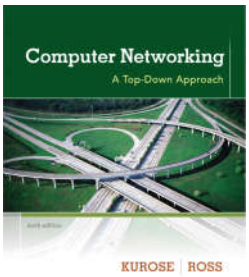
Chương 2
Tầng ứng dụng



Computer Networking: A Top Down Approach
7th edition
Jim Kurose, Keith Ross
Addison-Wesley
March 2017

Người dịch: Nguyễn Thanh Thủy

Tài liệu được dịch cho mục đích giảng dạy (được sự đồng ý của tác giả).

Chương 2: Tầng ứng dụng

Mục tiêu:

- Khái niệm, các vấn đề cài đặt giao thức ứng dụng mạng
  - Các mô hình dịch vụ tầng giao vận
  - Mô hình khách-chủ (client-server)
  - Mô hình điểm-điểm (peer-to-peer)
- Nghiên cứu một số giao thức tầng ứng dụng
  - HTTP
  - FTP
  - SMTP / POP3 / IMAP
  - DNS
- Tạo một ứng dụng mạng
  - socket API

Chương 2: Nội dung

2.1 Nguyên lý của ứng dụng mạng
- Kiến trúc của ứng dụng
- Các yêu cầu của ứng dụng

2.2 Web và HTTP

2.3 FTP

2.4 Thư điện tử
- SMTP, POP3, IMAP

2.5 DNS

2.6 Ứng dụng P2P

2.7 Lập trình socket với UDP và TCP

Một số ứng dụng mạng

- Thư điện tử (e-mail)
- web
- Tin nhắn văn bản (text messaging)
- Truy nhập từ xa (remote login)
- Chia sẻ file P2P
- Trò chơi nhiều người trên mạng
- streaming video (YouTube, Hulu, Netflix)
- Điện thoại Internet (voice over IP) (ví dụ Skype)
- Hội thảo video thời gian thực
- Mạng xã hội
- Các ứng dụng tìm kiếm
- …
- …

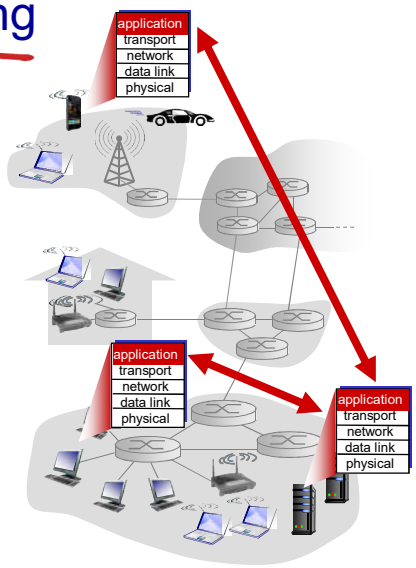
Tạo một ứng dụng mạng

Viết chương trình để:

- Chạy trên các *hệ thống đầu cuối* khác nhau
- Truyền thông qua mạng
- Ví dụ: phần mềm máy chủ web (web server) truyền thông với phần mềm trình duyệt (browser software)

Không cần viết chương trình ứng dụng cho các thiết bị trong phần lõi của mạng

- Các thiết bị trong phần lõi của mạng không chạy các ứng dụng người dùng.
- Các ứng dụng chạy trên thiết bị đầu cuối cho phép phát triển và phổ biến ứng dụng một cách nhanh chóng.

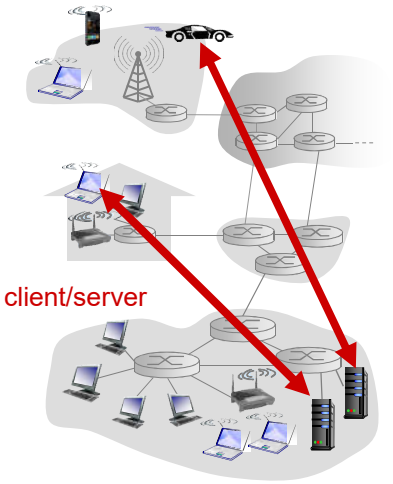


Kiến trúc của ứng dụng

Các ứng dụng có thể có kiến trúc dạng:

- Client-server (khách-chủ)
- Peer-to-peer (P2P, ngang hàng)

Kiến trúc client-server



server:

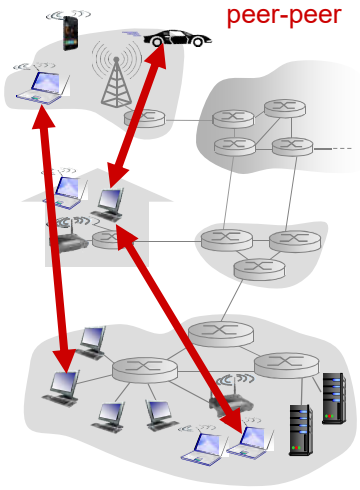
- Là host luôn hoạt động
- Có địa chỉ IP cố định
- Các trung tâm dữ liệu

client:

- Truyền thông với server
- Có thể được kết nối liên tục vào mạng hoặc không
- Có thể có địa chỉ IP thay đổi
- Không truyền thông trực tiếp với client khác

Kiến trúc P2P

- *Không có* server luôn hoạt động
- Các hệ thống đầu cuối (peer) truyền thông trực tiếp với nhau
- Mỗi peer yêu cầu dịch vụ từ một peer nào đó, và cung cấp dịch vụ lại cho các peer khác.
  - *Có khả năng tự mở rộng* – peer mới mang lại khả năng dịch vụ mới, cũng như có những yêu cầu dịch vụ mới.
- Các peer không kết nối liên tục và có thể thay đổi địa chỉ IP
  - Quản lý phức tạp



Tiến trình truyền thông

Tiến trình: chương trình chạy trên một host

- Trên cùng một host, hai tiến trình truyền thông với nhau qua truyền thông tiến trình nội bộ (được xác định bởi hệ điều hành)
- Các tiến trình trên các host khác nhau truyền thông với nhau bằng cách trao đổi các thông điệp

client, server

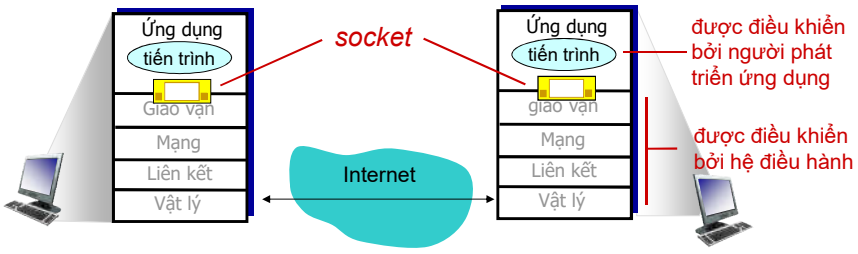
Tiến trình client: tiến trình khởi tạo truyền thông

Tiến trình server: tiến trình chờ được tiếp xúc

- Lưu ý: ứng dụng theo kiến trúc P2P có cả tiến trình client & tiến trình server

Socket

- Các tiến trình gửi/nhận các thông điệp đến/từ socket của nó
- Socket tương tự như một cửa vào/ra
  - Tiến trình gửi đẩy thông điệp ra bên ngoài cửa
  - Tiến trình gửi dựa trên cơ sở hạ tầng tầng giao vận ở phía bên kia cửa để truyền thông điệp đến socket của tiến trình nhận



Định địa chỉ cho tiến trình

- Để nhận các thông điệp, tiến trình phải có *định danh (identifier)*
- Thiết bị host phải có địa chỉ IP 32-bit duy nhất
- *Hỏi:* Địa chỉ IP của host mà trên đó tiến trình chạy có đủ để xác định định danh của tiến trình không?
  - *Trả lời:* Không, *do nhiều* tiến trình có thể cùng chạy trên một host.

