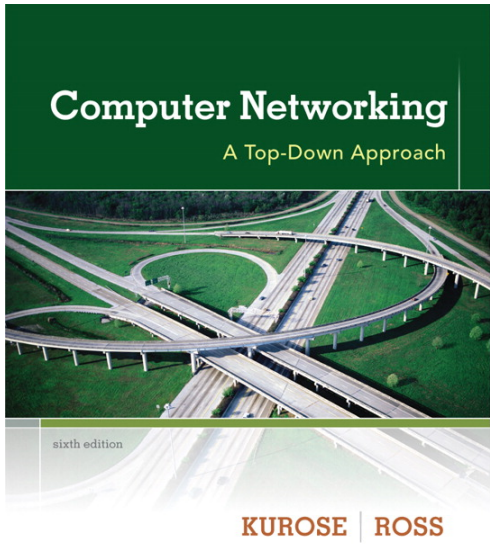
Chương 2
Tầng Application



A note on the use of these ppt slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK/KWR

Computer Networking: A Top Down Approach
6th edition
Jim Kurose, Keith Ross
Addison-Wesley
March 2012

Chương 2: Nội dung

2.1 Các nguyên lý của các ứng dụng mạng

2.2 Web và HTTP

2.3 FTP

2.4 E-Mail

- SMTP, POP3,IMAP

2.5 DNS

2.6 Các ứng dụng P2P

2.7 Lập trình socket với UDP và TCP

Chương 2: tầng application

Mục tiêu:

- Khái niệm và các khía cạnh thực hiện của các giao thức ở ứng dụng mạng
  - Mô hình client-server
  - Mô hình peer-to-peer
- Tìm hiểu các giao thức phổ biến của tầng application
  - HTTP
  - FTP
  - SMTP / POP3 / IMAP
  - DNS
- Lập trình ứng dụng mạng
  - socket API

Một số ứng dụng mạng

- E-mail
- Web
- Remote access
- Chia sẽ file P2P
- Trò chơi nhiều người trên mạng
- Streaming stored video (YouTube, Hulu, Netflix)
- Thoại trên nền IP (e.g., Skype)
- Hội thảo video thời gian thực
- Mạng xã hội
- Mạng tìm kiếm
- …
- …

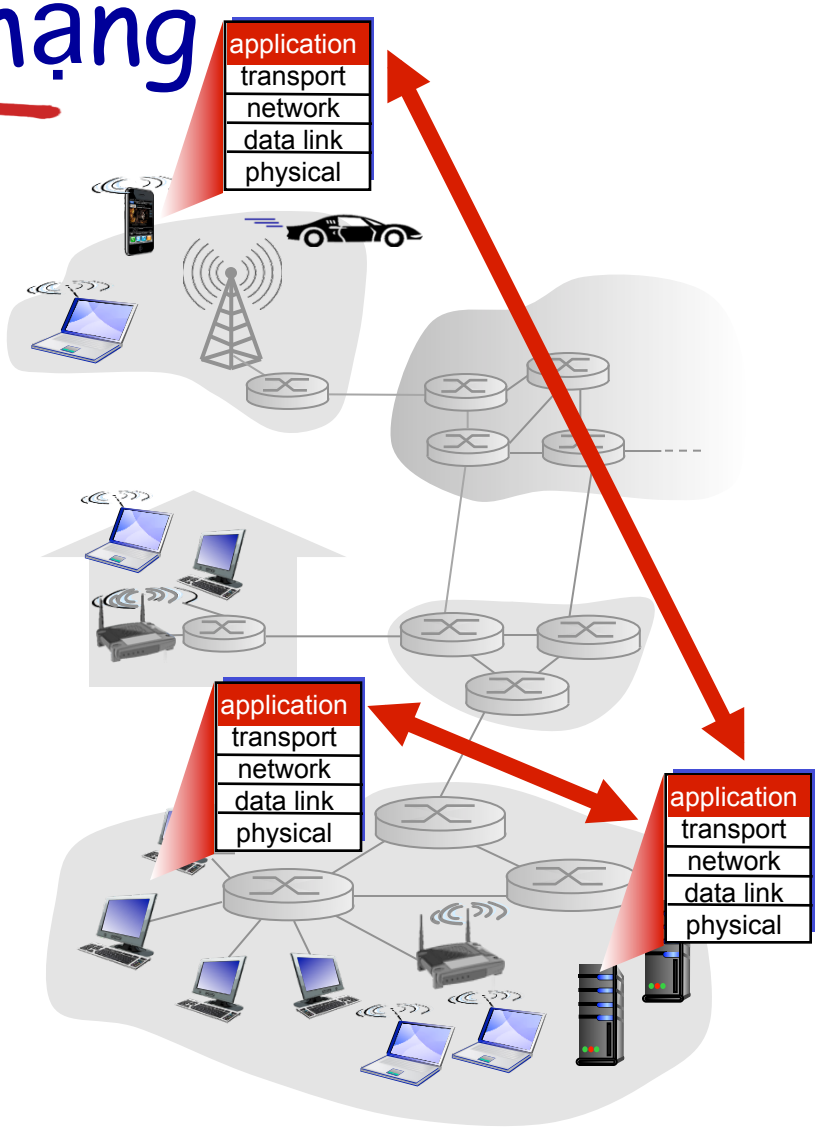
Tạo một ứng dụng mạng

Viết chương trình để:

- Chạy trên các hệ thống đầu cuối khác nhau
- Truyền thông qua mạng
- Ví du: phần mềm web server giao tiếp với trình duyệt

Không cần viết phần mềm cho các thiết bị mạng lõi

- Các thiết bị mạng lõi không chạy các ứng dụng của người dùng
- Các ứng dụng trên các hệ thống đầu cuối cho phép quảng bá và phát triển ứng dụng nhanh chóng

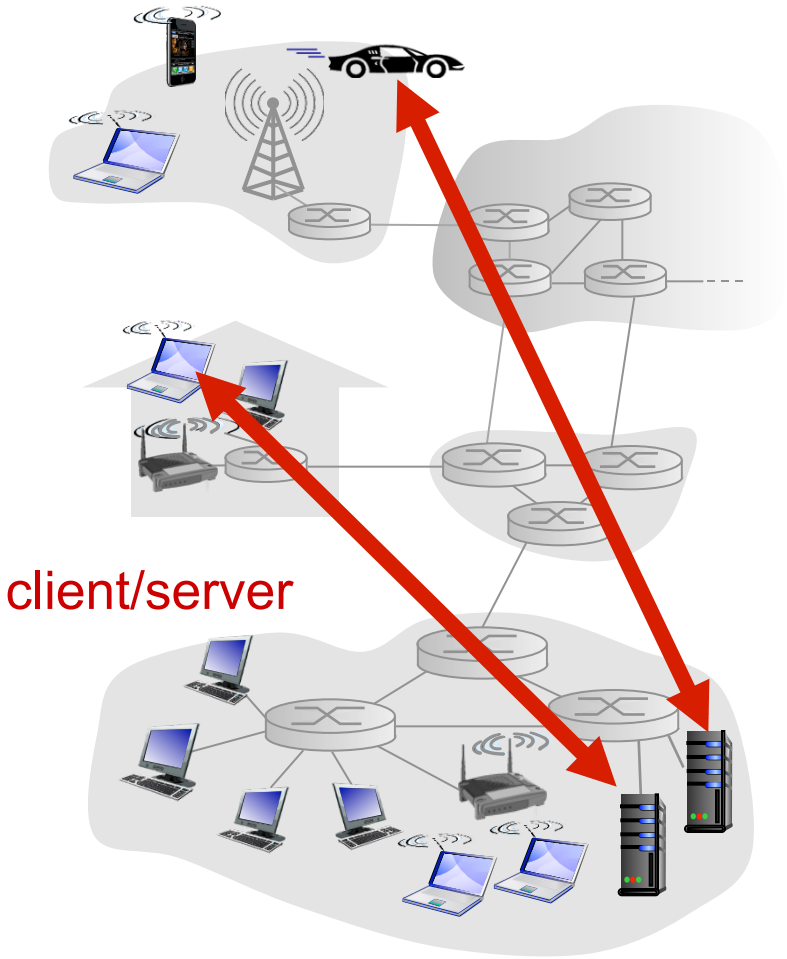


Các kiến trúc ứng dụng

Cấu trúc có thể của các ứng dụng:

- Client-Server
- peer-to-peer (P2P) (Mạng ngang hàng)
- Lai giữa Client-Server và P2P

Kiến trúc Client-server



server:

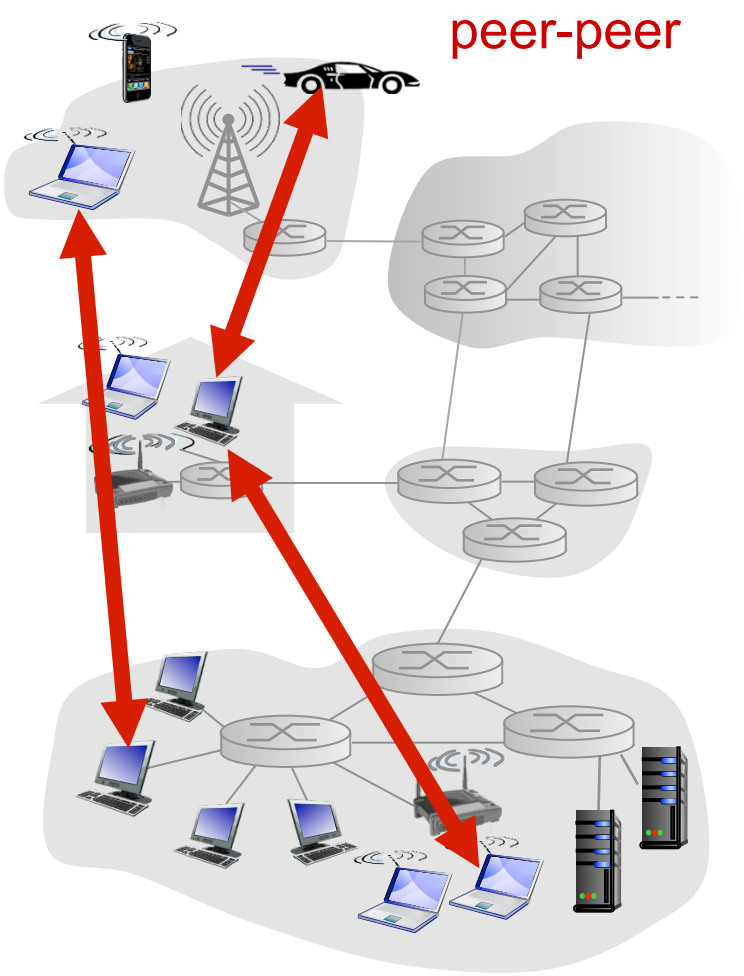
- Luôn luôn hoạt động
- Địa chỉ IP cố định
- Trung tâm phục vụ và lưu trữ dữ liệu

clients:

- Giao tiếp với server
- Có thể kết nối không liên tục
- Có thể dùng địa chỉ IP động
- Không giao tiếp trực tiếp với các client khác

Kiến trúc P2P (ngang hàng)

- *Không có* server
- Các hệ thống đầu cuối bất kỳ truyền thông trực tiếp với nhau
- Các peer yêu cầu dịch vụ từ các peer khác và cung cấp dịch vụ ngược lại cho các peer khác
  - *Có khả năng tự mở rộng* - các peer mới mang lại năng lực dịch vụ mới, cũng như các nhu cầu về dịch vụ mới
- Các peer được kết nối không liên tục và có thể thay đổi địa chỉ IP
  - Quản lý phức tạp



Các tiến trình liên lạc

Tiến trình (process): chương trình chạy trong một host.

- Trong cùng một host, hai tiến trình giao tiếp với nhau bằng cách sử dụng truyền thông liên tiến trình (inter-process communication) được định nghĩa bởi hệ điều hành.
- Các tiến trình trong các host khác nhau truyền thông với nhau bởi trao đổi các thông điệp (message)

clients, servers

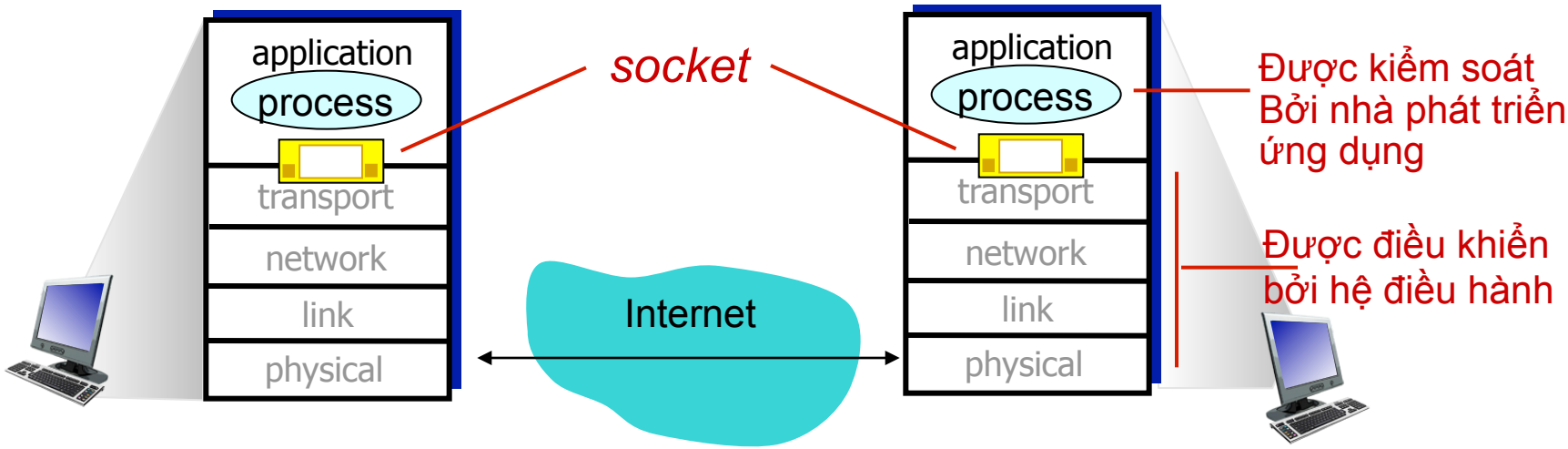
Tiến trình client: tiến trình khởi tạo truyền thông

Tiến trình server: tiến trình chờ đợi để được liên lạc

- Chú ý: các ứng dụng với kiến trúc P2P có cả các tiến trình client và server

Sockets

- Tiến trình gửi/nhận thông điệp đến/ từ socket của nó
- Socket tương tự như cổng ra vào
  - Tiến trình gởi đẩy thông điệp ra khỏi cửa
  - Tiến trình gởi dựa trên cổng của hạ tầng truyền thông bên kia để phân phối thông điệp đến socket tại tiến trình nhận



Tiến trình định địa chỉ

- Để nhận thông điệp, tiến trình phải có *định danh*
- Thiết bị host device có địa chỉ IP 32-bit duy nhất
- *Q:* dựa vào địa chỉ IP của host mà tiến trình đang chạy trên đó thì có đủ để xác định tiến trình đó hay không?
  - *A:* không, có nhiều tiến trình có thể đang được chạy trên cùng một host

- *Định danh (identifier)* bao gồm cả địa chỉ IP và số cổng (port numbers) được liên kết với tiến trình trên host.
- Ví dụ về số port:
  - HTTP server: 80
  - mail server: 25
- Để gởi thông điệp HTTP đến web server gaia.cs.umass.edu :
  - IP address: 128.119.245.12
  - port number: 80
- Còn nữa...

Định nghĩa giao thức tầng Application

- Các loại thông điệp được trao đổi
  - e.g., yêu cầu (request), đáp ứng (response)
- Cú pháp thông điệp:
  - Các trường trong thông điệp và cách mà các trường được định nghĩa
- Ngữ nghĩa của thông điệp
  - Ý nghĩa của thông tin trong các trường
- Các quy tắc (rules) khi nào và cách nào mà các tiến trình gởi và đáp ứng các thông điệp

Các giao thức mở:

- Được định nghĩa trong RFCs
- Cho phép khả năng tương tác
- e.g., HTTP, SMTP

Các giao thức độc quyền:

- e.g., Skype

Dịch vụ vận chuyển nào mà ứng dụng cần?

Khả năng mất mát dữ liệu (data loss)

- Một số ứng dụng (ví dụ truyền file, web transactions) yêu cầu độ tin cậy 100% khi truyền dữ liệu.
- Các ứng dụng khác (ví dụ audio) có thể chịu được một số mất mát.

