

## **BAB IV**

### **PROTOKOL APLIKASI**

#### **4.1 DOMAIN NAME SYSTEM**

Pengalamatan di protokol Internet yang menggunakan kombinasi angka cukup sulit untuk diingat. Penggunaan alamat dengan nama akan lebih mudah diingat. Untuk itu diperlukan suatu pemetaan alamat IP ke nama host atau end system yang ada dan sebaliknya.

Pada awalnya digunakan teknik yang dinamakan *host table*. Masing-masing host/komputer menyimpan daftar kombinasi nama komputer dan alamat IP, pada file yang dinamakan HOSTS.TXT. File ini berisi nama dan alamat IP seluruh komputer yang terkoneksi ke internet. File ini pula yang setiap kali diperbarui melalui FTP ke seluruh host di Internet, jika terdapat penambahan host baru.

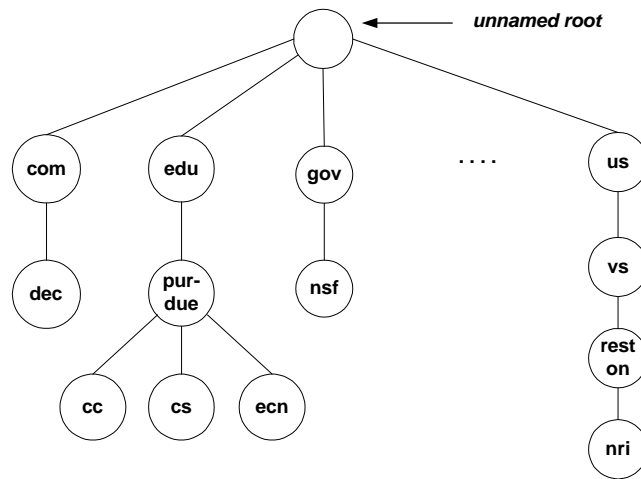
Kemudian dilakukan pendistribusian basis data *hostname* dan *IP address* ini. Dengan pendistribusian ini, masing-masing organisasi hanya bertanggung jawab terhadap basis data yang berisi informasi jaringan miliknya saja. Karena sifat basis data yang terdistribusi, maka harus ada suatu mekanisme bagi host lain untuk bisa menemukan host yang tepat, yang menyimpan data yang dibutuhkan.

Tahun 1984, Paul Mockapetris mengusulkan sistem basis data terdistribusi yang dinamakan DNS (*Domain Name System*). Sistem inilah yang digunakan hingga sekarang.

Selain untuk memetakan alamat IP dan nama host, DNS juga digunakan sebagai sarana bantu penyampaian *e-mail* (*e-mail routing*).

##### **4.1.1 Top Level Domain dan Pendelegasian**

Format penamaan host di Internet dibuat memiliki hirarki. Skema hirarki tersebut digambarkan berbentuk *tree*. Satu node/titik membentuk tree, memiliki beberapa *subnode*. Subnode ini membentuk tree yang memiliki beberapa subnode lagi, dan seterusnya. Pada masing-masing node ini terdapat label. Node berlabel ini disebut *domain*. Domain ini bisa berupa nama host, subdomain atau top level domain.



gambar model Tree domain

*Domain* teratas ialah **Root Domain**. Domain ini dituliskan dalam bentuk titik (“.”). **Top Level Domain** terdiri atas semua node yang tepat berada di bawah root. Pada gambar di atas hal ini ditunjukkan dengan node **com**, **edu**, **gov** dan seterusnya.

*Subdomain* merupakan kumpulan keturunan *Top Level Domain*. Node yang berada tepat di bawah *Top Level Domain* disebut *Second Level Domain*. Node di bawah *Second Level Domain* disebut *Third Level Domain* dan seterusnya.

Cara pembentukan serta pembacaan nama host dan domain, sesuai dengan diagram di atas, dimulai node paling bawah, mengikuti label yang tertera pada masing-masing node dan berakhir di root.

Sebagai contoh : **xinu.cs.purdue.edu.**

tanda “.” Menunjukkan **Root Domain**

**Edu** merupakan **Top Level Domain**

**Purdue** merupakan domain level dua ( *second level domain* )

**Cs** merupakan domain level tiga ( *third level domain* )

**Xinu** merupakan nama **host/komputer** yang bersangkutan.

Sesuai dengan konvensi, label yang menunjukkan domain ditulis dari kiri ke kanan, dipisahkan dengan tanda titik, dengan domain yang paling jauh dari root ditulis terlebih dahulu. Penulisan secara lengkap seperti di atas mulai dari nama host hingga

tanda titik yang melambangkan root disebut sebagai *Fully Qualified Domain Name (FQDN)*.

