BAB II
TINJAUAN PUSTAKA

Tinjauan pustaka yang digunakan dalam penelitian skripsi ini bersumber dari buku referensi, jurnal, skripsi, internet, dan forum-forum resmi *Video On Demand* dan *Virtual Private Network*. Tinjauan pustaka yang akan diuraikan dalam bab ini meliputi: *video streaming*, komponen VOD, karakteristik *video*, protokol pendukung pada *video streaming*, OpenVPN, komponen jaringan komputer, dan parameter performansi jaringan.

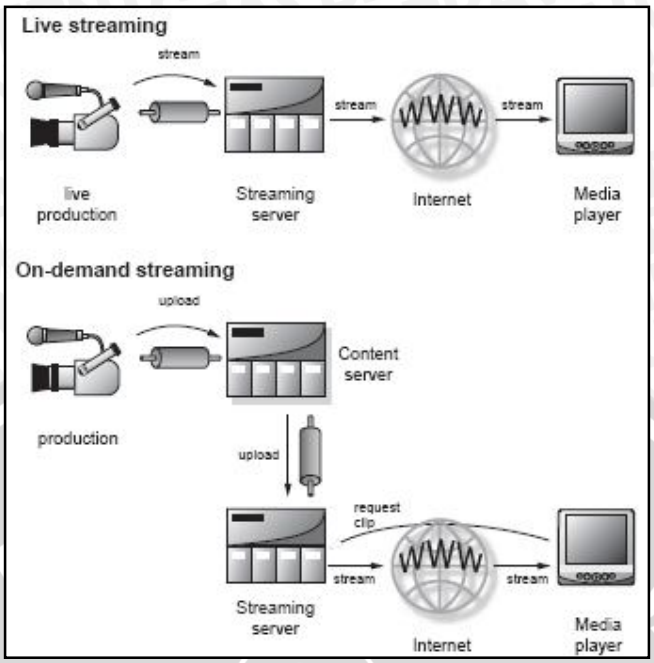
2.1 *Video Streaming*

Video streaming adalah urutan dari gambar yang bergerak, dikirimkan dalam bentuk yang telah dikompresi melalui jaringan *internet* dan ditampilkan oleh *player* ketika *video* tersebut telah diterima oleh *user* yang membutuhkan. Pengguna atau *user* membutuhkan *media player*, yaitu aplikasi khusus yang melakukan dekompresi dan mengirimkan *data* berupa *video* ke tampilan layar *monitor* dan suara ke *speaker*.

Ada dua jenis tipe *video streaming*, yaitu :

1. *Live Streaming*, dimana tayangan yang ditampilkan merupakan siaran langsung.
2. VOD (*Video On Demand*) dimana *video* yang ditampilkan sudah terlebih dahulu direkam (*pre encoded*) atau disimpan dalam *server*.

Faktor-faktor yang berpengaruh dalam distribusi *video streaming* melalui jaringan antara lain besar *bandwidth* tersedia yang bervariasi (terhadap waktu), *delay* (waktu tunda), *lost packet*, dan juga teknik mendistribusikan video tersebut ke beberapa tujuan secara merata dan efisien. (Apostolopoulos, 2002 : 1)



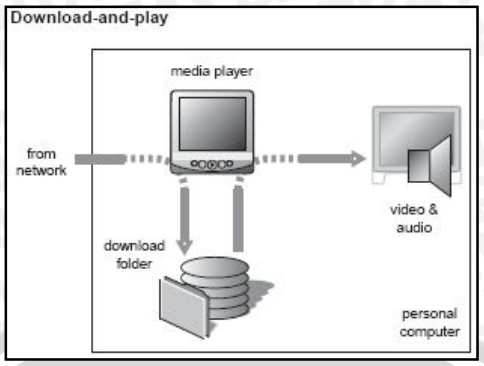
Gambar 2.1 *Live Streaming* dan *On-demand Streaming*

Sumber : Austerberry, 2004 : 144

Cara yang digunakan untuk menerima *stream data* (*video, audio* dan animasi) dari internet atau jaringan, yaitu : *download*, *streaming*, dan *progressive downloading*.

a. Download

Pada penerimaan *stream* data dengan cara *download*, akses *video* dilakukan dengan cara melakukan *download* terlebih dahulu suatu *file multimedia* dari *server*. Penggunaan cara ini mengharuskan keseluruhan suatu *file multimedia* harus diterima secara lengkap di sisi *client*. *File multimedia* yang sudah diterima kemudian disimpan pada *harddisk client*, dimana penyimpanan tersebut bersifat permanen. Setelah *file multimedia* tersebut berhasil diterima secara lengkap pada sisi *client*, *user* baru dapat mengakses *video* tersebut. Adapun salah satu keuntungan dari penggunaan cara ini adalah akses yang lebih cepat ke salah satu bagian dari *file* tersebut. Namum, kekurangan dari penggunaan cara ini adalah seorang *user* yang ingin mengakses secara langsung *video* yang diterima harus terlebih dahulu menunggu hingga keseluruhan suatu *file multimedia* selesai diterima secara lengkap.

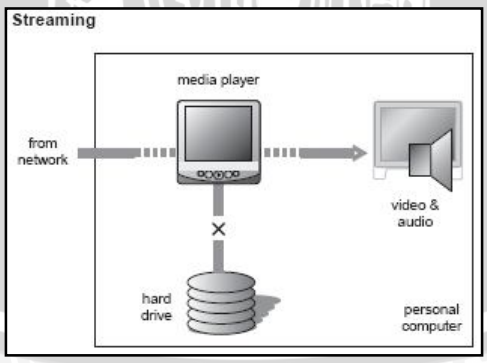


Gambar 2.2 Proses *Download video*

Sumber : Austerberry, 2004 : 151

b. Streaming

Pada penerimaan *video* dengan cara *streaming*, seorang *user* dapat mulai melihat suatu *file multimedia* hampir bersamaan ketika *file* tersebut mulai diterima. Penggunaan cara ini mengharuskan pengiriman suatu *file multimedia* ke *user* dilakukan secara konstan. Hal ini bertujuan agar seorang *user* dapat menyaksikan *video* yang diterima secara langsung tanpa adanya bagian yang hilang. Keuntungan utama dari penggunaan cara ini adalah seorang *user* tidak perlu menunggu hingga suatu *file multimedia* diterima secara lengkap. Cara ini juga mengurangi resiko penggandaan *file video* secara illegal, karena paket-paket data akan dibuang tanpa disimpan terlebih dahulu pada *harddisk client* seteleh proses *streaming* selesai dilakukan. Dengan demikian, penggunaan cara ini memungkinkan sebuah *server* untuk melakukan pengiriman siaran langsung (*live events*) kepada *user*.



Gambar 2.3 Proses *Streaming* video

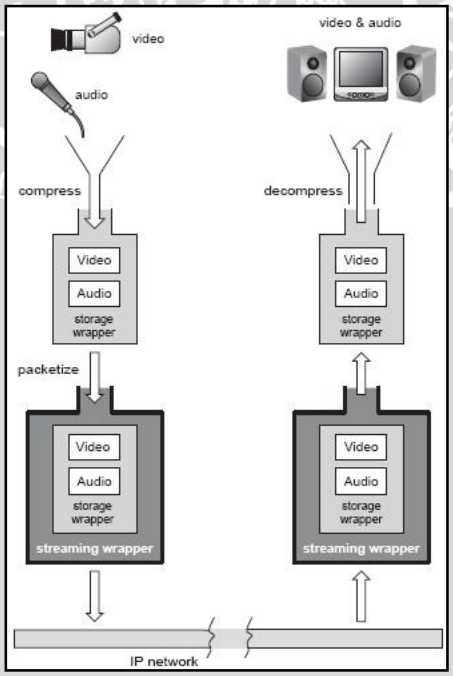
Sumber : Austerberry, 2004 : 151

c. Progressive Downloading

Progressive downloading adalah metode *hybrid* yang merupakan hasil penggabungan antara metode *download* dengan metode *streaming*, di mana *video* yang sedang diakses diterima dengan cara *download*, dan *player* pada sisi *user* sudah dapat mulai menampilkan *video* tersebut sejak sebagian dari *file* tersebut diterima walaupun *file* tersebut belum diterima secara sepenuhnya.

2.2 Komponen Pada VOD (*Video On Demand*)

Sistem VOD memungkinkan *user* untuk memilih dan menyaksikan *video* yang hendak diakses dalam jaringan sebagai bagian dari sistem interaktif. VOD dapat memanfaatkan proses *streaming, progressive downloading*, ataupun *download*. Sistem VOD juga memungkinkan pengguna untuk melakukan kendali, seperti *pause, fast forward, fast rewind, slow forward*, dan *slow rewind*. Namum pada sistem yang menggunakan metode *streaming*, hal ini akan membebani *server* dan memerlukan pemakaian *bandwidth* yang lebih besar.



