BAB II
LANDASAN TEORI

Pada bab ini akan dibahas mengenai teori dan komponen penunjang yang akan digunakan dalam perancangan dan analisa pengaturan *bandwidth* PC router menggunakan kernel GNU/linux dan freeBSD. Pembahasannya berisi tentang dasar-dasar jaringan, komponen fisik dalam jaringan, *firewall*, teknik antrian, sistem operasi dan kernel.

2.1 Dasar-dasar Jaringan Komputer

Jaringan komputer adalah sekelompok komputer otonom (*stand alone*) yang saling berhubungan antara satu dengan yang lainnya menggunakan protokol komunikasi melalu media komunikasi sehingga dapat saling berbagi informasi, data program aplikasi dan perangkat keras seperti *printer*. Selain itu jaringan komputer bisa diartikan sebagai kumpulan sejumlah terminal komunikasi yang berada diberbagai lokasi yang terdiri dari lebih satu komputer yang saling berhubungan[3].

Jaringan komputer dapat dibangun dan diolah dengan mudah jika pengguna yang akan membangun jaringan tersebut memahami konsep jaringan, terutama dalam hal tipe dan arsitektur suatu jaringan komputer. Hal ini penting karena tipe dan struktur sebuah jaringan menentukan perangkat apa yang harus disediakan untuk membangun jaringan tersebut. Berikut ini akan diuraikan tipe-tipe jaringan berdasarkan bagiannya masing-masing.

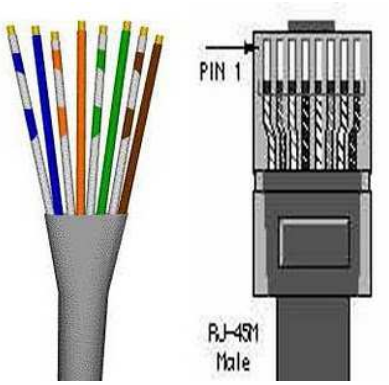
2.1.1 Komponen Jaringan Komputer

Sebuah sistem jaringan komputer akan terbentuk dari beberapa komponen yang menunjang dalam jaringan itu sendiri, oleh karena itu akan dipaparkan tentang komponen-komponen yang dibutuhkan untuk membangun sebuah jaringan komputer.

2.1.1.1 Kabel

Setiap kabel mempunyai kemampuan dan spesifikasi yang berbeda. Jenis kabel yang menjadi standar dalam penggunaan untuk komunikasi data dalam jaringan komputer adalah kabel *Twisted Pair*[3].

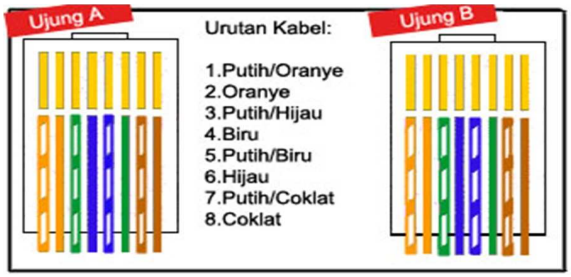
Ethernet juga dapat menggunakan jenis kabel UTP (*Unshielded Twisted Pair*). Kabel UTP yang umum dipakai adalah kabel yang terdiri dari 4 pasang kabel terpilin.



Gambar 2.1 *Kabel UTP dan Konektor RJ-45.*

Terdapat tipe penyambung kabel jenis UTP, yaitu *straight trough cable*, *crossover cable* ditambah satu jenis pemasangan khusus untuk *cisco* router, yaitu *roll over cable*. Perbedaannya, *straight cable* dipakai untuk menghubungkan untuk beberapa unit komputer melalui perantara konsentrator (*hub/switch*) maupun *repeater*, sedangkan *crossover cable* digunakan untuk media komunikasi antar komputer (tanpa *hub/switch*) atau dalam kasus tertentu berguna untuk menghubungkan *hub* ke *hub*. Adapun cara pemasangan kabel UTP model *straight trough* :

- Straight trough T568A



Gambar 2.2 *Pemasangan Kabel straight trough T568A.*

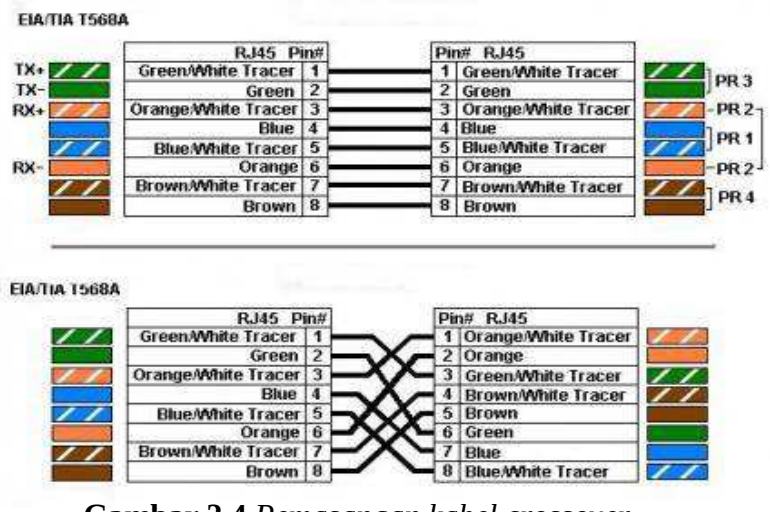
- Straight trough T568B



Gambar 2.3 *Pemasangan Kabel straight trough T568B.*

Pemasangan kabel model *cross over* merupakan penggabungan dari model *straight trough* T568A dengan Ts68B.

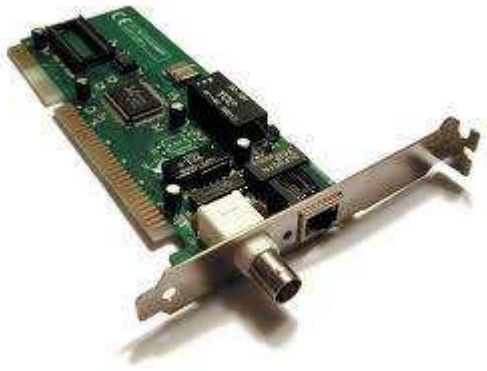
- Crossover



Gambar 2.4 *Pemasangan kabel crossover.*

2.1.1.2 Kartu Ethernet

Ethernet card atau *landcard* berfungsi sebagai media penghubung antara komputer dengan jaringan. Ada beberapa jenis *port* koneksi yang dapat digunakan. Jika didesain untuk jenis kabel *coaxial* maka konektor yang dipakai adalah konektor BNC (*Bareel Nut Connector*). Sementara jika di desain untuk kabel *twisted pair* maka konektor yang di pakai adalah konektor RJ-45[3].



Gambar 2.5 *Ethernet card.*

2.1.1.3 Hub dan Switch

Hub atau *swicth* adalah perangkat untuk menyatukan kabel-kabel jaringan dari tiap *workstation, server*, atau perangkat lainnya. *Hub* biasa dipakai pada topologi *star*. *Hub* dan *switch* umumnya mempunyai *port* RJ-45 sebagai *port* tempat menghubungkan komputer[3].



Gambar 2.6 *Hub dan switch.*

Perbedaannya, *switch* merupakan konsentrator yang memiliki kemampuan *manajemen traffic* data lebih baik dibandingkan *hub*.

2.1.1.4 Router

Router merupakan perangkat yang dikhususkan untuk menangani koneksi antara dua atau lebih jaringan yang terhubung melalui *packet switchi*ng. *Router* bekerja dengan melihat alamat asal dan alamat tujuan dari paket yang melewatinya dan memutuskan *rute* yang akan dilewati paket tersebut untuk sampai ke tujuan. *Router* mengetahui alamat masing-masing komputer di lingkungan jaringan lokal, mengetahui alamat *bridge*, dan *router* lainnya[3].



Gambar 2.7 *Router*.

