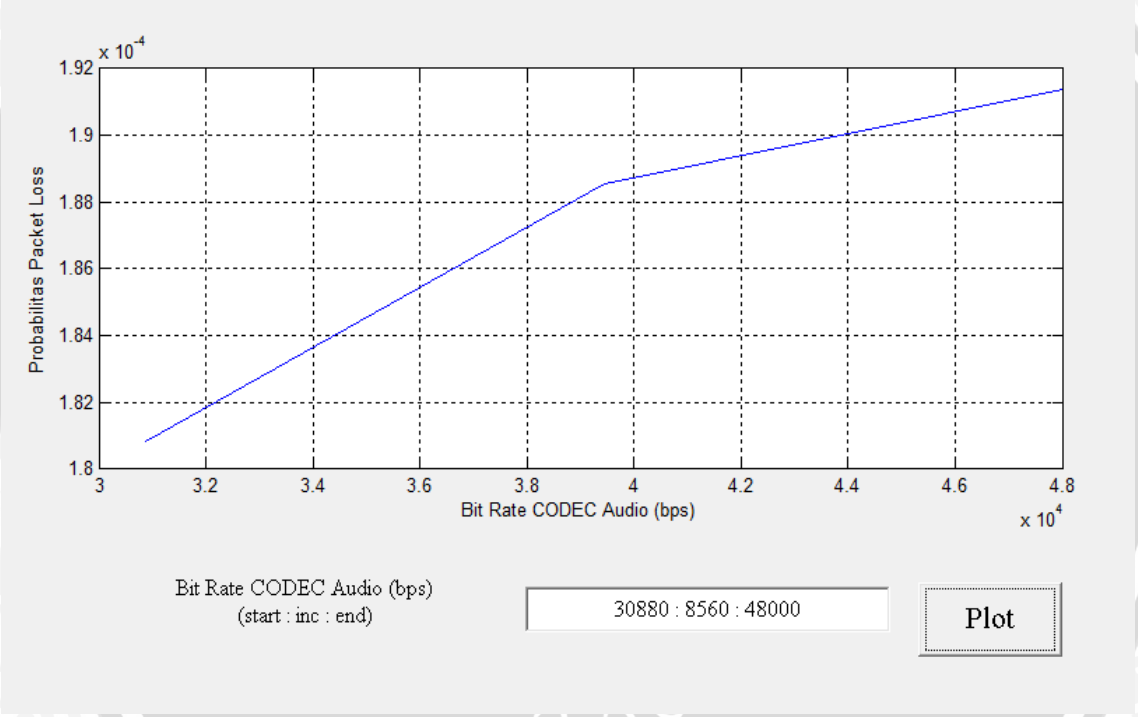
Dengan menggunakan cara yang sama akan didapatkan besar probabilitas *packet loss* total untuk *bit rate* CODEC audio berubah-ubah sebesar 30,88 kbps, 39,44 kbps, dan 48 kbps dengan bit rate CODEC video tetap sebesar 384 kbps serta faktor utilitas antara 0,1 sampai dengan 0,9 yang ditunjukkan pada tabel 4.19. Sedangkan hubungan

antara probabilitas packet loss dan bit rate CODEC video dapat dilihat pada Gambar 4.18

**Tabel 4.19** Hasil Analisis Probabilitas *Packet Loss* Aplikasi Video *Streaming* pada Jaringan HSDPA yang Menggunakan IPv6 dengan bit rate CODEC audio yang berubah-ubah

| No. | *Bit Rate* CODEC Audio | Probabilitas *Packet Loss* Total |
|---|---|---|
| 1 | 30,88 kbps | $1{,}8083.10^{-4}$ |
| 2 | 39,44 kbps | $1{,}8854.10^{-4}$ |
| 3 | 48 kbps | $1{,}9136.10^{-4}$ |

(**Sumber**: Hasil Perhitungan)




**Gambar 4.18** Hubungan Probabilitas *Packet Loss* terhadap Bit Rate CODEC Audio (**Sumber**: Hasil Perhitungan)

Dari analisis perhitungan dan grafik *packet loss* di atas, dapat kita ketahui bahwa:

1. Analisis probabilitas *packet loss* aplikasi video streaming meliputi probabilitas *packet loss* pada server yang terhubung melalui internet yang dipengaruhi oleh bit rate CODEC yang digunakan serta probabilitas *packet loss* pada jaringan HSDPA yang dipengaruhi oleh SNR, Eb/No, dan jenis modulasi yang digunakan.

2. Kualitas sinyal terima yang baik didapatkan jika nilai SNR dan Eb/No yang semakin besar sehingga probabilitas packet loss yang didapatkan akan semakin kecil.
3. Bit rate CODEC yang digunakan mempengaruhi besar *payload* pada aplikasi video *streaming*. Seiring dengan banyaknya paket data yang ditransmisikan, probabilitas *packet loss* yang dihasilkan akan semakin besar
4. Besar probabilitas packet loss yang didapatkan masih dapat diterima karena tidak melebihi 1% (ITU.T G.1010)

### 4.7 Analisis *Throughput*

*Throughput* merupakan parameter yang digunakan untuk mengetahui jumlah paket yang diterima dalam keadaan baik terhadap waktu total transmisi yang dibutuhkan dari *server* hingga ke *user*. Analisis perhitungan *throughput* yang dilakukan dalam skripsi ini adalah hubungan antara *server* hingga ke UE.

Waktu transmisi *frame* untuk *bit rate* CODEC video 384 kbps dapat diperoleh dengan persamaan 2.87:

$$t_l = \frac{Pl_{frame} + H_{frame} \; x8}{C_{trans}}$$
$$= \frac{2636 \; byte \, x8}{3,6.10^{6}}$$
$$= 5,8578.10^{-3} \; s$$

Sedangkan, *delay* propagasi total dan *delay* proses total untuk satu *frame* pada *bit rate* CODEC video 320 kbps dengan faktor utilisasi 0,1 sesuai dengan persamaan 2.89 dan 2.90 adalah:

$$t_p = \frac{t_{Ptot}}{N_{frame}}$$
$$= \frac{6,5966.10^{-4}}{1}$$
$$= 6,5966.10^{-4} \; s$$

$$t_{pro} = \frac{t_{proc}}{N_{frame}} + \frac{t_{wtot}}{N_{frame}}$$
$$= \frac{1,9303.10^{-3}}{1} + \frac{8,5635.10^{-4}}{1}$$
$$= 2,7867.10^{-3} s$$

Sehingga *fixed timed out interval* dapat diperoleh sesuai dengan persamaan 2.88 sebesar:

$$t_{out} = 2t_p + 2t_l + t_{pro}$$
$$= (2 \times 6{,}5955.10^{-4}\ s) + (2 \times 5{,}8578.10^{-3}\ s) + 2{,}7867.10^{-3}$$
$$= 1{,}5821.10^{-2}\ s$$

Maka konstanta α dapat diperoleh sesuai dengan persamaan 2.86, yaitu:

$$\alpha = 1 + \frac{t_{out}}{t_l}$$
$$= 1 + \frac{1{,}5821.10^{-2}}{5{,}8578.10^{-3}}$$
$$= 3{,}7008$$

Sehingga besarnya throughput aplikasi video *streaming* pada jaringan HSDPA yang menggunakan IPv6 pada *bit rate* CODEC video 320 kbps dan faktor utilisasi sebesar 0,1 dapat diperoleh sesuai dengan persamaan 2.84 sebesar:

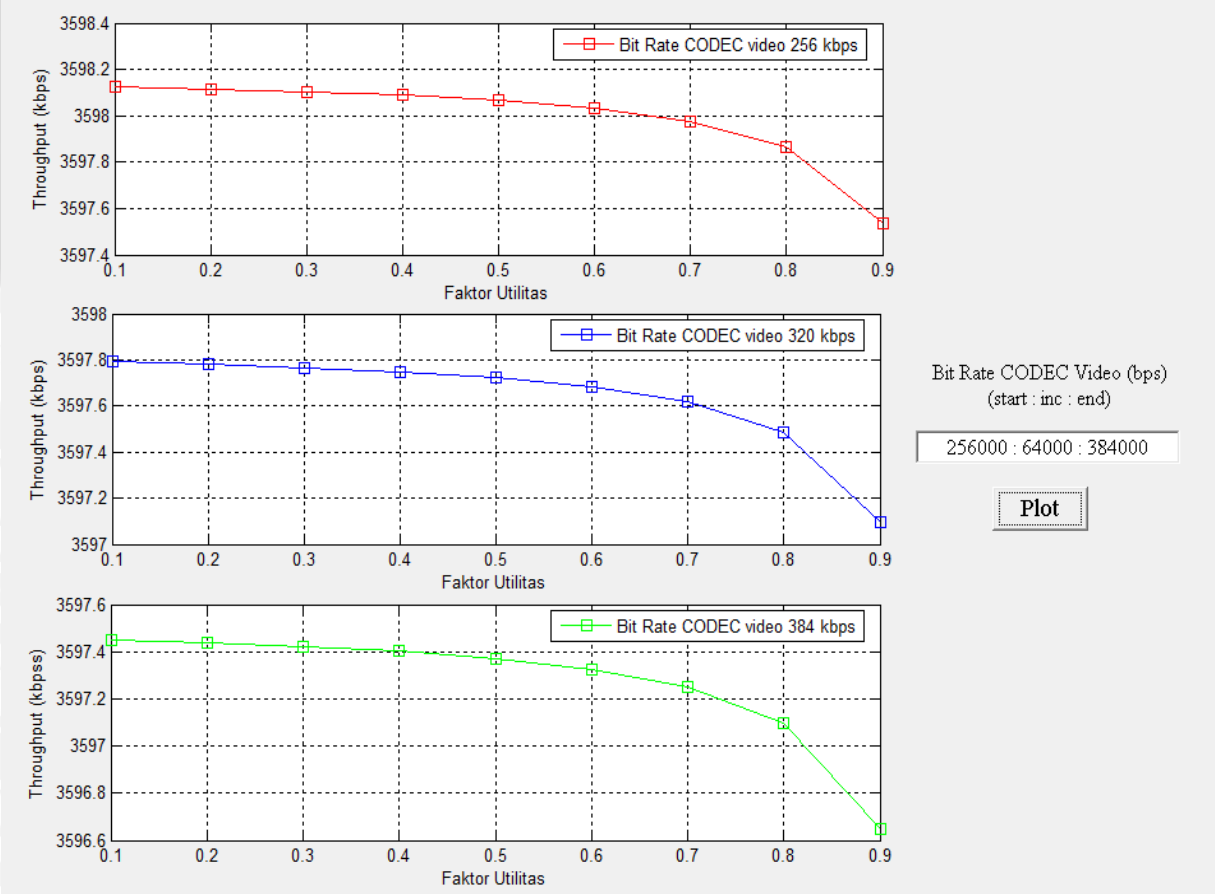
$$T = \frac{(1-\rho_{tot})}{t_1[1+(\alpha-1)\rho_{tot}]}$$
$$= \frac{(1-1{,}9136.10^{-4})}{5{,}8578.10^{-3}[1+(3.7008-1).1{,}9136.10^{-4}]}$$
$$= 170{,}5924\ x\ 2636\ x 8\ bps$$
$$= 3597{,}4517\ kbps$$

Menggunakan cara yang sama akan didapatkan besar *throughput* untuk *bit rate* CODEC video berubah-ubah sebesar 256 kbps, 320 kbps, dan 384 kbps dengan bit rate CODEC audio tetap sebesar 48 kbps serta faktor utilitas antara 0,1 sampai dengan 0,9 yang ditunjukkan pada tabel 4.20. Sedangkan hubungan throughput terhadap faktor utilitas serta probabilitas *packet loss* dengan bit rate CODEC video yang berubah-ubah dapat dilihat pada Gambar 4.19 dan Gambar 4.20.

**Tabel 4.20** Hasil Analisis *Throughput* Aplikasi Video *Streaming* pada Jaringan HSDPA yang Berbasis IPv6 dengan bit rate CODEC Video yang Berubah-ubah

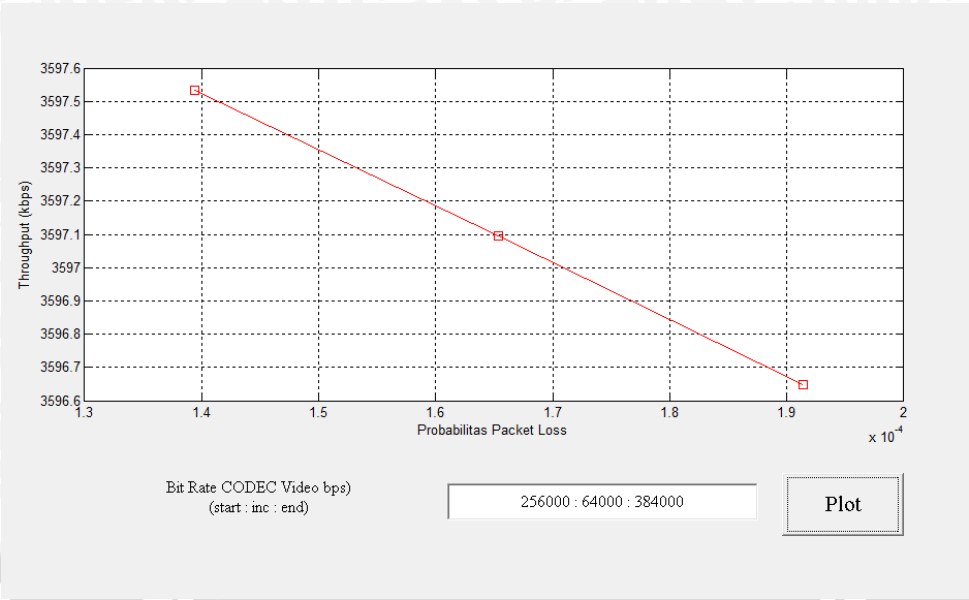
| No. | Faktor Utilisasi | *Throughput* (kbps) | | |
|---|---|---|---|---|
| | | ***Bit Rate* CODEC video 256 kbps** | ***Bit Rate* CODEC video 320 kbps** | ***Bit Rate* CODEC video 384 kbps** |
| 1 | 0,1 | 3598,1258 | 3597,7907 | 3597,4517 |
| 2 | 0,2 | 3598,1165 | 3597,7799 | 3597,4392 |
| 3 | 0,3 | 3598,1046 | 3597,7659 | 3597,4230 |
| 4 | 0,4 | 3598,0887 | 3597,7472 | 3597,7472 |
| 5 | 0,5 | 3598,0665 | 3597,7211 | 3597,7211 |
| 6 | 0,6 | 3598,0331 | 3597,6820 | 3597,3261 |
| 7 | 0,7 | 3597,9776 | 3597,6167 | 3597,2506 |
| 8 | 0,8 | 3597,8664 | 3597,4862 | 3597,0997 |
| 9 | 0,9 | 3597,5330 | 3597,0947 | 3596,6472 |

(**Sumber**: Hasil Pehitungan)




**Gambar 4.19** Hubungan Throughput terhadap Faktor Utilitas dengan Bit Rate CODEC Video yang berubah-ubah

(**Sumber**: Hasil Perhitungan)



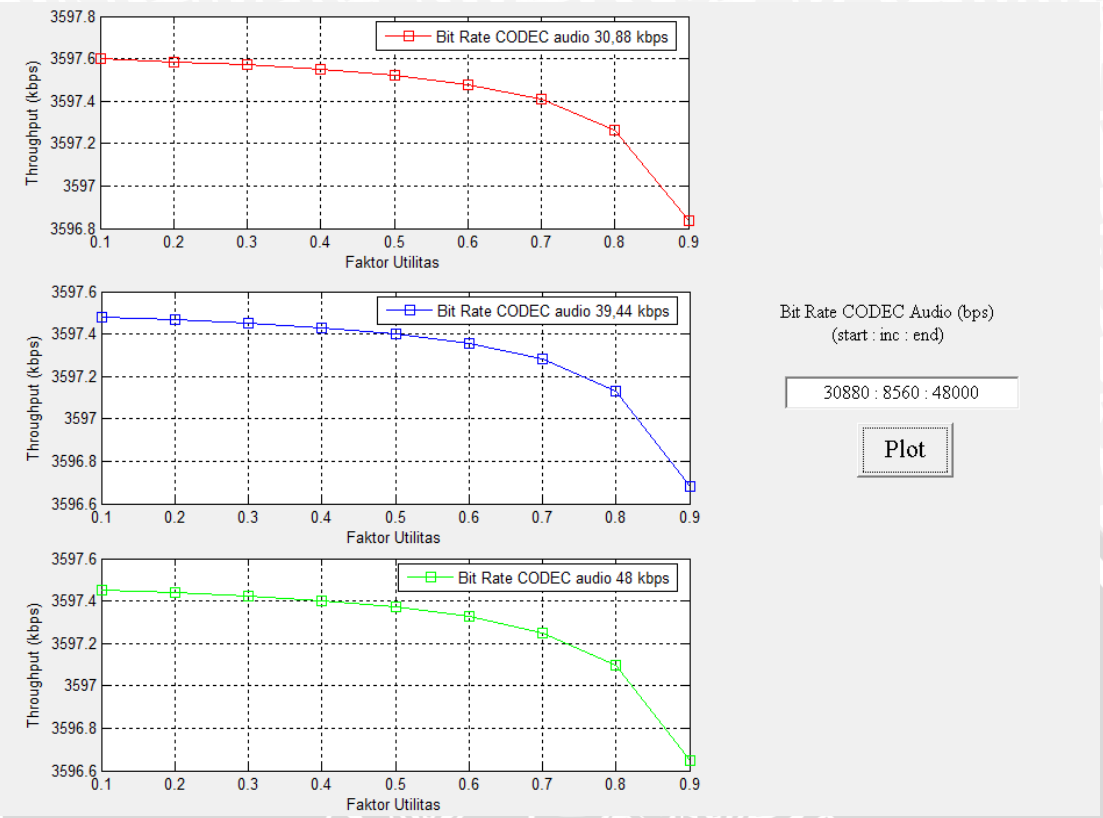

**Gambar 4.20** Hubungan Throughput terhadap Probabilitas *Packet Loss* dengan Bit Rate CODEC Video yang berubah-ubah
(**Sumber**: Hasil Perhitungan)

Dengan menggunakan cara yang sama akan didapatkan besar *throughput* untuk *bit rate* CODEC audio berubah-ubah sebesar 30,88 kbps, 39,44 kbps, dan 48 kbps dengan bit rate CODEC video tetap sebesar 384 kbps serta faktor utilitas antara 0,1 sampai dengan 0,9 yang ditunjukkan pada tabel 4.21. Sedangkan hubungan throughput terhadap faktor utilitas dan probabilitas *packet loss* dengan bit rate CODEC video yang berubah-ubah dapat dilihat pada Gambar 4.21 dan Gambar 4.22.

