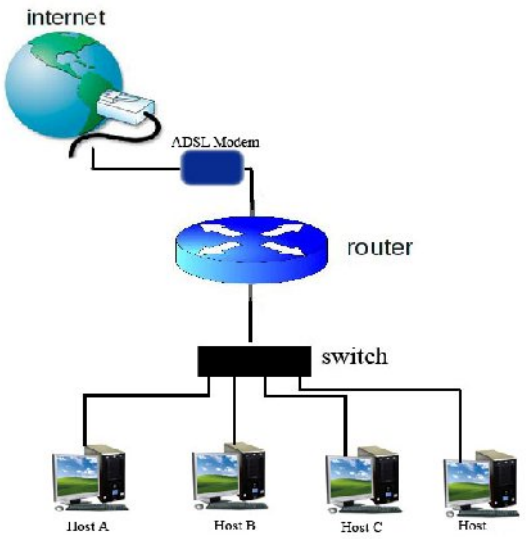
BAB II
LANDASAN TEORI

2.1 *Router* dan PC *(Personal Computer) Router*

Router adalah sebuah alat jaringan komputer yang mengirimkan paket data melalui sebuah jaringan atau internet menuju tujuannya, melalui sebuah proses yang dikenal sebagai *routing*. *Router* digunakan sebagai penghubung antar dua atau lebih jaringan untuk meneruskan paket data dari satu jaringan ke jaringan lainnya. Sedangkan PC *Router* adalah sebuah komputer yang dijadikan sebagai *router*.



Gambar 2.1 Topologi PC Router Sederhana

2.2 Protokol TCP/IP *(Transmission Control Protocol/InternetProtocol)*

Pada dasarnya jika 2 komputer akan melakukan pertukaran data/informasi, memerlukan sebuah protokol yang bertugas untuk mengatur bagaimana komunikasi antar komputer tersebut. Sekelompok komputer yang terhubung satu sama lain dengan *network interface* (antar muka jaringan) yang kemudian disebut *computer network* (jaringan komputer) dapat menggunakan banyak macam protokol, agar 2

buah komputer dapat berkomunikasi maka diperlukan protokol yang sama. Protokol berfungsi mirip bahasa manusia, dimana untuk dapat berbicara dan mengerti satu sama lain diperlukan bahasa yang sama.

TCP/IP merupakan sekelompok protokol yang mengatur komunikasi data komputer dan memungkinkan komputer berbagai jenis dan *vendor* serta berbeda sistem operasi untuk berkomunikasi bersama dengan baik. TCP/IP ini dikembangkan pertama kali oleh lembaga riset Departemen Pertahanan Amerika DARPA *(Defense Advance Research Project Agency)*. [7]

Perkembangan TCP/IP yang cepat dan diterima secara luas tidak hanya dikarenakan rekomendasi DARPA, melainkan fitur-fitur penting yang ada pada TCP/IP diantaranya:

- TCP/IP dikembangkan menggunakan standar protokol yang terbuka. Tersedia secara bebas dan dikembangkan tanpa bergantung pada perangkat keras ataupun sistem operasi tertentu.
- Tidak tergantung pada spesifik perangkat jaringan tertentu. Hal ini memungkinkan TCP/IP untuk mengintegrasikan berbagai macam jaringan.
- TCP/IP menggunakan pengalamatan yang unik dalam skala global. Dengan demikian memungkinkan komputer dapat saling berhubungan walaupun jaringannya seluas internet sekarang ini.
- Standardisasi protokol TCP/IP dilakukan secara konsisten dan tersedia secara luas untuk siapapun tanpa biaya. Hal ini diwujudkan dalam RFC *(Request For Comment)*.

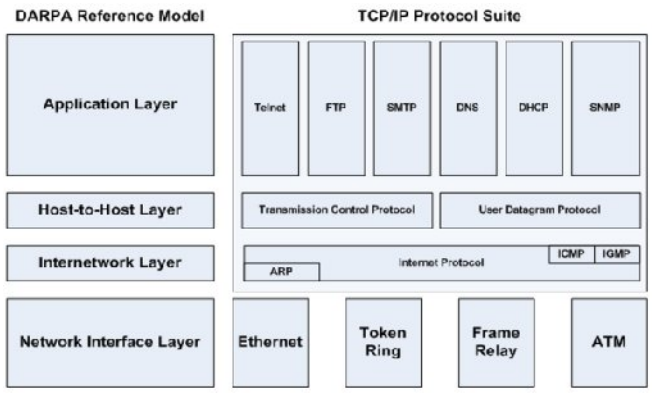
TCP/IP terdiri atas sekumpulan protokol yang masing-masing bertanggung jawab atas bagian-bagian tertentu dalam komunikasi data dan didesain untuk melakukan fungsi-fungsi komunikasi data pada LAN *(Local Area Network)* maupun WAN *(Wide Area Network)*. Dengan prinsip pembagian tersebut, TCP/IP menjadi protokol komunikasi data yang fleksibel dan dapat diterapkan dengan mudah disetiap jenis komputer dan antarmuka jaringan, karena sebagian besar isi kumpulan protokol ini

tidak spesifik terhadap satu komputer atau peralatan jaringan tertentu. Agar TCP/IP dapat berjalan pada antarmuka jaringan tertentu, hanya diperlukan perubahan pada bagian protokol yang berhubungan dengan antar-muka jaringan saja. Sekumpulan protokol TCP/IP ini dimodelkan dalam empat lapisan/*layer* yang bertingkat.

Keempat layer tersebut ialah:

1. *Application Layer,* merupakan *layer* program aplikasi yang menggunakan protokol TCP/IP. Beberapa diantaranya adalah: Telnet, FTP *(File Transfer Protocol)*, SMTP *(Simple Mail Transport Protocol),* SNMP *(Simple Network Management Protocol),* HTTP *(Hypertext Transfer Protocol),* DHCP *(Dynamic Host Configuration Protocol)* dan DNS *(Domian Name System)* .
2. *Transport Layer,* berisi protokol yang bertanggung jawab untuk mengadakan komunikasi antar dua komputer. Pada *layer* ini terdiri atas dua protokol, yaitu: TCP *(Transport Control Protocol)* dan UDP *(User Datagram Protocol)*.
3. *Internet Layer,* berfungsi untuk menangani pergerakan paket data dalam jaringan dari komputer pengirim ke komputer tujuan. Protokol yang berada dalam fungsi ini antara lain: IP *(Internet Protocol),* ICMP *(Internet Control Message Protocol),* dan IGMP *(Internet Group Management Protocol).*
4. *Network Layer,* merupakan *layer* paling bawah yang bertanggung jawab mengirim dan menerima data dari dan ke media fisik.