Top level domain digunakan untuk menunjukkan jenis perusahaan, instansi, lembaga atau negara tempat komputer ini berada. *Top Level Domain (TLD)* ini dapat dibagi menjadi 3 jenis yaitu :

- TLD generik (*generic domain*)
- TLD negara (*country domain*)
- TLD *arpa*

Pada mulanya TLD yang dipakai ialah TLD generik. TLD *generic* ini terdiri atas tujuh jenis domain yang terdiri atas tiga huruf. Domain **com** digunakan oleh *organisasi bersifat komersial* (ibm.com, microsoft.com). Domain **edu** (Berkeley.edu, purdue.edu, mit.edu) digunakan untuk *lembaga pendidikan* (universitas). Domain **gov**, digunakan untuk *lembaga pemerintahan* (whitehouse.gov, odci.gov). domain **int** digunakan oleh *organisasi internasional* (nato.int). domain **mil** digunakan oleh badan *kemiliteran Amerika Serikat* (army.mil, navy.mil, af.mil). Domain **net** digunakan oleh *penyedia jaringan Internet* (ibm.net, mci.net). Sedangkan domain **org** digunakan oleh *organisasi nonkomersial* (greenpeace.org). hingga saat ini sistem pembagian organisasional ini masih berlaku di Amerika.

Berdasarkan informasi ini kita dapat menyimpulkan bahwa komputer xinu.cs.purdue.edu adalah komputer milik suatu lembaga pendidikan.

Dengan semakin banyaknya negara-negara yang terhubung ke internet, kemudian diputuskan untuk menggunakan standar pembagian geografis yang ditetapkan sesuai standar **ISO 3166**. inilah yang disebut TLD Negara. Berdasarkan konvensi tersebut dialokasikan TLD yang merupakan pengenalan geografis (negara) dan terdiri atas dua huruf yang unik

Sebagai contoh negara Indonesia memiliki TLD **.id**, negara Inggris memiliki TLD **.uk** (united kingdom), negara Malaysia memiliki TLD **.my** dan sebagainya.

Pada umumnya pembagian TLD secara geografis ini kemudian diikuti dengan pembagian berdasarkan afiliasi organisasi bagi level domain di bawahnya. Ada yang mengadopsi sistem pembagian di Amerika, seperti edu.au atau com.au, ada juga yang mengikuti

sistem pembagian yang dipelopori Inggris, seperti co.uk (corporation) atau ac.uk (academic)

Konvensi pembagian nama domain di Indonesia ditetapkan sampai level kedua. Sedangkan aturan penamaan domainnya mengikuti sistem Inggris. Hal ini dapat ditunjukkan pada tabel berikut :

<b>Domain</b>	<b>Keterangan</b>
go.id	Subdomain untuk lembaga pemerintah
co.id	Subdomain untuk lembaga konvensional
ac.id	Institusi akademik
net.id	Penyedia jasa network
or.id	LSM dan lembaga non komersial

Untuk penamaan third level domain dan seterusnya, hal ini diserahkan pada pengelola jaringan yang bersangkutan. Misalnya pengelola jaringan di kampus kita digunakan sttelkom.ac.id.

Selain memetakan nama host ke alamat IP, DNS juga memiliki fasilitas *reverse mapping*. Fasilitas ini memetakan alamat IP ke domain-name. Reverse mapping juga digunakan untuk menghasilkan keluaran yang lebih manusiawi, mudah dibaca dan diinterpretasikan, misalnya untuk pembacaan log-file.

Saat suatu organisasi bergabung ke Internet dan mendapatkan otoritas untuk nama domain tertentu, dia juga mendapat otoritas untuk nama space in-addr.arpa, yang sesuai dengan IP address yang dimilikinya.

Keseluruhan domain name space di atas, baik yang biasa maupun yang reverse mapping, sebagaimana digambarkan di gambar.... Dan ..... tidak dikelola oleh satu server. Pengelolaannya dilakukan secara terdistribusi. Untuk itu, domain name space di atas dibagi dalam zone-zone. Sebuah zone meliputi seluruh host di bawah domain tertentu, kecuali yang didelegasikan ke zona lain.

Zona ini bisa berupa level kedua domain, level ketiga domain, level keempat domain dan seterusnya. Sebuah domain bisa membagi zonanya menjadi zona-zona yang lebih kecil.

Saat administrator jaringan memutuskan untuk membentuk zona baru, maka harus disediakan Name Server untuk zona tersebut. Nama dan alamat IP dari seluruh komputer

di zona ini harus diisikan ke Name Server tersebut. Name Server inilah nanti yang akan menjawab setiap pertanyaan tentang zona yang bersangkutan.

