BAB 3
PROTOKOL DAN FUNGSI LAPISAN APLIKASI

Capaian Pembelajaran :

- mahasiswa dapat menjelaskan fungsi, proses yang terjadi dan jenis – jenis protokol yang ada pada lapisan-lapisan atas dari model OS dan TCP **(C2)**

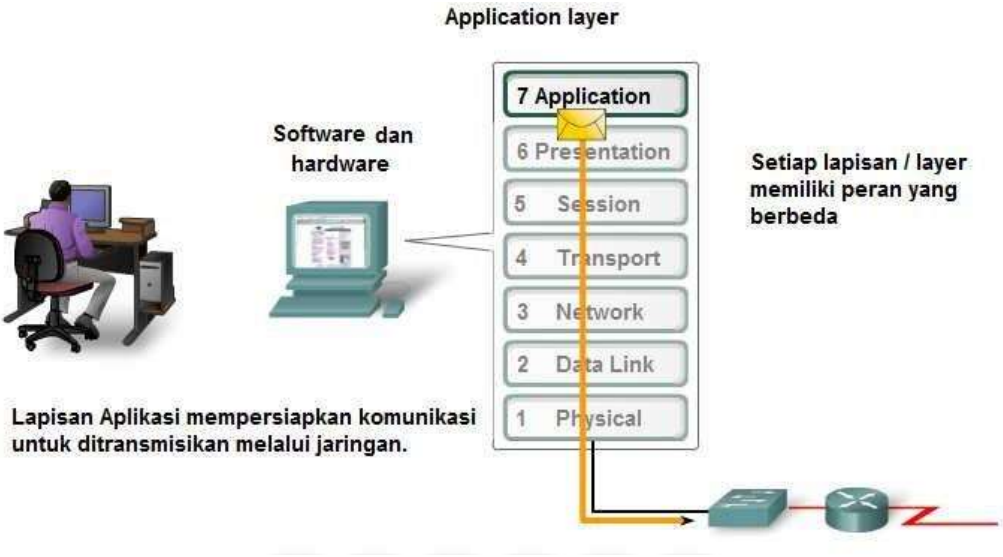
3.1 APLIKASI – INTERFACE ANTAR JARINGAN

3.1.1 Model OSI dan TCP/IP

Model referensi OSI adalah representasi abstrak berlapis, dibuat sebagai pedoman untuk mendesain protokol jaringan. Model OSI membagi proses jaringan menjadi tujuh lapisan logis, masing-masing memiliki fungsi yang unik dan bertugas memberi layanan tertentu serta memiliki protokol – protokol tertentu yang spesifik.

Dalam model ini, informasi dilewatkan dari satu lapisan ke lapisan yang berikutnya, dimulai dari lapisan aplikasi pada *host* transmisi, dilanjutkan ke lapisan-lapisan di bawahnya hingga ke lapisan fisik, kemudian melewati saluran komunikasi menuju ke *host* tujuan, di mana informasi tersebut melanjutkan kembali secara hirarki naik dari lapisan terbawah, hingga berakhir di lapisan aplikasi.

Lapisan Aplikasi, Lapisan ke tujuh, adalah lapisan paling atas, baik di model OSI dan model TCP/IP. Lapisan ini adalah lapisan yang menyediakan antarmuka antara aplikasi yang kita gunakan untuk berkomunikasi, dengan jaringan yang digunakan, di mana pesan kita ditransmisikan. Protokol lapisan aplikasi digunakan untuk melakukan pertukaran data antara program yang berjalan pada *host* sumber dan program yang berjalan pada *host* tujuan. Ada banyak protokol pada lapisan aplikasi.



Gambar 16. Lapisan Aplikasi

Meskipun protokol TCP / IP dibuat sebelum adanya model OSI, fungsi dari lapisan aplikasi model protokol TCP / IP, sangat sesuai jika dibandingkan dengan fungsi-fungsi dari tiga lapisan teratas dari model OSI, yaitu : Lapisan *Application*, *Presentation* dan *Session*.

Hampir semua protokol lapisan aplikasi dari model TCP/IP dibuat sebelum maraknya perkembangan PC, aplikasi grafis dan obyek-obyek multimedia. Akibatnya, protokol-protokol tersebut sangat sedikit mengimplementasikan fungsi-fungsi lapisan *Presentation* dan lapisan *Session* yang ditentukan model OSI.

Lapisan *Presentation*

Lapisan Presentasi memiliki tiga fungsi utama:

- Pengkodean (membuat format data) dan konversi data dari lapisan Aplikasi untuk memastikan data dari perangkat pengirim bisa dimengerti oleh aplikasi yang tepat di perangkat penerima.
- Kompresi data dengan cara yang dapat didekompresi oleh perangkat tujuan.

- Enkripsi data sebelum ditransmisi dan dekripsi data pada saat diterima oleh tujuan.

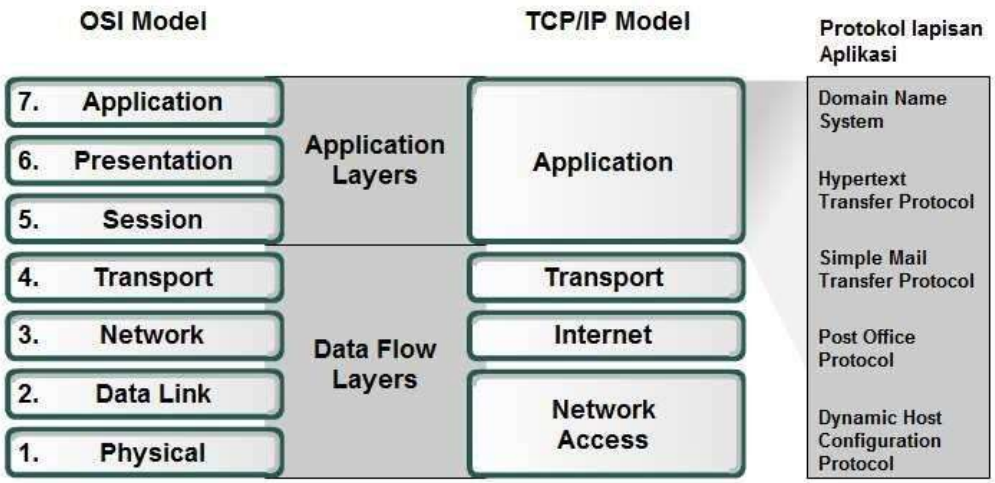
Implementasi lapisan Presentasi biasanya tidak terkait dengan penggunaan serangkaian protokol tertentu. Contohnya penggunaan format standar untuk video dan grafik. Beberapa format standar terkenal untuk video adalah QuickTime dan Motion Picture Expert Group (MPEG). QuickTime adalah spesifikasi standar video dan audio untuk Apple Computer, dan MPEG adalah standar umum untuk kompresi video dan format (kode) video.

Sedangkan untuk format gambar grafis yang terkenal adalah Graphics Interchange Format (GIF), Joint Photographic Experts Group (JPEG), dan TaggedImage File Format (TIFF). GIF dan JPEG adalah standar kompresi dan standar pengkodean untuk gambar grafis, dan TIFF adalah format pengkodean standar untuk gambar grafis – tanpa kompresi.

Lapisan *Session*

Seperti yang tersirat dari namanya, fungsi pada lapisan ini adalah untuk menciptakan dan menjaga dialog antara aplikasi sumber dan aplikasi tujuan. Lapisan *Session* menangani pertukaran informasi untuk memulai sutau dialog, menjaga agar tetap aktif, dan untuk memulai kembali sesi yang terganggu atau idle untuk jangka waktu yang panjang.

Kebanyakan aplikasi, seperti *web browser* atau *e-mail client*, menggabungkan fungsi dari lapisan-lapisan OSI 5, 6 dan 7.



Gambar 17. Persamaan fungsi lapisan Model OSI dan TCP / IP

Protokol-protokol lapisan Aplikasi TCP/IP yang dikenal secara luas adalah protokol-protokol yang menyediakan fungsi-fungsi untuk pertukaran informasi dari *user*. Protokol-protokol ini menetukan format-format data, dan informasi kontrol yang diperlukan dalam komunikasi internet secara umum. Protokol-protokol tersebut antara lain adalah :

- Protokol DNS (*Domain Name Service*) yang digunakan untuk menterjemahkan nama internet menjadi alamat IP
- HTTP (*Hypertext Transfer Protocol*) yang digunakan untuk mentransfer file-file yang membentuk halaman-halaman *web* dari WWW (*world wide web*).
- SMTP (*Simple Mail Transfer Protocol*) yang digunakan untuk pengiriman email dan attachment nya.
- telnet, protokol emulasi terminal, digunakan untuk menyediakan akses kontrol suatu *host* dari *host* lain yang terkoneksi dengan jaringan
- FTP (*File transfer protocol*) yang digunakan untuk mentransfer file secara interaktif.

3.1.2 Software Lapisan Aplikasi

Fungsi-fungsi dari protokol pada Lapisan Aplikasi memberikan interface antara *user* dengan jaringan komputer. Saat kita membuka *web browser* atau aplikasi *chatting*, program ditaruh di dalam memory sebuah perangkat dan dieksekusi. Untuk setiap program yang dijalankan dalam suatu perangkat, kita sebut dengan proses.

Dalam lapisan Aplikasi ada dua bentuk dari program *software* atau proses yang menyediakan akses kepada jaringan komputer : aplikasi dan servis.