- *Định danh* bao gồm cả địa chỉ IP và số hiệu cổng được kết hợp với tiến tình trên host.
- Ví dụ các số hiệu cổng:
  - HTTP server: 80
  - mail server: 25
- Để gửi thông điệp HTTP tới web server gaia.cs.umass.edu:
  - Địa chỉ IP: 128.119.245.12
  - Số hiệu cổng: 80

Các giao thức tầng ứng dụng

- Các loại thông điệp được trao đổi
  - Ví dụ: thông điệp yêu cầu (request), thông điệp đáp ứng (response)
- Cú pháp của thông điệp:
  - Các trường trong thông điệp & mô tả
- Ngữ nghĩa của thông điệp
  - Ý nghĩa thông tin của các trường
- Quy tắc về thời gian và cách thức các tiến trình gửi và đáp ứng thông điệp

Các giao thức mở (công khai):

- Được định nghĩa trong RFCs
- Cho phép tương tác
- Ví dụ: HTTP, SMTP

Các giao thức riêng (độc quyền):

- Ví dụ: Skype

Một ứng dụng cần những dịch vụ giao vận nào?

Toàn vẹn dữ liệu

- Một số ứng dụng (ví dụ: truyền file, web) yêu cầu truyền tin cậy 100%
- Các ứng dụng khác (ví dụ: audio) có thể chịu một số mất mát

Thời gian thực

- Một số ứng dụng (ví dụ: điện thoại Internet, trò chơi tương tác) yêu cầu độ trễ thấp mới "hiệu quả"

Băng thông

- Một số ứng dụng (ví dụ: đa phương tiện) yêu cầu băng thông tối thiểu để đảm bảo "hiệu quả".
- Các ứng dụng khác ("các ứng dụng mềm dẻo") có thể dùng băng thông nào cũng được

An toàn bảo mật

- Mã hóa, đảm bảo toàn vẹn dữ liệu, …

Một số yêu cầu dịch vụ giao vận với các ứng dụng phổ biến

Ứng dụng	Mất mát dữ liệu	Thông lượng	Thời gian thực
Truyền file	không	mềm dẻo	không
Thư điện tử	không	mềm dẻo	không
Web	không	mềm dẻo	không
Audio/video thời gian thực	chịu lỗi	audio: 5kbps-1Mbps video:10kbps-5Mbps	có, 100 msec
Lưu trữ audio/video	chịu lỗi	như trên	có, vài giây
Trò chơi tương tác	chịu lỗi	một vài kbps	có, 100 msec
Tin nhắn nhanh	không	mềm dẻo	có và không

Dịch vụ giao thức tầng giao vận của Internet

Dịch vụ TCP:

- *Truyền tin cậy* giữa tiến trình gửi và tiến trình nhận
- *Điều khiển luồng:* bên gửi không lấn át bên nhận
- *Điều khiển tắc nghẽn:* điều tiết bên gửi khi mạng bị quá tải
- *Không cung cấp:* thời gian thực, đảm bảo băng thông tối thiểu, an toàn bảo mật
- *Hướng kết nối:* yêu cầu thiết lập kết nối giữa tiến trình client và tiến trình server

Dịch vụ UDP:

- *Truyền dữ liệu không tin cậy* giữa tiến trình gửi và tiến trình nhận
- *Không cung cấp:* truyền tin cậy, điều khiển luồng, điều khiển tắc nghẽn, thời gian thực, đảm bảo băng thông, an toàn bảo mật, hoặc thiết lập kết nối

Hỏi: Tại sao lại dùng cả hai dịch vụ? Dùng UDP để làm gì?