Thời gian (timing)

- Một số ứng dụng (ví dụ, thoại Internet, game tương tác) yêu cầu độ trễ thấp để đạt được "hiệu quả"

Thông lượng (throughput)

- Một số ứng dụng (như là, đa phương tiện) yêu cầu số lượng thông lượng tối thiểu để đạt được "hiệu quả"
- Các ứng dụng khác ("ứng dụng mềm dẻo") có thể dùng bất kỳ thông lượng nào cũng được

An ninh

- Mã hóa, toàn vẹn dữ liệu, …

Các yêu cầu dịch vụ vận chuyển: các ứng dụng phổ biến

ứng dụng	mất dữ liệu	thông lượng	time sensitive
Truyền file	không	mềm dẻo	không
e-mail	không	mềm dẻo	không
Web documents	không	mềm dẻo	không
audio/video thời gian thực	chịu lỗi	audio: 5kbps-1Mbps video:10kbps-5Mbps	có, 100’s msec
audio/video đã lưu	chịu lỗi	như trên	có, vài giây
Game tương tác	chịu lỗi	Trên một vài kbps	Có , 100’s msec
text messaging	không	mềm dẻo	Có và không

Các dịch vụ giao thức Transport Internet

Dịch vụ TCP:

- *reliable transport* giữa tiến trình gởi và nhận
- *flow control:* người gởi sẽ không áp đảo người nhận
- *congestion control:* điều tiết người gởi khi mạng quá tải
- *connection-oriented:* thiết lập được yêu cầu giữa tiến trình client và server

Dịch vụ UDP:

- *Truyền dữ liệu không tin cậy* giữa tiến trình gởi và nhận
- *Không hỗ trợ:* độ tin cậy, điều khiển luồng, điều khiển tắt nghẽn, bảo đảm thông lượng, bảo mật, và thiết lập kết nối.

Q: tại sao phải quan tâm? Tại sao có UDP?

Ứng dụng Internet: Các giao thức tầng application, transport

application	Giao thức tầng application	Giao thức dưới tầng transport
e-mail	SMTP [RFC 2821]	TCP
remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
Truyền file	FTP [RFC 959]	TCP
streaming multimedia	HTTP (e.g., YouTube), RTP [RFC 1889]	TCP or UDP
Thoại Internet	SIP, RTP, độc quyền (e.g., Skype)	TCP or UDP

Bảo mật TCP

TCP & UDP

- Không mã hóa
- Mật mã chưa mã hóa được gởi đến socket để đi qua Internet trong dạng nguyên bản

SSL

- Hổ trợ kết nối TCP được mã hóa
- Toàn vẹn dữ liệu
- Chứng thực điểm cuối

SSL là giao thức ở tầng Application

- Các ứng dụng dùng thư viện SSL, cái mà "nói chuyện" với TCP

SSL socket API

- Mật mã dạng cleartext được gởi vào trong socket đi qua Internet được mã hóa
- Xem chương 7

Chương 2: Nội dung

2.1 Các nguyên lý của các ứng dụng mạng

2.2 Web và HTTP

2.3 FTP

2.4 thư điện tử

- SMTP, POP3, IMAP

2.5 DNS

2.6 các ứng dụng P2P

2.7 lập trình socket với UDP và TCP

Web và HTTP

Ôn lại...

- *web page* bao gồm các đối tượng (*objects*)
- Đối tượng có thể là file HTML, hình ảnh JPEG, Java applet, file audio,...
- Web page bao gồm file HTML cơ bản cái mà bao gồm một số đối tượng được tham chiếu
- Mỗi đối tượng có thể được định địa chỉ bởi một *URL,* ví dụ

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

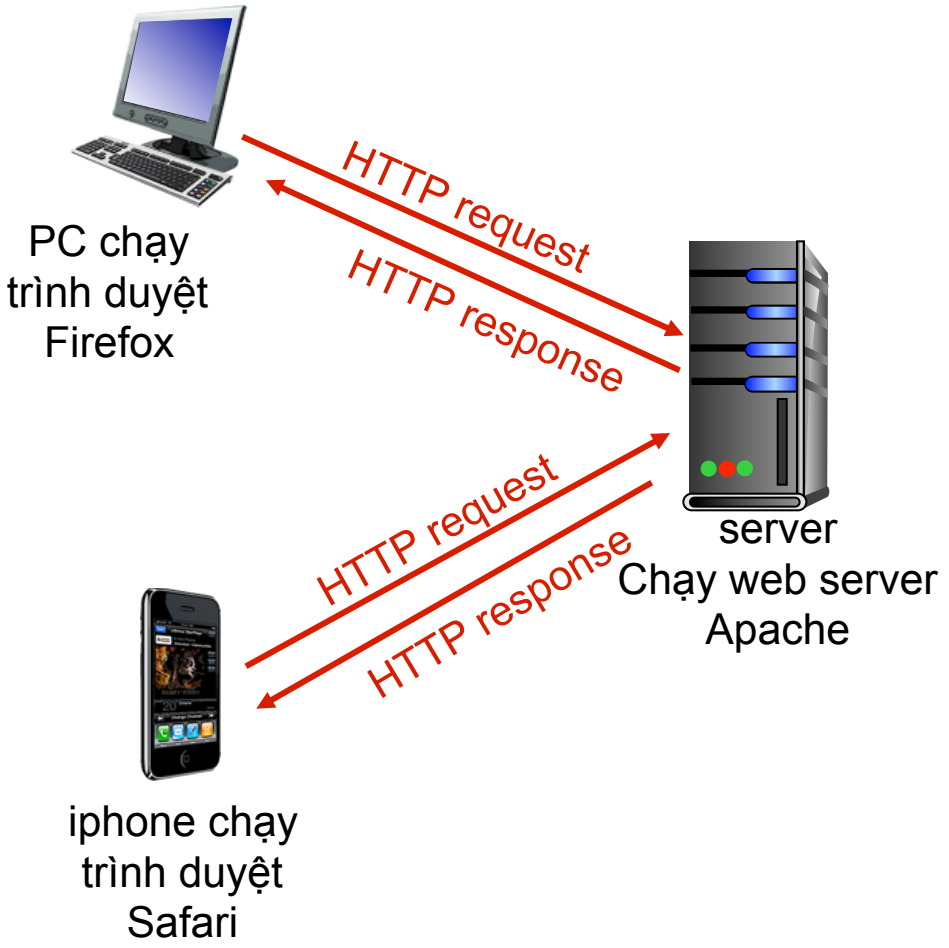
host name | path name

Tổng quan HTTP

HTTP: hypertext transfer protocol

- Giao thức web ở tầng Application

- Mô hình client/server
  - *client:* trình duyệt yêu cầu, nhận (dùng giao thức HTTP) và "hiện thị" các đối tượng Web
  - *server:* Web server gởi (dùng giao thức HTTP) các đối tượng để đáp ứng yêu cầu



Tổng quan HTTP (tt)

Dùng TCP:

- Client khởi tạo kết nối TCP (tạo socket) đến server, port 80.
- server chấp nhận kết nối TCP từ client.
- Các thông điệp HTTP (thông điệp giao thức tầng application) được trao đổi giữa trình duyệt (HTTP client) và web server (HTTP server).
- Kết nối TCP được đóng.

HTTP là "stateless"

- server không duy trì thông tin về các yêu cầu trước đó của client.

Vấn đề liên quan

Các giao thức nào mà duy trì "trạng thái" là phức tạp!

- Lịch sử trước đó (trạng thái) phải được duy trì
- Nếu server/client bị sự cố, cách nhìn về "trạng thái" của nó có thể bị mâu thuẫn, phải được điều chỉnh

Các kết nối HTTP

Nonpersistent HTTP

- Chỉ tối đa một đối tượng được gởi qua kết nối TCP.
  - Kết nối sau đó sẽ bị đóng.
- HTTP/1.0.

Persistent HTTP

- Nhiều đối tượng có thể được gởi qua một kết nối TCP giữa client và server.
- HTTP/1.1.

