BAB III
PERANCANGAN SIMULASI JARINGAN

Penelitian ini dilakukan dengan membuat 2 jenis simulasi dengan topologi yang sama tetapi berbeda skema, dimana pada simulasi pertama seorang *user* mengakses *streaming video* dengan mekanisme FEC yang statis, Sedangkan pada skema yang kedua *user* mengakses *streaming video* dimana pada *Access Point* (telah ditambahkan Skema EAFEC) dimana kedua skema ini memiliki topologi yang sama. Topologi yang digunakan adalah seperti pada gambar 3.1 dibawah ini. Simulasi ini menggunakan *Network simulator* 2 (NS2) 2.28. dengan

Rate transmisi untuk *traffic* : 1 Mbps

Time burst : 0.5 ms

Time idle : 0.5 ms

Link antara *wireless* dan AP : 11 Mbps.

Link antara internet dan *Access Point* : 100Mbps

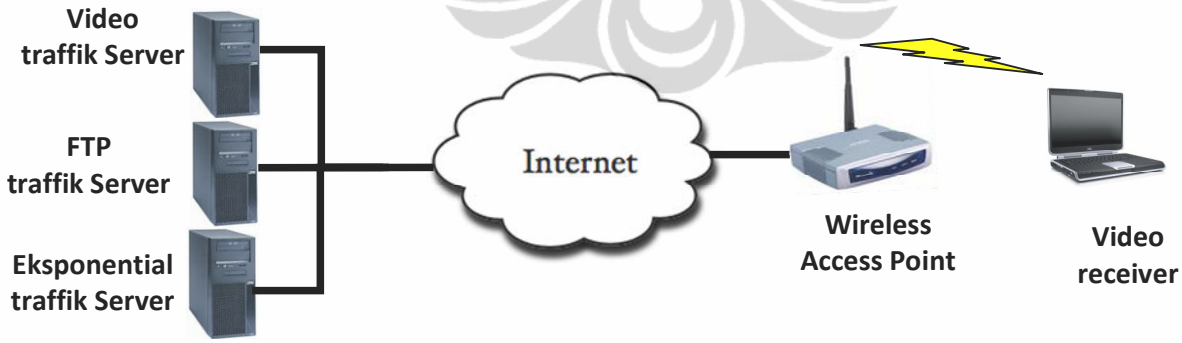
Link antara *source traffic* dan internet : 10 Mbps.