Network-Aware Application

Network-aware *application*, aplikasi yang dimaksud di sini adalah program *software* yang digunakan oleh *user* untuk berkomunikasi di atas jaringan. Beberapa *software* aplikasi

bersifat *network-aware*, yang berarti mereka mengimplementasikan protokol-protokol lapisan aplikasi dan dapat berkomunikasi langsung dengan lapisan di bawahnya dari suaturangkaian tumpukan protokol. Contoh dari aplikasi seperti ini adalah *web browser* dan *email client.*

Servis Lapisan Aplikasi

Aplikasi lain mungkin membutuhkan servis dari lapisan aplikasi untuk menggunakan sumber daya jaringan, seperti transfer file atau *network print spooling*. Walaupun proses ini transparan bagi *user* (*user* tidak mengetahui ada proses ini), tetapi servis-servis ini merupakan program yang memberikan interfaceke jaringan dan mempersiapkan data untuk bisa ditransfer. Tipe data yang berbeda – entah itu teks, gambar , musik, atau video – membutuhkan servis jaringan yang berbeda untuk bisa diproses oleh fungsi-fungsi yang terjadi di lapisan bawahnya dari model OSI.

Setiap aplikasi atau servis jaringan menggunakan protokol yang menentukan standar dan format yang akan digunakan untuk suatu data. Tanpa adanya protokol, data dalam jaringan tidak akan memiliki prosedur yang sama dalam memberi format data maupun mengirim data.

3.1.3 Aplikasi *User*, Servis dan Protokol

Seperti yang disebutkan sebelumnya, lapisan Aplikasi menggunakan protokol-protokol yang diimplementasikan dalam aplikasi dan servis. Jika aplikasi menyediakan metode untuk membuat suatu pesan informasi dan servis menetapkan interface dengan jaringan; protokol menyediakan aturan-aturan dan format-format yang mengatur bagaimana data pesan informasi tersebut harus diperlakukan. Ketiga komponen tersebut bisa digunakan dalam satu program atau bahkan menggunakan nama yang sama untuk ketiga komponen tersebut (aplikasi, servis dan protokolnya). Contohnya, jika kita menyebut “Telnet” kita bisa saja mengacu ke aplikasinya, servisnya atau protokolnya.

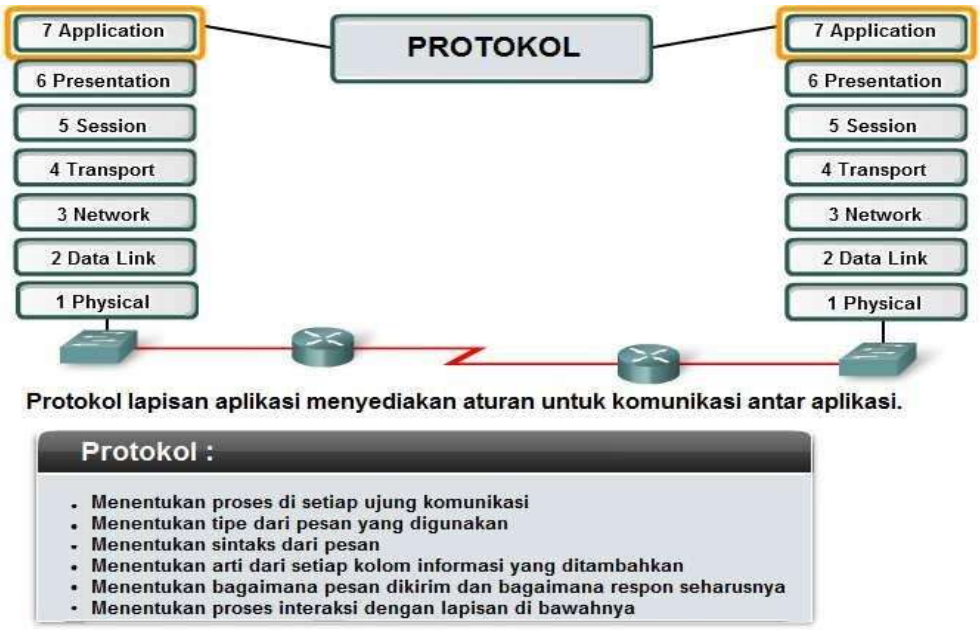
Pada model OSI, aplikasi-aplikasi yang berinteraksi langsung dengan *user*, berada di lapisan yang paling atas. Sama halnya dengan lapisan-lapisan lain, lapisan aplikasi ini mengandalkan fungsi-fungsi dari lapisan dibawahnya untuk bisa menyelesaikan proses komunikasi. Dalam lapisan aplikasi, protokol-protokol menentukan pesan-pesan yang

dipertukarkan antara *host* pengirim dan penerima, *syntax* dari perintah-perintah untuk mengontrolnya, tipe dan format data yang ditransmisikan, dan metode yang tepat untuk mendeteksi dan memperbaiki *error*/kesalahan yang terjadi.

3.1.4 Fungsi Protokol Layer Aplikasi

Protokol lapisan aplikasi digunakan di masing-masing perangkat pengirim maupun perangkat penerima selama sesi komunikasi berlangsung. Agar komunikasi dapat berjalan dengan sukses, protokol yang digunakan pada masing- masing perangkat harus benar-benar sama.

Protokol menetapkan aturan-aturan yang konsisten untuk pertukaran data antara aplikasi-aplikasi atau proses-proses yang berjalan pada perangkat- perangkat yang berkomunikasi. Protokol menentukan bagaimana struktur data yang ada di dalam pesan dan tipe pesan bagaimana yang dikirim antara pengirim dan penerima. Pesan-pesan ini bisa berupa *request for service* (permintaan layanan), *acknowledgment* (konfirmasi), data, pesan status, atau pesan error.



Gambar 18. Protokol Lapisan Aplikasi

Banyak jenis aplikasi berkomunikasi melalui jaringan data. Oleh karena itu, servis lapisan Aplikasi harus menerapkan beberapa protokol untuk memberikan semua layanan yang diinginkan dari setiap komunikasi. Setiap protokol memiliki tujuan tertentu dan memiliki karakteristik yang diperlukan untuk memenuhi tujuan itu. Setiap detail protokol di setiap lapisan harus diikuti fungsi-fungsi yang dimiliki suatu lapisan bisa bekerja sama dengan fungsi-fungsi lapisan di bawahnya.

3.2 CLIENT-SERVER DAN PEER-TO-PEER

3.2.1 Model Client-Server

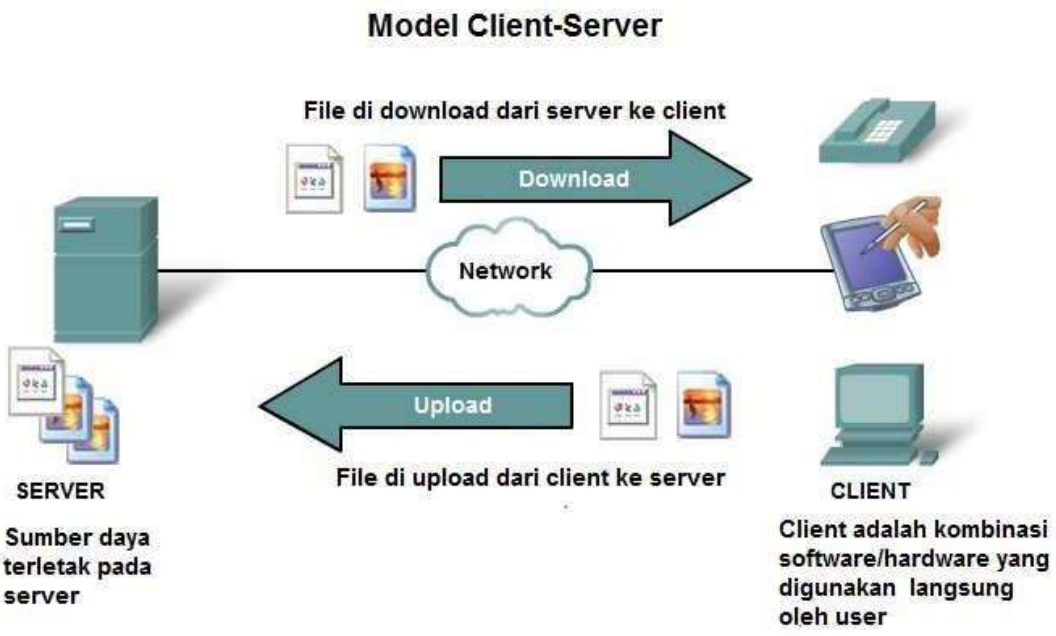
Saat seseorang berusaha mengakses informasi pada perangkatnya, apakah itu PC, laptop, PDA, telephone genggam, atau perangkat lain yang terkoneksi dengan jaringan, data yang diakses itu mungkin tidak terletak secara fisik pada perangkat tersebut. Jika itu yang terjadi, permintaan untuk mengakses informasi tersebut harus dibuat dan ditujukan kepada perangkat dimana informasi itu berada.

Pada model Client-Server, perangkat yang meminta informasi disebut dengan *client*, dan perangkat yang merespon permintaan disebut dengan *server*. Proses-proses *client* dan *server* termasuk berada di lapisan aplikasi. Client memulai pertukaran dengan meminta (request) data dari *server*, yang meresponnya dengan mengirimkan satu atau lebih aliran data ke *client*. Protokol lapisan aplikasi menjelaskan format dari permintaan dan respon antara *client* dan *server*. Selain data aktual yang ditransfer, pertukaran ini mungkin juga membutuhkan kontrol informasi, seperti autentikasi *user* dan identifikasi file data yang akan ditransfer.