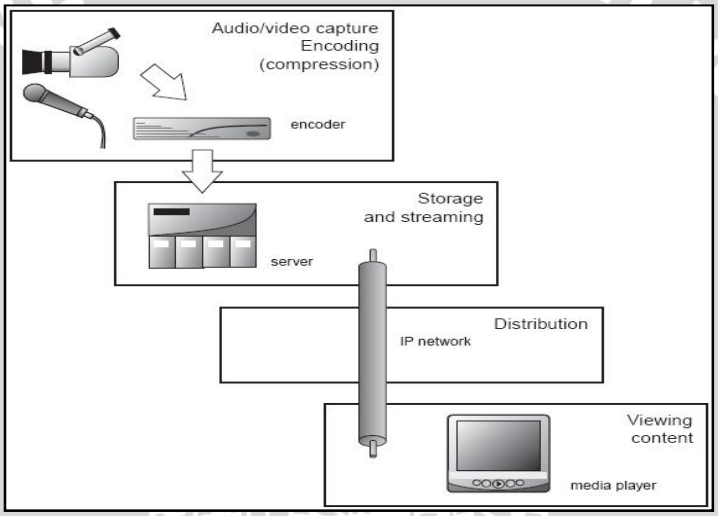
`Gambar 2.4 *Video On Demand*

Sumber : Austerberry, 2004 : 141

Gambar 2.4 menunjukkan proses *streaming* pada *video on demand. File video* yang di*request* oleh *client* tersimpan pada *server*, yang kemudian akan dilakukan

kompresi untuk menghemat *bandwidth* yang dibutuhkan jaringan. *File video* yang terkompresi akan dipecah menjadi paket-paket yang dikirimkan melalui jaringan menuju *client.* Proses pengiriman paket *video* menggunakan *User Datagram Protocol*. Pada sisi *client,* paket-paket *video* yang diterima akan disimpan dalam *memory* temporal komputer yang disebut dengan *buffer,* untuk digabungkan menjadi *file video* utuh dan dilakukan dekompresi. Selanjutnya *client* dapat memainkan *file video* melalui *media player*.

Secara umum, terdapat empat buah komponen dari *multimedia streaming* (Austerberry, 2004 :139) yaitu : *capture and encoding*, *serving*, *distribution*, dan *player/output*. Gambar 2.5 berikut menunjukkan empat buah komponen *streaming* pada suatu sistem.



Gambar 2.5 Empat Buah Komponen *Streaming*

Sumber : Austerberry, 2004 : 139

a. Capture and Encoding

Proses ini dilakukan kompresi dan konversi *file* dari mikropon dan kamera menjadi *file* terkompresi dengan ukuran yang lebih kecil. *File* hasil kompresi ini kemudian di-*upload* pada *server*, dimana akan dilakukan proses pengontrolan *streaming*. Ada beberapa proses dalam hal ini :

- *Capture* ke format *file* komputer
- Kompresi *file*
- Mengubah menjadi bentuk paket-paket data

Aplikasi yang disebut *compressor-decompressor* atau yang disebut dengan *codec*. *Compressor* akan mengubah *file video/audio* menjadi format yang sesuai dan mengurangi *data rate* untuk disesuaikan dengan kebutuhan *bandwidth streaming* yang tersedia. *Decompressor* berada pada sisi *media player client* dan mengubah *streaming* data menjadi *video/audio* kembali. Pada *encoder* juga dilakukan proses *wrapping stream data* dengan *index* tertentu, yang mana digunakan untuk proses pengontrolan proses pengiriman *real-time* (Austerberry, 2004 : 140).

b. Serving

File hasil *encoding* kemudian didistribusikan oleh *server* kepada *client*. Pada *server* juga dilakukan kontrol terhadap proses pengiriman *real-time*. Pada aplikasi yang digunakan, *encoder* dan *server* berada pada satu aplikasi yang terintegrasi satu dengan lainya.

c. Distribution

Secara mendasar, pada proses distribusi adalah sederhana. Selama terdapat konektivitas alamat IP antara *server* dengan *client*, paket-paket *video/audio* yang diperlukan akan tiba pada *client*. Beberapa pengembangan untuk meningkatkan kualitas pengiriman paket-paket data dari *server* ke *client*. Salah satunya adalah meningkatkan *bandwitdh* sehingga kualitas dari *streaming* itu sendiri akan lebih baik.

d. Player / Output

Player berfungsi untuk melakukan *decoding* terhadap *file* hasil *streaming* dan menampilkan pada sisi *client*.

2.3 Karakteristik *Video*

Karakteristik suatu *video* ditentukan oleh tiga faktor, yaitu : resolusi gambar (*Resolution*), kedalaman *pixel*, dan laju *frame*.

a. Resolusi Gambar (*Resolution*)

Resolusi gambar/*frame* adalah ukuran sebuah gambar/*frame*. Resolusi dinyatakan dengan perkalian antara panjang dan lebar gambar/*frame*. Semakin tinggi resolusi, semakin baik kualitas *video* tersebut. Namum, resolusi tinggi akan mengakibatkan jumlah bit yang diperlukan untuk menyimpan atau mentransmisikan akan meningkat. Tabel 2.1 menunjukkan standar resolusi *video*.

Tabel 2.1 Standar Resolusi *Video*

No	Nama	Resolusi (w x h)	*Aspect Ratio*	Keterangan
1	QCIF	176 x 144	4 : 3	*Quarter Common Intermediate Format*
2	CGA	320 x 200	16 : 10	*Color Graphic Array*
3	QVGA	320 x 240	4 : 3	*Quarter Video Graphic Array*
4	CIF	352 x 288	4:3	*Common Intermediate Format*
5	HVGA	480 x 320	4 : 3	*Half Video Graphic Array*
6	VGA	640 x 480	4 : 3	*Video Graphic Array*
7	SVGA	800 x 600	4 : 3	*Super Video Graphic Array*
8	WVGA	854 x 480	16 : 9	*Widecreen* VGA
9	XGA	1024 x 768	4 : 3	*Extended Graphic Array*
10	XGA+	1152 x 768	3 : 2	*Extended Graphic Array Plus*
11	WXGA	1280 x 800	16 : 10	*Widescreen* XGA
12	SXGA	1280 x 1024	3 : 2	*Super* XGA
13	SXGA+	1400 x 1050	4 : 3	*Super* XGA *Plus*
14	UXGA	1600 x 1200	4 : 3	*Ultra* XGA
15	WSXGA+	1680 x 1050	16 : 10	*Widescreen* SXGA+
16	WUXGA	1920 x 1200	16 : 10	*Widescreen Ultra* XGA
17	QXGA	2048 x 1536	4 : 3	*Quad* XGA
18	WQXGA	2560 x 1600	16 : 10	*Widescreen Quad* XGA
19	QSXGA	1560 x 2048	5 : 4	*Quad Super* XGA

Sumber : http://www.equasys.de/standardresolution.html

b. Kedalaman *Pixel*

Kedalaman *pixel* menentukan jumlah bit yang digunakan untuk mereprensentasikan tiap *pixel* pada sebuah *frame*. Kedalaman bit dinyatakan dalam bit/*pixel*. Semakin banyak jumlah bit yang digunakan untuk merepresentasikan sebuah *pixel*, semakin tinggi kedalaman *pixel*-nya yang berarti semakin tinggi pula kualitas *video*nya. Hal ini akan mengakibatkan jumlah bit yang diperlukan untuk menyimpan atau mentransmisikanya meningkat, misalnya : 24 bit menunjukkan 16.777.216 warna, 16 bit menunjukkan 65.535 warna, dan 8 bit menunjukkan 256 warna.

c. Laju Frame (*Frame Rate*)

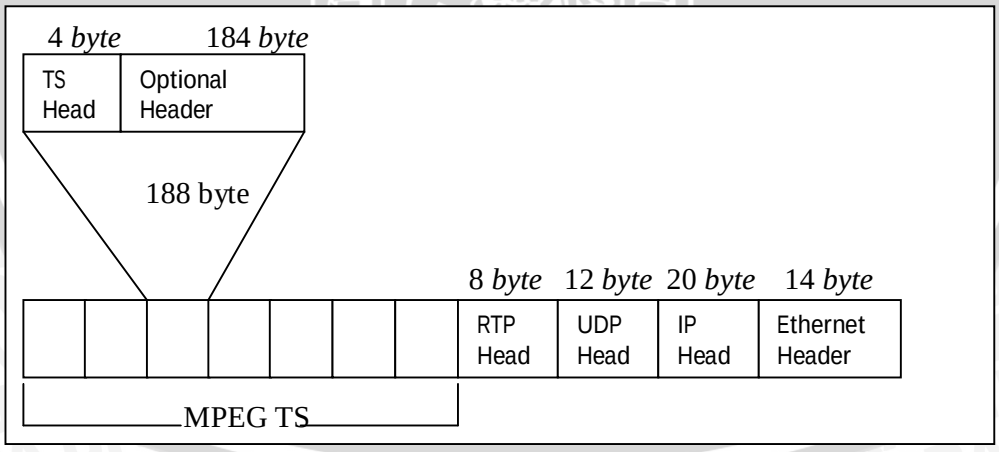
Laju *frame* menunjukkan jumlah *frame* yang ditampilkan secara bergantian dan urut setiap detiknya. Semakin tinggi laju *frame*, semakin tinggi pula kualitas *video*. Namum, hal ini akan mengakibatkan jumlah bit yang diperlukan untuk menyimpan atau mentransmisikanya meningkat.

