**22.3 HTTP**

**Hypertext Transfer Protocol (HTTP)** – giao thức truyền tải siêu văn bản, là một giao thức được sử dụng chủ yếu để truy cập dữ liệu trên mạng lưới toàn cầu World Wide Web. Chức năng của HTTP giống như sự tổ hợp của FTP (chương 21) và SMTP (chương 23). HTTP giống FTP vì đều truyền files và sử dụng các dịch vụ của TCP. Tuy nhiên, HTTP đơn giản hơn FTP vì chỉ sử dụng duy nhất một kết nối TCP. HTTP không có điều khiển kết nối riêng biệt, chỉ có dữ liệu được truyền đi giữa client và server.

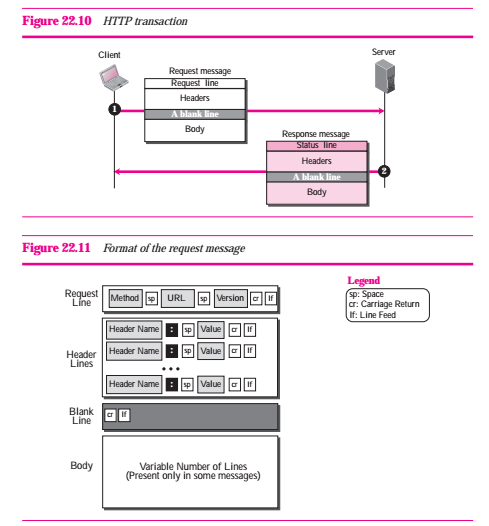
HTTP giống SMTP vì dữ liệu được truyền đi giữa client và server giống như những SMTP message và định dạng của những message được kiểm soát bởi headers như MIME (Multipurpose Internet Mail Extensions). Điều khác SMTP là HTTP message không dành cho người đọc mà được đọc và thông dịch bởi HTTP server và HTTP client (trình duyệt). STMP message thì được lưu trữ và chuyển đi nhưng HTTP message thì được chuyển đi ngay lập tức. Lệnh điều khiển từ client đến server được nhúng vào request message. Nội dụng của file được yêu cầu hoặc thông tin khác được nhúng vào response message**. HTTP sử dụng những dịch vụ của TCP trên port 80 mà chúng ta đều đã biết.**

**HTTP Transaction (giao tác HTTP)**

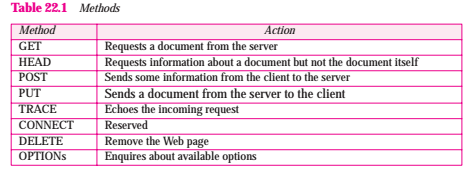
Hình 22.10 giải thích giao tác HTTP giữa client và server. Mặc dù HTTP sử dụng các dịch vụ của TCP nhưng HTTP lại là giao thức stateless (không trạng thái), nghĩa là server không giữ lại thông tin của client. Client khởi tạo giao tác bằng cách gửi request, server hồi đáp bằng cách gửi lại response.

*Request Message*

Định dạng của request được miêu tả ở hình 22.11. Một request message bao gồm request line, header và thỉnh thoảng có body.

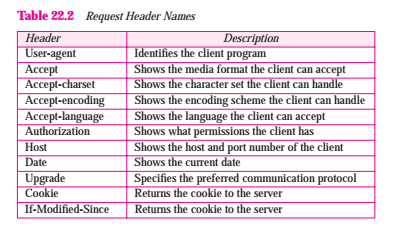


**Request Line**: Dòng đầu tiên trong request message được gọi là request line. Có ba trường trong dòng này, được phân cách bởi vài ký tự phân cách như trong hình 22.11. Các trường đó là called methods (phương thức được gọi), URL và Version (phiên bản). Ba trường này được phân cách bởi ký tự khoảng trắng. Hai ký tự ở cuối dòng là carriage return và line feed (để kết thúc dòng và trở về đầu dòng tiếp theo). Trường method định nghĩa **request type.** Ở version HTTP 1.1, có một vài phương thức được định nghĩa như trong bảng 22.1