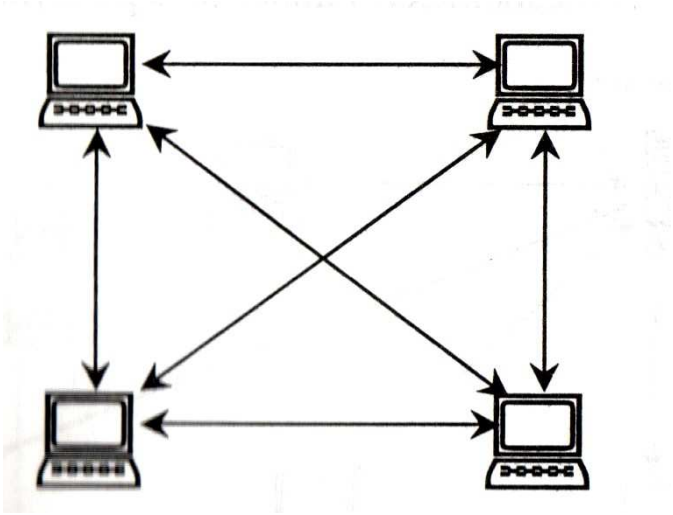
2.1.2 Konsep Hubungan Jaringan Komputer

Berdasarkan konsep hubungan terdiri dari jaringan *Peer to Peer* dan jaringan *Client Server*. Berikut uraian dari kedua jaringan tersebut.

2.1.2.1 Hubungan Peer to Peer

Pada jaringan *peer to peer* setiap komputer yang terhubung dalam jaringan dapat berkominukasi dengan komputer-komputer lain secara langsung tanpa melalui komputer perantara. Pada jaringan tipe ini sumber daya komputer terbagi pada seluruh komputer yang terhubung dalam jaringan tersebut, baik sumber daya yang berupa perangkat keras maupun perangkat lunak dan datanya[4].

Sebuah komputer yang terhubung dalam sebuah jaringan *peer to peer* pada prinsipnya mampu untuk bekerja sendiri sebagai sebuah komputer *standalone* (berdiri sendiri). Membangun sebuah jaringan seperti ini pengguna bisa menggunakan komputer-komputer yang memiliki kemampuan yang setara karena keamanan dalam jaringan tersebut diatur dan di kontrol oleh masing-masing komputer dalam jaringan tersebut.



Gambar 2.8 *Model hubungan peer to peer.*

Dari gambar 2.8 tampak bahwa masing-masing komputer dalam sebuah jaringan *peer to peer* terhubung secara langsung ke seluruh komputer yang terdapat dalam jaringan tersebut.

Keunggulan jaringan *peer to peer* yaitu :

- Antar komputer dalam jaringan dapat saling berbagi fasilitas yang dimilikinya seperti *harddisk* dan *printer*.
- Biaya operasional relatif lebih murah dibandingkan dengan tipe *client-server*, salah satunya adalah tidak memerlukan adanya *server* yang memiliki kemampuan khusus untuk mengorganisasikan dan menyediakan fasilitas jaringan.
- Kelangsungan kerja jaringan tidak tergantung pada satu *server*, sehingga bila salah satu komputer mati atau rusak, jaringan secara keseluruhan tidak akan mengalami gangguan.

Kelemahan jaringan *peer to peer* yaitu :

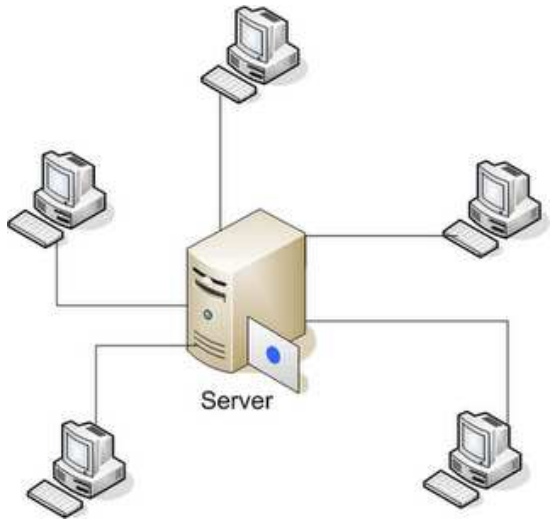
- *Troubleshooting* jaringan relatif lebih sulit, karena pada jaringan *peer to peer* setiap komputer dimungkinkan untuk terlibat dalam komunikasi yang ada. Di jaringan *client-server*, komunikasi adalah antara *server* dengan *workstation.*

- Kinerja lebih rendah dibandingkan dengan jaringan *client-server*. Karena setiap komputer disamping harus mengelola pemakaian fasilitas jaringan juga harus mengelola pekerjaan.
- Sistem keamanan jaringan ditentukan oleh masing-masing *user* dengan mengatur keamanan masing-masing fasilitas yang dimiliki. Karena data jaringan tersebut di masing-masing komputer dalam jaringan, maka *backup* harus dilakukan oleh masing-masing komputer tersebut.

2.1.2.2 Hubungan Client Server

Pada jaringan *client-server* terdapat sebuah komputer yang berfungsi sebagai *server* sedangkan komputer-komputer yang lain berfungsi sebagai *client.* Sesuai namanya maka komputer *server* berfungsi dan bertugas untuk melayani seluruh komputer yang terdapat dalam jaringan tersebut[4].

Pada sebuah kompuetr dimungkinkan untuk digunakannya lebih dari komputer *server*, bahkan dengan kemampuan dan fasilitas yang berbeda. Sedangkan komputer-komputer *client* sesuai dengan namanya menerima pelayanan dari komputer *server*,. Komputer-komputer ini disebut juga dengan *workstation*, yaitu komputer dimana pengguna jaringan dapat mengakses dam memanfaatkan pelayanan yang diberikan oleh komputer *server*.



Gambar 2.9 *Model hubungan client-server*

Pada gambar 2.9 dapat dilihat bahwa komputer-komputer dalam jaringan (*client*) dapat saling berkomunikasi melalui perantara *server.* Jika komputer *server* tidak aktif, maka komputer-komputer *client* tidak akan dapat saling berkomunikasi.

Keunggulan jaringan *client-server* yaitu :

- Kecepatan akses lebih tinggi karena penyedian fasilitas jaringan dan pengelolaannya dilakukan secara khusus oleh satu komputer (*server*) yang tidak dibebani dengan tugas lain sebagai *workstation.*
- Sistem keamanan lebih baik, karena adanya sistem keamanan jaringan.
- Administrasi jaringan lebih baik, karena terdapat seorang pemakai yang bertugas sebagai administrasi jaringan.
- Sistem *backup* data lebih baik, karena pada jaringan *client-server backup* dilakukan terpusat di *server*, sehingga kemungkinan adanya data kembar sangatlah kecil.

Kelemahan jaringan *client-server* yaitu :

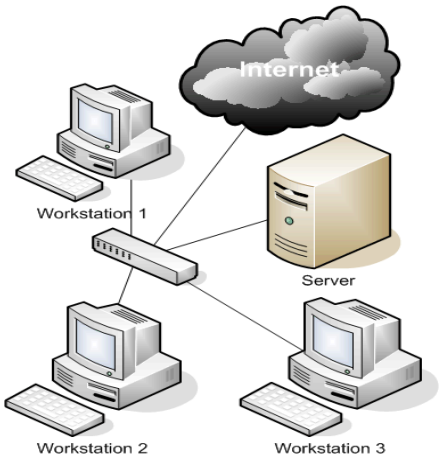
- Biaya operasional lebih mahal, karena harus ada orang yang ahli sebagai administrator jaringan.
- Diperlukan adanya satu komputer khusus yang berkemampuan lebih untuk ditugaskan sebagai *server.*
- Kelangsungan jaringan sangat tergantung pada *server*, bila *server* mengalami gangguan maka secara keseluruhan jaringan akan terganggu.

2.1.3 Jenis Jaringan Berdasarkan Cakupan Geografis

Jaringan komputer dapat dikelompokkan berdasarkan luas area yang dapat di jangkau atau dilayani. Secara umum jaringan komputer terbagi menjadi 3 jenis, yaitu *Local Area Network* (LAN), *Metropolitan Area Network* (MAN), dan *Wide Area Network* (WAN). Namun yang lebih banyak digunakan adalah jenis LAN.

2.1.3.1 Local Area Network (LAN)

Local Area Network (LAN) adalah jaringan yang dibatasi oleh area yang relatif kecil, umumnya dibatasi oleh area lingkungan seperti sebuah kantor pada sebuah gedung. LAN sering kali menggunakan teknologi transmisi kabel tunggal. LAN tradisional beroperasi pada kecepatan mulai 10 sampai 100 Mbps (Mega bit/detik) dengan *delay* rendah (puluhan mikro *second*) dan mempunyai faktor kesalahan yang kecil. LAN-LAN modern dapat beroperasi pada kecepatan yang lebih tinggi, sampai ratusan megabit/detik[3].