2.4 Protokol Pendukung pada *Video Streaming*

Protokol adalah aturan dalam komunikasi data yang berguna untuk menjaga agar pengiriman data dapat berlangsung. Dalam komunikasi data diperlukan protokol yang bersifat universal agar komunikasi bisa dilakukan oleh berbagai macam pengguna dengan *platform* yang berbeda. Beberapa protokol yang digunakan dalam *streaming*, yaitu : MPEG *transport stream,* TCP, UDP, dan RTP

a.* MPEG *Transport Stream

Menurut standar MPEG-4, dalam proses *delivery* paket data *audio* dan *video* yang di*encoding* dengan MPEG *codec* dienkapsulasi menjadi sebuah paket *multimedia* yang terdiri dari data *audio* dan *video* yang sudah termultipleks. Paket data ini disebut dengan MPEG *Transport Stream* (MPEG TS). Sebuah TS terdiri dari sebuah paket data dengan panjang maksimum 188 *byte*. Setiap paket TS terdiri dari 4 *byte* TS *header* dan 184 *byte* sisanya terdiri dari *optional header* dan *payload* data *multimedia*.



Gambar 2.6 Paket Data *Multimedia streaming* dengan enkapsulasi MPEG TS

Sumber : Vigato, 2005

b. TCP/IP

TCP/IP (*Transmission Control Protocol/Internet Protocol*) merupakan sekelompok protokol yang mengatur komunikasi data di internet. Karena menggunakan protokol yang sama, maka perbedaan komputer dan jenis sistem operasi antara dua buah komputer tidak menjadi masalah.

TCP/IP terdiri atas kumpulan protokol yang masing-masing bertanggung jawab atas bagian-bagian tertentu komunikasi data, sehingga tugas masing-masing protokol menjadi jelas dan sederhana. Protokol yang satu tidak perlu saling mengetahui protokol yang lain, sepanjang ia masih bisa saling mengirim dan menerima data. Terdiri dari empat *layer*, yaitu :

- *Network Layer/Interface Layer*
- *Internet Layer*
- *Host-to-host Transport layer*
- *Application layer*

Gambar 2.7 berikut menunjukkan ke-empat *layer* dalam TCP/IP.

Gambar 2.7 Arsitektur TCP/IP

Sumber : Forouzan, 2007 : 43

Masing-masing layer memiliki tugas yang berbeda :

1. *Physical and Data Link Layer*, bertanggung jawab mengirim dan menerima data ke dan dari media fisik. Media fisik dapat berupa kabel, serat optik, atau gelombang radio. Protokol pada *layer* ini harus mampu menerjemahkan sinyal listrik menjadi data digital yang dimengerti komputer, yang berasal dari peralatan lain yang sejenis.
2. *Internet Layer*, bertanggung jawab dalam proses pengiriman packet ke alamat yang tepat. Pada *layer* ini terdapat tiga macam protokol, yaitu : IP, ARP, dan

ICMP. IP (*Internet Protocol*) berfungsi untuk menyampaikan paket data ke alamat yang tepat. ARP (*Address Resolution Protocol*) ialah protokol yang digunakan untuk menemukan alamat *hardware* dari *host*/komputer yang terletak pada jaringan yang sama. ICMP (*Internet Control Message Protocol*) ialah protokol yang digunakan untuk mengirimkan pesan dan melaporkan kegagalan pengiriman data.

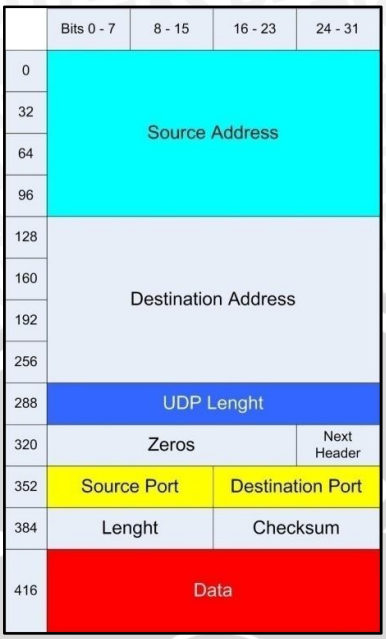
3. *Transport Layer*, berisi protokol-protokol yang bertanggung jawab untuk mengadakan komunikasi antara dua *host*/komputer. Kedua protokol tersebut adalah TCP (*Transmission Control Protocol*) dan UDP (*User Datagram Protocol*).
4. *Application Layer*, pada layer ini terletak semua aplikasi yang menggunakan TCP/IP, seperti : FTP (*File Transfer Protocol*), SMTP (*Simple Mail Transfer Protocol*), HTTP (*Hyper Text Transfer Protocol*), dan lain-lain.

c. TCP (*Transmission Control Protocol*)

TCP merupakan protokol yang terletak di *layer transport* pada arsitektur protokol TCP/IP. TCP bertanggung jawab dalam menyediakan layanan *connection oriented, reliable, byte stream service* bagi *Application layer*. *Connection oriented* berarti sebelum melakukan pertukaran data, dua aplikasi pengguna TCP harus melakukan pembentukan hubungan (*handshake*) terlebih dahulu. *Reliable* berarti TCP menerapkan proses deteksi kesalahan paket dan retransmisi. *Byte stream service* berarti paket dikirimkan sampai ke tujuan secara berurutan.

d. UDP (*User Datagram Protocol*)

UDP merupakan protokol yang berada di lapisan selain TCP. Protokol ini bersifat *connectionless* dan *unreliable.* Dalam pengiriman data, *connectionless* berarti tidak diperlukan pembentukan hubungan terlebih dahulu untuk mengirimkan data. Sedangkan *unreliable*, berarti protokol ini, tidak menjamin data akan sampai pada tujuan dalam kondisi benar. Aplikasi yang memanfaatkan UDP sebagai protokol *transport*, dapat mengirimkan data tanpa melalui proses pembentukan koneksi terlebih dahulu. Lebih sering diimplementasikan dalam aplikasi *multimedia streaming* dimana rugi-rugi paket data masih bisa ditoleransi daripada nilai *delay* yang lebih besar. Gambar 2.8 menunjukkan format paket data UDP.



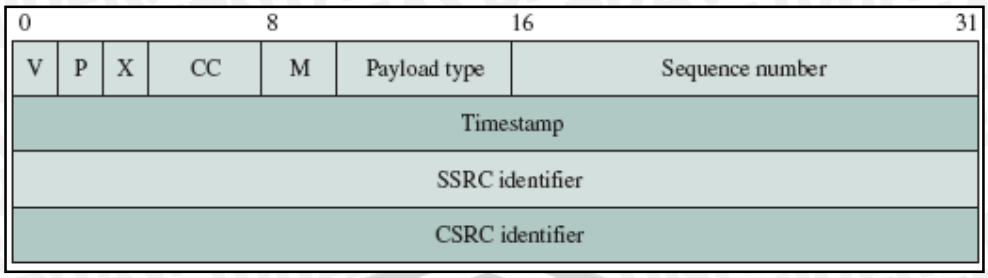
Gambar 2.8 Format Paket Data UDP

Sumber : Forouzan, 2007 : 712

e. RTP (*Real-Time Transport Protocol*)

RTP mendefinisikan sebuah format paket standar untuk mengirimkan *audio* dan *video* melalui internet. *Protocol* ini dikembangkan oleh IETF Audio-Video Transport Working Group. RTP tidak memiliki standar *port* TCP atau UDP untuk digunakan dalam berkomunikasi. Standar komunikasi yang digunakan adalah UDP dengan nomor *port* yang genap dan nomor *port* yang ganjil berikutnya memiliki nilai yang lebih tinggi digunakan untuk komunikasi RTP *Control Protocol* (RTCP). RTP dapat membawa data apapun dengan karakteristik *real-time*, seperti *audio* dan *video* interaktif.

Pada awalnya RTP dikembangkan untuk *protocol multicast*, namum banyak digunakan juga untuk aplikasi *unicast*. RTP dibangun berdasarkan protokol UDP. Aplikasi yang menggunakan RTP kurang peka terhadap hilangnya paket (*packet loss*), namum sangat peka terhadap *delay*, sehingga hal ini menjadikan UDP sebagai pilihan yang lebih baik daripada TCP untuk aplikasi semacam itu.



