BAB II

LANDASAN TEORI

Adapun landasan teori yang digunakan untuk membuat aplikasi pembatasan dan pengaturan bandwidth dengan CBQ (*Class Based Queueing*) pada sistem operasi Linux berbasis web antara lain landasan teori tentang Jaringan Komputer, Model TCP/IP, *Quality of Service* (QoS), Disiplin Antrian, *Class Based Queueing* (CBQ), Linux Kernel, Squid, Iptables, Apache, OpenSSL, dan landasan teori tentang pembuatan web antara lain CGI-Perl, PHP, dan MySQL.

2.1 Jaringan Komputer

Definisi jaringan menurut Dian Ardiansyah (2003:1) adalah "kumpulan komputer, printer dan peralatan lainnya yang terhubung. Informasi dan data bergerak melalui kabel-kabel sehingga memungkinkan pengguna jaringan komputer dapat saling bertukar dokumen dan data, mencetak pada printer yang sama dan bersama sama menggunakan *hardware/software* yang terhubung dengan jaringan."

Tiap komputer, printer atau perangkat jaringan yang terhubung dengan jaringan disebut *node*. Sebuah jaringan komputer dapat memiliki dua, puluhan, ribuan atau bahkan jutaan *node*. Sebuah jaringan biasanya terdiri dari dua atau lebih komputer yang saling berhubungan diantara satu dengan yang lain, dan saling berbagi sumber daya misalnya CDROM, Printer, pertukaran file, atau memungkinkan untuk saling berkomunikasi secara elektronik. Komputer yang terhubung (*host*) tersebut, dimungkinkan berhubungan dengan media kabel,

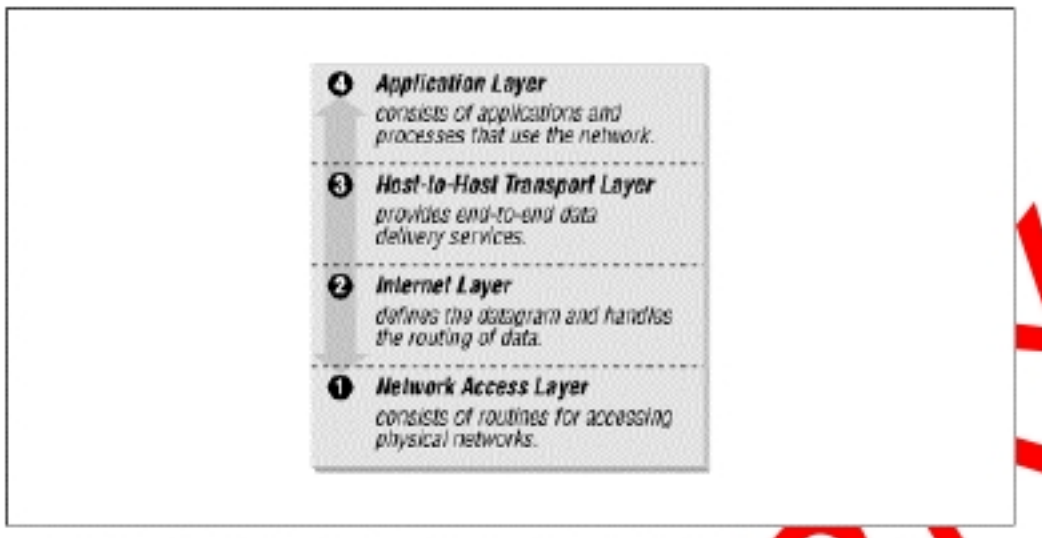
saluran telepon, gelombang radio, satelit, atau sinar infra merah. (Ardiansyah, Dian, 2003:2)

2.2 Model TCP/IP

TCP/IP (*Transfer Control Protocol/Internet Protocol*) merupakan sekumpulan protokol yang dikembangkan agar komputer (yang terhubung ke jaringan komputer) dapat saling berinteraksi dan saling berbagi (*share*) sumber daya (*resource*) yang dimiliki masing-masing. Protokol-protokol TCP/IP dikembangkan sebagai bagian dari riset yang dikembangkan oleh *Defense Advanced Research Projects Agency (DARPA)*. Pertama kalinya TCP/IP dikembangkan untuk komunikasi antar jaringan yang terdapat pada DARPA. Selanjutnya, TCP/IP dimasukkan pada distribusi software UNIX. Sekarang TCP/IP telah digunakan sebagai standar komunikasi *internetwork* dan telah menjadi protokol *transport* bagi internet, sehingga memungkinkan jutaan komputer berkomunikasi secara global. TCP/IP memungkinkan komunikasi diantara sekumpulan interkoneksi jaringan dan dapat diterapkan pada jaringan LAN (*Local Area Network*) ataupun WAN (*Wide Area Network*).

2.2.1 Arsitektur Model TCP/IP

Seperti telah disebutkan sebelumnya, TCP/IP berisi kumpulan dari protokol-protokol yang melakukan fungsinya masing-masing secara spesifik. Protokol-protokol ini dikumpulkan berdasarkan fungsinya dalam lapisan-lapisan tertentu. Arsitektur model TCP/IP seperti pada gambar 2.1 memiliki 4 lapisan (*layer*) yang antara satu dengan lainnya memiliki protokol dengan fungsi yang saling melengkapi satu sama lain. (Hunt, Craig, 1997:I)



Gambar 2.1 Arsitektur Model TCP/IP

a. Network Access Layer

Layer ini merupakan layer paling bawah dari arsitekur model TCP/IP. Protokol pada layer ini menyediakan informasi tujuan, kemana paket data akan dikirimkan kepada perangkat jaringan yang terpasang pada sebuah jaringan komputer, layer ini juga mendefinisikan bagaimana jaringan komputer melakukan transmisi IP (*Internet Protocol*) datagram. Tidak seperti protokol-protokol pada layer diatasnya, protokol pada layer ini harus mengetahui secara detail tentang hal-hal yang mendasar pada sebuah jaringan komputer seperti struktur paket data, pengalamatan, dan sebagainya agar dapat memastikan format paket data yang ditransmisikan dengan mengikuti aturan dari jaringan komputer yang dilewati. Layer ini sudah mencakup tiga layer dibawahnya dari model OSI (*Open System Interconnection*) yakni; *Network Layer, Data Link Layer, Physical Layer*. Fungsi yang terdapat pada layer ini meliputi proses *encapsulation* IP datagram kedalam *frame* yang ditransmisikan oleh jaringan komputer dan pemetaan (*mapping*) dari alamat IP (*IP Address*) ke alamat *physical* pada NIC (*Network Interface Card*). (Hunt, Craig, 1997:I).

b. Internet Layer

Pada layer ini terdapat protokol yang merupakan jantung dari TCP/IP dan merupakan protokol yang sangat penting yang terdapat pada layer ini yakni *Internet Protocol* (IP). IP menyediakan dasar dari sebuah proses pengiriman paket data TCP/IP pada jaringan komputer. Semua protokol yang berada pada layer diatasnya dan dibawahnya menggunakan *Internet Protocol* untuk mengirimkan paket data baik itu paket masuk (*incomming packet*) dan paket keluar (*outgoing packet*) (Hunt, Craig, 1997:I).

- **Internet Protocol (IP)**

Craig, Hunt (1997:I) menjelaskan bahwa "karakteristik dari IP adalah *connectionless protocol* yang berarti IP tidak dapat menukarkan informasi kontrol untuk membentuk koneksi *end-to-end* sebelum melakukan transmisi paket data, sebaliknya *connection-oriented protocol* menukarkan informasi kontrolnya dengan *remote system* untuk melakukan verifikasi apakah *remote system* sudah siap menerima paket data sebelum dilakukan pengiriman paket data. Ketika pembentukan koneksi berhasil dilakukan maka *remote system* akan memberi informasi untuk membentuk sebuah koneksi. IP mempercayakan pada layer yang lain untuk membentuk koneksi jika layer yang lain tersebut membutuhkan layanan *connection-oriented*. IP juga mempercayakan ke layer lainnya untuk menyediakan deteksi kesalahan dan perbaikan kesalahan. IP juga dikenal dengan *unreliable protocol* karena tidak berisikan deteksi kesalahan (*error detection*) dan perbaikan kesalahan (*recovery code*)".

Fungsi dari *Internet Protocol* meliputi:

– Mendefinisikan Datagram

Protokol TCP/IP dibangun untuk melakukan transmisi data melewati ARPNET (*packet switching network*) untuk melakukan paket switching network. *Packet switching network* berkerja menggunakan informasi pengalamatan yang terdapat dalam paket data untuk berpindah dari satu NIC (*Network Interface Card*) ke NIC lainnya, hingga sampai pada tujuannya. Paket data merupakan blok data yang membawa paket data dengan informasi yang diperlukan untuk mengirimkan paket data tersebut.

Datagram merupakan format paket data yang didefinisikan oleh *Internet Protocol* seperti terlihat pada gambar 2.2. dimana angka 1 s/d 6 merupakan 32-bit word dari datagram yang digunakan untuk mengontrol informasi sering disebut sebagai *header*. Secara default panjang header adalah 5 word dan word ke-6 adalah opsional. Berikut ini adalah penjelasan dari tiap-tiap tingkatan word :

– Word ke-1

Version; mengindikasikan versi dari nomor IP yang digunakan (IPv4 atau IPv6)

IP Header Length (IHL); mengindikasikan panjang header datagram (dalam 32-bit word)

Type of Service; menetapkan bagaimana sebuah protokol layer menangani sebuah datagram dan menugaskan datagram berdasarkan tingkatan

terpenting dengan kata lain mengindikasikan kualitas layanan yang diinginkan.

Total Length; menetapkan panjang keseluruhan paket IP (dalam byte) termasuk data dan header-nya

- Word ke-2

Identification; digunakan untuk membantu menyatukan fragmen datagram (field merupakan sebuah bilangan yang mengindentifikasikan datagram saat ini)

Flag; terdiri atas field 3 bit yang mengontrol fragmentasi

Fragmentation Offset; mengindikasikan posisi data fragmen relatif terhadap permulaan data dalam datagram orisinil. Hal ini memungkinkan proses IP tujuan dapat tepat dapat mengkonstruksi ulang datagram orisinil.