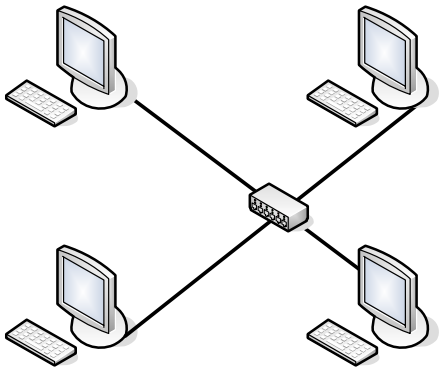
Gambar 2.10 *Local Area Network.*

2.1.4 Topologi Jaringan Komputer

Topologi atau arsitektur jaringan merupakan pola hubungan antar terminal dalam suatu sistem jaringan komputer. Topologi ini akan mempengaruhi tingkat efektifitas kinerja jaringan. Ada beberapa jenis topologi yang dapat diimplementasikan dalam jaringan. Namun, bentuk topologi yang utama adalah topologi *Star*.

2.1.4.1 Topologi Star

Pada topologi *star*, terdapat sebuah terminal pusat (*hub/switch*) yang mengatur dan mengendalikan semua kegiatan komunikasi data. *Traffic* data mengalir dari *node* ke terminal pusat dan diteruskan ke *node* (*station*) tujuan. Berikut adalah skema dari topologi *star*[4].



Gambar 2.11 *Topologi star.*

Keuntungan dari topologi *star*:

- Akses ke *station* lain (*client* atau *server*) lebih cepat.
- Dapat menerima *workstation* baru selama *port* di *central node* (*hub/switch*) tersedia.
- *Hub/switch* bertindak sebagai konsentrator.
- *Hub/switch* dapat disusun seri (bertingkat) untuk menambah jumlah *station* yang terkoneksi di jaringan.
- Mendukung pengguna yang banyak dibanding topologi *bus*, maupun *ring*.

Kerugian dari topologi *ring* :

- Bila *traffic* data cukup tinggi dan terjadi *collision*, semua komunikasi akan ditunda, dan koneksi akan dilanjutkan atau dipersilahkan dengan cara acak ketika *hub/switch* mendeteksi tidak ada jalur yang sedang di gunakan oleh *node* lain.

2.2 Protokol Jaringan Komputer

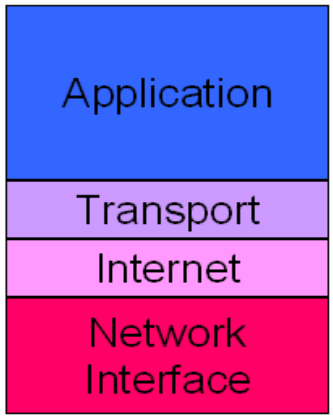
Komunikasi data merupakan proses mengirimkan data dari satu komputer ke komputer yang lain. Untuk dapat mengirim data, pada komputer harus ditambahkan alat khusus yang di kenal dengan *network interface*. Dalam proses pengiriman data ini terdapat beberapa masalah yang harus dipecahkan. Pertama, data harus dapat dikirim ke komputer yang tepat, sesuai dengan tujuannya. Namun hal ini akan menjadi rumit jika komputer tujuan tidak berada dalam

jaringan lokal. Untuk memecahkan masalah *transfer* di atas maka diciptakan sebuah solusi khusus berupa aturan-aturan yang dapat menangani semua masalah komunikasi data, keseluruhan aturan ini harus bekerja sama satu dengan yang lainnya. Sekumpulan aturan untuk mengatur proses pengiriman data ini disebut sebagai protokol komunikasi data[4].

2.2.1 Arsitektur Protokol TCP/IP

TCP/IP adalah sekumpulan protokol yang didesain untuk melakukan fungsi-fungsi komunikasi data pada *wide area network* (WAN). TCP/IP terdiri atas sekumpulan protokol yang masing-masing bertanggung jawab atas bagian-bagian tertentu dari komunikasi data. Berkat protokol ini, tugas masing-masing protokol menjadi jelas dan sederhana dan menjadi protokol komunikasi yang fleksibel[4].

Sekumpulan protokol TCP/IP ini dimodelkan dengan empat *layer* TCP/IP, sebagaimana terlihat pada gambar dibawah ini.



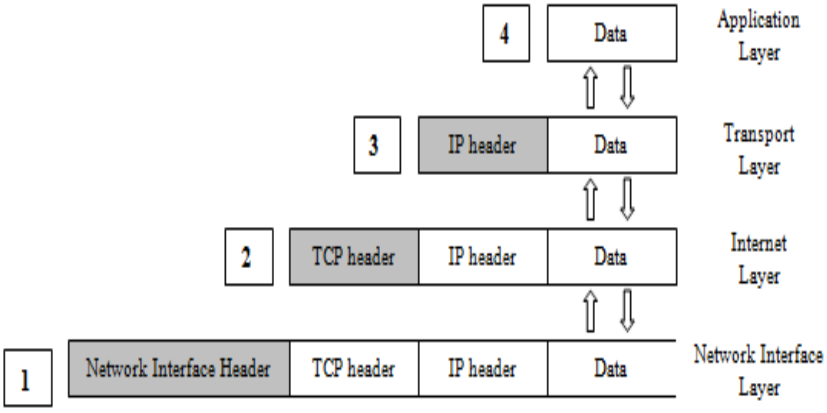
Gambar 2.12 *Layer TCP/IP.*

TCP/IP terdiri atas empat lapis kumpulan protokol yang bertingkat. Keempat lapisan/*layer* tersebut adalah :

1. *Network Interface Layer.*
2. *Internet Layer.*
3. *Transport Layer.*
4. *Application Layer.*

Dalam TCP/IP, terjadi penyimpanan data dari protokol yang berada di satu *layer* ke protokol yang berada di *layer* yang lain. Setiap protokol memperlakukan semua informasi yang diterimanya dari protokol lain sebagai data. Jika suatu protokol menerima data dari protokl lain di *layer* atasnya. Ia akan menambahkan informasi tambahan miliknya ke data tersebut. Informasi ini memiliki fungsi protokol yang sesuai dengan fungsi protokol tersebut. Setelah itu data ini diteruskan lagi ke protokol pada *layer* dibawahnya.

Hal yang sebaliknya terjadi jika suatu protokol menerima data dari protokol lain yang berada pada layer dibawahnya. Jika data ini dianggap *valid*, protokol akan melepas informasi tambahan tersebut, untuk kemudian meneruskan data itu ke protokol lain yang berada pada *layer* diatasnya. Seperti yang terlihat pada gambar di bawah ini.



Gambar 2.13. *Pergerakan data dalam layer TCP/IP*.

2.2.2 IP Address

IP adalah protokol yang sejak awal memang sudah dirancang untuk dipergunakan pada sistem *internetworking*. Alamat IP digunakan untuk mengidentifikasi *interface* jaringan pada *host* komputer. IP memiliki sifat yang dikenal dengan *unreliable*, *connectionless*, *datagram delivery service*. IP mengatur dan mencari cara terbaik untuk menyampaikan suatu paket data ke tujuannya dengan memakai pendekatan datagram, tidak peduli tujuannya berada di jaringan yang sama atau tidak[3].

Datagram adalah skema pensaklaran paket (*packet switching*) dimana tidak terdapat *rute* yang pasti antara sumber pengirim paket dan tujuannya. Paket

yang dikirim dengan IP akan mencari *rute* terbaiknya sendiri untuk mencapai tujuannya. Sebelum dikirim data yang secara ukuran cukup besar akan dipecah-pecah menjadi ukuran-ukuran yang lebih kecil yang disebut dengan paket data. Paket data diberi penomoran khusus dan dikirim untuk mencari *rute*-nya sendiri ke tujuan. Penomoran sangat berguna karena pada *datagram* sangat mungkin paket data yang dikirim terlebih dahulu akan tiba lebih lambat dibanding data yang dikirim belakangan. Penerima akan menyusun kembali paket data sesuai penomerannya apabila keseluruhan paket data lebih diterima.

