BAB II
DASAR TEORI

2.1 Sejarah Perkembangan *Wireless LAN*

Pada akhir 1970-an IBM mengeluarkan hasil percobaan mereka dalam merancang WLAN dengan teknologi IR, perusahaan lain seperti *Hewlett-Packard* (HP) menguji WLAN dengan RF. Kedua perusahaan tersebut hanya mencapai *data rate* 100 Kbps. Karena tidak memenuhi standar IEEE 802 untuk LAN yaitu 1 Mbps maka produknya tidak dipasarkan. Baru pada tahun 1985, (FCC) menetapkan pita *Industrial, Scientific and Medical* (ISM band) yaitu 902-928 MHz, 2400-2483.5 MHz dan 5725-5850 MHz yang bersifat tidak terlisensi, sehingga pengembangan WLAN secara komersial memasuki tahapan serius. Barulah pada tahun 1990 WLAN dapat dipasarkan dengan produk yang menggunakan teknik *spread spectrum* (SS) pada pita ISM, frekuensi terlisensi 18-19 GHz dan teknologi IR dengan *data rate* $\geq$1 Mbps.

Pada tahun 1997, sebuah lembaga independen bernama IEEE membuat spesifikasi/standar WLAN pertama yang diberi kode 802.11. Peralatan yang sesuai standar 802.11 dapat bekerja pada frekuensi 2,4GHz, dan kecepatan transfer data (*throughput*) teoritis maksimal 2Mbps.

Pada bulan Juli 1999, IEEE kembali mengeluarkan spesifikasi baru bernama 802.11b. Kecepatan transfer data teoritis maksimal yang dapat dicapai adalah 11 Mbps.

Pada saat hampir bersamaan, IEEE membuat spesifikasi 802.11a yang menggunakan teknik berbeda. Frekuensi yang digunakan 5Ghz, dan mendukung kecepatan transfer data teoritis maksimal sampai 54Mbps.

Pada tahun 2002, IEEE membuat spesifikasi baru yang dapat menggabungkan kelebihan 802.11b dan 802.11a. Spesifikasi yang diberi kode

802.11g ini bekerja pada frekuensi 2,4Ghz dengan kecepatan transfer data teoritis maksimal 54Mbps.

Pada tahun 2006, 802.11n dikembangkan dengan menggabungkan teknologi 802.11b, 802.11g. Teknologi yang diusung dikenal dengan istilah MIMO (*Multiple Input Multiple Output*) merupakan teknologi *WiFi* terbaru. MIMO dibuat berdasarkan spesifikasi Pre-802.11n. Kata "Pre-" menyatakan "*Prestandard versions of 802.11n*". MIMO menawarkan peningkatan *throughput*, keunggulan reabilitas, dan peningkatan jumlah klien yg terkoneksi. Daya tembus MIMO terhadap penghalang lebih baik, selain itu jangkauannya lebih luas sehingga Anda dapat menempatkan laptop atau klien *WiFi* sesuka hati.

Akses Point MIMO dapat menjangkau berbagai perlatan *WiFi* yg ada disetiap sudut ruangan. Secara teknis MIMO lebih unggul dibandingkan saudara tuanya 802.11a/b/g. *Access Point* MIMO dapat mengenali gelombang radio yang dipancarkan oleh adapter *WiFi* 802.11a/b/g. MIMO mendukung kompatibilitas mundur dengan 802.11 a/b/g. Peralatan *WiFi* MIMO dapat menghasilkan kecepatan transfer data sebesar 108Mbps.

2.1.1. WiFi (*Wireless* Fidelity)

Istilah *WiFi* diciptakan oleh sebuah organisasi bernama *WiFi* alliance yang bekerja menguji dan memberikan sertifikasi untuk perangkat-perangkat WLAN. Teknologi WLAN (menggunakan standar radio 802.11 yang sekarang umum disebut dengan WiFi) telah menjadi teknologi inventori yang handal. Sekarang kondisinya meluas. Perangkat *wireless* diuji berdasaarkan interoperabilitasnya dengan perangkat-perangkat *wireless* lain yang menggunakan standar yang sama. Setelah diuji dan lulus, sebuah perangkat akan diberi sertifikasi "*WiFi certified*". Artinya perangkat ini bisa bekerja dengan baik dengan perangkat-perangkat *wireless* lain yang juga bersertifikasi ini. *WiFi* sudah banyak digunakan di berbagai sektor seperti bisnis, akademis, perumahan, dan banyak lagi. Teknologi WiFi ini dapat juga digunakan untuk kegiatan memindahkan inventori secara cepat, memobilisasi para *floor manager* dan meningkatkan kepuasaan pelanggan.

2.1.2. Standarisasi WiFi 802.11a

Standard 802.11a, adalah *model* awal yang dibuat untuk umum. Mengunakan kecepatan 54Mbps dan dapat mentranfer data double dari tipe g dengan kemampuan *bandwidth* 72Mbps atau 108Mbps. Sayangnya sistem ini tidak terlalu standar, karena masing masing *vendor* atau pabrikan memberikan standar tersendiri. 802.11a mengunakan frekuensi tinggi pada 5Ghz sebenarnya sangat baik untuk kemampuan tranfer data besar. Tetapi 802.11a memiliki kendala pada harga , komponen lebih mahal ketika perangkat ini dibuat untuk publik dan jaraknya dengan frekuensi 5GHz konon lebih sulit menembus ruang untuk kantor.