Trường thứ hai, URL, đã được thảo luận ở chương trước. URL xác định địa chỉ và tên của trang Web tương ứng. Trường thứ ba, version, cho biết phiên bản của giao thức, phiên bản HTTP dùng nhiều nhất hiện nay là 1.1

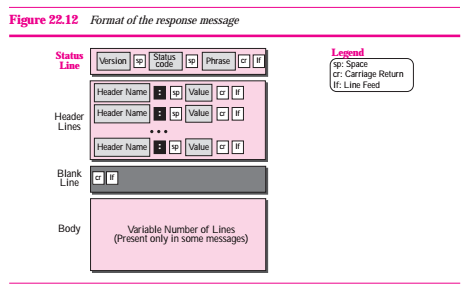
**Header Lines trong Request Message**: sau request line, có thể có hoặc không có các dòng **request header**. Mỗi header line gửi thông tin thêm từ client đến server. Ví dụ, client có thể yêu cầu tài liệu được gửi đi theo định dạng đặc biệt. Mỗi header line có header name, dấu hai chấm, khoảng trắng và header value (như trong hình 22.11). Chúng ta sẽ xem vài header line trong ví dụ ở cuối chương. Bảng 22.2 cho thấy vài header name thường được dùng trong request. Trường value xác định giá trị được liên kết với mỗi header name. Danh sách các value có thể được tìm trong RFCs tương ứng.



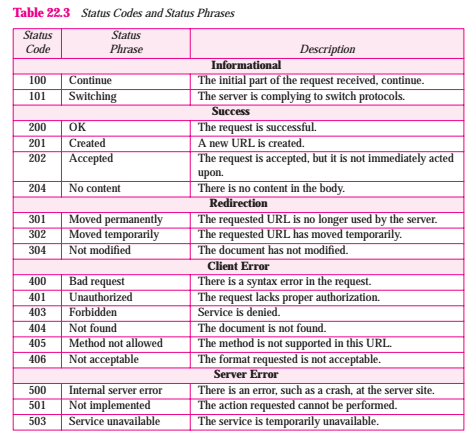
**Body trong Request Message**: body có thể có hoặc không trong request message. Body thường chứa chú thích được gửi đi.

*Response Message*

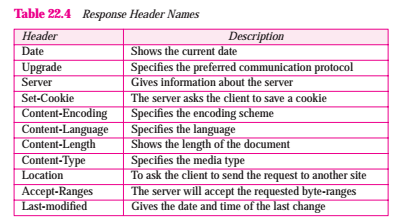
Định dạng của response message được chỉ ra ở hình 22.12. Response message bao gồm: status line, header line, blank line và thỉnh thoảng có body



**Status Line**: dòng đầu tiên của response message được gọi là status line. Có ba trường trong dòng này được phân cách bởi các khoảng trắng và kết thức bởi carriage return và line feed. Trường đầu tiên định nghĩa phiên bản của giao thức HTTP, hiện tại là 1.1. Trường status code gồm ba ký số định nghĩa trạng thái của request. Mã trong khoảng 100 chỉ mang tính chất thông tin, mã trong khoảng 200 biểu thị cho việc yêu cầu thành công, mã trong khoảng 300 chuyển hướng client đến URL khác, mã trong khoảng 400 biểu thị lỗi ở phí client, mã trong khoảng 500 biểu thị lỗi ở phía server. Status phrase (cụm từ trạng thái) giải thích mã trạng thái ở dạng chữ, các giá trị khả thi của mã trạng thái và cụm từ trạng thái được biểu thị ở bảng 22.3



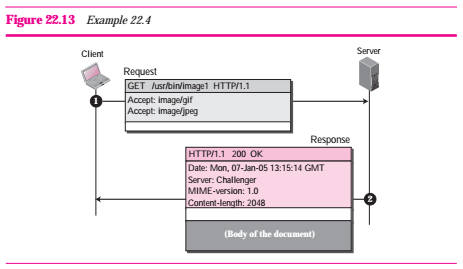
**Header Lines trong Response Message**: sau status line, có thể có 0 hoặc nhiều hơn những dòng **response header**. Mỗi header line gửi thông tin thêm từ server đến client. Ví dụ, người gửi có thể gửi thêm thông tin về tài liệu. Mỗi header line có: header name, dấu hai chấm, khoảng trắng và header value. Chúng ta sẽ xem vài header line trong ví dụ ở cuối chương này. Bảng 22.4 biểu thị vài header name thường dùng trong response message.