- Word ke-3

Time to Live; menunjukan waktu maksimum bagi sebuah datagram untuk berada pada sebuah jaringan komputer. Bila field ini bernilai 0 maka datagram akan dibuang. Field ini dimodifikasi selama tahap pemrosesan header IP dan umumnya dihitung dalam detik.

Protocol; mengindikasikan layer protokol diatasnya yang akan menerima paket setelah proses IP telah selesai.

Header Checksum; sebuah nilai checksum untuk header saja, dimana nilai ini harus dihitung ulang setiap kali sebuah proses header dimodifikasi.

- Word ke-4

Source Address; alamat IP dari host yang mengirimkan datagram

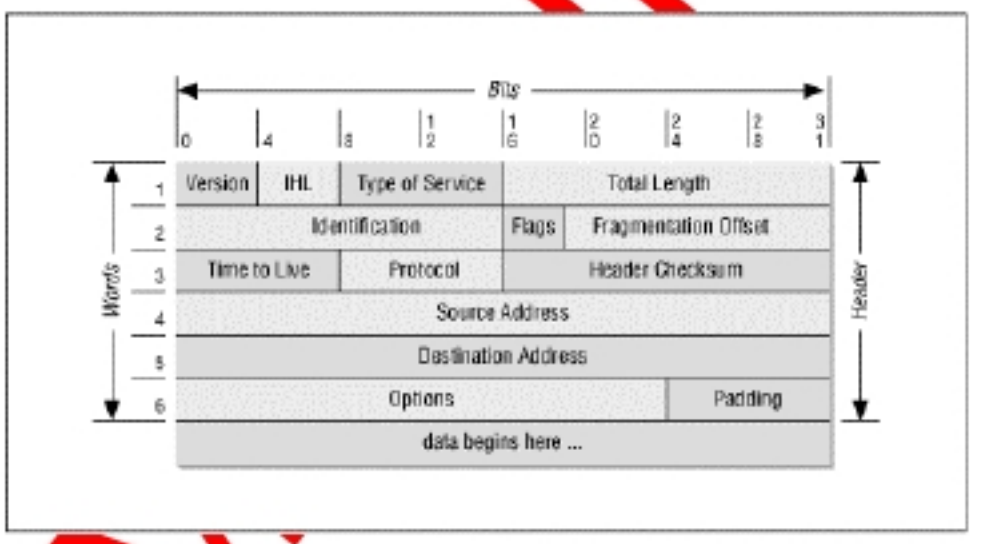
- Word ke-5

  Destination Address; alamat IP dari host yang merupakan tujuan akhir dari datagram

- Word ke-6

  Options dan *Padding*; memungkinkan IP mendukung beragam opsi lainnya seperti keamanan

Karena panjang header merupakan variabel yang termasuk field pada sebuah *Internet Header Length* (IHL) yang mengindikasikan panjang word. Header berisikan informasi yang diperlukan untuk mengirimkan paket data. (Hunt, Craig, 1997:I).



Gambar 2. 2 Format IP Datagram

IP mengirimkan datagram dengan memeriksa terlebih dahulu alamat tujuan (*destination address*) didalam word ke-5 dari header. Alamat tujuan merupakan IP Address stardard 32-bit yang mengidentifikasikan alamat jaringan dan spesifikasi *host* pada jaringan tersebut, jika berada pada jaringan lokal maka paket akan dikirimkan langsung ketujuan. Sebaliknya jika tujuan tidak berada pada jaringan lokal maka paket data akan di-*bypass* terlebih dahulu ke gateway untuk selanjutnya dilakukan pengiriman ke alamat tujuan.

Gateway merupakan perangkat untuk memindahkan paket data antara jaringan komputer yang berbeda, ini tergantung dari *routing* perangkat *gateway* tersebut.

- Mendefinisikan skema pengalamatan (*Routing Datagram*)

  Internet Gateway biasanya disebut *IP Routers* karena internet gateway mengarahkan perjalanan paket data antara jaringan komputer apakah paket data akan di-*forward*-kan (dilanjutkan) atau tidak.

- Melakukan Fragmentasi Datagram

  Masing-masing tipe dari NIC (*Network Interface Card*) memiliki *Maximum Transmission Unit* (MTU), yang menjadi standard besarnya paket yang dapat ditransfer. Jika datagram dapat diterima dari sebuah jaringan komputer lebih besar dari ukuran MTU-nya, maka datagram tersebut akan dibagi dalam *fragmen-fragmen* kecil untuk kemudian ditransmisikan, proses ini disebut *fragmentation*. Format dari masing-masing fragmen sama dengan format dari datagram, header word ke-2 berisikan informasi-informasi yang mengidentifikasikan masing-masing fragmen datagram dan menyediakan informasi bagaimana *re-assembly* (mengumpulkan kembali) fragmen-fragmen tersebuat menjadi sebuah datagram utuh dimana *field* yang mengidentifikasikan fragmen ini milik datagram mana pada saat di-fragmentasi dan *Fragmentation Offset* menginformasikan bahwa kepingan-kepingan fragmen milik datagram mana untuk di *re-assembly*, sedangkan *Flag* memiliki fragmen tersendiri berupa bit untuk menginformasikan kepada IP jika proses *re-assembly* semua fragmen menjadi datagram sudah dilakukan

- Melewatkan Datagram ke Transport Layer

Ketika IP menerima datagram yang dikirimkan dari sebuah alamat *host* lokal, datagram harus memberikan *data portion* (bagian dari datagram) untuk dikoreksi oleh protokol pada *Transport Layer*. Ini dilakukan menggunakan *protocol number* dari word ke-3 pada datagram header. Masing masing protokol *Transport Layer* memiliki *protocol number* yang unik untuk mengindenfikasikannya kepada IP.

- **Internet Control Message Protocol (ICMP)**

Suatu bagian terintegrasi dari IP adalah ICMP, protokol ini merupakan bagian dari Internet Layer dan digunakan oleh datagram IP sebagai fasilitas pengiriman untuk mengirimkan pesannya. Pesan yang dikirimkan oleh ICMP antara lain:

– *Flow Control*; jika datagram yang diproses datang lebih cepat dari yang diharapkan, maka host tujuan akan mengirimkan ICMP *Source Quench Message* kembali kepada pengirim. Hal ini menginformasikan kepada host asal agar pengiriman datagram dihentikan sementara.

– *Detecting Unreachable Destinations*; ketika tujuan pengiriman tidak dapat dijangkau (*unreachable*), system akan mendeteksi problem dan mengirimkan pesan *Destination Unreachable Message* kepada host asal dari datagram. Jika tujuan yang tidak bisa dijangkau adalah sebuah host atau network maka pesan akan dikirimkan dari gateway, sedangkan jika port yang tidak dapat dijangkau maka host tujuan yang akan mengirimkan pesan tersebut.

– *Redirecting Routers*; gateway mengirimkan ICMP *Redirect Message* untuk memberitahukan host agar mengunakan jalur gateway lain, yang dipilih oleh gateway tersebut.

– *Checking Remote Host*; host dapat mengirimkan ICMP *Echo Message*, untuk melihat apakah IP dari *remote system* dalam keadaan hidup dan sedang beroperasi. Ketika system menerima *echo message*, maka system akan membalas (*echo reply*) dan mengirimkan paket data kembali ke host asal. (perintah *ping* biasanya menggunakan pesan ini) (Hunt, Craig, 1997:I)

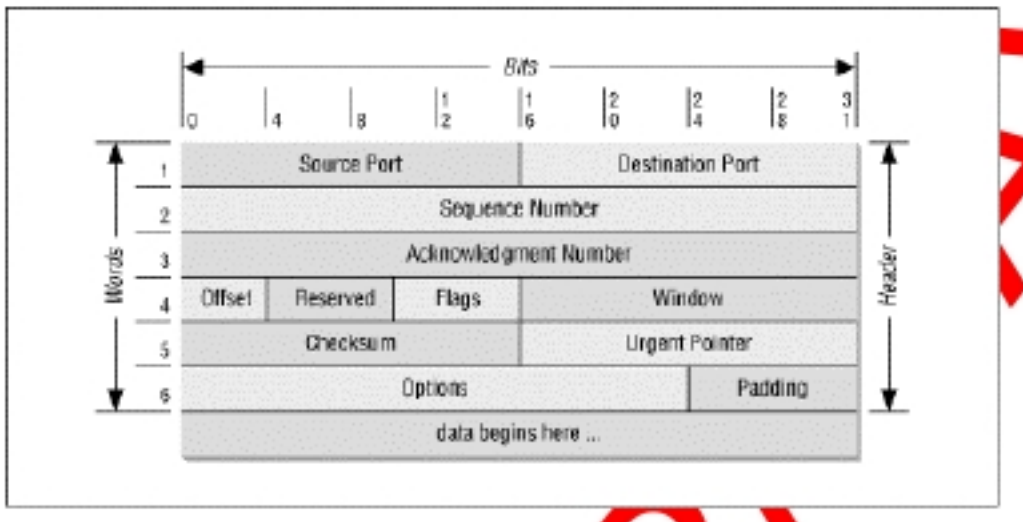
c. Transport Layer

Pada layer ini terdapat dua protokol yang sangat penting yakni protokol TCP (*Transmission Control Protocol*) dan protokol UDP (*User Datagram Protocol*). TCP menyediakan layanan pengiriman paket data yang *reliable* dengan *end-to-end error detection* dan *error correction*, sedangkan UDP menyediakan layanan pengiriman datagram yang *connectionless*. (Hunt, Craig, 1997:I)

▪ **Transmission Control Protocol (TCP)**

Aplikasi-aplikasi jaringan komputer membutuhkan protokol *transport* untuk menyediakan pengiriman data yang handal menggunakan protokol ini karena TCP memverifikasikan pengiriman data yang melewati jarigan komputer secara akurat dan sesuai urutan. TCP menyediakan mekanisme yang handal yang disebut *Positive Acknowledgment with Re-transmission* (PAR) untuk mengirimkan lagi data, jika mengetahui bahwa data yang dikirimkan telah sampai pada *remote system*. *Data Exchange* unit berkerja sama dengan modul TCP yang disebut segmen, seperti terlihat pada gambar 2.3. Masing-masing segment berisikan *checksum* yang digunakan oleh penerima untuk memverifikasi bahwa data tersebut tidak rusak. Jika segmen data tidak rusak, penerima akan mengirimkan *positive acknowledgment* kembali ke pengirim, dan jika segmen data yang dikirimkan rusak maka penerima akan menolak data pengiriman data tersebut,

sampai setelah periode *time-out* yang tepat TCP modul pengiriman akan melakukan transmisi lagi (*re-transmission*) untuk setiap segmen yang tidak terdapat *positive acknowledgment* (Hunt, Craig, 1997:II)



Gambar 2. 3 Format Segmen TCP

Rafiudin, Rahmat (2005:77) menjelaskan bahwa "TCP sering disebut protokol layer Transport berorientasi koneksi (*connection oriented*) yang *reliable* dalam penghantaran arus data (*byte stream*)".