Các ứng dụng Internet: các giao thức tầng ứng dụng và giao vận

Ứng dụng	Giao thức tầng ứng dụng	Giao thức tầng giao vận
Thư điện tử	SMTP [RFC 2821]	TCP
Truy nhập từ xa	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
Truyền file	FTP [RFC 959]	TCP
Streaming multimedia	HTTP (ví dụ: YouTube), RTP [RFC 1889]	TCP hoặc UDP
Điện thoại Internet	SIP, RTP, giao thức độc quyền (ví dụ: Skype)	TCP hoặc UDP

Bảo mật TCP

TCP & UDP

- Không mã hóa
- Mật khẩu dạng bản rõ được gửi vào socket để truyền trên Internet theo dạng bản rõ

SSL (Secure Sockets Layer)

- Hỗ trợ kết nối TCP được mã hóa
- Toàn vẹn dữ liệu
- Xác thực thiết bị đầu cuối

SSL là bảo mật tại tầng ứng dụng

- Ứng dụng dùng các thư viện của SSL để "nói" với TCP

Socket API của SSL

- Mật khẩu dạng bản rõ được gửi vào socket để truyền trên Internet theo dạng đã được mã hóa

Chương 2: Nội dung

2.1 Nguyên lý của ứng dụng mạng
- Kiến trúc của ứng dụng
- Các yêu cầu của ứng dụng

2.2 Web và HTTP

2.3 FTP

2.4 Thư điện tử
- SMTP, POP3, IMAP

2.5 DNS

2.6 Ứng dụng P2P

2.7 Lập trình socket với UDP và TCP

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

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 hòa giải

Kết nối HTTP

HTTP không bền vững

- Chỉ có tối đa một đối tượng được gửi qua một kết nối TCP
  - Kết nối sau đó sẽ được đóng lại
- Việc tải về nhiều đối tượng sẽ yêu cầu nhiều kết nối

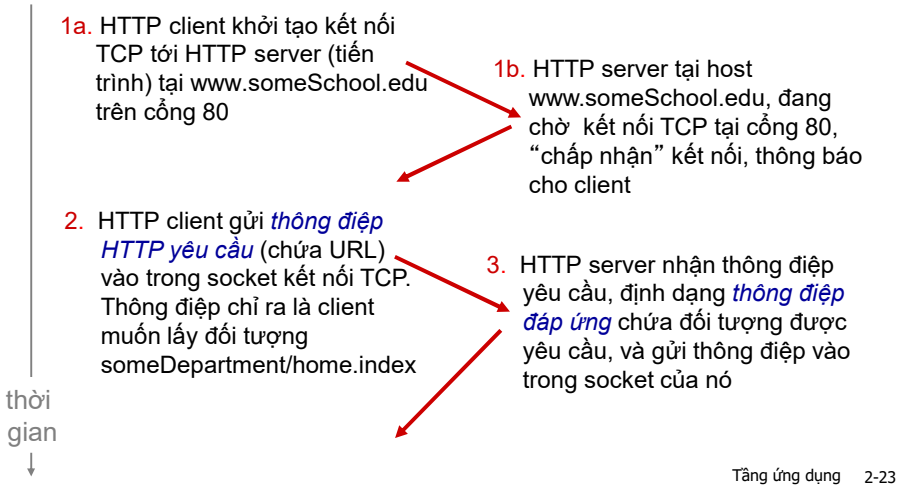
HTTP bền vững

- Nhiều đối tượng có thể được gửi qua một kết nối TCP duy nhất giữa client và server