Nilai *threshold*1 : 5

Nilai *threshold*2 : 40

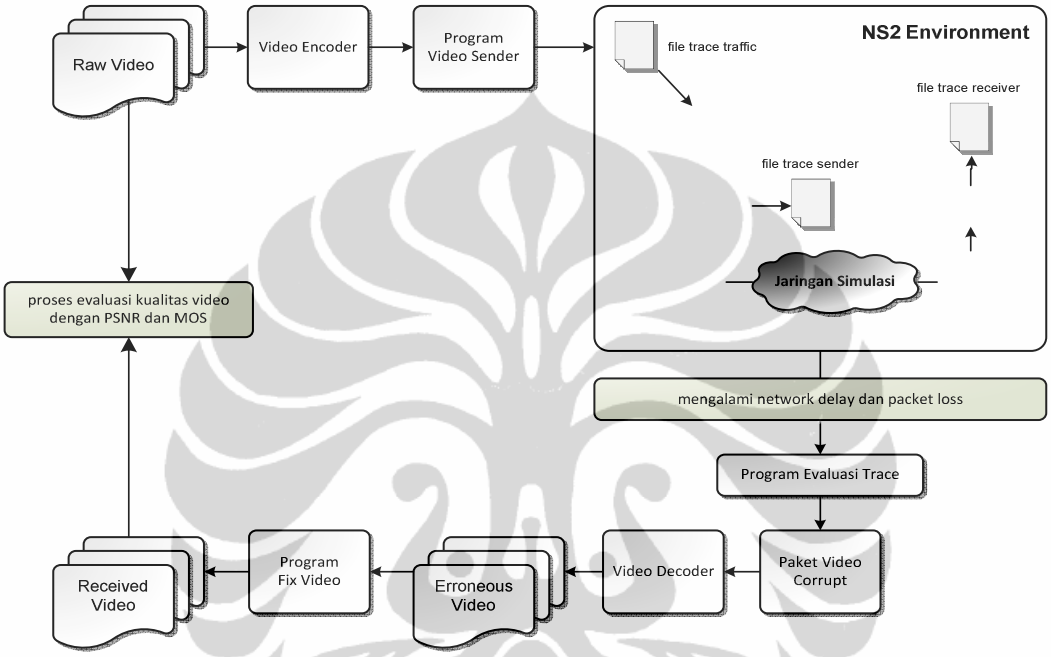
Nilai *threshold*3 : 5

Nilai *threshold*4 : 15



Gambar 3.1 Topologi Simulasi *Streaming video Over Wireless Network.*

Implementasi penelitian terdiri dari *encoder* H.264, video sender, Network *Simulator* (NS-2), H.264 *decoder*, program *Evaluate Trace (ET)*, program PSNR. Gambar 3.2 menggambarkan hubungan antara komponen-komponen yang berbeda, *file* video input, dan *file* output yang di-*generate* dari *tool -tool* yang digunakan pada penelitian ini.



Gambar 3.2 Alur Proses Keseluruhan

3.1 Skenario yang digunakan pada simulasi

Dari topologi diatas dapat dilihat bahwa data yang akan dikirimkan ke *node* akan melalui *Access Point* (AP) dan kemudian baru diteruskan ke *user*. Oleh karena itu *Access Point* merupakan tempat yang cocok untuk menambahkan mekanisme FEC agar terjadi peningkatan kualitas video. Untuk EAFEC , *Access Point* secara dinamis menentukan berapa banyak *redundant* paket yang akan di-*generate*, yang didasari oleh kondisi jaringan pada saat tersebut.

Hal yang dilakukan AP pertama kali adalah melakukan perkiraan lalu lintas jaringan. Misalnya, jika beban jaringan tinggi maka antrian akan panjang dan jika tidak maka sebaliknya antrian akan lebih pendek. Ketika antrian bertambah panjang, lebih sedikit *redundant* FEC yang akan dihasilkan untuk menghindari kemacetan yang tidak perlu pada jaringan. Disisi lain, waktu *Access*

Point melakukan retransmisi itu digunakan untuk menjadi indikator status jaringan. Ketika jaringan dalam kondisi baik maka packet pada retransmisi akan sedikit paket *redundant* yang dibutuhkan dan jika tidak maka akan dibutuhkan paket lebih.

Pada Algoritma EAFEC ketika sebuah blok pertama paket diterima, maka AP akan menghitung nilai *qweight*, kemudian membandingkan antrian yang terjadi dengan nilai ambang batas (*threshold*) Jika nilai panjang antrian lebih kecil dari nilai batas bawah (*low_threshold*l) maka Akan diset nilai maksimum FEC yang dihasilkan. Dan jika nilainya lebih besar dari batas ambang atas (*high_threshold*2) tidak ada paket FEC yang dihasilkan. Paket FEC dihitung kembali berdasarkan fraksi ukuran data dalam antrian. Jumlah paket FEC tersebut kemudian dihiutng kembali sesuai dengan waktu retransmisi ulang packet. Jika waktu retransmisi lebih kecil dari nilai *low_threshold*3, jumlah paket akan diset ke nilai 0 (nol). Dan jika waktu retransmisi ulang lebih besar dari nilai *high thrshold*4 jumlah packet FEC tidak berubah.

Skenario inilah yang ada pada skenario ke2 (dengan penambahan algoritma EAFEC. Dalam simulasi ini digunakan *Network simulator* NS-2 dengan tambahan modul evalvid untuk evaluasi kualitas video serta beberapa modifikasi lainya.

3.2 Format Parameter yang digunakan pada *Streaming video*.

Menggunakan *wireless Link* antara *Access Point* dengan video *receiver*.

- NS-2 *simulator*
- Video
  - *Codec*: JM 1.7 (H.264)
  - Video Test: *high*way (QCIF format)
  - Jumlah *Frame* : 2000
  - *GOP pattern*: IPPPPPPPPPPPPP
  - Tiap *Frame* dipecah menjadi *slice* kecil (~500 byte) dan ditransmisikan via RTP/UDP/IP packet (*unicast*).
  - Total paket dikirimkan: 4829 paket

- Tiga jenis *background traffic*
  - FTP *traffic* (over TCP)
  - *Exponential traffic* (over UDP)
  - IP

3.3 *Traffic Generator*

Aplikasi dalam simulasi ini merupakan layanan *streaming video*, yang menggunakan *Protokol* utama UDP karena menyediakan penyampaian paket tepat pada waktunya. Namun demikian, UDP tidak menjamin sampainya paket dengan baik. *Protokol transport* RTP berjalan diatas UDP, yang melakukan paketisasi dan menyediakan penyampaian *Frame* video berurut. RTCP digunakan oleh video klien untuk memberitahukan video server mengenai kualitas video yang diterima.