**Body**: body chứa tài liệu được gửi từ server đến client. Body luôn có trừ khi response là một message lỗi

**Example 22**.**4**

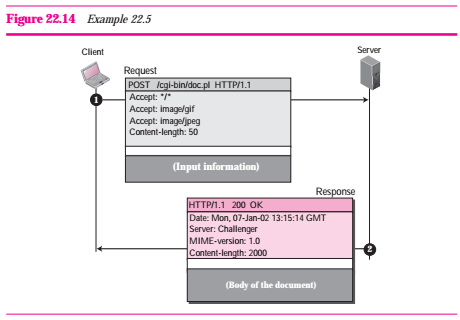
Đây là ví dụ lấy dữ liệu (xem hình 22.13)



Chúng ta sử dụng phương thức GET để truy xuất một ảnh với đường dẫn /usr/bin/image1. Request line biểu thị phương thức (GET), URL và HTTP version (1.1). Header có hai dòng biểu thị rằng client có thể chấp nhận hình ảnh ở định dạng GIF hoặc JPEG. Request không có body. Response message chứa status line và bốn dòng header. Header line xác định ngày, server, MIME version và độ dàng của dữ liệu (byte). Body của tài liệu dựa theo header.

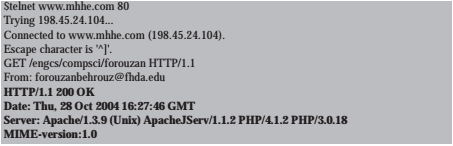
**Example 22.5**

Trong ví dụ này, client muốn gửi dữ liệu đến server. Chúng ta sử dụng phương thức POST. Request line cho biết phương thức (POST), URL và HTTP version (1.1). Có bốn dòng header. Body chứa thông tin nhập vào. Response message chứa status line và bốn dòng header. Tài liệu được tạo, là tài liệu CGI (Common Gateway Interface) được chứa trong body (xem hình 22.14)



**Example 22.6**

HTTP dùng ký tự ASCII. Ví dụ này cho thấy làm thế nào client có thể kết nối trự tiếp tới server sử dụng TELNET, đăng nhập vào port 80. Ba dòng đầu cho biết kết nối thành công. Sau đó chúng ta gõ ba dòng kế tiếp: dòng đầu tiên biểu thị request line (phương thức GET), dòng thứ hai là header (xác định host), dòng thứ ba là khoảng trống kết thúc request. Server response có bảy dòng bắt đầu với status line. Dòng trống ở cuối kết thúc server response. File gồn 14,230 dòng được nhận sau dòng trống (không được hiển thị ở ví dụ này). Dùng cuối là đầu ra của client





**Conditional Request (request có điều kiện)**

Client có thể thêm điều kiện vào request. Trong trường hợp này, server sẽ gửi yêu cầu trang Web nếu đáp ứng điều kiện hoặc thông báo cho client nếu không đáp ứng được điều kiện. Một trong những điều kiện thông dụng nhất bị bắt buộc bởi client là thời gian và ngày trang Web được chỉnh sửa. Client có thể gửi header line If-Modified-Since ở request để nói cho server biết rằng nó cần trang nếu như trang dã bị chỉnh sửa sau một mốc thời gian nhất định.

Example 22.7

Ví dụ dưới đây cho thấy làm thế nào một client áp đặt điều kiện về việc chỉnh sửa dữ liệu và điều kiện về thời gian lên một request.