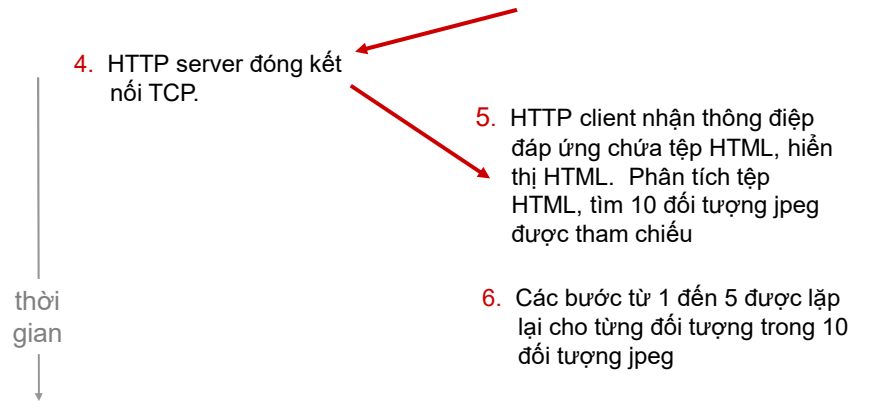
HTTP không bền vững

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

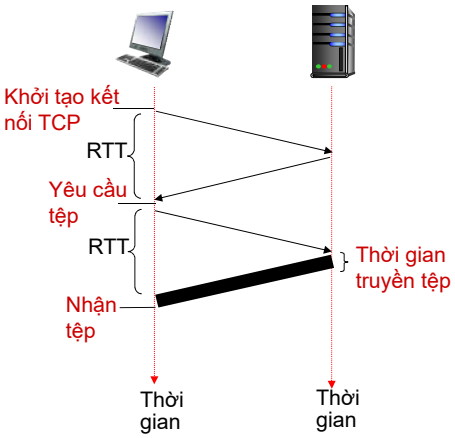


HTTP không bền vững: thời gian đáp ứng

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



HTTP bền vững

Vấn đề với HTTP không bền vững:

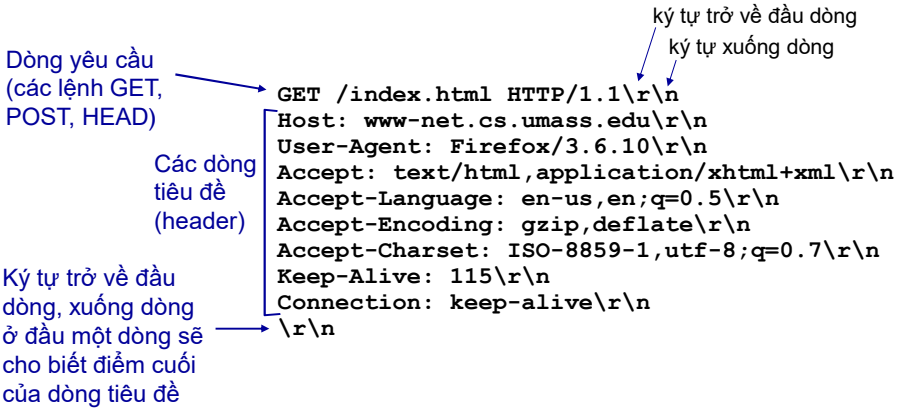
- Cần 2 RTT cho mỗi đối tượng
- Hệ điều hành liên quan đến *mỗi* kết nối TCP
- Các trình duyệt thường mở nhiều kết nối TCP song song để lấy các đối tượng được tham chiếu

HTTP bền vững:

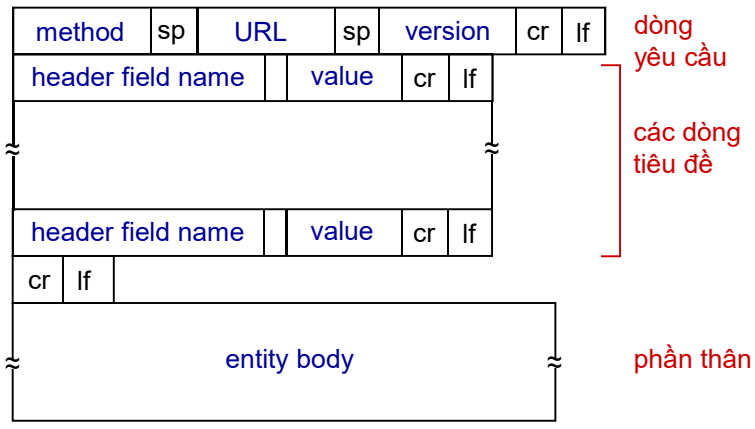
- Server để mở kết nối sau khi gửi đáp ứng
- Chuỗi các thông điệp HTTP tiếp theo giữa cùng client/server sẽ được gửi thông qua kết nối mở này
- Client gửi yêu cầu ngay sau khi gặp một đối tượng được tham chiếu
- Ít nhất là một RTT cho tất cả các đối tượng được tham chiếu