IP *Address* terdiri dari bilangan biner sepanjang 32 bit yang dibagi atas 4 segmen. Tiap segmen terdiri atas 8 bit yang berarti memiliki nilai desimal dari 0 - 255. Ada sebanyak 232 kombinasi *address* yang bisa dipakai diseluruh dunia (walaupun pada kenyataannya ada sejumlah IP *Address* yang digunakan untuk keperluan khusus). Jadi, jaringan TCP/IP dengan 32 bit *address* ini mampu menampung sebanyak 232 atau lebih dari 4 milyar *host*. Untuk memudahkan pembacaan dan penulisan, IP *Address* biasanya direpresentasikan dalam bilangan desimal. Jadi, range *address* di atas dapat diubah menjadi *address* 0.0.0.0 sampai *address* 255.255.255.255. Nilai desimal dari IP *Address* inilah yang dikenal dalam pemakaian sehari-hari. Contoh IP *Address* adalah : 167.205.9.35.

Ilustrasi IP Address dalam bilangan desimal dan bilangan biner dapat di lihat pada gambar 2.12 berikut :

Desimal	167	205	9	35
Biner	10100111	11001101	00001001	00100011

Gambar 2.14 *Format IP Address dalam biner dan desimal.*

Pada alamat IPv4 dikelompokan dalam 5 kelas yaitu kelas A, kelas B, kelas C, kelas D dan kelas E. Kelas D digunakan untuk *multicast address* dan kelas E digunakan untuk keperluan masa depan. Perbedaan pada setiap kelas tersebut adalah ukuran dan jumlahnya. Pembagian kelas-kelas IP address ini

didasarkan pada dua hal yakni *network* ID dan host ID. Pengelompokan kelas dapat dilihat pada table 2.1 berikut :

Tabel 2.1 *Pengelompokan Kelas menggunakan angka desimal.*

Bit inisial a	Range a	Jumlah Kelas	Kelas	Bagian Network	Bagian Host	Penggunaan
0. .	1 – 126	126	A	A	b, c, d	untuk jaringan besar
10. .	128 – 191	16,384	B	a, b	c, d	untuk jaringan menegah
110. .	192 – 223	2,097,152	C	a, b, c	d	untuk jaringan kecil
1110. .	224 – 247	-	D	a, b, c	d	cadangan : IP Multicasting
1111. .	248 - 255	-	E	a, b, c	d	Cadangan Eksperimen

Private IP Address IANA (*International Assigned Number Authority*) mengelompokkan alamat IP *address* yang dinyatakan "*private*" artinya hanya untuk digunakan dikalangan sendiri atau tidak berlaku di internet.

Kelas A : 10.0.0.0 sampai dengan 10.255.255.255
Kelas B : 172.16.0.0 sampai dengan 172.31.255.255
Kelas C : 192.168.0.0 sampai dengan 192.168.255.255

Gambar 2.15 *IP Private.*

Dengan demikian untuk menetukan kelas A, B, C. cukup dilihat dari delapan bit pertama. Untuk memisahkan *network* ID dan host ID diperlukan sebuah *netmask* dengan definisi sebagai berikut. Untuk bagian menjadi dari *network* ID maka mask yang digunakan adalah binary 1, sedangkan untuk *host* ID digunakan binary 0.

A : 11111111 00000000 00000000 00000000 255.0.0.0
B : 11111111 11111111 00000000 00000000 255.255.0.0
C : 11111111 11111111 11111111 00000000 255.255.255.0

Gambar 2.16 *Netmask natural.*

2.3 Server

Server adalah sebuah sistem komputer yang menyediakan jenis layanan tertentu dalam sebuah jaringan komputer. *Server* didukung dengan prosesor yang bersifat *scalable* dan RAM yang besar, juga dilengkapi dengan sistem operasi khusus, yang disebut sebagai sistem operasi jaringan atau *network operating system*. *Server* juga menjalankan perangkat lunak *administratif* yang mengontrol akses terhadap jaringan dan sumber daya yang terdapat di dalamnya, seperti halnya berkas atau alat pencetak (*printer*), dan memberikan akses kepada *workstation* anggota jaringan [4].

Umumnya, di atas sistem operasi *server* terdapat aplikasi-aplikasi yang menggunakan arsitektur *client/server*. Contoh dari aplikasi ini adalah DHCP *Server, Mail Server,* HTTP *Server,* FTP *Server, D*NS *Server* dan lain sebagainya. Setiap system operasi *server* umumnya membundel layanan-layanan tersebut atau layanan tersebut juga dapat diperoleh dari pihak ketiga. Setiap layanan tersebut akan merespons terhadap *request* dari klien. Sebagai contoh, klien DHCP akan memberikan *request* kepada *server* yang menjalankan *server* DHCP, ketika sebuah klien membutuhkan alamat IP, klien akan memberikan perintah/*request* kepada *server,* dengan bahasa yang dipahami oleh *server* DHCP, yakni protokol DHCP itu sendiri.

Contoh sistem operasi *server* adalah Windows NT 3.51, dan dilanjutkan dengan Windows NT 4.0. Saat ini sistem yang cukup populer adalah Windows 2000 *Server* dan Windows Server 2003, kemudian Sun Solaris, Unix, dan GNU/linux.

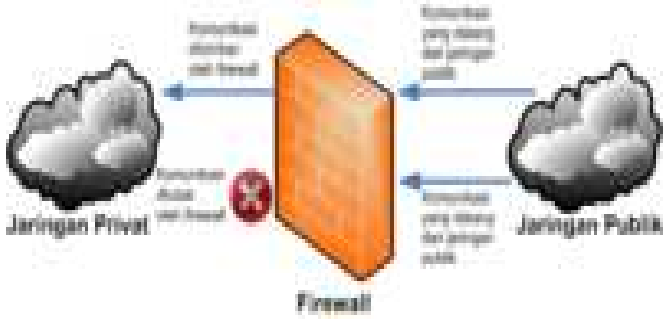
Server biasanya terhubung dengan *client* dengan kabel UTP dan sebuah *Network Card*. Kartu jaringan ini biasanya berupa kartu PCI atau ISA. Fungsi *server* sangat banyak, misalnya untuk situs internet, ilmu pengetahuan, atau sekedar penyimpanan data. Namun yang paling umum adalah untuk mengkoneksikan komputer *client* ke Internet.

2.3.1 DHCP Server

DHCP *(Dynamic Host Configuration Protocol)* adalah protokol yang berbasis arsitektur *client/server* yang dipakai untuk memudahkan pengalokasian alamat IP dalam satu jaringan. Sebuah jaringan lokal yang tidak menggunakan DHCP harus memberikan alamat IP kepada semua komputer secara manual. Jika DHCP dipasang di jaringan lokal, maka semua komputer yang tersambung di jaringan akan mendapatkan alamat IP secara otomatis dari server DHCP. Selain alamat IP, banyak parameter jaringan yang dapat diberikan oleh DHCP, seperti *default gateway* dan DNS *server*. DHCP didefinisikan dalam RFC 2131 dan RFC 2132 yang dipublikasikan oleh *Internet Engineering Task Force*. DHCP merupakan ekstensi dari protokol *Bootstrap Protocol (BOOTP)* [4].

2.4 Firewall

Firewall adalah suatu aturan yang diterapkan baik terhadap *hardware, software* maupun sistem itu sendiri dengan tujuan untuk melindungi, baik dengan melakukan filterisasi, membatasi, ataupun menolak suatu koneksi pada jaringan yang dilindunginya dengan jaringan luar lainnya seperti internet. Oleh karena seringnya *firewall* digunakan untuk melindungi jaringannya, *firewall* tersebut juga berfungsi sebagai pintu keluar jaringan yang dilindunginya dengan jaringan lainnya atau biasa disebut dengan *gateway*[1].



Gambar 2.17 *Analogi Firewall.*

2.4.1 Tipe Firewall

Firewall dapat dibedakan berdasarkan mekanisme atau cara *firewall* tersebut bekerja. Salah satunya adalah *packet filtering gateway.*

Packet Filtering Gateway dapat diartikan sebagai *firewall* yang bertugas melakukan filtrasi terhadap paket – paket yang datang dari luar jaringan yang dilindunginya. Filtrasi paket ini hanya sebatas berdasarkan sumber paket, tujuan paket, dan atribut – atribut dari paket tersebut, misalnya paket tersebut bertujuan ke *server* yang telah dibuat dengan menggunakan IP 192.168.0.1 dengan port 80. Port 80 adalah atribut yang dimiliki oleh paket tersebut[1].