Contoh jaringan *client*/*server* adalah lingkungan perusahaan dimana pegawai menggunakan *server* email perusahaan untuk mengirim, menerima dan menyimpan email. Email *client* pada komputer pegawai mengirimkan permintaan kepada *server* email untuk setiap email yang belum terbaca. Server merespon dengan mengirimkan email-email yang diminta kepada *client*.

Walaupun data secara khusus digambarkan sebagai aliran dari *server* ke *client*, beberapa data juga mungkin mengalir dari *client* ke *server.* Beberapa alirandata bisa sama banyaknya antara ke *client* maupun ke *server*, atau bahkan bisa saja lebih besar aliran data ke *server*,

contohnya *client* yang mentransfer file untukdisimpan di file *server.* **Transfer data dari *client* ke *server* disebut dengan *upload*, dan data dari *server* ke *client* disebut dengan *dowload*.**



Gambar 19. Model *Client-Server*

3.2.2 Server

Dalam konteks jaringan secara umum, setiap perangkat yang merespon permintaan (*request*) dari aplikasi-aplikasi *client*, berfungsi sebagai *server*. Server biasanya adalah sebuah komputer yang memiliki informasi untuk dibagi dengan banyak sistem *client*. Contohnya, halaman *web*, dokumen, database, gambar, video, dan file audio dapat disimpan di sebuah *server* dan dikirim ke *client-client* yang memintanya. Dalam kasus lain, seperti printer jaringan, *server* print mengirimkan permintaan pencetakan dari *client* ke printer yang dituju.

Tipe aplikasi *server* yang berbeda memiliki persyaratan yang berbeda- beda pula terhadap akses dari *client*. Beberapa *server* bisa membutuhkan autentikasi dari akun *user* untuk memeriksa apakah *user* tersebut memiliki ijin untuk mengakses data / operasi yang diminta. Beberapa *server* mengandalkan daftar terpusat dari akun-akun *user* dan otorisasi *user*. Saat menggunakan FTP *client*, contohnya, untuk melakukan *upload* data ke *server* FTP, kita bisa saja

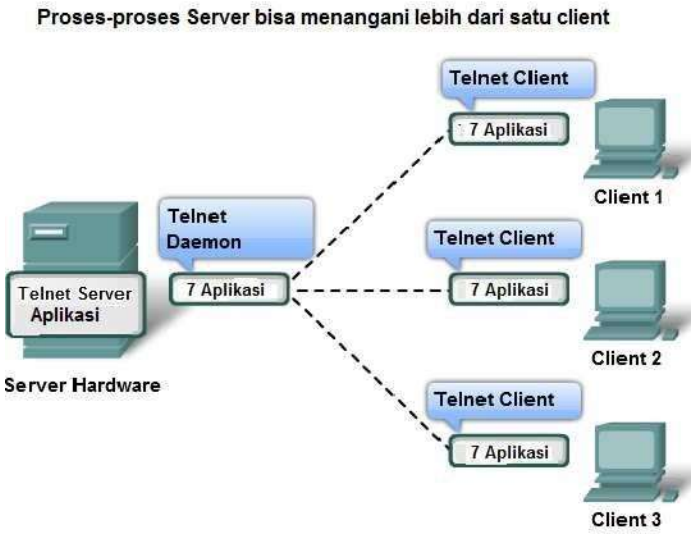
memiliki akses untuk menulis ke folder kita sendiri di *server*, tapi tidak memiliki akses untuk membaca file-file lain di situs yang sama.

Pada jaringan *client*/*server*, *server* menjalankan servis, atau proses, yang kadang disebut juga dengan *server daemon*. Seperti hampir semua servis, daemon berjalan secara background dan tidak langsung dibawah kendali/kontrol dari *user*. Daemon diibaratkan sebagai 'pendengar' dari permintaan *client* karena mereka diprogram untuk merespon jika *server* menerima permintaan terhadap suatu servis/layanan yang disediakan daemon. Saat daemon mendengar ada permintaan dari *client*, dia bertukar pesan-pesan yang sesuai dengan *client*, seperti yang ditentukan oleh protokolnya, dan dilanjutkan dengan mengirim data yang diminta *client* dalam format yang benar.

3.2.3 Servis dan Protokol Lapisan Aplikasi

Sebuah aplikasi bisa menggunakan beberapa dukungan servis lapisan aplikasi yang berbeda-beda; jadi, apa yang tampak oleh *user* sebagai sebuah permintaan terhadap suatu halaman *web* misalnya, pada kenyataanya, adalah terdiri dari sejumlah permintaan (request) yang tersendiri. Dan untuk setiap permintaan tunggal, bisa saja dijalankan beberapa proses untuk melayaninya. Sebuah contoh, sebuah *client* mungkin membutuhkan beberapa proses individual untuk membuat suatu request ke *server.*

Selain itu, *server-server* biasanya memiliki beberapa *client* yang meminta layanan dalam waktu yang sama. Contohnya, sebuah *server* Telnet bisa sajamemiliki beberapa *client* yang meminta koneksi terhadap dirinya. Permintaan- permintaan individual dari *client-client* ini harus ditangani secara simultan dan terpisah di dalam jaringan agar bisa teratasi dengan sukses. Proses-proses lapisan aplikasi dan servis-servisnya mengandalkan suport dari fungsi-fungsi lapisan-lapisan bawahnya agar bisa sukses mengatur beberapa koneksi.



Gambar 20. Proses Server

3.2.4 Jaringan dan Aplikasi Peer-to-PeerModel Peer-to-Peer

Selain model Client/Server untuk jaringan, ada juga model peer-to-peer. Jaringan peer-to-peer melibatkan 2 bentuk yang berbeda : desain jaringan peer-to-peer dan aplikasi peer-top-peer (P2P). Masing-masing bentuk memiliki fitur-fitur yang mirip, tetapi pada prakteknya bekerja dengan sangat berbeda.

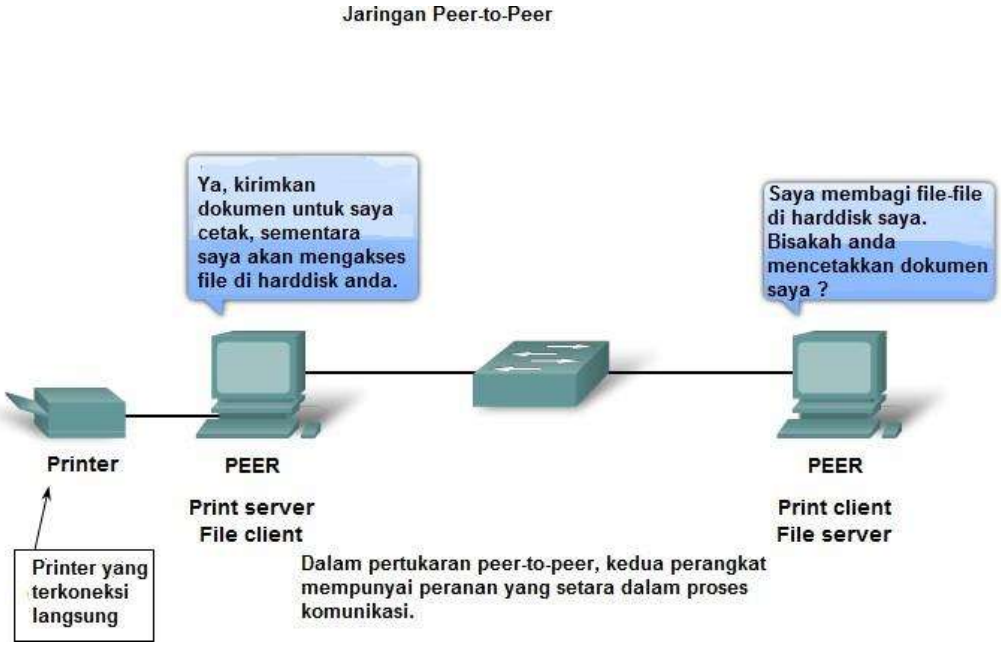
Jaringan peer-to-peer

Dalam jaringan peer-to-peer, 2 komputer atau lebih terkoneksi via sebuah jaringan dan dapat berbagi sumberdaya (seperti printer dan file) tanpa memiliki *server* khusus (*dedicated server*). Setiap *host* yang terkoneksi (disebut juga dengan peer) bisa berfungsi sebagai *server* dan sekaligus juga sebagai *client.* Sebuah kompter mungkin berperan sebagai *server* untuk sebuah transaksi sementara secara bersamaan bertindak sebagai *client* untuk transaksi lainya. Peranantara *server* dan klien diatur berdasarkan setiap permintaan.

Jaringan rumahan sederhana dengan 2 komputer yang terkoneksi saling berbagi printer adalah contoh dari jaringan peer-to-peer. Setiap orang dapat mengeset komputernya untuk

berbagai file-filenya, menjalankan game jaringan, atau berbagi koneksi internet. Contoh lain fungsional jaringan peer-to-peer adalah 2 komputer yang terkoneksi ke jaringan besar, menggunakan aplikasi-aplikasi *software* untuk berbagi sumber daya-sumber daya antar mereka lewat jaringan tersebut.