Pemilihan 5Ghz cukup beralasan, karena membuat pancaran signal frekuensi 802.11a jauh dari gangguan seperti oven microwave atau cordless phone pada 2GHz, tetapi frekuensi tinggi juga memberikan dampak pada daya jangkau relatif lebih pendek 802.11b Sempat menjadi dominasi pemakaian tipe b. Standard 802.11b mengunakan frekuensi 2.4GHz. Standard ini sempat diterima oleh pemakai didunia dan masih bertahan sampai saat ini. Tetapi sistem b bekerja pada band yang cukup kacau, seperti gangguan pada *Cordless* dan frekuensi *Microwave* dapat saling menganggu bagi daya jangkaunya.

Standard 802.11b hanya memiliki kemampuan tranmisi standard dengan 11Mbps atau rata rata 5MBbit/s yang dirasakan lambat, mendouble (turbo mode) kemampuan *wireless* selain lebih mahal tetapi tetap tidak mampu menandingi kemampuan tipe a dan g. 802.11g Standard yang cukup kompatibel dengan tipe 802.11b dan memiliki kombinasi kemampuan tipe a dan b. Mengunakan frekuensi 2.4GHz mampu mentransmisi 54Mbps bahkan dapat mencapai 108Mbps bila terdapat inisial G atau turbo. Untuk hardware pendukung, 802.11g paling banyak dibuat oleh *vendor*. Secara teoritis mampu mentranfer data kurang lebih 20Mbit/s atau 4 kali lebih baik dari tipe b dan sedikit lebih lambat dari tipe a

2.1.3. *Access Point*

Access Point atau yang lebih sering disebut dengan istilah AP merupakan sebuah perangkat penghubung antara jaringan *wire* dengan *wireless*. Maksudnya sebuah AP akan bertugas mengubah data yang lalu lalang di media kabel menjadi sinyal-sinyal radio yang dapat ditangkap oleh perangkat *wireless*. AP akan menjadi gerbang bagi jaringan *wireless* untuk dapat berkomunikasi dengan dunia luar maupun dengan antarsesama perangkat *wireless* di dalamnya. Biasanya pada perangkat AP terdapat satu atau lebih *interface* untuk media kabel. *Interface* media kabel tadi akan di-*bridging* oleh AP tersebut ke dalam bentuk sinyal-sinyal radio, sehingga perangkat *wireless* dengan kabel tadi dapat terkoneksi.

Access Point sangat dibutuhkan jika ingin membuat sebuah infrastruktur jaringan *wireless*. Dengan menggunakan AP, maka sebuah jaringan komunikasi akan terbentuk tidak hanya dua atau tiga perangkat saja yang dapat berkomunikasi tetapi cukup banyak yang dapat saling berbicara dengan perantara sinyal radio ini.

Pengaplikasian AP yang banyak dilakukan saat ini adalah melakukan pembagian *bandwidth* Internet dari link Internet ADSL atau Kabel, sehingga dapat digunakan oleh banyak orang. Namun jika ingin membangun koneksi hanya dengan sebuah perangkat *wireless* lainnya, AP tidaklah mutlak diperlukan. Untuk dapat mengoperasikan perangkat *wireless* dalam mode *peer-to-peer* atau yang lebih dikenal dengan istilah mode Ad-Hoc. Tetapi, kekurangan dari komunikasi mode Ad-Hoc ini adalah tidak dapat membangun jaringan *wireless* yang luas karena memang sifatnya yang *Point-to-Point*.

Sistem WLAN, terlepas dari keterbatasan perangkat AP, dapat melayani pengguna dalam jumlah yang tidak terbatas. Para penggunanya dapat menambahkan AP baru jika memang jumlah pengguna yang akan dilayaninya semakin membengkak. Dengan memasang banyak AP, maka banyak sekali keuntungan yang didapat. Hal ini memanjakan pengguna jaringan *wireless* dengan *bandwidth* yang lega, pengguna juga dapat bebas berkeliaran di manapun mereka suka karena area *coverage*-nya sudah pasti lebih luas, dan jumlah pengguna yang dapat dilayani oleh jaringan ini juga lebih banyak. Jadi sebenarnya sistem WLAN

tidak pernah memberikan batasan berapa banyak yang dapat terkoneksi ke sebuah jaringan *wireless*. Semua tergantung pada kemampuan dan fasilitas perangkatnya.

2.1.4. Sistem Keamanan WLAN

Untuk itu, ada beberapa teknik yang dapat digunakan untuk lebih mempersulit para pengganggu untuk mengacau jaringan *wireless*. Metode tersebut adalah WEP, WPA, dan 802.1x. • WEP, Teknik pengaman jaringan *wireless* yang satu ini merupakan kepanjangan dari *Wired Equivalent Privacy*. WEP menggunakan sistem enkripsi untuk memproteksi pengguna WLAN dalam level yang paling dasar. WEP memungkinkan administrator jaringan *wireless* membuat encription *key* yang akan digunakan untuk mengenkripsi data sebelum dikirimkan melalui jalan udara. Encription *key* ini biasanya dibuat dari 64 bit *key* awal dan dipadukan dengan algoritma enkripsi RC4. Ketika fasilitas WEP diaktifkan, maka semua perangkat *wireless* (AP dan *client*) yang ada di jaringan harus dikonfigurasi dengan menggunakan *key* yang sama.