2.4.2 Fungsi Firewall

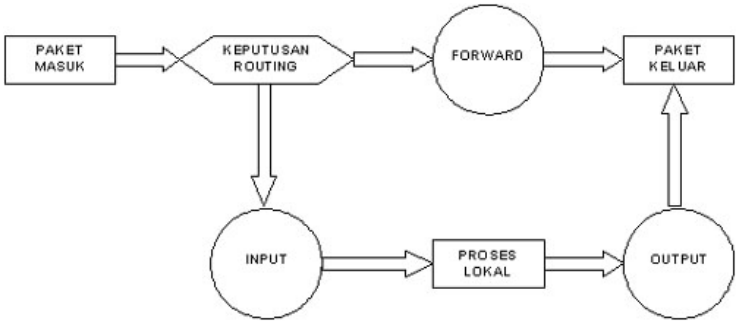
Secara Fundamental Fungsi *firewall* dalam sebuah jaringan adalah berfungsi dapat mengatur dan mengontrol lalu lintas jaringan yang diizinkan untuk mengakses jaringan privat atau komputer yang dilindungi oleh *firewall*. *Firewall* melakukan hal yang demikian, dengan melakukan inspeksi terhadap paket-paket dan memantau koneksi yang sedang dibuat, lalu melakukan penapisan (*filtering*) terhadap koneksi berdasarkan hasil inspeksi paket dan koneksi tersebut[1].

2.4.3 Aplikasi Firewall Di GNU/linux (*IPTables*)

Iptables merupakan salah satu aplikasi *firewall* yang berada di lingkungan sistem operasi GNU/linux. *Iptables* digunakan untuk administrasi *filtering* paket dan *Network Address Translation* (NAT) pada IPv4, gambaran umum *iptables* digunakan untuk konfigurasi, merawat dan memeriksa *rules tables* (tabel aturan) tentang *filter* paket IP yang terdapat di kernel GNU/linux. *Iptables* memiliki *chains*, setiap *chains* memiliki daftar (*list*) aturan untuk mencocokkan suatu paket yang datang. Setiap aturan tersebut berfungsi memberikan keputusan eksekusi apa yang akan dilakukan bila paket yang datang cocok dengan aturan yang telah dibuat[6].

IPTables memiliki tiga macam daftar aturan bawaan dalam tabel penyaringan, daftar tersebut dinamakan rantai *firewall* (*firewall chain*) atau sering

disebut *chain* saja. Ketiga *chain* tersebut adalah *INPUT, OUTPUT* dan *FORWARD*. (manual Iptables terdapat pada lampiran A).



Gambar 2.18 *Analogi IPTables.*

Pada diagram tersebut, lingkaran menggambarkan ketiga rantai atau *chain*. Pada saat sebuah paket sampai pada sebuah lingkaran, maka disitulah terjadi proses penyaringan. Rantai akan memutuskan nasib paket tersebut. Apabila keputusannnya adalah *DROP*, maka paket tersebut akan di-*drop*. Tetapi jika rantai memutuskan untuk *ACCEPT*, maka paket akan dilewatkan melalui diagram tersebut.

Sebuah rantai adalah aturan-aturan yang telah ditentukan. Setiap aturan menyatakan "jika paket memiliki informasi awal (*header)* seperti ini, maka inilah yang harus dilakukan terhadap paket". Jika aturan tersebut tidak sesuai dengan paket, maka aturan berikutnya akan memproses paket tersebut. Apabila sampai aturan terakhir yang ada, paket tersebut belum memenuhi salah satu aturan, maka kernel akan melihat kebijakan bawaan (*default*) untuk memutuskan apa yang harus dilakukan kepada paket tersebut. Ada dua kebijakan bawaan yaitu *default DROP* dan *default ACCEPT*.

Aplikasi *firewall* yang cocok di GNU/linux ini secara *default* telah terinstall di hampir semua distribusi GNU/linux dengan menggunakan kernel 2.6.x (bila di bawah versi kernel tersebut belum secara *default* terinstal), seperti, Ubuntu, RedHat, openSUSE, Fedora Core, Slackware, Debian, dan lain sebagainya. Pada saat kita menginstalasi Ubuntu, *iptables* memang sudah terinstall, tapi *default*-nya mengijinkan semua *traffic* untuk lewat. Namun untuk

Gentoo kita harus menginstall *iptables* secara manual dan mengkonfigurasi kernel sesuai dengan kebutuhan *router*.

2.4.4 Aplikasi Firewall Di FreeBSD (PF)

PF (*Packet Filter*) adalah salah satu *firewall* yang berjalan di sistem BSD, PF dapat memeriksa paket-paket jaringan melalui beberapa protokol, *port* dan level. PF menentukan apakah paket tersebut bisa dilewatkan atau ditahan. *Firewall* PF lebih memfokuskan *filtering* terhadap paket-paket jaringan, protokol, koneksi dan *port*[7].

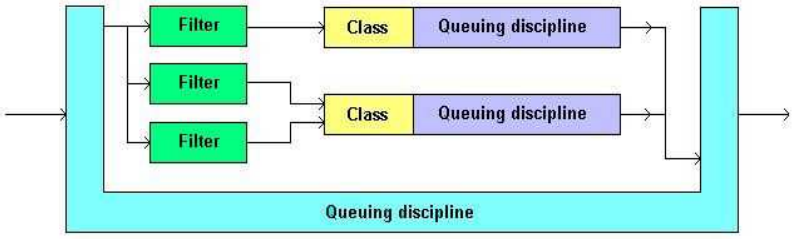
PF memeriksa paket-paket yang di kirim atau paket yang diterima dari mana pun melalui protokol, koneksi, dan *port*. Itu menunjukan bahwa PF dapat melakukan aksi yang akan dilakukan apakah paket yang melawati PF dapat di diteruskan atau dihentikan. Dengan demikian PF dapat membantu mengatur aliran paket dalam sebuah lalu lintas jaringan.

Aplikasi *Firewall* PF ini dapat berjalan dengan baik bila konfigurasi pada kernel sistem BSD, *firewall* PF akan aktif dan dapat digunakan sesuai dengan fungsinya. (manual pf terdapat pada lampiran A).

2.5 Disiplin Antrian Traffic Bandwidth

Pada dasarnya disiplin antrian ini, mengacu pada QoS (*Quality of Services*) dalam pembagian *bandwidth* sehingga *bandwidth* yang tersedia dapat digunakan dengan sebagai mana mestinya dan sesuai dengan kebutuhan yang diperlukan.

Antrian dalam setiap kartu *ethernet* disebut *qdisc* (queuing discipline) yang dipergunakan untuk menyimpan antrian paket data, paket data masuk ataupun keluar melalui *qdisc*. Paket data yang memasuki *qdisc* akan dipisahkan oleh bagian *filter* untuk menentukan port / alamat ip yang akan di atur aliran trafiknya. *qdis*c yang berwarna ungu (*lihat gambar 2.19*) dipergunakan untuk mengeluarkan paket data ke kartu ethernet[6].



Gambar 2.19 *Analogi Disiplin Antrian.*

Pengalokasian *bandwidth* salah satu hal yang paling penting dalam disiplin antrian. Alokasi *bandwidth* yang rata dapat dicapai dengan menggunakan disiplin antrian yang tepat. Tujuan penting lainnya adalah untuk mengontrol waktu tunda dan *jitter* (variasi waktu acak) dimana merupakan hal yang penting untuk membangun aplikasi-aplikasi yang *real-time*. Hal yang mungkin untuk membatasi waktu tunda dan *jitter* dari suatu aliran dengan mengembalikan *resource* jaringan yang dibutuhkan. Izin kontrol diperlukan untuk memutuskan apakah *resource* yang minta dapat di alokasikan. Juga diperlukan untuk mengatur laju aliran balik dalam arti pembentukan. Laju kedatangan haruslah lebih rendah dari laju kepulangan (balik) untuk menghindari waktu tunda yang disebabkan oleh aliran *trafik* sendiri.

