BAB II
TINJAUAN PUSTAKA

2.1 Secara Umum

Tinjauan pustaka ini bersifat akademik dengan tujuan untuk menyelesaikan permasalahan. Teori yang dibahas adalah tentang *monitoring* jaringan, cara kerja *port mirroring*, sistem kerja *switch*, layanan *Video on Demand* (VoD), parameter performansi jaringan yang terdiri dari *throughput*, *delay* dan *packet loss* serta perangkat lunak dan perangkat keras jaringan.

2.2 *Monitoring* Jaringan

Monitoring jaringan merupakan salah satu fungsi dari manajemen jaringan, yang melakukan proses pengumpulan informasi trafik jaringan (Wong, 2000) untuk kemudian dilakukan analisis atas informasi tersebut. Perkembangan dan penggunaan *internet* yang semakin meningkat, mengakibatkan jaringan semakin bertambah besar dan kompleks. Keadaan tersebut membuat aplikasi *monitoring* jaringan perlu digunakan untuk mengecek kondisi jaringan secara efektif sehingga aplikasi manajemen jaringan dapat mengontrol jaringan, agar tetap ekonomis namun dengan kualitas jaringan yang tinggi untuk pengguna. Adapun tiga tujuan *monitoring* jaringan yaitu sebagai berikut (Wong, 2000):

1. *Monitoring* performansi
2. *Monitoring* kesalahan
3. *Monitoring* akun

Monitoring performansi berhubungan dengan mengukur performansi jaringan. Terdapat tiga hal penting dalam *monitoring* performansi. Pertama, informasi *monitoring* performansi biasanya digunakan untuk merencanakan perkembangan jaringan kedepan dan menemukan masalah penggunaan jaringan. Kedua, *time frame* pada *monitoring* performansi harus cukup panjang untuk membangun suatu perilaku jaringan. Ketiga, memilih pengukuran yang penting dan dibutuhkan, karena terdapat banyak hal yang dapat diukur dalam jaringan.

Monitoring kesalahan berhubungan dengan mengukur masalah yang ada dalam jaringan. Terdapat dua permasalahan penting dalam *monitoring* kesalahan. Pertama, *monitoring* kesalahan berhubungan dengan beberapa *layer* jaringan. Permasalahan dapat terjadi di *layer* jaringan yang berbeda. Dengan demikian, penting untuk mengetahui pada *layer* mana yang memiliki permasalahan. Kedua, *monitoring* kesalahan perlu membangun

karakteristik jaringan yang normal dalam jangka waktu yang lama. Selalu ada *error* dalam jaringan tetapi saat terjadi *error*, bukan berarti jaringan memiliki permasalahan secara terus menerus. Sebagai contoh, *noise* pada *link* transmisi dapat menyebabkan *error* transmisi. Jaringan hanya memiliki masalah saat jumlah *error* tiba-tiba meningkat diatas keadaan normal. Untuk itu, catatan jaringan saat keadaan normal sangatlah penting.

Monitoring akun berhubungan dengan bagaimana pengguna menggunakan jaringan. Jaringan menyimpan daftar perangkat yang digunakan pengguna dan bagaimana perangkat tersebut digunakan. Informasi jenis ini digunakan untuk menghitung penggunaaan jaringan oleh pengguna, dan untuk memprediksi penggunaaan jaringan kedepan.

Setelah mengetahui tujuan *monitoring* jaringan, maka selanjutnya dapat memilih teknik yang tepat untuk melakukan *monitoring* jaringan. Pada penelitian ini akan menggunakan teknik *monitoring* jaringan pasif.

2.3 *Monitoring* Jaringan Pasif

Monitoring jaringan pasif dapat dideskripsikan sebagai suatu proses analisis trafik paket jaringan, utilisasi *link*, dan kerentanan layanan jaringan serta mendeteksi aktifitas yang sedang dilakukan jaringan dan melakukan *capture* secara *real time* (Mickevičiūtė *et al*, 2013). *Monitoring* jaringan pasif dapat melakukan *capture* data trafik jaringan dengan dua cara, yaitu dengan *Port mirroring* dan TAP (*Test Access Point*). Pada penelitian ini *monitoring* jaringan menggunakan *Port Mirroring*.

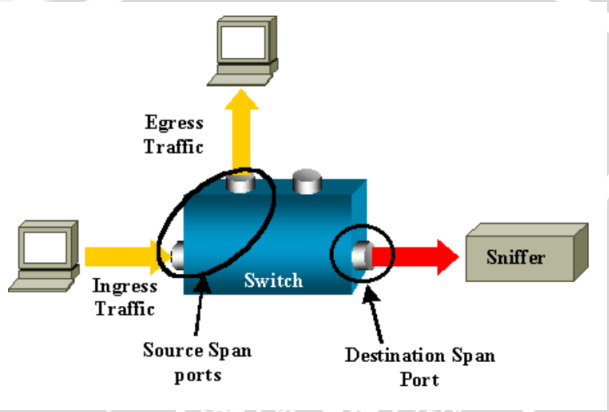
2.3.1 *Port Mirroring*

Port mirroring berfungsi untuk melakukan proses *monitoring* trafik jaringan dengan melakukan *capture* trafik jaringan. Implementasi *Port mirroring* mudah dilakukan, karena *Port mirroring* merupakan fitur yang tersedia pada *Switch Manageable*. Jika menggunakan *switch* biasa, trafik jaringan hanya dapat terlihat oleh komputer yang sedang berkomunikasi saja sedangkan komputer lain tidak dapat melihat trafik, karena bukan merupakan tujuan dari trafik tersebut. Berbeda jika pada *switch* yang dikonfigurasikan *port mirroring* memungkinkan komputer lain untuk dapat melihat trafik paket jaringan yang terjadi meskipun bukan merupakan tujuan trafik.

Port mirroring memungkinkan *switch* mengirimkan salinan (duplikasi) trafik paket jaringan yang terlihat pada salah satu *port switch* atau lebih yang sedang berkomunikasi (sebagai *source port*) ke *port* yang diinginkan (sebagai *destination port*). Pada *destination port* tersebut dihubungkan ke *Network Protocol Analyzer* untuk proses *capture* trafik. Di

dalam proses konfigurasi *port mirroring*, terdapat beberapa istilah seperti yang ditunjukkan pada Gambar 2.1 yaitu sebagai berikut:

- *Ingress traffic*: trafik yang masuk ke *switch*.
- *Egress traffic*: trafik yang keluar dari *switch.*
- *Source port*: *port* yang diamati.
- *Destination port*: *port* yang melakukan *monitoring* terhadap *source port* dan dihubungkan ke *Network Analyzer*.

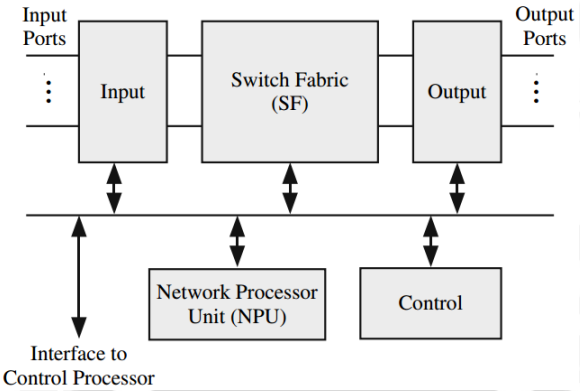


Gambar 2.1 Terminologi pada *Port Mirroring*
(Sumber: www.cisco.com)

Penerapan *Port Mirroring* memiliki beberapa keuntungan, diantaranya yaitu dapat melihat trafik yang terjadi dalam *switch* dengan tidak memerlukan perangkat tambahan hanya membutuhkan *switch manageable* yang mendukung *fitur port mirroring* sehingga lebih ekonomis jika diandingkan dengan TAP (*Test Access Point*).

2.4 Switch

Switch merupakan perangkat keras yang memiliki kemampuan untuk menghubungkan dua atau lebih perangkat yang lain di dalam jaringan komputer. *Switch* menerima paket sebagai *input* dan merutekan paket tersebut ke *output* menurut informasi *routing* pada *header* paket dan tabel *routing switch*. Dibandingkan dengan *hub*, *switch* memiliki kemampuan untuk membaca dan menyimpan alamat fisik (MAC *Address*) dan nomor *port switch* yang digunakan oleh setiap komputer yang terhubung ke dalam *switch* yang bersangkutan. *Switch* juga memiliki kemampuan untuk melakukan filter terhadap paket data yang keluar masuk *switch*.