Hak akses dari seseorang atau sebuah perangkat akan ditolak jika *key* yang dimasukkan tidak sama. • *WIFI Protected Access* atau disingkat dengan istilah WPA, merupakan teknik pengaman jaringan *wireless* LAN yang diklaim lebih canggih dari WEP. Dengan disertai teknik enkripsi yang lebih *advanced* dan tambahan pengaman berupa otentikasi dari penggunanya, maka WPA akan jauh lebih hebat mengamankan pengguna WLAN. • 802.1x, Teknik pengaman yang satu ini akan mengharuskan semua pengguna jaringan *wireless* untuk melakukan proses otentikasi terlebih dahulu sebelum dapat bergabung dalam jaringan. Sistem otentikasinya dapat dilakukan dengan banyak cara, namun sistem otentikasi menggunakan pertukaran *key* secara dinamis. Sistem pertukaran *key* secara dinamis ini dapat dibuat dengan menggunakan *Extensible Authentication Protocol* (EAP).

2.2 Streaming Video

Teknologi yang mampu mengirimkan *file* audio dan video digital secara *real time* dikenal dengan nama *Video streaming* dimana diperbolehkanya

pengolahan *steady* secara terus-menerus oleh *end-user*. *tool* yang mendukung *video streaming* 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 ada pada *video streaming* [13] sebagai berikut:

- *User Datagram Protocol (UDP),* mudah diimplementasikan serta efisien tetapi dapat menimbulkan banyak data yang hilang (data *loss*)
- *Transmission Control Protocol (TCP),* pengiriman yang cepat dan tepat namun membutuhkan *buffer* yang tinggi.menjamin penyampaian yang tepat namun dapat menuju timeout sehingga klien membutuhkan *buffer* yang cukup.
- *Real-time Streaming Protocol (RTSP)*, berfungsi sebagai *control remote* seperti '*play' 'pause' 'stop'*.
- *Real-time Transport Protocol (RTP)*, merupakan standarisasi format paket 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*.

2.3 H.264

H.264 [11] atau *MPEG-4 AVC (Advanced Video Coding)* merupakan sebuah standar terbaru untuk *video coding* yang dikembangkan oleh *Joint Video Team* dari MPEG dan ITU-T. standar ini berbasis pada standar MPEG yang ada sebelumnya dan merupakan penyempurnaan, seperti penyempurnaan pada efisiensi *bit rate* hingga 40% dibandingkan *codec* MPEG-4 sebelumnya dengan kualitas yang sama, *frame size* 4 kali lebih besar dibandingkan dengan MPEG-4 Part-2 pada *data rate* yang sama, dan memiliki efisiensi kompresi yang sangat baik. Pada Teknologi H.264 mengantarkan kualitas yang sangat baik dan luar biasa dan melalui *data rate* rendah yang mengesankan, baik dalam membuat video untuk *mobile phone, iChat AV, internet, broadcast*, atau transmisi satelit.

2.3.1 Data Video pada H.264

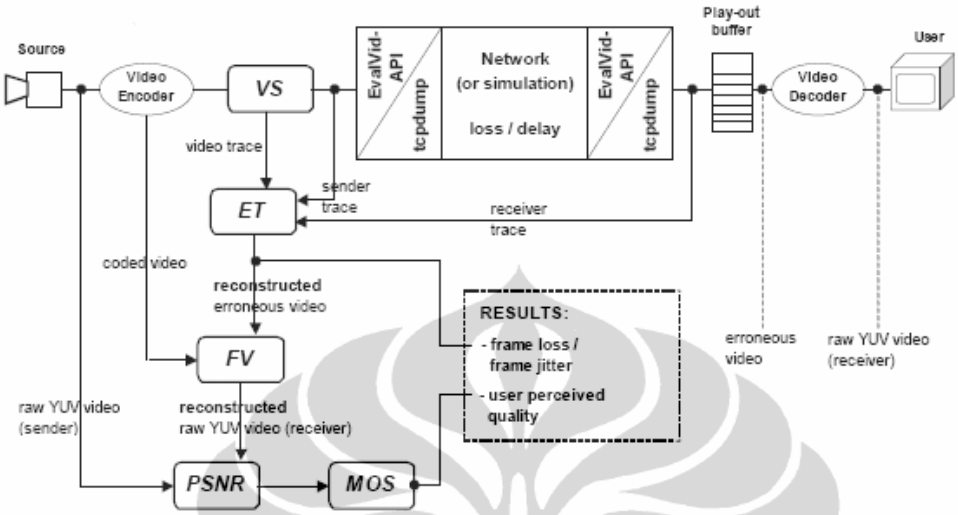
Video hanyalah sebuah dereten gambar-gambar yang disebut *Frame. S*ebuah *Encoder H.264* [12] akan menggabungkan deretan *frame* ini dalam sebuah deretan terpisah terlebih dahulu yang disebut *Group of Pictures (GOP)*. Sebuah GOP biasanya terdiri atas 12 sampai 15 *frame*. *Encoder* H.264 membagi-bagi 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.

2.4 EvalVid

Evaluation Video (EvalVid) [11] 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 *subjektif* 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 2.1 dibawah ini 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*, *delay* dan kualitas video.



Gambar 2.1 [11] Skema *framework* evaluasi pada Evalvid

2.5 Wireless Error Model

Banyak *error model* yang digunakan oleh para periset *wireless* memakai *model random uniform*. Padahal normalnya pada *wireless error* nya adalah *burst pattern*. Gilbert-Elliot [5] adalah salah satu *model error pattern* yang lebih menyerupai pada *wirelees pattern*. Pada gambar dibawah ini dapat kita lihat pola-pola dari Gilbert-Elliot *Model*.