Tujuan lainnya pada disiplin antrian adalah menghindari kemacetan paket (kongesti). TCP mempertimbangkan *packet loss* sebagai tanda kongesti. Suatu *router* dapat memberitahukan TCP kongesti secara lengkap dengan mengbatalkan paket secara sengaja. Berikut akan dijelaskan displin antrian di setiap kernel.

2.5.1 Quality of Service (QoS)

QoS merupakan kependekan dari *Quality of Service*, Qos didefinisikan bahwa QoS adalah suatu pengukuran tentang seberapa baik jaringan dan merupakan suatu usaha untuk mendefinisikan karakteristik dan sifat dari suatu service. QoS biasanya digunakan untuk mengukur sekumpulan atribut performansi yang telah dispesifikasikan dan biasanya diasosiasikan dengan suatu servis. Pada jaringan berbasis IP, IP QoS mengacu pada performansi dari paket-paket IP yang lewat melalui satu atau lebih jaringan[6].

QoS didesain untuk membantu pengguna akhir menjadi lebih produktif dengan memastikan bahwa dia mendapatkan performansi yang handal dari aplikasi-aplikasi berbasis jaringan.

QoS mengacu pada kemampuan jaringan untuk menyediakan layanan yang lebih baik pada lalu lintas jaringan tertentu melalui teknologi yang berbeda-beda. QoS merupakan suatu tantangan yang cukup besar dalam jaringan berbasis IP dan internet secara keseluruhan. Tujuan dari QoS adalah untuk memuaskan kebutuhan-kebutuhan layanan yang berbeda, yang menggunakan infrastruktur yang sama. QoS menawarkan kemampuan untuk mendefinisikan atribut-atribut layanan jaringan yang disediakan, baik secara kualitatif maupun kuantitatif.

Komponen-komponen dari QoS adalah:

- ***Packet loss*** terjadi ketika satu atau lebih paket data untuk melakukan perjalanan di sebuah jaringan komputer gagal mencapai tujuan. Packet loss bisa disebabkan oleh sejumlah faktor, termasuk degradasi sinyal jangka menengah jaringan karena *multi-path fading*, *drop* paket karena kemacetan saluran, paket rusak ditolak di-transit, perangkat keras jaringan yang rusak, *driver* jaringan yang rusak atau rutinitas normal routing . Bila disebabkan oleh masalah jaringan, hilang atau menjatuhkan paket dapat berakibat kinerja yang sangat nyata dan akan mempengaruhi semua aplikasi jaringan 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* yang paling sering dialami oleh lalu lintas jaringan yang lewat adalah *delay* transmisi, yang dapat dirumuskan sebagai berikut:

$$Delay = \frac{packet_size \times 8}{line_speed} \times 1000 \ ms \ldots\ldots\ldots\ldots (2.1)$$

  Untuk aplikasi-aplikasi suara dan video interaktif, kemunculan dari *delay* akan mengakibatkan sistem seperti tak merespon.

- ***Jitter***, merupakan variasi dari *delay end-to-end*. Level-level yang tinggi pada jitter dalam aplikasi-aplikasi berbasis UDP merupakan situasi yang tidak dapat diterima di mana aplikasi-aplikasinya merupakan aplikasi-aplikasi *real-time*. Menurut versi tiphon terdapat empat kategori penurunan performansi jaringan peak jitter berikut adalah kategorinya[9]:

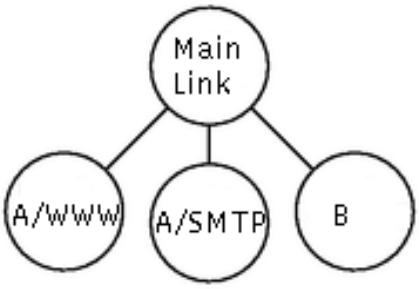
Tabel 2.2 Kategori performansi jaringan IP berdasarkan *jitter*

Kategori Degradasi	*Peak jitter*
Sangat bagus	0 ms
Bagus	75 ms
Sedang	125 ms
Jelek	225 ms

- ***Bandwidth***, merupakan *rate transfer* data maksimal yang dapat diteruskan antara dua titik.

2.5.2 Teknik Disiplin Antrian Di GNU/linux (HTB)

HTB (*Hierarchy Token Bucket*) merupakan salah satu disiplin antrian yang memiliki tujuan untuk menerapkan *link sharing* secara presisi dan adil. Dalam konsep *link sharing*, jika suatu kelas meminta kurang dari jumlah *service* yang telah ditetapkan untuknya, sisa *bandwidth* akan didistribusikan ke kelas-kelas yang lain yang meminta *service*[6].



Gambar 2.20 *Konsep Link Sharing.*

HTB menggunakan TBF sebagai estimator yang sangat mudah diimplementasikan. TBF sangat mudah di set karena banyak dari administrator jaringan yang memiliki ilmu tentangnya. Estimator ini hanya menggunakan

parameter *rate*, sebagai akibatnya seseorang hanya perlu mengeset *rate* yang akan diberikan ke suatu kelas.

Pada HTB terdapat parameter *ceil* sehingga kelas akan selalu mendapatkan *bandwidth* di antara *base rate* dan nilai *ceil rate*nya. Parameter ini dapat dianggap sebagai estimator kedua, sehingga setiap kelas dapat meminjam *bandwidth* selama *bandwidth total* yang diperoleh memiliki nilai di bawah nilai *ceil*. Hal ini mudah diimplementasikan dengan cara tidak mengijinkan proses peminjaman *bandwidth* pada saat kelas telah melampaui *rate* ini (keduanya leaves dan interior dapat memiliki *ceil*). Sebagai catatan, apabila nilai *ceil* sama dengan nilai *base rate*, maka akan memiliki fungsi yang sama seperti parameter *bounded* pada CBQ, di mana kelas-kelas tidak diijinkan untuk meminjam *bandwidth*. Sedangkan jika nilai *ceil* di set tak terbatas atau dengan nilai yang lebih tinggi seperti kecepatan *link* yang dimiliki, maka akan didapat fungsi yang sama seperti kelas *non-bounded.* Sebagai contoh, seseorang dapat menjamin *bandwidth* 1 Mbit untuk suatu kelas, dan mengijinkan penggunaan *bandwidth* sampai dengan 2 Mbit pada kelas tersebut apabila *link* dalam keadaan idle. Parameter *ceil* ini sangatlah berguna untuk ISP karena para ISP kemungkinan besar akan memakainya untuk membatasi jumlah *service* yang akan diterima oleh suatu pelanggan walaupun pelanggan lain tidak melakukan permintaan *service*, dengan kata lain kebanyakan ISP menginginkan pelanggan untuk membayar sejumlah uang lagi untuk memperoleh pelayanan yang lebih baik.

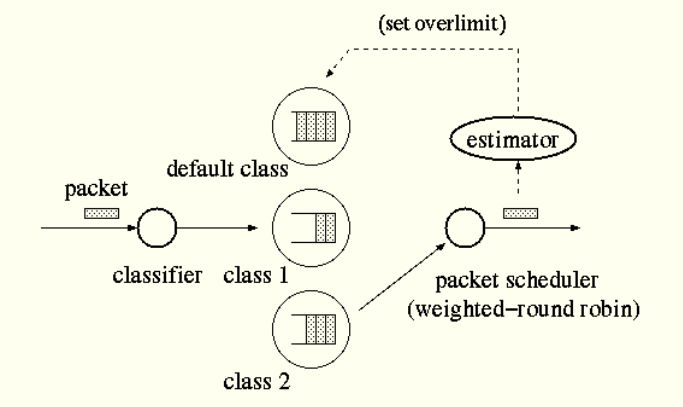
Untuk menjadwalkan pengiriman paket dari antrian, maka HTB menggunakan suatu proses penjadwalan yang dapat dijelaskan sebagai berikut:

- **Class** merupakan parameter yang diasosiasikan dengan *rate* yang dijamin (*assured rate*) AR, *ceil rate* CR, prioritas P, level dan quantum. *Class* dapat memiliki *parent*. Selain AR dan CR, didefinisikan juga *actual rate* atau R, yaitu *rate* dari aliran paket yang meninggalkan *class* dan diukur pada suatu perioda waktu tertentu.
- **Leaf** merupakan *class* yang tidak memiliki anak. Hanya *leaf* yang dapat memegang antrian paket.

