BAB II
DASAR TEORI

2.1 Sejarah Perkembangan *Mobile Telephony*

Permulaan dari sistem *commercial mobile communication* bermula pada akhir tahun 1970. Sistem tersebut dibagi menjadi tiga generasi. Saat ini sistem generasi ketiga (3G) telah tersebar di beberapa negara dan generasi keempat (4G) tengah dikembangkan. Perbedaan dengan generasi awal ada kaitannya dengan teknologi dasar yang digunakan dan pelayanan yang disediakan atau tindak lanjut dari desain awal tersebut.

2.1.1 Sistem Awal

Contoh pertama dari sistem mobile phone telah diperkenalkan pada tahun 1940 dan mereka bisa ditampilkan sebagai evolusi dari *walkie talkie*. Kebanyakan dari sistem awal bukanlah sistem selular. Disamping mereka hanya mempunyai satu *Base Station (BS)*, biasanya terletak pada sebuah kota besar, dan telepon bekerja dengan jangkauan pada stasiun dasar tersebut. *Transmit power* yang disalurkan sangat besar dibandingkan dengan sistem *mobile phone* saat ini

2.1.2 Generasi Pertama (1G)

Konsep revolusioner dari sistem selular telah disebutkan pada teori tahun 1974 oleh *Bell Labs*, tetapi tidak berdampak pada realita hingga pertengahan tahun 1970 saat percobaan sistem selular dimulai, yang memacu sistem *mobile phone* generasi pertama sekitar tahun 1980. Kebanyakan generasi pertama dari sistem *mobile phone* adalah murni sistem analog, yang berpindah secara langsung dari sistem *original wire-based telephone* ke *mobile systems*. Berikut adalah beberapa contoh dari sistem tersebut :

- *Nippon Telephone and Telegraph Corporation (NTT)*

  Diperkenalkan di Tokyo, Jepang pada tahun 1979. Sistem ini sangat mahal dan meraih popularitas yang sangat kecil.

- *Total Access Communication System (TACS)*

  Sistem ini didasari pada akses *Frequency Division Multiple (FDMA)* dan telah diperkenalkan di Inggris pada tahun 1980. Di Jepang sistem *JTACS (J for Japanese)* and *NTACS (N for Narrowband)* sesaat menjadi pesaing tangguh dari sistem NTT setelah diperkenalkan pada tahun 1991.

- *Advanced Mobile Phone System (AMPS)*

  Ini adalah sistem Amerika yang mengacu pada modulasi frekuensi dan FDMA. Pada tahun 1979 percobaan pertama telah dilakukan menggunakan sistem ini. Secara komersial diperkenalkan tahun 1983. Ada versi *narrowband* darii AMPS yang disebut NAMPS.

2.1.3 Generasi Kedua (2G)

Sebagai lawan dari generasi pertama, generasi kedua sistem *mobile phone* menerapkan sistem digital. Tujuan dari 2G adalah untuk menyediakan komunikasi *voice* yang *reliable*. Kata sandi digunakan untuk mengemas contoh data dan control error coding sebaik seperti teknik digital modulasi digunakan untuk meningkatkan kualitas komunikasi. Berikut adalah contoh-contohnya:

- *United States Digital Cellular (USDC)*

  Sistem *mobile communication* dasar pada *Time Division Multiple Access (TDMA)* diperkenalkan pada tahun 1991 di Amerika. USDA dikenal dengan beberapa nama seperti *NADS (North America Digital Cellular)*, TDMA berdasarkan metode aksesnya, dan IS-54 atau IS-136 (nomor standar).

- *Global System for Mobile telecommunication (GSM)*

  Sistem yang menggunakan *frequency hoping* yang dikombinasikan dengan TDMA. Diperkenalkan pada tahun 1990 dan kemudian digunakan di seluruh penjuru Eropa. Sistem pertama menggunakan *frequency band* sekitar 900 MHz dan di beberapa lokasi sekitar 850 MHz. Saat ini sistem ini digunakan di seluruh dunia dengan *frequency band* sekitar 1800 MHz dan 1900 MHz.

- *IS-95 Code Division Multiple Access (CDMA)*

  Ada beberapa sistem yang dikenal dengan CDMA. Seluruhnya menggunakan metode akses *Direct Sequence Code Division Multiple Access (DS-CDMA)*. IS-95 adalah *narrowband CDMA* yang utama digunakan diseluruh Amerika Utara dan Korea. *Frekuensi band* dari IS-95 CDMA adalah sekitar 1800 MHz dan 1900 MHz.

2.1.4 Generasi Ketiga (3G)

Sistem generasi ketiga dijuluki dengan nama *Universal Mobile Telecommunication System (UMTS)* atau sederhananya dijuluki 3G. Sebagaimana sistem generasi kedua, generasi ketiga merupakan sistem digital, tetapi didesain untuk layanan digital umum. Komunikasi *voice* hanyalah salah satu dari layanan tersebut. Berikut adalah dua varian dari teknologi UMTS yang menggunakan metode akses yang berbeda:

- Mode FDD: *Wideband CDMA (WCDMA)*

  WCDMA adalah versi Eropa, yang merupakan DS-CDMA dengan bandwidth yang besar. Sistem ini diperkenalkan di berbagai negara di Eropa pada tahun 2003.
- Mode TDD: *Time Division Synchronous CDMA (TD-SCDMA)*

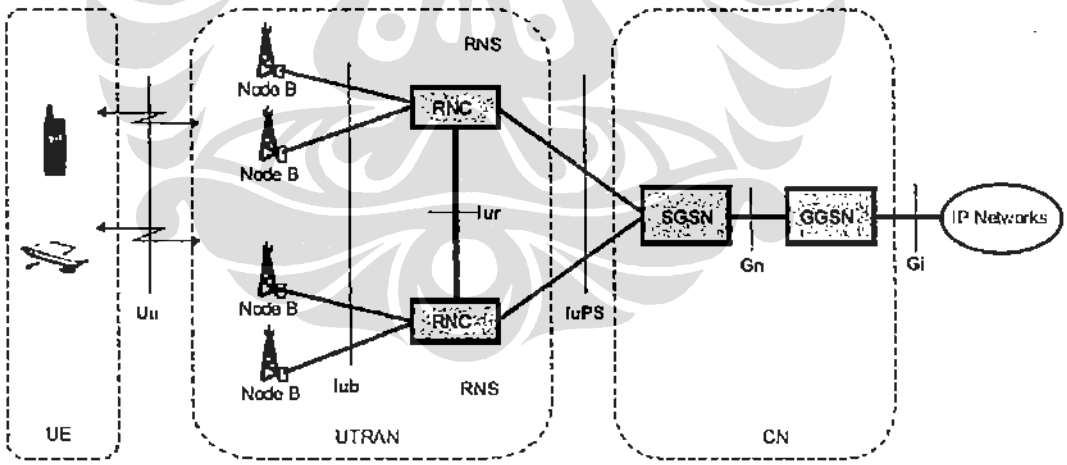
  TD-SCDMA merupakan Universal Mobile Telecommunication System 7 versi Chinese yang berbasis *Time Division Duplex (TDD)* dan DS-CDMA. TDD berarti komunikasi dua arah yang diperoleh dari pembagian waktu.

2.2 UMTS

Universal Mobile Telecommunication System (UMTS) merupakan salah satu sistem generasi ketiga yang dikembangkan di Eropa. Standarisasi dari UMTS ini dilakukan oleh *European Telecommunication Standard Institution (ETSI)*. Pada waktu yang sama, *International Telecommunications Union Telecommunication Standardisation Sector (ITU-T)* mengerjakan sistem yang sama yang bernama *International Mobile Telecommunication System 2000 (IMT 2000)*. Kedua badan standarisasi ini melakukan kerjasama sehingga terbentuk satu sistem untuk masa yang akan datang.

