Chương 2: Nội dung

Web và HTTP

Một số thuật ngữ:

- *Trang web* bao gồm một số *đối tượng (object)*
- Đối tượng có thể là tệp HTML, ảnh JPEG, Java applet, tệp audio,…
- Trang web bao gồm *tệp HTML cơ bản* chứa *một số đối tượng được tham chiếu*
- Mỗi đối tượng được định địa chỉ bởi một *URL,* ví dụ:

`www.someschool.edu/someDept/pic.gif`

Tên host | Tên đường dẫn

Khái quát về HTTP

HTTP: hypertext transfer protocol

- Giao thức tầng ứng dụng của Web
- Mô hình client/server
  - *client:* trình duyệt (browser) yêu cầu, nhận (sử dụng giao thức HTTP), và "hiển thị" các đối tượng Web
  - *server:* Máy chủ web (web server) gửi (sử dụng giao thức HTTP) các đối tượng đáp ứng cho yêu cầu.

Khái quát về HTTP

HTTP dùng TCP:

- Client khởi tạo kết nối TCP (tạo socket) tới server, cổng 80
- Server chấp nhận kết nối TCP từ client
- Thông điệp HTTP (thông điệp giao thức tầng ứng dụng) được trao đổi giữa trình duyệt (HTTP client) và máy chủ Web (HTTP server)
- Đóng kết nối TCP

HTTP là "không trạng thái"

- Server không lưu giữ thông tin về những yêu cầu trước đó của client.

Vấn đề liên quan

Giao thức lưu giữ "trạng thái" khá phức tạp!

- Lịch sử quá khứ (trạng thái) cần phải được lưu giữ
- Nếu server/client bị sự cố, thì quan điểm về "trạng thái" của chúng có thể không nhất quán, cần phải được điều chỉnh.

Kết nối HTTP: 2 loại

HTTP không bền vững

- Mở kết nối TCP
- Chỉ có tối đa một đối tượng được gửi qua một kết nối TCP
- Kết nối TCP sau đó sẽ được đóng lại.

➔ Việc tải về nhiều đối tượng sẽ yêu cầu nhiều kết nối.

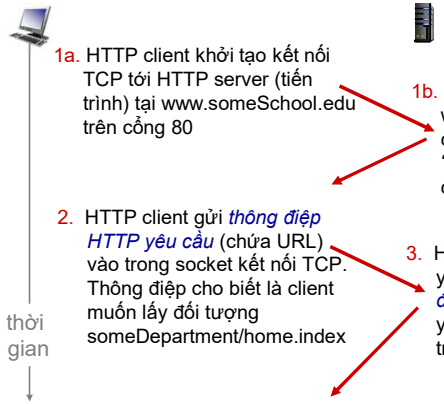
HTTP bền vững

- Kết nối TCP được mở tới một server
- Nhiều đối tượng có thể được gửi qua một kết nối TCP duy nhất giữa client và server đó
- Kết nối TCP sau đó sẽ được đóng lại.

Ví dụ: HTTP không bền vững

Giả sử người dùng gõ URL:
`www.someSchool.edu/someDepartment/home.index`

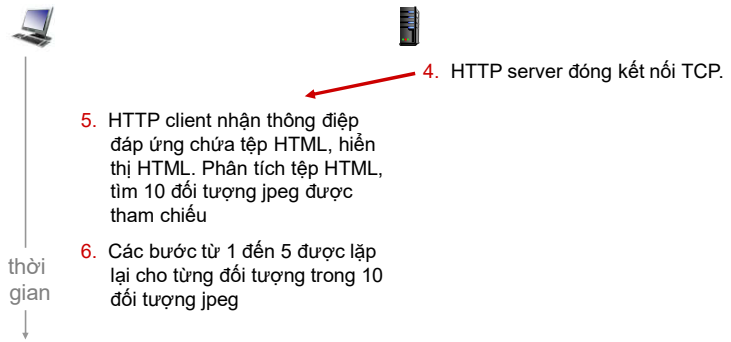
(chứa văn bản, tham chiếu tới 10 ảnh jpeg)



Ví dụ: HTTP không bền vững

Giả sử người dùng gõ URL:
`www.someSchool.edu/someDepartment/home.index`

(chứa văn bản, tham chiếu tới 10 ảnh jpeg)



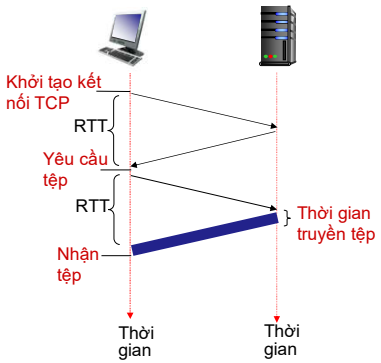
HTTP không bền vững: thời gian đáp ứng

RTT (định nghĩa): thời gian để một gói tin nhỏ đi từ client đến server và quay lại.