Byte Stream Delivery, TCP memberikan interface diantara layer yang lebih tinggi (*Application Layer*) dan layer dibawahnya (*Network Access Layer*). Saat sebuah aplikasi mengirimkan data ke TCP, TCP melakukannya dalam 8 bit *byte stream* yang berguna untuk mentransmisi data dalam pecahan yang dapat dikelola.

Connection Oriented, sebelum dapat berkomunikasi dan melakukan pertukaran data, dua buah TCP harus sepakat untuk menjalin komunikasi atau melakukan pembentukan hubungan (*handshake*). Analoginya adalah serupa dengan percakapan telepon, dimana sebuah koneksi harus dilakukan terlebih dahulu sebelum kedua pihak dalam melakukan pembicaraan atau pertukaran informasi.

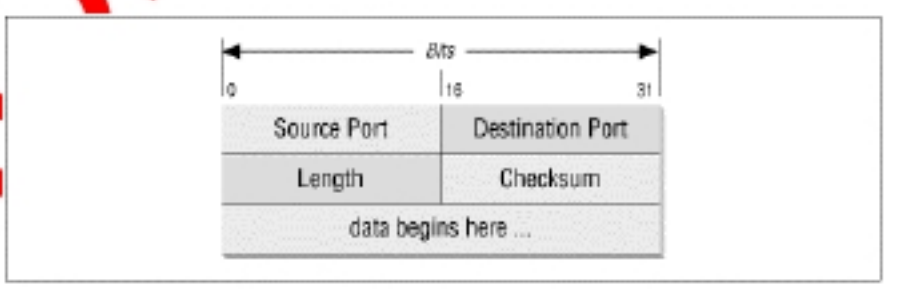
Reliabilitas, TCP menawarkan reliabilitas yang lebih baik dibanding UDP (*User Datagram Protocol*). Sejumlah mekanisme diterapkan untuk memberikan garansi penghantaran data, diantaranya :

- *Checksum*, seperti halnya UDP semua segmen TCP membawa fungsi *checksum* yang digunakan oleh penerima untuk mendeteksi seandainya terdapat kesalahan (*error*) pada header TCP atau data.
- *Duplicate data detection*, adalah sangat mungkin ada paket ganda dalam jaringan, sehingga TCP menjaga *track* byte yang diterimanya untuk membuang *copy* ganda data yang telah diterimanya.
- *Retransmission*, untuk menjamin suksesnya penghantaran data, TCP harus mengimplementasikan skema retransmisi terhadap data yang hilang atau rusak. Konfirmasi *acknowledgment* positif akan dikirim balik oleh penerima kepada pengirim data semula dan menyatakan bahwa data telah diterima dengan baik.
- *Sequencing*, aliran paket data yang mengalir pada jaringan komputer sangat mungkin dihantarkan dengan tidak berurutan berbeda dengan yang diharapkan. Salah satu tugas TCP adalah memperbaiki rangkaian segmen agar dapat menhantarkan arus *byte* data sampai kepada aplikasi tujuan.
- *Timers*, TCP merawat beragam informasi *timer* statis dan dinamis dalam penghantaran data. TCP pengirim akan menunggu *reply* penerima (berupa pesan *acknowledgment* dalam rentang waktu tertentu. Jika pesan tidak dikirim melewati batas waktu yang digariskan timer, pengirim berusaha mentransmisikan ulang segmen. (Rahmat, Rafiudin, 2005:85)

- Host B melakukan set control bit ACK untuk mengindikasikan bahwa byte selanjutnya dari Host A harus memuat data yang dimulai dengan urutan nomor x+1.
- Saat menerima ISN dan ACK dari Host B, Host A akan mengakhiri fase pembentukan koneksi dengan mengirimkan sebuah segmen *acknowledgment* final kepada Host B. (Rahmat, Rafiudin, 2005:86)

▪ **User Datagram Protocol (UDP)**

Menurut Rahmat, Rafiudin (2005:98) "lebih banyak program menggunakan UDP dari pada TCP hal ini dikarenakan alasan kecepatan, keringanan, dan reliabilitas transportasi data diantara host-host TCP/IP. Karakteristik dari UDP adalah tidak ada koneksi yang dibangun antara host-to-host (*connectionless*), UDP juga tidak memberikan garansi penghantaran data (*acknowledgment* atau *sequencing*). Program aplikasi yang menggunakan UDP bertanggung jawab sendiri dalam mewujudkan reliabilitas yang dibutuhkan untuk transport data. Karena sifatnya yang *connectionless* dan *unreliable* UDP digunakan oleh aplikasi-aplikasi yang secara periodik melakukan aktifitas tertentu (misalnya *query routing table* pada jaringan lokal)."



Gambar 2.5 Format Pesan UDP

Seperti terlihat pada gambar 2.5 yang menunjukan bahwa UDP menggunakan 16-bit *source port* dan *destination port* untuk mengirimkan data ke proses aplikasi

dengan benar. *Source* dan *Destination Port* memiliki fungsi sama seperti pada TCP. Datagram *length* berisi panjang dari datagram, sedangkan *checksum* berisi angka hasil perhitungan matematis yang digunakan untuk memeriksa kesalahan data. (Hunt, Craig, 1997:III)

d. Aplication Layer

Layer ini sudah termasuk semua proses yang mengunakan protokol pada *Transport Layer* untuk mengirimkan paket data. Banyak aplikasi protokol yang terdapat pada layer ini dan langsung berinteraksi dengan pengguna antara lain:

- Telnet; *Network Terminal Protocol* yang menyediakan layanan *remote login* pada jaringan komputer.
- FTP; *File Transfer Protocol* yang digunakan untuk interaksi *transfer file*
- SMTP; *Simple Mail Transfer Protocol* menyediakan layanan *Electronic Mail* (E-Mail)
- HTTP; *Hyper Text Transfer Protocol* yang merupakan layanan yang paling sering digunakan oleh user untuk membuka halaman web (*web pages*)

Disamping untuk pengguna pada tingkat *administrator* terdapat juga protokol aplikasi pada layer ini yang digunakan oleh tingkatan administrasi jaringan komputer (*Network Administrator*) untuk mengelola jaringan komputer antara lain DNS (*Domain Name Service*), OSPF (*Open Shortest Path First*), NFS (*Network File System*) (Hunt, Craig, 1997:III).

2.2.2 Protocol Number dan Port Number

Protocol Number merupakan *byte* tunggal pada word ke-3 dari datagram header. Nilai ini mengindentifikasikan protokol layer yang lebih tinggi yang akan menerima paket setelah proses IP telah selesai. Pada sistem operasi linux protokol

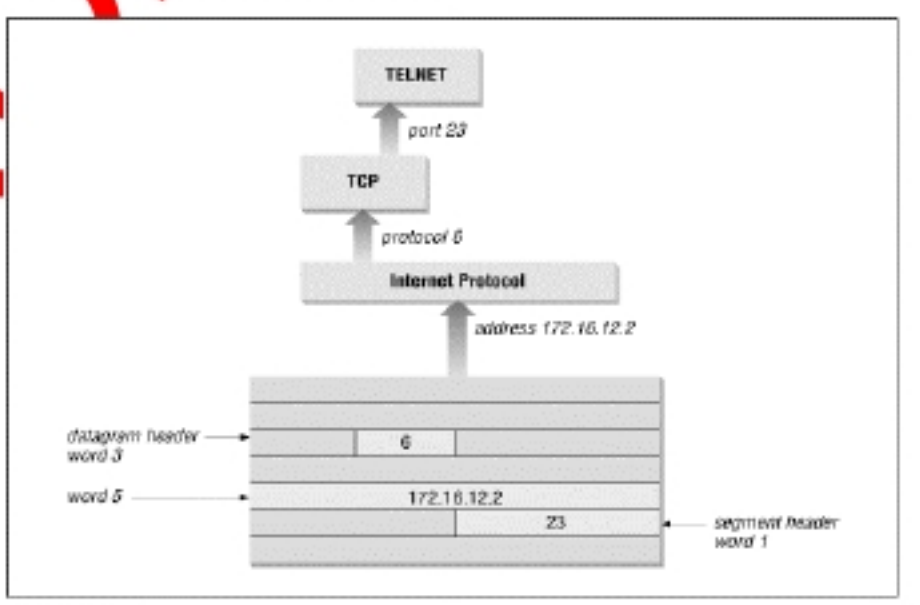
number dapat ditemukan didalam file */etc/protocols* seperti pada gambar 2.6 berikut :

```
% cat /etc/protocols
#ident  "@(#)protocols  1.2     90/02/03 SMI"   /* SVr4.0 1.1   */

#
# Internet (IP) protocols
#
ip      0       IP      # internet protocol, pseudo protocol number
icmp    1       ICMP    # internet control message protocol
ggp     3       GGP     # gateway-gateway protocol
tcp     6       TCP     # transmission control protocol
egp     8       EGP     # exterior gateway protocol
pup     12      PUP     # PARC universal packet protocol
udp     17      UDP     # user datagram protocol
hmp     20      HMP     # host monitoring protocol
xns-idp 22      XNS-IDP # Xerox NS IDP
rdp     27      RDP     # "reliable datagram" protocol
```