**Tabel 4.21** Hasil Analisis *Throughput* Aplikasi Video *Streaming* pada Jaringan HSDPA yang Menggunakan IPv6 dengan bit rate CODEC audio yang berubah-ubah

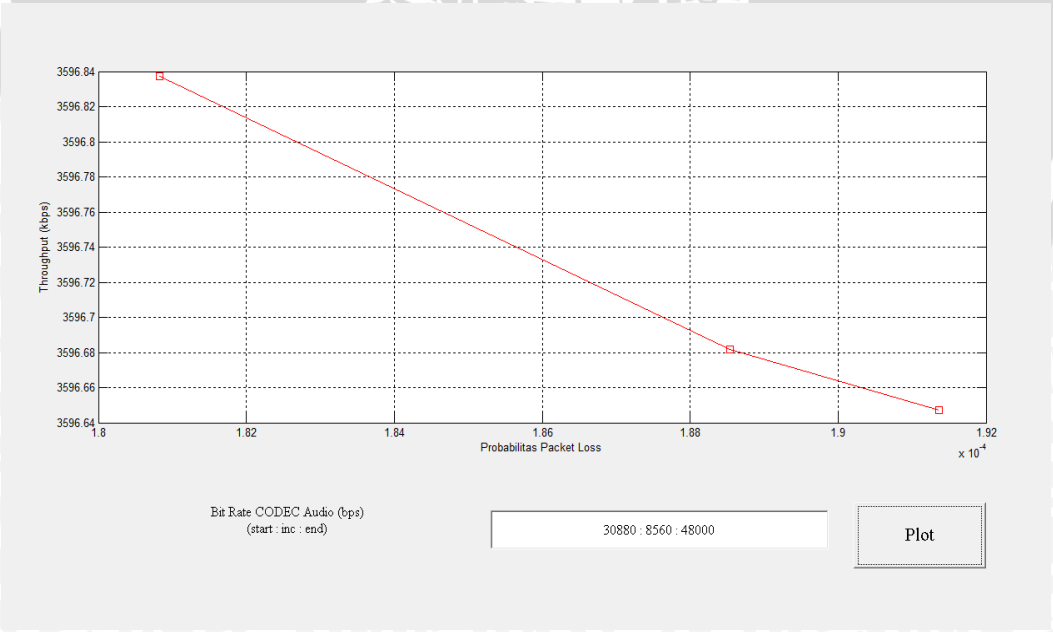
| **No.** | ***Faktor Utilisas*** | ***Throughput* (kbps)** | | |
|---|---|---|---|---|
| | | ***Bit Rate* CODEC audio 30,88 kbps** | ***Bit Rate* CODEC audio 39,44 kbps** | ***Bit Rate* CODEC audio 48 kbps** |
| 1 | 0,1 | 3597,5981 | 3597,4812 | 3597,4517 |
| 2 | 0,2 | 3597,5863 | 3597,4687 | 3597,4392 |
| 3 | 0,3 | 3597,5710 | 3597,4526 | 3597,4230 |
| 4 | 0,4 | 3597,5506 | 3597,4312 | 3597,7472 |
| 5 | 0,5 | 3597,5221 | 3597,4012 | 3597,7211 |
| 6 | 0,6 | 3597,4793 | 3597,3563 | 3597,3261 |
| 7 | 0,7 | 3597,4080 | 3597,2813 | 3597,2506 |
| 8 | 0,8 | 3597,2653 | 3597,1314 | 3597,0997 |
| 9 | 0,9 | 3596,8375 | 3596,6816 | 3596,6472 |

(**Sumber**: Hasil Perhitungan)




**Gambar 4.21** Hubungan Throughput terhadap Faktor Utilitas dengan bit rate CODEC audio yang berubah-ubah

(**Sumber**: Hasil Perhitungan)




**Gambar 4.22** Hubungan Throughput terhadap Probabilitas *Packet Loss* dengan bit rate CODEC audio yang berubah-ubah

(**Sumber**: Hasil Perhitungan)

Dari analisis perhitungan dan grafik *throughput,* di atas dapat kita ketahui bahwa:

1. Besarnya *throughput* aplikasi video *streaming* pada jaringan HSDPA yang menggunakan IPv6 dipengaruhi oleh probabilitas packet loss, faktor utilitasi, seta bit rate CODEC yang digunakan.
2. Probabilitas packet loss merupakan banyaknya kerusakan data yang terjadi saat sampai di penerima. Sehingga, throughput yang dihasilkan akan semakin kecil seiring dengan meningkatnya nilai probabilitas *packet loss* pada aplikasi video *streaming*.
3. Besar faktor utilisasi menunjukkan besar pemakaian jaringan oleh user. Besarnya faktor utilitas yang semakin besar akan mempengaruhi waktu pemrosesan paket data. Hal ini mengakibatkan besar *throughput* akan semakin kecil seiring dengan meningkatnya nilai faktor utilisasi.
4. Besar *throughput* terbesar adalah 2933,7168 kbps yang terjadi saat menggunakan *bit rate* CODEC video 128 kbps dan bit rate CODEC audio sebesar 48 kbps dengan faktor utilisasi sebesar 0,1. Sedangkan besar *throughput* terkecil adalah 2594,2236 kbps yang terjadi saat menggunakan *bit rate* CODEC video 384 kbps dan bit rate CODEC audio sebesar 30,88 kbps dengan faktor utilisasi sebesar 0,9.

# BAB V
# KESIMPULAN

## 5.1 Kesimpulan

Berdasarkan perhitungan dan analisis tentang performansi video *streaming* pada jaringan HSDPA yang menggunakan IPv6, maka dapat diperoleh kesimpulan sebagai berikut:

1. Penggunaan bit rate CODEC video berpengaruh terhadap konsumsi bandwidth, delay end-to-end, packet loss, serta throughput pada aplikasi video streaming.
2. Berdasarkan hasil analisis *bandwidth* aplikasi video *streaming* pada jaringan HSDPA yang menggunakan IPv6, dapat disimpulkan bahwa:
   a. *Bandwidth* aplikasi video *streaming* paling besar adalah 566,2879 kbps dengan menggunakan *bit rate* CODEC video sebesar 384 kbps dan bit rate CODEC audio sebesar 48 kbps. Sedangkan *bandwidth* aplikasi video *streaming* paling kecil sebesar 185,9394 kbps saat *bit rate* CODEC video sebesar 64 kbps dan bit rate CODEC audio sebesar 48 kbps.
   b. Besarnya bandwidth aplikasi video *streaming* dipengaruhi oleh *frame rate* dan bit rate CODEC audio maupun video.
3. Berdasarkan hasil analisis *delay end-to-end* aplikasi video *streaming* pada jaringan HSDPA yang menggunakan IPv6, dapat disimpulkan bahwa:
   a. Besarnya *delay end-to-end* aplikasi video *streaming* pada jaringan HSDPA yang menggunakan IPv6 paling kecil adalah 344,4598 ms, saat menggunakan *bit rate* CODEC video 256 kbps dan bit rate CODEC audio 48 kbps dengan faktor utilitas sebesar 0,1. Sedangkan besarnya *delay end-to-end* terbesar adalah 352,8047 ms yang terjadi pada saat menggunakan *bit rate* CODEC video 384 kbps dan bit rate CODEC audio 48 kbps dengan faktor utilisasi sebesar 0,9.
   b. Nilai *delay end-to-end* aplikasi video *streaming* pada jaringan HSDPA yang menggunakan IPv6 masih memenuhi standard ITU.T G.1010 untuk layanan video *streaming*, yaitu < 10 s.
   c. Penggunaan *bit rate* CODEC video maupun audio serta faktor utilitas mempengaruhi besarnya *delay end-to-end* aplikasi video *streaming*. Pada

panjang paket data yang sama, semakin besar faktor utilisasi maka semakin besar pula nilai *delay end-to-end.*

4. Berdasarkan hasil analisis probabilitas *packet loss* aplikasi video *streaming* pada jaringan HSDPA yang menggunakan IPv6, dapat disimpulkan bahwa:
   a. Besar probabilitas packet loss yang didapatkan masih dapat diterima karena tidak melebihi 1% (ITU.T G.1010)
   b. Analisis probabilitas *packet loss* aplikasi video streaming meliputi probabilitas *packet loss* pada server yang terhubung melalui internet yang dipengaruhi oleh bit rate CODEC yang digunakan serta probabilitas *packet loss* pada jaringan HSDPA yang dipengaruhi oleh SNR, Eb/No, dan jenis modulasi yang digunakan.
5. Berdasarkan hasil analisis probabilitas *throughput* aplikasi video *streaming* pada jaringan HSDPA yang menggunakan IPv6, dapat disimpulkan bahwa:
   a. Besar *throughput* terbesar adalah 3598,1258 kbps yang terjadi saat menggunakan *bit rate* CODEC video 256 kbps dan bit rate CODEC audio sebesar 48 kbps dengan faktor utilisasi sebesar 0,1. Sedangkan besar *throughput* terkecil adalah 3596,6472 kbps yang terjadi saat menggunakan *bit rate* CODEC video 384 kbps dan bit rate CODEC audio sebesar 48 kbps dengan faktor utilisasi sebesar 0,9.
   b. Besarnya *throughput* aplikasi video *streaming* pada jaringan HSDPA yang menggunakan IPv6 dipengaruhi oleh probabilitas packet loss, faktor utilitas, serta bit rate CODEC yang digunakan.