Tidak seperti model *client*/*server*, dimana menggunakan *server* khusus (dedicated *server*), jaringan peer-to-peer mendesentralisasi sumber daya pada jaringan. Informasi yang dipakai bersama tidak ditempatkan pada *server* khusus, tetapi bisa ditempatkan dimana saja pada perangkat/*host*/peer yang terkoneksi. Sebagian besar sistem operasi saat ini mendukung file dan print sharing tanpa memerlukan perangkat lunak tambahan. Karena jaringan peer-to-peer biasanya tidak mengganakan daftar akun-akun *user*, pengaturan akses atau pemantauan yang terpusat, sulit untuk menerapkan kebijakan keamanan dan akses dalam jaringan yang terdiri dari banyak komputer. Akun *user* dan hak akses harus diset secara individual pada setiap perangkat peer.



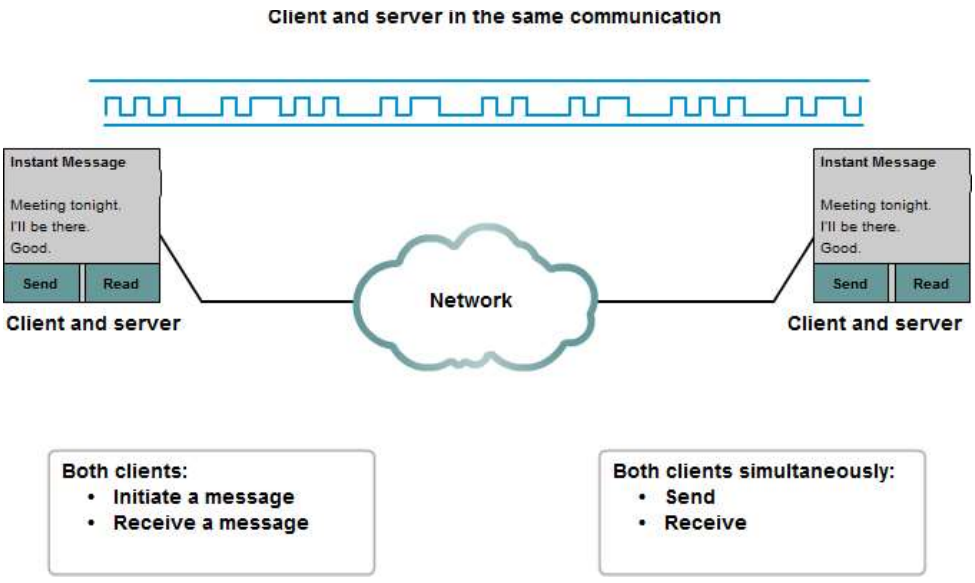
Gambar 21. Jaringan *Peer-to-Peer*

Aplikasi Peer-to-Peer

Sebuah aplikasi *peer-to-peer* (p2p), beda dengan jaringan peer-to-peer, mengijinkan sebuah perangkat untuk bertindak sebagai *client* sekaligus sebagai *server* dalam satu komunikasi yang sama. Dalam model ini, setiap *client* adalah *server* dan setiap *server* juga adalah *client.* Masing-masing dapat memulai sebuah komunikasi dan dianggap setara dalam proses komunikasi. Bagaimanapun juga, aplikasi peer-to-peer membutuhkan setiap *host* menyediakan sebuah *user* interface dan servis yang berjalan pada background. Saat anda menjalankan sebuah aplikasi p2p tertentu, aplikasi tersebut akan menjalankan *user* interface dan servis-servis pada background yang dibutuhkan. Selanjutnya perangkat-perangkat / *host-host* bisa langsung berkomunikasi.

Beberapa aplikasi p2p menggunakan sistem gabungan dimana sumber daya yang dibagi pakai terletak secara desentralisasi tetapi index-index yang mengacu terhadap lokasi – lokasi sumber daya tersebut terletak di sebuah direktori yang tersentral. Dalam sistem gabungan, setiap peer mengakses sebuah *server* index untuk mengetahui sumber daya yang tersimpan di peer lain. Index *server* juga bisamembantu dalam koneksi antara 2 peer, tetapi begitu sudah terkoneksi,komunikasi terjadi di antara 2 peer tanpa memerlukan koneksi dengan *server* index.

Aplikasi peer-to-peer dapat dijalankan di atas jaringan peer-to-peer, *client*/*server* dan juga di internet.



Gambar 22. Klien dan Server dalam komunikasi yang sama

3.3 PROTOKOL LAPISAN APLIKASI DAN SERVIS-SERVISNYA

3.3.1 Servis dan Protokol Lapisan Aplikasi

Setelah kita lebih paham bagaimana aplikasi-aplikasi memberikan interface kepada *user* dan akses terhadap jaringan, kita akan melihat beberapa protokol umum yang lebih spesifik.

Seperti yang kita ketahui, Lapisan Transport menggunakan skemapengalamatan yaitu nomor port. Nomor port mengidentifikasikan aplikasi-aplikasi dan servis-servis lapisan Aplikasi yang menjadi tujuan maupun sumber dari data. Program-program untuk *server* biasanya menggunakan nomor port yang telah ditentukan sebelumnya yang secara umum telah diketahui oleh *client.* Saat kita melihat protokol dan servis lapisan Aplikasi yang berbeda, kita mengacu pada nomor port TCP dan UDP yang terasosiasi dengan servis-servis dan protokol-protokol tersebut. Beberapa servis/protokol tersebut antara lain :

- Domain Name System (DNS) – TCP/UDP port 53
- Hypertext Transfer Protocol (HTTP) - TCP Port 80
- Simple Mail Transfer Protocol (SMTP) - TCP Port 25
- Post Office Protocol (POP) - UDP Port 110
- Telnet - TCP Port 23
- Dynamic Host Configuration Protocol - UDP Port 67
- File Transfer Protocol (FTP) - TCP Ports 20 and 21

3.3.2 DNS

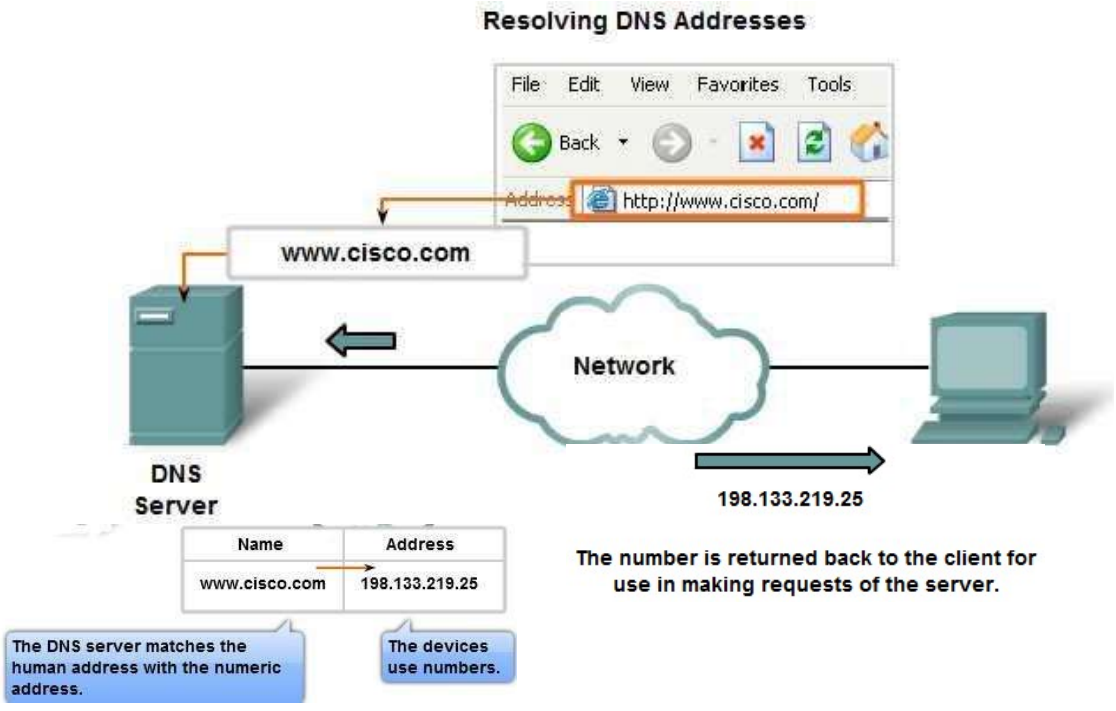
Dalam jaringan data, perangkat-perangkat diberi label dengan alamat- alamat IP numerik, sehingga mereka dapat berpartisipasi dalam mengirimkan dan menerima data dalam jaringan komputer. Bagaimanapun juga, kebanyakan orang kesulitan mengingat alamat numerik tersebut. Maka, nama-nama domain diciptakan untuk mewakili alamat numerik sehingga menjadi suatu nama yang simple dan mudah dikenali.

Pada internet, nama-nama domain ini, seperti www.cisco.com, lebih mudah untuk diingat orang daripada angka 198.132.219.25, yang mana merupakan alamat numerik yang sebenarnya dari *server* Ccisco. Selebihnya, jika Cisco memutuskan untuk mengganti alamat numerik tersebut, perubahan itu tidak akan diketahui oleh *user*, karena nama domainnya tetap

www.cisco.com. Alamat baru ini akan dihubungkan dengan nama domain yang telah ada dan konektivitas akan tetap terjaga. Saat jaringan komputer masih sedikit, mengelola pemetaan antara nama domain dengan alamat IP yang sesungguhnya adalah hal yang cukup mudah. Tetapi, seiring dengan tumbuhnya jaringan komputer dan meningkatnya jumlah perangkat jaringan yang terkoneksi, sistem manual untuk pemetaan tersebut jadi tidak bisa dilakukan.