UMTS didesain sedemikian rupa sehingga mampu menyediakan bandwidth sebesar 2 Mbits/s. Layanan yang dapat diberikan UMTS diupayakan dapat memenuhi permintaan user dimanapun berada, artinya UMTS diharapkan dapat melayani area seluas mungkin, jika tidak ada *cell* UMTS pada suatu daerah dapat di-*route*-kan melalui satelit. UMTS dapat digunakan oleh perkantoran, rumah dan kendaraan. Layanan yang sama dapat diberikan untuk *user environment indoor* dan *outdoor*, *public and private area*, *urban* dan *rural*.

Frekuensi radio yang dialokasikan untuk UMTS adalah 1885-2025 MHz dan 2110-2200 MHz. *Band* tersebut akan digunakan oleh *cell* yang kecil (*pico-cell*) sehingga dapat memberikan kapasitas yang besar pada UMTS. *Multiple access* yang digunakan dapat mengalokasikan bandwidth secara dinamis sesuai dengan kebutuhan pelanggan. *Research and Technology Development in Advanced Communications Technologies in Europe (RACE)* telah mengembangkan dua jenis multiple akses yakni *Code Division Multiple Acces (CDMA)* dan *Time Division Multiple Acces (TDMA)*. Diantara keduanya belum diputuskan yang akan digunakan.



Gambar 2.1 Topologi jaringan UMTS [2]

Gambar 2.1 mendeskripsikan topologi pada jaringan UMTS. Jaringan UMTS terdiri dari beberapa komponen utama, yaitu :

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

  Merupakan UMTS *mobile terminal*, mempunyai kemampuan untuk berhubungan dengan jaringan dan dapat menggunakan layanan UMTS

- ***UMTS Access Network (UTRAN)***

  Bertanggung jawab terhadap fungsi-fungsi yang berhubungan dengan radio seperti *handover* dan manajemen hubungan.

- ***Core Network (CN)***

  Berfungsi untuk *switching* dan transportasi data, dimana fungsi-fungsi yang berhubungan dengan pergerakan terminal diimplementasikan pada *Intelligent Network (IN)*.

UMTS mendefiniskan empat macam kelas *Quality of Service (QoS)*. Tabel 2.1 [3] menunjukkan parameter-parameter yang didefinisikan untuk masing-masing kelas dari yang tertinggi sampai yang terendah.

Tabel 2.1: Parameter-parameter QoS pada UMTS

	Conversational Class	Streaming Class	Interactive Class	Background Class
Delay	tetap dan kecil	kecil dan variabel	variabel dan lebih moderate	besar dan variabel
Buffering	tidak ada	diperbolehkan	diperbolehkan	diperbolehkan
Jenis trafik	simetrik	asimetrik	asimetrik	asimetrik
Bit rate	dijamin	Dijamin	tidak dijamin	tidak dijamin
Contoh	*voice (real-time* dan interaktif)	music, *video streaming*	web browsing	email

Radio Link Conrol (RLC) pada UMTS dapat dikonfigurasi oleh *Radio Network Controller (RNC)* ke dalam 3 mode operasi yaitu [2]:

1. ***Transparent Mode (TM)***

   Tidak ada overhead protokol yang ditambahkan ke layer yang lebih tinggi. *Protocol Data Unit (PDU)* yag eror dapat dibuang atau diberi tanda bahwa eror. *Service Data Unit (SDU)* dapat langsung ditansmisikan tanpa segmentasi tergantung dari tipe yang ditransmisikan. Berbeda dengan kedua mode operasi lainnya, TM tidak memberikan layanan *packet-switched* melainkan *circuit-switched*.

2. ***Unacknowledge Mode (UM)***

Tidak ada protokol retransmisi yang digunakan dan pengiriman data tidak dijamin. data yang diterima dengan error dapat dibuang begitu saja atau diberi tanda bahwa terjadi eror tergantung dari konfigurasinya. Struktur PDU mengikutsertakan di dalamnya *sequence number* sehingga kesatuan dari PDU pada layer yang lebih tinggi dapat diobservasikan. *Segmentation*, *concantenation*, dan *padding* disediakan di dalam header yang ditambahkan pada data. Entitas RLC pada mode ini adalah dalam mode unidirectional karena tidak ada hubungan yang dibutuhkan antara *uplink* dan *downlink*.

3. ***Acknowledge Mode (AM)***

Pada mode ini digunakan Automatic Repeat Request (ARQ) yang dipergunakan untuk mekanisme retransmisinya. Pada saat RLC tidak berhasil mengirim data secara benar, misalnya maximum number retransmisinya telah maksimal atau waktu telah lewat, maka SDU akan dibuang dan pasangan *user* akan diinformasikan. *Segmentation*, *concantenation*, dan *padding* serta data yang terduplikasi disediakan di dalam header yang ditambahkan pada data. Entitas RLC pada mode ini adalah dalam mode bidirectional dan memiliki kemampuan untuk *piggybacking* akan status link pada arah *user* yang berlawanan. AM adalah mode normal untuk pengiriman layanan bertipe paket, misalnya web browsing dan email.

2.3 HSDPA

High-Speed Downlink Packet Access (HSDPA) merupakan pengembangan dari jaringan UMTS yang sudah ada. Kelebihan yang ditawarkan oleh HSDPA antara lain memberikan kapasitas data yang lebih besar (hingga 14,4 Mbps pada *downlink*), meningkatkan kualitas pelayanan *mobile data* bagi pelanggan, mengurangi keterlambatan (*delay*), dan meningkatkan kapasitas sistem tanpa memerlukan spektrum frekuensi tambahan, sehingga dapat mengurangi biaya layanan *mobile data* secara signifikan.

Kecepatan HSDPA bisa meningkat karena pemindahan fungsi pemrosesan lebih dekat ke "udara". Proses yang dipindahkan adalah proses penjadwalan jaringan dan retransmisi. Semula, kedua proses itu berjalan di pengendali jaringan radio. HSDPA juga menambah sebuah mekanisme pembagian *channel*. Dengan perubahan-perubahan itu, HSDPA mengingkatkan kapasitas jaringan.

Operator tidak perlu mengeluarkan biaya investasi yang besar untuk menghadirkan HSDPA untuk pelanggannya. HSDPA hanya membutuhkan *software upgrade* pada *Base Station* UMTS yang sudah ada. Implementasi HSDPA ke juga tidak membutuhkan *node* tambahan untuk dikembangkan, melainkan *inclusion* dari fungsi tambahan ke dalam node UMTS yang telah ada. Fungsi ini cukup membutuhkan *inclusion* dari sub-layer MAC baru, MAC-sh, yang dibangun untuk mendukung transport channel yang baru, *High Speed Downlink Shared Channel (HS-DSCH)*. Sebagai perbandingan, proses upgrade jaringan UMTS yang sudah ada untuk menjadi evolusi UMTS dengan HSDPA, jauh lebih sederhana dari pada implementasi *Enhanced Data rates for Global Evolution (EDGE)* ke dalam jaringan GSM. Selain itu, HSDPA menggunakan lebar *channel* 5 MHz yang merupakan *channel* yang juga digunakan jaringan UMTS. Dengan demikian, para operator sudah bisa menggunakan infrastruktur yang sudah ada tanpa membutuhkan *channel* khusus.

Adapun beberapa karakteristik HSDPA adalah:

1. Menggunakan *channel* transmisi yang digunakan bersama, terbagi secara waktu. *Channel* transmisi ini dipergunakan bersama-sama untuk:
    - *Total downlink radio resources*
    - *Channelization code*
    - *Transmission power*
2. Memungkinkan perubahan pengalokasian secara cepat untuk sejumlah besar *resource downlink*
3. Menggunakan *channel dedicated HS-DSCH*
4. Menggunakan *Adaptive modulation Coding (AMC)/Link Adaptation*
5. Menggunakan mekanisme *Hybrid* ARQ (*hybrid Automatic Repeat Request*) untuk *fast retransmission.*
6. *Fast cell selection*

2.4 Video Streaming

Video streaming merupakan suatu teknologi yang mampu mengirimkan file audio dan video digital secara *real time* pada jaringan tertentu dan mengizinkan pengolahan *steady* dan terus-menerus oleh end-user. Beberapa tool untuk mendukung *video streaming* antara lain adalah server khusus untuk penyimpanan file yang akan di-*download*, *web browser plug-ins* atau aplikasi *stand-alone* khusus yang digunakan klien untuk mengakses, metode kompresi (*codec*) yang digunakan untuk kompresi data, dan protokol transport untuk transfer optimal.