HTTP không bền vững

Giả sử người dùng vào URL như sau:
www.someSchool.edu/someDepartment/home.index

(chứa text, tham chiếu đến 10 hình jpeg)

1a. HTTP client khởi tạo kết nối TCP đến HTTP server (tiến trình) tại www.someSchool.edu trên port 80

1b. HTTP server tại host www.someSchool.edu chờ kết nối TCP tại port 80. "chấp nhận" kết nối, thông báo cho client

2. HTTP client gởi thông điệp yêu cầu HTTP (chứa URL) vào trong socket kết nối TCP. Thông điệp chỉ ra rằng client muốn đối tượng someDepartment/home.index

3. HTTP server nhận thông điệp yêu cầu, định dạng *thông điệp phản hồi* chứa đối tượng được yêu cầu, và gởi thông điệp đến socket của nó

Thời gian

HTTP không bền vững(tt)

4. HTTP server đóng kết nối TCP.

5. HTTP client nhận thông điệp phản hồi chứa file html, hiển thị html. Phân tích cú pháp file html, tìm ra các đối tượng jpeg được tham chiếu

time

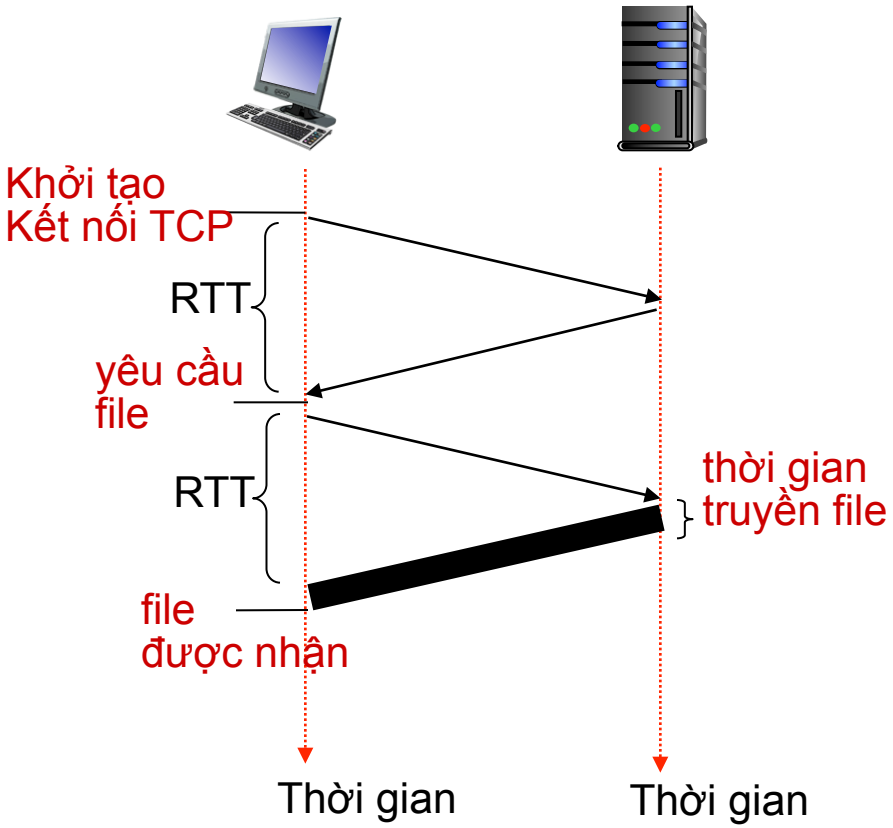
6. Các bước 1-5 được lặp lại cho mỗi đối tượng trong 10 đối tượng jpeg

HTTP không bền vững: thời gian đáp ứng

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

Thời gian đáp ứng HTTP:

- Một RTT để khởi tạo kết nối TCP
- Một RTT cho yêu cầu HTTP và vài byte đầu tiên của đáp ứng HTTP được trả về
- Thời gian truyền file
- Thời gian đáp ứng HTTP không bền vững = 2RTT + thời gian truyền file



Persistent HTTP

Nonpersistent HTTP:

- Yêu cầu 2 RTTs cho một đối tượng.
- Một TCP connection làm việc với 1 đối tượng
- Trình duyệt thường mở song song nhiều TCP connections đến các đối tượng được quan tâm.

Persistent HTTP

- Server giữ lại trạng thái mở của kết nối sau khi gởi response
- Nhiều HTTP messages giữa client/server được gởi qua kết nối đang mở

Persistent ***without*** pipelining:

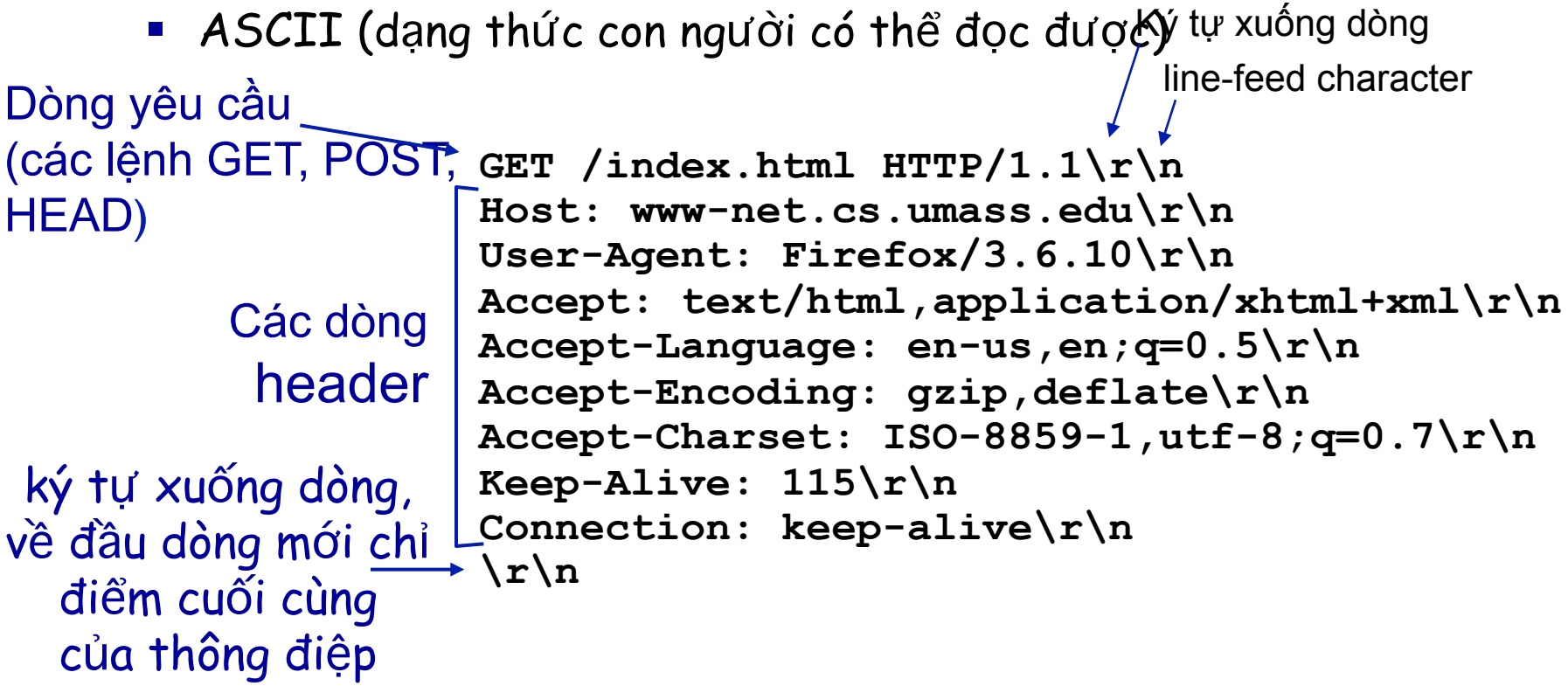
- Client chỉ gởi request khi đã nhận được response trước.
- 1 RTT cho 1 đối tượng được quan tâm.

Persistent ***with*** pipelining:

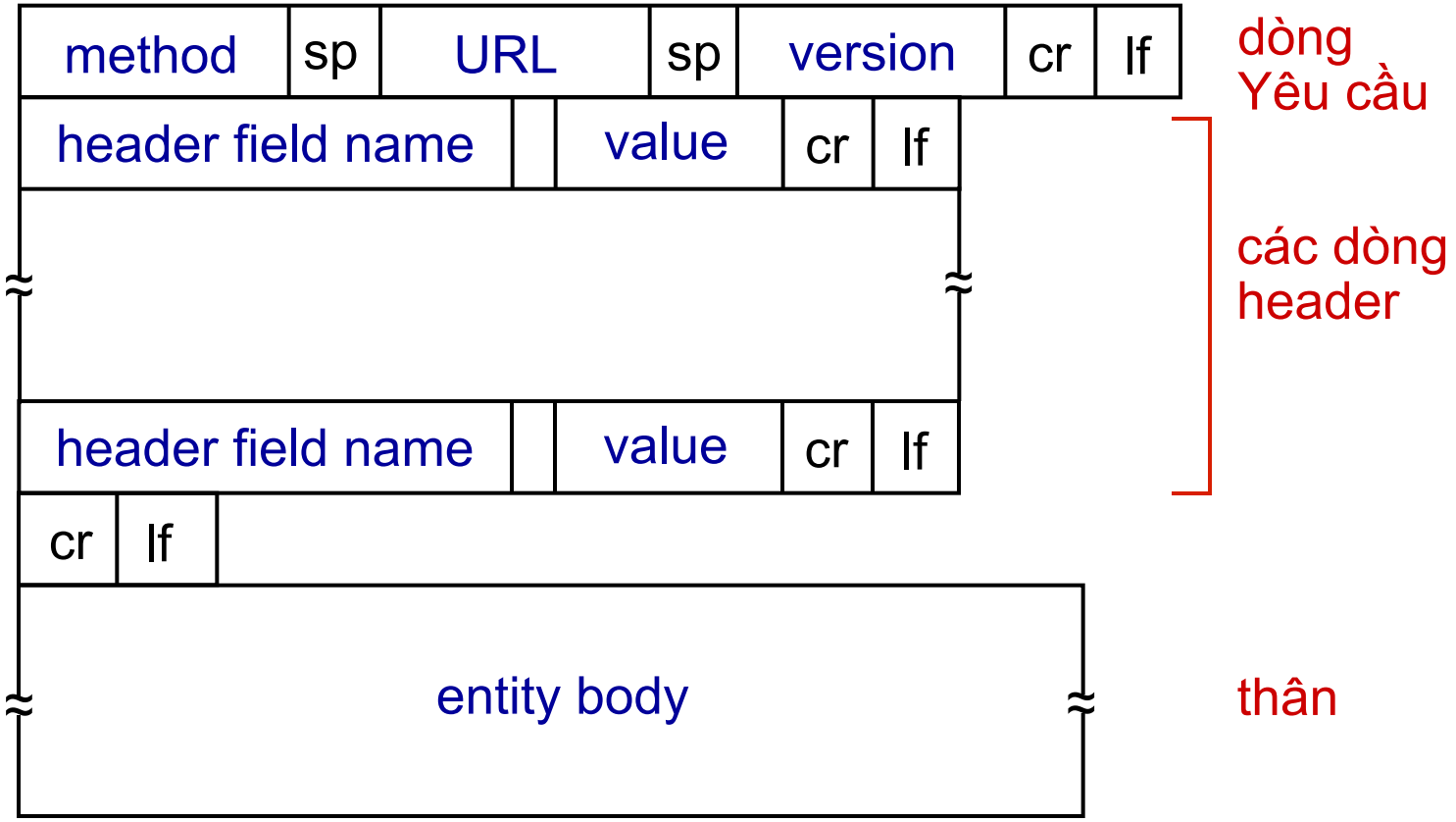
- Client gởi request liên tục đến các đối tượng được quan tâm
- Có thể 1 RTT cho tất cả các đối tượng được quân tâm

Thông điệp yêu cầu HTTP

- hai loại thông điệp HTTP: yêu cầu *(request)*, đáp ứng *(response)*
- Thông điệp yêu cầu HTTP:
  - ASCII (dạng thức con người có thể đọc được)



Thông điệp yêu cầu HTTP: định dạng tổng quát



Tải lên form input

Phương pháp POST:

- web page thường bao gồm form input
- input được tải lên server trong body thực thể

Phương pháp URL:

- Dùng phương thức GET
- input được tải trong trường URL của dòng yêu cầu:

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

Các phương thức

HTTP/1.0:

- GET
- POST
- HEAD
  - Yêu cầu server để lại đối tượng được yêu cầu ra khỏi sự đáp ứng

HTTP/1.1:

- GET, POST, HEAD
- PUT
  - Tải file trong body thực thể đến đường dẫn được xác định trong trường URL
- DELETE
  - Xóa file được chỉ định trong trường URL

Thông điệp đáp ứng HTTP

dòng trạng thái
(giao thức
mã trạng thái
cụm từ trạng thái)

các dòng
header

Dữ liệu, ví dụ,
file HTML
được 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 đáp ứng HTTP

- Mã trạng thái xuất hiện trong dòng đầu tiên trong thông điệp đáp ứng từ server tới client
- Một số code mẫu:

200 OK

- Yêu cầu thành công, đối tượng được yêu cầu sau ở trong thông điệp này

301 Moved Permanently

- Đối tượng được yêu cầu được di chuyển, vị trí mới được xác định sau trong thông điệp này (Location:)

400 Bad Request

- Thông điệp yêu cầu không được hiểu bởi server

404 Not Found

- Tài liệu được yêu cầu không tìm thấy trên server này

505 HTTP Version Not Supported

Kiểm tra HTTP (phía client)

1. Telnet đến Web server yêu thích của bạn:

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

Mở kết nối TCP ở port 80 (port server HTTP mặc định) tại cis.poly.edu. Mọi thứ nhập vào được gởi đến port 80 tại cis.poly.edu

2. Nhập vào yêu cầu trong lệnh GET HTTP:

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

Bằng cách gõ những dòng này (enter 2 lần), bạn đã gửi yêu cầu GET tối thiểu (nhưng đầy đủ) đến HTTP server

3. Xem thông điệp đáp ứng được gửi bởi HTTP server! (hoặc dùng Wireshark để xem thông điệp yêu cầu và đáp ứng của HTTP được bắt lại)

Trạng thái User-server: cookies

Nhiều Web site dùng cookies

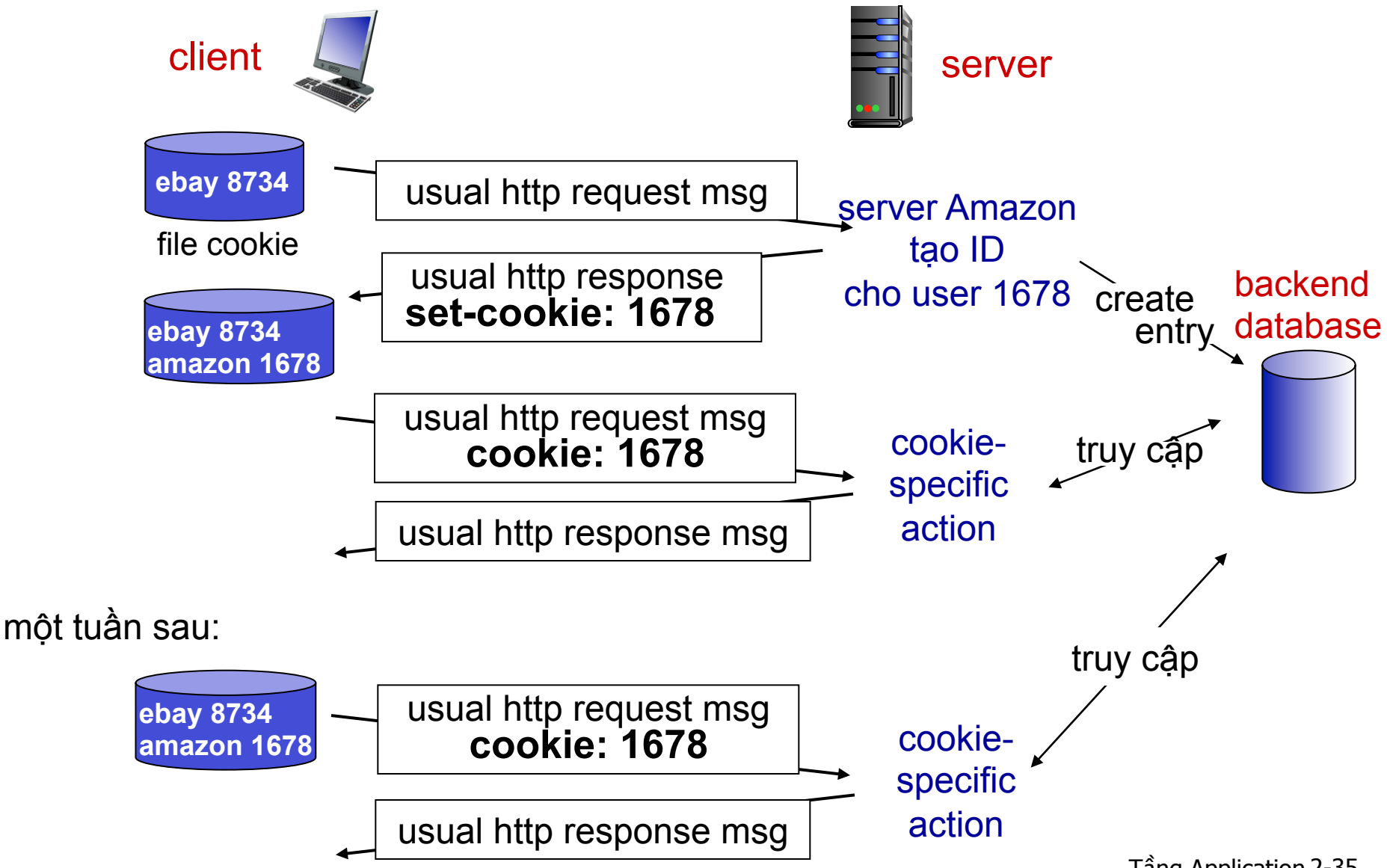
4 thành phần:

1) cookie header line của thông điệp đáp ứng HTTP
2) cookie header line trong thông điệp đáp ứng HTTP kế tiếp
3) File cookie được lưu trữ trên host của người dùng, được quản lý mởi trình duyệt của người sử dụng
4) Cở sở dữ liệu back-end tại Web site

Ví dụ:

- Susan thường truy cập Internet từ một PC
- Vào trang thương mại điện tử cho lần đầu tiên
- Khi yêu cầu khởi tạo HTTP đến trang web đó, thì trang đó tạo:
  - ID duy nhất
  - Một entry trong cơ sở dữ liệu backend cho ID đó

Cookies: lưu trữ "trạng thái" (tt.)