## 5.2 Saran

Saran yang diberikan berdasarkan analisis yang telah dilakukan pada skripsi ini adalah:

1. Menganalisis performansi video streaming pada jaringan yang lain seperti LTE (*Long Term Evolution*), WiBro, dan lain-lain.
2. Skripsi ini dapat dikembangkan dengan membahas dan mempertimbangkan *frame rate*, *frame resolution*, dan *color depth* pada aplikasi video *streaming*.

# DAFTAR PUSTAKA


3GPP TR 25.855 v2.0.0. 2001. *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; High Speed Downlink Packet Access; Overall UTRAN Description (Release 5).* 3GPP Organizational Partners.

3GPP TS 25.323 V5.10.0. 2007. *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Packet Data Convergence Protocol (PDCP) specification (Release 5).* France: 3GPP Organizational Partners.

Ahmad, Ashraf M. A and Ismail K. A. 2009. Multimedia Transcoding in Mobile and Wireless Networks. United Kingdom: IGI Global.

Andreas Handojo, dkk. 2009. *Aplikasi Video Conference dengan Kemampuan Beroperasi IPv4 dan IPv6.*

Anonim, 1998. *Streaming White Paper*. Compaq Computer Corporation.

Anonim. 2004. *HSDPA by Siemens White Paper*. Siemens

Apostopolous, John G. 2001. *Video Communications and Video Streaming*. Hewlett-Packard Laboratories: Streaming Media Systems Group.

Assad, Muhammad and Djamal Zeghlache. 2007. *TCP Performance over UMTS HSDPA System*. Taylor & francis Group, LLC.

Bertsekas, Dimitri and Robert Gallager. 1987. *Data Network.* New Jersey: Prentice Hall, Inc.

Blokzijl, Rob. 2009. *IPv4 Header vs IPv6 Header*. Amstredam: RIPE NCC Roundtable Meeting. http://www.ripe.net/meetings/roundtable/feb2009/presentations/Rob-Blokzijl-roundtable2009-Rob-2.pdf (diakses tanggal 15 Februari 2010)

Forouzan, Behrouz A. 2004. *Data Communication and Networking.* Singapore: Mc-Graw Hill Inc.

Goldsmith, Andrea. 2005. *Wireless Communication*. Cambridge University Press.

Gough, Michael. 2006. *Video Conferencing over IP: Configure, Secure, and Troubleshoot*. Canada: Syngress Publishing, Inc

Guo, Y. Jay. 2004. *Advances in mobile radio access networks*. Artech House, Inc.

Hagen, Silvia. 2002. *IPv6 Essentials*. United States of America: O'Reilly Media, Inc.

Holma, Harri and Antii Toskala. 2006. *HSDPA/HSUPA for UMTS: High Speed RadioAccess for Mobile Communication*. John Wiley and Sons, Ltd.

Holma, Harri and Antii Toskala. 2004. *WCDMA for UMTS*. John Wiley and Sons, Ltd.

http://www.scribd.com/doc/19864169/HSDPA-THESIS

http://www.ittelkom.ac.id/library

Iain E. G. Richardson. 2002. *Video CODEC Design*. England: John Wiley & Sons Ltd.

ITU.T Recommendation G.1010. 2001. *End-user Multimedia QoS Categories*. ITU-T Study Group 12.

John, Ajai. 2007. *HSDPA Thesis*. BM II College of Engineering.

Laili Aidi, 2008. *Analisis dan Simulasi Channel Switching pada Mobile Live Multi-Channel TV Streaming*. Skripsi tidak diterbitkan. Bandung: Sekolah Tinggi Teknik Telkom.

Lo, Anthony. 2004. *Performance Issue of TCP and MPEG-4 over UMTS.*

Proakis, John. G. 2000. *Digital Communications*. McGraw Hill Higher Education.

Purbo, Onno W. 1998. Buku Pintar Internet: TCP/IP. Jakarta: Elex Media Komputindo.

Retnoningsih, Linda Ekowati. 2008. *Penerapan HSDPA (High Speed Downlink Packet Access) pada Jaringan WCDMA*. Skripsi tidak diterbitkan. Malang: Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya.

Riza, Taufan. 2002. Teori dan Implementasi IPv6 Protokol internet Masa Depan. Jakarta: PT. Elex Media Komputindo.

Rohmat Arif C. B. 2006. *Rancang Bangun Perangkat Lunak Sistem Informasi Bioskop Berbasis Web dengan Teknologi OO4O*. Skripsi tidak diterbitkan. Surabaya: Politeknik Elektronika Negeri Surabaya-ITS.

RSYAVY Research. 2007. *EDGE, HSPA, LTE: The Mobile Broadband Advantage*. 3G Americas

S. Deering and R. Hinden. 1998. *RFC2460 - Internet Protocol, Version 6 (IPv6) Specification The Internet Society*.

Schwartz, Mischa. 1987. *Telecommunication Network.* Addison-Wesley.

Simpson, Wes. 2008. *Video Over IP Second Edition*. Elsevier, Inc.

Sjoberg, J, et al. 2006. *RFC 4352: RTP Payload Format for the Extended Adaptive Multi-Rate Wideband (AMR-WB+) Audio Codec*.

Smith, Clinth and Daniel Collins. 2002. 3G Wireless Network. United States of America: McGraw-Hill Companies, Inc.

Sukaridhoto, Sritrusta. 2008. *Buku Jaringan Komputer 2*. http://dhoto.wordpress.com/2008/07/23/buku-jaringan-komputer-2/ (diakses tanggal 11 Januari 2010)

Winch, Robert G. 1998. Telecommunication Transmission System. Unites Stated of America: McGraw-Hill Companies, Inc.

Wenger, S, et al. 2005. *RFC 3984: RTP Payload Format for H.264 Video*.

# LAMPIRAN - LAMPIRAN

**Lampiran 1.** *Listing Program* Matlab Perhitungan *Bandwidth* Aplikasi Video *Streaming*

```
function varargout = bandwidth2(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @bandwidth2_OpeningFcn, ...
                   'gui_OutputFcn',  @bandwidth2_OutputFcn, ...
                   'gui_LayoutFcn',  [], ...
                   'gui_Callback',   []);
if nargin & isstr(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    varargout{1:nargout} = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
handles.output = hObject;
guidata(hObject, handles);
function varargout = bandwidth2_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
function varargout = plot_button_Callback(h, eventdata, handles, varargin)
% Get user input from GUI
Bcodecv = eval(get(handles.Bcodecv,'String'));
% Calculate packet data
Bcodeca = 48000;
fr = 0.033;
H_NALU = 8;
H_RTP = 96;
headerUDP = 8;
headerIPv6 = 40;
max_PLv = 2032;
max_PLa = 640;
PLv = Bcodecv*fr
PLa = Bcodeca*fr;
Pv  = PLv/max_PLv;
fix_Pv = ceil(Pv);
Pa  = PLa/max_PLa;
fix_Pa = ceil(Pa);
Pv_size = PLv + (fix_Pv*(H_NALU + H_RTP + headerUDP*8 + headerIPv6*8));
Pa_size = PLa + (fix_Pa*(H_NALU + H_RTP + headerUDP*8 + headerIPv6*8));
Pvs_size = Pv_size + Pa_size;
Pvs_size_byte = Pvs_size/8;
Pvs_size_KB = Pvs_size_byte/1024;
Pvs_size_meby = Pvs_size_KB/1024;
% Calculate Bandwidth
```

```
Ba = Pa_size.*Bcodeca./PLa;
Bv = Pv_size.*Bcodecv./PLv
Bvs = Ba + Bv
Bvs_kbps = Bvs/1000
% Create bvideo plot
axes(handles.B_VS)
plot(Bcodecv,Bvs_kbps,'-*blue')
xlabel ('Bit Rate CODEC Video (bps)')
ylabel ('Bandwidth Video Streaming (kbps)')
set(handles.B_VS,'XMinorTick','on')
grid on
```