Gambar 2.2 Komponen Dasar *Switch*
(Sumber: Gebali, 2008)

Pada Gambar 2.2 ditunjukkan komponen dasar pada *switch*. *Switch* memiliki empat komponen dasar, yaitu *Network Processor Unit* (NPU), *controller*, *port input/output*, dan *switch fabric*. Diantara komponen tersebut, komponen *switch* yang memiliki pengaruh besar terhadap performansi yaitu *storage buffer/queue* (antrian) dan *switch fabric* (Gebali, 2008). Performansi tersebut seperti *packet loss* yang dihasilkan oleh *switch* akibat *buffer* yang *overflow* atau *switch* tidak mampu membangun jalur dari *port input* ke *port output* melalui *switch fabric*. Dan untuk *line rate* tergantung pada *memory access speed*.

2.4.1 *Network Processing Unit* (NPU)

Network Processing Unit (NPU) dibutuhkan untuk melakukan *compute-intensive* sehingga *switch* mampu memproses data dengan kecepatan tinggi. NPU merupakan komponen perangkat keras yang dirancang untuk melakukan tugas dalam jaringan secara efisien. Berikut beberapa tugas NPU:

1. Menerapkan protokol keamanan.
2. Menerapkan protokol *Quality of Service* (QoS)
3. *Traffic Shapping*
4. Merutekan paket menggunakan *lookup* tabel *routing*.

2.4.2 *Controller*

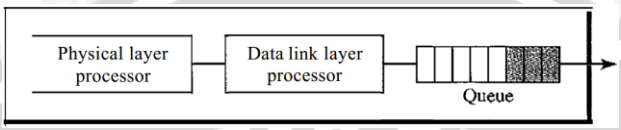
Controller melakukan beberapa fungsi sebagai berikut:

1. Mengatur isi tabel *routing*, dimana menentukan *output* tujuan yang sesuai dengan paket yang datang.
2. Memberikan *congestion control* untuk *switch* dengan melakukan *monitoring switch* secara kontinyu.
3. Mengatur *switch* membangun hubungan berdasarkan kelas layanan.

2.4.3 Data *Path* (Port *Input*/*Output*)

a. Port Input

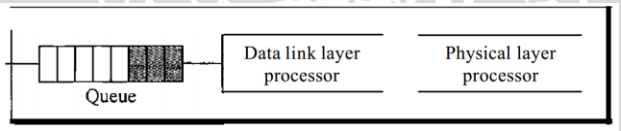
Port input terdiri *physical layer processor*, *data link layer processor*, dan *queue* seperti pada Gambar 2.3. Bit-bit dibentuk dari sinyal yang diterima. Paket didekapsulasi dari *frame*. *Error* dideteksi dan dikoreksi. Kemudian paket siap untuk dirutekan oleh *network layer*.Pada *port input* juga terdapat *buffer* (*queue*) untuk menahan paket sebelum menuju ke *switch fabric*.



Gambar 2.3 *Port Input*
(Sumber: Forouzan, 2007)

b. Port Output

Port output melakukan fungsi yang sama seperti *port input*, tetapi secara terbalik. Pertama, paket yang akan keluar terlebih dahulu berada di *buffer* (*queue*), kemudian paket dienkapsulasi ke dalam frame, dan terakhir fungsi *physical layer* diterapkan ke *frame* untuk menghasilkan sinyal yang akan dikirim. Gambar 2.4 merupakan diagram skematik *port output*.



Gambar 2.4 *Port Output*
(Sumber: Forouzan, 2007)

2.4.4 *Switch Fabric*

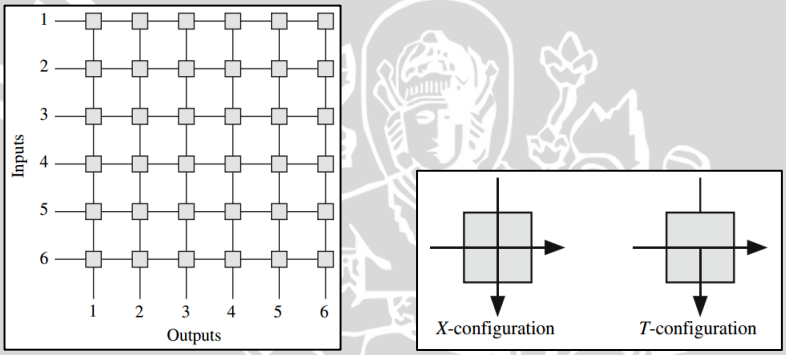
Switch fabric terdiri dari jaringan transmisi yang bersifat pasif dan elemen *switching* yang melakukan fungsi *internal routing* yaitu menetapkan jalur yang dibutuhkan paket untuk menghubungkan *port input* dengan *port output*. *Swtich Fabric* juga harus mampu melakukan berbagai tipe hubungan seperti *single cast* (satu *input* ke satu *output*), *multicast* (satu *input* ke beberapa *output*), *broadcast* (satu *input* ke semua *output*), atau *hot point* (beberapa *input* ke satu *output*). Kecepatan proses *switch fabric* mempengaruhi ukuran *input/output queue* dan juga *delay* pengiriman paket.

Terdapat beberapa macam *switch fabric* yang sering digunakan sebagai berikut (Forouzan, 2007):

a. *Crossbar Switch*

Crossbar Switch merupakan tipe *switching fabric* yang paling sederhana. *Crossbar switch* menghubungkan *n input* ke *m output* dalam sebuah *grid* menggunakan *electronic microswitches* (*transistor*) pada tiap *crosspoint* seperti yang ditunjukkan Gambar 2.5 (a). Keterbatasan utama dari tipe ini adalah jumlah *crosspoint* yang dibutuhkan. Untuk menghubungkan *n input* ke *m output* menggunakan tipe *crossbar switch* membutuhkan *n x m crosspoint*.

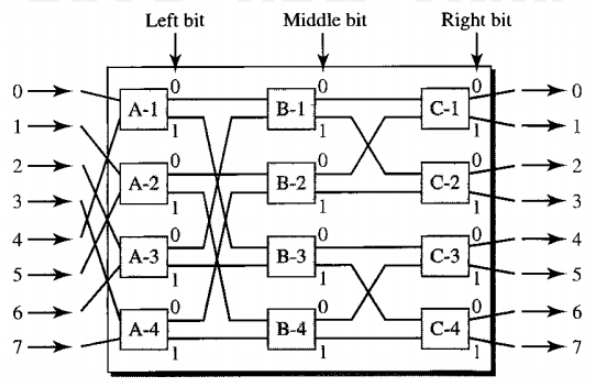
Tiap *crosspoint* bekerja dengan dua konfigurasi seperti pada Gambar 2.5 (b). Pada konfigurasi X, setelah data lewat secara horisontal dari input 3 dan dikirim ke seluruh titik potong pada baris yang sama. Data mengalir di kolom 5 secara vertikal yang dapat memiliki input dari mana saja. Pada konfigurasi T, *crosspoint* mengalirkan data secara horisontal dan mengganggu aliran data yang mengalir secara vertikal.



Gambar 2.5 (a) *Interconnection* Jaringan Pada *Crossbar Switch* (b) Konfigurasi *Crosspoint* Pada *Crossbar Switch*
(Sumber: Gebali, 2008)

b. *Banyan Switch*

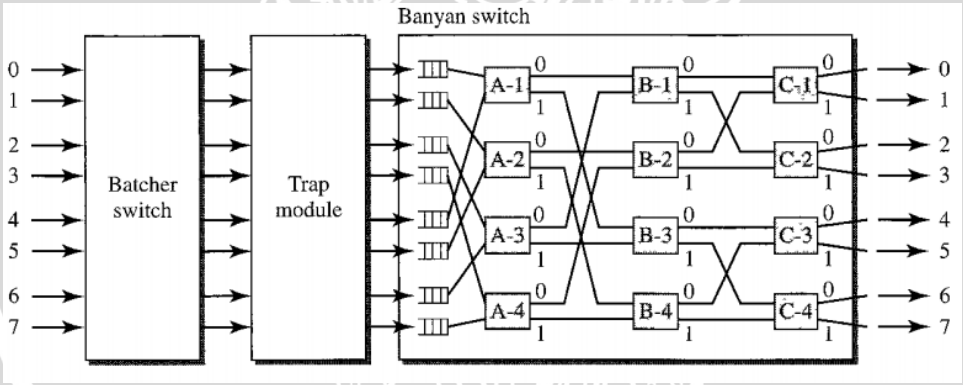
Banyan Switch merupakan *multistage switch* dengan *microswitches* pada tiap *stage* yang merutekan paket berdasarkan port *output* yang dipresentasikan dengen deret biner. Untuk *n input* dan *n output*, jumlah *stage* yang diperlukan sebanyak $\boldsymbol{log_2 n}$ *stage* dengan jumlah *microswitch* $\boldsymbol{n/2}$ pada tiap *stage*. Mekanisme kerja dari tiap *stage* pada *Banyan* Switch, yaitu *stage* pertama merutekan paket berdasarkan dari bit paling tinggi (dari bit paling kiri). *Stage* kedua merutekan paket berdasarkan dari bit tinggi kedua (dari kiri ke kanan), dan seterusnya. Gambar 2.6 menunjukkan *Banyan Switch* dengan 8 *input* dan 8 *output*.