Jika jumlah komputer di zona yang bersangkutan sedikit, maka tugas administrator DNS menjadi ringan. Jika sebaliknya, maka sebaiknya dilakukan pembentukan zona baru serta pendelegasian domain.

Jika pada suatu domain ingin dibentuk zona baru, ditugaskan sebuah primary Name Server dan satu atau lebih Secondary Name Server untuk menangani zona ini. Dengan demikian, Server DNS induk telah memberikan authority pada name server yang bersangkutan untuk menangani zona tersebut. Kedua jenis server ini pun menjadi Name Server yang authoritative untuk zona baru tersebut.

## **4.2 FILE TRANSFER PROTOCOL**

FTP atau File Transfer Protocol merupakan salah satu aplikasi TCP/IP yang banyak digunakan untuk memindahkan atau menyalin file dari komputer satu ke komputer lainnya. Aplikasi ini adalah aplikasi yang telah dikembangkan sejak awal perkembangan Internet. Hal ini terlihat dari mulai didefinisikannya protocol ini sejak Internet menggunakan RFC sebagai alat standarisasi. Kata FTP sendiri telah muncul di RFC 172 yang diterbitkan tahun 1971.

Operasi protocol FTP ini cukup sederhana. Dengan menggunakan client FTP, seorang pengguna dapat melihat isi direktori, memindahkan file dari dan ke server FTP, serta membuat dan menghapus direktori di server tersebut. Dalam melakukan operasi yang berhubungan dengan pengiriman isi file, FTP menggunakan koneksi TCP tambahan yang khusus untuk mengirim isi file. Sekarang kita akan melihat secara sederhana proses yang terjadi di dalam FTP sewaktu kita melakukan transfer file.

### **4.2.1 Model Protokol FTP**

FTP menggunakan dua jenis hubungan (*connection*) untuk mentransfer sebuah file, yaitu :

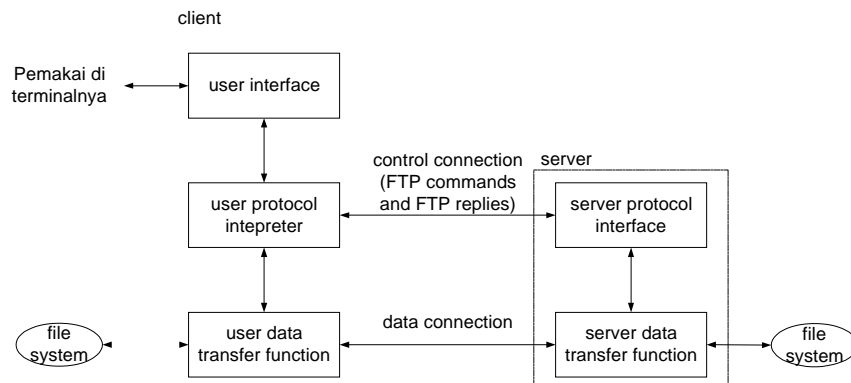
- *Control Connection*; yang digunakan pada pola hubungan antara *client* – *server* yang normal. Server membuka diri secara pasif di sebuah port khusus selanjutnya

server menunggu hubungan yang akan dilakukan oleh *client*. *Client* segera aktif membuka port tersebut untuk membangun *control connection*.

*Control connection* ini akan dipertahankan sepanjang waktu selama *client* masih berkomunikasi dengan server. Hubungan ini digunakan oleh *client* untuk mengirim perintah-perintah ke *server*, dan *server* menggunakannya untuk memberi respon.

- *Data Connection* yang dibangun setiap kali sebuah file ditransfer antara *client-server*. Hubungan ini bersifat “memaksimalkan ukuran data yang ditransfer (*throughput*)”, karena hubungan ini untuk transfer file.

Gambar berikut ini memperlihatkan susunan *client* dan *server* serta dua hubungan diantara mereka.



gambar model sebuah hubungan FTP

pada model di atas, pemakai di terminalnya melakukan aktivitas FTP, melalui user interface baik yang berupa program windows FTP, atau yang *command line*. *User protocol interpreter* selanjutnya yang akan melakukan hubungan *control connection* ke server. Perintah-perintah FTP yang standard dikeluarkan oleh interpreter ini kepada server melalui hubungan *control connection*.

Gambar tersebut juga memperlihatkan ada dua interpreter protocol yang menangani kedua fungsi transfer data ketika dibutuhkan.

#### 4.2.2 Fasilitas – Fasilitas FTP

- *Interactive Access*

Disediakan fasilitas interface interaksi antara client dengan server.