Penerapan aplikasi *video streaming* ini pada simulasi NS-2 membutuhkan pembangunan, konfigurasi *agent,* dan proses *attach* pada sebuah *source* data level aplikasi yang juga dinamakan *traffic Generator* [10] . *Traffic Generator* akan dibangun sesuai dengan karakteristik dari layanan *video streaming*. Setelah itu, simulasi akan menjalankan *agent* dan *traffic Generator*.

Tipe *agent* [10] simulasi yang ditambahkan pada *traffic Generator* adalah *MyTrafficTrace, MyUDP*, dan *MyUDPSink*. *Agent-agent* ini didesain baik untuk membaca *file trace* video atau untuk men-*generate* data yang dibutuhkan untuk mengevaluasi kualitas video yang dikirimkan. Cara kerja dan kegunaan masing-masing *agent* adalah sebagai berikut:

- ***Agent MyTrafficTrace*** meng-*extract* tipe *Frame* dan ukuran *Frame* dari *file trace* video yang di-*generate* oleh *file trace traffic*, memfragmentasi *Frame* video pada segmen yang lebih kecil, dan mengirim segmen-segmen ini pada layer UDP yang lebih rendah pada waktu yang baik sesuai dengan konfigurasi *user* yang ditentukan pada *file script* simulasi.
- ***Agent MyUDP*** adalah extension dari *agent* UDP. *Agent* ini mengizinkan *user* untuk menentukan *file* name output dari *file trace* pengirim dan merekam *Timestamp* pada setiap paket yang ditransmisikan, *packet ID*, dan *packet payload size*. Tugas dari *agent* MyUDP serupa dengan tugas

beberapa *tool* seperti *tcp-dump* atau *win-dump* pada environment *real network*.

- ***Agent MyUDPSink*** adalah *agent* penerima untuk paket *Frame* video yang terfragmen yang dikirimkan oleh MyUDP. *Agent* ini juga merekam *Timestamp*, *packet ID*, dan *payload size* dari masing-masing paket yang diterima pada *file trace* penerima *user* yang telah ditentukan. Setelah simulasi, berdasarkan *file trace* dan video asli yang di-*encode* ini, program *errinsert* memproduksi *file* video yang *corrupt*. Setelah itu, video *corrupt* di-*decode* dan *error* disembunyikan. Akhirnya, video *error* YUV yang terekonstruksi dapat dibandingkan dengan video *raw* YUV untuk mengevaluasi kualitas video *end-to-end* yang dikirimkan.

3.4 Pengolahan Video Streaming

Awal dari pengelolaan *streaming video* yaitu mengolah *raw* video melalui proses en*coding* untuk mendapatkan data yang dibutuhkan untuk kemudian dimasukkan ke dalam simulasi NS-2. Setelah itu, hasil yang diperoleh dari simulasi di-*decode* kembali dan dijalankan program untuk memperbaiki hasil video pada sisi penerima. Selama proses *encoding* maupun *decoding*, data yang diproses pada arus transmisi video akan ditandai dan disimpan menjadi *file-file* tertentu sebagai output atau rekaman proses yang terjadi.

3.5. Karakteristik Video (*raw video*)

File source raw video yang digunakan mempunyai format YUV, format ini yang paling banyak digunakan dan *matching* dengan banyak *tool encoder* dan *decoder*. yang merupakan format input yang diterima pada banyak *video encoder*. *File raw video* ini kemudian di-*encode* dan setelah itu *file trace* video diproduksi. *File trace* ini berisi seluruh informasi yang relevan pada modul untuk memperoleh hasil yang diinginkan sesuai dengan parameter QoS. *File raw* video ini akan dibuat melalui kompresi *encode* pada 30 fps selama 67 detik dan ukuran *Frame* adalah 176x144 pixel, yang juga dikenal sebagai *Quarter Common Intermediate Format (QCIF)*. QCIF digunakan karena merupakan format umum untuk *video streaming*.