Gambar 2.6 *Banyan Switch*
(Sumber: Forouzan, 2007)

c. *Batcher-Banyan Switch*

K.E. Batcher mendesain suatu *switch* yang mampu melakukan sortir terhadap paket yang datang menurut tujuan dari paket tersebut dan kemudian dikombinasikan dengan *Banyan Switch*. Gambar 2.7 menunjukkan desain tipe *Batcher-Banyan Switch*, terdapat *Trap Module* yang berada diantara *Batcher Switch* dan *Banyan Switch*. *Trap Module* mencegah adanya beberapa paket dengan tujuan yang sama dikirim dalam waktu bersamaan. Paket akan menunggu untuk dikirimkan satu per satu.



Gambar 2.7 *Batcher-Banyan Switch*
(Sumber: Forouzan, 2007)

2.5 Fungsi *Switch*

Switch melakukan beberapa fungsi selain merutekan paket dari *input* ke *output* yaitu sebagai berikut:

2.5.1 *Routing*

Switch harus mampu membaca *header* pada masing-masing paket yang datang untuk menentukan *link* keluaran yang harus digunakan untuk mengirimkan paket ke tujuannya. Hal tersebut menjelaskan bahwa paket yang datang tidak bisa dirutekan hingga proses klasifikasi paket berdasarkan informasi *header* telah dilakukan. Penentuan *routing* dilakukan dengan menggunakan tabel *routing switch*.

2.5.2 Manajemen Trafik

Congestion akan terjadi jika terdapat banyak paket dalam suatu jaringan. Saat *congestion* terjadi, *buffer* menjadi penuh dan mulai kehilangan paket. Kondisi ini menjadi buruk karena penerima akan mengirimkan *negative acknowledgment* dan pengirim akan mulai mentransmisikan kembali *frame* yang hilang. Hal tersebut menyebabkan paket dalam jaringan akan semakin bertambah dan performansi semakin memburuk.

2.5.3 *Scheduling*

S*cheduling* digunakan untuk dua alasan. Pertama, untuk memberikan *Quality of Service* (QoS) yang berbeda untuk tipe pengguna yang berbeda. Kedua, untuk melindungi pengguna dari pengguna yang lain yang mungkin dapat memonopoli *bandwidth* dan *buffer space*. *Scheduling* juga dapat mengumpulkan beberapa pengguna kedalam kelas *broad service* untuk mengurangi beban kerja. Mekanisme kerja pada *scheduling* ini adalah paket data yang akan dikirim dari *storage* buffer/*queue* ke tujuan ditentukan penjadwalan pengirimannya.

2.5.4 *Congestion Control*

Switch melakukan fungsinya sebagai *congestion control* dengan s*witch* membuang paket untuk mengurangi *congestion* jaringan dan meningkatkan efisiensi. *Switch* harus memilih paket mana untuk dibuang saat sistem kelebihan beban. Terdapat beberapa pilihan seperti membuang paket yang datang setelah *buffer* mencapai tingkat *occupancy* tertentu. Pilihan yang lain adalah membuang paket dari mana saja yang berada di *buffer* berdasarkan pada *tag* atau prioritasnya.

2.6 Pengukuran Performansi *Switch*

Beberapa *switch* dirancang dengan tujuan tertentu, namun yang perlu diperhatikan dalam merancang *switch* yaitu implikasi terhadap performansi *switch*. Parameter pengukuran performansi *switch*, diantaranya:

1. *Data rate* maksimum pada *input*.
2. Jumlah *port input* dan *output*.
3. Jumlah *independent logical channel* yang dapat mendukung.
4. Rata-rata *delay* paket saat berada di *switch*.
5. Rata-rata *packet loss* yang terjadi di *switch*.
6. *Quality of Service* (QoS), yang terdiri dari *data rate*, *delay* paket dan *packet loss*.
7. *Multicast* dan *broadcast*.

8. Skalabilitas *switch*, yaitu kemampuan *switch* untuk berkerja dengan maksimal jika jumlah *input* dan *output* meningkat.
9. Kapasitas *switch*
10. Fleksibilitas arsitektur *switch* yaitu komponen *switch* dapat diperbaharui.

2.7 Klasifikasi *Switch*

Terdapat dua komponen penting dalam *switch*, yaitu *buffer* dan *switching fabric*. Sehingga dapat diuraikan arsitektur *switch* berdasarkan dua kriteria berikut ini:

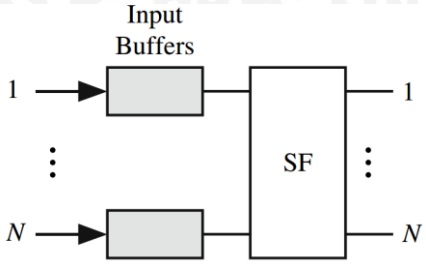
- Tipe *switch fabric* (SF), digunakan *switch* untuk merutekan paket dari *switch input* (*ingress*) ke *switch output* (*egress*).
- Lokasi *buffer* dan *queue* dalam *switch*. Karakteristik penting pada *switch* adalah jumlah dan lokasi *buffer* yang digunakan untuk menyimpan trafik yang masuk untuk diproses. Penempatan *buffer* dan *queue* di dalam *switch* sangat penting karena dapat mempengaruhi pengukuran performansi *switch* seperti *packet loss*, kecepatan¸ kemampuan untuk mendukung layanan yang berbeda-beda, dan lain-lain.

2.8 *Input Queuing Switch*

Gambar 2.8 menunjukkan *input queuing switch*, dimana tiap *port input* memiliki antrian (*queue*) atau *buffer* FIFO (*First Input First Output*) untuk menyimpan sementara paket yang datang sebelum diteruskan ke *port output* melalui *switch fabric*. Paket tersebut disimpan pada bagian akhir antrian (*queue*) dan hanya naik saat paket yang berada di bagian atas antrian (*queue*) telah dirutekan melalui *switch fabric* ke *port output* yang tepat. Sebuah *controller* pada tiap *port input* mengklasifikasi setiap paket dengan memeriksa *header* paket untuk menentukan jalur yang tepat melalui *switch fabric*. *Controller* juga harus melakukan fungsi manajemen trafik.

Selanjutnya, sebuah *input queue* harus mampu melakukan proses menulis dan membaca sehingga tidak terjadi *bottleneck* kecepatan *memory access time*. Diasumsikan sebuah *switch N x N*, maka *switch fabric* (SF) harus menghubungkan *N port input* ke *N port output*. Keuntungan dari *input queuing* adalah sebagai berikut:

1. Membutuhkan kecepatan memori yang rendah.
2. Manajemen trafik didistribusikan pada tiap *port input*.
3. *Lookup* tabel *routing* didistribusikan pada tiap *port input*.



Gambar 2.8 *Input Queuing Switch*
(Sumber: Gebali, 2008)

Sedangkan, kekurangan *input queuing* adalah sebagai berikut:

1. Munculnya permasalahan *Head of Line* (HOL).
2. Kesulitan dalam menerapkan QoS.
3. Kesulitan untuk menerapkan *scheduling*.

Head of Line (HOL) muncul ketika paket yang berada pada bagian atas antrian (*queue*) diblok atau tidak diperbolehkan untuk mengakses oleh *port output* yang dituju. Pemblokan tersebut muncul akibat *switch fabric* tidak dapat memberikan jalur (*internal blocking*) atau jika paket lain sedang mengakses *port output* (*output blocking*). Saat HOL terjadi, paket-paket yang lain mungkin diantrikan di belakang paket yang diblok sehingga sebagai konsekuensinya paket tersebut akan diblok juga oleh *port output*. Oleh karena itu, HOL membatasi *throughput* maksimum *switch*.