Model sekumpulan protokol TCP/IP tersebut dapat digambarkan sebagaimana terlihat pada gambar 2.2



Gambar 2.2 Layer pada TCP/IP

2.2.1 DHCP *(Dynamic Host Configurating System)*

DHCP 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.*

2.2.2 DNS *(Domain Name System)*

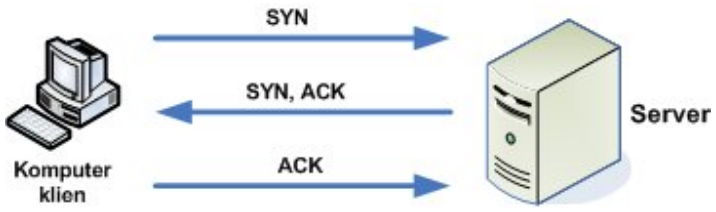
DNS adalah salah satu jenis sistem yang melayani permintaan pemetaan IP *address* ke FQDN *(Fully Qualified Domain Name)* dan dari FQDN ke IP *address*. FQDN lebih mudah diingat oleh manusia daripada IP *address.* Sebagai contoh, sebuah komputer memiliki IP *address* 167.205.22.114 dan memiliki FQDN nic.itb.ac.id. Nama nic.itb.ac.id tentunya lebih mudah diingat daripada nomor IP *address* di atas. Apalagi setelah lahirnya konsep IP *Version* 6 yang memiliki 6 segmen untuk setiap komputer sehingga nomor IP *address* menjadi semakin panjang dan lebih sulit untuk diingat. Selain itu, DNS juga menyediakan layanan *mail routing,* informasi mengenai *hardware*, sistem operasi yang dijalankan, dan aplikasi jaringan yang ditangani oleh *host* tersebut.

2.2.3 TCP *(Transmision Control Protocol)*

TCP merupakan protokol yang berada pada *layer transport* dari *layer* TCP/IP. TCP adalah protokol yang bersifat *byte stream*, *connection-oriented* dan *reliable* dalam pengiriman data. TCP menggunakan komunikasi *byte-stream,* yang berarti bahwa data dinyatakan sebagai suatu urutan-urutan *byte. Connecton-oriented* berarti sebelum terjadi proses pertukaran data antar komputer terlebih dahulu harus dibentuk suatu hubungan. Hal ini dapat doanalogikan dengan proses pendialan nomor telepon dan akhirnya terbentuk hubungan. [7]

Kehandalan TCP dalam mengirimkan data didukung oleh mekanisme yang disebut *Positive Acknowledgement with Re-transmission* (PAR). Data yang dikirim dari *layer* aplikasi akan dipecah-pecah dalam bagian-bagian yang lebih kecil dan diberi nomor urut sebelum dikirim ke *layer* berikutnya. Unit data yang sudah dipecah-pecah tadi disebut *segment.* TCP selalu meminta konfirmasi setiap kali selesai mengirimkan data, apakah data tersebut sampai pada komputer tujuan dan tidak rusak. Jika data berhasil sampai tujuan, TCP akan mengirimkan data urutan berikutnya. Jika tidak berhasil, maka TCP akan melakukan pengiriman ulang urutan data yang hilang atau rusak tersebut. Dalam kenyataannya TCP menggunakan sebuah *acknowledgement* (ACK) sebagai suatu pemberitahuan antara komputer pengirim dan penerima.

Proses pembuatan koneksi TCP disebut juga dengan *Three-way Handshake*. Tujuan metode ini adalah agar dapat melakukan sinkronisasi terhadap nomor urut dan nomor *acknowledgement* yang dikirimkan oleh kedua pihak dan saling bertukar ukuran TCP *Window*. Prosesnya dapat digambarkan sebagai berikut:



Gambar 2.3 Proses Pembuatan koneksi TCP
(Three-way Handshake)

Keterangan dari gambar 2.3 adalah sebagai berikut:

- *Host* pertama (yang ingin membuat koneksi) akan mengirimkan sebuah segmen TCP dengan flag SYN diaktifkan kepada *host* kedua (yang hendak diajak untuk berkomunikasi).
- *Host* kedua akan meresponsnya dengan mengirimkan segmen dengan *acknowledgment* dan juga SYN kepada *host* pertama.

- *Host* pertama selanjutnya akan mulai saling bertukar data dengan *host* kedua.

TCP menggunakan proses jabat tangan yang sama untuk mengakhiri koneksi yang dibuat. Hal ini menjamin dua *host* yang sedang terkoneksi tersebut telah menyelesaikan proses transmisi data dan semua data yang ditransmisikan telah diterima dengan baik. Itulah sebabnya, mengapa TCP disebut dengan koneksi yang *reliable*.

2.2.4 UDP *(User Datagram Protocol)*

UDP merupakan protokol yang juga berada pada *layer transport* selain TCP. Protokol ini bersifat *connectionless* dan *unreliable* dalam pengiriman data. *Connectionless* berarti tidak diperlukannya suatu bentuk hubungan terlebih dahulu untuk mengirimkan data. *Unreliable* berarti pada protokol ini tidak dijamin akan sampai pada tujuan yang benar dan dalam kondisi yang benar pula. Kehandalan pengiriman data pada protokol ini menjadi tanggung jawab dari program aplikasi pada *layer* atasnya. Jika dibandingkan dengan TCP, UDP adalah protokol yang lebih sederhana dikarenakan proses yang ada didalamnya lebih sedikit. Dengan demikian aplikasi yang memanfaatkan UDP sebagai protokol *transport* dapat mengirimkan data tanpa melalui proses pembentukan koneksi terlebih dahulu. Hal ini pun terjadi pada saat mengakhiri suatu koneksi, sehingga dalam banyak hal proses yang terjadi sangatlah sederhana dibanding jika mengirimkan data melalui protokol TCP.