DNS diciptakan untuk meresolusikan nama domain menjadi alamat IP. DNS menggunakan sejumlah *server* yang terdistribusi untuk menerjemahkan nama-nama domain dengan alamat-alamat numerik IP.

Protokol DNS mendefinisikan servis otomatis yang mencocokkan nama suatu sumberdaya jaringan (domain name) dengan alamat numerik jaringan. Di dalamnya termasuk format-format untuk query, respon, dan format data.Komunikasi protokol DNS menggunakan format tunggal yang disebut message. Format dari message ini digunakan untuk semua tipe query dari *client* dan respon dari *server*, error message, dan transfer informasi suatu sumber daya antara *server-server* DNS lainnya.



Gambar 23. Permintaan Alamat ke DNS

DNS adalah servis bertipe *client*/*server*; tetapi, berbeda dengan servis- servis *client*/*server* lainnya yang kita bahas. Jika servis-servis lain menggunakan *client* yang berupa aplikasi (seperti *web browser*, email *client*), *client* DNS berjalan seperti sebuah servis sendiri, bukan dalam bentuk aplikasi. *Client* DNS, kadang juga disebut sebagai DNS resolver, menyediakan resolusi nama untuk aplikasi-aplikasi jaringan dan servis-servis yang membutuhkannya.

Saat mengkonfigurasi sebuah perangkat jaringan, kita menyediakan satu atau lebih alamat-alamat *server* DNS yang bisa digunakan oleh DNS *client* untuk meresolusi nama domain. Biasanya ISP (internet *service provider*) menyediakan alamat-alamat dari *server* DNS. Jika sebuah aplikasi *user* meminta koneksi ke sebuah nama dari perangkat jaringan, *client* DNS pada *host user* meminta kepada *server* DNS yang ada untuk merubah nama perangkat menjadi alamat numeriknya.

Sistem operasi memiliki utilitas yang dinamakan `nslookup` yang mengijinkan seorang *user* untuk secara manual meminta *server* DNS untuk merubah nama *host* yang diberikan menjadi alamat numerik. Pada contoh digambar, dapat kita lihat bahwa DNS *server* yang digunakan adalah UnKnown dengan alamat 10.0.2.1. Lalu diketikkan nama domain dari suatu *host* yang akan resolusikan, yaitu www.google.co.id. *Server* DNS merespon dengan memberitahukan kepada kita bahwa nama tersebut memiliki nama lain yaitu **www-cctld.l.google.com** dengan beberapa alamat numerik 173.194.38.152, 173.194.38.159 dan 173.194.38.151.

Gambar 24. Perintah nslookup

3.3.3 Servis WWW dan HTTP

Saat alamat *web* (URL) diketikkan dalam *web-browser*, *web-browser* melakukan koneksi ke servis *web* yang berjalan di *server* menggunakan protokol HTTP. URL (*Uniform Resource Locator*) dan URI *(Uniform Resource Identifier)* adalah nama-nama yang paling umum diketahui orang yang terasosiasi dengan alamat-alamat *web*.

Alamat URL http://www.cisco.com/index.html adalah contoh URL yang mengacu pada suatu sumberdaya – yaitu halaman *web* yang bernama index.html- pada sebuah *server* yang beridentitas cisco.com.

Web-*browser* adalah aplikasi *client* pada komputer kita yang digunakan untuk melakukan koneksi dengan World Wide Web dan mengakses sumberdaya-sumberdaya yang ada pada *web server*. Seperti umumnya proses-proses *server*, *web server* menjalankan servis background dan menjadikan file-file yang dimilikinya bisa digunakan/dipanggil oleh *client-client web* (*web-browser*).

Dalam rangka mengakses sebuah konten, *web client* membuat koneksi ke *server*, dan meminta sumber daya tertentu yang diinginkan. *Server* menjawab dengan mengirimkan sumberdaya tersebut, dan saat diterima *client*, *browser* menerjemahkan data dan menyajikannya kepada *user*.

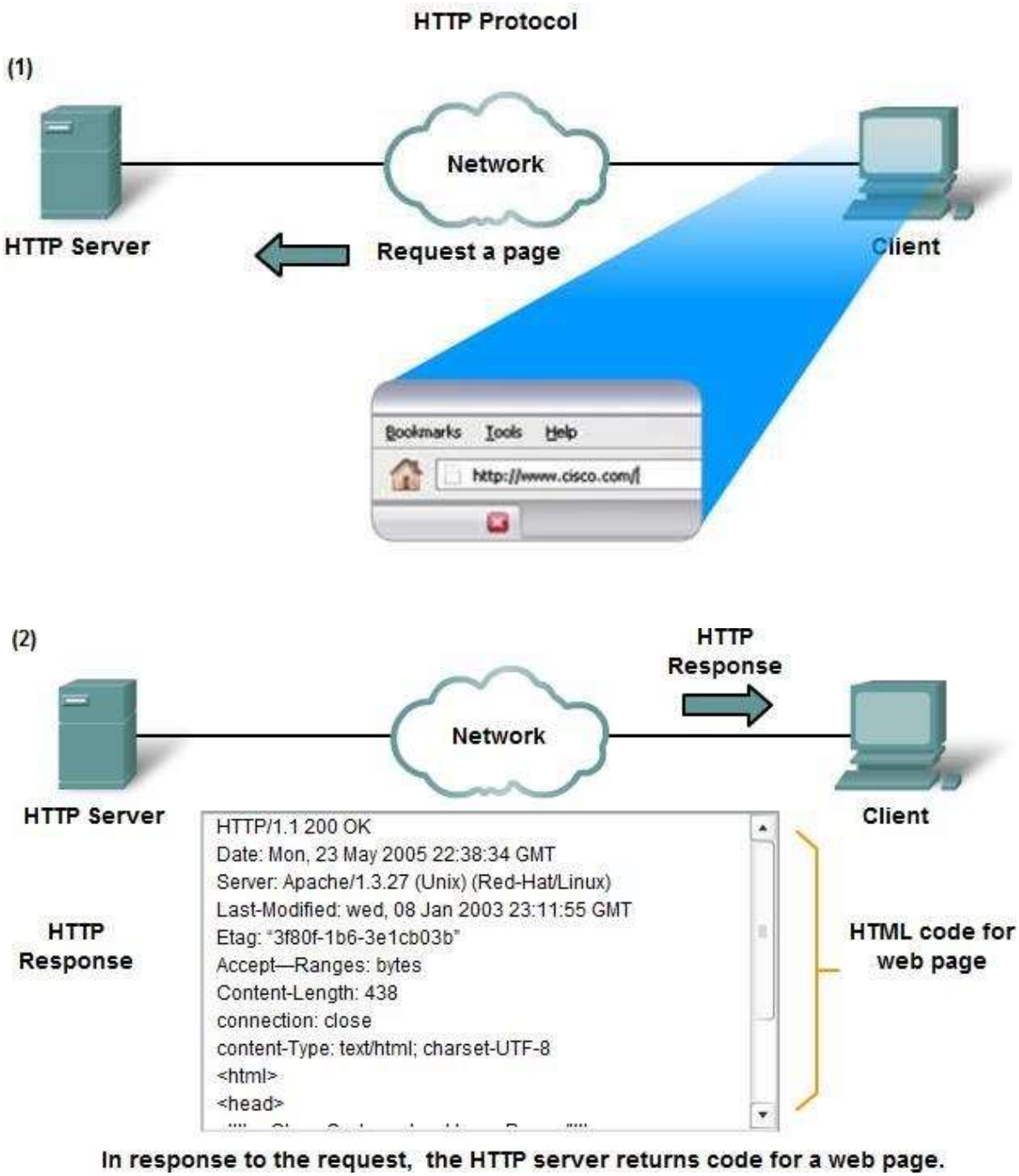
Untuk lebih jelasnya bagaimana *web browser* dan *web server* berinteraksi,kita telaah bagaimana sebuah halaman *web* bisa terbuka pada *browser*. Sebagai contoh, kita gunakan URL yaitu http://www.cisco.com/*web-server*.htm

Pertama, *browser* menerjemahkan tiga bagian dari URL tersebut :

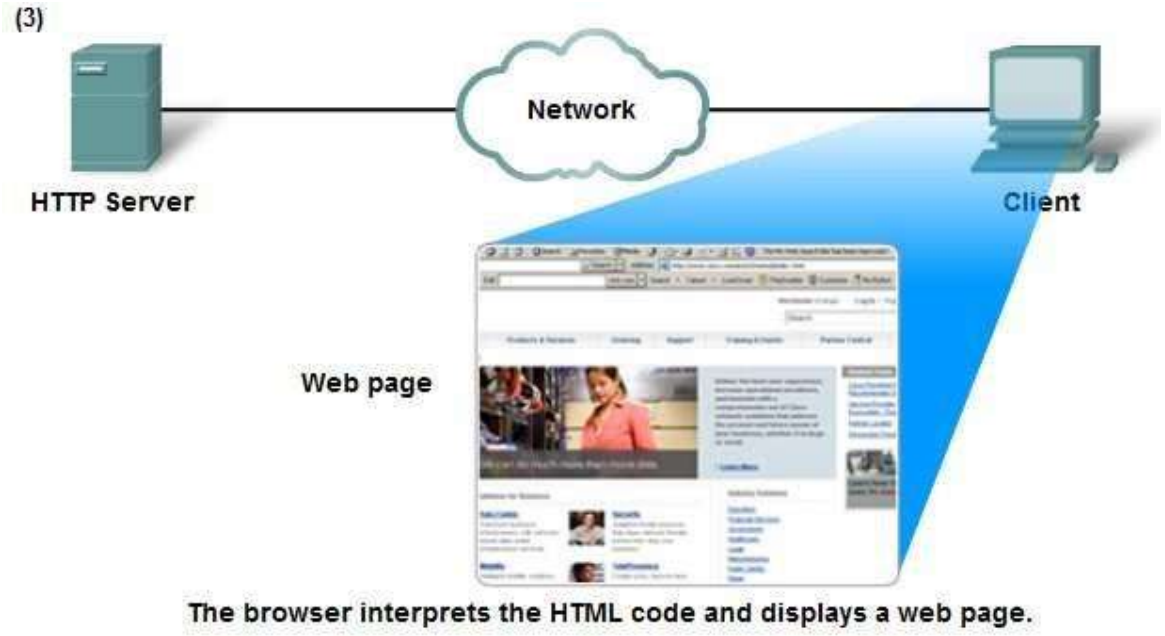
- http (protokol atau skema yang digunakan)
- www.cisco.com (nama dari *server* yang dimaksud)
- *web-server.*htm (file spesifik yang dimintakan ke *server* untukditampilkan)