Thông điệp yêu cầu HTTP

- 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



Tải lên form input

Phương pháp POST:

- Trang web thường chứa form input
- Đầu vào (input) được tải lên server trong phần thân thực thể (entity body)

Phương pháp URL:

- Dùng phương thức GET
- Đầu vào được tải lên trong trường URL của dòng yêu cầu:

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


Các loại phương thức

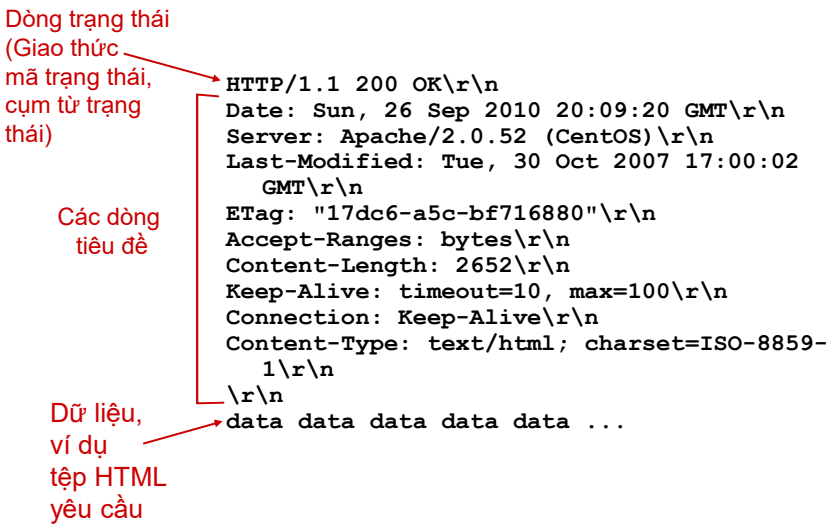
HTTP/1.0:

- GET
- POST
- HEAD
  - Yêu cầu server dời đối tượng được yêu cầu ra khỏi đáp ứng

HTTP/1.1:

- GET, POST, HEAD
- PUT
  - Tải tệp lên trong phần thân thực thể tới đường dẫn được xác định trong trường URL
- DELETE
  - Xóa tệp được xác định trong trường URL

Thông điệp đáp ứng HTTP



Các mã trạng thái đáp ứng HTTP

- 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:)