Protokol yang digunakan pada *video streaming* adalah sebagai berikut:

1. Protokol utama

   User Datagram Protocol (UDP), merupakan solusi lebih mudah dan lebih efisien namun menimbulkan banyaknya *data loss*.

   Transmission Control Protocol (TCP), menjamin penyampaian yang tepat namun dapat menuju timeout sehingga klien membutuhkan buffer yang cukup.

2. Protokol transport

   Real-time Streaming Protocol (RTSP), mengizinkan klien untuk mengendalikan streaming media server secara remote, mengeluarkan perintah VCR-like seperti 'play' dan 'pause' dan mengizinkan akses file berbasis waktu pada server.

   Real-time Transport Protocol (RTP), mendefinisikan standarisasi format paket untuk penyampaian audio dan video di internet.

   Real-time Transport Control Protocol (RTCP), merupakan protokol pengendalian paket data pada RTP yang juga berguna untuk menjamin QoS *video streaming*. RTCP digunakan secara periodik untuk mentransmisikan *control packet* untuk pengemasan pada sesi *video streaming*.

2.5 H.264

H.264 atau *MPEG-4 AVC (Advanced Video Coding)* adalah standar internasional terbaru untuk *video coding* yang dikembangkan oleh *Joint Video*

Team dari MPEG dan ITU-T. Teknologi H.264 sebenarnya berbasis pada standar MPEG yang sebelumnya. Hanya saja, standar ini mendapat penyempurnaan, diantaranya efisiensi *bit rate* hingga 40% dibandingkan *codec* MPEG-4 dengan kualitas yang sama, *frame size* 4 kali lebih besar dibandingkan dengan MPEG-4 Part 2 pada *data rate* yang sama, dan efisiensi kompresi yang sangat baik.

Teknologi H.264 mengantarkan kualitas yang sangat baik melintasi jangkauan operasi yang sangat luas, dari 3G hingga HD. H.264 menyediakan kualitas video yang luar biasa dan melalui *data rate* rendah yang mengesankan, baik dalam membuat video untuk *mobile phone*, iChat AV, internet, broadcast, atau transmisi satelit.

Tabel 2.2 Kualitas H.264 pada berbagai ukuran dan jenis aplikasi [6]

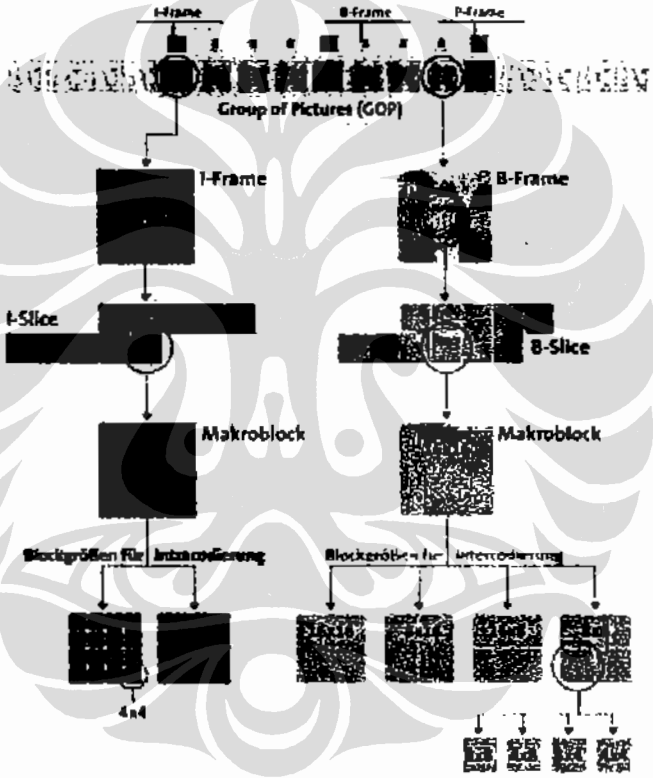
Scenario/Use	Resolution and frame rate	Example data rates
Mobile content (3G)	176 by 144, 10–24 fps	50–160 Kbps
Internet/Standard definition (SD)	640 by 480, 24 fps	1–2 Mbps
High definition (HD)	1280 by 720, 24p	5–6 Mbps
Full high definition (full HD)	1920 by 1080, 24p	7–8 Mbps

2.5.1 Penyusunan Data Video pada H.264

Sebuah film hanyalah sebuah dereten gambar-gambar yang disebut *Frame. Standar TV PAL* dapat mengirimkan 25 frame per detik. *Encoder H.264* akan menggabungkan deretan frame ini dalam sebuah deretan terpisah terlebih dahulu yang disebut *Group of Pictures (GOP)*. Sebuah GOP biasanya terdiri atas 12 sampai 15 frame. *Encoder* H.264 membagi-bagi setiap frame dalam *macroblock* menjadi 16x16 *pixel*. Dari *macroblock* ini, *encoder* akan menentukan nilai pencahayaan (*luminance*) dan nilai warna (*chromaticity*). serta mendefinisikannya ke dalam beberapa blok kecil yang berbeda. H.264 menggunakan 5 tipe ukuran, yaitu potongan tipe *Intra-frame* (*I-frame*), *Predicted-frame* (*P-frame*) dan *Bidirectional-frame* (*B-Frame*) I, P, dan B. Dua tipe lainnya, *Switching P (SP)* dan *Switching I (SI)*, jarang diproses dan berfungsi untuk mengolah data video dengan *bitrate* variabel secara efisien. Seperti diilustrasikan pada Gambar 2.2, pada setiap blok, *encoder H.264* akan menggunakan metode-metode pengkodean sebagai berikut:

1. **Intracoding**

Pada H.264 blok-blok ukuran 4x4 merupakan ukuran standar untuk *intracoding* nilai pencahayaan. Blok kecil yang proporsional ini memungkinkan *encoder* untuk memproses perubahan kecil dalam sebuah gambar. Jadi, H.264 dapat menghilangkan elemen-elemen asing yang biasanya muncul di pinggiran sebuah teks. Sebaliknya, pada jangkauan frekuensi rendah, yakni potongan gambar dengan sedikit detail seperti permukaan yang luas, blok gambar yang diproses *encoder* ini bisa menggunakan ukuran 16x16 *pixel* pada *intracoding*.



Gambar 2.2 Penyusunan data video pada H.264

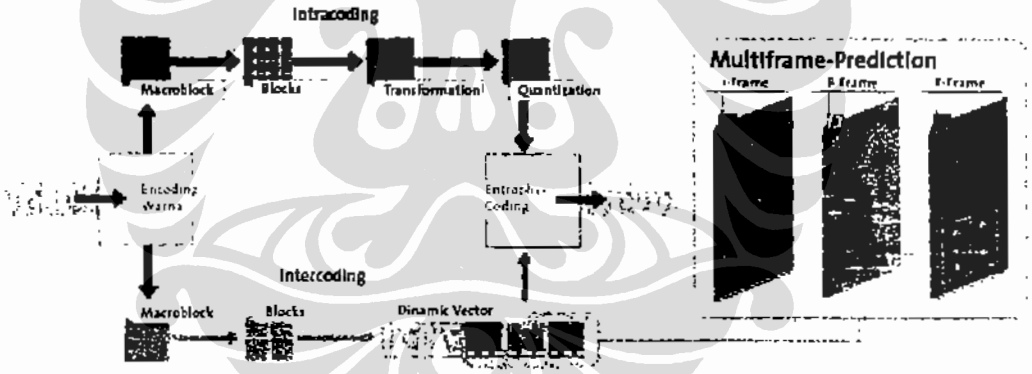
2. **Intercoding**

Dalam sebuah GOP, hanya ada sebuah *I-frame*. Frame yang lainnya, yaitu *P-frame* dan *B-Frame* akan di-*intercode*. H.264 melakukannya dengan sedikit fleksibel. *Encoder* ini bisa membuat blok ukuran berbeda dari 16x16 sampai 4x4. Untuk blok ukuran 8x8, *encoder* dapat membaginya lagi sampai berukuran 4x4. Frame-frame ini akan digambarkan dengan

dynamic vector untuk menampilkan perubahan dibandingkan dengan frame asal. Keuntungan dari *intercoding*, *encoder* dapat memproses setiap *macroblock* yang memiliki sampai dengan 16 *dynamic vector* yang berbeda, dan hasilnya tentu saja dapat merekam perubahan yang lebih akurat dari sebuah gambar ke gambar lain dengan menggunakan *dynamic vector*.