Browser lalu mengecek ke *server* DNS (name *server*) untuk mengkonversi www.cisco.com menjadi alamat numerik, yang mana digunakannya untuk melakukan koneksi

ke *server* cisco tersebut. Menggunakan ketentuan protokol HTTP, *browser* mengirmkan permintaan GET ke *server* dan meminta file *web- server.*htm. *Server* mengirimkan kode HTML untuk halamat *web* itu kepada *browser*, kemudian *browser* menerjemahkan kode HTML dan memformat halamat tersebut untuk ditampilkan pada jendela *browser*.



In response to the request, the HTTP server returns code for a web page.



Gambar 25. Cara kerja Protokol HTTP

HTTP merupakan salah satu protokol dalam rangkaian protokol TCP/IP, yang dibuat untuk mempublikasikan dan mengambil halaman-halaman HTML. HTTP menggunakan protokol dengan tipe *request/response*. Saat *client*, biasanya *web browser*, mengirim pesan request ke *server*, protokol HTTP menentukan tipe dari pesan yang digunakan *client* untuk meminta halaman *web* dan juga tipe pesan yang digunakan *server* untuk merespon. Ada 3 tipe pesan yaitu GET, POST dan PUT.

GET adalah request *client* untuk meminta data. Apabila *server* menerima pesan GET, dia akan merespon dengan pesan berisi status request *client* (seperti status OK atau tidak), lalu diikuti dengan pesan *server* sendiri yang isinya bisa berisi file yang diminta *client*, ataukah pesan error, atau informasi lainnya.

POST dan **PUT** digunakan untuk mengirim pesan-pesan yang meng*upload* data ke *web-server.* Contohnya bila seorang *user* memasukkan data dalam sebuah form yang ada pada halaman *web*, POST akan memasukkan data tersebut dalam pesan yang akan dikirim ke *server.*

PUT digunakan meng*upload* suatu sumber daya (contohnya file) atau konten ke *server web*.

Walaupun HTTP terlihat fleksibel dan digunakan secara luas pada internet,protokol ini bukanlah protokol yang aman. Pesan POST melakukan *upload* informasi ke *server* dalam bentuk text sederhana yang bisa saja disadap dan dibaca secara langsung. Sama halnya dengan respon dari *server*, khususnya halaman-halaman HTML, juga tidak terenkripsi.

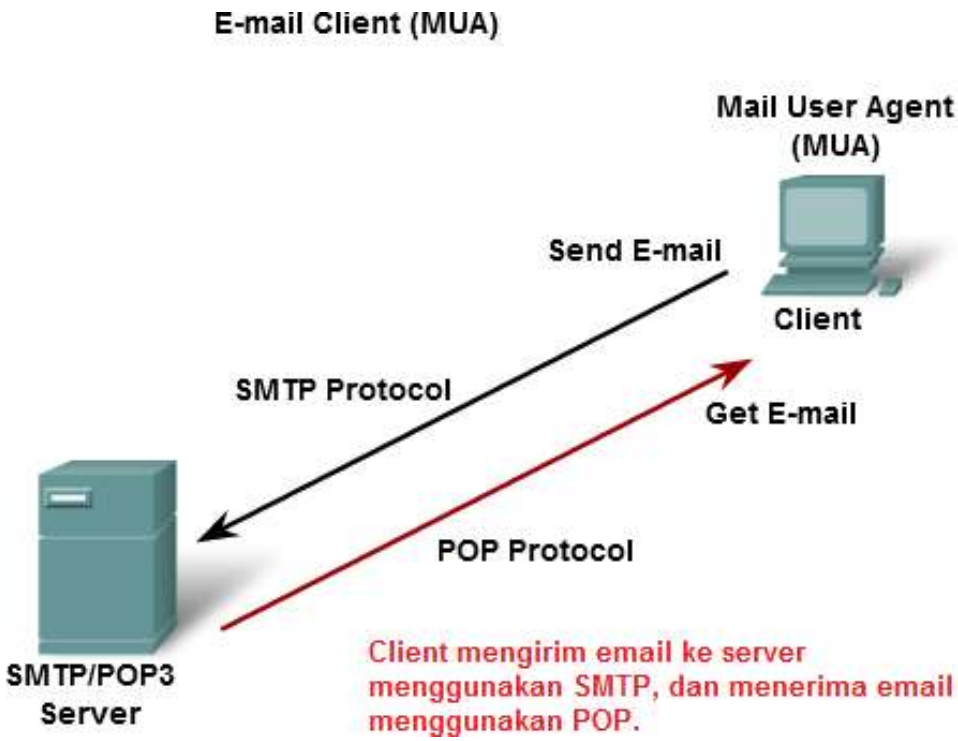
Untuk komunikasi yang aman pada internet, protokol HTTP yang lebih aman (HTTPS) digunakan untuk mengakses maupun memposting informasi *web server.* HTTPS menggunakan autentikasi dan enkripsi untuk mengamankan data selama perjalanan antara *client* dan *server.* HTTPS menambahkan aturan-aturan keamanan dalam melewatkan data antara lapisan Aplikasi ke lapisan Transport.

3.3.4 Servis Email dan protokol SMTP/POP

Email, servis jaringan yang cukup populer, telah melakukan revolusi pada manusia dalam berkomunikasi karena kemudahannya dan kecepatannya. Namun untuk menjalankannya pada komputer atau perangkat jaringan lain, email membutuhkan beberapa aplikasi dan servis. Dua contoh protokol lapisan aplikasi tersebut adalah ***Post Office Protocol (POP)*** dan ***Simple Mail Transfer Protocol (SMTP).*** Sama juga dengan protokol HTTP, protokol-protokol ini bekerja dengan proses-proses *client*/*server.*

Saat orang menulis pesan email, mereka menggunakan aplikasi yang disebut dengan *Mail User Agent (MUA)*, atau email *client.* MUA bertugas mengirimkan pesan email dan menempatkan pesan kedalam mailbox dari *client*, yang masing-masing merupakan proses yang berbeda.

Untuk menerima pesan email dari *server* email, email *client* menggunakanprotokol POP. Mengirim email, dari *client* maupun *server* email, digunakan protokol SMTP. Biasanya email *client* menyediakan fungsionalitas keduaprotokol tersebut dalam sebuah aplikasi email *client.*



Gambar 26. Protokol Email

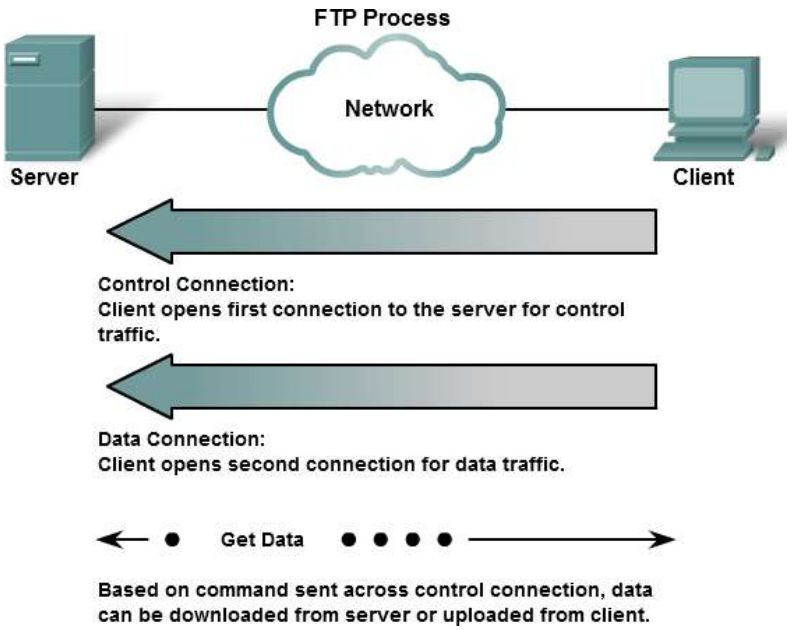
3.3.5 FTP

Protokol FTP (*File Transfer Protocol*) adalah protokol lain dari lapisan Aplikasi yang umum digunakan. FTP dikembangkan untuk melakukan transferfile antara *client* dan *server. Client* FTP adalah aplikasi yang berjalan di komputer yang digunakan untuk menaruh dan mengambil file-file dari *server* yang menjalankan FTP daemon (FTPd).