Dalam state (G) "*good*" terjadi *loss* dengan mengurangi probabilitas pG selagi dalam state "*bad*" (B) terjadi kemungkian yang tinggi pB. Juga pGB adalah probabilitas dari state transisi antar "*good*" state ke "*bad*" state dan pBG adalah transisi dari sebuah state "*bad*" ke state "*good*"

$$\pi_G = \frac{p_{BG}}{p_{BG} + p_{GB}} \qquad \pi_B = \frac{p_{GB}}{p_{BG} + p_{GB}} \quad \text{..... (2.1)}$$

Rata-rata *packet loss rate* yang dihasilkan GE *error model* adalah

$$p_{avg} = p_G \pi_G + p_B \pi_B \quad \text{..... (2.1)}$$

Pada jaringan *wireless* tak ada retrasnmisi pada *broadcasting* dan *multicasting*, jadi paket *error rate* dari *network-level* nya sama dengan pada *application-level*. Bagaimanapun dalam *unicasting MAC senders* dapat mengirimkan sebuah paket pada maksimum dari N kalo sebelum paket tersebut dibuang. Dengan demikian *correct rate* pada *application-level* adalah

$$p_{CORECT} = \sum_{i=1}^{N} (1-p)p^{i-1} = 1 - p^{N} \quad \text{..... (2.2)}$$

Dimana N adalah maksimum jumlah retransmisi pada *MAC Layer* dan p adalah *packet loss rate* dari *network–level*. Sehingga *error rate* pada aplikasi level adalah

$$p_{\text{effective}} = p^{N} \quad \text{..... (2.3)}$$

Pada simulasi yang dibuat, parameter yang diset untuk *packet error rate* nya berdasarkan karakteristik dari *error model* yang digunakan.

2.6 Parameter Kinerja

Penelitian ini mengukur kualitas dari performa layanan *video streaming*. Oleh karena itu dibutuhkan parameter-parameter QoS [2] dan parameter-parameter kualitas video yang mendukung.

2.6.1 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)*. 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 dalam satuan persen.

$$PL_T = 100\ \frac{n\tau_{err}}{n\tau_{sent}} \qquad \text{..... (2.4)}$$

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

nT_{sent} : jumlah tipe paket T yang terkirim

nT_{err}: jumlah tipe paket T yang hilang

2.6.2 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*. 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 st*rate*gi 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 2.6, 2.7, dan 2.8. 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

$$it_{Po} = 0$$
$$it_{Po} = t_{Pn} - t_{Pn-1} \quad \textbf{(2.5)}$$

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

inter-frame time

$$it_{Fo} = 0$$
$$it_{Fm} = t_{Fm} - t_{Fm-1} \quad \textbf{(2.6)}$$

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

packet jitter

$$j_P = \frac{1}{N}\sum_{i=0}^{N}(it_i - \bar{it}_N)^2 \quad \textbf{(2.7)}$$

dimana: N : paket N

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

frame jitter

$$j_F = \frac{1}{n}\sum_{i=1}^{n}(it_i - \bar{it}_M)^2 \quad \textbf{(2.8)}$$

dimana: N : *frame* N

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

2.6.3 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 *Subjektif*

  selalu merenggut faktor krusial, kesan *user* melihat video ketika sedang berhemat, menghabiskan banyak waktu, kebutuhan sumberdaya manusia yang tinggi dan perlengkapan khusus yang dibutuhkan.

❖ Pengukuran Kualitas *objektif*

secara detail pada ITU, ANSI, dan MPEG. *Human quality impression* biasanya diberikan pada skala dari 5 (terbaik) ke 1 (terburuk) dinamakan *Mean Opinion Score (MOS).*

2.6.4 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 *subjektif* dengan SNR yang konvensional. Persamaan 2.9 adalah definisi dari PSNR antara komponen *luminance* Y dari gambar sumber S dan gambar tujuan D.

$$PSNR(n)_{dB} = 20\,log_{10}\left(\frac{V_{peak}}{\sqrt{\frac{1}{N_{col}N_{row}}\sum_{i=0}^{N_{col}}\sum_{i=0}^{N_{row}}[Y_S(n,i,j)-Y_D(n,i,j)]^2}}\right) \quad \text{..... (2.9)}$$

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 dikirimkan yang tidak terdistorsi. Hal ini dapat dilakukan dengan membandingkan PSNR dari *encoced video* dengan *frame* video yang diterima *frame by frame* atau membandingkan rata-rata mereka dan standar deviasi.

2.7 *NETWORK SIMULATOR*

2.7.1 Perkembangan Awal

Network simulator (NS) dibangun sebagai varian dari REAL *Network Simulator* pada tahun 1989 di UCB (University of California Berkeley). Dari awal tim ini dibangun sebuah perangkat lunak simulasi jaringan Internet untuk kepentingan riset interaksi antar protokol dalam konteks pengembangan protokol internet pada saat ini dan masa yang akan datang.

2.7.2 Kelebihan NS2

Kelebihan dari NS2 adalah sebagai perangkat lunak simulasi pembantu analisis dalam riset atau penelitian. NS2 dilengkapi dengan *tool* validasi. *Tool* validasi digunakan untuk menguji validitas pe*model*an yang ada pada NS2.

Pembuatan simulasi dengan menggunakan NS2 jauh lebih mudah daripada menggunakan *software developer* lainnya. Pada *software* NS2 ini *user* tinggal membuat topologi dan skenario simulasi yang sesuai dengan riset. Pe*model*an media, protokol dan net*work* component lengkap dengan perilaku trafiknya sudah tersedia pada *library* NS2.

NS2 bersifat open *source* di bawah GPL (Gnu Public License), sehingga NS2 dapat didownload melalui website NS2 http://www.isi.edu/nsnam/dist.