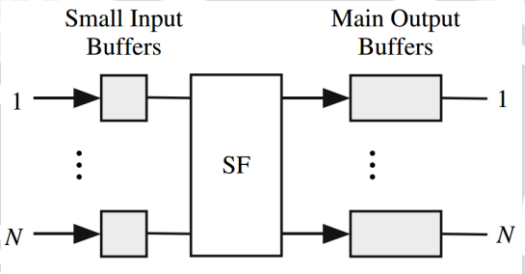
Proses *scheduling* paket sulit dilakukan karena *scheduler* harus membaca seluruh paket yang ada di *port input*. *Scheduler* akan kesulitan untuk mengatur *bandwidth* dan *buffer space* secara merata. Pada *input queuing*, terdapat tiga hal yang berpotensi untuk menyebabkan *packet loss*:

1. *Input queuing* penuh. Paket yang datang tidak memiliki tempat di antrian (*queue*), kemudian dibuang.
2. *Internal blocking*. Paket diblok oleh *switch fabric* dan kemudian dibuang. Hal tersebut terjadi hanya jika *input queue* mengirim paket ke *switch fabric* tanpa menunggu untuk verifikasi jalur paket dapat disediakan.
3. *Output blocking*. Paket yang melewati *switch fabric* mencapai *port output* yang dituju tetapi *port output* menolak karena sedang sibuk melayani paket yang lain. Hal tersebut dapat terjadi jika *input queue* mengirim paket ke *output* tanpa verifikasi ketersediaan *link output*.

2.9 *Output Queuing Switch*

Untuk mengatasi permasalahan *Head of Line* (HOL) pada *input queuing*, yaitu dengan menempatkan *buffer* pada *input queuing* dan *output queuing* seperti yang

ditunjukkan pada Gambar 2.9. *Output queuing switch* harus memiliki nilai *input buffer* yang kecil untuk dapat menyimpan sementara paket yang datang ketika sedang diklasifikasi dan diproses untuk *routing*. Paket yang datang disimpan di *input buffer* dan *controller* pada *input* harus membaca informasi *header* untuk menentukan *output queue* mana yang diperbarui. Paket harus dirutekan melalui *switch fabric* ke *port output* yang sesuai.



Gambar 2.9 *Output Queuing Switch*
(Sumber: Gebali, 2008)

Keuntungan *output queuing* adalah sebagai berikut:

1. Manajemen trafik didistribusikan.
2. *Lookup* tabel *routing* didistribusikan pada tiap *port input*.
3. Mudah dalam mengimplementasikan *Quality of Service* (QoS).
4. Mudah dalam mengimplementasikan distribusi *scheduling* paket pada tiap *port output*.

Sedangkan kekurangan pada *output queuing* adalah sebagai berikut:

1. Memerlukan kecepatan memori yang tinggi untuk *output queue*.
2. Kesulitan untuk mengimplementasikan *broadcast* dan *multicast*.
3. Membutuhkan duplikasi pada data yang sama pada *buffer* yang berbeda yang berhubungan dengan masing-masing *port output*.
4. *Head of Line* (HOL) masih muncul karena *switch* memiliki *input queue*.

Throughput pada *switch* dapat ditingkatkan jika *switch fabric* dapat mengirimkan lebih dari satu paket ke banyak *output* dibandingkan hanya satu *output*. Hal ini dapat dilakukan dengan meningkatkan kecepatan pada *switch fabric* dimana dikenal sebagai *speedup*. Alternatif lain, *switch fabric* dapat menambah dengan menduplikasi jalur atau dengan memilih *switch fabric* yang memiliki lebih dari satu *link* ke banyak *port output*. Saat hal tersebut terjadi, *output queue* harus mampu menangani trafik tambahan dengan meningkatkan kecepatan kerja atau menyediakan antrian (*queue*) terpisah untuk masing-masing *link* masukan.

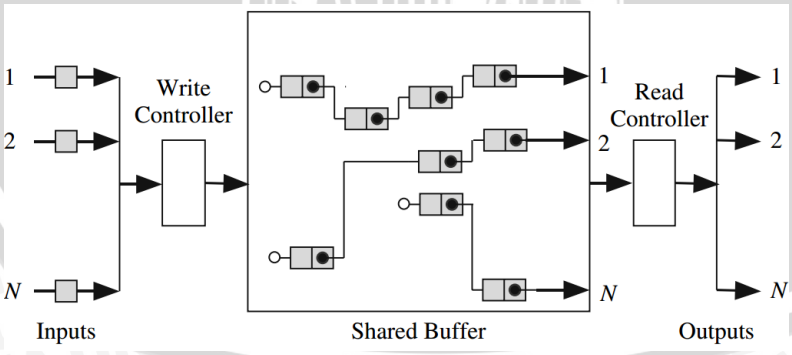
Pada *input queuing*, terdapat empat potensi penyebab *packet loss*:

1. *Input buffer* penuh. Paket yang datang tidak memiiliki tempat di *buffer*, kemudian dibuang.
2. *Internal blocking*. Paket diblok oleh *switch fabric* dan kemudian dibuang.
3. *Output blocking*. Paket yang melewati *switch fabric* mencapai *port output* yang dituju tetapi *port output* menolak karena sedang sibuk melayani paket yang lain.
4. *Output queue* penuh. Paket yang datang tidak memiiliki tempat di *buffer*, kemudian dibuang.

2.10 *Shared Buffer Space*

Gambar 2.10 menunjukkan *shared buffer switch* yang menggunakan satu buah *buffer* bersama dimana semua paket yang datang disimpan di *buffer* tersebut. *Buffer* mengantrikan data dalam antrian (*queue*) yang terpisah yang berada dalam satu memori yang sama. Tiap antrian dihubungkan dengan *port output*. Hampir sama dengan antrian pada *input* dan *output*, tiap *port input* membutuhkan sebuah *buffer* local untuk menyimpan paket yang datang hingga *controller* dapat mengklasifikasi paket-paket tersebut.

Gambar 2.10 menjelaskan mekanisme yang terjadi pada *Shared Buffer Space*, tiap lokasi penyimpanan di dalam *Shared Space Buffer* menyimpan sebuah paket dan sebuah *pointer* (lingkaran hitam) yang berisi alamat paket selanjutnya dalam antrian. *Controller* pada memori menjaga jalur lokasi paket terakhir pada tiap *queue*, ditunjukkan dengan lingkaran kosong.



Gambar 2.10 *Shared Buffer Switch*
(Sumber: Gebali, 2008)

Saat sebuah paket yang baru datang pada *port input*, *controller* pada *buffer* menentukan ke antrian (*queue*) yang mana paket tersebut tuju dan menyimpan paket pada lokasi manapun yang tersedia pada memori, kemudian memperbarui *pointer* yang

diperlukan. Saat sebuah paket meninggalkan antrian (*queue*), *pointer* paket selanjutnya menunjuk ke *port output*.

Keuntungan dari *shared buffer space* adalah sebagai berikut:

1. Kemampuan untuk menetapkan *buffer space* yang berbeda untuk tiap *port output* karena terbatas pada jumlah *free space* di dalam *shared buffer space*.
2. Tidak membutuhkan *switch fabric*.
3. *Lookup* tabel *routing* didistribusikan pada tiap *port input*.
4. Tidak muncul *Head of Line* (HOL) di *shared buffer switch*.
5. Mudah dalam mengimplementasikan *Quality of Service* (QoS).

Kekurangan dari *shared buffer space* adalah sebagai berikut:

1. Membutuhkan kecepatan memori yang tinggi untuk *shared buffer space*.
2. Implementasi fungsi *scheduling* yang terpusat akan memperlambat kinerja *switch*.