Protokol UDP akan melakukan fungsi *multiplexing/demultiplexing* seperti yang dilakukan protokol TCP, bila suatu program aplikasi akan memanfaatkan protokol UDP untuk mengirimkan informasi dengan menentukan nomor *port* pengirim (*source port)* dan nomor *port* penerima *(destination port),* kemudian menambahkan sedikit fungsi koreksi kesalahan lalu meneruskan segmen yang terbentuk ke protokol *layer internet.* Pada *layer Internet*segmen tersebut ditambahi informasi dalam bentuk datagram IP dan keudian ditentukan cara terbaik untuk mengantarkan segmen tersebut ke sisi penerima. Jika segmen tersebut tiba pada sisi penerima, protokol UDP

menggunakan nomor *port* informasi IP pengirim dan penerima untuk mengantarkan data dalam segmen ke proses program aplikasi yang sesuai.

Beberapa hal yang harus diperhatikan jika suatu program aplikasi akan menggunakan protokol UDP sebagai protokol *transport:*

- Tidak ada pembentukan koneksi. Protokol UDP hanya mengirim informasi begitu saja tanpa melakukan proses awal sebelumnya.
- Tidak ada pengkondisian koneksi. Protokol UDP tidak melakukan penentuan kondisi koneksi yang berupa parameter-parameter seperti *buffer* kirim dan terima, kontrol kemacetan, nomor urutan segmen, dan *acknowledgement*.
- Memiliki *header kecil*. Protokol UDP meiliki 8 *byte header* dibanding 20 *header byte* pada TCP.
- Tidak ada pengaturan laju pengiriman. Protokol UDP hanya menekankan kecepatan kirim pada laju program aplikasi dalam menghasilkan data, kemampuan sumber kirim (berdasarkan CPU, laju pewaktuan, dan lain-lain) dan *bandwidth* akses menuju Internet. Jika terjadi kemacetan jaringan, sisi penerima tidak perlu menerima seluruh data yang dikirim. Dengan demikian laju penerimaan data dibatasi oleh faktor kemacetan jaringan yang terjadi, walaupun pada sisi kirim tidak memperhatikannya.

2.2.5 IP *(InternetProtocol)*

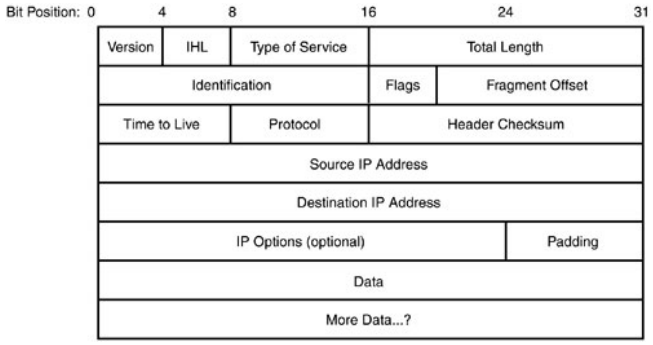
IP merupakan protokol yang paling penting yang berada pada *layer Internet*TCP/IP. Semua protokol TCP/IP yang berasal dari *layer* atasnya mengirimkan data melalui protokol IP ini. Seluruh data harus dilewatkan, diolah oleh protokol IP dan dikirimkan sebagai datagram IP untuk sampai ke sisi penerima. Dalam melakukan pengiriman data, protokol IP ini bersifat *unreliable, connectionless* dan datagram *delivery service*.

Unreliable berarti protokol IP tidak menjamin datagram yang dikirim pasti sampai ke tujuan. Protokol IP hanya melakukan cara terbaik untuk menyampaikan datagram yang dikirim ke tujuan. Jika pada perjalanan datagram tersebut terjadi hal-

hal yang tidak diinginkan (putusnya jalur, kemacetan, atau sisi penerima yang dituju sedang mati), protokol IP hanya memberikan pemberitahuan pada sisi kirim kalau telah terjadi permasalahan pengiriman data ke tujuan melalui protokol ICMP. *Connectionless* berarti tidak melakukan pertukaran kontrol informasi *(handshake)* untuk membentuk koneksi sebelum mengirimkan data.

Datagram *delivery service* berarti setiap datagram yang dikirim tidak tergantung pada datagram yang lainnya. Dengan demikian kedatangan datagram pun bisa jadi tidak berurutan. Metode ini dipakai untuk menjamin sampainya datagram ketujuannya, walaupun salah satu jalur menuju tujuan mengalami masalah.

2.2.5.1 IPv4 (IP vesrion 4)



Gambar 2.4 Format datagram IPv4

Pada bagian *header* dari protokol IPv4 seperti yang terlihat pada gambar 2.4 terdapat bagian pengalamatan sumber kirim dan tujuan masing-masing sebesar 32-bit. Pengalamatan *(IP Addressing)* adalah bagian yang terpenting dalam jaringan TCP/IP. Alamat inilah yang sering dinamakan sebagai alamat internet yang harus dimiliki setiap node yang terhubung dalam jaringan internet. Format IP Address yang dinyatakan dalam bilangan 32-bit dimana tiap 8 bitnya dipisahkan oleh tanda titik. Untuk memudahkan distribusinya, IP Address dibagi dalam beberapa kelas. Pembagian IP Address bisa dilihat sebagai berikut ini:

1. **Kelas A**

   Jika bit pertama dari IP address adalah 0, maka IP tersebut digolongkan dalam kelas A seperti berikut :

0 – 127	0 – 255	0 – 255	0 -255
0XXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX

   IP address kelas A biasanya digunakan untuk IP *backbone*.

2. **Kelas B**

   Jika 2 bit pertama dari IP address adalah 10, maka IP tersebut digolongkan dalam kelas B sebagai berikut :

128 – 191	0 - 255	0 – 255	0 -255
10XXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX

   IP address kelas B biasanya digunakan untuk IP publik dan ISP.

3. **Kelas C**

   Jika 3 bit pertama dari IP address adalah 110, maka IP tersebut digolongkan dalam kelas C sebagai berikut :

192– 223	0 - 255	0 – 255	0 -255
10XXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX

IP address kelas C biasanya digunakan untuk internet, IP kelas ini yang biasanya kita pakai.

4. **Kelas D**

   Jika 4 bit pertama dari IP address adalah 1110, maka IP tersebut digolongkan dalam kelas D sebagai berikut :

224– 239	0 - 255	0 – 255	0 -255
1110XXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX

   IP address kelas D biasanya digunakan untuk *broadcast* (Tv, Radio)

5. **Kelas E**

   Jika 4 bit pertama dari IP *address* adalah 1111, maka IP tersebut digolongkan dalam kelas E sebagai berikut :

240– 255	0 - 255	0 – 255	0 -255
1111XXX	XXXXXXXX	XXXXXXXX	XXXXXXXX

   Sedangkan IP *address* kelas E dicadangkan untuk kegiatan ekperimental.