- *Format Specification*

Client diperbolehkan menentukan tipe & format data.

- *Authentication Control*

Ada kontrol autentifikasi untuk client yang meminta pengiriman file dari server berupa masukkan login dan password dari client.

#### 4.2.3 Representasi Data

Transfer file hanya dilakukan melalui *data connection*. Sedangkan *control connection* digunakan untuk mentransfer perintah-perintah dan balasan. Bagaimana file ditransfer dan disimpan telah disebutkan dalam spesifikasi protocol FTP dan ada banyak pilihan cara.

#### 4.2.4 Perintah-Perintah FTP

Perintah-perintah FTP merupakan karakter ASCII sebanyak 3 sampai 4 byte, dan menggunakan huruf besar. Keseluruhan ada 30 perintah dan beberapa perintah yang umum digunakan diperlihatkan oleh Tabel berikut

Perintah	Keterangan
<b>ABORT</b>	Hentikan ( <i>abort</i> ) perintah FTP dan transfer data sebelumnya
<b>LIST</b> filelist	Sebutkan daftar file atau direktori
<b>PASS</b> password	Password di server
<b>QUIT</b>	Keluar dari server
<b>RETR</b> namafile	Ambil sebuah file
<b>STOR</b> namafile	Kirim sebuah file

<b>SYST</b>	Tipe sistem dari server
<b>TYPE type</b>	Menentukan tipe file A = ASCII, I = Image
<b>USER namauser</b>	Nama pemakai di server

#### 4.2.5 REPLY FTP

Balasan (reply) FTP berupa bilangan tiga digit dalam ASCII. Software selanjutnya harus tahu bagaimana cara mengartikan balasan yang berupa bilangan tersebut. Maksud dari digit pertama dan kedua dari kode balasan ditunjukkan oleh Tabel berikut :

#### 4.2.6 Pengaturan Hubungan (Connection)

Ada tiga jenis pemakaian pada *data connection*, yaitu :

- mengirim sebuah file dari *client* ke *server*
- mengirim sebuah file dari *server* ke *client*
- mengirim sebuah daftar file atau direktori dari *server* ke *client*

prosedur normal untuk mentransfer file atau direktori adalah sebagai berikut :

1. Client mengatur pembuatan data connection
2. Client memilih sebuah nomor port di host client sebagai ujung dari data connection pada sisi client.  
Client secara pasif membuka port ini.
3. Client mengirim nomor port ini ke server melalui control connection menggunakan perintah PORT

Server menerima port tersebut dari control connection, dan mengirim balasan secara aktif ke port di host client. Nomor port untuk data connection pada sisi server selalu 20.

### 4.3 SIMPLE MAIL TRANSPORT PROTOCOL (SMTP)

Untuk pengiriman pesan, email relatif lebih mudah dan cepat dibandingkan dengan prosedur transfer file. Protocol standar untuk pengiriman e-mail adalah SMTP.

Protokol SMTP menitikberatkan khususnya pada metode sistem pengiriman pesan melalui suatu link dari suatu sumber ke tujuan. Protocol ini tidak mendefinisikan cara sistem mail menerima mail dari suatu user atau cara interface user menampilkan user

yang mendapat mail. Protocol ini juga tidak menspesifikasikan bagaimana suatu mail disimpan.

SMTP merupakan protocol sederhana dimana komunikasi antara client dengan server dilakukan dengan pertukaran text yang dapat dibaca. Tahapan-tahapan komunikasi dengan protocol SMTP :

- saat inialisasi client membangun koneksi dengan server dan menunggu server mengirim pesan 220 REASY FOR MAIL.
- Setelah menerima pesan tersebut client mengirimkan command HELLO
- Server merespon dengan mengirimkan identifikasi dirinya.
- Sekali komunikasi terbangun, sender dapat mengirim satu atau lebih pesan mail, mengakhiri hubungan, atau meminta server bertukar aturan sehingga pesan dapat mengalir ke arah sebaliknya.
- Setiap penerima harus mengirim ACK untuk setiap pesan yang diterimanya.

Transaksi mail diawali dengan command MAIL. Field yang berisi alamat untuk mengirimkan pesan kesalahan adalah FROM. Penerima menyiapkan struktur data untuk menerima pesan mail baru dan menyalinnya ke command MAIL dengan mengirim respon 250. respon 250 berarti segalanya dalam kondisi baik.

Setelah sukses menerima command MAIL, pengirim menuliskan rangkaian command RCPT yang mengidentifikasi pesan mail penerima. Penerima harus mengirim ACK untuk setiap command RCPT dengan mengirim pesan 250 OK atau dengan mengirim pesan 550 *No such user here*.