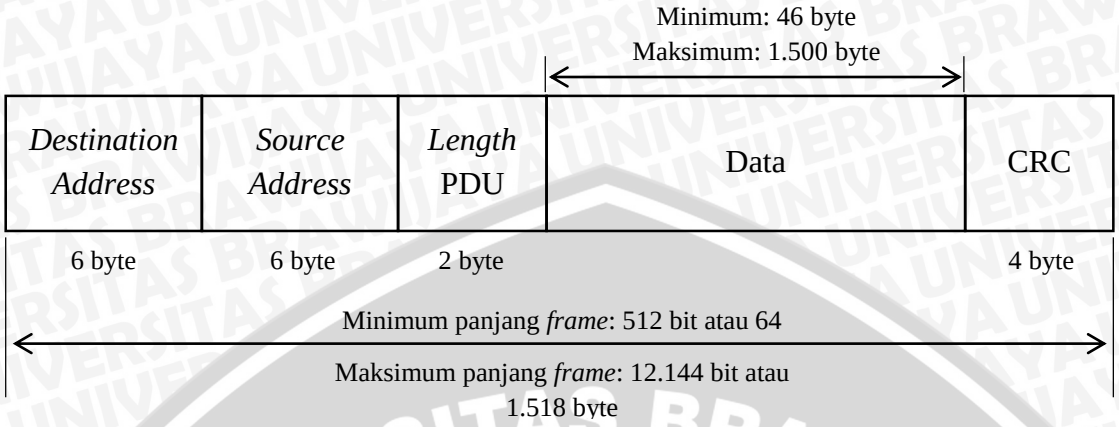
Shared buffer Space harus bekerja dengan kecepatan minimal *2N* karena harus melakukan *N write* dan *N read* tiap waktu. Untuk melakukan *multicast* di *shared buffer switch*, paket *multicast* harus diduplikasi di semua *linked list* pada daftar *multicast*. Hal ini tentunya memakai area penyimpanan yang sebenarnya dapat digunakan untuk paket yang lain. Untuk mendukung layanan yang berbeda, *switch* harus mengatur beberapa antrian (*queue*) pada tiap *port input* untuk tiap layanan. Pada *shared buffer space*, terdapat dua penyebab terjadinya *packet loss*:

1. *Input buffer* penuh. Paket yang datang tidak memiliki tempat di *buffer* dan kemudian dibuang.
2. *Shared buffer space* penuh. Paket yang datang tidak memiliki tempat di *buffer* dan kemudian dibuang.

2.11 Prinsip Kerja *Switch*

Prinsip kerja *switch* penelitian adalah menggunakan standar IEEE 802.3 atau lebih dikenal dengan *Ethernet*. Protokol *Ethernet* meliputi *data link layer* dan *physical layer* pada model OSI. Teknologi *Ethernet* menggunakan *Carrier Sense Multiple Access/Collision Detection* (CSMA/CD) saat mentransmisikan data. Sebelum mengirimkan data, *station* CSMA/CD melakukan pengecekan terhadap jalur transmisi apakah sedang digunakan untuk mentransmisikan data atau tidak. Jika jalur kosong atau tidak digunakan, maka *station* dapat mengirimkan data. *Collision* terjadi saat dua *station* mengirimkan data secara bersamaan. Jika *collision* terjadi maka proses transmisi akan dihentikan, dan data harus dikirim ulang setelah menunggu dalam jeda waktu yang acak

(*backoff*). *Station* memeriksa kembali jalur transmisi, jika kosong maka *station* akan mengirimkan data kembali.



Gambar 2.11 Minimum dan Maksimum Panjang *Frame*
(Sumber: Forouzan, 2007)

Gambar 2.11 merupakan format *frame Ethernet* dengan panjang *frame* minimum dan maksimum. Satu *frame* terdiri dari:

- *Destination Address*: berisi alamat fisik dari *station* tujuan atau *station* yang akan menerima paket.
- *Source Address*: berisi dari alamat fisik pengirim paket.
- *Length*: berisi jumlah byte yang terdapat pada data.
- Data: berisi data yang memiliki panjang minimum 46 byte dan panjang maksimum 1500 byte.
- CRC (*Cyclic Redudancy Check*): berisi informasi deteksi *error*.

2.12 *Video*

Video merupakan suatu teknologi pemrosesan sinyal elektronik yang mewakilkan gambar bergerak (Iwan, 2010). Terdapat dua kategori video, yaitu video analog dan video digital. Pada video analog, informasi gambar dikodekan dengan variasi nilai tegangan dan frekuensi sinyal. Sedangkan video digital, terdiri dari serangkaian gambar digital (*frame*) yang ditampilkan secara cepat pada kecepatan yang konstan. Satuan ukuran untuk menghitung *frame* rata-rata yang ditampilkan disebut *frame per second* (fps). Tiap *frame* terdiri dari banyak *pixel*, dimana *pixel* tersebut mempunyai warna yang dipresentasikan dengan jumlah bit yang tetap. Semakin banyak bit maka semakin banyak warna yang dapat dihasilkan.

Selain *frame per second* (fps), terdapat beberapa parameter *video* lainnya yaitu *video bitrate* dan resolusi *video*. *Video bitrate* adalah jumlah rata-rata bit yang diperlukan

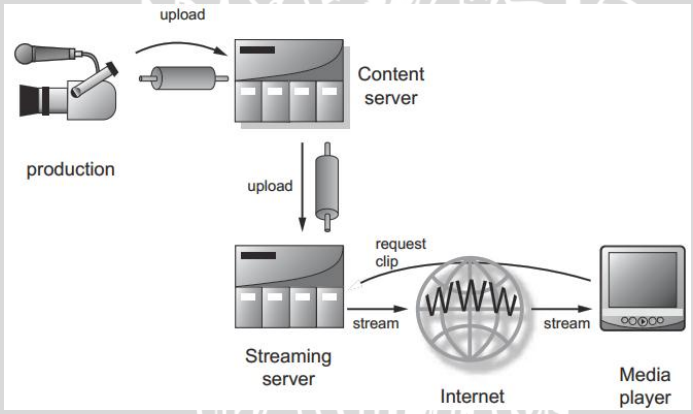
data *video* dikirimkan dalam satu detik. Resolusi *video* menunjukkan jumlah *pixel* yang ditampilkan pada gambar suatu *video*. Dimana semakin besar nilai *video bitrate* dan Resolusi *video,* maka semakin baik kualitas tampilan *video* dan semakin besar ukuran *file video* tersebut.

2.12.1 *Internet Video*

Internet video merupakan layanan *multimedia* yang menyediakan *video* melalui *internet* dengan cara *streaming* atau *download*. *Streaming* adalah proses dimana konten media yang berada di *server* dikirim ke *media player* pada *client* secara *real time* atau bersamaan. *Video streaming* dibedakan menjadi tiga, yaitu *Video on Demand* (VoD), *video live streaming* dan *video interactive* (Forouzan, 2007).

1. *Video on Demand* (VoD)

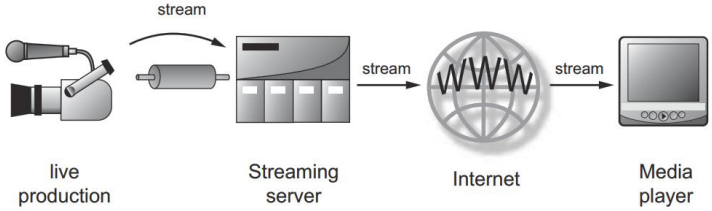
Pada *Video on Demand* (VoD) *video* yang ditampilkan merupakan hasil rekaman dan *video* tersebut telah dikompres serta disimpan pada *server*, misalnya film, siaran TV atau *video* musik. Gambar 2.12 menjelaskan proses *streaming Video on Demand* (VoD) dari proses produksi video, *upload* ke *server* hingga ke *media player*.



Gambar 2.12 *Video on Demand* (VoD)
(Sumber: Austerberry, 2005)

2. *Video Live Streaming*

Pada *video live streaming*, *video* yang ditampilkan merupakan *video* siaran langsung yang disiarkan secara *broadcast* melalui internet dan tidak ada proses *upload* pada *server* seperti yang ditunjukkan pada Gambar 2.13.



Gambar 2.13 *Video Live Streaming*
(Sumber: Austerberry, 2005)

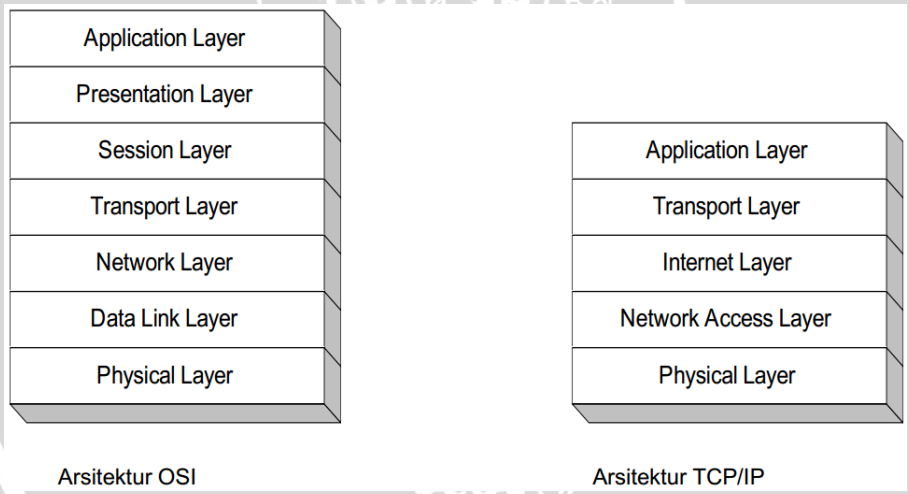
3. *Video Interactive*

Video Interactive merupakan jenis *video* yang digunakan manusia untuk berkomunikasi secara interaktif satu sama lain yang jaraknya berjauhan. Contoh aplikasi dari *video interactive* adalah telepon *internet* dan *video teleconference* yang memungkinkan komunikasi data, suara dan gambar yang bersifat dua arah secara *real time*.