Thời gian đáp ứng HTTP:

- Một RTT để khởi tạo kết nối TCP
- Một RTT để gửi HTTP yêu cầu và một vài byte đầu tiên của HTTP đáp ứng được trả về
- Thời gian truyền tệp/đối tượng

Thời gian đáp ứng của HTTP không bền vững = 2RTT+ thời gian truyền tệp



HTTP bền vững

Vấn đề với HTTP không bền vững:

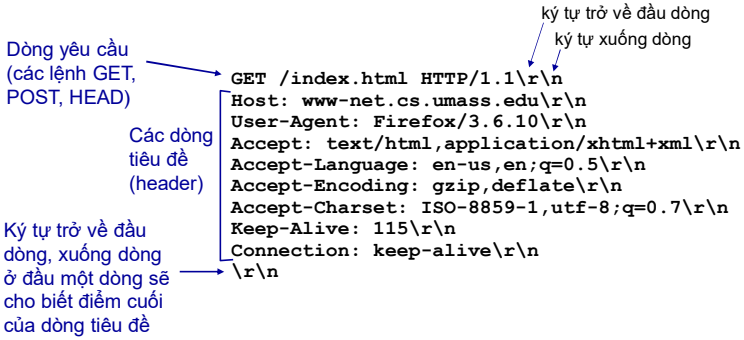
- Cần 2 RTT cho mỗi đối tượng
- Hệ điều hành liên quan đến *mỗi* kết nối TCP
- Các trình duyệt thường mở nhiều kết nối TCP song song để lấy các đối tượng được tham chiếu

HTTP bền vững (HTTP 1.1):

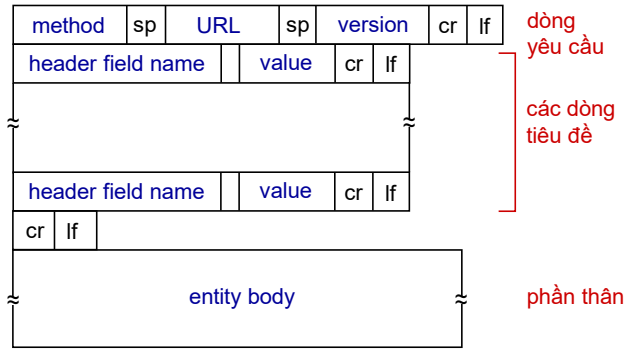
- Server để mở kết nối sau khi gửi đáp ứng
- Chuỗi các thông điệp HTTP tiếp theo giữa cùng client/server sẽ được gửi thông qua kết nối mở này
- Client gửi yêu cầu ngay sau khi gặp một đối tượng được tham chiếu
- Ít nhất là một RTT cho tất cả các đối tượng được tham chiếu (giảm thời gian phản hồi xuống một nửa).

Thông điệp HTTP yêu cầu

- Có hai loại thông điệp HTTP: *yêu cầu* và *đáp ứng*
- Thông điệp HTTP yêu cầu:
  - ASCII (định dạng con người có thể đọc được)



Thông điệp HTTP yêu cầu: định dạng tổng quát



Các thông điệp HTTP yêu cầu khác

Phương thức POST:

- Trang web thường chứa form input
- Đầu vào (input) được tải lên server trong phần thân thực thể (entity body) của thông điệp yêu cầu HTTP POST.

Phương thức GET (để gửi dữ liệu đến server):

- Bao gồm dữ liệu của người dung trong trường URL của thông điệp yêu cầu HTTP GET (theo sau dấu '?'):

```
www.somesite.com/animalsearch?monkeys&banana
```

Phương thức HEAD:

- Các yêu cầu về tiêu đề (chỉ) được trả về khi có URL xác định được yêu cầu bằng một phương thức HTTP GET.

Phương thức PUT:

- Tải file (/đối tượng) mới lên server
- Thay thế hoàn toàn file tồn tại tại URL xác định bằng nội dung bên trong phần thân thực thể của thông điệp yêu cầu HTTP POST.