Supaya transfer file berhasil, FTP membutuhkan 2 koneksi antara *client* dan *server* : satu untuk melewatkan command-command dan responnya, satu lagi untuk lewatnya transfer file yang sesungguhnya.

Client membuat koneksi pertama ke *server* pada port TCP nomor 21. Koneksi ini digunakan untuk mengontrol lalu lintas data, berisi perintah-perintah *client* dan jawaban dari *server.*

Client membentuk juga koneksi yang kedua pada *server* dengan nomor port TCP 20. Koneksi ini dibuat mengirimkan file dan dibuat setiap kali ada file yang ditransfer. Transfer file dapat berjalan dua arah. *Client* bisa men*dowload* filedari *server*, bisa juga meng*upload* file ke *server.*



Gambar 27. Proses FTP

3.3.6 DHCP

Servis DHCP (*Dynamic Host Confirmation Protocol*) memberikan layanan kepada perangkat jaringan untuk mendapatkan alamat IP dan informasi-informasi lain dari *server* DHCP. Servis ini secara otomatis memberikan alamat-alamat IP, *subnet mask*, alamat *gateway* dan parameter-parameter jaringan IP lain.

DHCP membuat sebuah *host* mendapatkan alamat IP secara dinamis (tidakstatis) saat dia terhubung ke dalam jaringan. Caranya dengan mengontak DHCP *server* dan meminta sebuah alamat. *Server* DHCP memilih sebuah alamat dari suatu *range* alamat yang disebut dengan *pool*, dan memberikannya kepada *host* untuk suatu periode tertentu.

Dalam jaringan lokal yang besar, atau dimana populasi *user*nya silih berganti dengan cepat, penggunaan DHCP *server* sangat direkomendasikan. Untuk mengakomodasi banyaknya

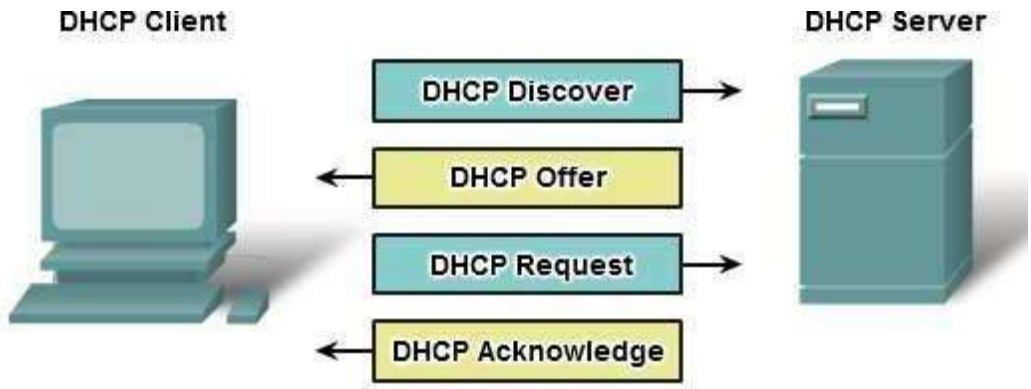
user yang baru dengan membawa laptop dan membutuhkan koneksi jaringan. Yang lain mungkin memiliki perangkat jaringan baru yang juga butuh jaringan. Daripada menugaskan administrator jaringan untuk memberikan alamat IP untuk setiap perangkat, lebih efisien apabila pengalokasian alamat IP diberikan secara otomatis oleh DHCP.

DHCP mendistribusikan alamat-alamat yang tidak permanen kepada para *host*, yang berlaku untuk waktu periode tertentu. Apabila suatu *host* telah dimatikan atau pergi dari jaringan, alamat itu akan dikembalikan ke *pool* untuk bisa digunakan kembali. Hal ini sangat membantu para *user* yang mobilitasnya tinggi dari suatu lokasi ke lokasi lain dan butuh koneksi. Host bisa memperoleh alamat IP apabila koneksi secara *hardware* telah terbentuk, baik via kabel maupun *wireless*.

DHCP memungkinkan kita mengakses internet menggunakan hotspot di tempat umum. Saat kita memasuki area, DHCP *client* di laptop kita akan mengkontak *server* DHCP lokal melalui koneksi *wireless*. Lalu *server* DHCP memberikan alamat IP pada laptop kita.

DHCP dapat menimbulkan resiko keamanan, karena setiap perangkat yangterkoneksi ke jaringan bisa mendapatkan alamat jaringan. Resiko ini membuat keamanan fisik menjadi faktor penting saat menentukan apakah sebaiknya menggunakan pengalamatan dinamis ataukan secara manual.

Pengalamatan dinamis dan statis masing-masing memiliki peranan tersendiri dalam mendesain suatu jaringan. Banyak jaringan menggunakan kedua- duanya. DHCP digunakan untuk *host-host* umum seperti perangkat-perangkat dari *user*, dan alamat tetap / statis digunakan untuk perangkat jaringan seperti g*ateway*, access point (AP), *switch*, *server*, *router*, dan mungkin saja printer dan kamera.



Gambar 27. Komunikasi DHCP

Tanpa DHCP, *user* harus menginputkan alamat IP, *subnet mask* dan seting jaringan lainnya secara manual agar bisa terhubung ke dalam suatu jaringan.

Server DHCP mengelola sebuah *pool* dari alamat-alamat IP dan memberikan alamat IP ke *client* yang mengaktifkan fungsi DHCPnya. Karena alamat-alamat IP-nya dinamis, alamat-alamat yang sudah tidak dipakai lagi akan dikembalikan kepada *pool* untuk bisa dialokasikan ulang. Saat perangkat yang terkonfiguras dengan DHCP dinyalakan, atau melakukan koneksi ke jaringan, perangkat ituakan melakukan *broadcast* suatu paket DHCP DISCOVER untuk mengetahui apakah ada *server* DHCP yang tersedia dalam jaringan. Serve DHCP akan menjawab dengan paket DHCP OFFER, yang berisi alamat IP yang ditawarkan untuk dipakai oleh *client*/*host*/perangkat yang bersangkutan, juga *subnet mask*nya, informasi *server* DNS, dan default *gateway*, serta durasi masa pakai alamat IP tersebut.

Sebuah *client* DHCP mungkin saja menerima beberapa paket DHCP OFFER apabila dalam jaringan lokalnya terdapat lebih dari satu *server* DHCP,jadi dia harus memilih satu *server* dari beberapa *server* DHCP, dengan mengirimkan secara *broadcast* paket DHCP REQUEST yang mengidentifikasi secara spesifik *server* yang dipilih dan alamat IP yang diambil untuk dipakai. *Client* juga bisa me-request sebuah alamat yang sebelumnya telah dialokasikan oleh *server.*

Jika diasumsikan bahwa alamat IP yang diminta oleh *client*, atau ditawarkan oleh *server*, masih tersedia, maka *server* akan mengirimkan paket DHCP ACK yang mengkonfirmasi kepada *client* bahwa proses dapat diselesaikan. Apabila IP yang ditawarkan sudah tidak valid – mungkin karena telah time-out atau *client* yang lain telah menggunakannya – maka *server* akan merespon dengan mengirim pesan DHCP NAK (Negative Acknowledgement). Jika paket DHCP NAK yang diterima, maka proses akan dimulai lagi dari awal dengan mengirimkan paket DHCP DISCOVER yang baru.

Begitu periode penggunaan alamat IP telah habis, *client* juga harus memperbarui proses DHCP dengan mengirim paket DHCP REQUEST baru.

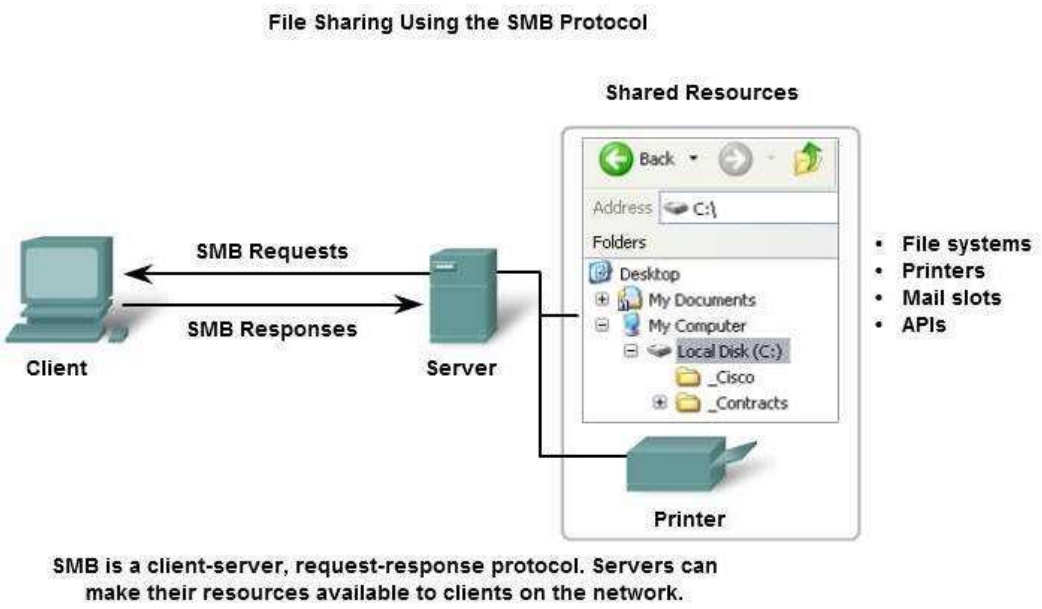
DHCP *server* memastikan bahwa alamat-alamat IP yang digunakan dalam jaringan bersifat unik (tidak ada alamat yang digunakan lebih dari satu perangkat).