2.13 Protokol TCP/IP (*Transmission Control Protocol/Internet Protocol*)

Protokol adalah sejumlah aturan yang mengatur format frame, paket atau pesan dalam komunikasi data. Protokol TCP/IP adalah rangkaian protokol komunikasi untuk menghubungkan komputer atau *server* pada *internet* (Supriyanto, 2013)..

ISO (*International Standard Organization*) mengeluarkan standar arsitektur jaringan komputer yang dikenal dengan *Open System Interconnection* (OSI) yang terdiri dari tujuh lapisan. Sedangkan, TCP/IP hanya terdapat lima lapisan. Perbandingan Arsitektur OSI dan TCP/IP ditunjukan pada Gambar 2.14.



Gambar 2.14 Arsitektur Protokol TCP/IP
(Sumber: Supriyanto, 2013)

Fungsi dari masing-masing lapisan arsitektur TCP/IP adalah sebagai berikut:

- *Physical Layer* (Lapisan fisik) merupakan lapisan yang menerjemahkan sinyal elektrik menjadi data digital yang dimengerti komputer.
- *Network Access Layer* berfungsi untuk memberikan enkapsulasi datagram dari *network layer* ke dalam frame yang akan dikirim melalui jaringan.
- *Internet Layer* menentukan jalur terbaik pada jaringan untuk pengiriman data dan menjamin agar paket yang dikirimkan dapat sampai ke tujuan paket tersebut.

- *Transport Layer* merupakan lapisan yang menjamin bahwa informasi yang diterima pada sisi penerima adalah sama dengan informasi yang dikirimkan pengirim. Terdapat dua protokol yang digunakan pada lapisan ini yaitu *Transmission Control Protocol* (TCP) dan *User Datagram Protocol* (UDP).
- *Application Layer* merupakan lapisan yang berfungsi mendefinisikan aplikasi-aplikasi yang dijalankan pada jaringan atau *encoding* dan menyajikan data ke pengguna.

Layanan *streaming* membutuhkan protokol transmisi yang dapat mengabaikan data *error*, seperti protokol pada *transport layer* UDP (Austerberry, 2005). Pada umumnya, untuk *streaming* tidak hanya menggunakan protokol UDP tetapi juga terdapat protokol *Real Time Protocol* (RTP).

2.13.1 *User Datagram Protocol* (UDP)

User Datagram Protocol **(**UDP) merupakan protokol yang bersifat *connectionless* dimana tidak melakukan pemeriksaan data *error* dan *flow control* seperti pada TCP. Sehingga tugas tersebut harus ditangani oleh aplikasi yang berada di lapisan lebih tinggi yaitu *application layer*. Pemutar media biasanya dapat menyembunyikan data yang *error*.

2.13.2 *Real Time Protocol* (RTP)

Real Time Protocol (RTP) merupakan protokol *transport layer* yang dikembangkan untuk *streaming* data. RTP menyediakan *timestamp* dan *sequence number* untuk memudahkan waktu transportasi data, dan melakukan kontrol pada media *server* sehingga *video* hasil *streaming* dilayani pada *rate* yang benar untuk tampilan *real time*. *Timestamp* digunakan penerima setelah menerima paket data untuk merekonstruksi paket sesuai dengan wakt pengirim agar data dapat dimainkan pada rate yang benar. UDP mengirimkan data secara acak, sehingga s*equence number* digunakan untuk menyusun paket data sesuai dengan urutan yang benar dan mendeteksi terjadinya *packet loss*.

2.14 Parameter Performansi Jaringan

Quality of Service (QoS) merupakan sekumpulan teknik dan mekanisme yang menjamin performansi dari jaringan komputer di dalam penyediaan layanan pada aplikasi-aplikasi di dalam jaringan komputer. *Quality of Service* (QoS) berkaitan erat dengan layanan *multimedia*.

2.14.1 *Throughput*

Throughput didefinisikan sebagai suatu pengukuran yang menyajikan informasi aktual terkait dengan kemampuan suatu perangkat di dalam mentransmisikan paket data untuk kurun waktu satu detik. Nilai *throughput* diukur dengan satuan bps (*bit per second*). Persamaan 2.1a dan 2.1b digunakan untuk menghitung *throughput* tanpa *losses* secara matematis (Amerasinghe, 2009).

$$\lambda_{FL} = \lambda_{FLVideo} + \lambda_{FLAudio} \quad (2.1a)$$

$$\lambda_{FLVideo} = Width\ x\ Height\ x\ Frame\ Rate\ x\ 0{,}07\ x\ Motion\ Rank \quad (2.1b)$$

dengan:

λ_{FL} = *Throughput* tanpa *losses* (bps)
$\lambda_{FLVideo}$ = *Throughput video* (bps)
$\lambda_{FLAudio}$ = *Throughput audio* (bps)
$Width$ = Lebar *video*
$Height$ = Tinggi *video*
$Frame\ Rate$ = Jumlah *frame* yang terdapat pada *video* per sekon (fps)
$Motion\ Rank$ = Bernilai 1 hingga 4 tergantung gerakan yang ada pada *video*

Jumlah gerakan yang terjadi dalam *video* disebut *motion rank*. Terdapat tiga jenis *motion rank* yaitu:

- *Low motion* bernilai 1, dimana *video* tidak memiliki gerakan yang terlalu banyak.
- *Medium motion* bernilai 2, dimana *video* memiliki gerakan yang dinamis dan perubahan *scene* yang cukup drastic.
- *High motion* bernilai 4, dimana *video* memliliki gerakan yang sangat dinamis dengan pergerakan dan perubahan *scene* yang sangat cepat.

Persamaan 2.1c digunakan untuk memperoleh nilai *Throughput* secara matematis dengan adanya *losses* (Forouzan, 2007).

$$Throughput\ (bps) = \frac{(L+L')\ x\ N_T}{t_v} \quad (2.1c)$$

dengan:

t_v = waktu total transmisi untuk mengirimkan paket yang benar (s)
L = panjang paket data yang diterima (bit/paket)
L' = panjang *header* paket (bit/paket)
N_T = jumlah paket yang diterima dengan benar (paket)

2.14.2 *Packet loss*

Salah satu hal yang perlu diperhatikan di dalam perhitungan performansi jaringan komputer, khususnya pada *network layer* adalah nilai *packet loss*. *Packet loss* dapat diartikan sebagai hilangnya sejumlah paket data pada jaringan komputer selama proses transmisi paket data. Semakin kecil nilai *packet loss* dalam suatu jaringan maka semakin baik pula kinerja yang dimiliki jaringan tersebut. Berdasarkan ITU-T G.1010, standar *packet loss* untuk aplikasi *streaming* ditunjukkan Tabel 2.1.