2.5.2 Proses Encoding pada H.264

Pada proses *encoding* berdasarkan lingkup warna, *encoder* memproses framenya melalui dua cara, seperti dapat dilihat pada Gambar 2.3. Pada proses *intracode* (merah), hanya posisi tengah dari *brightness* dan informasi warna saja yang dikompresi. Sedangkan pada proses *intercode* (biru), *encoder*-nya menganalisa bagian-bagian mana saja dari sebuah frame yang mengalami perubahan dibandingkan dengan frame lainnya. Informasi ini digambarkan dalam bentuk sebuah *dynamic vector*.



Gambar 2.3 Proses Encoding H.264

Pada metode *intracoding*, gambar pertama (*I-frame*) dalam GOP selalu di-*intracode*. Artinya, *encoder* hanya menggunakan informasi-informasi yang ada dalam gambar untuk proses *encoding*. Ini bisa dilakukan melalui *Separated Integer-Transformation*. Disini, informasi gambar dikonversikan menjadi frekuensi dan dioptimalkan menjadi blok berukuran 4x4 *pixel* sehingga kualitas gambar bisa bertambah. Saat proses *quantizing*, frekuensi-frekuensi ini dibulatkan pada seluruh angka sehingga dalam jangkauan frekuensi tinggi akan banyak nilai

yang terhapus. Nilai yang telah ditentukan ini akhirnya dikompresi tanpa kehilangan data, melalui metode *Entrophy Encoding*.

Pada metode intercoding, H.264 dapat melakukan hal yang lebih banyak dengan *dynamic vector*. Semakin banyak informasi gambar yang direkam oleh *encoder* mengenai *dynamic vector*, semakin tinggi tingkat kompresi data videonya. *Encoder* H.264 tidak hanya dapat mengkontruksi *dynamic vector* dari frame sebelumnya atau setelahnya, melainkan juga dari sejumlah frame yang dipilih, baik letaknya di depan maupun di belakangnya. Selain itu, H.264 juga dapat menentukan *dynamic vector* dari *B-Frame*. Hasilnya, kompresi lebih menghemat ruang, karena semua tipe frame bisa dijadikan sebagai gambar referensi untuk *dynamic vector*.

Proses terakhir adalah *entrophy coding* yang akan mengkompresi data hasil *intracoding* dan *intercoding* sebelumnya tanpa ada data yang hilang. Prosesnya sama seperti kompresi ZIP sebuah file. Untuk itu, H.264 melakukan dua metode, yaitu *Context Adaptive Variable Length Coding (CAVLC)* sehingga *encoder* menghasilkan lebih banyak *code table* dan kemudian dipilih berdasarkan *data analyse*. Hasilnya, kompresi menjadi optimal. Metode yang lain adalah *Context Adaptive Binary Arithmetic Coding (CABAC)* yang lebih kuat. Disini, tidak ada tabel yang dihasilkan seperti pada CAVLC. CABAC sudah merupakan tabel itu sendiri sehingga kompresi jadi lebih baik. Dibandingkan dengan CAVCL, CABAC bekerja lebih efisien sekitar 10 sampai 15 persen.

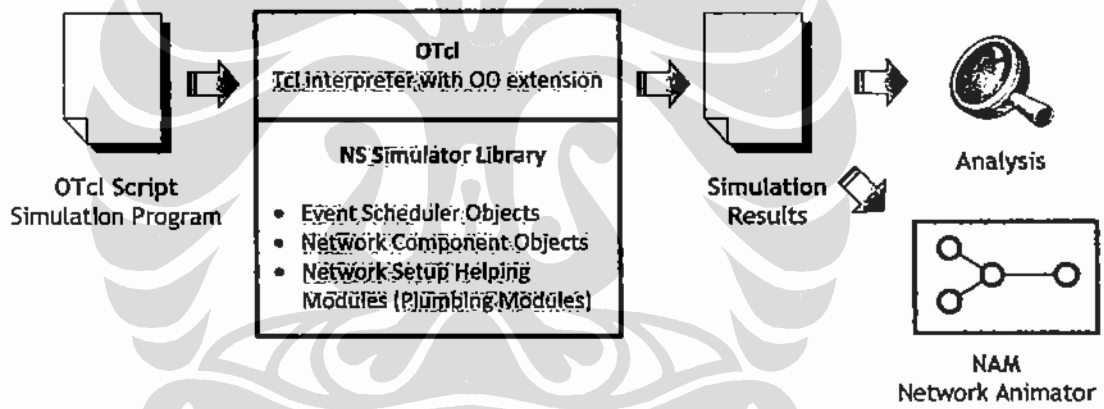
2.6 Network Simulator-2

Network Simulator-2 (NS-2) dikembangkan pertama kali tahun 1995 di University of California Berkeley yang didukung oleh DARPA. NS-2 adalah aplikasi yang dapat mensimulasikan suatu jaringan sesuai dengan keadaan sebenarnya yang berjalan diatas *operating system* Unix/Linux. NS-2 juga dapat dijalankan dalam system Windows namun harus menggunakan Cygwin sebagai *enviroment* Linux-nya.

Pada dasarnya NS-2 adalah interpreter Otcl dengan objek *library network simulation*. Simulator ini mendukung hierarki *class* dalam C++ (*compiled hierarchy*) dan hierarki *class* yang serupa pada interpreter OTcl (*interpreted*

hierarchy). Dua hierarki ini saling terkait satu sama lain. Root dari hierarki ini adalah TclObject. User membuat obyek Simulator (dari *class Simulator*) baru melalui interpreter, dan dicerminkan dengan obyek sebanding pada *compiled hierarchy*. Setelah obyek Simulator dibuat, maka metode-metode untuk membuat topologi, node, dan komponen jaringan lainnya dipanggil.

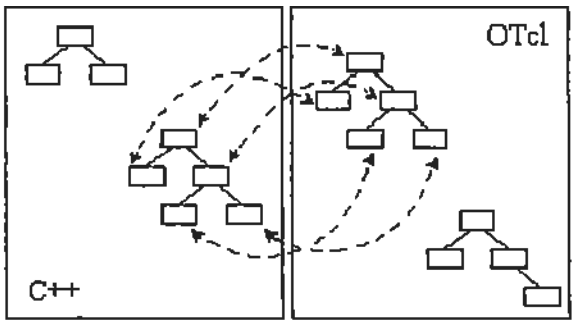
NS-2 juga merupakan simulator yang dipicu oleh event (*event-driven simulator*). *Scheduler* berjalan dengan cara mengeksekusi *event* berikutnya yang paling dahulu sampai selesai, kemudian kembali menjalankan *event* berikutnya. Walaupun jaringan berkomunikasi dengan melewatkan paket, namun itu tidak menghabiskan banyak waktu. Jika komponen yang dibuat memerlukan *delay*, maka digunakan *event scheduler* untuk menerbitkan event untuk paket tersebut dan menunggu event berhenti sebelum event berikutnya dijalankan. Flowchart sistem kerja NS secara global dapat dilihat pada Gambar 2.4 [7].



