HTTP (HyperText Transfer Protocol)

Hypertext Transfer Protocol (HTTP) adalah sebuah protokol jaringan lapisan aplikasi yang digunakan untuk sistem informasi terdistribusi, kolaboratif, dan menggunakan hipermedia. Penggunaannya banyak pada pengambilan sumber daya yang saling terhubung dengan tautan, yang disebut dengan dokumen hiperteks, yang kemudian membentuk World Wide Web pada tahun 1990 oleh fisikawan Inggris, Tim Berners-Lee. Hingga kini, ada dua versi mayor dari protokol HTTP, yakni HTTP/1.0 yang menggunakan koneksi terpisah untuk setiap dokumen, dan HTTP/1.1 yang dapat menggunakan koneksi yang sama untuk melakukan transaksi. Dengan demikian, HTTP/1.1 bisa lebih cepat karena memang tidak usah membuang waktu untuk pembuatan koneksi berulang-ulang.

HTTP adalah sebuah protokol meminta/menjawab antara klien dan server. Sebuah klien HTTP (seperti web browser atau robot dan lain sebagainya), biasanya memulai permintaan dengan membuat hubungan ke port tertentu di sebuah server Webhosting tertentu (biasanya port 80). Klien yang mengirimkan permintaan HTTP juga dikenal dengan user agent. Server yang meresponsnya, yang menyimpan sumber daya seperti berkas HTML dan gambar, dikenal juga sebagai origin server. Di antara user agent dan juga origin server, bisa saja ada penghubung, seperti halnya proxy, gateway, dan juga tunnel.  
  
HTTP tidaklah terbatas untuk penggunaan dengan TCP/IP, meskipun HTTP merupakan salah satu protokol aplikasi TCP/IP paling populer melalui Internet. Memang HTTP dapat diimplementasikan di atas protokol yang lain di atas Internet atau di atas jaringan lainnya. seperti disebutkan dalam "implemented on top of any other protocol on the Internet, or on other networks.", tapi HTTP membutuhkan sebuah protokol lapisan transport yang dapat diandalkan. Protokol lainnya yang menyediakan layanan dan jaminan seperti itu juga dapat digunakan.."  
  
Sumber daya yang hendak diakses dengan menggunakan HTTP diidentifikasi dengan menggunakan Uniform Resource Identifier (URI), atau lebih khusus melalui Uniform Resource Locator (URL), menggunakan skema URI http: atau https: Sesuai dengan perkembangan infrastruktur internet maka pada tahun 1999 dikeluarkan HTTP versi 1.1 untuk mengakomodasi proxy, cache dan koneksi yang persisten.

Sebelum ini telah dijelaskan bahwa Hypertext Transfer Protocol digunakan untuk jenis layanan WWW di jaringan TCP/IP. Spesifikasi protokol ini didefinisikan oleh Tim Berners-Lee dalam [RFC 1945](http://tools.ietf.org/html/rfc1945) dan digunakan di Internet sejak tahun 1990. Pada bagian ini akan dijelaskan bagaimana protokol HTTP bekerja, dimulai dari model hubungan HTTP, format pesan, dan cache di protokol HTTP.

[RFC 1945](http://tools.ietf.org/html/rfc1945) yang mendefinisikan protokol HTTP versi 1.0 ternyata dianggap masih memiliki kekurangan dan kemudian IETF menspesifikasikan protokol HTTP versi baru, yaitu 1.1 dalam [RFC 2068](http://tools.ietf.org/html/rfc2068). Perbaikan atas HTTP 1.0 antara lain adalah koneksi persistent dan pipelined serta model cache yang lebih baik.

Protokol HTTP bersifat request-response, yaitu dalam protokol ini client menyampaikan pesan request ke server dan server kemudian memberikan response yang sesuai dengan request tersebut. Request dan response dalam protokol HTTP disebut sebagai request chain dan response chain. Hubungan HTTP yang paling sederhana terdiri atas hubungan langsung antara user agent dengan server asal. Hubungan HTTP tidak selalu seperti ini karena spesifikasi HTTP mengenal adanya beberapa komponen yang dapat terlibat dalam membentuk sebuah hubungan HTTP, yaitu client, user agent, server asal, proxy, gateway, dan tunnel.

Menurut spesifikasi HTTP, istilah-istilah di atas adalah sebagai berikut:

Client - 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 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 tiga 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 menerjemahkan request ke protokol server yang dilayaninya. Tunnel bertindak sebagai titik relay antara dua hubungan HTTP tanpa mengubah request dan response HTTP. Tunnel digunakan jika komunikasi perlu melalui sebuah perantara dan perantara tersebut tidak mengetahui isi dari pesan dalam hubungan tersebut. Gambar contoh hubungan HTTP yang melibatkan beberapa komponen dapat dilihat di gambar

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.