Thông điệp HTTP đáp ứng

Dòng trạng thái (Giao thức mã trạng thái, cụm từ trạng thái)

Các dòng tiêu đề

Dữ liệu, ví dụ tệp HTML yêu cầu

```
HTTP/1.1 200 OK\r\n
Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 17:00:02
   GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-8859-
   1\r\n
\r\n
data data data data data ...
```


Các mã trạng thái HTTP đáp ứng

- Mã trạng thái xuất hiện ngay trong dòng đầu tiên của thông điệp đáp ứng từ server đến client.
- Ví dụ một số mã:

200 OK
- Yêu cầu thành công, đối tượng yêu cầu nằm ở phía sau thông điệp này

301 Moved Permanently
- Đối tượng yêu cầu đã bị di chuyển, vị trí mới được xác định ở phía sau thông điệp này (Location: field)

400 Bad Request
- Server không hiểu thông điệp yêu cầu

404 Not Found
- Tài liệu yêu cầu không có trong server này

505 HTTP Version Not Supported

Tự kiểm tra HTTP (phía client)

1. Telnet đến Web server ưa thích của bạn:

```
telnet cis.poly.edu 80
```

Mở kết nối TCP ở cổng 80 (cổng mặc định của HTTP server) tại cis.poly.edu. Nhập yêu cầu gì đó và gửi tới cổng 80 tại cis.poly.edu

2. Nhập yêu cầu trong lệnh GET HTTP:

```
GET /kurose_ross/interactive/index.php
HTTP/1.1

Host: cis.poly.edu
```

Bằng cách gõ lệnh này (nhấn enter 2 lần), nghĩa là gửi yêu cầu GET tối thiểu (nhưng đầy đủ) tới HTTP server

3. Xem thông điệp đáp ứng được gửi về từ HTTP server!

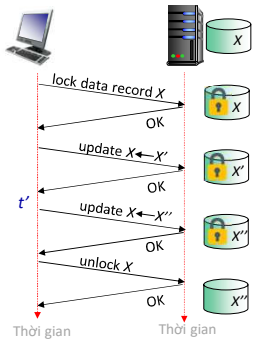
(hoặc dùng Wireshark để xem thông điệp HTTP yêu cầu/đáp ứng bắt được)

Duy trì trạng thái user-server: cookie

Nhớ lại: HTTP GET/tương tác phản hồi là *không trạng thái*

- Không có khái niệm về việc trao đổi các thông điệp HTTP nhiều bước để hoàn thành một "giao dịch" trên Web
  - Không cần client/server theo dõi "trạng thái" trao đổi nhiều bước
  - Tất cả các HTTP yêu cầu đều độc lập với nhau
  - Không cần client/server "khôi phục" giao dịch đã thực hiện được một phần nhưng không bao giờ hoàn thành.

Một giao thức có trạng thái: client thực hiện 2 thay đổi với X, hoặc không thực hiện thay đổi nào.



Câu hỏi: Điều gì sẽ xảy ra khi kết nối mạng hoặc máy client bị lỗi tại thời điểm *t'*?

Duy trì trạng thái user-server: cookie

Các trang web và trình duyệt của khách hàng sử dụng cookie để duy trì một số trạng thái giữa các giao dịch.

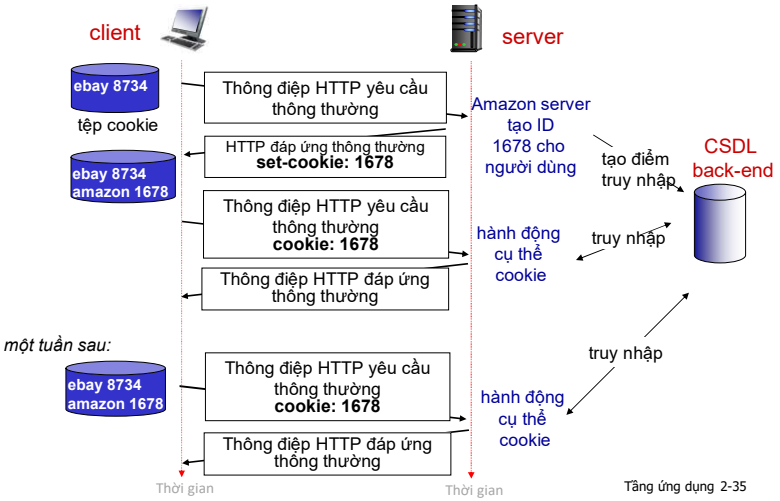
Bốn thành phần:

1) Dòng tiêu đề cookie của thông điệp HTTP *đáp ứng*
2) Dòng tiêu đề cookie trong thông điệp HTTP *yêu cầu* tiếp theo
3) Tệp cookie được lưu giữ trên host của người dùng, được quản lý bởi trình duyệt của người dùng.
4) Cơ sở dữ liệu back-end tại Web site

Ví dụ:

- Susan sử dụng trình duyệt trên laptop, lần đầu tiên truy nhập vào một trang web thương mại điện tử
- Khi yêu cầu HTTP khởi tạo đi đến trang web, trang sẽ tạo ra:
  - Một ID duy nhất (còn gọi là "cookie")
  - Điểm truy nhập vào cơ sở dữ liệu back-end cho ID
- Các HTTP yêu cầu tiếp theo từ Susan tới trang web này sẽ chứa giá trị ID cookie, cho phép trang web "nhận diện" Susan.