Gambar 2.9 *Real-time Transport Protocol*

Sumber : Garcia, 2007 : 796

Keterangan :

- V (*Version*) : (2 bit) digunakan sebagai identitas versi dari protokol. Versi ini adalah 2
- P (*Padding*) : (1 bit) Mengindikasikan adanya Alas tambahan pada akhir dari paket RTP. Sebuah *padding* digunakan untuk memenuhi sebuah blok pada ukuran tertentu, khususnya diperlukan pada metode algoritma tertentu.
- X (*Extension*) : (1 bit) Mengindikasikan adanya dari sebuah *Extension Header* diantara *standard header* dan *payload data*.
- CC (CSRC *Count*) : (4 bit) Mengandung nomor dari pengalamatan CSRC yang mengikuti *fixed header*.
- M (*Marker*) : (1 bit) Digunakan pada level aplikasi dan menentukan sebuah profil.
- PT (*Payload Time*) : (7 bit) Mengindikasikan *format* dari *payload* dan penafsiran oleh aplikasi.
- *Sequence Number* : (16 bit) Jumlah dari *sequence number* meningkat satu kali untuk setiap packet data RTP dikirimkan dan digunakan penerima untuk mengetahui *packet loss* dan mengembalikan *packet sequence*.
- *Timestamp* : (32 bit) digunakan *receiver* untuk *play back sample* yang diterima pada interval yang sesuai.
- SSRC : (32 bit) Pengalamatan sumber sinkronisasi yang bersifat unik digunakan untuk mengetahui sumber dari *stream*.
- CSRC : berkonstribusi dalam menghitung sumber *stream* yang mana dihasilkan dari bermacam-macam sumber.

RTP mengirimkan data-data yang diperlukan agar aplikasi dapat menyusun paket data yang dikirimkan dengan urutan yang benar. Selain itu juga RTP menyediakan informasi mengenai kualitas penerimaan yang dapat digunakan oleh

aplikasi untuk dibuat penyesuaian. Sebagai contoh, bila ada kemungkinan terbentuknya *congestion*, maka aplikasi dapat memutuskan untuk menurunkan *data rate*.

2.5 OpenVPN

OpenVPN adalah salah satu jenis *Virtual Private Network* (VPN) yang bebas dan bersifat *opensource* program untuk membuat koneksi *point-to-point* atau *server-to-multiclient* dengan enkripsi *tunneling* diantara *host* komputer. Hal itu memungkinkan terjadinya hubungan langsung dengan komputer dibelakang *firewall* tanpa perlu konfigurasi ulang.

OpenVPN memungkinkan terhubungnya komputer melalui autentifikasi dengan komputer yang lain dengan menggunakan kunci rahasia (*secret key*), *certificates*, atau *username-password*. Ketika digunakan sebagai *multiclient-server* konfigurasinya dimungkinkan *server* membuat autentifikasi untuk setiap *client*nya, dengan menggunakan *signature* (tanda tangan digital) dan *certificate authority*. OpenVPN menggunakan OpenSSL sebagai enkripsinya. OpenSSL tersedia pada operating system seperti Solaris, Linux, OpenBSD, FreeBSD, NetBSD, Mac OS X, dan Windows 2000/XP/Vista.

Virtual Private Network merupakan perpaduan dari dua teknologi yaitu : teknologi *tunneling* dan teknologi enkripsi.

a. Teknologi *Tunneling*

Teknologi *tunneling* merupakan teknologi yang bertugas untuk menangani dan menyediakan koneksi *point-to-point* dari sumber ke tujuannya. Disebut *tunnel* karena koneksi *point-to-point* tersebut sebenarnya terbentuk dengan melintasi jaringan umum, namun koneksi tersebut tidak memperdulikan paket-paket data milik orang lain yang sama-sama melintasi jaringan umum tersebut, tetapi koneksi tersebut hanya melayani transportasi data dari pembuatnya. Hal ini sama dengan seperti penggunaan jalur *busway* yang pada dasarnya menggunakan jalan raya, tetapi dia membuat jalur sendiri untuk dapat dilalui bus khusus.

Koneksi *point-to-point* ini sesungguhnya tidak benar-benar ada, namun data yang dilewatkan terlihat seperti benar-benar melewati koneksi pribadi yang bersifat *point-to-point*.

Teknologi ini dapat dibuat di atas jaringan dengan pengaturan *IP Addressing* dan *IP Routing* yang sudah matang, antara sumber *tunnel* dengan tujuan *tunnel* telah dapat saling berkomunikasi melalui jaringan dengan pengalamatan IP. Apabila komunikasi

antara sumber dan tujuan dari *tunnel* tidak dapat berjalan dengan baik, maka *tunnel* tersebut tidak akan terbentuk dan VPN pun tidak dapat dibangun.

Apabila *tunnel* tersebut telah terbentuk, maka koneksi *point-to-point* tersebut dapat langsung digunakan untuk mengirim dan menerima data. Namun, di dalam teknologi VPN, *tunnel* tidak dibiarkan begitu saja tanpa diberikan sistem keamanan tambahan. *Tunnel* dilengkapi dengan sebuah sistem enkripsi untuk menjaga data-data yang melewati *tunnel* tersebut. Proses enkripsi inilah yang menjadikan teknologi VPN menjadi mana dan bersifat pribadi.

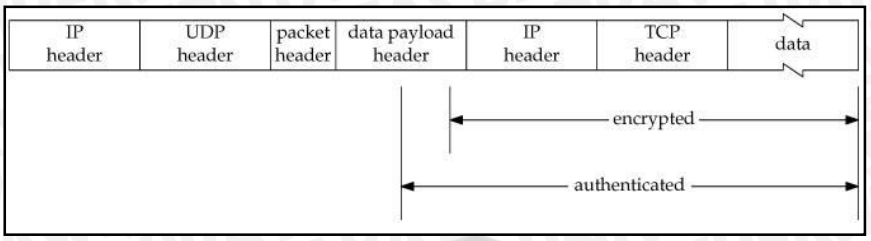
VPN dapat mengirimkan data dengan proses *encapsulasi*, teknologi *tunneling* merupakan pokok dari VPN yaitu membuat jalur *virtual* untuk melewatkan data – data yang sifatnya pribadi.

b. Teknologi Enkripsi

Teknologi enkripsi menjamin data yang berlalu-lalang di dalam *tunnel* tidak dapat dibaca dengan mudah oleh orang lain yang bukan merupakan komputer tujuannya. Semakin banyak data yang lewat di dalam *tunnel* yang terbuka di jaringan publik, maka teknologi enkripsi ini semakin dibutuhkan. Enkripsi akan mengubah informasi yang ada dalam *tunnel* tersebut menjadi sebuah *ciphertext* atau teks yang dikacaukan dan tidak dapat dibaca secara langsung. Untuk dapat membuatnya kembali memiliki arti atau dapat dibaca, maka dibutuhkan proses dekripsi. Proses dekripsi terjadi pada ujung-ujung dari hubungan VPN. Pada kedua ujung ini telah menyepakati sebuah algoritma yang aka digunakan untuk melakukan proses enkripsi dan dekripsinya. Dengan demikian, data yang dikirim aman sampai tempat tujuan, karena orang lain di luar *tunnel* tidak memiliki algoritma untuk membuka data tersebut.

2.5.1 OpenVPN Data Channel

Open VPN dapat mengirimkan data dengan baik menggunakan UDP *datagram*. maupun menggunakan koneksi TCP *datagram*. Open VPN memiliki *encapsulasi* seperti Gambar 2.10 berikut.

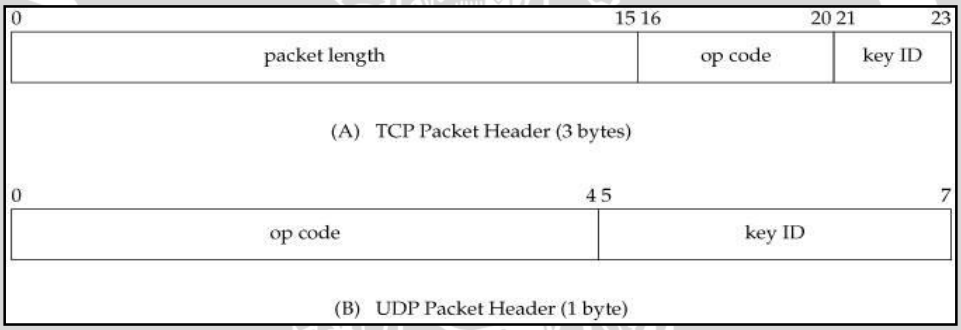


Gambar 2.10 *The OpenVPN Data Channel Encapsulation*

Sumber : Snader, 2007:45

Gambar 2.10 merupakan contoh *datagram* dasar OpenVPN dengan *payload datagram* TCP/IP atau dapat berupa *frame* yang lain misal RTP atau UDP.

OpenVPN membagi *payload header* menjadi dua bagian: yaitu *packet header* yang berfungsi mengdentifikasi *type* paket data dan material keamanan, dan data *payload header* berfungsi membawa informasi authentifikasi, dan membawa rangkaian paket data.



Gambar 2.11 *The OpenVPN Packet Header*

Sumber : Snader, 2007:50

2.6 Komponen Jaringan Komputer

Komponen jaringan komputer yang digunakan dalam proses pengujian sistem, yaitu : *Switch*, *Network Interface Card*, Kabel UTP dan Komputer.

a. Switch

Switch merupakan perangkat keras yang berfungsi menyatukan antar *client* dan *server* pada jaringan *Video On Demand*.