Gambar 2.6 Protocol Number pada Sistem Operasi Linux

Proses aplikasi layanan jaringan komputer (*network service*) diidentifikasi dengan nomor (*port number*) yang berisikan nilai 16-bit. *source port number* mengidentifikasikan proses pengiriman data sedangkan *destination port number* mengidentifikasi proses penerimaan data. Nomor port dibawah 255 (0 s/d 255) disebut *well-know port* yang berisikan nomor untuk proses aplikasi seperti http, ftp, telnet, dan sebagainya sedangkan nomor port 256 s/d 1024 digunakan untuk proses aplikasi untuk system seperti *rlogin* dan sebagainya. (Hunt, Craig, 1997:II)



Gambar 2.7 Protocol dan Port Number

Pada gambar 2.7 menunjukan sebuah datagram diarahkan ketujuannya sesuai dengan *destination address* pada word ke-5 dari datagram header. IP mengirimkan paket data dari datagram ke protokol yang tepat pada layer transport menggunakan *protocol number* pada word ke-3 dari datagram header. Word ke-1 pada pengiriman data ke layer transport berisikan *destination port number* dan mengkonfirmasikan ke protokol transport agar melewatkan data ke aplikasi proses yang dituju. Pada sistem operasi linux daftar *port number* terdapat pada file */etc/services* seperti pada gambar 2.8 berikut (Hunt, Craig, 1997:II) :

```
peanut% cat head -20 /etc/services
#ident  "@(#)services   1.13    95/07/28 SMI"   /* SVr4.0 1.8   */

#
# Network services, Internet style
#
tcpmux          1/tcp
echo            7/tcp
echo            7/udp
discard         9/tcp           sink null
discard         9/udp           sink null
systat          11/tcp          users
daytime         13/tcp
daytime         13/udp
netstat         15/tcp
chargen         19/tcp          ttytst source
chargen         19/udp          ttytst source
ftp-data        20/tcp
ftp             21/tcp
telnet          23/tcp
smtp            25/tcp          mail
```

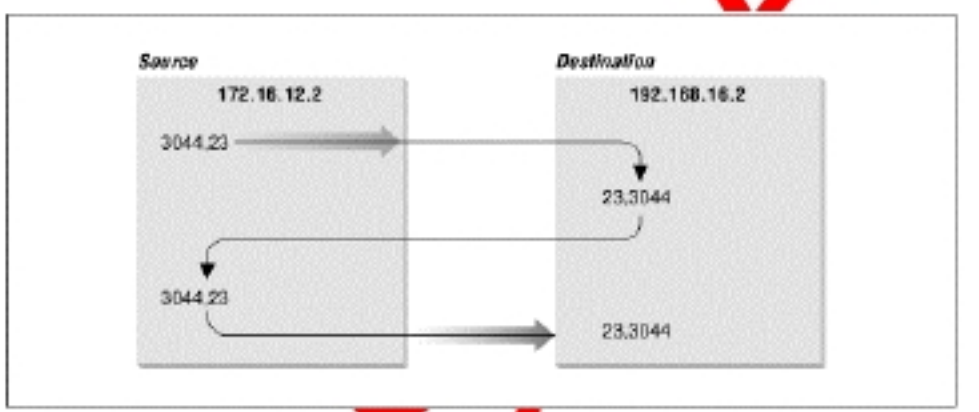
Gambar 2.8 Port Number pada Sistem Operasi Linux

2.2.3 Socket

Menurut Craig, Hunt (1997:II) "Socket disebut juga *Dynamically Allocated Port*, nomor port yang dialokasikan secara dinamis ketika sebuah proses membutuhkannya, sistem akan memastikan bahwa nomor port tersebut tidak digunakan oleh proses yang lain dan juga tidak sama dengan nomor-nomor dari

well-know port. *Dynamically allocated port* menyediakan fleksibilitas yang mendukung multiple user".

Gambar 2.9 menunjukkan perubahan seiring dengan proses *handshake* TCP/IP. Host asal akan menggunakan *source port* secara random pada misalnya 3044 dan mengirimkan segment dengan source port 3044 dan *destination port* adalah 23. Host tujuan akan menerima segment dan merespon kembali menggunakan nomor port 23 sebagai *source port* dan 3044 sebagai *destination port*.



Gambar 2.9 Dynamically Allocated Port (Socket)

Kombinasi dari *IP Address* dan *Port Number* disebut sebagai *Socket*. Socket secara unik mengidentifikasikan network proses tunggal secara keseluruhan dalam internet. (Hunt, Craig, 1997:II).

2.3 Quality of Service

Dalam buku *Quality of Service (QoS)* yang ditulis oleh Paul Ferguson dan Geoff Huston, didefinisikan bahwa "QoS adalah suatu pengukuran tentang seberapa baik jaringan komputer dan merupakan suatu usaha untuk mendefinisikan karakteristik dan sifat dari suatu layanan. QoS biasanya digunakan untuk mengukur sekumpulan atribut performansi yang telah dispesifikasikan dan biasanya diasosiasikan dengan suatu layanan. QoS didesain untuk membantu

pengguna jaringan dengan memastikan bahwa pengguna jaringan mendapatkan performansi yang handal dari aplikasi-aplikasi berbasis jaringan. QoS mengacu pada kemampuan jaringan untuk menyediakan layanan yang lebih baik pada trafik jaringan tertentu melalui teknologi yang berbeda-beda." (Ferguson, Paul, Huston, Geoff, 1998:1)

Komponen-komponen dari QoS antara lain:

- *Delay*, merupakan total waktu yang dilalui suatu paket dari pengirim ke penerima melalui jaringan. Delay dari pengirim ke penerima pada dasarnya tersusun atas *hardware latency*, delay akses, dan delay transmisi. Delay paling sering dialami oleh trafik yang lewat adalah delay transmisi, yang dapat dirumuskan sebagai berikut:

$$Delay = \frac{packet\_size \times 8}{line\_speed} \times 1000\ ms \qquad (1)$$

- *Jitter*, merupakan variasi dari delay *end-to-end*.. Jitter dapat dirumuskan sebagai berikut:

$$jitter = jitter + \frac{\left|D(i-1,i)\right| - jitter}{16} \qquad (2)$$

dengan

$$D(i,j)=(Rj-Ri)-(Sj-Si)=(Rj-Sj)-(Ri-Si) \qquad (3)$$

Keterangan:

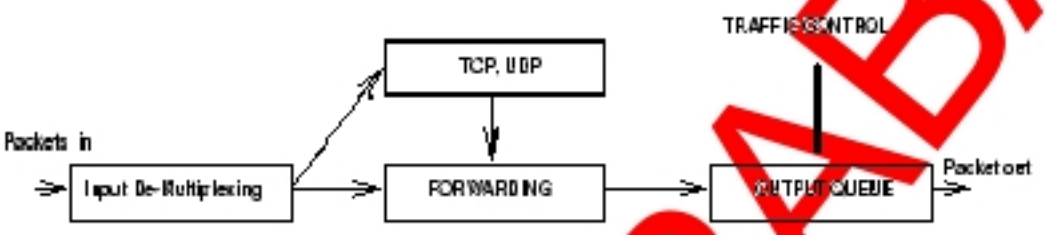
D = beda waktu yang timbul antara paket data ke i dengan paket data ke i-1.

R = waktu paket diterima

S = waktu paket dikirim

- *Bandwidth*, merupakan *rate transfer* data maksimal yang dapat diteruskan antara dua titik.
- *Throughput*, merupakan ukuran bandwidth aktual yang dipakai oleh sebuah alamat IP maupun layanan pada jaringan komputer.

Prinsip dasar dari implementasi manajemen bandwidth pada sistem operasi linux dapat dijelaskan pada gambar berikut:



Gambar 2.10 Prinsip Dasar Manajemen Bandwidth pada Linux

Gambar 2.10 menunjukkan bagaimana kernel memproses paket yang datang, dan bagaimana kernel mengolah paket-paket untuk dikirimkan ke jaringan. Input *demultiplexer* akan memeriksa apakah paket yang datang ditujukan untuk host lokal. Jika ya, maka paket akan dikirimkan ke *layer* yang lebih tinggi untuk pemrosesan lebih lanjut. Jika tidak, maka paket akan diteruskan ke blok *forwarding*. Blok *forwarding*, yang mungkin juga dapat menerima paket lokal dari *layer* yang lebih tinggi, akan melihat pada tabel *routing* dan menentukan *hop* selanjutnya bagi paket tersebut. Setelah itu, paket tersebut akan diantrikan untuk ditransmisikan pada interface output. Di titik inilah fungsi dari pengontrolan traffik pada sistem operasi linux akan diterapkan. Pengontrolan trafik pada sistem operasi linux dapat digunakan untuk membangun kombinasi yang kompleks dari disiplin antrian, kelas-kelas, dan filter-filter yang akan mengontrol paket-paket yang dikirimkan pada interface output (Ferguson, Paul & Huston, Geoff. 1998:55).

2.4 Disiplin Antrian

Menurut Shultz, Brian (2000:2) "setiap perangkat jaringan dapat diasosiasikan dengan suatu disiplin antrian. Dengan antrian, dapat ditentukan bagaimana data akan dikirimkan". Pada sistem operasi Linux, dikenal 2 macam disiplin antrian, yaitu disiplin antrian non kelas (*classless queueing discipline*) dan disiplin antrian dengan kelas-kelas (*classfull queuing discipline*).