Setelah selesai dengan command RCPT, pengirim mulai dengan command DATA. Penerima akan merespon dengan pesan 354 *Start mail input* dan menentukan urutan karakter yang digunakan untuk mengakhiri pesan mail. Urutan terminasi terdiri dari 5 karakter :, *carriage return*, *line feed*, *period*, *carriage return* dan *line feed*.

Contoh :

User Smith di terminal Alpha.EDU mengirim suatu pesan ke user Jones, Green dan Brown di host Beta.GOV. Software client SMTP di host Alpha.EDU menghubungi software server SMTP di host Beta.GOV dan mulai pertukaran pesan :

```
S : 20 Beta.GOV Simple Mail Transfer Service Ready
C : HELO Alpha.EDU
S : 250 Beta.GOV

C : MAIL FROM:<Smith@Alpha.EDU>
S : 250 OK

C : RCPT TO : <Jones@Beta.GOV>
S : 250 OK

C : RCPT TO : <Green@Beta.GOV>
S : 550 No such user here

C : RCPT TO : <Brown@Beta.GOV>
S : 250 OK

C : DATA
S : 354 Start mail input ; end with <CR><LF>.<CR><LF>
C : ...sends body of mail message...
C : ...continues for as many lines as message contains
C : <CR><LF>.<CR><LF>
S : 250 OK

C : QUIT
S : 221 Beta.GOV Service closing transmission channel.
```

Pada contoh server menolak penerima Green karena tidak mengenal nama tersebut sebagai tujuan yang valid. Protocol SMTP tidak menentukan detail penanganan error oleh client.

Sekali client selesai mengirimkan semua pesan mail ke tujuan-tujuan tertentu, client dapat memberikan command TURN. Penerima kemudian dengan 250 OK.

#### **4.4 TELNET**

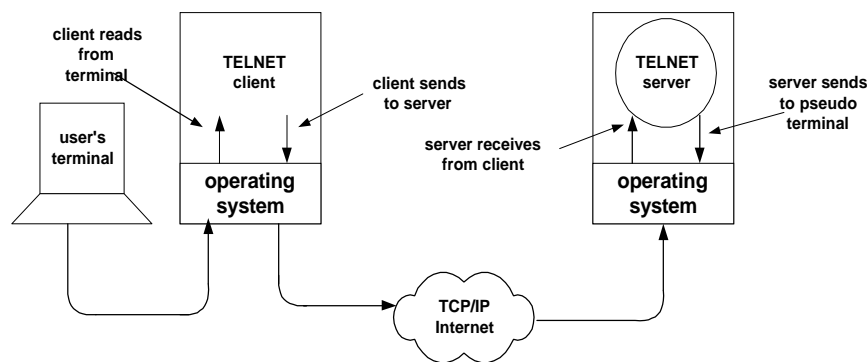
TELNET merupakan protokol akses remote terminal sederhana. Dengan TELNET suatu user dapat membangun sebuah koneksi TCP ke server di tempat yang berbeda, mengaksesnya seolah user berada pada posisi server. TELNET juga dapat membawa keluaran dari mesin yang berada pada remote ke terminal user.

Biasanya software client TELNET mengizinkan user untuk menentukan mesin yang akan diremote dengan memberikan nama domain atau alamat IP-nya.

TELNET memberikan tiga layanan dasar :

1. penentuan *network virtual terminal* yang menyediakan interface standar untuk sistem remote.
2. mekanisme yang memungkinkan client dan server bernegosiasi mengenai option dan menyediakan option standar seperti option mengenai format data yang digunakan : 7 bit karakter ASCII atau 8 bit karakter .
3. TELNET dapat treat dua koneksi secara simetris

Ilustrasi mengenai cara program aplikasi mengimplementasikan TELNET untuk client & server adalah sebagai berikut :



gambar ilustrasi aplikasi TELNET untuk client & server

pada gambar terlihat, jika suatu user ingin menjalankan aplikasi TELNET ke komputer lain, maka program aplikasi pada user tersebut berfungsi sebagai client. Client membangun koneksi TCP ke server, komputer tempat user ingin berkomunikasi. Server harus menerima koneksi TCP dari client dan kemudian me-relay data antara koneksi TCP dan sistem operasi lokal.