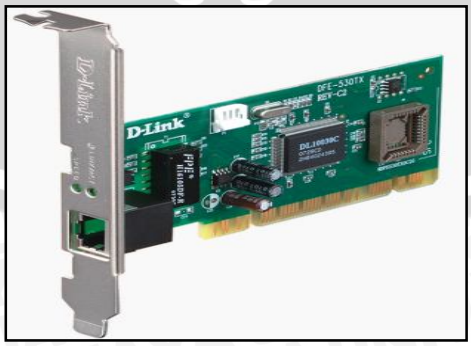
Gambar 2.12 *Switch*

Sumber : Perangkat Keras yang digunakan dalam perancangan sistem

Penggunaan *Switch* dibutuhkan untuk menghubungkan komponen-komponen pada jaringan *Video On Demand* dan *Virtual Private Network. Switch* jaringan mampu memeriksa paket data yang diterima, menentukan sumber dan perangkat tujuan masing-masing paket informasi, dan melanjutkan informasi serta data secara tepat. Pada perancangan ini digunakan *Switch* Prolink PSW 510.

b. *Network Interface Card*

Network Interface Card (NIC) adalah sebuah perangkat keras yang berfungsi sebagai *interface* dari komputer ke sebuah jaringan komputer. Pada perancangan ini, NIC digunakan sebagai antarmuka jaringan yang menghubungkan antara komputer *client* dan jaringan sehingga dapat mentransmisikan informasi ke jaringan dengan media transmisi berupa kabel UTP Cat.5. Prinsip kerja NIC adalah mengubah aliran data paralel dalam *bus* komputer menjadi bentuk data serial kemudian dipaketkan menjadi beberapa *frame*. Proses pembuatan *frame* ini, akan menambahkan *header* terhadap data yang hendak dikirimkan, yang mengandung alamat, pensinyalan, dan layanan yang digunakan. *Frame-frame* tersebut akan kemudian diubah menjadi pulsa-pulsa elekronik.

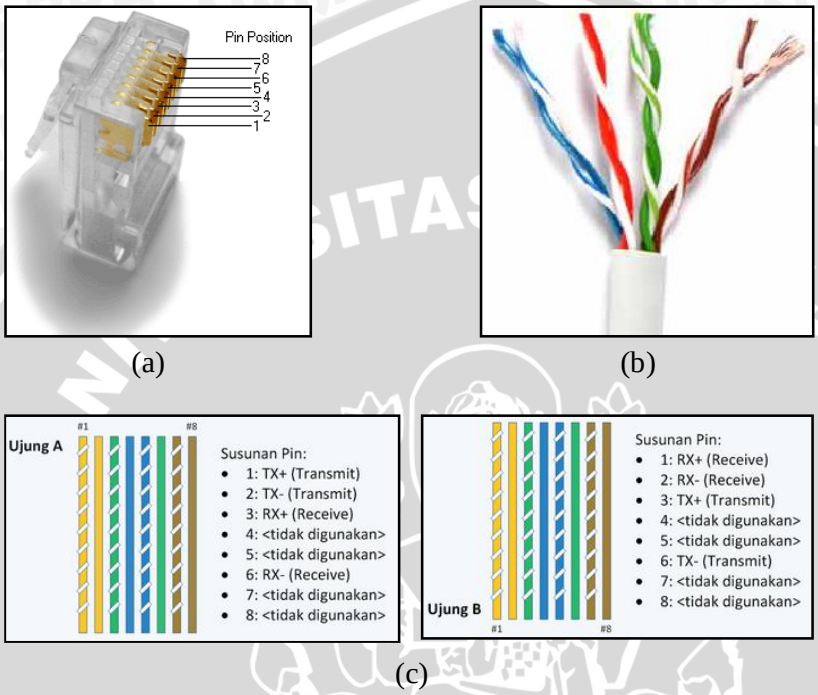


Gambar 2.13 *Network Interface Card*

Sumber : Perangkat Keras yang digunakan dalam perancangan sistem

c. Kabel UTP dan Konektor RJ-45

Kabel UTP dengan konektor RJ-45 merupakan perangkat keras yang berfungsi sebagai sistem pengkabelan pada jaringan *Video On Demand* yang digunakan secara umum dengan standart IEEE 802.3. Pada perancangan ini digunakan kabel UTP Cat.5 dengan konfigurasi *straight* ditunjukkan pada Gambar 2.14.



Gambar 2.14 (a) Konektor RJ-45 (b) Kabel UTP (c) Konfigurasi Kabel *Straight*
Sumber : Perangkat Keras yang digunakan dalam perancangan sistem

d. Komputer

Komputer adalah sistem elektronik untuk memanipulasi data dengan cepat dan tepat serta dirancang dan diorganisasikan agar secara otomatis menerima dan menyimpan data *input*, memrosesnya, dan menghasilkan output di bawah pengawasan suatu langkah-langkah instruksi program yang tersimpan di didalam penyimpannya (Sanderes, 1985). Pada proses perancangan komputer *server Virtual Private Network* memiliki beberapa fungsi diantaranya sebagai :

- Perangkat untuk menyediakan layanan *Virtual Private Network* dan *Video On Demand*, yang meliputi pembentukan proses *tunneling* menuju *client*, proses enkripsi, authentifikasi *client*, dan *server streaming*.

- Pada perangkat di sisi *server*, digunakan spesifikasi *hardware* yang lebih baik daripada perangkat komputer *client*. Server bertugas membentuk koneksi *Virtual Private Network* dan menyediakan layanan *streaming*.

Gambar 2.15 Komputer *Desktop*

Sumber : http://www.wisegeek.com/what-is-a-computer.htm

2.7 Parameter Performansi Jaringan

Menurut ITU-T E.800, QoS adalah : *"Sekumpulan efek performansi yang menentukan derajat kepuasan pengguna terhadap service yang diperikan oleh jaringan"*. Sedangkan dari sudut pandang jaringan telekomunikasi QoS adalah : *"Kemampuan suatu jaringan untuk menediakan layanan yang lebih baik pada trafik data tertentu pada berbagai jenis platform teknologi"* (Onno W. Purbo, 2001 : 125). Performansi sistem disini, merupakan ukuran baik buruknya suatu sistem dengan menggunakan parameter *Throughput*, *Delay* dan *Packet loss*.

a. *Throughput*

Throughput merupakan salah satu parameter yang menunjukkan kinerja dari suatu komunikasi data, yaitu menunjukkan jumlah data yang diterima dengan benar pada penerima setelah melewati media transmisi pada data *link layer* dari *client to client*. *Throughput* ditentukan dengan Persamaan (2.1) (Mischa Schwartz, 1987:129)

$$\lambda = \frac{1}{t_v} = \frac{(1-\rho)}{t_i\left[1+(\alpha-1)\rho\right]} \qquad (2.1)$$

Dimana :

$$\alpha = 1+\left(\frac{t_{out}}{t_i}\right) \qquad (2.2)$$

Keterangan:

λ = *Throughput* (paket/s)

t_v = waktu rata-rata untuk mentransmisikan 1 paket yang benar (s)

t_i = waktu yang dibutuhkan untuk mentransmisikan 1 paket (s)

t_{out} = waktu tunggu pada paket berikutnya (s)

α = konstanta transmisi ternormalisasi (tanpa satuan)

ρ = probabilitas *frame error* pada jaringan (tanpa satuan)

b. Kecepatan Transmisi Rata-Rata Diterima

Kecepatan transmisi rata-rata yang diterima merupakan jumlah bit yang dapat diterima di client setiap detik. Kecepatan transmisi rata-rata ditentukan dengan Persamaan (2-3) (Mischa Schwartz, 1987 : 132)

$$D = \lambda lx8 = \frac{(1-\rho)lx8}{t_i[1+(\alpha-1)\rho]} \quad (2.3)$$

Dengan :

D = Kecepatan transmisi rata-rata yang diterima (bps)

λ = *Throughput* (paket/s)

l = Panjang packet data (byte/packet)

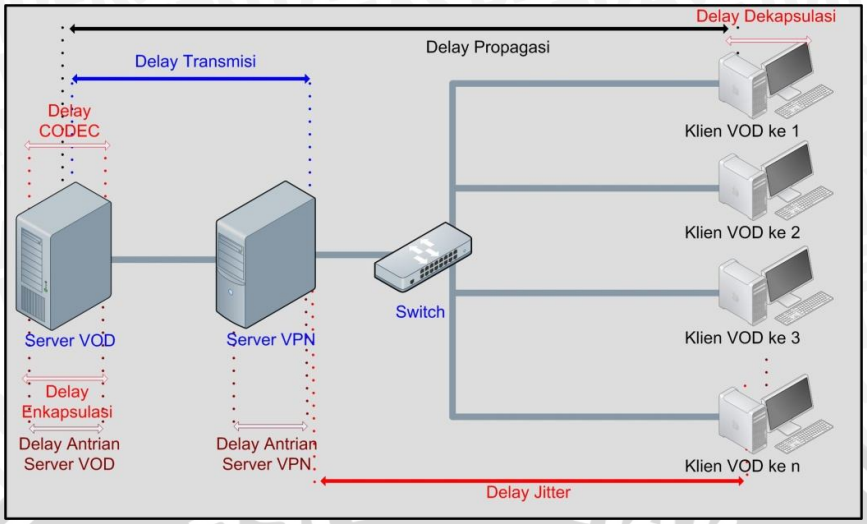
t_i = waktu yang dibutuhkan untuk mentransmisikan 1 paket (s)