Cookies (tt)

Cookie có thể được sử dụng cho:

- Sự cấp phép
- Giỏ mua hàng
- Các khuyến cáo
- Trạng thái phiên làm việc của user (Web e-mail)

Ngoài ra

cookies và sự riêng tư:

- cookie cho phép các site biết nhiều hơn về bạn
- Bạn có thể cung cấp tên và email cho site

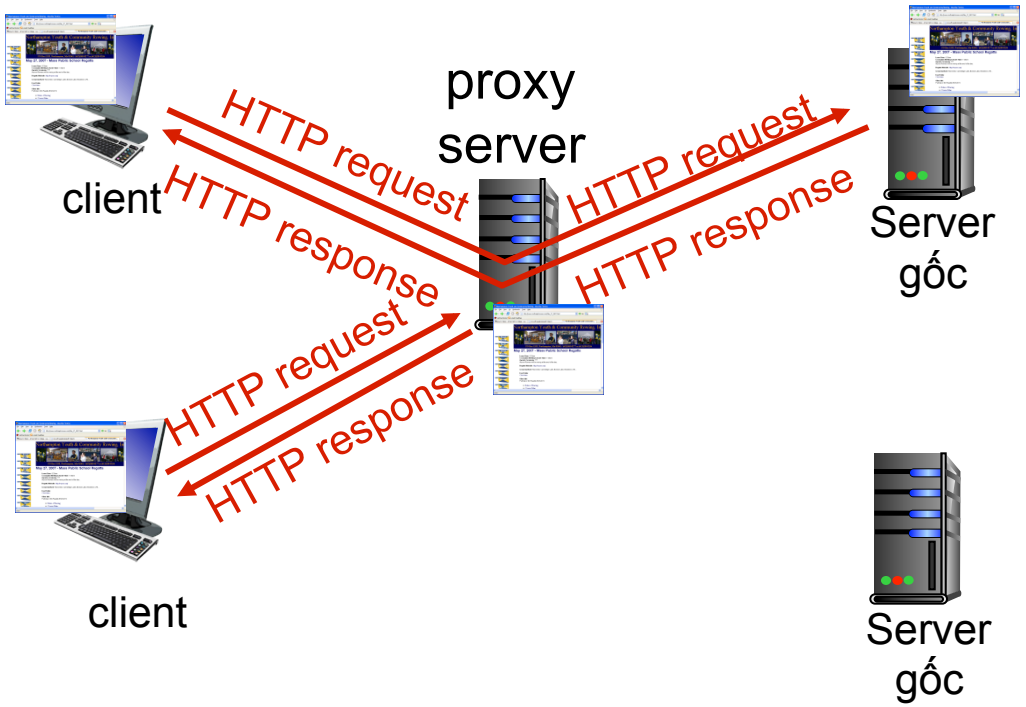
Làm thế nào để giữa "trạng thái":

- các thời điểm kết thúc giao thức: duy trì trạng thái tại người gởi/ nhận thông qua nhiều giao dịch
- cookies: các thông điệp http mang trạng thái

Web caches (proxy server)

Mục tiêu: đáp ứng yêu cầu của client không cần liên quan đến server gốc (server chứa đối tượng mà client cần)

- User thiết lập trình duyệt: truy cập Web thông qua cache
- Trình duyệt gởi tất cả yêu cầu HTTP đến cache
  - Đối tượng cache: cache trả về đối tượng
  - Ngược lại cache yêu cầu đối tượng từ server gốc, sau đó trả đối tượng đó cho client



Thông tin thêm về Web caching

- Cache hoạt động ở cả client và server
  - server cho client yêu cầu ban đầu
  - client đến server ban đầu
- Thông thường cache được cài đặt bởi ISP (trường đại học, công ty, ISP riêng)

Tại sao dùng Web caching?

- Giảm thời gian đáp ứng cho yêu cầu của client
- Giảm lưu lượng trên đường link truy cập của một tổ chức
- Caches dày đặc trên Internet: cho phép những nhà cung cấp dịch vụ có thể cung cấp nội dung một cách hiệu quả hơn. (chia sẽ file P2P cũng vậy).

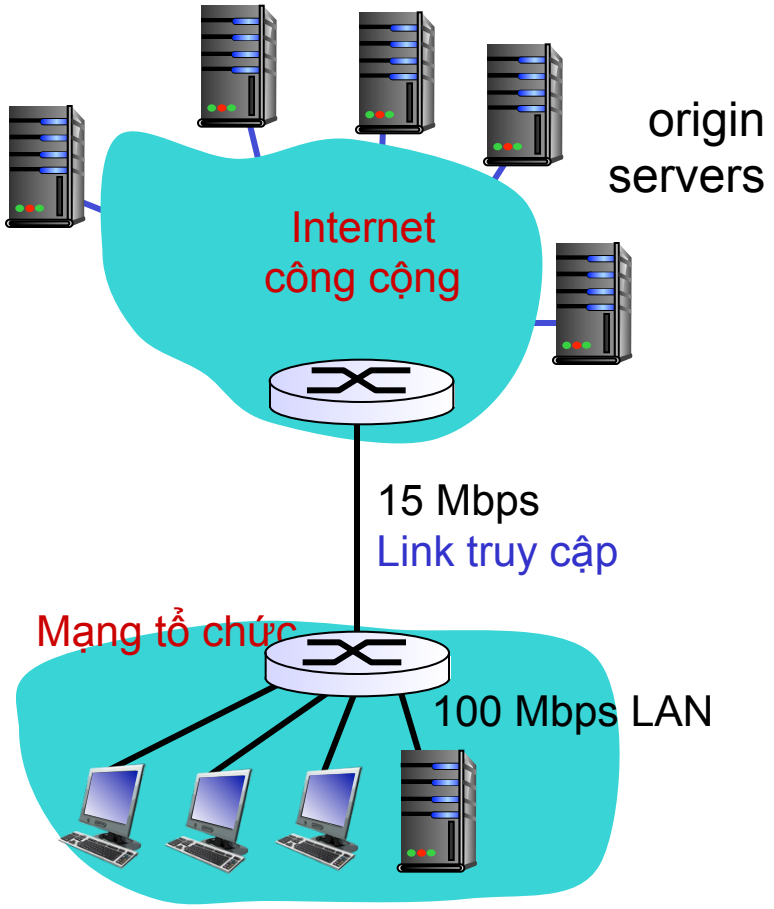
Ví dụ Caching:

Giả sử:

- Kích thước trung bình của đối tượng: 1 Mb.
- Tốc độ gởi yêu cầu trung bình từ trình duyệt đến server gốc: 15/giây
- Từ bộ định tuyến của tổ chức đến bất kỳ server gốc: 2 giây
- Tốc độ truy cập Link truy cập: 15 Mbps
- Tốc độ truy cập trong LAN: 100 Mbps

Kết quả: *Vấn đề!*

- Độ khả dụng của LAN: 15%
- Độ khả dụng của link truy cập = 99%
- Tổng thời gian trễ = trễ Internet + trễ truy cập + trễ LAN
  = 2 giây + minutes + usecs