Status line ở response cho thấy tệp đã không được chỉnh sửa sau mốc thời gian đã xác định. Body của response message cũng rỗng



**Persistence (sự bền vững)**

HTTP, từ các bản trước phiên bản 1.1, quy định một kết nối không bền vững,trong khi đó kết nối bền vững là mặc định trong phiên bản 1.1

*Nonpersistent Connection*

Trong một kết nối không bền vững, một kết nối TCP được tạo ra cho mỗi cặp request/response

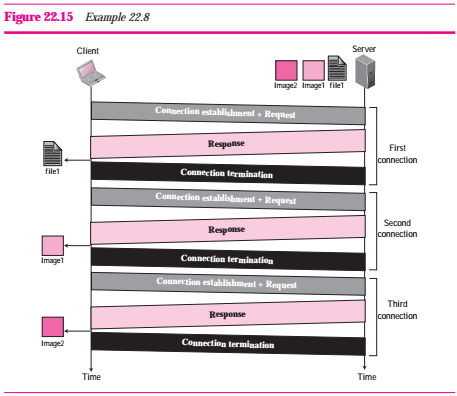
Sau đây là danh sách các bước thực hiện trong chiến lược kết nối không bền vững:

1. Client mở một kết nối TCP và gửi request
2. Server gửi response và đóng kết nối
3. Client đọc dữ liệu cho đến khi nó gặp đánh dấu hết tệp. Sau đó nó đóng kết nối

Trong chiến lược này, nếu tệp chứa liên kết đến N bức ảnh khác nhau trong các tệp khác nhau (tất cả được chứa trong cùng một server), kết ni61 phải được mở và đóng N+1 lần. Chiến lược không bền vững áp đặt lượng lớn tải lên server vì server cần N+1 buffer khác nhau và yêu cầu phương pháp slow start mỗi lần mở kết nối.

**Example 22.8**

Hình 22.15 cho thấy ví dụ về kết nối không bền vững. Client cần truy cập một tệp chứa hai liên kết hình ảnh. Tệp văn bản và hình ảnh được đặt ở cùng một server.



**Cookies**

Mạng lưới toàn cầu ban đầu được thiết kế là stateless. Client gửi một yêu cầu; server hồi đáp. Mối quan hệ của chúng chỉ có vậy. Thiết kế ban đầu của mạng lười toàn cầu truy xuất tài liệu có sẵn một cách công khai, hoàn toàn phù hợp với mục đích này. Nhưng ngày nay Web có những chức năng khác, một vài chức năng đó được liệt kê dưới đây:

* Website đang được sử dụng như một *electronic store* (cửa hàng điện tử) cho phép người dùng duyệt qua cửa hàng, chọn sản phẩm mình muốn, đặt chúng vào trong giỏ điện tử và cuối cùng là thanh toán với thẻ tín dụng
* Vài website cần cho phép chỉ những *registered client* (khách hàng đã đăng ký) mới có thể truy cập
* Vài website được dùng như *portal* (cổng thông tin): để người dùng chọn trang Web mà mình muốn xem
* Vài website chỉ để *advertising* (quảng cáo)

Vì những mục đích này, kỹ thuật cookie được lập ra. Chúng ta đã đề cập về công dụng của cookies ở tầng Transport trong chương 15; bây giờ chúng ta cùng thảo luận về công dụng của nó đối với trang Web

*Tạo và lưu trữ Cookies*

Tạo và lưu trữ cookies phụ thuộc vào việc cài đặt. Tuy nhiên, về nguyên tắc thì đều như nhau

1. Khi server nhận yêu cầu từ client, nó lưu lại thông tin về client trong một tệp hoặc một chuỗi. Thông tin này có thể bao gồm domain name (tên miền) của client, nội dung của cookie (thông tin server đã thu thập về client như: tên, số đăng ký,...), dấu thời gian, và những thông tin khác tùy thuộc vào việc cài đặt
2. Server đính kèm cookie vào trong response mà nó gửi cho client
3. Khi client nhận được response, trình duyệt chứa cookie trong thư mục cookie được sắp xếp theo tên miền server