α = konstanta transmisi ternormalisasi (tanpa satuan)

ρ = probabilitas *frame error* pada jaringan (tanpa satuan)

c. *Delay End To End*

Dalam perancangan jaringan, *delay* merupakan suatu permasalahan yang harus diperhitungkan karena kualitas suara bagus tidaknya tergantung dari waktu *delay*. Besarnya *delay* maksimum yang direkomendasikan oleh ITU-T G.1010 dapat dilihat pada Tabel 2.2. Delay maksimum pada aplikasi suara adalah 400 ms, sedangkan *Delay* maksimum pada aplikasi *video streaming* satu arah adalah 10s. *Delay* pada sistem secara umum disebut *delay end to end* berdasarkan Gambar 2.16 berikut :



Gambar 2.16 Komponen *Delay* pada Sistem

Sumber : Wallace, Kevin. 2009:56

Tabel 2.2 Standar *Delay* ITU-T G.1010 Untuk Aplikasi *Streaming*

Medium	***Application***	***Degree of Symmetry***	***One-way Delay***
Audio	*Conversational Voice*	*Two-way*	< 150ms *preferred* < 400ms *limit*
Audio	*Voice Messaging*	*One-way*	< 1s *for playback* < 2s *for record*
Audio	*High Quality Audio Streaming*	*One-way*	< 10s
Video	*Videophone*	*Two-way*	< 150ms *preferred* < 400ms *limit*
Video	*Streaming*	*One-way*	< 10s

Sumber : http://www.itu.int/rec/T-REC-G.1010-200111-I/en

Delay end-to end ditentukan dengan persamaan (2.4)

$$t_{end-to-end} = t_{codec} + t_{enc} + t_i + t_p + t_w + t_{dec} + t_{jitter} + t_{codec} \quad (2.4)$$

Dengan :

$t_{end-to-end}$ = *delay end-to-end* (ms)

t_{codec} = *delay codec* (ms)

t_{enc} = *delay* enkapsulasi (ms)

t_i = *delay* transmisi (ms)

t_p = *delay* propagasi (ms)

t_w = *delay* antrian (ms)

t_{jitter} = *delay jitter* (ms)

t_{dec} = *delay* dekapsulasi (ms)

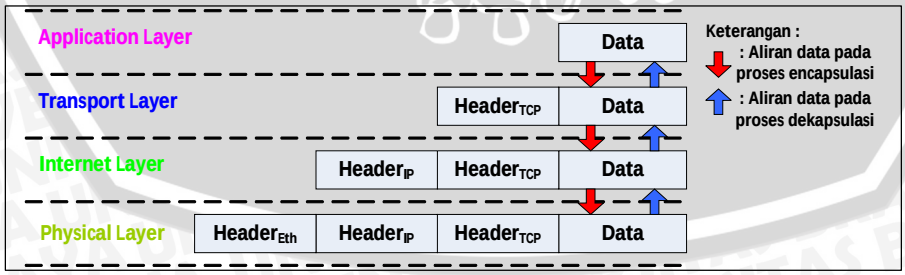
1. *Delay Codec*

Delay codec adalah waktu yang dibutuhkan aplikasi untuk melakukan proses *encoding* dan *decoding* pada paket data *multimedia* pada komputer. *Delay codec* dipengaruhi berbagai macam faktor seperti algoritma kompresi, rasio kompresi, proses sampling dan lain-lain. Setiap metode *encoding* menggunakan MPEG dengan *simple profiles* akan mengalami *delay* sebesar 50 ms (Vigato, 2005:52). Hal ini cocok untuk digunakan aplikasi yang menuntut *delay* rendah seperti *video conference.*

Delay codec pada MPEG relatif lebih besar daripada metode *encoding* lainya karena *encoding* MPEG ditujukan untuk kualitas Gambar yang lebih baik dengan *bitrate* yang rendah. Hal ini mengakibatkan algoritma kompresi MPEG yang lebih kompleks daripada metode algoritma kompresi yang lainnya.

2. *Delay* Enkapsulasi dan Dekapsulasi

Enkapsulasi adalah proses menambahkan *header* pada paket data, sehingga paket data tersebut dapat tepat sampai ke tujuan. Proses enkapsulasi ditunjukkan pada Gambar 2.17.



Gambar 2.17 Proses Enkapsulasi dan Dekapsulasi Paket Data

Sumber : http://www.citap.com/documents/tcp-ip/tcpip011.htm

Delay enkapsulasi adalah waktu yang dibutuhkan untuk menambahkan seluruh *header* pada sebuah paket data. Sedangkan *delay* dekapsulasi adalah waktu yang dibutuhkan untuk melepaskan seluruh *header* dari sebuah paket data. Pada saat pembentukan hubungan (*connection established*) antara *host* tujuan dengan *host* sumber terjadi kesepakatan dalam enkapsulasi dan dekapsulasi paket data.

Apabila *host* sumber ingin mengirim paket data ke *host* tujuan, maka proses yang terjadi adalah paket data tersebut dikirimkan ke *transport layer*. Pada *transport layer* dengan menggunakan protokol TCP, paket data dienkapsulasi menjadi segmen TCP dengan ditambahkannya *header* TCP.

Kemudian, segmen TCP kemudian diteruskan ke internet *layer*. Di internet *layer* dengan menggunakan *protocol* IP, segmen TCP dienkapsulasi menjadi *datagram* IP dengan ditambahkannya *header* IP.

Dari internet *layer*, datagram IP dikirimkan ke *network interface layer*. Di *network interface layer* dengan menggunakan *protocol Ethernet*, datagram IP dienkapsulasi menjadi *frame Ethernet* dengan ditambahkannya *header Ethernet*. Sehingga *delay* enkapsulasi secara matematis dapat ditulis berdasarkan persamaan berikut ini (Forouzan,2007) :

$$t_{enc} = \frac{L_{Header\ TCP} + L_{Header\ IP} + L_{Header\ Eth}}{C_{pros}} x8 \qquad (2.5)$$

Dengan C_{pros},

$$C_{pros} = \frac{N_k}{T_k} \qquad (2.6)$$

Keterangan:

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

$L_{Header\ TCP}$ = panjang *header* TCP (*byte*/paket)

$L_{Header\ IP}$ = panjang *header* IP (*byte*/paket)

$L_{Header\ Eth}$ = panjang *header Ethernet* (*byte*/paket)

C_{pros} = kecepatan pemrosesan pada terminal pengirim (bps)

N_k = jumlah total data yang dikirimkan (bit)

T_k = waktu pengiriman total data (s)

Berdasarkan pengertian diatas maka *delay* dekapsulasi ditentukan dengan persamaan berikut ini:

$$t_{dec} = \frac{L_{Header\,TCP} + L_{Header\,IP} + L_{Header\,Eth}}{C_{pros2}} x8 \quad (2.7)$$

Dengan C_{pros2},

$$C_{pros2} = \frac{N_t}{T_t} \quad (2.8)$$

Keterangan:

t_{enc} = *delay* dekapsulasi (s)
$L_{Header\ TCP}$ = panjang *header* TCP (*byte*/paket)
$L_{Header\ IP}$ = panjang *header* IP (*byte*/paket)
$L_{Header\ Eth}$ = panjang *header Ethernet* (*byte*/paket)
C_{pros} = kecepatan pemrosesan pada terminal penerima (bps)
N_t = jumlah total data yang diterima (bit)
T_t = waktu penerimaan total data (s)

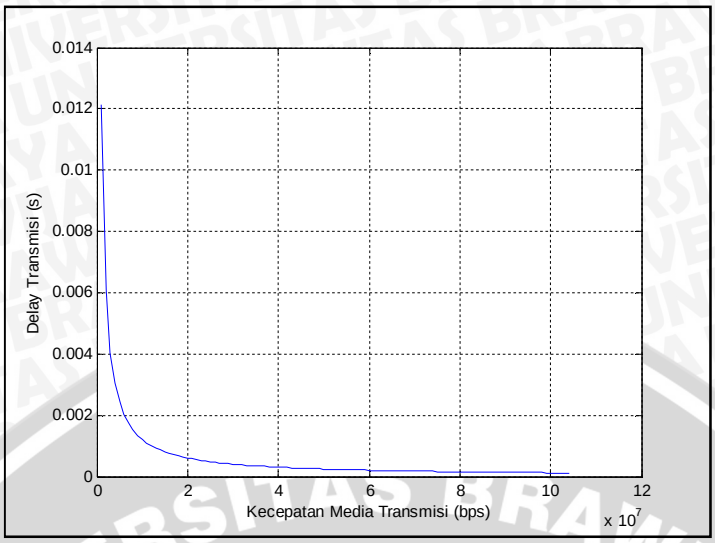
3. *Delay* Transmisi

Delay transmisi adalah waktu yang dibutuhkan untuk meletakkan sebuah paket data ke media transmisi. *Delay* transmisi ini dipengaruhi oleh ukuran paket data dan kapasitas kanal intranet (Mischa Schwartz, 1987). *Delay* transmisi ditentukan dengan Persamaan (2.9)

$$t_t = \frac{(L + L')}{C} \times 8 = \frac{W_{frame}}{C} \times 8 \quad (2.9)$$

Keterangan:

t_t = *delay* transmisi (s)
L = panjang paket data (*byte*/paket)
L' = panjang *header* (*byte*/paket)
C = kecepatan media transmisi (bps)
W_{frame} = panjang *frame Ethernet* (*byte*/paket)