2.7.3 Simulasi yang menggunakan NS2

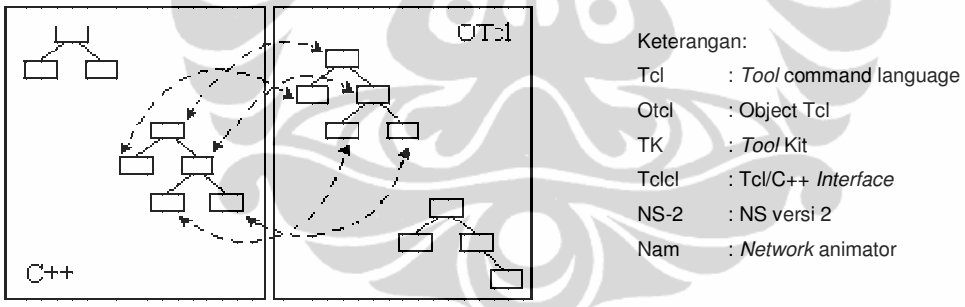
Network Simulator (NS2) mensimulasikan jaringan berbasis TCP/IP dengan berbagai macam medianya. Sehingga dapat mensimulasikan protokol jaringan (TCPs/UDP/RTP), *Traffic behaviour* (FTP, Telnet, CBR, dan lain - lain), *Queue management* (RED, FIFO, CBQ) *algoritma routing unicast* (*Distance Vector, Link State*) dan *multicast*, (PIM SM, PIM DM, DVMRP, *Shared Tree* dan *Bi directional Shared Tree*), aplikasi multimedia yang berupa *layered* video, *Quality of Service* video-audio dan transconding. NS2 juga mengimplementasikan beberapa MAC (IEEE 802.3, 802.11), di berbagai media misalnya jaringan wired (seperti LAN, WAN, *point to point*), *wireless* (seperti mobile IP, *Wireless* LAN), bahkan simulasi hubungan antar – *node* jaringan yang menggunakan media satelit.

2.7.4 Konsep Dasar NS2

Network Simulator merupakan salah satu perangkat lunak atau *software* yang dapat menampilkan secara simulasi proses komunikasi dan bagaimana proses komunikasi tersebut berlangsung. *Network Simulator* melayani simulasi untuk komunikasi dengan kabel dan komunikasi *wireless*.

Pada *Network Simulator* terdapat tampilan atau display baik dengan *node* yang bergerak atau *node* yang tidak bergerak. Yang tentunya tidak sama dengan keadaan yang sebenarnya.

Network Simulator dibangun dengan menggunakan 2 bahasa pemrograman, yaitu C++ da Tcl/Otcl. C++ digunakan untuk *library* yang berisi *event scheduler*, protokol dan *network component* yang diimplementasikan pada simulasi oleh *user*. Tcl/OTcl digunakan pada skript simulasi yang ditulis oleh NS *user* dan pada *library* sebagai *simulator* objek. OTcl juga nantinya berperan sebagai interpreter. Hubungan antarbahasa pemrograman dapat dideskripsikan seperti Gambar 2.2 dibawah ini.

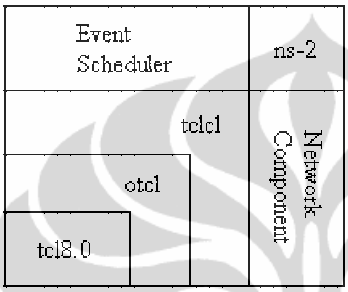


Gambar 2.2 Hubungan C++ dan Otcl (1)

Bahasa C++ digunakan pada *library* karena C++ mampu mendukung runtime simulasi yang cepat, meskipun simulasi melibatkan simulasi jumlah paket dan dan sumber data dalam jumlah besar.

Bahasa Tcl memberikan respon runtime yang lebih lambat daripada C++, namun jika terdapat kesalahan *syntax* dan perubahan skript berlangsung dengan cepat dan interaktif. *User* dapat mengetahui letak kesalahannnya yang dijelaskan pada konsole, sehingga *user* dapat memperbaiki dengan cepat. Karena alasan itulah bahasa ini dipilih untuk digunakan pada skrip simulasi.

Berdasarkan deskripsi pada Gambar 2.3 [1], 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.



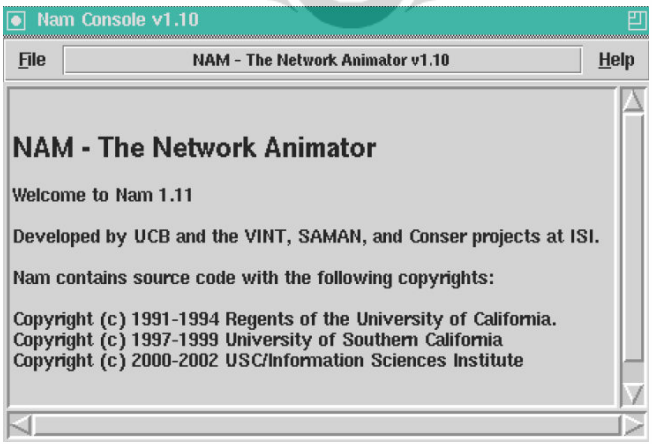
Gambar 2.3 Hubungan Antar-Komponen Pembangun NS-2

2.7.5 Output Simulasi NS2

Pada saat satu simulasi berakhir, NS membuat satu atau lebih *file output text-based* yang berisi detail simulasi jika dideklarasikan pada saat membangun simulasi. Ada dua jenis output NS, yaitu :

• *File trace*, yang akan digunakan untuk analisa numerik, dan

• *File namtrace*, yang digunakan sebagai input tampilan grafis simulasi seperti yang ada pada Gambar 2.4 yang disebut *network animator* (nam).