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 /~ross/ HTTP/1.1
Host: cis.poly.edu
```

Bằng cách gõ lệnh này (nhấn enter 2 lần), bạn đã 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 yêu cầu/đáp ứng HTTP bắt được)

Trạng thái User-server: các cookie

Nhiều trang Web sử dụng các cookie

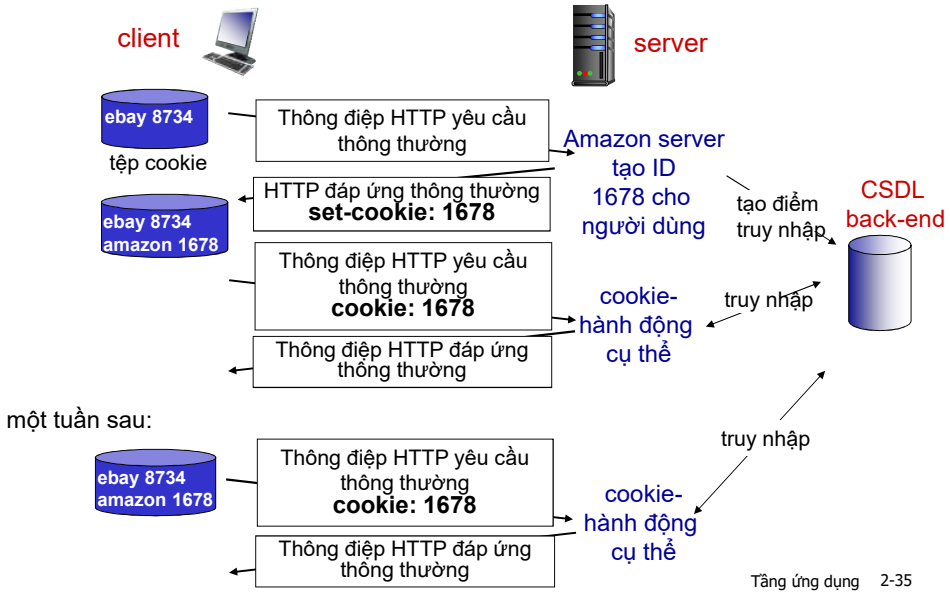
Bốn thành phần:

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 thường truy nhập Internet từ một PC
- Cô ấy vào một trang thương mại điện tử lần đầu tiên
- Khi yêu cầu HTTP khởi tạo đi đến trang, trang sẽ tạo ra:
  - Một ID duy nhất
  - Điểm truy nhập vào cơ sở dữ liệu back-end cho ID

Các cookie: lưu giữ "trạng thái" (tiếp)



Các cookie (tiếp)

Các 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
- Người dùng có thể cung cấp tên và e-mail cho các site

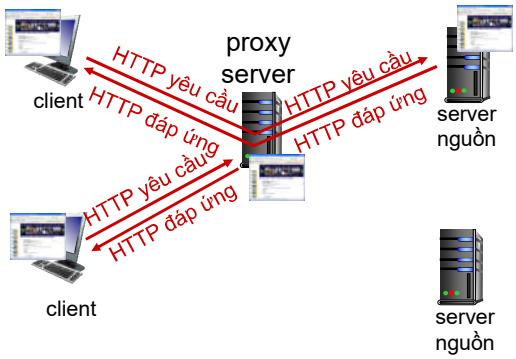
Làm cách nào để lưu giữ "trạng thái":

- Các điểm cuối giao thức: duy trì trạng thái tại bên gửi/bên nhận thông qua nhiều giao dịch
- Các cookie: 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 caching

- Bộ nhớ cache hoạt động ở 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?

- Làm giảm thời gian đáp ứng cho yêu cầu của client
- Làm giảm lưu lượng trên một liên kết truy nhập của tổ chức
- Nếu Internet thực hiện cache nhiều thì sẽ làm cho các nhà cung cấp nội dung "nghèo nàn" phân phối nhiều nội dung đó (cũng tương tự như chia sẻ file P2P).

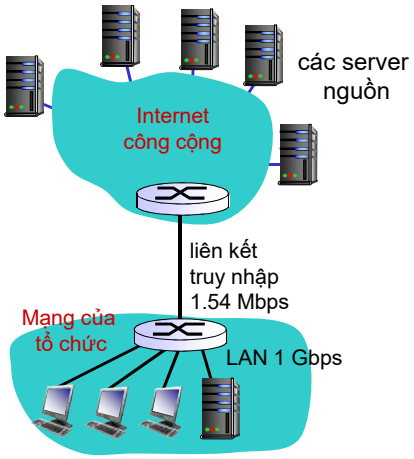
Ví dụ caching

Giả thiết:

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

Kết quả:

- Độ khả dụng của LAN: 15%
- Độ khả dụng của liên kết truy nhập = 99% *Vấn đề!*
- Trễ tổng = trễ của Internet + trễ truy nhập + trễ của LAN
  = 2 giây + một số phút + (~ 1 giây)



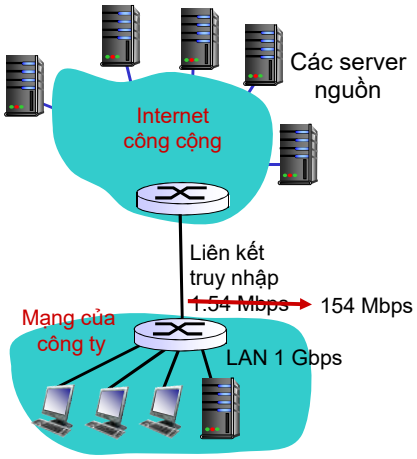
Ví dụ caching: tăng tốc độ của liên kết truy nhập lên

Giả thiết:

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