Gambar 2.18 Grafik Perbandingan *Delay* Transmisi terhadap Kecepatan Media Transmisi

Sumber : Simulasi

Gambar 2.18 menunjukkan grafik perbandingan *delay* transmisi terhadap kecepatan media transmisi berdasarkan persamaan (2.9). Nilai W_{frame} *frame Ethernet* sebesar 1518 byte (Garcia, 2001:402), dengan kecepatan media transmisi mulai dari 1 mbps hingga 100 mbps. Mengacu pada Gambar 2.18, dapat disimpulkan nilai *delay* transmisi berbanding lurus dengan kecepatan media transmisi.

4. *Delay* Propagasi

Delay propagasi *(propagation delay)* adalah waktu yang dibutuhkan untuk merambatkan paket data melalui media transmisi UTP *(unshield twisted pair)* dari *host* sumber ke *host* tujuan. *Delay* ini ditentukan oleh karakteristik jarak antara sumber dan tujuan, serta media transmisi yang digunakan untuk pengiriman informasi. *Delay* propagasi untuk standar IEEE 802.3i dapat dihitung dengan rumus berikut (Blanchard, 2001) :

$$t_p = \frac{L}{V_{prop}} \qquad (2.10)$$

Keterangan:

t_p = *Delay* propagasi (s)

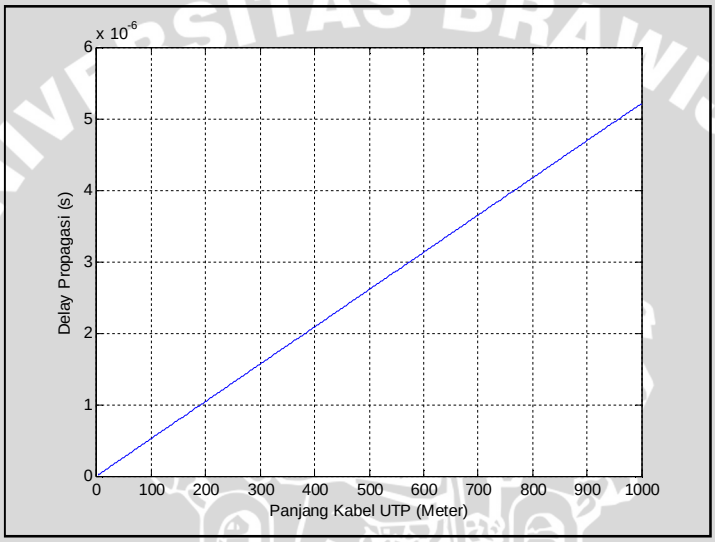
L = Panjang Kabel (m)

V_{prop} = Kecepatan sinyal melalui kabel UTP (m/s)

Tabel 2.3 Kecepatan Propagasi Pada Berbagai Media

Jenis Kabel	Kecepatan Propagasi	Keterangan
Thick Coax	.77c	c merupakan kecepatan cahaya = 3.108 m/s
Thin Coax	.65c	
Twisted Pair	.59c	
Fiber	.66c	
UTP Cable	.64c	

Sumber : http://stason.org/TULARC/networking/lans-ethernet/3-11-What-is-propagation-*delay*-Ethernet-Physical-Layer.html



Gambar 2.19 Grafik Perbandingan Panjang kabel UTP terhadap *Delay* Propagasi

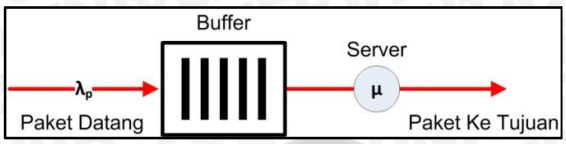
Sumber : Simulasi

Gambar 2.19 menunjukkan grafik perbandingan panjang kabel UTP terhadap *delay* propagasi berdasarkan persamaan (2.10). panjang kabel UTP bervariasi, mulai dari 1 meter sampai dengan 1000 meter. Mengacu pada Gambar 2.14, dapat disimpulkan nilai *delay* propagasi berbanding lurus dengan panjang kabel UTP.

5. *Delay* Antrian

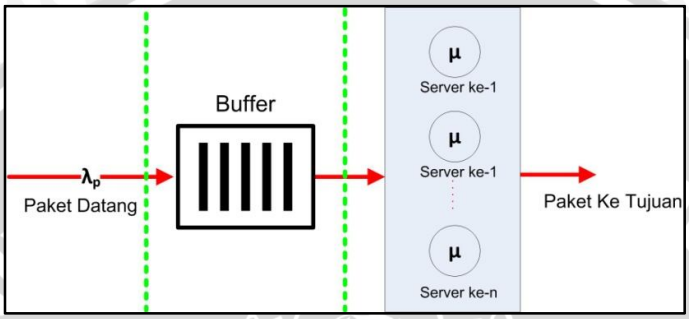
Delay antrian adalah waktu di mana paket data berada dalam antrian untuk diproses oleh *server*. *Delay* antrian dihitung dengan menggunakan model antrian M/M/1. M pertama menunjukkan distribusi kedangan *Poisson,* M kedua menunjukkan bahwa jumlah *server* adalah tunggal yang diwakili oleh sebuah *node* yang dilengkapi

buffer . Disiplin antrian yang digunakan yaitu FIFO (*First In First Out*), seperti yang terlihat pada Gambar 2.20.



Gambar 2.20 Model Antrian M/M/1

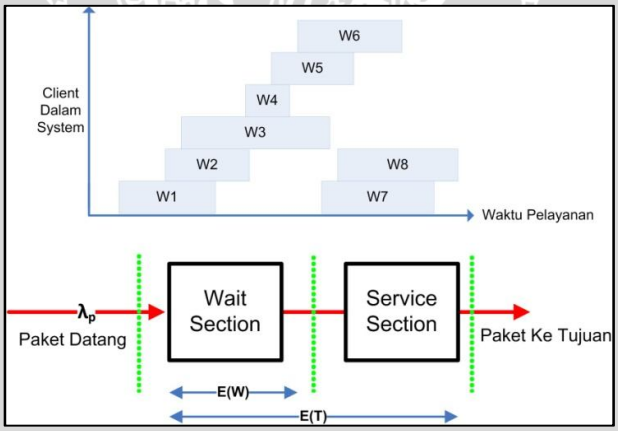
Sumber: Mischa Schwartz, 1987 :22



Gambar 2.21 Model Antrian M/M/S

Sumber: Mischa Schwartz, 1987 :22

Pemodelan antrian yang digunakan dalam perhitungan *delay* antrian sistem menerapkan *Little's Formula* berdasarkan Gambar 2.21 berikut :



Gambar 2.22 Penerapan *Little's Formula* Pada Sistem Antrian

Sumber : Mischa Schwartz, 1987 : 42

Pada Gambar 2.21, diatas jika *E(W)* waktu antrian atau tunggu sebelum dilakukan pelayanan pada sisi *server* dan *E(T)* merupakan waktu paket dalam sistem atau sering disebut sebagai *delay* antrian. Jika kecepatan kedatangan paket (λ) dinyatakan dengan :

$$\lambda_p = \frac{N}{T} \tag{2.11}$$

Dengan :

λ_p = kecepatan kedatangan paket pada *server* (paket/s)

N = total paket yang dikirim (paket)

T = waktu pengiriman paket total (s)

Sedangkan kecepatan pelayanan ditentukan dengan persamaan berikut ini (Mischa Schwartz, 1987 : 23)

$$\mu = \frac{C}{L_t} \tag{2.12}$$

Keterangan:

μ = kecepatan pelayanan *server* (paket/s)

C = kapasitas kanal (bps)

L_t = panjang paket data (bit/paket)