Gambar 2.4 Flowchart Sistem Kerja NS-2

2.6.1 Komponen Pembangun NS-2

Pengetahuan tentang komponen pembangun NS dan letaknya akan sangat berguna dalam membangun simulasi. Komponen pembangun NS dapat dilihat pada Gambar 2.5 [7] berikut.



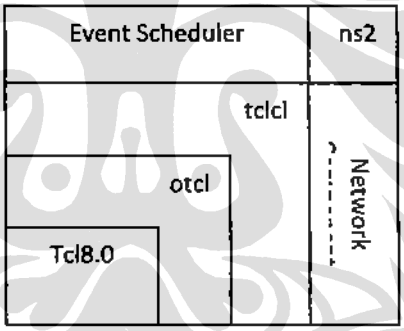
Keterangan:

Tcl	: Tool command language	Tclcl	: Tcl/C++ Interface
Otcl	: Object Tcl	NS-2	: NS versi 2
TK	: Tool Kit	Nam	: *Network* animator

Gambar 2.5 Komponen Pembangun NS-2

2.6.2 Hubungan Antar Komponen Pembangun NS-2

Deskripsi dibawah ini menunjukkan struktur umum hubungan antar komponen pembangun NS.

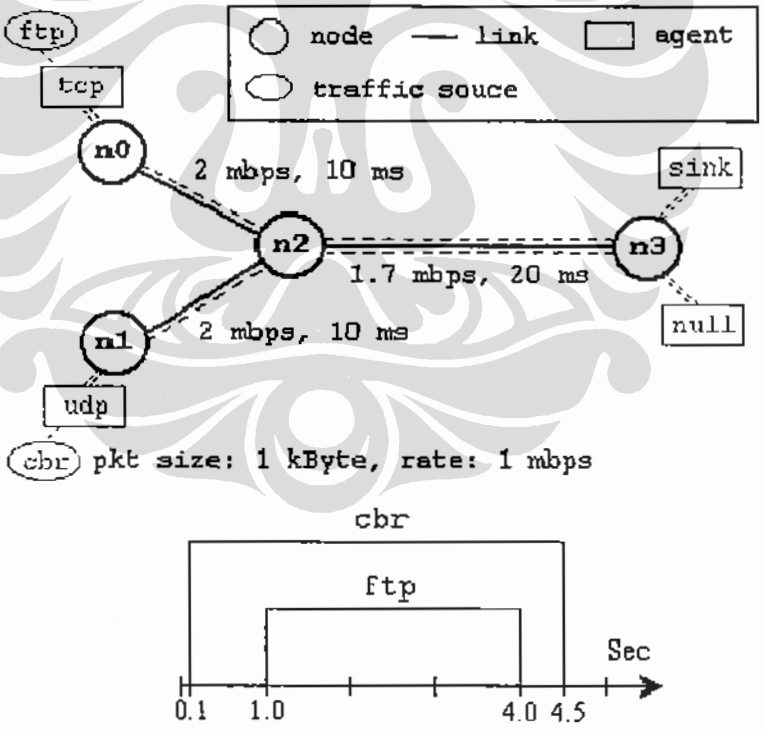


Gambar 2.6 Hubungan Antar-Komponen Pembangun NS-2

Berdasarkan deskripsi pada Gambar 2.6 [7], pengguna NS-2 berada pada pojok kiri bawah, melakukan desain dan menjalankan simulasi dalam bahasa Tcl. Dalam simulasi pengguna NS memanggil dan menggunakan objek simulator pada *library* Otcl. *Event scheduler* dan sebagian besar *network* component pada NS ditulis dalam bahasa C++. Event scheduler dan *network* component ini diakses oleh Otcl melalui Otcl linkage yang diimplementasikan dengan menggunakan Tclcl. Contoh *script* dari NS-2 dapat dilihat pada skema berikut:

```
# Membuat node
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
# Menambahkan link
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n3 $n2 1Mb 10ms DropTail
# layout
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right
# right(0), right-up(45), right-down(315), left(180),
# left-up(135),left-down(225), up(90), down(270)
```

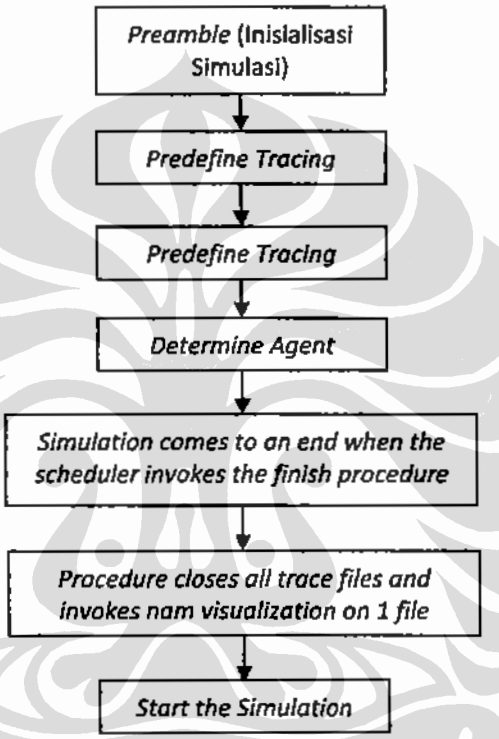
Script ini membangun konektivitas 4 *node* yang masing-masing terhubung dengan *link DropTail*. Jika kita menjalankan script diatas pada simulasi NS-2 maka akan menghasilkan topologi jaringan seperti pada Gambar 2.7 berikut [7].



Gambar 2.7 Simulasi Topologi Jaringan

2.6.3 Cara Membuat dan Menjalankan *Script* NS

Script simulasi dibuat dengan teks editor dan disimpan dengan nama file *extension* .tcl. Untuk menjalankan *script* yang telah dibuat tinggal masuk ke dalam folder tersebut dan mengetikkan NS serta nama file tcl simulasi yang ingin dijalankan. Jika dilakukan perintah 'ns filename.tcl', maka NS akan melakukan eksekusi terhadap file 'filename.tcl' dengan flowchart seperti terlihat pada Gambar 2.8 [7].

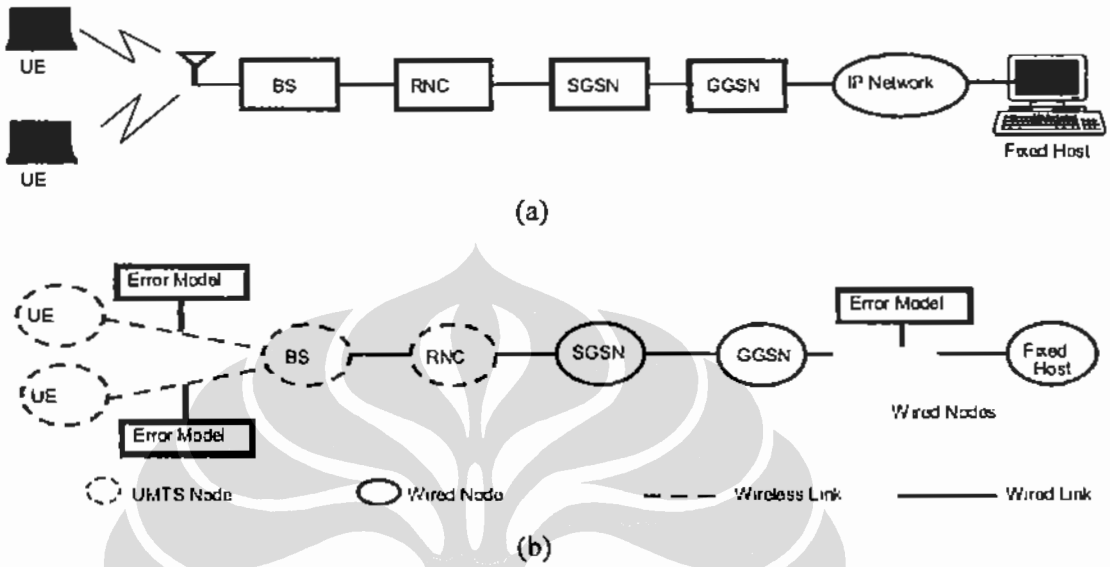


Gambar 2.8 Flowchart pada 'ns filename.tcl'