Kết quả:

- Độ khả dụng của LAN: 15%
- Độ khả dụng của liên kết truy nhập = ~~99%~~ **9.9%**
- Tổng trễ = trễ của Internet + trễ truy nhập + trễ của LAN
  = 2 giây + ~~một số phút~~ **Một số giây** + (~ 1 giây)



Chi phí: **để làm tăng tốc độ liên kết truy nhập (không hề rẻ!)**

Ví dụ caching: cài đặt cache cục bộ

Giả thiết:

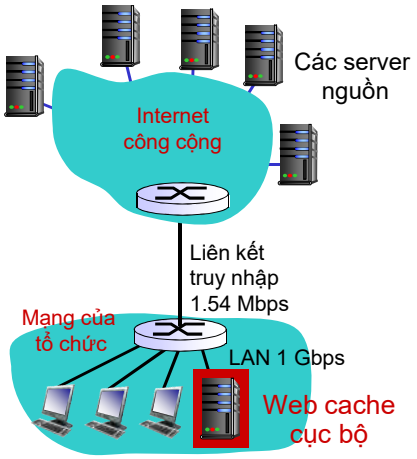
- Kích thước của đối tượng trung bình: 100K bits
- Tốc độ yêu cầu trung bình từ các trình duyệt tới server nguồn: 15/giây
- Tốc độ dữ liệu trung bình tới trình duyệt: 1.50 Mbps
- RTT từ bộ định tuyến của tổ chức đến server nguồn bất kỳ: 2 giây
- Tốc độ của liên kết truy nhập: 1.54 Mbps

Kết quả:

- Độ khả dụng của LAN: 15%
- Độ khả dụng của liên kết truy nhập = ?
- Tổng trễ = ?

Tính độ khả dụng của liên kết và trễ như thế nào?

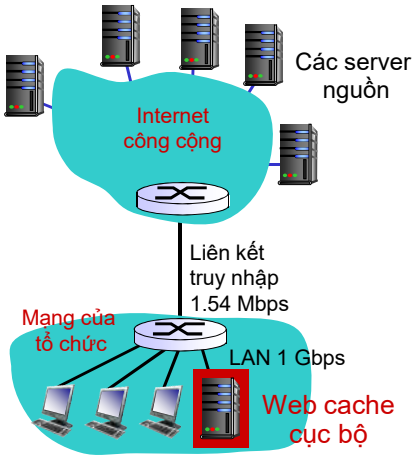
Chi phí: **web cache (rất rẻ!)**



Ví dụ caching: cài đặt cache cục bộ

Tính toán độ khả dụng của liên kết truy nhập, độ trễ khi dùng cache:

- Giả sử tỷ lệ hỗ trợ của cache là 0.4
  - 40% yêu cầu được thỏa mãn tại cache, 60% yêu cầu được thỏa mãn tại nguồn
- Độ khả dụng của liên kết:
  - 60% yêu cầu dùng liên kết truy nhập
- Tốc độ dữ liệu tới các trình duyệt qua liên kết truy nhập = 0.6*1.50 Mbps = 0.9 Mbps
  - Độ khả dụng = 0.9/1.54 = 0.58
- Tổng trễ:
  - = 0.6 * (trễ từ các server nguồn) +0.4 * (trễ khi được thỏa mãn tại cache)
    = 0.6 * (2.01s) + 0.4 * (0.01s)
    = 1.21s
  - Ít hơn so với liên kết 154 Mbps (và chi phí rẻ hơn rất nhiều!)



GET có điều kiện

- *Mục tiêu:* không gửi đối tượng nếu cache đã cập nhật
  - Không có trễ truyền đối tượng
  - Độ khả dụng của liên kết thấp hơn
- *cache:* xác định ngày của bản sao cache trong yêu cầu HTTP

  If-modified-since: <date>
- *server:* đáp ứng không chứa đối tượng nếu bản sao cach đã được cập nhật:

  HTTP/1.0 304 Not Modified