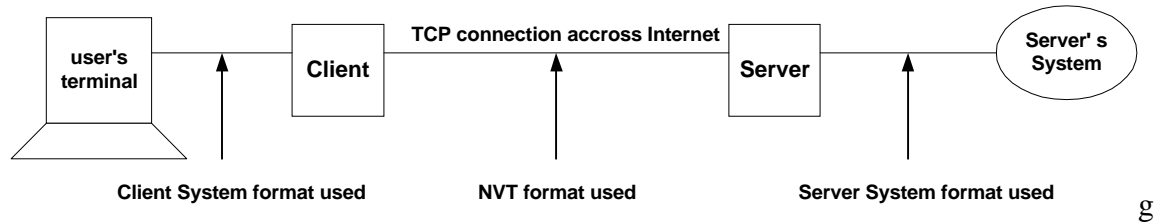
Pada kenyataannya server mengerjakan sesuatu yang lebih kompleks, karena harus menangani banyak koneksi secara konkuren. Biasanya satu proses server master akan menunggu koneksi baru dan membentuk slave baru untuk menangani setiap koneksi. Pada gambar di atas, server TELNET merepresentasikan slave yang menangani satu koneksi. Di gambar tersebut tidak memperlihatkan master server yang menangani request baru, juga tidak menunjukkan slave yang sedang menangani koneksi.

Pseudo-terminal disini digunakan untuk menerangkan entry point sistem operasi yang memungkinkan aplikasi misalnya TELNET server berjalan.

#### 4.4.1 Mengakomodasi Heterogenitas

Untuk bisa membuat aplikasi TELNET dapat berjalan diantara banyak sistem, aplikasi TELNET harus mampu mengakomodasi keanekaragaman komputer dan sistem operasi.

Untuk mengatasi keanekaragaman TELNET menentukan format data & perintah yang dikirimkan melalui Internet. Definisi ini disebut dengan *network virtual terminal (NVT)*.



ambar format NVT yang digunakan dalam TELNET

pada gambar di atas terlihat software pada client mentranslasikan rangkaian data & perintah dari terminal user ke format NVT dan mengirimnya ke server. Software server mentranslasikan rangkaian data dan perintah yang tiba dari format NVT ke format yang sesuai dengan sistem remote. Untuk pengiriman kembali data server remote akan mentranslasi dari format sistem remote ke format NVT untuk kemudian client lokal mentranslasinya kembali dalam format mesin lokal

#### 4.4.2 Melewatkan Perintah untuk Mengontrol Sistem Remote

TELNET NVT mengakomodasi fungsi kontrol dengan menentukan cara control tersebut dilewatkan dari client ke server. Secara konsep kita memandang NVT sebagai penerimaan input dari terminal yang dapat membangkitkan lebih dari 128 karakter. Kita mengasumsikan terminal user memiliki virtual key yang sesuai dengan fungsi-fungsi untuk mengontrol proses. Contoh NVT menentukan key interrupt konseptual yang merupakan request untuk mengakhiri program.

Untuk melewati fungsi-fungsi kontrol melalui koneksi TCP, TELNET mengencode fungsi-fungsi tersebut menggunakan *escape sequence*. Escape sequence menggunakan reserved oktet yang mengindikasikan ada suatu oktet kode kontrol yang sedang mengikuti. Dalam TELNET sebuah reserved oktet yang mengawali suatu escape sequence dikenal sebagai *interpret as command (IAC)*.

## 4.5 HYPERTEXT TRANSFER PROTOCOL (HTTP)

Spesifikasi protokol ini didefinisikan oleh Tim Berners-lee dalam RFC 1945 dan digunakan di Internet sejak tahun 1990. RFC 1945 mendefinisikan protokol ini untuk versi 1.0 dimana masih dianggap memiliki kekurangan sehingga dilengkapi dengan versi 1.1 yang tertuang dalam RFC 2068. Perbaikan dilakukan pada koneksi persistent dan pipelined serta model cache yang lebih baik.

### 4.5.1 Model Hubungan HTTP

Model hubungan protokol ini adalah *request-response*, yaitu client menyampaikan pesan request ke server dan server kemudian memberikan response yang sesuai dengan request tersebut. Request & response dalam HTTP disebut sebagai *respon chain & request chain*. Hubungan HTTP yang paling sederhana terdiri atas hubungan langsung antara user agent dengan server asal. Beberapa komponen yang terlibat dalam sebuah hubungan HTTP adalah : client, user agent, server asal, proxy, gateway dan tunnel. :

- ***client***  
adalah program yang membentuk hubungan HTTP dengan tujuan untuk mengirimkan request
- ***user agent***  
client yang melakukan request dapat berupa browser, editor, spider atau perangkat lain
- ***server asal***  
server tempat menyimpan atau membuat resource
- ***proxy***  
program perantara yang bertindak sebagai server dan client dengan tujuan untuk membuat request atas nama client yang lain.
- ***Gateway***  
Server yang bertindak sebagai perantara untuk server lain. Gateway menerima request seolah-olah ia adalah server asal dan client tidak mengetahui bahwa gateway yang menerima request yang akan dikirim
- ***Tunnel***

Program perantara yang bertindak sebagai perantara buta antara dua hubungan HTTP. Tunnel tidak dianggap sebagai pihak yang terlibat dalam hubungan HTTP, walaupun ia dapat membuat HTTP request.