- **Level** dari kelas menentukan posisi dalam suatu hirarki. *Leaf-leaf* memiliki level 0, *root class* memiliki level=jumlah level-1 dan setiap *inner class* memiliki level kurang dari satu dari parentnya. Untuk gambar di bawah (jumlah level=3).

2.5.3 Teknik Disiplin Antrian Di FreeBSD (CBQ)

CBQ (*Class Based Queue*) dapat menerapkan pembagian kelas dan men*share link bandwidth* melalui struktur kelas-kelas secara hirarki. Setiap kelas memiliki antriannya masing-masing dan diberikan jatah bandwidthnya. Sebuah kelas *child* dapat meminjam *bandwidth* dari kelas parent selama terdapat kelebihan *bandwidth*. Gambar di bawah ini menunjukkan komponen dasar dari CBQ. CBQ bekerja sebagai berikut: *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 yang telah ditentukannya, maka estimator akan menandai kelas tersebut sebagai kelas yang overlimit. Scheduler 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[7].



Gambar 2.21 *Cara Kerja Penjadwalan CBQ.*

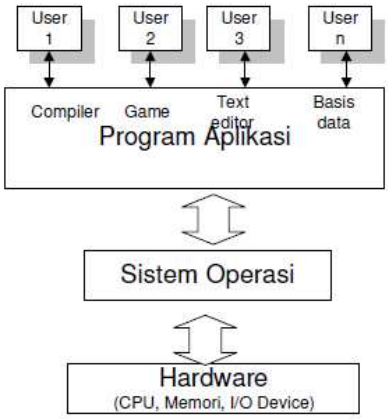
2.6 Sistem Operasi

Sistem operasi merupakan sekumpulan rutin perangkat lunak yang berada di antara program aplikasi dan perangkat keras. Semua perangkat lunak berjalan

di bawah kendali sistem operasi, mengakses perangkat keras lewat sistem operasi dan mengikuti aturan-aturan yang dijalankan oleh sistem operasi[2].

Pengertian sistem operasi secara umum ialah pengelola seluruh sumber daya yang terdapat pada sistem komputer dan menyediakan sekumpulan layanan (*system calls*) ke pemakai sehingga memudahkan dan menyamankan penggunaan serta pemanfaatan sumber daya sistem komputer.

Sistem operasi mengatur pengguna mana yang dapat mengakses suatu sumber daya. Sistem operasi juga sering disebut *resource allocator*. Satu lagi fungsi penting sistem operasi ialah sebagai program pengendali yang bertujuan untuk menghindari kekeliruan (*error*) dan penggunaan komputer yang tidak perlu.



Gambar 2.22 *Struktur sistem operasi*

Sistem Operasi juga bisa disebut sebagai *software* pada lapisan pertama yang ditaruh pada memory komputer pada saat komputer dinyalakan. Sedangkan *software-software* lainnya dijalankan setelah Sistem Operasi berjalan, dan Sistem Operasi akan melakukan layanan inti umum untuk *software-software* itu. Layanan inti umum tersebut seperti akses ke *disk*, manajemen memori, *scheduling task*, dan antar-muka *user*. Sehingga masing-masing *software* tidak perlu lagi melakukan tugas-tugas inti umum tersebut, karena dapat dilayani dan dilakukan oleh Sistem Operasi. Bagian kode yang melakukan tugas-tugas inti dan umum tersebut dinamakan dengan Kernel suatu Sistem Operasi.

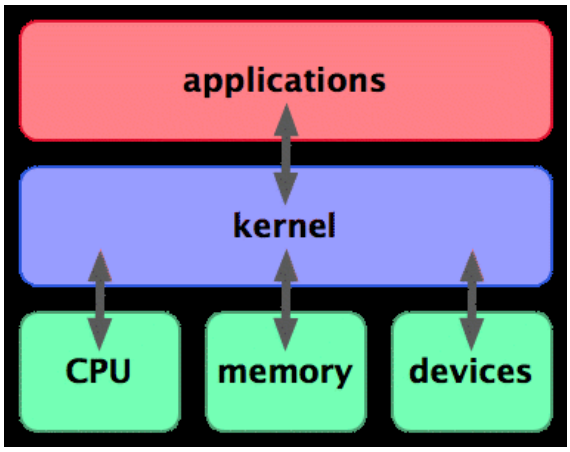
Sistem Operasi secara umum terdiri dari beberapa bagian:

- Mekanisme *boot*, yaitu meletakkan kernel ke dalam memory.

- Kernel, yaitu inti dari sebuah Sistem Operasi.
- *Command Interpreter* atau *shell*, yang bertugas membaca input dari pengguna.
- Pustaka-pustaka, yaitu yang menyediakan kumpulan fungsi dasar dan standar yang dapat dipanggil oleh aplikasi lain.

2.6.1 Kernel

Dalam ilmu komputer, kernel adalah suatu perangkat lunak yang menjadi bagian utama dari sebuah sistem operasi. Tugasnya melayani berbagai macam program aplikasi untuk mengakses perangkat keras komputer secara aman[2].



Gambar 2.23 *Struktur Kernel sistem operasi.*

Karena akses terhadap perangkat keras terbatas, sedangkan ada lebih dari satu program yang harus dilayani dalam waktu bersamaan, maka kernel juga bertugas untuk mengatur kapan dan berapa lama suatu program dapat menggunakan satu bagian perangkat keras tersebut. Hal tersebut dinamakan *multiplexing.* Akses kepada perangkat keras secara langsung merupakan masalah yang kompleks, oleh karena itu kernel biasanya mengimplementasikan sekumpulan abstraksi *hardware*. Abstraksi-abstraksi tersebut merupakan sebuah cara untuk menyembunyikan kompleksitas, dan memungkinkan akses kepada perangkat keras menjadi lebih mudah dan seragam. Sehingga abstraksi pada akhirnya memudahkan pekerjaan *programmer*.

Sebuah kernel sistem operasi tidak harus ada dan dibutuhkan untuk menjalankan sebuah komputer. Program dapat langsung dijalankan secara langsung di dalam sebuah mesin (contohnya CMOS *setup*).

Para arsistek sistem operasi mengembangkan kernel system operasi yang pada akhirnya terbagi menjadi 4 bagian yang secara desaign berbeda, salah satunya adalah *Monolithic Kernel*

2.6.2 Monolitic Kernel

Pendekatan kernel monolitik didefinisikan sebagai sebuah antar muka virtual yang berada pada tingkat tinggi perangkat keras, dengan sekumpulan primitive atau *system call* untuk mengimplementasikan layanan-layanan sistem operasi, seperti halnya manajemen proses, konkurensi, dan manajemen memori pada modul-modul kernel yang berjalan di dalam *mode supervisor*[2].

Meskipun setiap modul memiliki layanan operasi-operasi tersebut terpisah dari modul utama, integrasi kode yang terjadi di dalam monolitik kernel sangatlah kuat, dan karena semua modul berjalam dalam *address space* yang sama, sebuah *bug* dalam salah satu modul dapat merusak keseluruhan sistem. Akan tetapi, ketika implementasi dilakukan secara benar, integrasi komponen *internal* yang sangat kuat tersebut justru mengijinkan fitur-fitur yang dimiliki oleh sistem yang berada dibawahnya dieksploitasi secara efektif, sehingga membuat sistem operasi dengan kernel monolitik sangatlah efisien, meskipun sangat sulit membuatnya.

Pada sistem operasi *modern* yang menggunakan kernel monolitik, seperti halnya GNU/linux, FreeBSD, Open Solaris, dan Microsoft Windows, dapat memuat modul modul yang dapat dieksekusi pada saat kernel tersebut dijalankan sehingga mengijinkan *ekstensi* terhadap kemampuan kernel sesuai kebutuhan, dan tentu saja dapat membantu menjaga agar kode yang berjalan pada ruangan kernel seminim mungkin.