2.7 EURANE

Enhanced UMTS Radio Access Network Extensions for NS-2 (EURANE) [2] dibangun dan dikembangkan oleh European Commission 5th *framework* melalui projeknya yang bernama *Simulation of Enhanced UMTS access and Core Network (IST-SEACORN)*. EURANE menerapkan topologi jaringan UMTS pada simulasi NS-2, yang mencakup teknologi *radio interface*, *access network*, dan *core network*. Tujuan akhirnya adalah untuk mengevaluasi performa *end-to-end* dari Enhanced UMTS dan menguji QoS dari sistem UMTS dan memperoleh QoS yang bisa direalisasi dengan optimisasi parameter yang dapat dikonfigurasi dan

penambahan algoritma. Struktur environment pada *top-level* sesungguhnya adalah seperti terlihat pada Gambar 2.9a dan simulasi environment tersebut pada NS-2 dapat dilihat pada Gambar 2.9b [2].



Gambar 2.9 Struktur Jaringan (a) *Real Network* (b) Simulasi pada NS-2

EURANE mengandung tiga *node* tambahan, yaitu *Radio Network Controller (RNC)*, *Base station (BS)*, dan *User Equipment (UE)*, yang secara fungsinya dapat mendukung transport-transport *channel* seperti *Forward Link Access Channel (FACH)*, *Random Access Channel (RACH)*, *Dedicated Channel (DCH)*, dan *High Speed Downlink Shared Channel (HS-DSCH)*. Lebih lanjut node-node dan transport channel EURANE akan dijelaskan berikut ini.

1. **Radio Network Controller (RNC)**

   RNC merupakan *node* pertama yang harus dikonfigurasi, tetapi juga yang termudah. Konfigurasi RNC hanya membutuhkan *script* berikut:

```
$ns node-config -UmtsNodeType rnc
set rnc [$ns create-Umtsnode]
```

2. **Base Station (BS)**

   BS merupakan node yang selanjutnya dikonfigurasi dimana terdapat parameter tambahan yang berhubungan dengan *bandwidth* dan TTI dari FACH (*downlink*) dan RACH (*uplink*), dimana secara parsial dilakukan setup selama konfigurasi BS.

```
$ns node-config -UmtsNodeType bs \
                -downlinkBW 32kbs \
                -downlinkTTI 10ms \
                -uplinkBW 32kbs \
                -uplinkTTI 10ms
```

3. **User Equipment (UE)**

   Akhirnya UE dapat dibangun dan dikonfigurasi dengan perintah berikut:

```
$ns node-config -UmtsNodeType ue \
                -basestation $bs
                -radioNetworkController $rnc
set ue1 [$ns create-Umtsnode]
```

4. **FACH dan RACH**

   Kedua komponen *link layer* ini dibentuk pada saat konfigurasi *Base Station* dan *User Environment*, dan hanya membutuhkan prosedur *attachment* pada entitas RLC. *Script*-nya adalah sebagai berikut:

```
$ns attach-common <node> <agent>
# untuk banyak aplikasi
$ns attach-dch <node> <agent> <dch>
```

5. **HS-DSCH**

   Seperti halnya DCH, HS-DSCH juga harus dapat dibangun dengan baik. Implementasinya adalah seperti pada *script* berikut:

```
$ns create-hsdsch <node> <agent>
$ns attach-hsdsch <node> <agent>
```

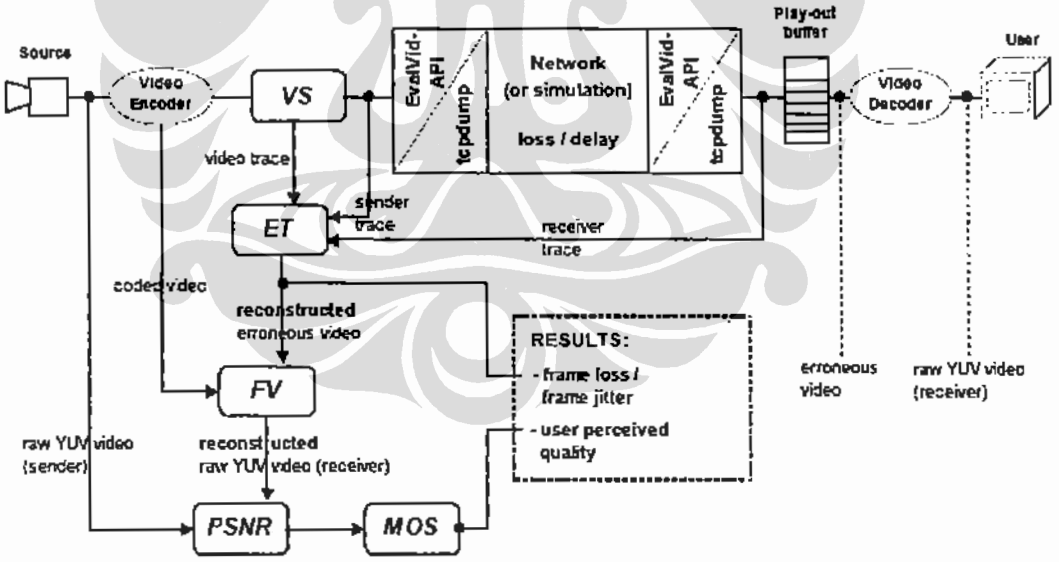
Fungsi utama dari penambahan fungsi NS-2 muncul dalam bentuk *RLC Acknowledged Mode (AM)*, *Unacknowledged Mode (UM)*, *MAC-d/-c/sh support*, dan *MC-hs support* untuk HS-DSCH, seperti HSDPA. Untuk menggabungkan modul EURANE ke dalam simulasi NS-2 diperlukan proses instalasi *patch* EURANE pada folder NS-2.

2.8 EvalVid

Evaluation Video (EvalVid) merupakan pengukuran kualitas video yang dikembangkan oleh *Technical University of Berlin, Telecommunicion Networks*

Group (TKN). EvalVid menyediakan *framework* dan kumpulan tool untuk mengevaluasi kualitas video yang ditransmisikan pada jaringan komunikasi yang asli atau simulasi. Disamping mengukur parameter QoS pada jaringan utama, seperti *loss rate*, *delay*, dan *jitter*, EvalVid juga mendukung evaluasi kualitas video subyektif dari video yang diterima berdasarkan perhitungan PSNR *frame-by-frame*. Kumpulan tool EvalVid memiliki konstruksi modular, yang memungkinkan dilakukannya pertukaran jaringan dan *codec*.

Struktur framework EvalVid dapat dilihat pada Gambar 2.10 [8], yang mengilustrasikan interaksi antara arus data dan tool yang diimplementasikan. Framework ini berisi transmisi lengkap dari video digital mulai dari source video, *reordering* pada source, *encoding*, paketisasi, transmisi jaringan, reduksi *jitter* oleh *buffer play-out*, *decoding*, hingga tampilan video yang diterima oleh *end-user*. Selama pengoperasiannya, data yang diproses pada arus transmisi akan ditandai dan disimpan pada file-file yang beranekaragam. File-file ini kemudian digunakan untuk memperoleh hasil yang diinginkan, seperti *loss rate*, *jitter*, dan kualitas video.



Gambar 2.10 Skema framework evaluasi pada Evalvid

Beberapa komponen utama pada framework evaluasi EvalVid ini dijelaskan secara singkat di bawah ini. Adapun untuk penjelasan lebih lanjut berikut implementasinya akan dijabarkan secara detail pada Bab 3.