Pada protokol HTTP terdapat 3 jenis hubungan dengan perantara : *proxy*, *gateway* dan *tunnel*. Proxy bertindak sebagai agen penerus, menerima request dalam bentuk *Uniform Resource Identifier (URI)* absolut, mengubah format request, dan mengirimkan request ke server yang ditunjukkan oleh URI. Gateway bertindak sebagai agen penerima dan menterjemahkan request ke protokol server yang dilayaninya. Tunnel bertindak sebagai titik relay antara dua hubungan HTTP tanpa mengubah request & response HTTP. Tunnel digunakan jika komunikasi perlu melalui sebuah perantara dan perantara tersebut tidak mengetahui isi dari pesan dalam hubungan tersebut.

Contoh hubungan HTTP yang melibatkan beberapa komponen dapat dilihat pada gambar berikut :



gambar komponen-komponen dalam rantai request/response HTTP

Proxy atau gateway dapat menggunakan mekanisme cache untuk memperpendek rantai hubungan HTTP. Proxy pada gambar di atas dapat menggunakan cache dan memberikan response cache sehingga request dari UA tidak perlu dilayani oleh server asal karena proxy telah memberikan response atas request tersebut.

#### 4.5.2 Format HTTP

Kita mengenal protokol HTTP menggunakan format URL (Universal Resource Locator) HTTP dalam bentuk :

[http://host\[:port\]\[abs\\_path\]](http://host[:port][abs_path])

host adalah nama domain internet yang legal

port adalah bilangan yang menunjukkan port HTTP di host

jika port tidak disebutkan maka port HTTP diasumsikan sebagai 80

abs\_path menyatakan lokasi resource di dalam host.

Contoh :

<http://www.sttelkom.ac.id/home.html>

Jika kita mengisikan URL tersebut ke browser, browser bertugas untuk mengartikan URL tersebut dan menerjemahkannya dalam komunikasi protokol HTTP. Aturan dalam mengartikan format URL HTTP mengikuti aturan umum URI, yaitu case sensitive, kecuali nama dan skema URL case insensitive.

Komunikasi protokol HTTP terdiri atas pesan request yang diberikan oleh user agent dan response yang dikeluarkan oleh server. Setiap request dan response HTTP menggunakan format pesan generic seperti yang didefinisikan oleh RFC 822.

Pesan HTTP terdiri atas **baris mulai**, **header pesan** dan **isi pesan** dipisahkan oleh sebuah baris kosong, yaitu hanya berisi karakter CRLF.

- **Baris mulai**

pada pesan request berisi pesan permintaan dari client, sementara pada pesan response, baris ini berisi status response atas request yang diterima.

- **Header pesan**

dapat terdiri atas beberapa baris, bergantung pada field-field yang perlu disertakan dalam header tersebut. Terdapat 4 jenis header pesan, yaitu **header pesan umum** yang berlaku di setiap jenis pesan, **header request**, **header response** dan **header entity**.

Header yang umum pada pesan HTTP request dan response adalah sebagai berikut :

Header-umum = Cache-  
Control|Connection|Date|Pragma|Transfer-  
Encoding|Upgrade|Via

Field **Cache-control** memberikan aturan yang harus ditaati oleh seluruh mekanisme cache dalam rantai request/response.

Field **Connection** mengatur tipe hubungan HTTP, apakah akan menggunakan hubungan persisten atau tidak.

Field **Date** memberikan informasi mengenai waktu asal pesan.

Field **Transfer-Encoding** menentukan jenis transfer yang diberikan kepada isi pesan agar dapat sampai dengan aman ke client.

Field *Upgrade* digunakan untuk mengganti protokol yang hendak digunakan.

Field *Via* digunakan oleh proxy dan gateway untuk memberitahu jalur yang digunakan dalam sebuah rantai request /response

- **Isi Pesan**

Digunakan untuk mengirimkan isi entity. Keberadaan isi pesan dalam pesan request ditandai dengan adanya header *Content-Length*. Dalam pesan response, keberadaan isi pesan ini tergantung atas kode status yang diberikan. Dalam sebuah pesan HTTP header *Content-Length* & *Transfer-Encoding* tidak boleh muncul bersama-sama. Kedua header ini menunjukkan hal yang berlawanan, adanya header *Transfer-Encoding* menunjukkan bahwa panjang isi pesan tidak diketahui sementara header *Content-Length* menunjukkan panjang isi pesan dalam byte. Jika dalam sebuah pesan terdapat kedua header ini, maka header *Content-Length* harus diabaikan.