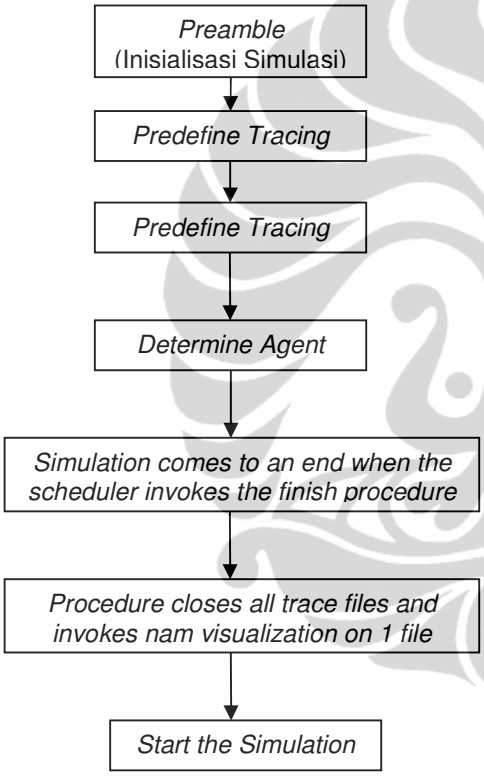
Gambar 2.4 Nam Konsole

2.7.6 Pengambilan Data Simulasi

Simulasi dibangun untuk mengambil data yang akan diolah pada analisis. Data hasil simulasi terdiri dari beberapa jenis yaitu:

1. *File Trace*

   File trace merupakan pencatatan seluruh *event* (kejadian) yang dialami oleh suatu simulasi paket pada simulasi yang dibangun. Pembuatan *file trace* dilakukan dengan memanggil obyek *trace* pada *library*.Sama seperti *file* nam, pembuatan output *trace file* dinyatakan pada inisialisasi simulasi.



Gambar 2.5 Flowchart pada 'ns filename.tcl'

Berikut ini merupakan contoh dari suatu *file trace*:

r 1.3556 3 2 ack 40.....1.3.4.0.0.0 15 201

+ 1.3556 2 0 ack 40.....1.3.4.0.0.0 15 201

- 1.3556 2 0 ack 40.....1.3.4.0.0.0 15 201

Pada Gambar 2.6 [1] mendeskripsikan format *file trace* yang disimulasikan oleh NS-2. 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.

Event	time	From *node*	To *node*	Pkt type	Pkt Size	*Flag*s	Fid	Src addr	Dst Addr	Seq Num	Pkt Id

Gambar 2.6 [1] Format Header *File Trace*

Penjelasan dari Gambar 2.6 adalah :

Event adalah Kejadian yang dicatat oleh NS yaitu

r : *receiver* (paket diterima oleh to *node*)

+ : enqueue (paket masuk ke dalam antrian atau keluar dari *form node*)

- : *dequeue* (paket keluar dari antrian)

d : *drop* (paket *drop* dari antrian)

Time adalah waktu terjadinya suatu kejadian dalam detik

From node dan *to node* menyatakan keberadaan paket. Saat pencatatan kejadian, paket berada pada link diantara from *node* dan to *node*.

Pkt type adalah tipe paket yang dikirim seperti udp, tcp, ack, atau cbr.

Pkt Size Adalah ukuran paket dalam *byte*.

Flag digunakan sebagai penanda. Pada data diatas *flag* tidak digunakan Macam-macam *Flag* yang bisa digunakan yaitu:

- E untuk terjad Kongesti (*Congstion Experienced/CE*)
- N untuk indikasi ECT (*ECN-Capable-transport*) pada *header* IP
- C untuk *ECN-Echo*
- A untuk pengurangan window kogesti pada *header* TCP
- P untuk prioritas
- F untuk TCP *fast start*

Fid Adalah penomoran unik dari tiap aliran data.

scr_addr adalah alamat asal paket, Format scr_addr adalah : *node*.port, misalnya 3.0.

dst_addr adalah alamat tujuan paket, Format dst_addr adalah *node*.port misalnya 0.0

sequence number adalah nomor urut tiap paket

paket Id adalah penomoran unik dari tiap paket

2. *Parsing*

Parsing adalah suatu teknik untuk mendapatkan informasi yang diinginkan dari *file trace* hasil simulasi. Untuk mendapatkan informasi tersebut diperlukan *file* 'awk' [3] . *File* 'awk' berfungsi untuk memfilter *trace file*, sehingga diperoleh informasi yang diinginkan. Perintah *parsing* ditulis pada prosedur *finish* dengan format sebagai berikut:

```
exec awk –f delay.awk out.tr > delay.tr
```

Delay.awk merupakan nama *file* awk, dimana perintah ini harus diletakkan dengan folder yang sama dengan skrip simulasinya. Jika *file* 'awk' diletakkan pada folder yang berbeda, harus disertakan path yang menuju *file parsing* berada, pada saat eksekusi *file*. 'Out.tr' adalah *file trace* total keluaran simulasi dan 'delay.tr' adalah *file* keluaran hasil *parsing*.