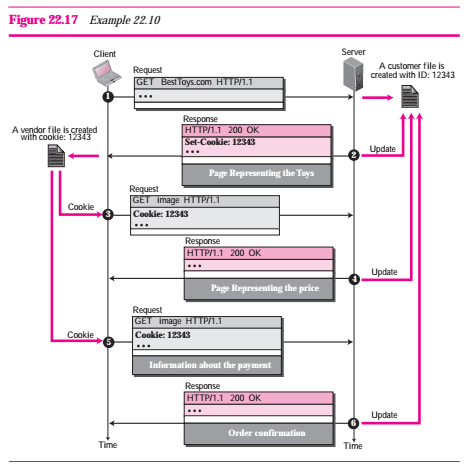
*Dùng Cookies*

Khi client gửi yêu cầu cho server, trình duyệt tìm trong thư mục cookie để xem có cookie được gửi bởi server đó hay không. Nếu có, cookie sẽ được đính vào trong request. Khi server nhận request, nó biết rằng đây là client cũ, không phải client mới. Lưu ý rằng nội dung của cookie không bao giờ được đọc bởi trình duyệt và không được tiết lộ với người dùng, có nghĩa là cookie được *made* (làm) ra bởi server và *eaten* (ăn) bởi server. Bây giờ chúng ta xem làm thế nào cookie được dùng cho bốn mục đích đã đề cập ở trên:

* Cửa hàng điện tử (e-commerce: thương mại điện tử) có thể dùng cookie cho khách hàng mua sắm. Khi khách chọn một sản phẩm và thêm vào trong giỏ, một cookie chứa thông tin về sản phẩm như số của sản phẩm và giá tiền của đơn vị sản phẩm được gửi đến trình duyệt. Nếu khách hàng chọn sản phẩm thứ hai, cookie được cập nhật với thông tin mới. Và hơn nữa, khi khách hàng kết thức mua sắm và muốn thanh toán, cookie cuối cùng sẽ được truy xuất và tính tổng số tiền
* Website hạn chế truy cập chỉ cho khách hàng đã đăng ký chỉ gửi cookie cho khách hàng khi khách hàng đăng ký lần đầu tiên. Mỗi lần truy cập lại, chỉ có client gửi cookie phù hợp mới được cho phép
* Web cổng thông tin dùng cookie theo các tương tự. Khi người dùng chọn trang yêu thích, cookie được tạo và gửi đi. Nếu trang đó được truy cập lần nữa, cookie được gửi đến server cho biết người dùng đang tìm trang nào
* Cookie cũng có thể được dùng cho các cơ quan quảng cáo. Một cơ quan quảng cáo có thể đặt banner quảng cáo trên một vài website chính mà người dùng thường truy cập. Cơ quan quảng cáo chỉ cung cấp một URL cho địa chỉ của banner thay vì banner. Khi người dùng truy cập website chính và nhấp vào biểu tượng của công ty quảng cáo, một yêu cầu được gửi tới cơ quan quảng cáo. Cơ quang quảng cáo gửi banner, ví dụ như một tệp GIF, và cũng bao gồm một cookie với ID của người dùng. Bất kỳ khi nào banner được dùng trong tương lai sẽ được thêm vào cơ sở dữ liệu miêu tả hành vi của người dùng trên Web. Cơ quan quảng cáo đã biên soạn sự quan tâm của người dùng và có thể bán thông tin này cho các tổ chức khác. Công dụng này của cookie đã gây rất nhiều tranh cãi. Hy vọng, vài quy định mới sẽ được đặt ra để bảo toàn sự riêng tư của người dùng