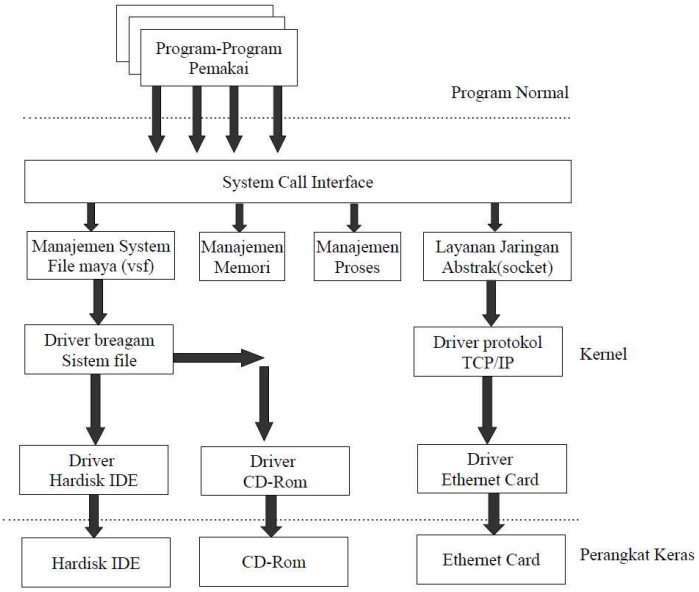
2.6.3 Sistem Operasi GNU/linux

GNU/linux adalah tiruan (*clone*) UNIX. Mulanya, pengembangan linux dilakukan Linus Benedict Torvalds, Universitas Helsinki, Finlandia sebagai proyek hobby. GNU/linux tidak memuat kode UNIX, namun di tulis ulang berdasarkan standar POSIX[2].

Terdapat banyak distribusi (pemaket) GNU/linux untuk mempermudah mendapatkan GNU/linux. Distribusi GNU/linux antara lain Red hat, Slackware, Debian, Gentoo dan masih banyak lagi. Distribusi-distibusi kernel di setujui oleh linus sehingga menjamin kompatibilitas. Karena kode sumber GNU/linux diedarkan maka GNU/linux dapat menjadi sarana untuk belajar sistem operasi.

kernel GNU/linux merupakan inti dari proyek GNU/linux, tetapi komponen lainlah yang membentuk secara komplit sistem operasi GNU/linux. Dimana kernel GNU/linux terdiri dari kode-kode yang dibuat khusus untuk proyek GNU/linux, kebanyakan perangkat lunak pendukungnya tidak eksklusif terhadap GNU/linux, melainkan biasa dipakai dalam beberapa sistem operasi yang mirip UNIX.

Pembagian (*sharing*) alat-alat telah bekerja dalam dua arah. Sistem perpustakaan utama GNU/linux awalnya dimulai oleh proyek GNU, tetapi perkembangan perpustakaannya diperbaiki melalui kerjasama dari komunitas GNU/linux terutama pada pengalamatan, ketidak efisienan, dan *bugs*. Komponen lain seperti GNU C Compiler, gcc, kualitasnya sudah cukup tinggi untuk dipakai langsung dalam GNU/linux. Alat-alat administrasi jaringan dibawah GNU/linux berasal dari kode yang dikembangkan untuk 4.3 BSD, tetapi BSD yang lebih baru, salah satunya FreeBSD, sebaliknya meminjam kode dari GNU/linux, contohnya adalah perpustakaan matematika Intel *floating-point-emulation*. Berikut adalah arsitektur dari system operasi GNU/linux :



Gambar 2.24 *Arsitektur Sistem operasi GNU/Linux*

Sistem GNU/linux terdiri dari tiga bagian kode penting yaitu:

1. Kernel bertanggung jawab memelihara semua abstraksi penting dari sistem operasi, termasuk hal seperti proses- proses dan memori virtual.
2. Perpustakaan sistem menentukan kumpulan fungsi standar dimana aplikasi dapat berinteraksi dengan kernel, dan mengimplementasi hampir semua fungsi sistem operasi yang tidak memerlukan hak penuh atas kernel.
3. *Utilitas* Sistem adalah program yang melakukan pekerjaan manajemen secara individu dan khusus.

2.6.3.1 Gentoo GNU/linux

Gentoo GNU/linux adalah suatu distribusi GNU/linux yang memakai paket sistem manajemen *Portage* Manajemen paket ini dirancang untuk modular (mudah ditambah-tambah), *portabel* (dapat di *port* ke distro lain), mudah ditata, fleksibel, dan dioptimalkan untuk masing-masing komputer pengguna. Paket-paketnya biasanya dibangun dari kode sumber (*source*), walaupun untuk kenyamanan, sebagian besar paket perangkat lunak juga tersedia dalam bentuk binari. Gentoo GNU/linux, distribusi yang mempunyai keunikan tersendiri dan dapat dioptimisasi dan diubah untuk hampir segala jenis aplikasi atau kegunaan.

Konfigurasi yang sangat ekstrim, dukungan dari sesama *user* dan *developer* yang sangat bagus adalah salah satu kelebihan Gentoo[8].

Berkat adanya teknologi bernama *Portage*, Gentoo GNU/linux dapat menjadi *server* yang aman, sistem *develop*, profesional desktop, *game* sistem, solosi buat '*embedded*' atau apa pun sesuai yang kita ingini. Karena kemampuan adaptasi Gentoo yang hampir tidak ada batasnya, maka Gentoo GNU/linux disebut sebagai sistem operasi MetaDistribusi.

2.6.4 Sistem Operasi FreeBSD

FreeBSD adalah sistem operasi berbasis unix dan merupakan turunan asli dari sistem operasi UNIX AT & T yang di distribusikan oleh *Barkeley Sistem Distribution* (BS) yang dikembangkan dari system 4.4BSD-lite. Unix sendiri merupakan sistem operasi yang dikembangkan dengan menggunakan bahasa *assembly*. Pada perkembangan selanjutnya, UNIX kemudian ditullis ulang dalam bahasa C dan di kembangkan lebih lanjut di laboratorium Bell milik AT-T. Keistimewaan FreeBSD yang merupakan turunan langsung dari system UNIX dibandingkan sistem operasi lain adalah[2] :

- Multiuser

  FreeBSD mendukung pengunaan sistem dan berbagai sumber daya yang ada dalam sistem (misalnya *hardisk*) secara bersamaan oleh banyak *user* dalam waktu yang bersamaan.

- Multitasking

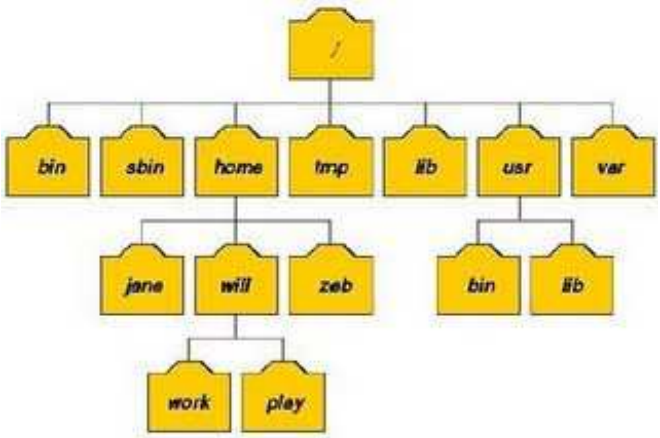
  Dengan sifat yang *multitasking* ini *user* dapat melakukan berbagai pekerjaan sekaligus dalam satu waktu yang bersamaan dengan satu terminal. Pada saat system sedang proses suatu pekerjaan, misalnya proses *compiling*, *user* dapat melakukan pekerjaan lain tanpa harus menunggu proses lain selesai terlebih dahulu

- Portabilitas

  FreeBSD dapat di install dalam arsitektur mesin yang berbeda-beda. Sehingga dapat bekerja pada keluarga mesin intel x86, DEC Alpha, Sun UltraSPARC,IA-64,AMD64,Power PC, dan NEC PC-98. Dengan *portabilitas*nya yang tinggi dapat mengihilangkan ketergantungan *user* pada satu *vendor*, sehingga *user* memiliki kebebasan untuk berganti-ganti jenis mesin sesuai dengan kebutuhan.

- Hirarki sistem file

  Dengan adanya sistem *file* yang terstruktur dengan baik, memudahkan bagi *user* untuk mengatur letak *file*, hal ini memudahkan bagi *user* dalam melakukan pengaturan terhadap sistem



Gambar 2.25 *Struktur Hirarki standar file sistem.*

Dengan minimalnya kebutuhan *hardware* dan kelebihan yang dimiliki FreeBSD, menjadi sistem ini cukup handal jika di jadikan sebuah system *server*.