**Lampiran 2.** *Listing Program* Matlab Perhitungan *Delay End-toEnd* Aplikasi Video *Streaming*

```
function varargout = delay_tot(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @delay_tot_OpeningFcn, ...
                   'gui_OutputFcn',  @delay_tot_OutputFcn, ...
                   'gui_LayoutFcn',  [], ...
                   'gui_Callback',   []);
if nargin & isstr(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
   varargout{1:nargout} = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
function delay_tot_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
function varargout = delay_tot_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
function varargout = plot_button_Callback(h, eventdata, handles, varargin)
% Get user input from GUI
Bcodecv = eval(get(handles.Bcodecv,'String'));
% Calculate data
p = [0.1:0.1:0.9];
% Calculate packet data video streaming
Bcodeca = 48000;
fr = 0.033;
H_NALU = 8;
H_RTP = 96;
headerUDP = 8;
headerIPv6 = 40;
max_PLv = 2032;
max_PLa = 640;
PLv = Bcodecv*fr;
PLa = Bcodeca*fr;
Pv  = PLv/max_PLv;
fix_Pv = ceil(Pv);
Pa  = PLa/max_PLa;
fix_Pa = ceil(Pa);
Pv_size = PLv + (fix_Pv*(H_NALU + H_RTP + headerUDP*8 + headerIPv6*8));
Pa_size = PLa + (fix_Pa*(H_NALU + H_RTP + headerUDP*8 + headerIPv6*8));
Pvs_size = Pv_size + Pa_size;
Pvs_size_byte = Pvs_size/8;
```

```
% Calculate delay CODEC
ta = 0.04;
tv = 0.3;
tCODEC = ta + tv;
% Calculate delay in server/internet
MTU_eth = 1500;
if Pvs_size_byte <= MTU_eth
  Nframe_Eth = 1;
else
   Nframe_Eth = Pvs_size_byte/MTU_eth;
end;
header_eth = 14;
FCS = 4;
C_int = 1000000000;
fix_Nframe_Eth = ceil(Nframe_Eth);
Wframe_int = (fix_Nframe_Eth*(header_eth + FCS)+ Pvs_size_byte);
Wframe_int_bit =  Wframe_int*8;
tE1 = (Wframe_int - Pvs_size_byte)*8/C_int;
% Calculate delay in GGSN
Wvs_GGSN = Wframe_int - fix_Nframe_Eth*header_eth - fix_Nframe_Eth*FCS;
C_GGSN = 100000000;
tD1 = (Wframe_int-Wvs_GGSN)*8/C_GGSN;
header_GTP = 8;
MSS_GGSN = 1500 - header_GTP - headerUDP - headerIPv6;
if Wvs_GGSN <= MSS_GGSN
  Ndatagram_GGSN = 1;
else
   Ndatagram_GGSN = Wvs_GGSN./MSS_GGSN;
end;
fix_Ndatagram_GGSN = ceil(Ndatagram_GGSN);
Wdatagram_GGSN = Wvs_GGSN + fix_Ndatagram_GGSN.*(header_GTP +
headerUDP + headerIPv6);
if Wdatagram_GGSN <= MTU_eth
  Nframe_eth_GGSN = 1;
else
  Nframe_eth_GGSN = Wdatagram_GGSN/MTU_eth;
end;
fix_Nframe_eth_GGSN = ceil(Ndatagram_GGSN);
Wframe_GGSN = (fix_Nframe_eth_GGSN*(header_eth + FCS)) +
Wdatagram_GGSN;
C_GGSN = 100000000;
tE2 = (Wframe_GGSN - Wvs_GGSN)*8/C_GGSN;
tT2 = Wframe_GGSN*8/C_GGSN;
RGGSN_SGSN = 2000;
v = 300000000;
tP2 = fix_Nframe_eth_GGSN*RGGSN_SGSN/v;
L_GGSN = Wframe_GGSN*8;
u_GGSN = C_GGSN./L_GGSN;
lambda_GGSN =  u_GGSN'*p;
```

```
lambda_GGSN1 = lambda_GGSN(1,:);
lambda_GGSN2 = lambda_GGSN(2,:);
lambda_GGSN3 = lambda_GGSN(3,:);
x2_1 =  u_GGSN(1,1) - lambda_GGSN1;
x2_2 = u_GGSN(1,2) - lambda_GGSN2;
x2_3 = u_GGSN(1,3) - lambda_GGSN3;
y2_1 = lambda_GGSN1 ./ [u_GGSN(1,1) * [x2_1]];
y2_2 = lambda_GGSN2 ./ [u_GGSN(1,2) * x2_2];
y2_3 = lambda_GGSN3 ./ [u_GGSN(1,3) * x2_3];
z2_1 = 1 / u_GGSN(1,1);
z2_2 = 1 / u_GGSN(1,2);
z2_3 = 1 / u_GGSN(1,3);
tw2_1 = y2_1 + z2_1;
tw2_2 = y2_2 + z2_2;
tw2_3 = y2_3 + z2_3;
% Calculate delay in SGSN
Wvs_SGSN = Wframe_GGSN - fix_Ndatagram_GGSN*header_GTP -
fix_Ndatagram_GGSN*headerUDP - fix_Ndatagram_GGSN*headerIPv6 -
fix_Nframe_eth_GGSN*header_eth - fix_Nframe_eth_GGSN*FCS;
C_SGSN = 100000000;
tD2 = (Wframe_GGSN - Wvs_SGSN)*8/C_SGSN;
MSS_SGSN = MTU_eth - header_GTP - headerUDP - headerIPv6;
if Wvs_GGSN <= MSS_SGSN
   Ndatagram_SGSN = 1;
else
   Ndatagram_SGSN = Wvs_SGSN./MSS_SGSN;
end;
fix_Ndatagram_SGSN = ceil(Ndatagram_SGSN);
Wdatagram_SGSN = Wvs_SGSN + fix_Ndatagram_SGSN.*(header_GTP +
headerUDP + headerIPv6);
header_AAL5 = 8;
NframeCSAAL5 = Wdatagram_SGSN./65535;
fix_NframeCSAAL5 = ceil(NframeCSAAL5);
WframeCSAAL5 = Wdatagram_SGSN + fix_NframeCSAAL5.*header_AAL5;
PDU_SAR = 48;
Nframe_ATM = WframeCSAAL5/PDU_SAR;
fix_frame = ceil(Nframe_ATM);
header_ATM = 5;
Wframe_ATM = PDU_SAR + header_ATM;
Wframe_SGSN = fix_frame*Wframe_ATM;
C_SGSN_RNC = 155520000;
tE3 = (Wframe_SGSN - Wvs_SGSN)*8/C_SGSN_RNC;
tT3 = Wframe_SGSN*8/C_SGSN_RNC;
RSGSN_RNC = 3000;
v = 300000000;
tP3 = fix_frame*RSGSN_RNC/v;
L_SGSN = Wframe_SGSN * 8;
u_SGSN = C_SGSN./L_SGSN;
lambda_SGSN=  u_SGSN'*p;
```

```
lambda_SGSN1 = lambda_SGSN(1,:);
lambda_SGSN2 = lambda_SGSN(2,:);
lambda_SGSN3 = lambda_SGSN(3,:);
x3_1 =  u_SGSN(1,1) - lambda_SGSN1;
x3_2 = u_SGSN(1,2) - lambda_SGSN2;
x3_3 = u_SGSN(1,3) - lambda_SGSN3;
y3_1 = lambda_SGSN1 ./ [u_SGSN(1,1)*x3_1];
y3_2 = lambda_SGSN2 ./ [u_SGSN(1,2)*x3_2];
y3_3 = lambda_SGSN3 ./ [u_SGSN(1,3)*x3_3];
z3_1 = 1 /u_SGSN(1,1);
z3_2 = 1 /u_SGSN(1,2);
z3_3 = 1 /u_SGSN(1,3);
tw3_1 = y3_1 + z3_1;
tw3_2 = y3_2 + z3_2;
tw3_3 = y3_3 + z3_3;
% Calculate delay in RNC
header_ATM = 5;
Wvs_RNC = Wframe_SGSN - fix_Ndatagram_SGSN*header_GTP -
fix_Ndatagram_SGSN*headerUDP - fix_Ndatagram_SGSN*headerIPv6 -
fix_NframeCSAAL5*header_AAL5 - fix_frame*header_ATM;
C_RNC = 155520000;
tD3 = (Wframe_SGSN - Wvs_RNC)*8/C_RNC;
header_PDCP = 1;
N_datgramRNC = Wvs_RNC/1500;
fix_N_datgramRNC = ceil(N_datgramRNC);
Wframe_PDCP = Wvs_RNC + fix_N_datgramRNC*header_PDCP;
PDU_RLC = 40;
Nframe_RLC = Wframe_PDCP/PDU_RLC;
fix_frameRLC = ceil(Nframe_RLC);
header_RLC = 2;
Wframe_RLC = PDU_RLC + header_RLC;
Wframe_RLCtot = fix_frameRLC * Wframe_RLC;
MACd_SDU = 42;
Nframe_MACd = Wframe_RLCtot/MACd_SDU;
fix_Nframe_MACd = ceil(Nframe_MACd);
headerMACd = 0.5;
Wframe_MACd = MACd_SDU + headerMACd;
header_FP = 7;
CRC =  2;
Wframe_FP = Wframe_MACd + header_FP +CRC;
Wframe_FPtot = fix_Nframe_MACd*Wframe_FP;
CSPDU = 24;
H_CSPDU = 3;
N_CSPDU = Wframe_FPtot/CSPDU;
fix_NCSPDU = ceil(N_CSPDU);
Wframe_CSPDUtot = Wframe_FPtot + fix_NCSPDU*H_CSPDU;
SARPDU = 47;
H_SARPDU = 1;
N_SARPDU = Wframe_CSPDUtot/SARPDU;
```

```
fix_N_SARPDU = ceil(N_SARPDU);
Wframe_SARPDU = SARPDU + H_SARPDU;
Wframe_SARPDUtot = fix_N_SARPDU*Wframe_SARPDU;
Nframe_ATM_RNC = Wframe_SARPDUtot/PDU_SAR;
fix_frameATM_RNC = ceil(Nframe_ATM_RNC);
Wframe_RNC = fix_frameATM_RNC*Wframe_ATM;
tE4 = (Wframe_RNC - Wvs_RNC)*8/C_RNC;
tT4 = Wframe_RNC*8/C_RNC;
RRNC_NodeB = 400;
v = 300000000;
tP4 = fix_frameATM_RNC*RRNC_NodeB/v;
L_RNC = Wframe_RNC * 8;
u_RNC = C_RNC./L_RNC;
lambda_RNC =  u_RNC'*p;
lambda_RNC1 = lambda_RNC(1,:);
lambda_RNC2 = lambda_RNC(2,:);
lambda_RNC3 = lambda_RNC(3,:);
x4_1 =  u_RNC(1,1) - lambda_RNC1;
x4_2 = u_RNC(1,2) - lambda_RNC2;
x4_3 = u_RNC(1,3) - lambda_RNC3;
y4_1 = lambda_RNC1./[u_RNC(1,1)*x4_1];
y4_2 = lambda_RNC2./[u_RNC(1,2)*x4_2];
y4_3 = lambda_RNC3./[u_RNC(1,3)*x4_3];
z4_1 = 1./u_RNC(1,1);
z4_2 = 1./u_RNC(1,2);
z4_3 = 1./u_RNC(1,3);
tw4_1 = y4_1 + z4_1;
tw4_2 = y4_2 + z4_2;
tw4_3 = y4_3 + z4_3;
% Calculate delay in Node B
Wvs_NodeB = Wframe_RNC - fix_frameATM_RNC*header_ATM -
fix_NCSPDU*H_CSPDU - fix_N_SARPDU*H_SARPDU -
Nframe_MACd*header_FP - Nframe_MACd*CRC;
C_NodeB = 155520000;
tD4 = (Wframe_RNC - Wvs_NodeB)*8/C_NodeB;
TBS = 7298;
N_TBS = Wvs_NodeB*8/TBS;
fix_N_TBS = ceil(N_TBS);
header_MAChs = 21;
Wframe_NodeB = Wvs_NodeB + fix_N_TBS*CRC;
C_UE = 3600000;
tE5 = (Wframe_NodeB - Wvs_NodeB)*8/C_NodeB;
tT5 = Wframe_NodeB*8/(3*C_UE);
RNodeB_UE = 1500;
v = 300000000;
tP5 = fix_N_TBS*RNodeB_UE/v;
L_NodeB = Wframe_NodeB.* 8;
u_NodeB= C_NodeB ./ L_NodeB;
lambda_NodeB =  u_NodeB'*p;
```

```
lambda_NodeB1 = lambda_NodeB(1,:);
lambda_NodeB2 = lambda_NodeB(2,:);
lambda_NodeB3 = lambda_NodeB(3,:);
x5_1 =  u_NodeB(1,1) - lambda_NodeB1;
x5_2 = u_NodeB(1,2) - lambda_NodeB2;
x5_3 = u_NodeB(1,3) - lambda_NodeB3;
y5_1 = lambda_NodeB1./[u_NodeB(1,1)*x5_1];
y5_2 = lambda_NodeB2./[u_NodeB(1,2)*x5_2];
y5_3 = lambda_NodeB3./[u_NodeB(1,3)*x5_3];
z5_1 = 1./u_NodeB(1,1);
z5_2 = 1./u_NodeB(1,2);
z5_3 = 1./u_NodeB(1,3);
tw5_1 = y5_1 + z5_1;
tw5_2 = y5_2 + z5_2;
tw5_3 = y5_3 + z5_3;
% Calculate delay in UE
Wvs_UE = Wframe_NodeB - (fix_Pv + fix_Pa)*H_RTP/8 - (fix_Pv +
fix_Pa)*headerIPv6 - (fix_Pv + fix_Pa)*headerUDP -
fix_N_datgramRNC*header_PDCP - fix_frameRLC*header_RLC -
fix_Nframe_MACd*headerMACd - fix_N_TBS*header_MAChs/8 - fix_N_TBS*CRC;
C_UE = 3600000;
tD5= (Wframe_NodeB- Wvs_UE)*8/C_UE;
% Calculate delay of HSDPA Network
tEtot = tE1 + tE2 + tE3 + tE4 + tE5;
tDtot = tD1 + tD2 + tD3 + tD4 + tD5;
tproc = tEtot + tDtot;
tTtot = tT2 + tT3 + tT4 + tT5;
tPtot = tP2 + tP3 + tP4 + tP5;
twtot_1 = tw2_1 + tw3_1 + tw4_1 + tw5_1
twtot_2 = tw2_2 + tw3_2 + tw4_2 + tw5_2
twtot_3 = tw2_3 + tw3_3 + tw4_3 + tw5_3
tnet_1 = tproc(1,1) + tTtot(1,1) + tPtot(1,1) + twtot_1;
tnet_2 = tproc(1,2) + tTtot(1,2) + tPtot(1,2) + twtot_2;
tnet_3 = tproc(1,3) + tTtot(1,3) + tPtot(1,3) + twtot_3
% Calculate end to end delay
ttot1 = tCODEC + tnet_1
ttot2 = tCODEC + tnet_2
ttot3 = tCODEC + tnet_3
% Create end to end delay plot
axes(handles.delay_tot)
set(handles.delay_tot,'XMinorTick','on')
plot(p,ttot1,'-^r',p,ttot2,'-*b', p,ttot3,'-*g')
grid on
xlabel ('Faktor Utilitas')
ylabel ('Delay End-to-End (s)')
legend('Bit Rate CODEC video 256 kbps','Bit Rate CODEC video 320 kbps','Bit Rate
CODEC video 384 kbps')
```