- **Source.** Source video dapat berupa baik dalam format YUV QCIF (176 x 144) atau pada YUV CIF (352 x 288).
- **Video Encoder dan Video Decoder.** EvalVid mampu mendukung berbagai jenis codec MPEG4, yaitu antara lain: National Chiao Twig University (NCTU) codec, ffmpeg, Xvid, dan H.264.
- **Video Sender (VS).** Komponen VS membaca file video yang dikompresi dari output *video encoder*, memfragmentasi masing-masing frame video yang besar menjadi segmen-segmen yang kecil, dan kemudian mengirimkan segmen-segmen ini via paket UDP pada *real network* atau simulasi. Untuk tiap paket UDP yang ditransmisikan, framework merekam *timestamp*, *packet ID*, dan *packet payload size* pada file trace pengirim dengan bantuan *third-party tool*. Untuk *real-network* dapat digunakan *tcp-dump* atau *win-dump*, sedangkan untuk simulasi jaringan, dapat digunakan NS-2, Qualnet, atau OPNet. Komponen VS juga men-*generate* file trace video yang berisi informasi tentang tiap frame pada file video real. File trace video dan file trace pengirim digunakan nanti untuk evaluasi kualitas video berikut.
- **Evaluate Trace (ET).** Evaluasi berlangsung pada sisi pengirim ketika transmisi video berakhir. Berdasarkan file video asli yang di-*encode*, file trace video, file trace pengirim, dan file trace penerima, komponen ET membuat laporan *frame/packet loss* dan *frame/packet jitter* serta men-*generate* rekonstruksi file video, yang berkaitan dengan kemungkinan video *corrupt* pada sisi penerima yang akan direproduksi pada *end-user*.
- **Fix Video (FV).** Pengujian kualitas video digital dilakukan frame demi frame. Oleh karena itu, jumlah frame video pada sisi penerima termasuk frame *erroneous* harus sama dengan video asli pada sisi pengirim. Jika *codec* tidak dapat menangani frame yang hilang, komponen FV digunakan untuk memecahkan masalah dengan memasukkan frame *decode* di tiap frame yang hilang sebagai metode penyembunyian error.
- **Peak Signal Noise Ratio (PSNR).** PSNR merupakan salah satu metric obyektif yang banyak digunakan untuk menguji QoS pada level aplikasi dari transmisi video.

- **Mean Opinion Score (MOS)**. MOS adalah metric subyektif untuk mengukur kualitas video digital pada level aplikasi.

Box pada Gambar 2.10 yang bernama VS, ET, FV, PSNR dan MOS merupakan program sebenarnya pada framework. Adapun data yang dibutuhkan untuk implementasi pengukuran dengan menggunakan EvalVid ini antara lain:

1. Dari sisi pengirim:
   - *raw uncompressed video*
   - *encoded video*
   - *time-stamp* dan tipe dari tiap paket yang dikirimkan
2. Dari sisi penerima:
   - *time-stamp* dan tipe dari tiap paket yang diterima
   - *reassembled encoded video* (kemungkinan *errorneous*)
   - *raw uncompressed video* untuk di-*display*

Untuk tool di dalam EvalVid hanyalah dibutuhkan file-file *trace* ini, file video asli dan *decoder*. Oleh karena itu, konteks dari EvalVid pada jaringan adalah hanya "*black box*" yang meng-*generate delay*, *loss*, dan *reordering* paket yang mungkin. Hal ini dapat berupa *real link* seperti Ethernet atau WLAN maupun simulasi atau emulasi dari jaringan. Karena hanya interaksi dari EvalVid dan jaringan yang direpresentasikan oleh dua file *trace* (pengirim dan penerima), *network box* dapat dengan mudah digantikan, yang membuat EvalVid sangat fleksibel. Demikian halnya dengan *codec* video, dapat juga dengan mudah digantikan.

2.9 Parameter Kinerja

Seperti yang telah disebutkan sebelumnya, penelitian ini bertujuan untuk mengukur kualitas dari performa layanan *video streaming*. Oleh karena itu dibutuhkan parameter-parameter QoS dan parameter-parameter kualitas video yang mendukung. Parameter tersebut antara lain *packet loss*, *frame loss*, *delay*, dan *jitter* untuk parameter QoS, serta PSNR dan MOS untuk parameter kualitas video.

2.9.1 Packet Loss

Packet loss merupakan ukuran *error rate* dari transmisi paket data yang diukur dalam persen. Pada umumnya *packet loss* dikarenakan *buffer* yang terbatas dan urutan paket yang salah. Packet loss biasanya dihitung berdasarkan *packet identifier (packet id)*. Karenanya *network black box* harus menyediakan *packet id* yang unik. Hal ini tidak menjadi masalah dalam simulasi, karena id unik dapat di-*generate* dengan mudah. Pada pengukuran, *packet id* seringkali diambil dari IP, yang menyediakan *packet id* yang unik. *Packet id* yang unik juga digunakan untuk membatalkan efek *reordering*.

Pada konteks transmisi video tidak hanya menarik berapa banyak paket yang hilang, tetapi juga jenis data yang terkandung di dalam paket. Contohnya, *codec* H.264 mendefiniskan 5 tipe frame yang berbeda yaitu I, P, B, SP, dan SI, serta sejumlah *header* umum. Hal ini dikarenakan pentingnya membedakan antara jenis paket yang berbeda di dalam transmisi video. Evaluasi *packet loss* seharusnya dapat dilakukan oleh tipe *dependent* (*tipe frame*, *header*). *Packet loss* didefinisikan pada Persamaan 2.1 dalam satuan persen [8].

$$PL_T = 100\,\frac{nT_{recv}}{nT_{sent}} \qquad (2.1)$$

dimana T : Type data paket (salah satu dari *header*, I, P, B, SP, SI)

nT_{sent} : jumlah tipe paket T yang terkirim

nT_{recv} : jumlah tipe paket T yang diterima

2.9.2 Frame Loss

Sebuah frame video (sebenarnya menjadi *single coded image*) relatif berukuran besar. Tidak hanya pada kasus video VBR, tetapi juga CBR, karena masa tetap mengaplikasikan pada waktu pendek rata-rata. *I-frame* biasanya dianggap lebih besar dari target (*short time average*) CBR bahkan pada video CBR.

Sangatlah mungkin beberapa atau kemungkinan seluruh frame lebih besar dibandingkan *maximum transfer unit (MTU)* pada jaringan. Ini adalah ukuran paket maksimum yang didukung oleh jaringan (seperti Ethernet = 1500 dan 802.11b WLAN = 2312 bytes). Frame-frame ini harus disegmentasi pada paket

yang lebih kecil yang cocok pada jaringan MTU. Kemungkinan segmentasi frame ini menimbulkan masalah perhitungan dari *frame loss*. Pada dasarnya *frame loss rate* dapat diperoleh dari *packet loss rate*. Tetapi proses ini tergantung kapabilitas dari kegunaan *decoder* video aktual, karena beberapa *decoder* dapat memproses frame walaupun beberapa bagian hilang. Lebih lanjut, frame dapat di-*decode* tergantung pada dimana paket hilang. Jika paket pertama hilang, frame hampir tidak bisa didekode. Kemudian, kapabilitas dari beberapa *decoder* harus dipertimbangkan dengan tujuan untuk menghitung *frame loss rate*. Persamaan 2.2 menghitung tipe frame masing-masing [8].

$$FL_T = 100 \frac{nT_{recv}}{nT_{sent}} \quad \text{..... (2.2)}$$

dimana T : Type data frame (salah satu dari *header*, I, P, B, SP, SI)

nT_{sent} : jumlah tipe frame T yang terkirim

nT_{recv} : jumlah tipe frame T yang diterima

2.9.3 Delay dan Jitter

Ukuran *delay* penerimaan paket yang melambangkan *smoothness* dari *audio/video playback*. Pada sistem transmisi video tidak hanya *loss* actual penting untuk kualitas video yang dirasakan, tetapi juga *delay* dari frame dan *delay variation* (*frame jitter*). Video digital selalu terdiri dari frame yang harus ditampilkan pada *constant rate*. Menampilkan frame sebelum atau setelah waktu yang didefinisikan menghasilkan "*jerkiness*". Isu ini dialamatkan oleh yang dinamakan *play-out buffer*. *Buffer* ini memiliki tujuan menyerap *jitter* yang ditimbulkan oleh *delay* penyampaian network. Hal ini jelas *play-out buffer* yang cukup besar yang dapat mengkompensasi sejumlah *jitter*. Pada kasus yang ekstrim, *buffer* dapat sebesar seluruh video dan melakukan start tidak setelah frame terakhir diterima. Hal ini akan mengeliminasi beberapa kemungkinan *jitter* pada waktu *delay* tambahan dari seluruh waktu transmisi. Hal ekstrim lainnya yaitu kapabilitas *buffer* dalam memegang 1 frame dengan tepat. Pada kasus ini tidak ada *jitter* sama sekali yang dapat dieliminasi tetapi tidak ada *delay* tambahan yang ditimbulkan.