2.2.6 *Ethernet*

Protokol *Ethernet* diciptakan oleh perusahaan *Xerox* sekitar tahun 1970. Pada tahun 1980, perusahaan *Xerox* bersama dengan perusahaan *Digital Equipment Corporation* (DEC) dan *Intel* menciptakan spesifikasi *Ethernet* versi-2 yang kompatibel dengan spesifikasi *IEEE 802.3*.

Protokol *Ethernet* sejauh ini adalah yang paling banyak digunakan, *Ethernet* menggunakan metode akses yang disebut CSMA/CD (*Carrier Sense Multiple Access/Collision Detection)*. Sistem ini menjelaskan bahwa setiap komputer memperhatikan ke dalam kabel dari *network* sebelum mengirimkan sesuatu kedalamnya. Jika dalam jaringan tidak ada aktifitas transmisi data, komputer akan mentransmisikan data. Jika ada transmisi lain di dalam kabel, komputer akan menunggu dan akan mencoba kembali mentransmisikan jika tidak ada aktivitas transmisi data. Terkadang dua buah komputer melakukan transmisi pada saat yang sama, ketika hal ini terjadi masing-masing komputer akan mundur dan akan menunggu kesempatan secara acak untuk mentransmisikan data kembali. Metode ini dikenal dengan *collision*, dan tidak akan berpengaruh pada kecepatan transmisi dari *network.* [1]

Protokol *Ethernet* dapat digunakan pada model jaringan *Bus, Star*, atau *Tree*. Data dapat ditransmisikan melewati kabel *twisted pair*, c*oaxial,* ataupun kabel *fiber optic* pada kecepatan 10 Mbps.

Jenis-jenis *Ethernet* diantaranya yaitu:

1. *Ethernet*

   Memiliki kecepatan akses data 10 Mbit/detik. Standar yang digunakan adalah: *10BaseT, 10BaseF, 10Base2 dan 10Base5.*

2. *Fast Ethernet*

   Memiliki kecepatan akses data 100 Mbit/detik. Standar yang digunakan adalah: *100BaseFX, 100BaseT*, *100BaseT4* dan *100BaseTX*. Protokol ini cepat menjadi populer, karena memberikan kecepatan 10 kali lebih tinggi dibandingkan *10BaseT* dengan harga yang relatif murah.

3. *Gigabit Ethernet*

   Memiliki kecepatan akses data 1000 Mbit/detik atau 1 Gbit/detik. Standar yang digunakan adalah: *1000BaseCX, 1000BaseLX, 1000BaseSX* dan *1000BaseT*. *Gigabit Ethernet* merupakan protokol jenis *Ethernet* terbaru yang mendukung kecepatan 1000 Mbps.

2.3 Topologi Fisik Jaringan Komputer

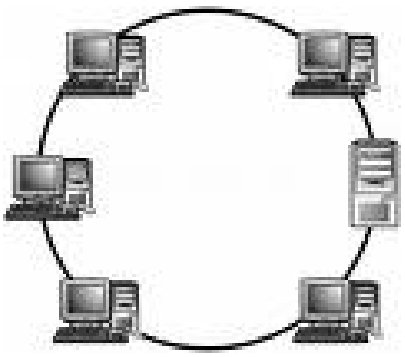
Topologi fisik adalah bagian yang menjelaskan hubungan antar komputer yang dibangun berdasarkan kegunaan, keterbatasan, *resource* dan keterbatasan biaya. Dengan demikian, topologi-topologi jaringan yang ada bisa disesuaikan keadaan dilapangan. Topologi ini menjelaskan hubungan perkabelan dan lokasi *node* atau *workstation.*

Jenis-jenis topologi fisik jaringan komputer diantaranya, yaitu:

- Topologi Cincin *(Ring Topology)*
- Topologi Bus *(Bus Topology)*
- Topologi Bintang *(Star Topology)*
- Topologi *Mesh*
- Topologi *Tree*

2.3.1 Topologi Cincin *(Ring Topology)*

Topologi jenis cincin ini menghubungkan satu komputer ke dalam satu *loop* tertutup. Pada topologi jenis ini, data atau *message* berjalan mengelilingi jaringan dengan satu arah pengiriman ke komputer selanjutnya terus hingga mencapai komputer yang dituju. Waktu yang dibutuhkan untuk mencapai terminal tujuan disebut *walk time* (waktu transmisi).



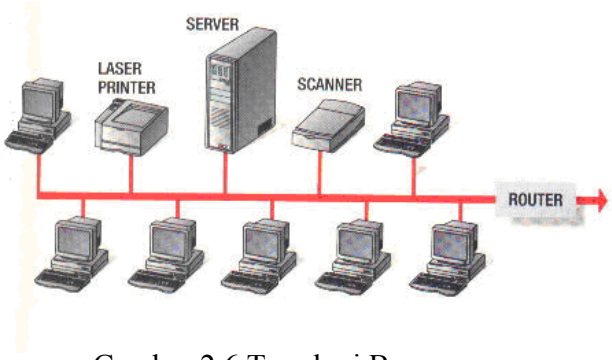
Gambar 2.5 Topologi Cincin

Ada dua hal yang dilakukan oleh suatu terminal ketika menerima data dari komputer sebelumnya, yaitu:

- Memeriksa alamat yang dituju dari data tersebut dan menerimanya jika terminal ini merupakan tujuan data tersebut.
- Terminal akan meneruskan data ke komputer selanjutnya dengan memberikan tanda negatif ke komputer pengirim.

2.3.2 Topologi Bus *(Bus Topology)*

Topologi jaringan jenis ini menggunakan sebuah kabel pusat yang merupakan media utama dari jaringan. Topologi bus terdiri dari satu jalur kabel utama dimana pada masing-masing ujungnya diberikan sebuah *terminator.* semua *nodes* pada jaringan (*file server, workstasion,* dan perangkat lainnya) terkoneksi ke sebuah kabel utama *(backbone).* Jaringan ini biasanya menggunakan kabel koaksial sebagai media transmisinya. Jaringan-jaringan *Ethernet* dan *Local Talk* menggunakan jaringan ini.