3.6 *Video Sender (VS)*

Parser video H.264 dikembangkan berdasarkan standar video H.264. *Parser* ini digunakan untuk membaca H.264 yang diproduksi oleh *encoder* yang telah ditetapkan. Tujuan VS adalah untuk men-*generate file trace* dari *file* video yang telah di-*encode*. *File* output yang diproduksi oleh VS adalah 2 *file trace*, yaitu *file trace* pengirim dan *file trace* video.

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, VS merekam *Timestamp*, *packet ID*, dan *packet payload size* pada *file trace* pengirim dengan bantuan *third-party tool* .

3.7 *File* Output Simulasi

Setelah dilakukan simulasi jaringan berdasarkan *file trace* pengirim, NS-2 akan men-*generate* sebuah *file trace* utama, *file trace* video sender, *file trace* video *receiver*, dan *file* video. *File trace* utama merupakan pencatatan seluruh *event* (kejadian) yang dialami oleh suatu simulasi paket pada simulasi yang dibangun. Hasil-hasil *file* inilah yang dianalisa yang kemudian diterjemahkan dalam tampilan bentuk grafik.

3.8 Parameter Kinerja

Parameter yang diukur pada penelitian ini antara lain:

- **Analisa *Throughput***

  Throughput [7] merupakan laju data aktual yang terukur pada suatu ukuran dalam waktu tertentu. Walaupun *throughput* memiliki satuan dan rumus yang sama dengan laju data, tetapi *throughput* lebih pada menggambarkan laju data yang sebenarnya (aktual) pada suatu waktu tertentu dan pada kondisi dan jaringan internet tertentu yang digunakan untuk men-*download* suatu *file* atau data dengan ukuran tertentu.

Perhitungan *throughput* yang digunakan pada proyek akhir ini adalah jumlah paket data yang di terima oleh mobile station dibagi dengan waktu pengiriman data ke *mobile station* yang dituju. Dengan satuanya kbps atau Mbps.

$$Throughput = \frac{UkuranDataYangDiterima}{TotalWaktuPengirimanData}$$

- **Analisa *Packet Loss***

  Pada konteks *recovery* jaringan *wireless*, packet *loss* dapat diketahui dengan menghitung selisih jumlah paket yang dikirim dengan jumlah paket yang diterima.

  Pada perhitungan *packet loss* seperti yang dijelaskan di atas akan menghasilkan jumlah *packet loss* yang merupakan representasi jumlah paket yang terbuang pada *node* atau tidak sampai pada *node destination*.

- **Analisa Delay**

  Pada saat komunikasi akan terdapat *delay*, dalam hal ini adalah *delay* transmisi. Suatu paket data dari *source* menuju ke *destination* akan memerlukan *delay* waktu yang berbeda – beda tergantung parameter – parameternya antara lain jarak *mobile station* dengan *mobile station* yang lain, pergerakan *mobile station* dan masih banyak lagi yang dapat menyebabkan *delay* tersebut.

  Delay Time merupakan waktu yang diperlukan oleh suatu *node* untuk mengirimkan data ke *node* yang lain pada saat simulasi berlangsung.

- ***Jitter***

  ukuran *delay* penerimaan paket yang melambangkan *smoothness* dari *audio/video playback*.

3.9 Spesifikasi Perangkat

Dibawah ini spesifikasi perangkat yang akan digunakan dibutuhkan dalam membangun simulasi jaringan ini:

1. ***Interface Hardware***

   Hardware yang digunakan adalah 1 buah PC (*laptop*) dengan spesifikasi sebagai berikut:

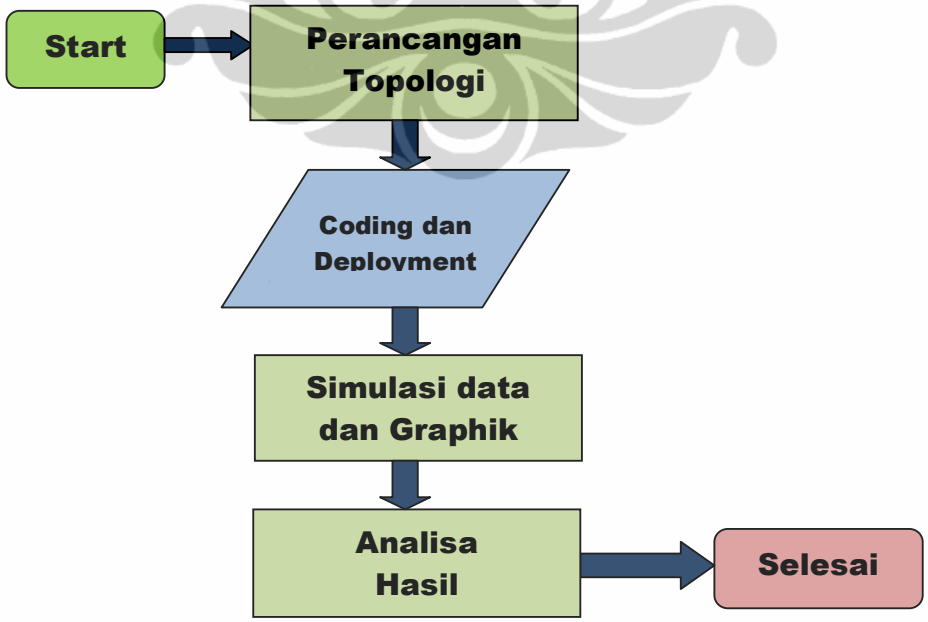
   - Acer TravelMate 6291
   - Processor: Intel Core 2 Duo 1.66 GHz
   - Memory: 2 Gb DDR2
   - HDD: 80 GB

2. ***Interface Software***

   Simulasi ini harus berjalan baik pada spesifikasi *software* sebagai berikut:

   - *Operating system*: Microsoft Windows Xp SP2
   - CYGWIN versi 1.5.25-15
   - *Simulation Application*: NS-2.28. Evalvid
   - *Text Editor*: Notepad++
   - *Grafik* : Microsoft Exel 2007
   - *Tool* s lain : *YUV viewer, converter yuv to avi.*
   - JM 1.7 *Decoder Encoder*

3.10 Tahap Implementasi



Gambar 3.3 Tahapan Kerja Implementasi Penelitian

Perencanaan implementasi penelitian dibagi menjadi 4 tahap utama seperti pada Gambar 3.3 diatas, yaitu tahap perancangan topologi, tahap *coding* dan *deployment*, tahap simulasi data dan grafik, dan tahap analisa hasil. Berikut adalah penjabaran lebih lanjut mengenai tahap-tahap implementasi:

1. **Perancangan Topologi**

   Perncangan topologi jaringan yang bersumber studi literatur merupakan langkah awal yang ditempuh. Topologi jaringan ini berupa jumlah *node* yang dibutuhkan, letak *node*, besarnya *bandwidth*, , batasan paket data, *transport agent* yang digunakan dan *traffik* yang akan dibangkitkan. Dengan adanya perancangan topologi ini akan membantu proses pembangunan simulasi jaringan secara keseluruhan.

2. ***Coding* dan *Deployment***

   Tahap ini merupakan tahapan inti dari implementasi ini, yaitu *coding* dan pemrograman bahasa Tcl pada aplikasi NS-2. Deskripsi dari pemrograman ini yaitu layanan *streaming video* akan diletakkan pada ujung *node* yang berada pada *node errored* (tidak bergerak) di jaringan IP, dimana *user* yang akan mencoba mengakses layanan yang disediakan di *node* sumber.

3. **Simulasi Data dan Grafik**

   Pada tahap ini simulasi akan dijalankan di atas aplikasi NS-2. Hasil simulasi adalah berupa *file trace*, yang kemudian akan ditampilkan ke dalam bentuk grafik dengan bantuan Microsoft Exel 2007.