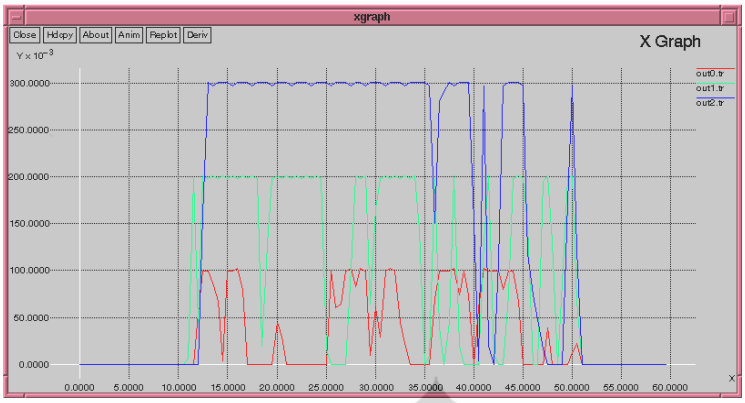
3. Xgraph

Xgraph adalah suatu cara untuk memplot output simulasi pada *network simulator* dalam bentuk grafik. Xgraph melakukan plot dengan membaca dua kolom data pada *file tr.* Kolom pertama akan diplot menjadi sumbu X dan kolom kedua di plot menjadi sumbu Y .Jadi jika ingin memplot hasil simulasi dalam bentuk Xgraph, *file *.tr* –nya disusun sedemikian rupa menjadi 2 kolom. Perintah pembuatan Xgraph pada prosedur *finish*. Format perintahnya sebagai berikut:

```
exec xgraph out.tr –geometry 300x 300
```

xgraph diplot pada bidang berukuran 300 x 300 pixel.

Gambar 2.7 memperlihatkan contoh hasil simulasi dengan menggunakan xgraph berdasarkan *script* NS berikut.



Gambar 2.7 Contoh Hasil Simulasi xgraph

2.8 *FORWARD ERROR CORRECTION*

Dalam teori telekomunikasi dan informasi dijelaskan bahwa *Forward Eror Correction* adalah sebuah mekanisme sistem untuk melakukan *control error* pada data transmisi. Dimana pengirim menambahkan data tambahan (*redundant*) kedalam pesan/paket data tersebut, yang juga dikenal sebagai *error correction code*. Hal ini membolehkan *receiver* untuk melakukan deteksi dan memperbaiki *error* (dengan beberapa batasan) tanpa membutuhkan izin dari pengirim dalam menambahkan datanya.

Keuntungan dari FEC ini adalah *back-channel* tidak dibutuhkan, atau retransmisi data dapat dihindarkan, sehingga tidak menimbulkan biaya yang besar dalam kebutuhan *bandwidth*. oleh karena itu FEC diterapkan dalam situasi dimana retransmisi sangat relatif terhadap biaya atau tidak dimungkinkan untuk melakukan retransmisi. Khususnya, informasi FEC biasanya ditambahkan pada banyak *mass storage devices* untuk melindungi dari kerusakan data yang tersimpan.

FEC *devices* seringkali lokasinya tertutup terhadap penerima pada sinyal analog, pada proses sinyal digital setelah sinyalnya sampai. FEC merupakan integral dari bagian proses konversi dari *analog-to-digital* juga menyangkut modulasi digital dan demudulasi atau *lide coding dan decoding*. Banyak koder FEC hanya dapat membangkitkan sinyal BER (*bit-error-rate*) dimana dapat digunakan sebagai *feedback* untuk melakukan *tuning* pada alat penerima analog.

Dan juga banyak algoritma *detector* FEC seperti *EAFEC, viterbi, Reed Solomon* dll. Tergantung mana yang digunakan sesuai dengan kondisi dan desain yang cocok.

2.8.1 Cara Kerja

FEC sangat pandai, hanya dengan menambahkan *redundant* kedalam informasi yang dikirimkan menggunakan sebuah Algoritma untuk mengantisipasinya. Tiap informasi bit yang ditambahkan (*redundant*) bervariasi tergantung kekomplekan fungsi pada bit informasi yang aslinya. Informasi asli (*original*) dapat dimunculkan atau tidak dimunculkan pada *output encoded*. Kode yang termasuk *unmodified input* dalam output merupakan *systematic*, sebaliknya yang tidak termsuk merupakan *nonsystematic*.

Contoh yang simple seperti pada konversi analog ke digital, dengan 3 contoh bit dari *signal strength* data untuk setiap bit data yang ditransmisikan. Jika pada ketiga contoh pada umumnya nol (0) maka kemungkinan bit yang ditransmisikan juga nol (0), dan jika pada ketiga bit umunya satu (1) maka bit yang terkirimkan seharusnya satu (1). Seperti pada Table 2.1 *"democratic voting*" dibawah ini.

Table 2.1. [9] Contoh tabel "*democratic voting"*

Triplet *receiver*	Interpreted as
000	0
001	0
010	0
100	0
111	1
110	1
101	1
011	1

2.8.2 Tipe FEC

Ada dua kategori dari FEC yaitu :

- *Blok Coding*
- *Convolutional coding*

Blok code bekerja pada blok (paket) yang ukuranya bitnya tetap atau siymbol dari antisipasi ukuran. *Konvolutional code* bekerja pada bit atau simbol *stream* dari *length arbitrary.*

2.8.3. EAFEC algorithm

Sebuah mekanisme FEC pada level paket akan ditambahkan *h* (persamaan paket) ke blok lain yaitu *k* (paket data). Pada sisi *receiver*, paket *parity* ini di buang dan paket data yang hilang di regenerasi. *Decoder* FEC mampu untuk merekonstruksi paket asli (*original*) dari banyak *k* paket hilang pada *h+k* paket yang terkirim. FEC menghantarkan paket jumlah banyak di jaringan *wireless*. Ketika jaringannya kongesti atau mendekati batas kongesti yang membangkitkan performance yang buruk.