Duy trì trạng thái user-server: cookie



HTTP cookie

Cookie có thể được dùng cho những việc gì?

- Cấp phép
- Giỏ mua hàng
- Khuyến nghị
- Trạng thái phiên làm việc của người dùng (Web, e-mail)

Ngoài ra

cookie và sự riêng tư:

- Cookie cho phép các site biết nhiều hơn về người dùng
- Cookie của bên thứ ba (cookie theo dõi) cho phép theo dõi định dang chung (giá trị cookie) trên nhiều trang web.

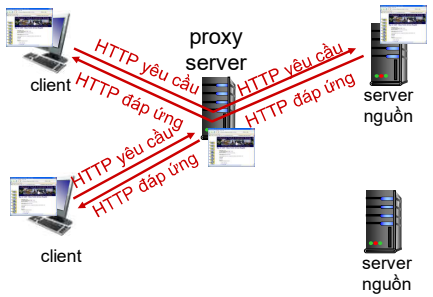
Làm cách nào để lưu "trạng thái":

- *Tại các điểm cuối giao thức*: duy trì trạng thái tại bên gửi/bên nhận thông qua nhiều giao dịch
- *Trong các thông điệp*: cookie trong các thông điệp HTTP mang trạng thái.

Web caches (proxy server)

Mục tiêu: thỏa mãn yêu cầu của client mà không cần liên quan đến server nguồn

- Người dùng thiết lập trình duyệt: truy nhập Web qua vùng nhớ đệm (cache)
- Trình duyệt gửi tất cả các yêu cầu HTTP tới cache
  - Nếu đối tượng có trong cache: cache sẽ trả về đối tượng
  - Ngược lại, cache sẽ yêu cầu đối tượng từ server nguồn, sau đó sẽ trả đối tượng về cho client



Web caches

- Bộ nhớ cache hoạt động như cả client và server
  - Server đáp ứng cho yêu cầu đầu tiên từ một client
  - Client gửi yêu cầu tới server gốc
- Cache thường được cài đặt bởi ISP (trường học, công ty, khu dân cư)

Tại sao lại cần Web caching?

- Giảm thời gian đáp ứng cho yêu cầu của client:
  - Cache gần với client hơn
- Giảm lưu lượng truy nhập vào liên kết của tổ chức
- Nếu Internet thực hiện cache nhiều:
  - thì sẽ cho phép các nhà cung cấp nội dung "nghèo nàn" phân phối nhiều nội dung đó hơn (tương tự như chia sẻ file P2P).

Ví dụ Caching

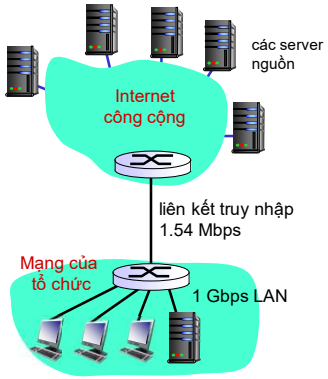
Kịch bản:

- Tốc độ liên kết truy nhập: 1.54 Mbps
- RTT từ bộ định tuyến của tổ chức đến server: 2 giây
- Kích thước đối tượng web: 100K bits
- Tốc độ yêu cầu trung bình từ trình duyệt đến server gốc: 15/giây
  - tốc độ dữ liệu trung bình tới trình duyệt: 1.50 Mbps

Vấn đề: trễ xếp hàng rất lớn khi mức độ sử dụng cao!