**Lampiran 3.** *Listing Program* Matlab Menghitung Probabilitas *Packet Loss* Aplikasi Video *Streaming*

```
function varargout = packet_loss(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',         mfilename, ...
                   'gui_Singleton',    gui_Singleton, ...
                   'gui_OpeningFcn',   @packet_loss_OpeningFcn, ...
                   'gui_OutputFcn',    @packet_loss_OutputFcn, ...
                   'gui_LayoutFcn',    [], ...
                   'gui_Callback',     []);
if nargin & isstr(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
   varargout{1:nargout} = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
function packet_loss_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
 function varargout = packet_loss_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
function varargout = plot_button_Callback(h, eventdata, handles, varargin)
% Get user input from GUI
Bcodecv = eval(get(handles.Bcodecv,'String'));
pb = 0.00000001;
% Calculate packet loss of video streaming
Bcodeca = 48000;
fr = 0.033;
H_NALU = 8;
H_RTP = 96;
headerUDP = 8;
headerIPv6 = 40;
max_PLv = 2032;
max_PLa = 640;
PLv = Bcodecv*fr;
PLa = Bcodeca*fr;
Pv  = PLv/max_PLv;
fix_Pv = ceil(Pv);
Pa  = PLa/max_PLa;
fix_Pa = ceil(Pa);
Pv_size = PLv + (fix_Pv*(H_NALU + H_RTP + headerUDP*8 + headerIPv6*8));
Pa_size = PLa + (fix_Pa*(H_NALU + H_RTP + headerUDP*8 + headerIPv6*8));
Pvs_size = Pv_size + Pa_size
p_VS = Pvs_size* pb
% Calculate packet loss on HSDPA Network
```

```
PG = 12;
NF = 8;
Pt = 43;
Lt = 2;
Lr = 0;
Gt = 18;
Gr = 2;
K = 1.381 * 10^-23;
T = 300;
B = 5*10^6;
R = 3.6*10^6;
c = 3*10^8;
f = 2.14*10^9;
d = 1500;
N = 10*log10(K*T) + 10*log10(B) + NF;
lmbda = c/f;
FSL = 20*log10((4*pi*d)/lmbda);
Pr = Pt - FSL - Lt - Lr + Gt + Gr;
SNR = Pr - N;
Eb_per_No_dB = SNR - 10*log10(B/R);
M = 16;
x = sqrt((3*Eb_per_No_dB*log2(M))/(M-1));
BER = ((4*(sqrt(M)-1))/(sqrt(M)*log2(M)))*0.5*erfc(x/sqrt(2))
% Calculate total packet loss
p_tot1 = 1 - ((1 - BER) *(1 - p_VS(1,1)));
p_tot2 = 1 - ((1 - BER) *(1 - p_VS(1,2)));
p_tot3 = 1 - ((1 - BER) *(1 - p_VS(1,3)));
p_tot = [p_tot1 p_tot2 p_tot3]
% Create packet loss plot
axes(handles.packet_loss)
set(handles.packet_loss,'XMinorTick','on')
plot(Bcodecv,p_tot)
grid on
xlabel ('Bit Rate CODEC Video (bps)')
ylabel ('Probabilitas Packet Loss')
```