EAFEC [4] mencoba untuk menghindari kongesti (penumpukan) dengan menentukan jumlah yang benar dari *parity* paket untuk ditambahkan berdasar dua parameter yaitu : *queue length* pada *base stasion* dan jumlah maksimum retransmisi pada mekanisme ARQ.

Pertama EAFEC menghitung variable *nrFEC*, dimana juga mendefinisikan jumlah paket FEC yg akan di-*generate* (dibangkitkan) dengan melihat *queue length* nya. Jika *queue length* nya lebih kecil dari dari nilai *threshold* (th_1) maka *nrFEC* di set maksimum nomor paket FEC yang ditambahkan. Ketika *queue length* melebihi *thereshold* tertentu, *nrFEC* di set ke nilai 0 (*zero*). Dengan kata lain *nrFEC* di set berdasarkan ukuran fraksi dalam *queue*. Selanjutnya EAFEC mengkalkulasikan jumlah yang tepat pada paket FEC untuk di-gene*rate* (bangkitkan) dengan menguji jumlah retransmisi pada DFWMAC. Jika restransmisi lebih kecil dari pada nilai *threshold* 3 (th_3), maka tidak ada paket

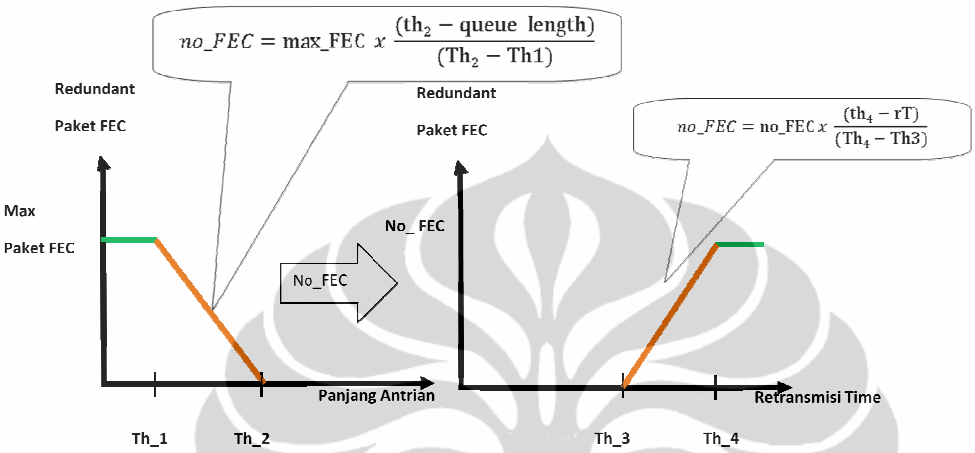
FEC yang di *generate*., jika *retransmisi* lebih besar dari nilai *threshold* 4 (th_4) maka jumlah dari paket FEC akan di *generate* dan di set ke *nrFEC*. Dengan kata lain, jumlah dari paket FEC akan di *generate* naik dengan jumlah retransmisi. Hal ini dapat dilihat *pseudocode* nya pada Gambar 2.8. dibawah ini.

```
Initialization:
 qlen = 0; rT = 0;
 When a block of packets arrive:
 /* use queue length to determine number of redundant FEC packets */
qlen = (1 – qweight ) * current_q + qweight * qlen
 if (qlen < threshold1)
  no_FEC = Max_FEC;
 else if (qlen < threshold2)
= Max_FEC * (threshold2-qlen) / (threshold2-threshold1);
 else
   no_FEC = 0;
 /* use retransmission time to reduce the no_FEC */
rT = (1 – rweight) * current_rT + rweight * rT
 if (rT < threshold3)
   no_FEC = 0;
 else if (rT < threshold4)
= no_FEC* ( 1 – ((threshold4 – rT)/ (threshold4-threshold3)));
 else
   no_FEC = no_FEC;
```

Gambar 2.8 EAFEC Pseudocode

Ketika sebuah blok paket pertama kali diterima, *Akses Point* (AP) akan mengkalkulasi panjang antrian dengan nilai *weight*. Kemudian *access point* (AP) melakukan komparasi *queue length* dengan nilai *threshold*. Jika nilai *queue length* lebih kecil dari dari nilai *high_threshold* (*threshold1*) maka maksimum *packet* akan akan di *generate* (dibangkitkan) . Jika lebih besar dari nilai *low_threshold* (*threshold2*), maka tidak ada paket FEC yang di *generate*. Dengan kata lain paket

FEC di *generate* berdasarkan pecahan ukuran data dalam antrian. Jumlah paket FEC kemudian dikalkulasikan lagi menurut waktu paket yang dibutuhkan untuk merenstranmisi. Mekanisme ini dapat dilihar pada Gambar 2.9 dibawah ini.



Gambar 2.9 Mekanisme EAFEC Algoritma

Table 2.2 : The Notation of EAFEC algorithm.

qlen	*weighted moving average queue length*
qweight	*queue weight*
current_q	*current queue length*
threshold1	*low threshold* untuk *queue*
threshold2	*high threshold* untuk *queue*
no_FEC	Jumlah *redundant Paket FEC* yang akan di-*generated*
Max_FEC	nilai *maximum* Jumlah Paket *redundant FEC*
rT	*average* waktu *retransmission*
current_rT	waktu *current retransmission*
threshold3	*low threshold untuk waktu retransmission*
threshold4	*high threshold* untuk waktu *retransmission*