Tabel 2.1 Standar *Packet Loss*

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: ITU-T G.1010, 2001)

Persamaan 2.2 digunakan untuk menghitung *packet loss*.

$$Packet\ loss\ (\%) = \frac{N_{packet\ loss}}{N_{packet}+N_{packet\ loss}}\ x\ 100\% \qquad (2.2)$$

dengan:

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

N_{packet} = jumlah paket multimedia rata-rata (paket)

2.14.3 *Delay*

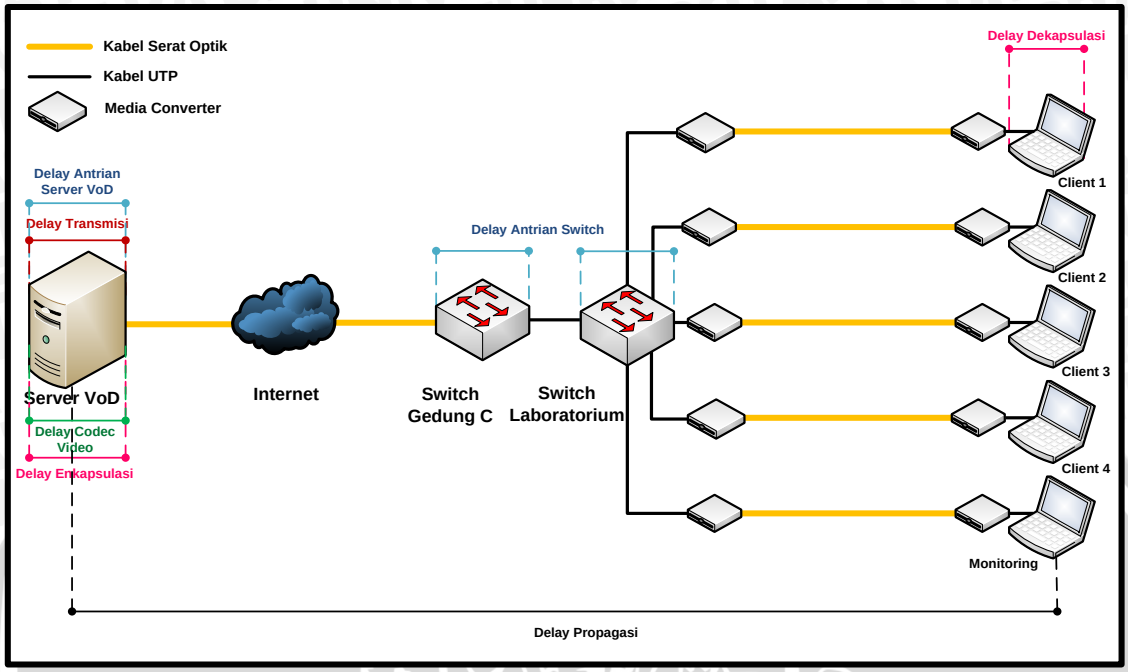
Delay didefinisikan sebagai lamanya waktu yang diperlukan oleh paket data untuk sampai ke tujuan. Tidak semua paket data didalam layanan *multimedia* jaringan komputer yang menggunakan *internet* yang dapat mentolerir adanya delay, seperti *file audio* dan *video* digital. Hal tersebut memiliki dampak pada tingkat kepuasan pelanggan saat melakukan *streaming audio* maupun *video* karena akan sangat mengganggu data yang diterima oleh pelanggan. Berdasarkan ITU-T G.114, terdapat tiga pengelompokan nilai *delay* seperti pada Tabel 2.2.

Tabel 2.2 Standar *Delay*

***Delay* (ms)**	**Kualitas**
0-150	Baik
150-400	Cukup, masih dapat diterima
>400	Buruk

(Sumber: ITU-T G.114, 2000)

Delay ($t_{end-to-end}$) terdiri dari empat macam *delay* yaitu *delay* propagasi, *delay* transmisi, *delay* proses dan *delay* antrian (*queue*) seperti yang ditunjukkan Gambar 2.15.



Gambar 2.15 Blok Diagram *Delay end-to-end*

Delay ($t_{end-to-end}$) diperoleh dengan menjumlahkan empat macam *delay* tersebut sesuai dengan persamaan 2.3.

$$t_{end-to-end}(s) = t_{propagasi}(s) + t_{transmisi}(s) + t_{proses}(s) + t_{queue}(s) \qquad (2.3)$$

1. *Delay* Propagasi

Delay propagasi ($t_{propagasi}$) adalah waktu yang dibutuhkan oleh paket data untuk merambat dari *server* menuju *client* melalui media transmisi. Tabel 2.3 merupakan kecepatan propagasi beberapa media transmisi.

Tabel 2.3 Kecepatan Propagasi Media Transmisi

Jenis Media Transmisi	**Kecepatan Propagasi** (c = 3 x 10^8 m/s)
Serat Optik	0,66c
UTP	0,64c

(Sumber: Messer, 2014)

Berikut adalah persamaan 2.4 untuk menghitung *delay* propagasi (Forouzan, 2007):

$$t_{propagasi}\ (s) = \frac{Panjang\ Media\ Transmisi\ (m)}{Kecepatan\ Propagasi\ (m/s)} \quad (2.4)$$

2. *Delay* Transmisi

Delay transmisi ($t_{transmisi}$) adalah waktu yang dibutuhkan untuk transmisi paket data, dan begantung pada ukuran paket yang dikirimkan dan *bandwidth* kanal. Persamaan 2.5 digunakan untuk menghitung *delay* propagasi (Forouzan, 2007).

$$t_{transmisi} = \frac{(L+L')}{B} \quad (2.5)$$

Keterangan:

$t_{transmisi}$ = *delay* transmisi (sekon)
L = panjang paket data (bit)
L' = panjang *header* paket data (bit)
B = *bandwidth* kanal (bps)

3. *Delay* Proses

Delay proses (t_{proses}) terdiri dari *delay* enkapsulasi dan *delay* dekapsulasi. *Delay* enkapsulasi adalah waktu yang dibutuhkan untuk proses pemaketan data dengan menambahkan *header* pada paket data, sehingga paket data tersebut dapat dikirimkan ke tujuan. Sedangkan *delay* dekapsulasi adalah waktu yang dibutuhkan untuk proses pembacaan keseluruhan *header* dari sebuah paket yang diterima suatu *node*. Perhitungan *delay* enkapsulasi dan *delay* dekapsulasi diperoleh menggunakan persamaan 2.6 dan 2.7 (Purbo *et al*. 2001).

$$t_{enc} = \frac{L'}{C_{proses}} x8 \quad (2.6)$$

$$t_{dec} = \frac{L'}{C_{proses}} x8 \quad (2.7)$$

Keterangan:

t_{enc} = *Delay* enkapsulasi (s)
t_{dec} = *Delay* dekapsulasi (s)
L' = Panjang *header* paket data (bit)
C_{proses} = Kecepatan pemrosesan data pada *node* (bps)

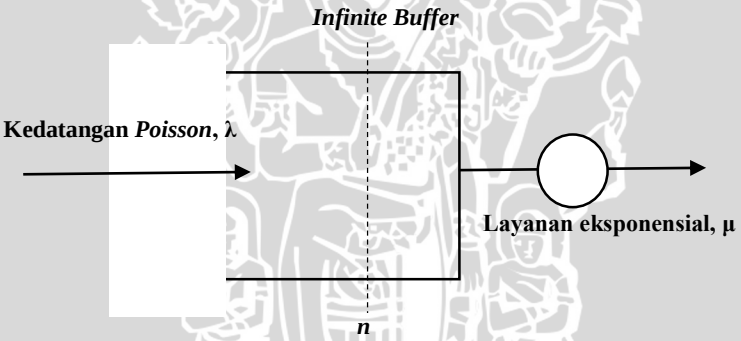
Maka, *delay* proses diperoleh menggunakan persamaan 2.8.

$$t_{proses}(s) = t_{enc}(s) + t_{dec}(s) \quad (2.8)$$

4. *Delay* Antrian

Delay antrian adalah lamanya waktu yang diperlukan maupun keterlambatan yang dialami oleh paket data dalam kurun waktu tertentu sebagai akibat terjadinya pemberlakuan antrian untuk pemrosesan di dalam jaringan komputer. Peran *delay* antrian pada jaringan komputer yaitu menentukan seberapa cepat paket data diproses pada suatu *node* (*Server, Router* atau *Switch*), terkait dengan pengiriman dan penerimaan paket data. Semakin lama paket menunggu di dalam antrian, maka semakin besar nilai *delay* antrian yang dihasilkan. Hal ini dapat mengakibatkan penurunan performansi jaringan.

Delay antrian yang akan dihitung menggunakan model antrian M/M/1. Model antrian M/M/1 merupakan model antrian dengan *single-server*, kedatangan *Poisson*, distribusi waktu pelayanan eksponensial, dan menerapkan model *First Input First Output* (FIFO) yaitu data pertama yang masuk akan diproses dan dikeluarkan dari antrian terlebih dahulu. Gambar 2.16 menunjukkan model antrian M/M/1.