Terdapat metode cerdas yang dikembangkan untuk mengoptimisasi *play-out buffer* berkenaan dengan *particular trade-off*. Metode ini tidak di dalam cakupan framework yang dijelaskan. Ukuran *play-out buffer* hanya sebuah parameter pada proses evaluasi. Hal ini terbatas pada framework *play-out buffer* statik. Bagaimanapun, karena strategi integrasi *play-out buffer* pada proses evaluasi, *loss* tambahan dikarenakan *play-out buffer* dapat dipertimbangkan.

Definisi formal jitter sebagaimana yang digunakan pada paper ini diberikan pada Persamaan 2.3, 2.4, dan 2.5. Ini adalah variasi dari waktu *inter-packet* atau *inter-frame*. "*Frame time*" ditentukan oleh waktu dimana segmen terakhir dari frame yang telah tersegmentasi diterima.

inter-packet time ..... **(2.3)**

$$it_{p_o} = 0$$
$$it_{p_o} = t_{p_n} - t_{p_{n-1}}$$

dimana: t_{pn} : time-stamp dari paket n

inter-frame time

$$it_{F_o} = 0$$
$$it_{F_m} = t_{F_m} - t_{F_{m-1}}$$

dimana: t_{pn} : time-stamp dari segmen terakhir dari frame m

packet jitter

$$j_P = \frac{1}{N}\sum_{i=0}^{N}(it_i - \bar{it}_N)^2 \quad \text{..... (2.4)}$$

dimana: N : paket N

i_{tN} = rata-rata dari waktu *inter-packet*

frame jitter

$$j_F = \frac{1}{M}\sum_{i=1}^{M}(it_i - \bar{it}_M)^2 \quad \text{..... (2.5)}$$

dimana: N : frame N

i_{tN} = rata-rata dari waktu *inter-frame*

2.9.4 Evaluasi Kualitas Video

Pengukuran kualitas video digital harus berdasarkan pada kualitas yang dirasakan pada video aktual yang sedang diterima oleh user pada sistem digital video karena kesan user adalah apa yang dihitung di akhir. Terdapat dua pendekatan dasar untuk mengukur kualitas video digital, yaitu pengukuran

kualitas subyektif dan pengukruan kualitas obyektif. Pengukuran kualitas subyektif selalu merenggut faktor krusial, kesan user melihat video ketika sedang berhemat, menghabiskan banyak waktu, kebutuhan sumberdaya manusia yang tinggi dan perlengkapan khusus yang dibutuhkan. Beberapa metode obyektif dijelaskan secara detail pada ITU, ANSI, dan MPEG. *Human quality impression* biasanya diberikan pada skala dari 5 (terbaik) ke 1 (terburuk) sebagaimana terlihat pada Tabel 2.3 [8]. Skala ini dinamakan *Mean Opinion Score (MOS)*.

Tabel 2.3 Kualitas dan Tingkat Kerusakan ITU-R [8]

Skala	Kualitas	Kerusakan
5	Excellent	Imperceptible
4	Good	Perceptible, but not annoying
3	Fair	Slightly annoying
2	Poor	Annoying
1	Bad	Very annoying

Beberapa *task* di industri dan penelitian membutuhkan metode otomatis untuk mengevaluasi kualitas video. Tes subyektif yang kompleks dan mahal seringkali tidak dapat diterima. Oleh karena itu, *objective metric* telah dikembangkan untuk mengemulasi *quality impression* dari *human visual system (HVS)*. Terdapat diskusi mendalam dari berbagai macam *objective metric* dan performa mereka dibandingkan dengan tes subyektif.

2.9.4.1 Peak Signal to Noise Ratio (PSNR)

Metode pengukuran kualitas video yang tersebar luas adalah perhitungan dari *peak signal to noise ratio (PSNR)* gambar demi gambar. PSNR merupakan turunan dari *signal to noise ratio (SNR)* yang membandingkan sinyal energi dengan error energi. PSNR merupakan dasar dari *quality metric* yang digunakan pada framework untuk menguji hasil dari kualitas video.

PSNR membandingkan kemungkinan maksimum sinyal energi dengan error energi, dimana telah memperlihatkan hasil pada korelasi yang lebih tinggi dengan persepsi kualitas subyektif dengan SNR yang konvensional. Persamaan

2.6 adalah definisi dari PSNR antara komponen *luminance* Y dari gambar sumber S dan gambar tujuan D.

$$PSNR(n)_{dB} = 20\,log_{10}\left(\frac{V_{peak}}{\sqrt{\frac{1}{N_{col}N_{row}}\sum_{i=0}^{N_{col}}\sum_{i=0}^{N_{row}}[Y_S(n,i,j)-Y_D(n,i,j)]^2}}\right) \quad \text{..... (2.6)}$$

dimana : $V_{peak} = 2^k - 1$

k = jumlah bit per pixel (komponen *luminance*)

Bagian di penyebut merupakan *mean square error (MSE)*. Formula untuk PNSR dapat di-ringkas sebagai $PSNR = 20\log\frac{V_{peak}}{MSE}$. Karena PSNR dihitung berdasarkan *frame by frame* dapat membuat ketidaknyamanan, ketika diaplikasikan pada video yang terdiri dari beberapa ratus atau ribu frame. Lebih lanjut, orang-orang sering tertarik pada distorsi yang diperkenalkan oleh jaringan sendiri. Jadi mereka ingin membandingkan video yang diterima (kemungkinan distorsi) dengan video yang dikirmkan yang tidak terdistorsi. Hal ini dapat dilakukan dengan membandingkan PSNR dari *envoced video* dengan frame video yang diterima *frame by frame* atau membandingkan rata-rata mereka dan standar deviasi.

2.9.4.2 Mean Opinion Score (MOS)

Oleh karena *time series* PSNR sangat tidak ringkas, *metric* tambahan disediakan. PSNR dari tiap frame tunggal akan dipetakan pada skala MOS seperti terlihat pada Tabel 2.4 [8]. MOS merupakan *human impression* dari kualitas video, dimana diberikan pada skala dari 5 ke 1. Skala 5 merujuk pada kualitas terbaik sedangkan skala 1 untuk kualitas terburuk. Metode ini memiliki keuntungan untuk memperlihatkan dengan jelas distorsi yang disebabkan oleh *network at a glance*.

Untuk memperoleh MOS, nilai PSNR pada video yang diberikan ditentukan dengan menggunakan program PSNR, yang kemudian memetakannya pada skala MOS. Program MOS digunakan untuk melakukan pemetaan. PSNR menghitung kemungkinan maksimum *signal energy to noise energy*, dimana secara matematis sama dengan *root mean squared error*.

Tabel 2.4 Konversi PSNR ke MOS

PSNR (dB)	MOS
> 37	5 (Excellent)
31 – 37	4 (Good)
25 – 31	3 (Fair)
20 – 25	2 (Poor)
< 20	1 (Bad)