2.4.1 Disiplin Antrian Non Kelas

Classless queuing discipline merupakan suatu disiplin antrian yang hanya dapat menunda ataupun menolak paket yang diterimanya. Ini dapat digunakan untuk membatasi trafik pada semua interface tanpa adanya pembagian-pembagian kelas. Pada sistem operasi Linux, dikenal beberapa jenis disiplin antrian non kelas, di antaranya adalah:

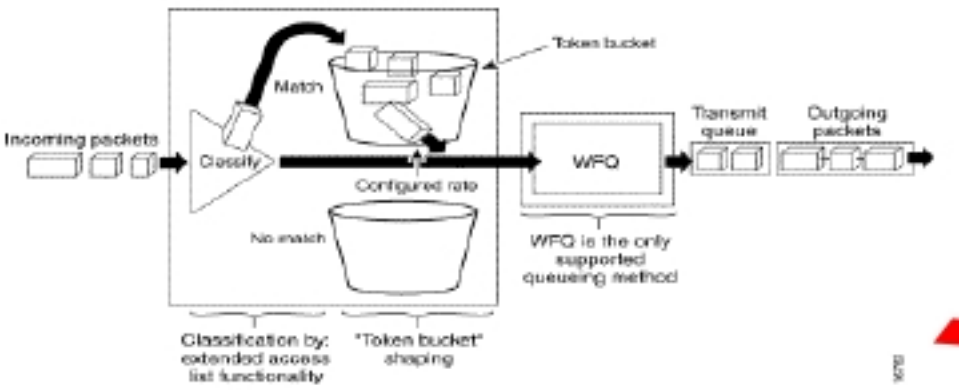
a. FIFO Queueing

FIFO queueing merupakan teknik antrian interface yang paling sederhana dan umum digunakan. Antrian ini dapat bekerja dengan baik apabila koneksi-koneksi dari layanan jaringan komputer yang sedang dilakukan tidak mengalami kemacetan. (Bogi, Nyoman, Ardiansyah, Andi, Pratania, Eka, 2003:13)

b. Token Bucket Filter

TBF merupakan suatu definisi formal dari suatu transfer rate. Antrian ini memiliki tiga buah komponen: ukuran burst, mean rate, dan interval waktu.

Mekanisme TBF terlihat pada gambar 2.11 berikut:



Gambar 2.11 Token Bucket Filter

Implementasi TBF terdiri dari sebuah buffer (*bucket*), yang secara konstan diisi oleh beberapa informasi virtual yang dinamakan token, pada rate yang spesifik (*token rate*). Parameter paling penting dari bucket adalah ukurannya, yaitu banyaknya token yang dapat disimpan. (Bogi, Nyoman, Ardiansyah, Andi, Pratania, Eka, 2003:13)

c. Stochastic Fairness Queueing

SFQ merupakan suatu implementasi sederhana dari keluarga algoritma *fair queueing*. SFQ kurang akurat apabila dibandingkan dengan yang lain. Kata kunci dari SFQ adalah *conversation* (aliran), yang sangat berhubungan dengan sebuah sesi TCP atau sebuah aliran UDP. Traffik akan dibagi ke dalam beberapa jumlah besar antrian FIFO, satu untuk setiap *conversation*. Traffik kemudian dikirim secara *round-robin*, dengan memberikan kesempatan kepada tiap sesi untuk mengirimkan data pada gilirannya. SFQ disebut stochastic karena tidak menyediakan sebuah antrian untuk setiap sesi, SFQ juga memiliki suatu algoritma yang membagi trafik melalui sejumlah antrian yang terbatas dengan menggunakan algoritma *hashing*. Dengan algoritma *hashing* sesi yang banyak dapat berakhir di bucket yang sama, yang akan membagi dua tiap sesi kesempatan untuk mengirimkan paket, sehingga membagi dua kecepatan efektif yang tersedia. (Bogi, Nyoman, Ardiansyah, Andi, Pratania, Eka, 2003:13)

2.4.2 Disiplin Antrian Dengan Kelas

Menurut Santosa, Budi (2003:7) "disiplin antrian dengan menggunakan kelas-kelas merupakan suatu disiplin antrian yang akan membagi trafik berdasarkan kelas-kelas. Hal ini sangat berguna apabila traffik yang berbeda-beda yang harus memiliki pembedaan penanganan. Ketika traffik memasuki suatu *classful queueing discipline*, maka paket tersebut akan dikirimkan ke kelas-kelas didalam *qdisc (queueing discipline)*, dengan kata lain paket tersebut perlu diklasifikasikan terlebih dahulu. Untuk menentukan apa yang harus dilakukan dengan sebuah paket yang datang, maka filter-filter akan digunakan".

Filter-filter yang terdapat pada *qdisc* tersebut akan menghasilkan suatu keputusan dan *qdisc* akan menggunakan hasil ini untuk mengantrikan paket ke salah satu kelas yang telah tersedia. Setiap sub kelas dapat mencoba filter-filter yang lainnya untuk melihat apakah ada instruksi lain yang berlaku. Jika tidak, maka kelas akan mengantrikan paket tersebut ke *qdisc* yang dimilikinya. Di samping memiliki suatu *qdisc*, kebanyakan dari disiplin antrian menggunakan kelas-kelas juga menerapkan fungsi *shaping*. Hail ini sangat berguna untuk melakukan penjadwalan paket dan pengontrolan kecepatan (*rate*) sekaligus. (Santosa, Budi, 2003:9)

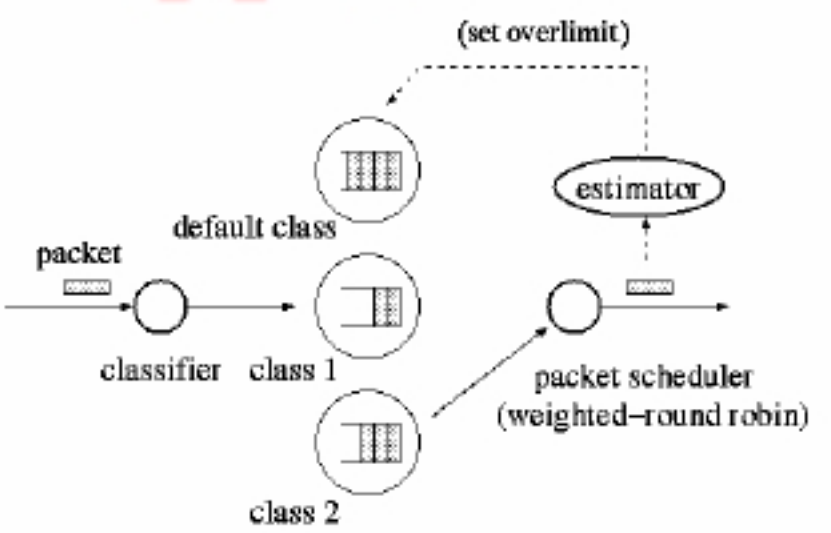
Beberapa contoh *classful queueing discipline* pada sistem operasi Linux antara lain :

a. Class Based Queueing

CBQ dapat menerapkan pembagian kelas dan membagi bandwidth melalui struktur kelas-kelas secara hirarki. Setiap kelas memiliki antriannya masing-

masing dan diberikan porsi bandwidth-nya. Sebuah kelas *leaf* (cabang) dapat meminjam bandwidth dari kelas *parent* (induk) jika terdapat kelebihan bandwidth. Gambar 2.12 menunjukkan komponen dasar dan mekanisme dari CBQ:

- *Classifier;* akan mengarahkan paket-paket yang datang ke kelas-kelas yang bersesuaian.
- *Estimator;* akan mengestimasi bandwidth yang sedang digunakan oleh sebuah kelas. Jika sebuah kelas telah melampaui *limit* (batasan) yang telah ditentukannya, maka estimator akan menandai kelas tersebut sebagai kelas yang *overlimit*.
- *Scheduler;* akan menentukan paket selanjutnya yang akan dikirim dari kelas-kelas yang berbeda-beda, berdasarkan pada prioritas dan keadaan dari kelas-kelas. *Weighted Round Robin Scheduling* digunakan antara kelas-kelas dengan prioritas yang sama. (Santosa, Budi, 2003:12)



Gambar 2.12 Class Based Queueing

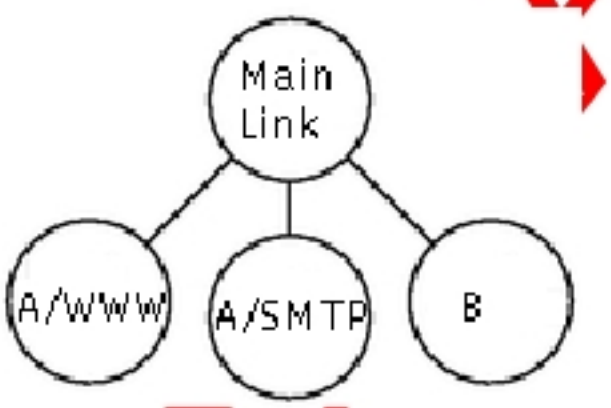
b. PRIO (Priority)

Priority Queueing memungkinkan untuk menentukan bagaimana trafik diprioritaskan dalam suatu jaringan. Dengan menentukan serangkaian filter berdasarkan karakteristik dari paket, traffik ditempatkan pada suatu antrian.

Antrian dengan prioritas yang lebih tinggi dilayani lebih dahulu, kemudian antrian yang lebih rendah lagi dilayani sesuai urutannya. Jika antrian dengan prioritas paling tinggi selalu penuh, maka antrian ini akan secara terus menerus dilayani dan paket-paket dari antrian yang lain akan dibatalkan (*drop*). (Santosa, Budi, 2003:13)

c. Hierarchical Token Bucket

HTB merupakan salah satu disiplin antrian yang memiliki tujuan untuk menerapkan *link sharing* secara tepat dan adil seperti terlihat pada gambar 2.13.