**Lampiran 4.** *Listing Program* Matlab Menghitung Throughput Aplikasi Video *Streaming*

```
function varargout = throughput(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',         mfilename, ...
                   'gui_Singleton',    gui_Singleton, ...
                   'gui_OpeningFcn',   @throughput_OpeningFcn, ...
                   'gui_OutputFcn',    @throughput_OutputFcn, ...
                   'gui_LayoutFcn',    [], ...
                   'gui_Callback',     []);
if nargin & isstr(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
   varargout{1:nargout} = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
function throughput_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
function varargout = throughput_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
function varargout = plot_button_Callback(h, eventdata, handles, varargin)
% Get user input from GUI
Bcodecv = eval(get(handles.Bcodecv,'String'));
% Calculate data
p = [0.1:0.1:0.9];
% Calculate bandwidth of video streaming
Bcodeca = 48000;
fr = 0.033;
H_NALU = 8;
H_RTP = 96;
headerUDP = 8;
headerIPv6 = 40;
max_PLv = 2032;
max_PLa = 640;
PLv = Bcodecv*fr;
PLa = Bcodeca*fr;
Pv  = PLv/max_PLv;
fix_Pv = ceil(Pv);
Pa  = PLa/max_PLa;
fix_Pa = ceil(Pa);
Pv_size = PLv + (fix_Pv*(H_NALU + H_RTP + headerUDP*8 + headerIPv6*8));
Pa_size = PLa + (fix_Pa*(H_NALU + H_RTP + headerUDP*8 + headerIPv6*8));
Pvs_size = Pv_size + Pa_size;
Pvs_size_byte = Pvs_size/8;
```

```
% Calculate data in server/internet
MTU_eth = 1500;
if Pvs_size_byte <= MTU_eth
  Nframe_Eth = 1;
else
   Nframe_Eth = Pvs_size_byte/MTU_eth;
end;
header_eth = 14;
FCS = 4;
C_int = 1000000000;
fix_Nframe_Eth = ceil(Nframe_Eth);
Wframe_int = (fix_Nframe_Eth*(header_eth + FCS)+ Pvs_size_byte);
Wframe_int_bit =  Wframe_int*8;
tE1 = (Wframe_int - Pvs_size_byte)*8/C_int;
% Calculate delay in GGSN
Wvs_GGSN = Wframe_int - fix_Nframe_Eth*header_eth - fix_Nframe_Eth*FCS;
C_GGSN = 100000000;
tD1 = (Wframe_int-Wvs_GGSN)*8/C_GGSN;
header_GTP = 8;
MSS_GGSN = 1500 - header_GTP - headerUDP - headerIPv6;
if Wvs_GGSN <= MSS_GGSN
  Ndatagram_GGSN = 1;
else
   Ndatagram_GGSN = Wvs_GGSN./MSS_GGSN;
end;
fix_Ndatagram_GGSN = ceil(Ndatagram_GGSN);
Wdatagram_GGSN = Wvs_GGSN + fix_Ndatagram_GGSN.*(header_GTP +
headerUDP + headerIPv6);
if Wdatagram_GGSN <= MTU_eth
  Nframe_eth_GGSN = 1;
else
  Nframe_eth_GGSN = Wdatagram_GGSN/MTU_eth;
end;
fix_Nframe_eth_GGSN = ceil(Ndatagram_GGSN);
Wframe_GGSN = (fix_Nframe_eth_GGSN*(header_eth + FCS)) +
Wdatagram_GGSN;
C_GGSN = 100000000;
tE2 = (Wframe_GGSN - Wvs_GGSN)*8/C_GGSN;
RGGSN_SGSN = 2000;
v = 300000000;
tP2 = fix_Nframe_eth_GGSN*RGGSN_SGSN/v;
L_GGSN = Wframe_GGSN*8;
u_GGSN = C_GGSN./L_GGSN;
lambda_GGSN =  u_GGSN'*p;
lambda_GGSN1 = lambda_GGSN(1,:);
lambda_GGSN2 = lambda_GGSN(2,:);
lambda_GGSN3 = lambda_GGSN(3,:);
x2_1 =  u_GGSN(1,1) - lambda_GGSN1;
x2_2 = u_GGSN(1,2) - lambda_GGSN2;
```

```
x2_3 = u_GGSN(1,3) - lambda_GGSN3;
y2_1 = lambda_GGSN1 ./ [u_GGSN(1,1) * [x2_1]];
y2_2 = lambda_GGSN2 ./ [u_GGSN(1,2) * x2_2];
y2_3 = lambda_GGSN3 ./ [u_GGSN(1,3) * x2_3];
z2_1 = 1 / u_GGSN(1,1);
z2_2 = 1 / u_GGSN(1,2);
z2_3 = 1 / u_GGSN(1,3);
tw2_1 = y2_1 + z2_1;
tw2_2 = y2_2 + z2_2;
tw2_3 = y2_3 + z2_3;
% Calculate delay in SGSN
Wvs_SGSN = Wframe_GGSN - fix_Ndatagram_GGSN*header_GTP -
fix_Ndatagram_GGSN*headerUDP - fix_Ndatagram_GGSN*headerIPv6 -
fix_Nframe_eth_GGSN*header_eth - fix_Nframe_eth_GGSN*FCS;
C_SGSN = 100000000;
tD2 = (Wframe_GGSN - Wvs_SGSN)*8/C_SGSN;
MSS_SGSN = MTU_eth - header_GTP - headerUDP - headerIPv6;
if Wvs_GGSN <= MSS_SGSN
  Ndatagram_SGSN = 1;
else
  Ndatagram_SGSN = Wvs_SGSN./MSS_SGSN;
end;
fix_Ndatagram_SGSN = ceil(Ndatagram_SGSN);
Wdatagram_SGSN = Wvs_SGSN + fix_Ndatagram_SGSN.*(header_GTP +
headerUDP + headerIPv6);
header_AAL5 = 8;
NframeCSAAL5 = Wdatagram_SGSN./65535;
fix_NframeCSAAL5 = ceil(NframeCSAAL5);
WframeCSAAL5 = Wdatagram_SGSN + fix_NframeCSAAL5.*header_AAL5;
PDU_SAR = 48;
Nframe_ATM = WframeCSAAL5/PDU_SAR;
fix_frame = ceil(Nframe_ATM);
header_ATM = 5;
Wframe_ATM = PDU_SAR + header_ATM;
Wframe_SGSN = fix_frame*Wframe_ATM;
C_SGSN_RNC = 155520000;
tE3 = (Wframe_SGSN - Wvs_SGSN)*8/C_SGSN_RNC;
RSGSN_RNC = 3000;
v = 300000000;
tP3 = fix_frame*RSGSN_RNC/v;
L_SGSN = Wframe_SGSN * 8;
u_SGSN = C_SGSN./L_SGSN;
lambda_SGSN=  u_SGSN'*p;
lambda_SGSN1 = lambda_SGSN(1,:);
lambda_SGSN2 = lambda_SGSN(2,:);
lambda_SGSN3 = lambda_SGSN(3,:);
x3_1 =  u_SGSN(1,1) - lambda_SGSN1;
x3_2 = u_SGSN(1,2) - lambda_SGSN2;
x3_3 = u_SGSN(1,3) - lambda_SGSN3;
```

```
y3_1 = lambda_SGSN1 ./ [u_SGSN(1,1)*x3_1];
y3_2 = lambda_SGSN2 ./ [u_SGSN(1,2)*x3_2];
y3_3 = lambda_SGSN3 ./ [u_SGSN(1,3)*x3_3];
z3_1 = 1 /u_SGSN(1,1);
z3_2 = 1 /u_SGSN(1,2);
z3_3 = 1 /u_SGSN(1,3);
tw3_1 = y3_1 + z3_1;
tw3_2 = y3_2 + z3_2;
tw3_3 = y3_3 + z3_3;
% Calculate delay in RNC
header_ATM = 5;
Wvs_RNC = Wframe_SGSN - fix_Ndatagram_SGSN*header_GTP -
fix_Ndatagram_SGSN*headerUDP - fix_Ndatagram_SGSN*headerIPv6 -
fix_NframeCSAAL5*header_AAL5 - fix_frame*header_ATM;
C_RNC = 155520000;
tD3 = (Wframe_SGSN - Wvs_RNC)*8/C_RNC;
header_PDCP = 1;
N_datgramRNC = Wvs_RNC/1500;
fix_N_datgramRNC = ceil(N_datgramRNC);
Wframe_PDCP = Wvs_RNC + fix_N_datgramRNC*header_PDCP;
PDU_RLC = 40;