Gambar 2.6 Topologi Bus

Kelebihan dari topologi bus adalah, sebagai berikut:

1. Mudah dalam mengkonfigurasi komputer atau perangkat lain ke sebuah kabel utama.
2. Tidak terlalu banyak menggunakan kabel dibandingkan dengan topologi bintang *(star)*.

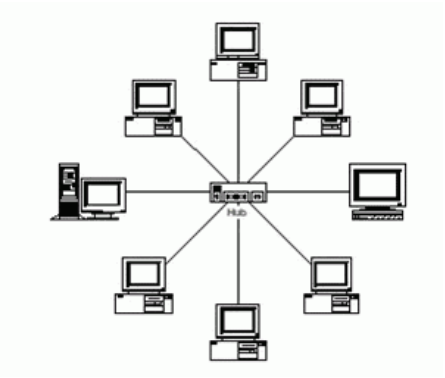
Kekurangan topologi bus adalah, sebagai berikut:

1. Seluruh jaringan akan mati jika ada kerusakan pada kabel utama *(backbone)*

2. Membutuhkan terminator pada kedua sisi dari kabel utamanya
3. Sangat sulit mengidentifikasi permasalahan jika jaringan sedang rusak
4. Sangat tidak disarankan dipakai sebagai salah satu solusi pada penggunaan jaringan di gedung besar.

2.3.3 Topologi Bintang *(Star Topology)*

Jenis topologi jaringan ini menggunakan satu terminal sebagai terminal sentral yang menghubungkan ke semua terminal *client.* Terminal sentral ini yang mengarahkan setiap data yang dikirimkan ke komputer yang dituju. Jenis jaringan ini apabila ada salah satu terminal client tidak berfungsi atau media transmisi terputus, maka tidak akan mempengaruhi kerja dari jaringan, karena gangguan tersebut hanya mempengaruhi terminal yang bersangkutan.



Gambar 2.7 Topologi *Star*

Kelebihan dari topologi *star* adalah sebagai berikut:

1. Mudah didalam pemasangan dan pengkabelan
2. Tidak mengakibatkan gangguan pada jaringan ketika akan memasang atau memindahkan perangkat jaringan lainnya
3. Mudah mendeteksi kesalahan dan memindahkan perangkat-perangkat lainnya.

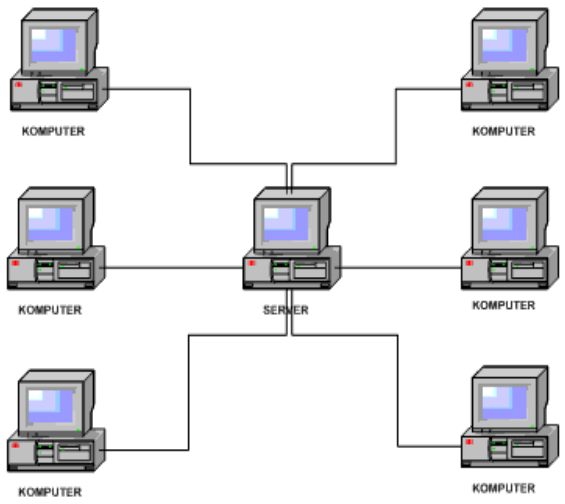
Kekurangan topologi *star* adalah sebagai berikut:

1. Membutuhkan lebih banyak kabel daripada topologi *bus*

2. Membutuhkan *concentrator,* dan bilamana concentrator tersebut rusak, maka semua node yang terkoneksi tidak dapat terdeteksi
3. Lebih mahal daripada topologi bus, karena biaya untuk pengadaan *concentrator.*

2.3.4 Topologi *Mesh*

Topologi jaringan ini menerapkan hubungan antar sentral secara penuh. Jumlah saluran harus disediakan untuk membentuk jaringan *Mesh* adalah jumlah sentral dikurangi 1 (n-1, n = jumlah sentral). Tingkat kerumitan jaringan sebanding dengan meningkatnya jumlah sentral yang terpasang. Dengan demikian disamping kurang ekonomis juga relatif mahal dalam pengoperasiannya.



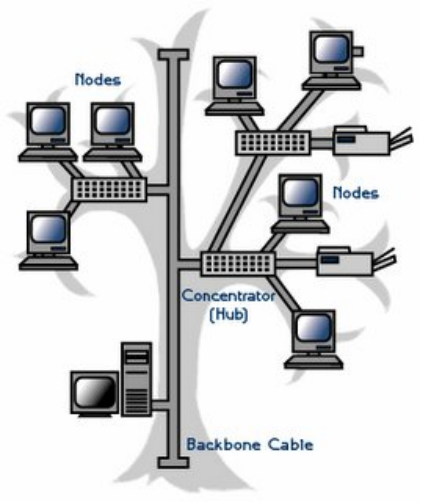
Gambar 2.8 Topologi *Mesh*

2.3.5 Topologi *Tree*

Topologi Jaringan Pohon *(Tree)* ini disebut juga sebagai topologi jaringan bertingkat. Topologi ini biasanya digunakan untuk interkoneksi antar sentral dengan hirarki yang berbeda. Untuk hirarki yang lebih rendah digambarkan pada lokasi yang

rendah dan semakin keatas mempunyai hirarki semakin tinggi. Topologi jaringan jenis ini cocok digunakan pada sistem jaringan komputer .

Pada jaringan pohon, terdapat beberapa tingkatan simpul *(node).* Pusat atau simpul yang lebih tinggi tingkatannya, dapat mengatur simpul lain yang lebih rendah tingkatannya. Data yang dikirim perlu melalui simpul pusat terlebih dahulu. Misalnya untuk bergerak dari komputer dengan *node-3* ke komputer *node-7* seperti halnya pada gambar 2.6 di bawah ini, data yang ada harus melewati *node-3, 5* dan *node-6* sebelum berakhir pada *node-7.* Keungguluan jaringan model pohon seperti ini adalah, dapat terbentuknya suatu kelompok yang dibutuhkan pada setiap saat. Sebagai contoh, perusahaan dapat membentuk kelompok yang terdiri atas terminal pembukuan, serta pada kelompok lain dibentuk untuk terminal penjualan. Adapun kelemahannya adalah, apabila simpul yang lebih tinggi kemudian tidak berfungsi, maka kelompok lainnya yang berada dibawahnya akhirnya juga menjadi tidak efektif. Cara kerja jaringan pohon ini relatif menjadi lambat.