Gambar 2.13 Link Sharing HTB

Dalam konsep *link sharing*, jika suatu kelas meminta kurang dari jumlah layanan yang telah ditetapkan untuknya, maka sisa bandwidth akan didistribusikan ke kelas-kelas yang lain yang meminta layanan. HTB menggunakan TBF (*Token Bucket Filter*) sebagai *estimator* yang sangat mudah diimplementasikan. (Wijaya, Hanny, 2003:1)

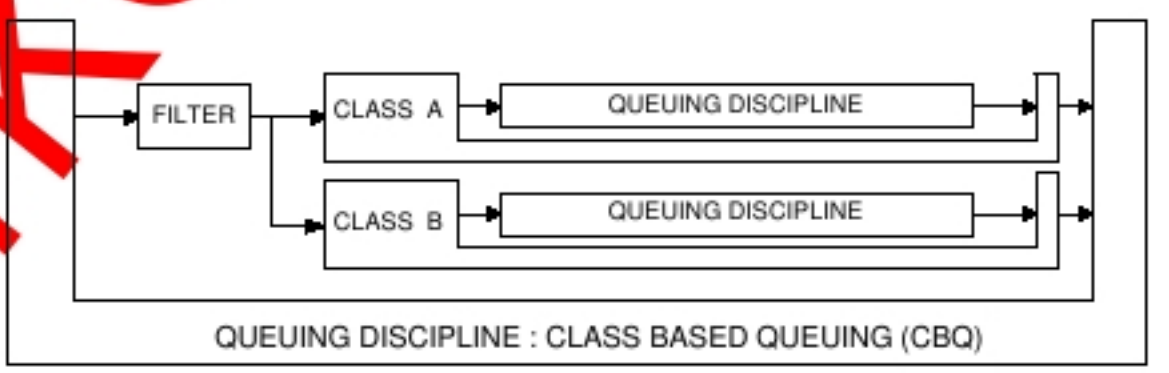
2.5 Class Based Queueing (CBQ)

Menurut Sally Floyd (1996:2) "CBQ merupakan suatu mekanisme penjadwalan, bertujuan menyediakan *link sharing* antar layanan yang menggunakan jalur fisik yang sama, dan sebagai acuan untuk membedakan trafik yang memiliki prioritas-prioritas yang berlainan. Dengan CBQ, setiap layanan

dapat mengalokasikan bandwidth miliknya untuk berbagai jenis trafik yang berbeda, sesuai dengan pembagiannya yang tepat untuk masing-masing trafik. CBQ berinteraksi dengan *link sharing* memberikan keunggulan yaitu pemberian bandwidth yang tidak terpakai bagi cabang kelas (*leaf class*) sebelum diberikan kepada layanan-layanan lain".

Blok-blok utama CBQ adalah sebagai berikut :

- *Classifier*, bekerja dengan cara yang sama dengan filter paket yaitu mengekstrak aliran informasi dari suatu paket (untuk IP misalnya *Ipsrc, Ipdest, PROTO, PORTsrc, PORTdesc*) kemudian menempatkannya pada kelas yang sesuai
- *General Scheduler*, merupakan mekanisme penjadwalan bertujuan untuk membagi bandwidth saat seluruh kelas memiliki antrian paket. *General Scheduler* menjamin hak kuantitas layanan untuk tiap cabang kelas, dengan membagikan bandwidth sesuai dengan alokasinya masing-masing.
- *Link-sharing Scheduler*, yang bertujuan membagikan bandwidth yang tidak terpakai sesuai dengan struktur *link sharing*-nya.
- *Estimator*, merupakan bagian blok umpan balik bagi mekanisme penjadwalan CBQ.



Gambar 2.14 Blok CBQ Traffic Control

CBQ pada sistem operasi linux untuk pengendalian trafik terbagi atas tiga bagian seperti terlihat pada gambar 2.14. Penjelasan dari masing-masing bagian adalah sebagai berikut :

1. *Queueing Discipline*, menggambarkan cara suatu NIC (*Network Interface Card*) memperlakukan paket dalam antrian, yaitu berupa proses pelayanan terhadap suatu paket dalam antrian untuk mendapatkan giliran pengiriman.
2. *Classes*, bertanggung jawab terhadap alokasi *bandwidth* dengan kriteria yang telah ditentukan.
3. *Filters*, merupakan bagian pemilah trafik, mengklasifikasikan paket dan menempatkannya pada antrian yang sesuai. (Santosa, Budi, 2003:13)

2.5.1 CBQ Algoritma

CBQ menggunakan beberapa metode algoritma yang sudah terintegrasi didalamnya seperti *algoritma shaping*, *algoritma link* sharing, dan *algoritma classification*. Berikut penjelasan dari masing-masing algoritma :

a. Algoritma shaping

Ketika membentuk suatu kecepatan aliran data antara 10Mbit/s sampai 1Mbit/s, maka koneksi tersebut akan mengalami apa yang disebut waktu tunggu (*idle-time*), namun walaupun hal itu tidak terjadi tetap akan dipaksakan agar mengalami *idle*. Selama dalam beroperasi, waktu tunggu yang efektif dapat diukur dengan menggunakan perhitungan rata-rata yang bersifat eksponensial EWMA (*Exponential Weighted Moving Average*), dimana akan mempertimbangkan paket yang baru masuk dan paket yang sudah lama masuk. Perhitungan waktu tunggu dikurangi EWMA akan menghasilkan rata-rata waktu tunggu (*avgidle*), sebuah koneksi yang berisikan aliran data mempunyai suatu *avgidle* yang bernilai 0 (nol)

hal ini dikarenakan paket data yang tiba tepat pada selang waktu yang dihitung. Sebuah koneksi yang dipenuhi oleh aliran data yang berlebihan (*overload*) akan menghasilkan nilai negatif pada *avgidle*, hal ini yang akan diatur oleh CBQ dan mengeluarkan pesan *overlimit*. Dan sebaliknya jika suatu koneksi dalam keadaan kosong kemungkinan menghasilkan jumlah *avgidle* yang cukup besar, dimana efek dari keadaan tersebut akan memberikan kecepatan bandwidth yang besar, untuk mencegah hal ini parameter *maxidle* dari CBQ harus diatur. (Floyd, Sally, 1996:10)

b. Algoritma link sharing

Ketika dalam kondisi *dequeueing* (keluar dari antrian) untuk mengirimkan paket ke media jaringan, CBQ akan memutuskan bagian yang mana yang akan dikirimkan terlebih dahulu. Algoritma ini hampir sama dengan algoritma *Round Robin* dimana masing-masing kelas dengan paketnya akan memperoleh kesempatan yang sama untuk mengirimkan paket datanya. Proses akan dengan mencari kelas yang membawa paket data dengan level prioritas yang paling tinggi terlebih dahulu dan akan berlanjut terus hingga tidak ada lagi paket data yang dikirim. (Floyd, Sally, Jacobson, Van, 1998:3)

c. Algoritma Classification

Didalam CBQ bisa saja terdapat banyak kelas (*class*), masing-masing kelas berisi *qdisc* secara default akan diset ke *tcpfifo*. Paket-paket data akan dimasukkan kedalam kelas-kelas. Algoritma ini mempertegas bahwa sebuah paket pasti akan berakhir (terkirim) kesuatu tempat. (Floyd, Sally, Jacobson, Van, 1998:4)

2.5.2 CBQ Parameter

Untuk melakukan mekanisme penjadwalan dan pembagian kelas CBQ mempunyai parameter-parameter penting antara lain :

a. *Device* Parameter

Parameter *device* memiliki tiga opsi yakni *ifname*, *bandwidth*, *weight*. dimana:

- *Ifname* merupakan nama dari interface (NIC) yang akan dikontrol trafiknya misalnya eth0 (*ethernet* ke-0).
- *Bandwidth* merupakan ukuran kecepatan pengiriman data secara fisik dari NIC misalnya ada NIC yang mempunyai ukuran kecepatan 10Mbit/s, 100Mbit/s, maupun hanya 2Mbit/s.
- Sedangkan *Weight* merupakan parameter *tuning* yang didapat dari jumlah bandwidth dibagi 10 (*weight = bandwidth / 10, is the rule of thumb*).

b. *Class* Parameter

- *Rate*, besarnya kapasitas bandwidth yang dialokasikan pada kelas yang bersangkutan. Ukurannya bisa Kbit/s (*Kilo bit per second*), Mbit/s (*Mega bit per second*), atau bps (*bit per second*). Namun jika tidak diberikan ukuran maka secara default ukurannya akan diset bit per second (bps).
- *Weight*, parameter *tuning* yang berguna untuk proses *Weighted Round Robin* (WRR) yaitu memberikan kesempatan yang proporsional dalam mengirimkan paket berdasarkan pada alokasi *bandwidth*-nya (*weight = rate / 10, is the rule of thumb*)

- *Prio*, merepresentasikan prioritas yang diberikan pada suatu kelas (*range*-nya antara 1 – 8)
- *Parent*, spesifikasi sebuah kelas pada queuing discipline pada sebuah hirarki atau merupakan identitas kelas yang akan dipakai pada saat membangun hirarki kelas-kelas dibawahnya.
- *Leaf* (none/tbf/sfq), parameter yang menentukan suatu kelas akan diset menjadi dari disiplin antrian menjadi CBQ, secara default akan diset tbf (*token bucket filter*).
- *Bounded* (yes/no), jika "*yes*" maka kelas yang bersangkutan tidak diijinkan untuk meminjam bandwidth dari parent-nya, jika "*no*" maka kelas diijinkan meminjam bandwidth dari kelas parentnya pada kondisi *overlimit*.
- *Isolated* (yes/no), jika "*yes*" maka kelas yang bersangkutan tidak akan meminjamkan bandwidthnya kepada lain kelas, jika "*no*" maka kelas boleh meminjamkan *bandwidth*nya.

c. *SFQ qdisc* Parameter

Stochastic Fairness Queuing (SFQ) memiliki kemampuan membagi setiap paket data yang diterima dalam jumlah yang sama rata, setiap paket data yang telah terbagi dimasukkan ke dalam suatu antrian dan menunggu dikeluarkan oleh penjadwalan, antrian dikeluarkan dengan algoritma *round robin*. Berikut ini adalah opsi dari teknik antrian SFQ:

- *Perturb* nilai[detik], nilai ini akan di konfigurasi ulang secara otomatis tergantung dari nilai yang diberikan [detik].