Berdasarkan Gambar 2.26 maka *delay* antrian rata-rata E(T) dapat ditentukan dengan persamaan (Schwartz,Mischa. 1987:42)

$$E(T) = E(W) + 1/\mu \tag{2.13}$$

Dimana,

$$E(W) = \frac{1}{\mu - \lambda_p} - \frac{1}{\mu} \tag{2.14}$$

Sehingga besarnya *delay* antrian *E(T)* dapat dinyatakan :

$$E(T) = \frac{1}{\mu - \lambda_p} \tag{2.15}$$

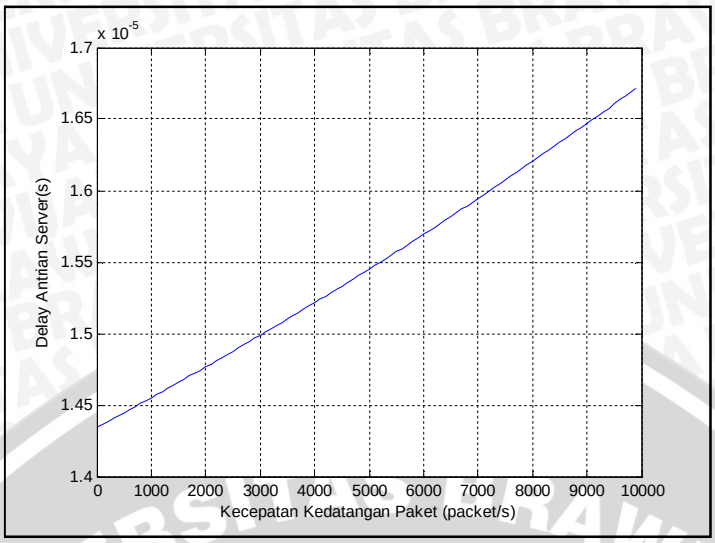
Keterangan:

E(T) = *delay* antrian pada *server* (s)

E(W) = *delay* tunggu paket pada *server* (s)

λ_p = kecepatan kedatangan paket pada *server* (paket/s)

μ = kecepatan pelayanan *server* (paket/s)



Gambar 2.23 Grafik Perbandingan λ_p terhadap *E(W)*

Sumber : Simulasi

Gambar 2.23 menunjukkan grafik perbandingan *delay* antrian terhadap *E(W)* berdasarkan persamaan (2.15). Dapat disimpulkan berdasarkan Gambar 2.17, nilai delay antrian berbanding lurus dengan besar nilai kecepatan kedatangan paket.

6. *Delay Jitter*

Delay jitter disebabkan oleh kedatangan paket yang acak karena setiap paket melewati jalur yang berbeda-beda pada jaringan (Mischa Schwartz, 1987). *Delay jitter* rata-rata ditentukan dengan persamaan (2.16).

$$\theta = \frac{1}{N_{packet} - 1} \sum_{k=0}^{N_{packet}-1} t_{end-to-end(n+1)} - t_{end-to-end(n)}$$

$$= \frac{t_v}{N_{packet}-1} \qquad (2.16)$$

Dengan :

θ = *delay jitter* rata-rata (s)

t_v = waktu transmisi (s)

$t_{end-to-end}$ = *delay end to end* pada urutan ke-n (s)

$t_{end-to-end(n+1)}$ = *delay end to end* pada urutan ke-(n+1) (s)

N_{packet} = jumlah paket *multimedia* yang diterima

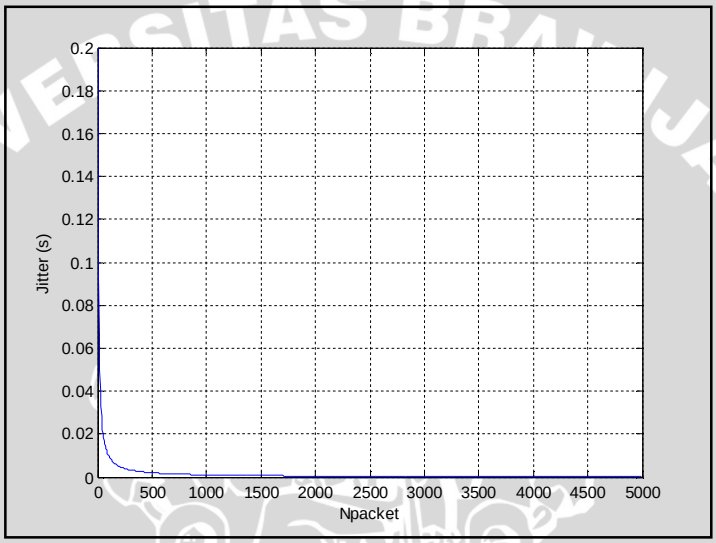
Aplikasi *video streaming* harus mempunyai nilai jitter yang sesuai dengan batas persepsi manusia. untuk nilai *jitter* pada *video* kualitas *High Definition* tidak lebih dari

50ms, pada aplikasi *video broadcasting* atau *on demand* kurang dari 100ms, dan pada aplikasi *videoconferencing* harus kurang dari 400ms (Fluckiger, 1995).

Tabel 2.4 Standar *Jitter* Untuk Aplikasi *Streaming*

Aplikasi	Jitter Maksimum
Video HD	50ms
Broadcast dan *On Demand*	100ms
Videoconferencing	400ms

Sumber : Fluckiger, 1995



Gambar 2.24 Grafik Perbandingan *Jitter* terhadap N*packet*

Sumber : Simulasi

Gambar 2.24 menunjukkan grafik perbandingan *jitter* terhadap jumlah paket yang hilang berdasarkan persamaan (2.16) dengan waktu transmisi data 60 detik. Jumlah N*packet* yang hilang bervariasi, mulai dari 1 paket sampai dengan 5000 paket. Mengacu pada Gambar 2.17, semakin besar nilai N*packet* maka semakin kecil nilai *jitter* yang didapatkan.

d. *Packet loss*

Packet loss adalah jumlah paket data hilang yang dikirimkan *host* sumber. Hal ini dapat terjadi karena berbagai hal, mulai dari *layer* fisik seperti degradasi sinyal, tumbukan antar paket dalam perangkat jaringan, hingga pada *layer* aplikasi seperti kesalahan *software*. Pada *Streaming*, ketika *Bandwidth Multimedia* lebih besar daripada kecepatan transmisi rata-rata yang diterima maka akan terjadi kongesti sehingga

menyebabkan beberapa paket hilang. Tabel 2.5 menunjukkan Standar *Packet Loss* ITU-T G.1010 Untuk Aplikasi *Streaming*.

Tabel 2.5 Standar *Packet Loss* ITU-T G.1010 Untuk Aplikasi *Streaming*

Medium	***Application***	***Degree of Symmetry***	***Information Loss***
Audio	*Conversational Voice*	*Two-way*	< 3% Packet Loss Ratio (PLR)
Audio	*Voice Messaging*	*One-way*	< 3% PLR
Audio	*High Quality Audio Streaming*	*One-way*	< 1 % PLR
Video	*Videophone*	*Two-way*	< 1 % PLR
Video	*Streaming*	*One-way*	< 1 % PLR

Sumber : http://www.itu.int/rec/T-REC-G.1010-200111-I/en

Prosentase *Packet loss* ditentukan dengan Persamaan berikut (Mischa Schwartz, 1987)

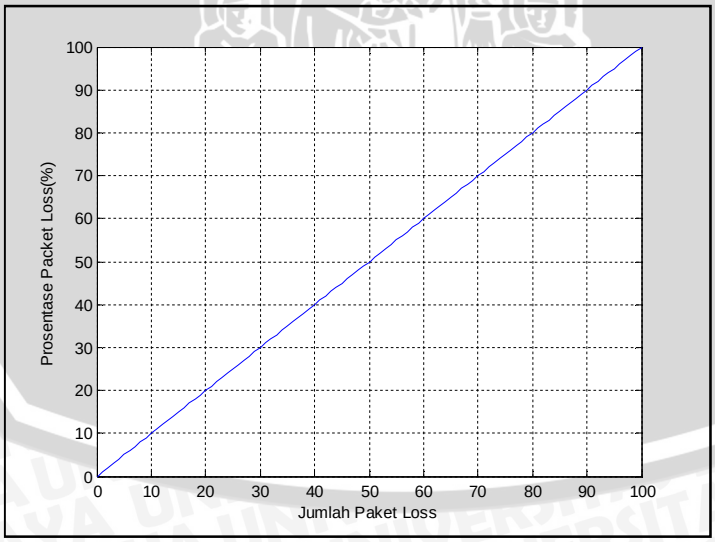
$$Packet\ Loss = \frac{N_{packet\ loss}}{N_{packet} - N_{packet\ loss}} \times 100\% \qquad (2.17)$$

Keterangan :

Packet Loss = prosentase *Packet Loss*

$N_{Packet\ loss}$ = jumlah paket multimedia yang hilang (paket)

N_{paket} = jumlah paket multimedia yang diterima dengan benar (paket)



Gambar 2.25 Grafik Perbandingan Prosentase *Packet Loss* terhadap Paket yang Hilang

Sumber : Simulasi

Gambar 2.25 menunjukkan grafik perbandingan prosentase *packet loss* terhadap jumlah paket yang hilang berdasarkan persamaan (2.17). Jumlah paket yang hilang bervariasi, mulai dari 1 paket sampai dengan 100 paket. Dapat disimpulkan prosentase *paket loss* berbanding lurus dengan jumlah paket yang hilang.