Gambar 2.9 Topologi *Mesh*

2.4 Pengenalan Sistem Operasi

Tugas dari sistem operasi yaitu mengurus jalannya program diatasnya, koordinasi *input, output,* pemrosesan, serta pemasangan dan pembuangan *software.* Bagian kode

yang melakukan tugas-tugas inti dan umum dinamakan dengan *kernel* suatu sistem operasi. Sistem operasi dapat dikatakan sebagai penghubung antara pengguna dengan *hardware* komputer. Sistem operasi merupakan *software* pada lapisan pertama yang ditaruh pada memori komputer pada saat komputer dinyalakan.

2.4.1 Sistem Operasi *Linux*

Linux merupakan sistem operasi bebas yang sangat populer. *Linux* ialah salah satu sistem operasi *open source* sehingga para pengguna *Linux* dapat melakukan perbaikan – perbaikan terhadap sistem operasi ini. Inilah yang menyebabkan terdapat banyak versi dari *Linux* itu sendiri. *Linux* merupakan suatu sistem operasi yang bersifat *free,* dalam hal ini yang dimaksudkan bukan gratis, tetapi *Linux* membicarakan kebebasan pada penggunanya untuk mengetahui kode sumbernya serta dapat melakukan pengubahan terhadap kode sumbernya, serta dapat melakukan pengubahan terhadap kode sumber itu sendiri sesuai dengan keinginan dan kebutuhan pengguna di bawah lisensi GPL (GNU *Public License*) sehingga *Linux* disebut sebagai sistem operasi *open source*.

Nama *Linux* berasal dari *Kernel Linux*, yang ditulis oleh Linus Torvalds pada tahun 1991. Distribusi *Linux* biasanya disebut sebuah distro, yang merupakan suatu kumpulan *system software* dan kumpulan *software* aplikasi yang dapat diunduh dan dipasang melalui internet, sehingga pengguna dapat mengadaptasi sistem operasi sesuai dengan kebutuhannya. Salah satu varian dari *Linux* adalah *OpenSUSE.*
OpenSUSE adalah distro *Linux* versi komunitas yang didukung dan disponsori oleh *Novell. OpenSUSE* merupakan distro *Linux opensource* dan gratis yang menjadi dasar pengembangan bagi distro *Linux* komersil yang disediakan oleh *Novell*, *SUSE Linux Enterprise Server* (SLES) dan *SUSE Linux Enterprise Desktop* (SLED).

Salah satu keunggulan utama dari *OpenSUSE* dibandingkan distro *Linux* lainnya adalah kelengkapan pustaka dan berlimpahnya *software* yang disertakan. Bersama *Red Hat, SUSE* adalah distro *Linux* versi awal yang terus bertahan dan berkembang hingga sekarang. Banyak orang yang takut menggunakan *OpenSUSE* karena bias

pada lisensi yang digunakan. *OpenSUSE* adalah distro *Linux* yang *free* dan *open source. OpenSUSE* dapat digunakan secara bebas dan tanpa biaya. Jika suatu perusahaan atau lembaga menginginkan varian distro berbasis *SUSE* yang disertai dukungan *support*, tersedia SLES dan SLED. *Feature* yang sudah stabil dan sudah teruji pada *OpenSUSE* merupakan dasar dari *software* yang disertakan pada SLES dan SLED. Distro *Linux openSUSE* ini stabil, mudah dalam melakukan deteksi perangkat keras, mudah dikelola dan didukung penuh oleh komunitas pengembang di seluruh dunia serta memiliki dukungan sponsor dari perusahaan besar.

2.5 *IPTables*

IPTables adalah suatu *tools* dalam sistem operasi *Linux* yang berfungsi sebagai alat untuk melakukan *filter* (penyaringan) terhadap lalu lintas data. Bisa diartikan juga sebagai pengatur lalu lintas data.

2.5.1 Sintaks *IPTables:*

iptables [-t table] command [match] [target/jump]

untuk lebih jelasnya tentang sintaks *iptables,* bisa dilihat pada keterangan di bawah ini:

- *Table*

  IPTables memiliki 3 buah tabel, yaitu NAT, *MANGLE* dan *FILTER*. Penggunannya disesuaikan dengan sifat dan karakteristik masing-masing. Fungsi dari masing-masing tabel tersebut sebagai berikut :

  1. NAT : Secara umum digunakan untuk melakukan *Network Address Translation.* NAT adalah penggantian *field* alamat asal atau alamat tujuan dari sebuah paket.
  2. *MANGLE* : Digunakan untuk melakukan penghalusan *(mangle)* paket, seperti TTL, TOS dan *MARK*.

3. *FILTER* : Secara umum, inilah pemfilteran paket yang sesungguhnya.. Di sini bisa dintukan apakah paket akan di-*DROP, LOG, ACCEPT* atau *REJECT*

- *Command*

  Command pada baris perintah *Iptables* akan memberitahu apa yang harus dilakukan terhadap lanjutan sintaks perintah. Umumnya dilakukan penambahan atau penghapusan sesuatu dari tabel atau yang lain.

- *Match*

  Match dibagi kedalam 3 bagian, yaitu:

  1. *Generic Matches*

     Generic matches artinya pendefinisian kriteria yang berlaku secara umum. Dengan kata lain, sintaks *generic matches* akan sama untuk semua protokol. Setelah protokol didefinisikan, maka baru didefinisikan aturan yang lebih spesifik yang dimiliki oleh protokol tersebut. Hal ini dilakukan karena tiap-tiap protokol memiliki karakteristik yang berbeda, sehingga memerlukan perlakuan khusus.

  2. *Implicit Matches*

     Implicit Matches adalah *match* yang spesifik untuk tipe protokol tertentu. *Implicit Match* merupakan sekumpulan *rule* yang akan di-*load* setelah tipe protokol disebutkan. Ada 3 *implicit match* berlaku untuk tiga jenis protokol, yaitu TCP *matches*, UDP *matches* dan ICMP *matches.*

  3. *Explicit Matches*

- *Target/Jump*

  Target atau *jump* adalah perlakuan yang diberikan terhadap paket-paket yang memenuhi kriteria atau *match. Jump* memerlukan sebuah *chain* yang lain dalam tabel yang sama. *Chain* tersebut nantinya akan dimasuki oleh paket yang memenuhi kriteria. Analoginya ialah *chain* baru nanti berlaku sebagai prosedur/fungsi dari program utama. Sebagai contoh dibuat sebuah *chain* yang bernama *tcp_packets*. Setelah ditambahkan aturan-aturan ke dalam *chain* tersebut, kemudian chain tersebut akan direferensi dari *chain input*.