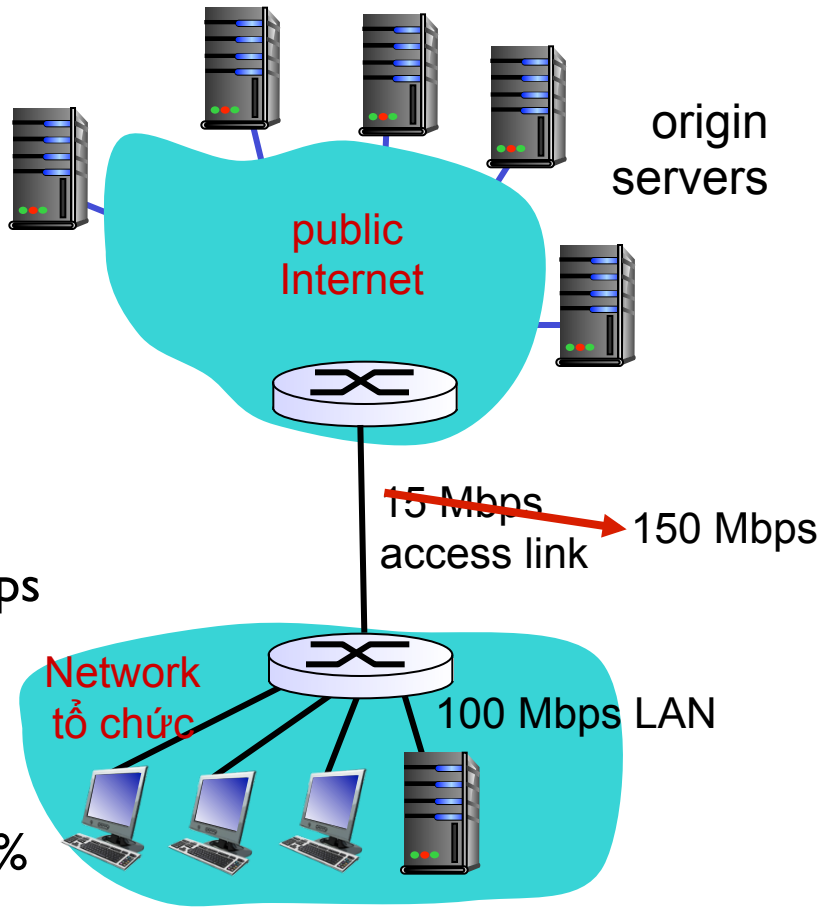
Ví dụ Caching: đường link truy cập lớn hơn

Giả sử:

- Kích thước trung bình của đối tượng: 1 Mb
- tốc độ trung bình yêu cầu từ trình duyệt đến server = 15/s
- Tốc độ trung bình dữ liệu đến trình duyệt: 1.50 Mbps
- RTT từ bộ định tuyến của tổ chức đến bất kỳ server gốc: 2 giây
- Tốc độ link truy cập: 15 Mbps → 150 Mbps

Kết quả:

- độ khả dụng của LAN = 15%
- độ khả dụng trên liên kết truy cập= 99% → 9.9%
- Tổng độ trễ= trễ Internet + trễ truy cập + trễ LAN
  = 2 sec + minutes + usecs → msecs



Chi phí: tốc độ link truy cập được tăng lên (không rẻ!)

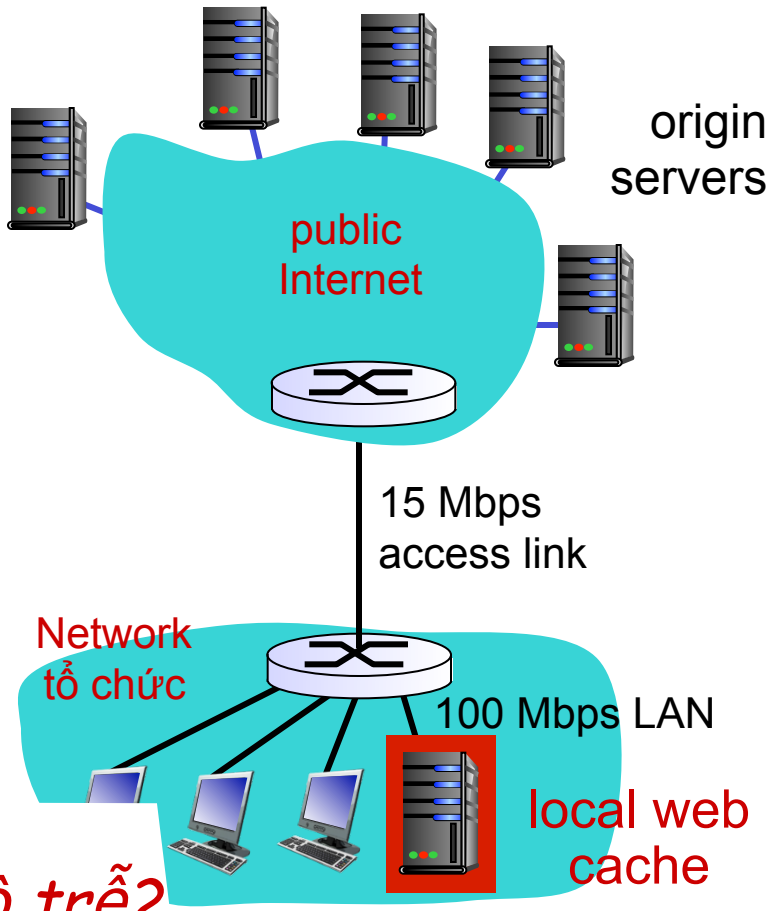
Ví dụ Caching: install local cache

Giả sử:

- Kích thước trung bình của đối tượng: 1 Mbits
- tốc độ trung bình yêu cầu từ trình duyệt đến server = 15/s
- RTT từ bộ định tuyến của tổ chức đến bất kỳ server gốc: 2 giây
- Tốc độ link truy cập: 15 Mbps

Kết quả:

- Độ khả dụng LAN: 15%
- access link utilization = 100%
- total delay = Internet delay + access delay + LAN delay
  = 2 sec + minutes + usecs



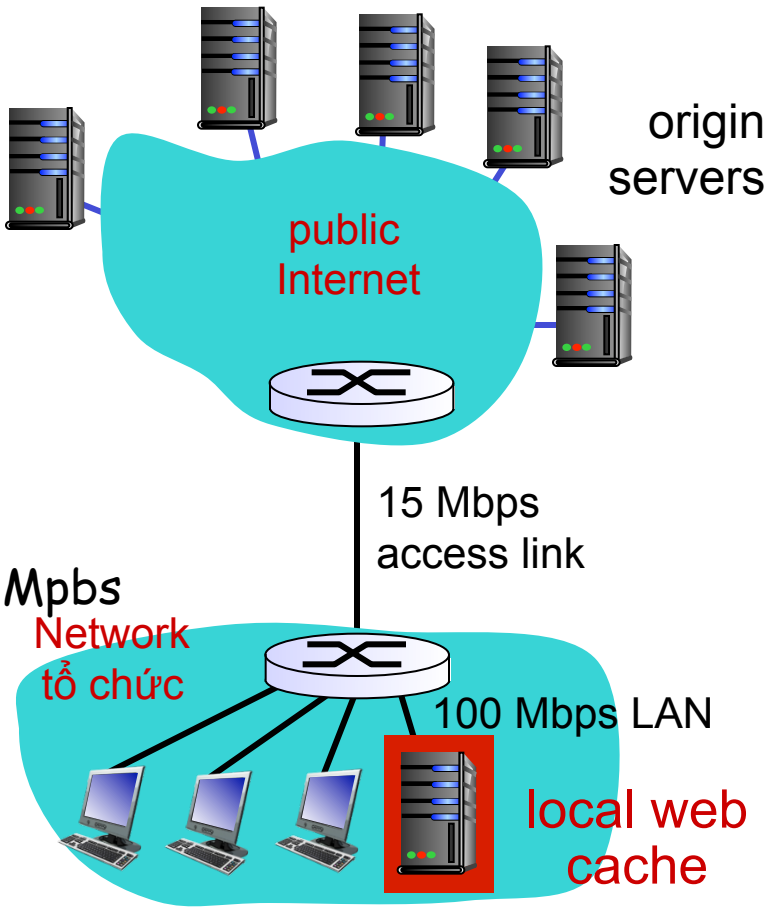
Làm cách nào để tính độ khả dụng đường link, độ trễ?

Chi phí: web cache (rẻ!)

Ví dụ Caching: install local cache

Tính độ khả dụng của đường link truy cập, độ trễ với cache:

- Giả sử tốc độ cache là 0.4
  - 40% yêu cầu được hài lòng ở cache, 60% yêu cầu được hài lòng ở server gốc
  - Độ hiệu dụng access link:
    - 60% yêu cầu dùng link truy cập
    - Lưu lượng cần dùng: 0.6*15Mbps=9Mpbs
    - Độ khả dụng: 9/15=0.6
  - Tổng độ trễ
    - = 0.6 * (độ trễ từ server gốc) +0.4 * (độ trễ khi được hài lòng tại cache)
    - = 0.6 (2) + 0.4 (~msecs)
    - = ~ 1.2 secs
    - Ít hơn với link 100 Mbps (và cũng rẻ hơn!)