3.3.7 Servis *File sharing* dan Protokol SMB

Server Message Block (SMB) adalah protokol *file sharing* yang bertipe *client/server.* IBM mengembangkan SMB di akhir tahun 1980n untuk menggambarkan struktur dari sumber daya jaringan yang bisa dipakai bersama- sama, seperti direktori, file-file, printer dan port serial. Tidak seperti *file sharing* pada FTP, disini *client* membentuk koneksi ke *server* untuk waktu yang lama. Begitu koneksi terbentuk, *user* dari *client* dapat mengakses sumber daya pada *server* seakan-akan sumber daya itu berada lokal di *host client.*

File sharing SMB dan print *service* merupakan andalan dari jaringanMicrosoft. Dari diperkenalkannya seri Windows 2000, Microsoft merubah struktur dasar sistemnya untuk menggunakan SMB. Pada versi Microsoft sebelumnya, layanan SMBnya menggunakan protokol non TCP/IP untuk melakukan resolusi penamaannya. Diawali dari Windows 2000, hingga produk- produk selanjutnya, semua produk Microsoft menggunakan penamaan DNS. Hal ini memungkinkan protokol TCP/IP untuk secara langsung menggunakan SMB.

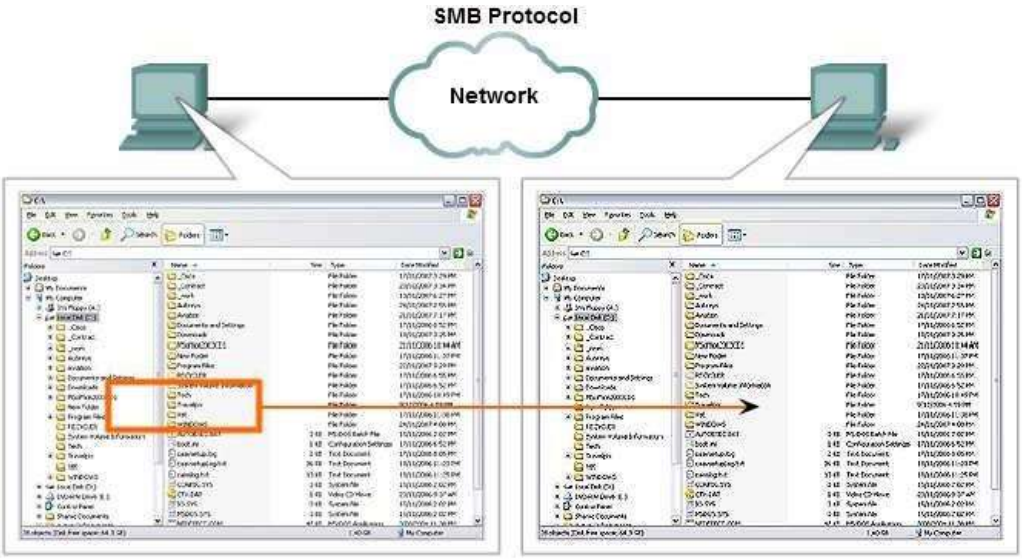
Sistem operasi Linux dan Unix juga menyediakan metode berbagi sumber daya dengan jaringan Microsoft menggunakan suatu versi dari SMB yang disebut dengan SAMBA. Sistem operasi Apple Macintosh juga bisa berbagi sumber daya dengan menggunakan protokol SMB.



Gambar 29. File Sharing menggunakan Protokol SMB

Protokol SMB menjelaskan pengaksesan sistem file dan bagaimana *client* bisa melakukan permintaan file-file tertentu. Semua pesan-pesan SMB memiliki format yang sama. Format ini menggnakan header dengan ukuran tetap diikuti dengan parameter dan komponen data yang ukurannya bervariasi.

- Fungsi dari pesan-pesan SMB :
- Memulai, melakukan autentikasi, dan mengakhiri sebuah sesi komunikasi
- Mengontrol akses file dan printer
- Memungkinkan aplikasi untuk mengirim dan menerima pesan-pesan dari dan ke perangkat lain.



Gambar 30. Mengcopy File dengan Windows Explorer menggunakan protokol SMB

3.3.8 Servis dan Protokol Telnet

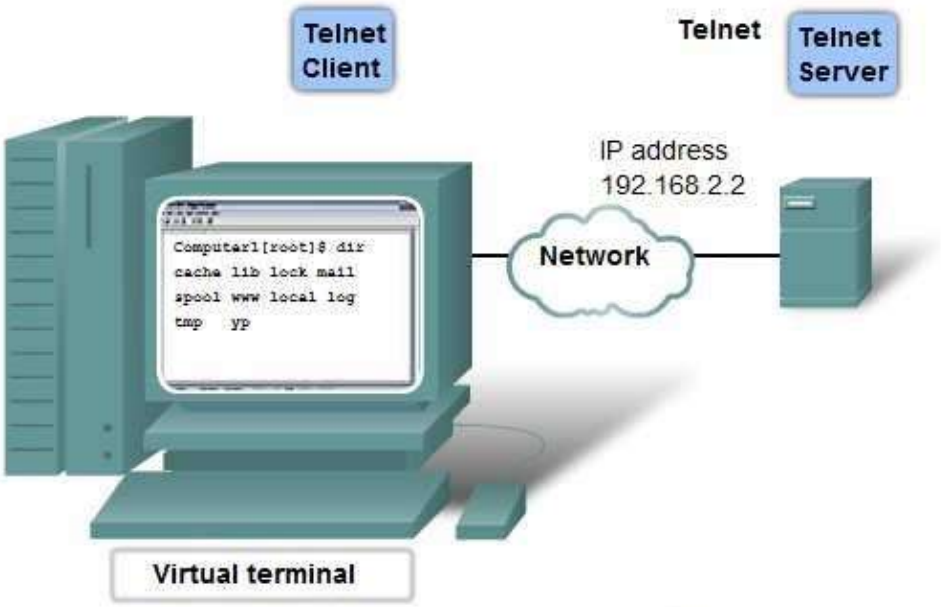
Jauh sebelum komputer desktop dengan interface grafiknya yang kompleks ada, orang menggunakan sistem berbasis teks dimana umumnya terdapat sebuah terminal display terkoneksi secara fisik ke komputer. Saat jaringan ditemukan dan tersedia, orang membutuhkan suatu cara untuk bisa mengakses dari jauh suatu sistem komputer dengan cara yang sama seperti bila ia mengakses langsung dari terminal tersebut.

Telnet dikembangkan untuk memenuhi kebutuhan tersebut. Telnet termasuk protokol Lapisan Aplikasi yang termasuk paling tua di antara rangkaian protokol TCP/IP. Telnet menyediakan metode standar untuk mengemulasikan perangkat terminal berbasis teks di atas sebuah jaringan.

Koneksi yang menggunakan servis Telnet disebut koneksi / sesi terminal virtual / Virtual Terminal (vty). Telnet menggunakan *software* untuk membuat perangkat virtual yang menyediakan fitur yang sama seperti terminal asli dengan akses ke *command line interface* (CLI) suatu *server.*

Supaya bisa melakukan koneksi Telnet dari *client*, sebuah *server*menjalankan servis yang dinamakan Telnet daemon. Koneksi virtual dibangun dari sebuah *host* yang menggunakan aplikasi Telnet *client*. Hampir semua sistem operasi memiliki Telnet *client.* Pada Ms Windows, telnet dapat dijalankan dari *command prompt*. Aplikasi terminal lain yang umum dan bisa menjalankan telnet *client* adalah HyperTerminal, Minicom dan TeraTerm.

Begitu koneksi telnet terbangun, *user* dapat melakukan fungsi-fungsi pada*server*, seperti apabila mereka menggunakan *command line interface* dari *server* itu sendiri. Jika sudah terotorisasi, mereka dapat memulai / menghentikan proses- proses *server*, mengkonfigurasi perangkat, dan bahkan mematikan sistem pada *server* tersebut.



Telnet memberikan suatu cara bagi komputer yang terkoneksi dengan jaringan, untuk mengakses suatu host dalam jaringannya seakan-akan keyboard dan monitornya terhubung langsung dengan host tersebut.

Gambar 31. Telnet

Telnet merupakan protokol *client*/*server* yang menetapkan bagaimana suatu sesi vty dibuat dan diakhiri. Protokol ini juga menyediakan *syntax* dan rangkaian perintah yang digunakan untuk memulai sesi telnet, dan juga perintah- perintah kontrol yang bisa digunakan selama sesi telnet tersebut. Walaupun protokol Telnet menyediakan autentikasi *user*, tetapi dia tidak menyediakan enkripsi data yang ditransfer. Semua pertukaran data selama sesi Telnet

ditransfer dalam bentuk teks biasa sepanjang jaringan. Ini berarti data bisa saja disadap dan bisa langsung dibaca.

Jika keamanan diperlukan, maka ada metode alternatif dan aman untuk mengakses suatu *server* yaitu protokol *Secure Shell* (SSH). SSH menyediakan struktur untuk mengakses secara *remote* melalui login dan servis-servis keamananjaringan lainnya. SSH juga menggunakan proses autentikasi yang lebih kuat daripada Telnet dan mendukung enkripsi data selama sesi transport data dalam jaringan.