4. **Analisa Hasil**

   Dalam tahap ini dilakukan analisa hasil berdasarkan data dan grafik yang diperoleh. Analisa ini meliputi nilai dari parameter-parameter yang ingin dicari yaitu *throughput, delay, jiter, PSNR.* Nilai parameter ini akan diperoleh dengan menggunakan *script AWK.*

3.11 Instalasi dan Implementasi

3.11.1 Instalasi *Software* NS-2

Simulasi NS-2 berjalan diatas *operating system WINDOWS XP SP2*. NS yang digunakan sendiri yaitu NS-2.28. Proses penginstalan pertama dengan men-

download re*source file* ns-allinone-2.28.tar.gz dari website www.isi.edu . *File* ini sudah mencakup beberapa paket *dependancy* yang dibutuhkan oleh NS-2, yaitu xlibs-dev, tcl, tk, otcl, nam, dan xgraph.

```
# tar xvzf ns-allinone-2.28.tar.gz
# cd ns-alllinone-2.28
# ./install
```

Waktu yang dibutuhkan kurang lebih 20 menit, tergantung dari spesifikasi komputer yang digunakan. Setelah instalasi selesai, update *environment* variabel pada '*/etc/bashrc*' sebagai berikut:

```
# LD_LIBRARY_PATH
OTCL_LIB=/usr/local/ns/otcl-1.12
NS2_LIB=/usr/local/ns/lib
X11_LIB=/usr/X11R6/lib
USR_LOCAL_LIB=/usr/local/lib
export
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$OTCL_LIB:$NS2_LIB:$X11_LIB:$USR_LOCAL_LIB

# TCL_LIBRARY
TCL_LIB=/usr/local/ns/tcl8.4.13/library
USR_LIB=/usr/lib
export TCL_LIBRARY=$TCL_LIB:$USR_LIB

# PATH
XGRAPH=/usr/local/ns/bin:/usr/local/ns/tcl8.4.13/unix:/usr/local/ns/tk8.4.13/unix
NS=/usr/local/ns/ns-2.28/
NAM=/usr/local/ns/nam-1.12/
PATH=$PATH:$XGRAPH:$NS:$NAM
```

3.11.2 Instalasi EvalVid

NS-2 secara default tidak memiliki modul untuk *agent traffic Generator Codec* H.264. Oleh karena itu dibutuhkan *tool* khusus EvalVid berupa *file* kompresi 'evalvid-2.4.tar.bz2' yang dapat diperoleh dari website http://www.tkn.tu-berlin.de/research/evalvid/ serta penambahan modul EvalVid pada NS-2 melalui modifikasi *packet, agent* dan *traffic agent*. Modifikasi ini dilakukan dengan beberapa langkah manual sebagai berikut

```
Modifikasi packet.h
Modifikasi agent.h
Modifikasi agent.cc
Penambahan file myudp.cc, myudp.h, myudpsink3.cc, myudpsink3.h, mytraffictrace3.cc
Modifikasi ns-default.tcl
Modifikasi Makefile
./configure && make clean && make
```

3.11.3 *Codec File* Video

Proses evaluasi dilakukan dengan mengkompresi *raw video* dengan *encode* pada 30 fps,. Ukuran *Frame* adalah 176x144 pixel, yang juga dikenal sebagai *Quarter Common Intermediate Format (QCIF)*. QCIF digunakan karena merupakan format umum untuk *video streaming* pada jaringan *mobile*. Untuk itu dibutuhkan *file* '*encoder*.cfg' yang berfungsi sebagai konfigurasi khusus *Codec* H.264.

```
# ./lencod encoder.cfg
```

Beberapa metode pengujian kualitas video dibutuhkan video referensi. Hal ini baik *file* YUV asli sebelum *encoding* atau *file* YUV yang dibuat oleh *decoding* video yang di-*code*. Dibutuhkan atau tidaknya YUV yang di-*decode* bergantung pada apa yang ingin diperoleh. Jika ingin menguji kualitas video pada sistem jaringan dan tidak hanya kualitas *encoder* harus dibuat *file* YUV *decoded*. Untuk memproduksi *file-file* YUV ini harus digunakan *video decoder* open *source*.

```
# ./ldecod decoder_distorted.cfg
```

3.11.4 Menjalankan *Script* Tcl

Script Tcl yang dibangun disimpan dengan nama *file* '11.tcl', dan berikut adalah *command* untuk menjalankannya.

```
# ./ns 11.tcl seed pGG pBB pG  pB Loss_Model
```

NS-2 akan menghasilkan output *trace* umum yaitu 'out.tr' sesuai dengan *event-event* yang dilakukan oleh simulasi. *File-file* ini kemudian di-*compile* dan dianalisa lebih lanjut untuk memperoleh hasil akhir parameter QoS yang diinginkan.