Gambar 2.16 Model Antrian M/M/1
(Sumber: Schwartz, 1987)

Delay antrian (t_{queue}) dengan model antrian M/M/1 diperoleh dengan menggunakan persamaan 2.9 (Schwartz, 1987):

$$t_{queue} = 1/\mu/(1-\rho) \quad (2.9)$$

sedangkan,

$$\mu = \frac{C}{(L+L')} \quad (2.10)$$

dan,

$$\rho = \frac{\lambda}{\mu} \quad (2.11)$$

dan,

$$\lambda = \frac{N}{T} \quad (2.12)$$

Keterangan:

t_{queue} = *Delay* antrian (s)
μ = Kecepatan pelayanan *node* (bps)
C = Kapasitas Kanal (bps)
L = panjang paket data (bit)
L' = panjang *header* paket data (bit)
ρ = Faktor utilitas ($0 < \rho < 1$)
λ = Kecepatan kedatangan paket pada node (paket/sekon)
N = Jumlah paket pada *node* (paket)
T = Waktu pengiriman paket total (s)

2.14.4 *Delay* Total Sistem *Video on Demand* (VoD)

Penelitian ini menggunakan layanan *Video on Demand* (VoD), sehingga untuk mendapatkan besarnya *delay* total (t_{total}) dengan menjumlahkan *delay* jaringan ($t_{end-to-end}$) dengan *delay codec* (t_{codec}) seperti persamaan 2.13.

$$t_{total}(s) = t_{codec}(s) + t_{end-to-end}(s) \quad (2.13)$$

Delay codec diperoleh dengan menggunakan persamaan 2.14. Spesifikasi *codec video* dan *audio* yang digunakan pada penelitian ditunjukkan Tabel 2.4.

$$t_{codec} = t_{video} + t_{audio} \quad (2.14)$$

Keterangan:

t_{codec} = *Delay codec* (s)
t_{video} = *Delay codec video* (s)
t_{audio} = *Delay codec audio* (s)

Tabel 2.4 Spesifikasi *Codec Video* dan *Audio*

Video *CODEC*	***Bit Rate* (kbps)**	***Maximum Payload* (byte)**	***Delay CODEC* (ms)**
AVC/H.264 MPEG4	96-384	254	16-50
Audio *CODEC*	***Bit Rate* (kbps)**	***Maximum Payload* (byte)**	***Delay CODEC* (ms)**
AAC	128	63	24-60

(Sumber: RFC 3640, 2003, dan RFC 3984, 2005)

Sedangkan *header* paket data yang digunakan pada penelitian ini adalah *header* NALU, RTP, UDP, IPv4, CRC, dan *Ethernet*. Panjang *header-header* tersebut ditunjukkan Tabel 2.4.

Tabel 2.5 *Header Format*

Protokol	Kuantitas	Sumber
RTP *Header*	12 byte	Peterson, L, *et al*, 2007
IPv4 *Header*	20 byte	RFC 791,
UDP *Header*	8 byte	Stalling, 2007
Cyclic Redudancy Check	4 byte	Forouzan, 2007
Ethernet *Header*	14 byte	Forouzan, 2007

2.15 Sistem Komunikasi Serat Optik

Sistem komunikasi serat optik merupakan suatu sistem komunikasi yang menggunakan serat optik sebagai media transmisi. Sistem komunikasi serat optik terdiri dari:

2.15.1 *Optical Source*

Pada blok *optical source* terdapat sumber optik. Sumber optik merupakan perangkat pembangkit gelombang elektromagnetik pada frekuensi optik yaitu $3x10^{11}$-$3x10^{16}$ Hz yang digunakan untuk membawa informasi yang akan ditransmisikan. Terdapat dua tipe sumber optik, yaitu *Light Emitting Diode* (LED) dan *Laser Diode* (LD).

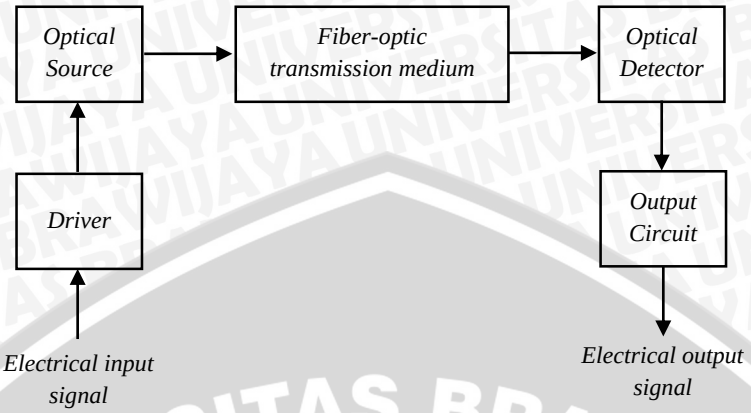
2.15.2 *Optical Detector*

Pada blok *optical detector* terdapat detektor optik. Detektor optik adalah perangkat yang dapat mengubah sinyal optik (cahaya) menjadi sinyal listrik kembali. Terdapat dua tipe detektor optik, yaitu *Positive-Intrinsic Negative* (PIN) dan *Avalanched Photo Diode* (APD).

2.15.3 Kabel serat optik

Ada dua jenis kabel serat optik berdasarkan mode perambatan cahaya, yaitu *singlemode* dan *multimode*. *Singlemode* mempunyai ukuran inti yang kecil, sehingga hanya ada satu mode cahaya yang lewat di dalamnya. Sedangkan, *multimode* mempunyai ukuran inti yang lebih besar dari *singlemode* sehingga mode cahaya yang mampu dilewatkan di dalam serat lebih banyak.

Gambar 2.17 merupakan blok diagram dari sistem komunikasi serat optik secara umum.



Gambar 2.17 Blok Diagram Sistem Komunikasi Serat Optik
(Sumber: Freeman, 2005)

2.16 Perangkat Lunak Wireshark

Wireshark merupakan aplikasi *network protocol analyzer* yang bersifat *open source* pada Windows dan Linux. Awalnya dikenal dengan *Ethereal*. Aplikasi tersebut bertujuan untuk menganalisis trafik jaringan yang melewati suatu perangkat, umumnya komputer atau laptop. Wireshark menerapkan berbagai macam filter yang mendukung lebih dari 1100 protokol (*Inteco-Cert*, 2011) secara sederhana dan memungkinkan untuk memisahkan paket hasil *capture* berdasarkan *layer*.

Gambar 2.18 Logo Perangkat Lunak *Wireshark*
(Sumber: www.wireshark.org)

Beberapa kelebihan dari Wireshark diantaranya mampu menangkap paket data secara langsung dari *network interface*, mampu menampilkan secara rinci mengenai hasil *capture* paket pada sebuah jaringan, serta mampu menampilkan hasil statistika dari hasil *capture* pada sebuah jaringan.

2.17 VLC *Media Player*

VLC *Media Player* merupakan perangkat lunak yang berfungsi sebagai pemutar berbagai macam *file multimedia*, baik *video* maupun *audio* dalam berbagai format seperti MPEG, DivX, Ogg, dan lain-lain. VLC *Media Player* bersifat *open* source dan tersedia untuk berbagai sistem operasi, seperti *Windows*, *Linux*, *Mac OS*, dan lain-lain. C*odec* yang dimiliki *media player* ini sangat lengkap. Sehingga VLC *Media Player* mampu

memutar *file multimedia*, baik *file* yang ada di komputer atau laptop, *file* CD atau DV, hingga *streaming file* melalui *internet*.

Gambar 2.19 Logo VLC *Media Player*
(Sumber: www.videolan.org)

2.18 Perangkat Keras Jaringan

Selain *switch* yang telah dijelaskan pada subbab 2.4, terdapat beberapa perangkat keras yang digunakan dalam jaringan:

2.18.1 *Media Converter*

Media Converter merupakan perangkat yang berfungsi untuk mengubah sinyal listrik pada kabel tembaga (UTP) menjadi gelombang cahaya pada kabel serat optik (*Patchcord*). Selain itu, *media converter* juga digunakan untuk konversi jenis serat optik, *singlemode* menjadi *multimode* atau sebaliknya.

2.18.2 *Patchcord*

Patchcord merupakan kabel serat optik dengan panjang tertentu yang terpasang konektor di kedua ujungnya. *Patchcord* berfungsi untuk menghubungkan antar perangkat jaringan dan digunakan untuk di dalam ruangan saja.

2.18.3 UTP (*Unshielded Twisted Pair*)

UTP merupakan kabel jaringan yang paling umum digunakan untuk *wired network*. UTP terbuat dari bahan tembaga, yang berfungsi untuk menghubungkan antar perangkat jaringan.