Hubungan Persistent

Perbedaan mendasar antara HTTP/1.1 dengan HTTP/1.0 adalah penggunaan hubungan persistent. Jika HTTP/1.0 membutuhkan sebuah koneksi TCP untuk setiap permintaan URI, maka HTTP/1.1 dapat menggunakan sebuah koneksi TCP saja untuk beberapa permintaan URI. Server HTTP/1.1 dapat mengasumsikan bahwa hubungan yang digunakan dengan client HTTP/1.1 adalah hubungan persistent, kecuali jika client menyatakan tidak hendak menggunakan hubungan persistent. Dalam mekanisme ini, server dan client dapat mengirim sinyal untuk menutup koneksi TCP menggunakan header

Connection: close.

Setelah sinyal ini dikirim, client tidak boleh lagi mengirimkan request ke server.

Server atau client yang berhubungan menggunakan protokol HTTP dengan versi lebih rendah dari 1.1 harus tidak mengasumsikan terjadinya hubungan persistent kecuali dinyatakan dengan jelas melalui sinyal

Connection: Keep-Alive.

Hubungan persistent juga mendukung request yang di-“pipeline”, yaitu client mengirimkan beberapa request sekaligus tanpa menunggu response selesai datang dari server. Server yang menerima request yang di-“pipeline” harus memberikan response sesuai urutan request. Client yang mendukung pipeline harus siap untuk kembali mengirimkan request tersebut jika server tidak memberikan response.

Format HTTP

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

“http:” “//” host [ “:”port ] [ abs\_path ]

Dengan,

host - nama domain Internet yang legal. port - bilangan yang menunjukkan port HTTP di host (default 80). abs\_path - lokasi resource di dalam host.

Contoh: <http://www.google.co.id/index.html> [http://sandbox.bellanet.org:80/~onno/the-guide/](http://sandbox.bellanet.org/~onno/the-guide/)

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 URI case-insensitive. Dengan aturan ini maka [http://sandbox.bellanet.org:80/~onno/the-guide/](http://sandbox.bellanet.org/~onno/the-guide/) sama dengan[http://sandbox.BELLANET.org:80/~onno/the-guide/](http://sandbox.bellanet.org/~onno/the-guide/) tetapi berbeda dengan [http://sandbox.bellanet.org:80/~onno/The-guide/](http://sandbox.bellanet.org/~onno/The-guide/).

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 generik seperti yang didefinisikan oleh [RFC 822](http://tools.ietf.org/html/rfc822). Pesan HTTP terdiri atas baris mulai, header pesan, dan isi pesan beserta entity (opsional). 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, tergantung dari field-field yang perlu disertakan dalam header tersebut. Terdapat empat 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

Header-umum = Cache-Control | Connection | Date | Pragma

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 persistent 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 pada pesan HTTP digunakan untuk mengganti protokol yang hendak digunakan, field ini berguna sebagai mekanisme transisi dari protokol HTTP/1.1 ke protokol tipe lain. Field Via digunakan oleh proxy dan gateway untuk memberitahu jalur yang digunakan dalam sebuah rantai request/response.

Isi pesan HTTP 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 dan 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 dan 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 dapat saja terjadi dalam sebuah hubungan HTTP, server dan client menggunakan versi yang berbeda. Jika ini terjadi maka server dan client harus menggunakan versi HTTP tertinggi yang mungkin agara keduanya dapat berkomunikasi dengan baik.

Request

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

Baris-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 & 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).. Dari semua metode di atas, hanya metode GET dan 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).

Metode GET mengambil informasi apa saja dalam bentuk entity yang diidentifikasikan oleh Request-URI. Metode HEAD sama dengan metode GET kecuali bahwa server harus tidak mengirimkan entity yang ditunjukkan oleh Request-URI. Metode POST digunakan untuk meminta server menempatkan entity yang dikirim bersama request sebagai subordinat dari Request-URI yang dituju. Metode ini biasa digunakan dalam mengirimkan form.

Metode PUT meminta server untuk menempatkan entity request sebagai Request-URI. Perbedaan antara POST dan PUT adalah Request-URI dalam metode POST bertindak sebagai proses penerima data atau sebagai gateway, sementara dalam metode PUT, Request-URI adalah entity yang terdapat dalam request.

Metode DELETE meminta server untuk menghapus URI yang dikirim client. Client tidak dapat menjamin server berhasil melaksanakan request yang diminta. Metode TRACE digunakan untuk meminta loop-back pesan. Metode TRACE ini dapat digunakan untuk diagnostik.

Request-URI menunjukkan resource yang hendak diakses melalui pesan request. Request-URI dapat berupa URI absolut, path absolut, atau tanda asteriks (‘\*’). Request-URI tergantung pada request itu sendiri. Tanda asteriks ‘\*’ berarti bahwa request tidak merujuk pada resource tertentu, melainkan pada server. Request-URI asteriks hanya boleh digunakan jika metode yang digunakan tidak harus merujuk pada sebuah resource. Contoh:

OPTIONS \* HTTP/1.1

Request-URI absolut diperlukan jika request dilakukan ke server proxy. Server proxy yang biasa digunakan squid dengan port 3128. Proxy dapat memberikan jawaban berdasarkan cache atau meneruskan request tersebut. Proxy boleh meneruskan request tersebut langsung ke server asal atau ke server proxy yang lain. Jika proxy meneruskan request ke server proxy lain, maka Request-URI tidak boleh diubah. Contoh request absolut:

GET <http://www.google.co.id/index.html> HTTP/1.1

Request dalam bentuk path absolut digunakan untuk menentukan resource pada server atau gateway. Jika sebuah request menggunakan Request-URI path absolut maka request juga harus mengirimkan lokasi jaringan URI dalam sebuah field header host. Dalam kasus seperti request di atas, proxy dapat menghubungi server www.google.co.id pada port 80dan mengirimkan request

GET /index.html HTTP/1.1 Host: www.google.co.id

Perhatikan bahwa dalam Request-URI path absolut informasi host tujuan juga turut disertakan. Server HTTP/1.1 harus mengetahui bahwa sebuah request ditentukan dengan melihat Request-URI dan header host. Server HTTP/1.1 yang tidak membolehkan resource berbeda untuk field Host yang berbeda dapat mengabaikan field header Host tersebut. Jika server tersebut membolehkan resource yang berbeda maka server tidak boleh mengabaikan field header Host dalam Request-URI path-absolut. Jika server tersebut menerima Request-URI absolut, maka field header Host harus diabaikan. Jika ternyata host yang dirujuk pada Request-URI absolut atau path-absolut bukan host yang valid di server, maka server harus memberikan respon kesalahan 400 (Bad Request).

Setelah baris-request, client mengirimkan request header yang berisi informasi mengenai request atau mengenai client itu sendiri ke server. Header-header tersebut adalah:

Header-request = Accept | Accept-Charset | Accept-Encoding  
Field Header Keterangan Accept Menentukan tipe media yang dapat diterima sebagai respon dari server Contoh: Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, \*/\* Accept-Charset Mengindikasikan set karakter yang dapat diterima sebagai respon Contoh: Accept-Charset: iso-8859-1,\*,utf-8 Accept-Language Memperkecil jenis bahasa yang lebih disukai untuk diterima oleh client Contoh: Accept-Language: da, en-gb;q=0.8, en;q=0.7 Host Menentukan host dan port tempat resource hendak diambil. Client HTTP/1.1 harus menyertakan field ini dalam setiap request. Contoh: Host: www.google.co.id:80 If-Modified-Since Digunakan bersama metode GET memberikan kondisi bahwa jika resource yang hendak diakses belum diubah sejak waktu yang ditentukan oleh field ini, maka resource tersebut tidak akan dikirim oleh server, melainkan server memberikan response 304 (not modified) tanpa isi pesan. Contoh: If-Modified-Since: Sat, 21 Feb 2006 19:43:31 GMT User-Agent Berisi informasi mengenai user agent yang melakukan request ke server. Contoh: User-Agent: Lynx/2.7.1 libwww-FM/2.14

Contoh request lengkap yang berasal dari browser Internet Explorer 6.0 yang hendak mengakses [http://www.yahoo.com](http://www.yahoo.com/)

GET / HTTP/1.1 Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, application/x-shockwave-flash, \*/\* Accept-Language: en-us Accept-Encoding: gzip, deflate User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1) Host: www.yahoo.com Connection: Keep-Alive Cookie: FPB=881h90lik11rsfl3; B=5ut53s91rs4no&b=2;

Contoh request lengkap yang berasal dari browser Opera 7.52 yang hendak mengakses [http://www.google.co.id](http://www.google.co.id/).

GET / HTTP/1.1 User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1) Opera 7.52 [en] Host: www.google.co.id Accept: text/html, application/xml;q=0.9, application/xhtml+xml, image/png, image/jpeg, image/gif, image/x-xbitmap, \*/\*;q=0.1 Accept-Language: en Accept-Charset: windows-1252, utf-8, utf-16, iso-8859-1;q=0.6, \*;q=0.1 Accept-Encoding: deflate, gzip, x-gzip, identity, \*;q=0 Connection: Keep-Alive 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 request.

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 lebih lanjut ke URI.

Header-response = Age | Location | Proxy-Authenticate

Field Header Keterangan Age Perkiraan lama waktu sejak response atas URI diambil dari server asal. Header ini harus digunakan oleh cache HTTP/1.1. Location Digunakan untuk mengarahkan client ke URL lain. Contoh: Location: <http://www.w3c.org/pub/WWW/People.html> Proxy-Authenticate Field ini memungkinkan client untuk mengirimkan identitasnya untuk menggunakan proxy yang membutuhkan autentikasi. Public Memperlihatkan jenis metode yang didukung oleh server Retry-After Dapat digunakan bersama kode-status 503 untuk menunjukkan berapa lama layanan belum dapat diberikan. Contoh: Retry-After: Fri, 31 Mar 2006 23:59:59 GMT Server Berisi informasi mengenai server yang menangani request Contoh: Server: Apache/2.0.54 (Fedora) Vary Digunakan server untuk mengirimkan sinyal bahwa entity yang dikirim adalah hasil negosiasi yang ‘server driven’. URI dengan header ini harus tidak di-cache. Warning Digunakan untuk memberi informasi tambahan selain dari kode-status. WWW-Authenticate Dikirimkan beserta kode-status 401. Terdiri atas setidaknya satu challenge yang menentukan scheme autentikasi dan parameter-parameternya.