Hiệu suất:

- Việc sử dụng liên kết truy nhập = .97
- Việc sử dụng LAN: .0015
- trễ đầu cuối-đầu cuối
  - = trễ Internet + trễ liên kết truy nhập + trễ LAN
  - = 2 giây + một số phút + (~1 giây)



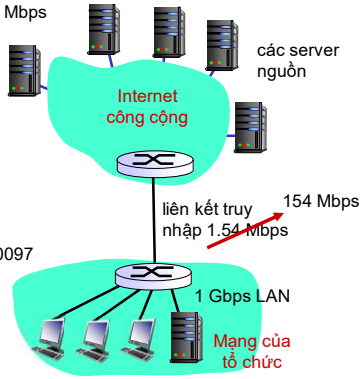
Giải pháp 1: mua liên kết truy nhập tốc độ cao hơn

Kịch bản :

- Tốc độ liên kết truy nhập: 1.54 Mbps → 154 Mbps
- RTT từ bộ định tuyến của tổ chức đến server: 2 giây
- Kích thước đối tượng web: 100K bits
- Tốc độ yêu cầu trung bình từ trình duyệt đến server gốc: 15/giây
  - tốc độ dữ liệu trung bình tới trình duyệt: 1.50 Mbps

Hiệu suất:

- Việc sử dụng liên kết truy nhập = 0.97 → 0.0097
- Việc sử dụng LAN: .0015
- trễ đầu cuối-đầu cuối
  - = trễ Internet + trễ liên kết truy nhập + trễ LAN
  - = 2 giây + một số phút + (~1 giây)
  - msecs



Chi phí: liên kết truy nhập tốc độ cao hơn (đắt đỏ!)

Giải pháp 2: cài đặt web cache

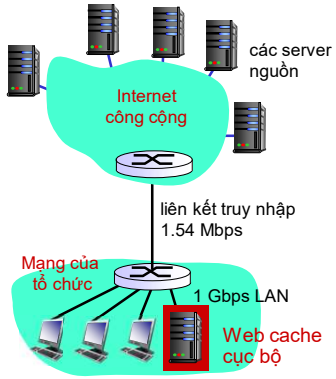
Kịch bản:

- Tốc độ liên kết truy nhập: 1.54 Mbps
- RTT từ bộ định tuyến của tổ chức đến server: 2 giây
- Kích thước đối tượng web: 100K bits
- Tốc độ yêu cầu trung bình từ trình duyệt đến server gốc: 15/giây
  - tốc độ dữ liệu trung bình tới trình duyệt: 1.50 Mbps

Chi phí: web cache (rẻ!)

Hiệu suất:

- Việc sử dụng LAN: ?
- Việc sử dụng liên kết truy nhập = ?
- trễ trung bình đầu cuối-đầu cuối = ?

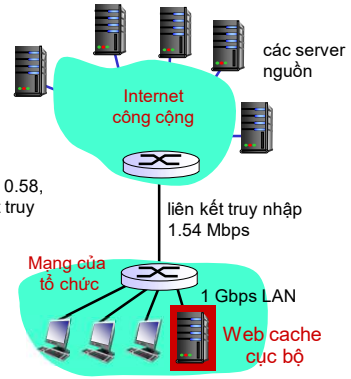


Tính toán việc sử dụng liên kết, độ trễ như thế nào?

Tính toán việc sử dụng liên kết truy nhập, trễ đầu cuối-đầu cuối với bộ đệm (cache)

Giả sử tỷ lệ hỗ trợ của cache là 0.4:

- 40% yêu cầu được cung cấp bởi cache, với độ trễ thấp (msec)
- 60% yêu cầu được cung cấp bởi server nguồn
  - tỷ lệ tới trình duyệt qua liên kết truy nhập = 0.6 * 1.50 Mbps = 0.9 Mbps
  - Việc sử dụng liên kết truy nhập = 0.9/1.54 = 0.58, nghĩa là trễ xếp hàng thấp (msec) tại liên kết truy nhập.
- Trễ trung bình đầu cuối-đầu cuối:
  = 0.6 * (trễ từ các server gốc)
  + 0.4 * (trễ khi được cung cấp tại cache)
  = 0.6 * (2.01) + 0.4 (~msecs) = ~ 1.2 secs



Trễ đầu cuối-đầu cuối trung bình thấp hơn so với liên kết 154 Mbps (và cũng rẻ hơn!)

Trình duyệt đệm: GET có điều kiện

- *Mục tiêu:* không gửi đối tượng nếu trình duyệt đã có phiên bản được cập nhật trong bộ nhớ đệm
  - Không có trễ truyền đối tượng (hoặc sử dụng tài nguyên mạng).
- *Client:* xác định ngày của bản sao được lưu trong bộ nhớ đệm trong HTTP yêu cầu
  If-modified-since: <date>
- *server:* đáp ứng không chứa đối tượng nếu bản sao được lưu trong bộ nhớ đệm của trình duyệt đã được cập nhật:
  HTTP/1.0 304 Not Modified