**Example 22.10**

Hình 22.17 cho thấy ngữ cảnh một cửa hàng điện tử có thể kiếm lợi từ công dụng của cookie. Giả sử người mua hàng muốn mua một món đồ chơi từ của hàng điện tử tên BestToys. Trình duyệt của người mua (client) gửi yêu cần đến BestToys server. Server tạo một giỏ hàng rỗng (một danh sách) cho client và gắn một ID cho giỏ hàng (ví dụ là 12343). Server sau đó gửi response message chứa hình ảnh của tất cả các đồ chơi có sẵn với một liên kết dưới mỗi món đồ chơi để chọn món đồ chơi đó nếu nhấp vào. Response message này cũng bao gồm dòng Set-Cookie header với giá trị 12343. Client hiển thị hình ảnh và lưu giá trị của cookie trong một tệp tên BestToys. Cookie không được tiết lộ với khách hàng. Bây giờ khách hàng chọn một món đồ chơi và nháp vào. Client sẽ gửi yêu cầu và bao gồm cả ID 12343 trong dòng Cookie header. Mặc dù server có thể bận và quên mất khách hàng, nhưng khi nó nhận được yêu cầu và kiểm tra header, nó sẽ tìm được giá trị 12343 của cookie. Server biết đây không phải khách mới, nó sẽ tìm giỏ hàng có ID 12343. Giỏ hàng (danh sách) được mở ra và những món đồ chơi đã được chọn sẽ được thêm vào trong danh sách. Bây giờ server gửi response khác đến cho khách để báo tổng giá tiền và yêu cầu cung cấp hình thức thanh toán. Khách hàng cung cấp thông tin về thẻ tín dụng của mình và gửi một request mới với giá trị cookie là ID 12343. Khi request đến server, server lại thấy ID 12343 và chấp nhận đơn đặt hàng và hình thức thanh toán và gửi lại xác nhận bằng response. Những thông tin khác về client như số thẻ tín dụng, tên, địa chỉ được lưu trong server. Nếu khách hàng truy cập cửa hàng lúc nào đó trong tương lai, client gửi cookie lại; cửa hàng truy xuất tệp và có tất cả thông tin về client.



**Web Caching: Proxy Server**

HTTP hỗ trợ proxy servers. Proxy server là một máy tính giữ các bản sao chép của các response của các request gần đây. HTTP client gửi request đến proxy server, proxy server kiểm tra cache của nó. Nếu response không được lưu trong cache, proxy server gửi request đến server tương ứng. Response được trả về được gửi đến proxy server và lưu trự lại cho các request trong tương lại từ các client khác.

Proxy server giảm việc tải dữ liệu từ server gốc, giảm tác nghẽn và cải thiện độ trễ. Tuy nhiên, để dùng proxy server, client buộc phải cấu hình để truy cập vào proxy thay vì server đích.

Chú ý rằng proxy server hình động như cả server và client. Khi nó nhận yêu cầu từ client mà nó có sẵn response, nó như server gửi lại response cho client. Khi nó nhận request từ client mà nó không có sẵn response, lúc đầu nó như client gửi request đến server đích. Khi nhận được response trả về, nó một lần nữa như server và gửi response cho client.

*Proxy Server Location*

Proxy server thường được đặt ở phía client. Có nghĩa là chúng ta có thể có một hệ thống phân cấp các proxy server như dưới đây:

1. Máy client cũng có thể được dùng như proxy server với dung lượng nhỏ chứa các response của các request được gửi bởi client
2. Trong một công ty, proxy server có thể được cài đặt trên máy tính LAN để giảm tải dữ liệu ra vào LAN
3. Một ISP với nhiều khách hàng có thể cài đặt proxy server để giảm tải dữ liệu ra vào mạng ISP

*Cache Update*

Một câu hỏi rất quan trọng là response nên tồn tại lại trong proxy server bao lâu trước khi bị xóa và thay thế. Nhiều chiến lược khác nhau đã được sử dụng cho mục đích này. Một giải pháp đó là chứa danh sách các trang mà thông tin được lưu lại trong một thời gian giống nhau. Ví dụ một cơ quan tin tức có thể thay đổi trang tin mỗi sáng. Điều này có nghĩa là một proxy server có thể lấy thông tin từ sáng sớm và giữ nó đến ngày hôm sau. Một đề cử khác là thêm vài header để biểu thị thời gian thay đổi thông tin cuối cùng. Proxy server sau đó có thể dùng thông tin trong header này để đoán xem thông tin sẽ hợp lệ trong bao lâu. Còn nhiều đề cử khác cho Web caching, nhưng chúng ta hãy để chúng cho những cuốn sách chuyện biệt khác trong môn học này.

**HTTP Security**

Giao thức HTTP không cung cấp cơ chế an ninh. Tuy nhiên, chúng ta sẽ được giới thiệu trong chương 30, HTTP có thể chạy trên tầng Secure Socket Layer (SSL). Trong trường hợp này, HTTP được gọi là HTTPS. HTTPS cung cấp sự bảo mật, chứng thực client và server, và toàn vẹn dữ liệu