Nframe_RLC = Wframe_PDCP/PDU_RLC;
fix_frameRLC = ceil(Nframe_RLC);
header_RLC = 2;
Wframe_RLC = PDU_RLC + header_RLC;
Wframe_RLCtot = fix_frameRLC * Wframe_RLC;
MACd_SDU = 42;
Nframe_MACd = Wframe_RLCtot/MACd_SDU;
fix_Nframe_MACd = ceil(Nframe_MACd);
headerMACd = 0.5;
Wframe_MACd = MACd_SDU + headerMACd;
header_FP = 7;
CRC =  2;
Wframe_FP = Wframe_MACd + header_FP +CRC;
Wframe_FPtot = fix_Nframe_MACd*Wframe_FP;
CSPDU = 24;
H_CSPDU = 3;
N_CSPDU = Wframe_FPtot/CSPDU;
fix_NCSPDU = ceil(N_CSPDU);
Wframe_CSPDUtot = Wframe_FPtot + fix_NCSPDU*H_CSPDU;
SARPDU = 47;
H_SARPDU = 1;
N_SARPDU = Wframe_CSPDUtot/SARPDU;
fix_N_SARPDU = ceil(N_SARPDU);
Wframe_SARPDU = SARPDU + H_SARPDU;
Wframe_SARPDUtot = fix_N_SARPDU*Wframe_SARPDU;
Nframe_ATM_RNC = Wframe_SARPDUtot/PDU_SAR;
fix_frameATM_RNC = ceil(Nframe_ATM_RNC);
Wframe_RNC = fix_frameATM_RNC*Wframe_ATM;
```

```
tE4 = (Wframe_RNC - Wvs_RNC)*8/C_RNC;
RRNC_NodeB = 400;
v = 300000000;
tP4 = fix_frameATM_RNC*RRNC_NodeB/v;
L_RNC = Wframe_RNC * 8;
u_RNC = C_RNC./L_RNC;
lambda_RNC =  u_RNC'*p;
lambda_RNC1 = lambda_RNC(1,:);
lambda_RNC2 = lambda_RNC(2,:);
lambda_RNC3 = lambda_RNC(3,:);
x4_1 =  u_RNC(1,1) - lambda_RNC1;
x4_2 = u_RNC(1,2) - lambda_RNC2;
x4_3 = u_RNC(1,3) - lambda_RNC3;
y4_1 = lambda_RNC1./[u_RNC(1,1)*x4_1];
y4_2 = lambda_RNC2./[u_RNC(1,2)*x4_2];
y4_3 = lambda_RNC3./[u_RNC(1,3)*x4_3];
z4_1 = 1./u_RNC(1,1);
z4_2 = 1./u_RNC(1,2);
z4_3 = 1./u_RNC(1,3);
tw4_1 = y4_1 + z4_1;
tw4_2 = y4_2 + z4_2;
tw4_3 = y4_3 + z4_3;
% Calculate delay in Node B
Wvs_NodeB = Wframe_RNC - fix_frameATM_RNC*header_ATM -
fix_NCSPDU*H_CSPDU - fix_N_SARPDU*H_SARPDU -
Nframe_MACd*header_FP - Nframe_MACd*CRC;
C_NodeB = 155520000;
tD4 = (Wframe_RNC - Wvs_NodeB)*8/C_NodeB;
TBS = 7298;
N_TBS = Wvs_NodeB*8/TBS;
fix_N_TBS = ceil(N_TBS);
header_MAChs = 21;
Wframe_NodeB = Wvs_NodeB + fix_N_TBS*CRC;
C_UE = 3600000;
tE5 = (Wframe_NodeB - Wvs_NodeB)*8/C_NodeB;
RNodeB_UE = 1500;
v = 300000000;
tP5 = fix_N_TBS*RNodeB_UE/v;
L_NodeB = Wframe_NodeB.* 8;
u_NodeB= C_NodeB ./ L_NodeB;
lambda_NodeB =  u_NodeB'*p;
lambda_NodeB1 = lambda_NodeB(1,:);
lambda_NodeB2 = lambda_NodeB(2,:);
lambda_NodeB3 = lambda_NodeB(3,:);
x5_1 =  u_NodeB(1,1) - lambda_NodeB1;
x5_2 = u_NodeB(1,2) - lambda_NodeB2;
x5_3 = u_NodeB(1,3) - lambda_NodeB3;
y5_1 = lambda_NodeB1./[u_NodeB(1,1)*x5_1];
y5_2 = lambda_NodeB2./[u_NodeB(1,2)*x5_2];
```

```
y5_3 = lambda_NodeB3./[u_NodeB(1,3)*x5_3];
z5_1 = 1./u_NodeB(1,1);
z5_2 = 1./u_NodeB(1,2);
z5_3 = 1./u_NodeB(1,3);
tw5_1 = y5_1 + z5_1;
tw5_2 = y5_2 + z5_2;
tw5_3 = y5_3 + z5_3;
% Calculate delay in UE
Wvs_UE = Wframe_NodeB - (fix_Pv + fix_Pa)*H_RTP/8 - (fix_Pv +
fix_Pa)*headerIPv6 - (fix_Pv + fix_Pa)*headerUDP -
fix_N_datgramRNC*header_PDCP - fix_frameRLC*header_RLC -
fix_Nframe_MACd*headerMACd - fix_N_TBS*header_MAChs/8 - fix_N_TBS*CRC;
C_UE = 3600000;
tD5= (Wframe_NodeB- Wvs_UE)*8/C_UE;
% Calculate delay
tEtot = tE1 + tE2 + tE3 + tE4 + tE5;
tDtot = tD1 + tD2 + tD3 + tD4 + tD5;
tproc = tEtot + tDtot;
tPtot = tP2 + tP3 + tP4 + tP5;
twtot_1 = tw2_1 + tw3_1 + tw4_1 + tw5_1;
twtot_2 = tw2_2 + tw3_2 + tw4_2 + tw5_2;
twtot_3 = tw2_3 + tw3_3 + tw4_3 + tw5_3;
tpro1 = (tproc(1,1)/1) + (twtot_1./1);
tpro2 = (tproc(1,2)/1) + (twtot_2./1);
tpro3 = (tproc(1,3)/1) + (twtot_3./1);
tl = (Wframe_NodeB*8)/C_UE;
tp = tPtot./1;
tout1 = (2*tp(1,1)) + (2*tl(1,1)) + tpro1;
tout2 = (2*tp(1,2)) + (2*tl(1,2)) + tpro2;
tout3 = (2*tp(1,3)) + (2*tl(1,3)) + tpro3;
% Calculate konstanta
a1 = 1 + tout1/tl(1,1);
a2 = 1 + tout2/tl(1,2);
a3 = 1 + tout3/tl(1,3);
% Calculate packet loss total
pb = 0.00000001;
p_VS = Pvs_size * pb;
PG = 12;
NF = 8;
Pt = 43;
Lt = 2;
Lr = 0;
Gt = 18;
Gr = 2;
K = 1.381 * 10^-23;
T = 300;
B = 5*10^6;
R = 3.6*10^6;
c = 3*10^8;
```

```
f = 2.14*10^9;
d = 1500;
N = 10*log10(K*T) + 10*log10(B) + NF;
lmbda = c/f;
FSL = 20*log10((4*pi*d)/lmbda);
Pr = Pt - FSL - Lt - Lr + Gt + Gr;
SNR = Pr - N;
Eb_per_No_dB = SNR - 10*log10(B/R);
M = 16;
x = sqrt((3*Eb_per_No_dB*log2(M))/(M-1));
BER = ((4*(sqrt(M)-1))/(sqrt(M)*log2(M)))*0.5*erfc(x/sqrt(2));
p_tot1 = 1 - ((1 - BER) *(1 - p_VS(1,1)));
p_tot2 = 1 - ((1 - BER) *(1 - p_VS(1,2)));
p_tot3 = 1 - ((1 - BER) *(1 - p_VS(1,3)));
p_tot = [p_tot1 p_tot2 p_tot3];
% Calculate T1
T1 = ( 1 - p_tot1) ./ (tl(1,1) * (1 + (a1 -1).*p_tot1));
T2 = ( 1 - p_tot2) ./ (tl(1,2) * (1 + (a2 -1).*p_tot2));
T3 = ( 1 - p_tot3) ./ (tl(1,3) * (1 + (a3 -1).*p_tot3))
T1_kbps = T1 * Wframe_NodeB(1,1) * 8 / 1000
T2_kbps = T2 * Wframe_NodeB(1,2) * 8 / 1000
T3_kbps = T3 * Wframe_NodeB(1,3) * 8 / 1000
T_kbps = [T1_kbps T2_kbps T3_kbps];
% Create T1 bit rate CODEC video 128 kbps plot
axes(handles.T1)
set(handles.T1,'XMinorTick','on')
plot(p,T1_kbps,'-sr')
grid on
legend('Bit Rate CODEC video 256 kbps')
xlabel ('Faktor Utilitas')
ylabel ('Throughput (kbps)')
% Create T1 bit rate CODEC video 192 kbps plot
axes(handles.T2)
set(handles.T2,'XMinorTick','on')
plot(p,T2_kbps,'-sb')
grid on
legend('Bit Rate CODEC video 320 kbps')
xlabel ('Faktor Utilitas')
ylabel ('Throughput (kbps)')
% Create T1 bit rate CODEC video 256 kbps plot
axes(handles.T3)
set(handles.T3,'XMinorTick','on')
plot(p,T3_kbps,'-sg')
grid on
legend('Bit Rate CODEC video 384 kbps')
xlabel ('Faktor Utilitas')
ylabel ('Throughput (kbpss)')
```