– *Quantum* nilai[bytes], jumlah paket *data stream* yang dizinkan untuk dikeluarkan sebelum antrian lain diproses.

d. *TBF qdisc* Parameter

Token bucket filter (TBF) membatasi bandwidth dengan metode *shape* & *drop*, prinsip kerja menggunakan aliran token yang memasuki bucket dengan kecepatan (*rate*) konstan, jika token dalam bucket habis maka paket data akan di antri dan kelebihannya dibuang, setiap paket data yang dikeluarkan identik dengan token. Token dalam bucket akan lebih cepat habis jika aliran paket data melampaui kecepatan token memasukki bucket, jadi kita asumsikan bahwa trafik melebihi batas konfigurasi. TBF juga mempunyai beberapa parameter yakni:

– *Limit dan latency*, *limit* merupakan jumlah byte yang dapat diantri sebelum token tersedia, sedangkan latency adalah lama waktu (dalam mili detik [msec]) paket dapat diantri.

– *Burst/buffer/maxburst*, kapasitas bucket dalam byte, paket data yang melebihi nilai ini akan dibuang atau mengalami penundaan.

– *Peakrate*, batas maksimum rate menangani lonjakan jumlah bandwitdh sesaat dengan syarat paket data tidak boleh melebihi kapasitas bucket dan MTU (*Maximum Transfer Unit*).

e. *Filter* Parameter

– *Rule*, parameter untuk *tc filter rule* yang diikuti oleh *u32* untuk klasifikasi paket data.

- *Realm*, parameter untuk *tc filter rule* yang diikuti oleh *route* untuk klasifikasi paket data.
- *Mark*, parameter untuk *tc filter rule* yang diikuti oleh *fw* untuk klasifikasi paket data

f. *Time Ranging* Parameter

Parameter ini menjadikan kelas-kelas CBQ diset menurut waktu dan hari disesuaikan dengan besarnya bandwidth. (Floyd, Sally, 1996:17)

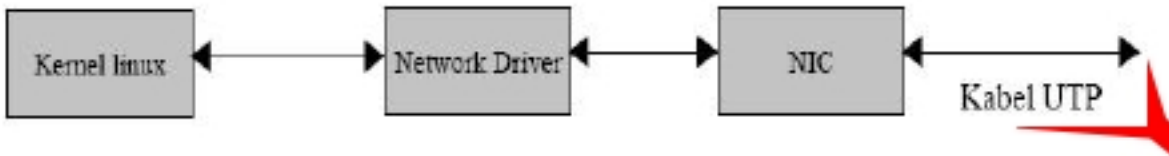
2.6 Linux Kernel

Kernel merupakan inti dari sistem operasi yang berisikan instruksi-instruksi yang bertindak sebagai mediator (penghubung) antara hardware dengan software. Kernel merupakan program yang pertama kali dijalankan yang berfungsi sebagai penghubung (*interface*) antara hardware dan user level program, mengatur penggunaan *memory*, *task switching*, melayani *read* dan *write* pada hardware. Kernel baru pada umumnya menawarkan dukungan yang lebih banyak terhadap berbagai jenis hardware, memiliki manajemen proses yang lebih baik, berjalan lebih baik dari versi sebelumnya, dan lebih stabil karena adanya perbaikan pada bug-bug yang ditemukan pada versi sebelumnya. (Hermawan, Chandra, 2001:5).

2.6.1 Linux Kernel Traffic Control

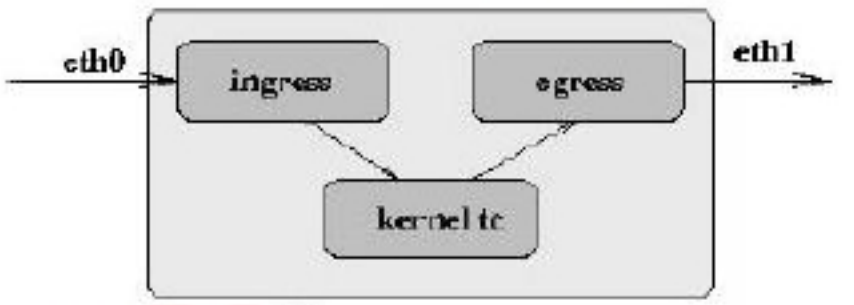
Trafik jaringan berhubungan dengan paket data yang dibangkitkan oleh kartu ethernet pada komputer. Pada gambar 2.15 paket data yang dikirimkan oleh komputer lain diterima NIC (*Network Interface Card*), kemudian teruskan oleh driver kartu ethernet (*Network Driver*) kebagian kernel linux untuk diproses. Proses ini hanya mengatur paket data yang keluar maupun masuk melalui satu

kartu ethernet. Kernel linux yang bertanggung jawab mengatur aliran data disebut *Kernel Traffic Control*.



Gambar 2.15 Arsitektur Linux Kernel, Network Driver, dan NIC

Pada dasarnya struktur kernel *traffic control* memiliki tiga bagian, yang pertama adalah *Ingress*. Paket data yang memasuki komputer gateway akan diproses terlebih dahulu oleh qdisc *ingress*, sebelum diberikan ke *kernel traffic control*. *Ingress* merupakan teknik antrian untuk mengatur data yang memasuki NIC (*Network Interface Card*).



Gambar 2.16 Struktur Kernel Traffic Control

Gambar 2.16 menunjukkan paket data diterima dari NIC 1 (*eth0*) kemudian dimasukkan ke dalam antrian *ingress* dan diproses oleh kernel *traffic control* biasanya *ingress* dipakai untuk mengendalikan traffik *upload / uplink*. Kemudian perangkat *Egress* dipergunakan untuk mengendalikan paket data yang keluar dari NIC 2 (*eth1*), sehingga trafik *download* oleh komputer client dapat dibatasi sesuai konfigurasi. (Santosa, Budi, 2003:8)

2.7 Squid

Squid adalah salah satu aplikasi *proxy* yang berjalan pada sistem operasi Linux, squid juga dapat membantu untuk menghemat pemakaian bandwidth dengan dua cara antara lain :

- Cara pertama dengan menyimpan halaman, gambar, dan obyek lain yang telah di*download* sebelumnya kedalam *memory* atau *disk*, sehingga jika seseorang melakukan *request* (permintaan) dengan halaman yang sama proxy akan mengambil halaman yang telah disimpan tersebut.
- Yang kedua, terpisah dari penyimpanan *cache* yang umumnya dilakukan oleh *proxy*, squid mempunya fasilitas spesial yang disebut *delaypools*. *Delaypools* dengan ACL-nya (*Access Control List*) sangat mungkinkan sekali untuk membatasi aliran data internet (*internet traffic*) dengan cara yang masuk akal tergantung kata kunci yang ada di URL (*Uniform Resouce Locator*), kata kunci tersebut bisa berupa ekstension file misalnya mp3, exe, avi, mpeg dan lain-lain. Dengan delaypools bisa juga membatasi pemakaian bandwidth dengan menggunakan tingkatan misalnya tinggi, rendah, staff, mahasiswa, dan sebagainya. (Wijaya, Hanny, 2003:5)

2.8 IPTables

Aplikasi *packet filtering* pada sistem operasi Linux dengan versi kernel 2.4. keatas adalah IPTables yang dapat juga digunakan sebagai *firewall* dan pada kernel Linux versi sebelumnya masih menggunakan *IPChains*. IPTables merupakan perbaikan dan peningkatan fitur dari versi sebelumnya. Dukungan untuk *statefull filtering* sudah ada hal ini ditujukan untuk meningkatkan performa laju aliran data. (Christian, Diyan, 2000:8)

Paramter-parameter dasar untuk IPTables antara lain:

-N untuk membuah CHAINS baru (NEW)

-F untuk menghapus semua rule yang ada (FLUSH)

-P untuk mengubah *policy* default bila suatu paket tidak cocok dengan rule yang ada (POLICY)

-X untuk menghapus *non-builtin* CHAINS

-A untuk menambah rule baru dibawah CHAINS yang sudah ada (APPEND)

-I untuk mengisikan rule diantara CHAINS yang sudah ada (INSERT)

-D untuk menghapus satu rule (DELETE)

-R untuk mengganti rule yang sudah ada (REPLACE)

-Z menginisialisasi counter kembali ke nilai 0 (ZERO)

Sedangkan parameter untuk mengubah (modifikasi) rule adalah sebagai berikut:

-p <*protocol*> protokol yang akan diperiksa

-s <*source IP*> alamat IP asal paket data

-d <*destination IP*> alamat IP tujuan paket data

-i <*interface*> input paket dari *network interface*

-o <*interface*> output paket dari *network interface*

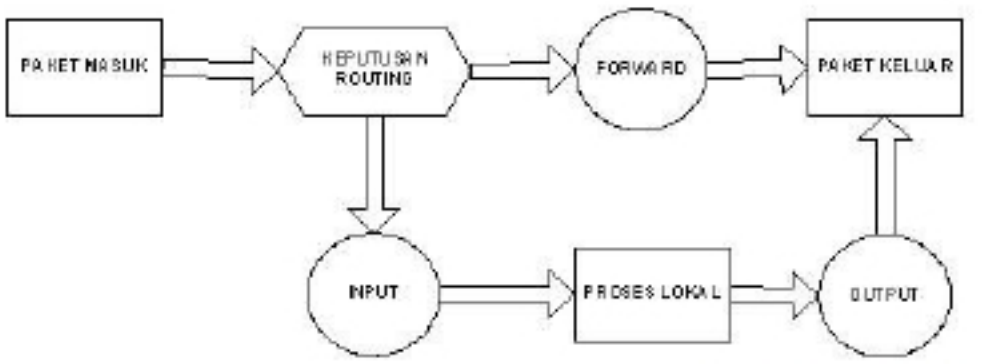
-j <*target*> kebijakan akhir dari paket bila cocok dengan rule

-m <*match*> paket cocok dengan *match* yang telah ditentukan

-t <*table*> tabel yang terdiri dari *filter* untuk penyaringan, *nat* untuk *network address translation*, dan *mangle* untuk modifikasi paket.