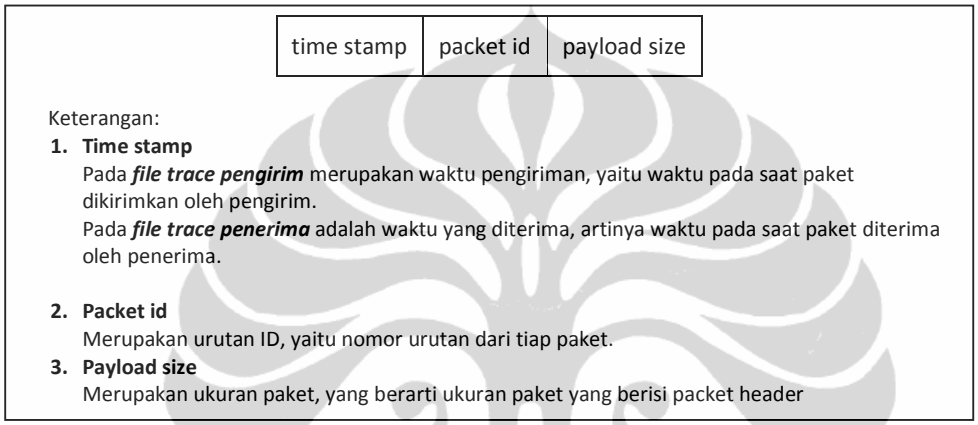
3.11.5 Menjalankan *Script* Awk [3]

Satu-satunya parameter QoS yang tidak dihasilkan langsung oleh *tool -tool* EvalVid pada NS-2, yaitu *throughput* dilakukan dengan menggunakan *script Awk* yang dibuat oleh Marco Fiore [3]. Untuk perhitungan *throughput* ini dilakukan dengan perintah sebagai berikut:

```
# ./awk –f cal_th_traffic.awk rd_trafic > throughput.out
```

3.11.6 Evaluasi *File Trace*

Selain *file trace* utama, NS-2 juga menghasilkan *file trace* pengirim dan *file trace* penerima. Kedua *file trace* ini memiliki format yang sama seperti Gambar 3.4 dibawah ini. Agar hasil simulasi dapat dianalisa dan ditampilkan dalam bentuk grafik, maka dapat dilakukan *record* atau *parsing* terhadap *trace file* untuk mengambil data yang benar-benar diperlukan.



Gambar 3.4 Format *File Trace* Pengirim dan Penerima

Perhitungan *loss* sungguh mudah, mengingat ketersediaan *packet id* yang unik. Dengan bantuan *file trace* video, tiap paket ditetapkan sebuah tipe. Tiap paket pada tipe ini yang tidak termasuk pada *trace* penerima dihitung *loss*. *Frame loss* dihitung dengan melihat pada *Frame* manakah salah satu segmen (paket) hilang. Jika segmen pertama dari *Frame* adalah diantara segmen yang hilang, *Frame* dihitung *loss*. Ini dikarenakan video *decoder* tidak bisa men-*decode Frame*, dimana bagian awal hilang.

File trace penerima memiliki *Time-stamp* yang valid pada tiap paket, *inter-packet* (termasuk *inter-Frame*) *delay* dapat dihitung berdasarkan persamaan berikut.

Arrival *Time* (*packet loss*) $tR_n = tR_{n-1} + (tS_n - tS_{n-1})$ …….. (3.1)

dimana: tS_n : *Time-stamp* dari paket ke-n yang dikirim

tR_n : *Time-stamp* dari paket ke-n yang diterima atau tidak

Langkah dalam proses menghitung video PSNR referensi. PSNR referensi ini diambil dari video yang di-*code* dan di-*decode* (*Codec*) tanpa adanya *error* atau *loss* pada transmisi yang mengacu pada *raw video source* yang tidak mengalami proses *Codec*. Kemudian dilakukan perbandingan antara video referensi dengan video hasil output (kemungkinan *corrupt*).

```
# ./psnr 176 144 420 highway_qcif.yuv highway_qcif_distorted.yuv > psnr_perbandingan.txt
```

3.11.7 Grafik dan Analisa

Untuk mempermudah proses analisa, data perhitungan akhir yang diperoleh yaitu *throughput*, *packet loss*, *delay*, dan *jitter* divisualisasikan ke dalam bentuk grafik. Proses pengolahan data ke dalam bentuk grafik ini dengan cepat dibuat dengan menggunakan Microsoft Excel 2007.