2.6 *Apache Web Server*

Apache web server adalah *web server* yang dapat dijalankan di banyak sistem operasi *(Unix,* BSD*, Linux, Microsoft Windows* dan *Novell Netware* serta *platform* lainnya) yang berguna untuk melayani dan memfungsikan situs *web*. Protokol yang digunakan untuk melayani fasilitas *web* ini menggunakan *Hypertext Transfer Protocol* (HTTP). *Apache* memiliki fitur-fitur canggih seperti pesan kesalahan yang dapat dikonfigur, autentikasi berbasis basis data dan lain-lain. *Apache* juga didukung oleh sejumlah antarmuka pengguna berbasis grafik (GUI) yang memungkinkan penanganan *server* menjadi mudah. *Apache* merupakan perangkat lunak sumber terbuka dikembangkan oleh komunitas terbuka yang terdiri dari pengembang-pengembang dibawah naungan *Apache Software Foundation.(ASF).*

Apache Web Server mempunyai kelebihan sebagai berikut :

1. *Apache* termasuk dalam kategori *freeware*.
2. *Apache* mudah sekali proses instalasinya jika dibanding *web server* lainnya seperti NCSA *(National Center for Supercomputing Applications),* IIS *(InternetInformation Server)*, dan lain-lain.
3. Mampu beroperasi pada berbagai *platform* sistem operasi.
4. Mudah mengatur konfigurasinya. *Apache* mempunyai hanya empat *file* konfigurasi.
5. Mudah dalam menambahkan *peripheral* lainnya ke dalam *platform web server*-nya.

2.7 PHP *(Personal Home Page/PHP Hypertext Processor)*

PHP adalah bahasa *scripting* yang menyatu dengan HTML dan dijalankan pada *server side*. Artinya semua sintaks yang diberikan akan sepenuhnya dijalankan pada *server* sedangkan yang dikirimkan ke *browser* hanya hasilnya saja. PHP bersifat *open source* dan telah digunakan oleh hampir seluruh *web developer* di seluruh dunia. Karena sifatnya yang *open source* dan semakin banyaknya *user* yang menggunakannya, membuat bahasa pemrograman ini mengalami perkembangan yang

sangat cepat. PHP hampir dapat berjalan di semua sistem operasi seperti *Windows, Unix, Linux* dan varinnya*, Mac OS X, RIC OS* dan lain-lain. PHP juga bisa berjalan hampir di semua *web server* yang ada sekarang ini, seperti *Apache Web Server,* IIS*, Personal Web Server, Caudium, Xitami, Omni* dan masih banyak lagi. Dengan begitu sistem operasi dan *web server* apapun yang digunakan, PHP dapat berjalan dengan baik.

2.8 *Mysql*

Mysql adalah *database* yang cepat dan tangguh, dan sangat cocok jika digabungkan dengan PHP. *Database* adalah kumpulan informasi yang disimpan di dalam komputer secara sistematik sehingga dapat diperiksa menggunakan suatu program komputer untuk memperoleh informasi dari basis data tersebut. Dengan *database* kita bisa meyimpan, mencari dan mengklasifikasikan data dengan lebih akurat dan professional. *Mysql* menggunakan SQL *language (Structure Query Language)* artinya *Mysql* menggunakan *query* atau bahasa pemrograman yang sudah standar di dalam dunia *database.*

Kelebihan *Mysql* diantaranya:

1. Dari segi performa*, Mysql* tidak bisa diragukan, pemrosesan *database* sangat cepat
2. Bersifat *open source*
3. Mudah untuk dipelajari
4. Kompabilitas dengan berbagai sistem operasi dan *web server* yang ada

2.9 *PhpMyAdmin*

PhpMyAdmin adalah sebuah aplikasi yang ditulis dalam PHP yang berbasis *web* yang memungkinkan pengguna mengadministrasi *database mysql*. Dengan *PhpMyAdmin* konfigurasi *mysql* dapat dilakukan dengan mudah dan cepat.

2.10 *Hierarchical Token Bucket* (HTB)

HTB adalah aplikasi yang berfungsi untuk mengatur pembagian *bandwidth,* pembagian dilakukan secara hirarki yang dibagi-bagi kedalam kelas sehingga mempermudah pengaturan *bandwidth*. HTB merupakan teknik penjadwalan paket yang sering digunakan bagi *router-router* berbasis *Linux*, dikembangkan pertama kali oleh Martin Devera.

Cara Kerja HTB tidak memiliki perbedaan dengan pendahulunya yaitu CBQ (*Class-Based Queueing*) hanya saja pada *General Scheduler* HTB menggunakan mekanisme *Deficit Round Robin* (DRR) dan pada blok umpan baliknya, *Estimator* HTB tidak menggunakan *Eksponential Weighted Moving Average* (EWMA) melainkan *Token Bucket Filter* (TBF).

Pada HTB terdapat parameter *ceil* sehingga kelas akan selalu mendapatkan *bandwidth* diantara *base link* dan nilai *ceil link*-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 *link* ini (keduanya *leaves* dan interior dapat memiliki *ceil*). Sebagai catatan, apabila nilai *ceil* sama dengan nilai *base link*, maka akan memiliki fungsi yang sama seperti parameter *bounded* pada CBQ, di mana kelas-kelas tidak diijinkan untuk meminjam *bandwidth*. Sedangkan jika nilai *ceil* diset tak terbatas atau dengan nilai yang lebih tinggi seperti kecepatan *link* yang dimiliki, maka akan didapat fungsi yang sama seperti kelas *non-bounded*.

2.10.1 *General Scheduler HTB*

HTB menganggap hirarki kelas lengkap dan trafik dipisah-pisah menjadi beberapa aliran trafik, algoritma untuk penjadwalan paket adalah sebagai berikut: pertama memilih kelas pada cabang terendah (*leaf class*) yang *link*-nya belum mencapai batas kemudian mulai mengirimkan paket dari kelas yang memiliki prioritas tertinggi kemudian berlanjut ke yang rendah, apabila *link* semua kelas

melampaui batas *link* maka dilakukan suatu tes melalui suatu putaran lengkap untuk menemukan *leaf class* yang dapat meminjam *bandwidth* dari kelas diatasnya (*parent class*) jika tidak ada maka putaran diulangi dengan mencoba meminjam *bandwidth* dari kelas diatas *parent class* (*grandfather class*).