-f pencarian paket yang mengalami *fragmentasi*

Agar lebih memperjelas alur dari aturan-aturan (*rules*) yang terdapat pada iptables berikut ini adalah gambar alur aturan yang terdapat pada aplikasi firewal iptables



Gambar 2.17 Aturan Iptables

Pada gambar 2.17 jika ada paket data yang masuk maka akan diambil keputusan *routing*, apakah paket tersebut diinputkan atau di-*forward*-kan. Jika keputusan paket tersebut diinput maka paket data tersebut akan menjalani proses lokal dan kemudian menghasilkan output dan paket tersebut dikeluarkan, sedangkan jika keputusan *routing* adalah di-*forward*-kan maka paket data tersebut hanya dilewatkan saja.

IPTables mempunyai beberapa fitur yang berbentuk *modular*, berarti diperbolehkan untuk menambahkan fitur baru kedalam fitur yang sudah ada salah satunya adalah dengan "*connection tracking*" dimana paket data dari suatu koneksi akan dimonitor sekali ketika paket diteruskan tanpa harus melihat informasi paketnya, karena untuk koneksi-koneksi selanjutnya dianggap sebagai suatu paket yang diperbolehkan untuk dilewatkan.

Koneksi ini akan diingat oleh firewall kedalam suatu state table dan akan berhenti untuk diperiksa setelah koneksi firewall mengetahui *flags FIN* untuk mengakhiri koneksi dikirimkan oleh host yang bersangkutan. IPTables juga

mendukung mekanisme NAT (*Network Address Translation*), dimana suatu paket informasi header mengenai tujuan dan/atau asal IP dapat dimodifikasi untuk suatu tujuan tertentu. Didalam IPTables juga disertakan suatu mekanisme *packet limitting*, dimana suatu paket data dapat dikontrol lajunya dalam suatu interval waktu tertentu. (Christian, Diyan, 2002:9).

2.9 Apache Web Server

Web Server yang digunakan untuk membangun aplikasi ini adalah Apache Apache adalah sebuah HTTP server yang didesain sebagai pengganti dari NCSA server versi 1.3 atau 1.4. Yang didalamnya banyak terdapat perbaikan bug yang ada pada NCSA server, dan mempunyai banyak fitur yang banyak dibutuhkan oleh HTTP protokol. Apache banyak digunakan untuk lingkungan komersial maupun untuk tujuan yang bersifat eksperimental, kerena sifatnya yang *robust*, reliable dan bersifat *opensource*

2.10 OpenSSL

Untuk mendukung kinerja web server Apache pada tingkat keamanan, aplikasi ini menggunakan SSL (*Secure Socket Layer*) yang merupakan mekanisme "pembungkus" paket data yang mengalir antara server dan client dengan teknik enkripsi data. Pada apache web server komunikasi antara browser dan server web dapat menggunakan protokol SSL yang terdapat pada aplikasi OpenSSL dengan mengaktifkan module-nya (*mod_ssl*) yang terdapat pada file konfigurasi apache (*httpd.conf*), disamping itu protokol SSL membutuhkan port tersendiri pada level sistem operasi yakni port 443 HTTPS (*Hyper Text Transfer Protocol Secure*).
Untuk mengaktifkan protokol SSL server web diharuskan terlebih dahulu memasang file sertifikat (*certificate file*), yang berisikan kunci public dan kunci

pribadi server. File sertifikat ini sendiri biasanya dapat diperoleh dengan cara membelinya pada suatu badan/organisasi yang disebut dengan *CERTIFICATE AUTHORITY (CA)*, ada 2 (dua) badan CA yg cukup terkenal yaitu *Verisign* (*http://www.verisign.com*) dan *Thawte* (*http://www.thawte.com*)

2.11 Landasan Teori Pembuatan Web

Halaman – halaman web dibuat dengan menggunakan CGI-Perl dan PHP yang berbasis Linux. CGI-Perl merupakan bahasa pemrograman yang mempunyai kemampuan mengakses sampai ke library dari sistem operasi linux dan juga dapat membentuk suatu tampilan berdasarkan permintaan terkini sedangkan PHP untuk mengkondisikan pengaturan grafik. Kode CGI-Perl dan PHP juga bisa berkomunikasi dengan database dan melakukan perhitungan – perhitungan yang kompleks

2.11.1 CGI-Perl

CGI (*Common Gateway Interface*) merupakan suatu standar yang memungkinkan sembarang program yang ditulis dengan menggunakan bahasa pemrograman apa saja berinteraksi dengan web server dan dapat dipanggil melalui web client. CGI berkerja atas dasar statndard HTML, dan HTTP. HTML (*HyperText Markup Language*) adalah standar informasi yang berbasis *hypertext* yang dipakai pada Web. CGI kode dibuat dengan bahasa pemrograman Perl biasanya diletakkan dalam direktori bernama cgi-bin

Cara kerja CGI adalah sebagai berikut :

- Aplikasi CGI, yang berupa file skrip, diletakkan di lokasi direktori yang dapat diakses oleh web server.

- Jika ada permintaan terhadap aplikasi dari klien, web server akan membaca skrip dan menjalankan program.
- Web server memberi program ini masukan (termasuk masukan yang diberikan klien) dari masukan standar (*stdin*) serta beberapa lingkup (*environment variable*).
- Web server menunggu program berjalan hingga selesai lalu menangkap keluaran standar program (*stdout*).
- Keluaran inilah yang diberikan ke klien.

Kelebihan dari CGI adalah sebagai berikut :

- Skrip CGI dapat ditulis dalam bahasa apa saja, namun barangkali sekitar 90% program CGI yang ada ditulis dalam Perl.
- Protokol CGI yang sederhana, serta
- Kefasihan Perl dalam mengolah teks, menjadikan menulis sebuah program CGI cukup mudah dan cepat.
- Meski tertua, hingga saat ini menurut survei dari Netcraft (www.netcraf.com) sekitar 70% aplikasi di web masih menggunakan CGI. Ini berarti, lebih dari separuh situs web dinamik yang ada dibangun dengan CGI

Kecepatan (*skalabilitas*) dimana untuk menghasilkan keluaran program CGI, overhead yang harus ditempuh cukup besar antara lain :

- Web server terlebih dahulu akan menciptakan sebuah proses baru dan menjalankan interpreter Perl.

Perl kemudian mengkompilasi (*compile*) script CGI tersebut, baru kemudian menjalankan skrip. (Haryanto, Steven. Januari 2000)

2.11.2 PHP

PHP adalah server side scripting environment yang dapat digunakan untuk membuat dan menjalankan aplikasi – aplikasi di web server agar menjadi lebih interaktif dan *programmable*. Dengan PHP aplikasi aplikasi yang ada di web server benar benar akan dijalankan di web server tanpa mengharuskan adanya tambahan atau syarat tertentu untuk sisi client (*web browser*). PHP biasanya dijadikan sebagai module dalam suatu web server agar bisa mengeksekusi file – file PHP yang tersedia di web server. PHP dapat berjalan di hampir seluruh platform, open source dan berlisensi GNU Public License (GPL).

Sebagai tambahan untuk memanipulasi isi dari halaman web, PHP juga dapat mengirimkan HTTP header yang dapat digunakan untuk setting cookies, mengatur proses authentikasi dan meredirect user. PHP juga mempunyai koneksi dengan banyak database termasuk dengan ODBC serta berintegrasi dengan beragam library eksternal yang membantu web developer untuk melakukan semuanya.PHP menyatu dengan halaman web sehingga tidak dibutuhkan aplikasi khusus untuk membuatnya. Secara sintaks PHP serupa dengan bahasa C dan Perl. Web developer tidak harus mendeklarasikan variable sebelum menggunakannya dan dengan PHP mudah untuk membuat array dan hash (array berassosiasi).

2.11.3 MySQL

Database yang digunakan dalam pembuatan web ini adalah MySQL. MySQL adalah *Relational Database Management System* (RDBMS) yang didistribusikan secara gratis dibawah lisensi GPL (*General Public License*). Sebagai database server yang memiliki konsep database modern, MySQL

memiliki banyak sekali keistimewaan. Keistimewaan/kehandalan yang dimiliki oleh database MySQL adalah sebagai berikut :

- Portability

  MySQL dapat berjalan stabil pada berbagai sistem operasi diantaranya adalah Windows, Linux, FreeBSD, Mac OS X Server, Solaris, Amiga, HP UX dan masih banyak lagi.

- Open Source

  MySQL dididtribusikan secara open source, dibawah lisensi GPL sehingga kita dapat menggunakannya secara gratis.

- Multiuser

  MySQL dapat digunakan oleh beberapa user dalam waktu yang bersamaan tanpa mengalami masalah atau konflik.

- Performance Tuning

  MySQL memiliki kecepatan yang tinggi dalam menangani query sederhana, dengan kata lain dapat memproses lebih banayk SQL per satuan waktu.

- Column Types

  MySQL memiliki tipe kolom yang sangat kompleks.

- Command dan Functions

  MySQL memiliki operator dan fungsi secara penuh yang mendukung perintah SELECT dan WHERE dalam query.

- Security

  MySQL memiliki beberapa Lapisan sekuritas seperti level subnetmask, nama host dan ijin akses user dengan sistem perijinan yang mendetail serta password terenkripsi.

- Scalability dan Limits

  MySQL mampu menangani database dalam skala besar, dengan jumlah record lebih dari 50 juta dan 60 ribu tabel serta 5 milliar baris. Selain itu, batas indeks yang dapat ditampung mencapai 32 indeks pada tiap tabelnya.

- Connectivity

  MySQL dapat melakukan koneksi dengan Client menggunakan protokol TCP/IP, Unit soket (UNIX) atau Named Pipes (NT).

- Localisation

  MySQL dapat mendeteksi pesan kesalahan pada client dengan menggunakan lebih dari dua puluh bahasa.

- Interface

  MySQL memiliki interface terhadap berbagai aplikasi dan bahasa pemrograman dengan menggunakan fungsi API.

- Clients and Tools

  MySQL dilengkapi dengan berbagai tool yang dapat digunakan untuk administrasi database, dan pada setiap tool yang ada disertakan petunjuk on line.

- Struktur Tabel

  MySQL memiliki struktur tabel yang lebih fleksibel dalam menangani ALTER TABLE, dibandingakan database lainnya semacam PostgresSQL ataupun Oracle.