Setiap request & response dalam komunikasi HTTP harus menyertakan versi protokol yang digunakan. Format penulisan versi protokol ini adalah :

**HTTP/<major>.<minor>**

Major dan minor adalah bilangan yang menunjukkan versi dari protokol HTTP. Dengan aturan ini penulisan versi protokol untuk versi 1.0 adalah HTTP/1.0 dan HTTP/1.1 untuk versi 1.1.

Penyertaan versi protokol ini diperlukan karena dalam sebuah hubungan HTTP, server dan client menggunakan versi yang berbeda. Untuk penanganannya digunakan versi yang tertinggi.

### **4.5.3 REQUEST**

Format baris mulai dari pesan request HTTP dimulai dengan metode request, diikuti oleh URL untuk request, versi protokol yang digunakan dan diakhiri oleh karakter CRLF.

**Request = Method SP Request-URI SP HTTP-Version CRLF**

Method menunjukkan metode apa yang hendak dilakukan atas resource yang ditunjuk oleh Request-URI. Ada beberapa metode yang didefinisikan oleh HTTP/1.1, yaitu : OPTIONS, GET, HEAD, POST, PUT, DELETE dan TRACE. Untuk setiap pesan request, server harus memberikan kode jawaban untuk memberitahu apakah client diperbolehkan mengakses menggunakan method yang diinginkan. Jika method tidak

boleh digunakan, server harus menjawab dengan kode 405 (Method Not Allowed). Diantara metode-metode di atas hanya metode GET & HEAD yang harus diimplementasikan oleh semua server. Jika server tidak mengimplementasikan sebuah metode atau tidak mengenal metode yang diminta client, maka server harus memberikan response 501 (not implemented).

#### **4.5.4 RESPONSE**

Setelah menerima request, server harus memberikan response HTTP atas request tersebut, yang terdiri atas **baris status**, **header-header** dan **isi pesan**. Baris status berisi kode-status yang berupa kode tiga digit dan frasa-alasan, yaitu penjelasan singkat atas kode-status tersebut.

Digit pertama kode-status menentukan kelas dari response. Protokol HTTP/1.1 mendefinisikan 5 nilai untuk digit pertama :

- 1xx** : Informational – request diterima, dan proses berlanjut
- 2xx** : Success – request diterima dan dimengerti
- 3xx** : Redirection – request membutuhkan tindakan lebih lanjut
- 4xx** : Client Error – request mengandung sintaks yang salah
- 5xx** : Server Error – server gagal melakukan tindakan sesuai server

Server HTTP dapat menghasilkan kode-status selain yang didefinisikan dalam RFC sepanjang digit pertama kode-status tersebut dimengerti oleh aplikasi HTTP.

Format baris-status adalah sebagai berikut :

**Baris-status = HTTP-version SP Status-Code SP Reason-Phrase CRLF**

Setelah baris response, server HTTP mengirimkan header-header response ke client. Header ini memberikan informasi mengenai server serta mengenai akses lanjutan ke URI.

**Header-response = Age|Location|Proxy-Authenticate  
|Public|Retry-After|Server|Vary  
|Warning|WWW-Authenticate**

#### **4.5.5 Entity**

Setiap pesan HTTP baik request maupun response dapat menyertakan isi pesan atau entity tergantung dari apakah pesan tersebut memungkinkan untuk membawa entity. Entity HTTP terdiri atas header entity dan isi entity. Header entity berisi informasi mengenai isi entity atau mengenai resource yang ditunjuk oleh Request-URI.

#### **4.5.6 Cache HTTP**

Salah satu cara untuk mempercepat response HTTP di jaringan adalah dengan menggunakan cache. Dalam protokol HTTP dimungkinkan dalam sebuah rantai request/response terdapat beberapa proxy yang dapat bertindak sebagai server cache. Spesifikasi protokol HTTP juga menyertakan elemen-elemen agar cache dapat dilakukan sebaik mungkin. Tujuan adanya cache adalah mencegah pengiriman request dan menghilangkan kebutuhan untuk mengirim response lengkap dari server. Cache mencegah pengiriman request ke server asal dengan menggunakan mekanisme kadaluwarsa sehingga memperpendek rantai request/response dan round trip. Menghilangkan kebutuhan untuk mengirim respon lengkap dari server asal dengan menggunakan mekanisme validasi berarti mengurangi kebutuhan bandwidth.