2.10.2 Mekanisme *Deficit Round Robin* (DRR)

Deficit Round Robin (DRR) merupakan skema penjadwalan hasil dari modifikasi *Weighted Round Robin* (WRR), DRR membolehkan tiap kelas/prioritas memiliki panjang paket yang berbeda-beda sehingga tidak diperlukan lagi informasi mengenai ukuran paket rata-rata. Ide dasar DRR adalah suatu kelas berhak untuk mengirimkan paket dalam suatu putaran jika ukuran paket yang dimilikinya lebih kecil atau sama dengan ambang batas yang ditetapkan, ambang batas ini dinamakan *Quantum* yang merupakan padanan terhadap pembagian layanan untuk mendapatkan status *Deficit Counter*. Pada saat permulaan tiap putaran (round), *Quantum* ditambahkan ke *Deficit Counter*, kemudian jika paket *Head-of-Line* (HoL) di antrian tersebut berukuran lebih kecil atau sama dengan *DeficitCounter*, maka paket HoL tersebut akan dikirimkan dan *Defict Counter* akan berkurang, dikurangi oleh ukuran paket HoL, jika tidak maka paket *HoL* tak akan dikirim pada putaran tersebut dan *Deficit Counter*nya berukuran sama dengan nilai *Quantum* dimasing-masing antrian. Bila di dalam antrian tidak ada paket yang akan dikirimkan, *Deficit Counter* akan dikembalikan ke nilai 0.

2.10.3 *Estimator*

Hierarchical Token Bucket (HTB) menggunakan *Token Bucket Filter* (TBF) sebagai *Estimator* untuk menentukan apakah suatu kelas/prioritas berada dalam keadaan *underlimit*, *atlimit* atau *overlimit*. TBF bekerja dengan dasar algoritma ember *token,* setiap paket yang akan dikirimkan harus memiliki *token* yang berada dalam ember *token*, jika *token* tak tersedia didalam ember maka paket-paket yang akan

dikirimkan harus menunggu sampai tersedia *token* yang cukup untuk mengirimkan paket yang sedang menunggu.

Implementasi TBF terdiri dari sebuah *buffer (bucket)*, yang secara konstan diisi oleh beberapa informasi virtual yang dinamakan *token,* pada *link* yang spesifik *(token link)*. Parameter paling penting dari *bucket* adalah ukurannya, yaitu banyaknya *token* yang dapat disimpan. Setiap *token* yang masuk mengumpulkan satu paket yang datang dari antrian data dan kemudian dihapus dari *bucket*. Dengan menghubungkan algoritma ini dengan dua aliran *token* dan data, akan didapati tiga buah kemungkinan skenario:

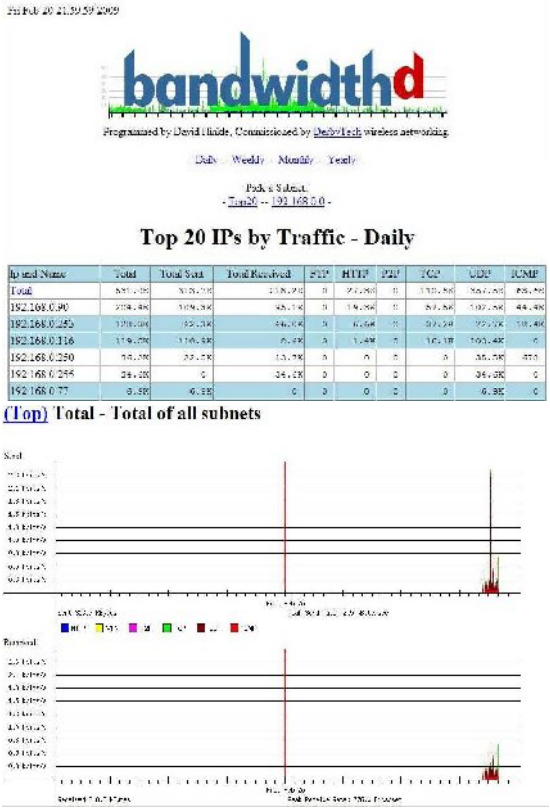
1. Data yang datang pada TBF memiliki *link* yang sama dengan masuknya *token*. Dalam hal ini, setiap paket yang masuk memiliki *token*-nya masing-masing dan akan melewati antrian tanpa adanya *delay*.
2. Data yang datang pada TBF memiliki *link* yang lebih kecil daripada *link token.* Hanya sebagian *token* yang dihapus pada *output* pada tiap paket data yang dikirim ke antrian, dan *token* akan menumpuk, memenuhi ukuran *bucket. Token* yang tidak digunakan kemudian akan dapat digunakan untuk mengirimkan data pada kecepatan yang melampaui *link token* standar, ini terjadi jika ada ledakan data yang pendek.
3. Data yang datang pada TBF memiliki *link* yang lebih besar daripada *link token.* Hal ini berarti *bucket* akan segera kosong dari *token,* yang menyebabkan TBF akan menutup alirannya untuk sementara. Hal inilah yang dinamakan situasi *overlimit*. Jika paket-paket tetap datang, maka paket-paket akan segera dibuang.

2.10.4 *HTB.init*

HTB.INIT adalah suatu *shell script* yg berasal dari *CBQ.INIT* yang mempermudah pengaturan *traffic control HTB-BASED* di *Linux.* [6]

2.11 *Banwidthd*

Bandwidthd merupakan salah satu *tools* monitoring *bandwidth* yang mampu berjalan disemua distro *Linux*. *Bandwidthd* menampilkan akumulasi pemakaian *bandwidth* suatu jaringan TCP/IP selama periode tertentu dengan bentuk tabel/angka/grafik dalam format HTML. Grafik yang ditampilkan berdasarkan alamat IP masing-masing yang berada dalam jaringan yang dipantau. Dari tabel IP yang dipantau, dapat dilihat *hostname* dari IP tersebut (jika terdapat *Name Server* yang dapat me-*resolve* IP tersebut), total *bandwidth* yang digunakan *(sent/receive),* pemakaian tcp, udp, icmp, http, p2p, vpn, semuanya dapat dilihat. Gambar 2.10 akan menunjukan tampilan *(screenshoot)* dari *tools bandwidthd.*



Gambar 2.10 *Tools Bandwidthd*