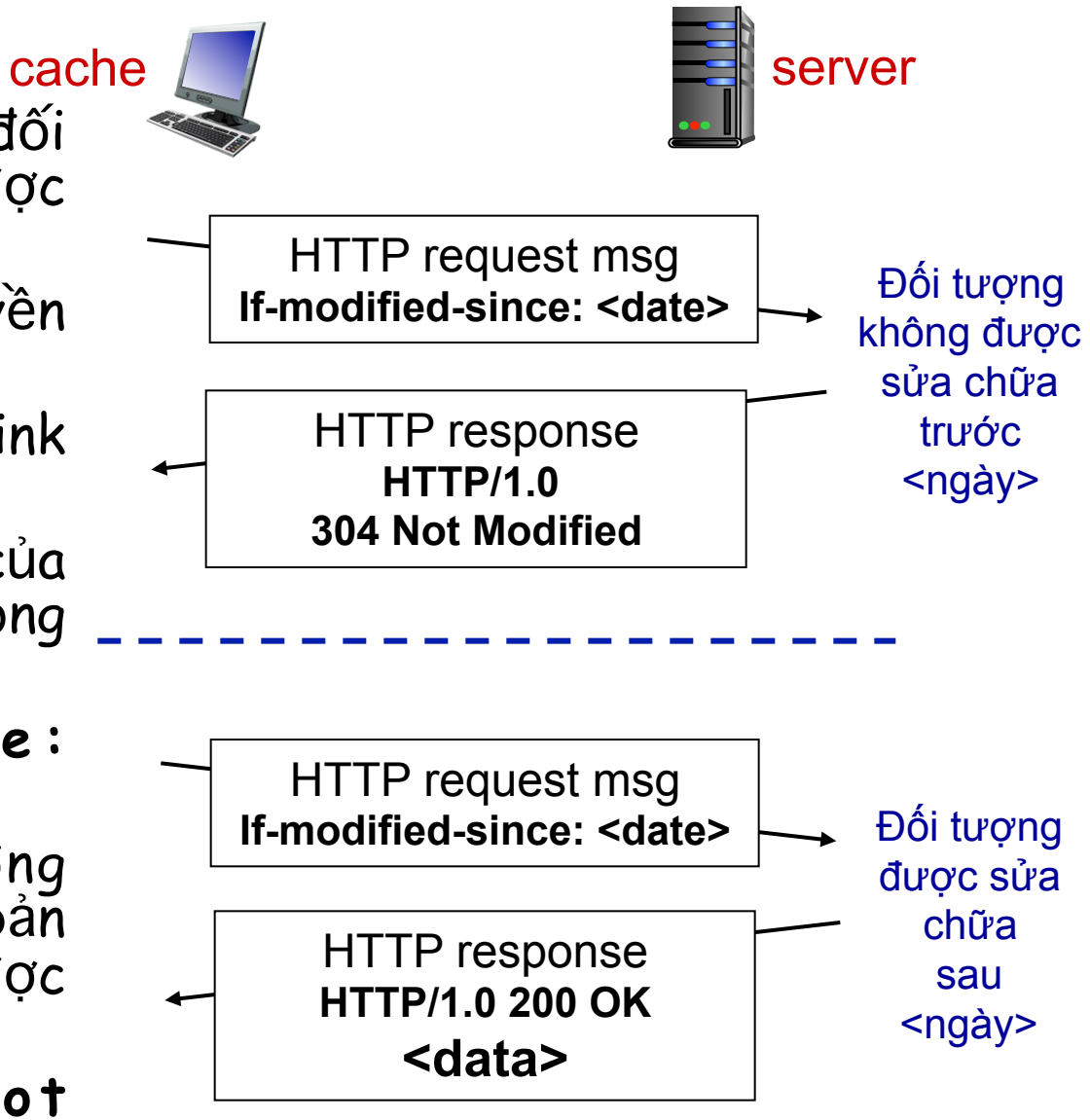


GET có điều kiện

- *Mục tiêu:* không gửi đối tượng nếu cache đã được cập nhật
  - Không có độ trễ truyền dữ liệu
  - Sự sử dụng đường link thấp hơn
- *Cache:* xác định ngày của bản sao được cache trong yêu cầu HTTP

  If-modified-since: <date>
- *Server:* đáp ứng không chứa đối tượng nếu bản copy trong cache đã được cập nhật:

  HTTP/1.0 304 Not Modified



Chương 2: Nội dung

2.1 Các nguyên lý của các ứng dụng mạng

2.2 Web và HTTP

2.3 FTP

2.4 electronic mail

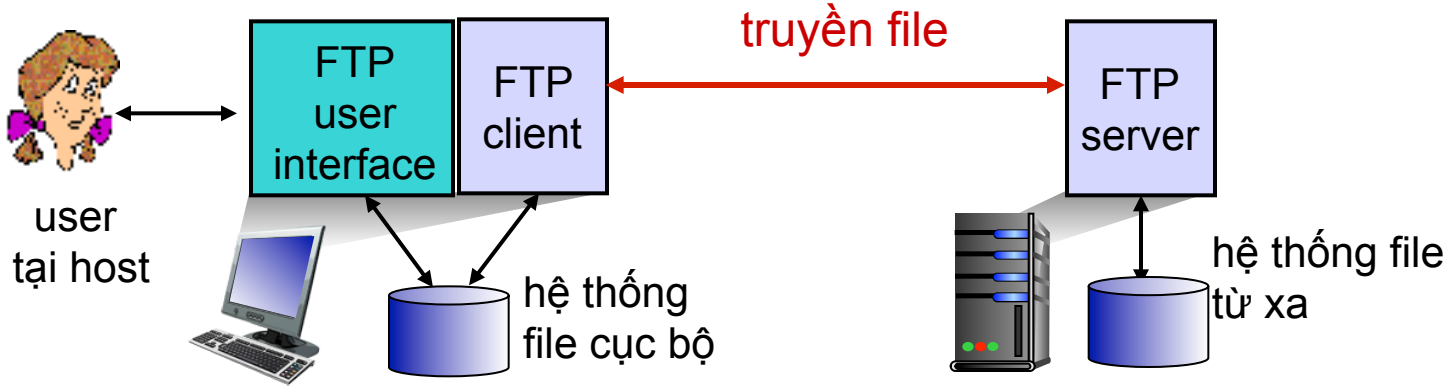
- SMTP, POP3, IMAP

2.5 DNS

2.6 các ứng dụng P2P

2.7 lập trình socket với UDP và TCP

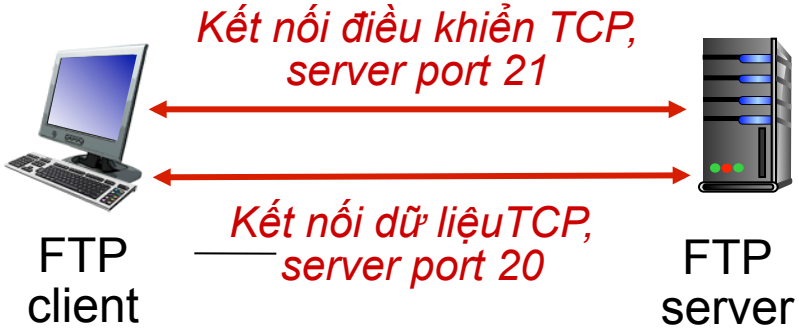
FTP: giao thức truyền file



- Truyền file đến/từ host từ xa
- Mô hình client/server
  - *client:* phía khởi tạo truyền (đến/từ host từ xa)
  - *server:* host ở xa
- Ftp: RFC 959
- Ftp server: port 21

FTP: điều khiển riêng biệt, kết nối dữ liệu

- FTP client liên hệ với FTP server tại port 21, dùng TCP
- client được cấp phép trên kết nối điều khiển
- client duyệt thư mục từ xa, gởi các lệnh trên kết nối điều khiển
- Khi server nhận lệnh truyền file, *server* mở kết nối dữ liệu TCP thứ 2 (truyền file) *đến* client
- Sau khi truyền một file, server đóng kết nối dữ liệu



- Server mở kết nối dữ liệu TCP khác để truyền file khác
- Kết nối điều khiển: *"out of band"*
- FTP server duy trì "trạng thái": thư mục hiện tại, xác thực trước đó

Các lệnh và phản hồi FTP

Các lệnh mẫu:

- Gởi văn bản ASCII trên kênh điều khiển
- **USER *username***
- **PASS *password***
- **LIST** trả về danh sách file trên thư mục hiện tại
- **RETR filename** lấy file
- **STOR filename** lưu trữ (đặt) file vào trong host ở xa

Ví dụ code trả về

- Mã trạng thái và cụm (như HTTP)
- **331 Username OK, password required**
- **125 data connection already open; transfer starting**
- **425 Can't open data connection**
- **452 Error writing file**

Chương 2: Nội dung

2.1 Các nguyên lý của các ứng dụng mạng

2.2 Web và HTTP

2.3 FTP

2.4 Thư điện tử

- SMTP, POP3, IMAP

2.5 DNS

2.6 các ứng dụng P2P

2.7 lập trình socket với UDP và TCP

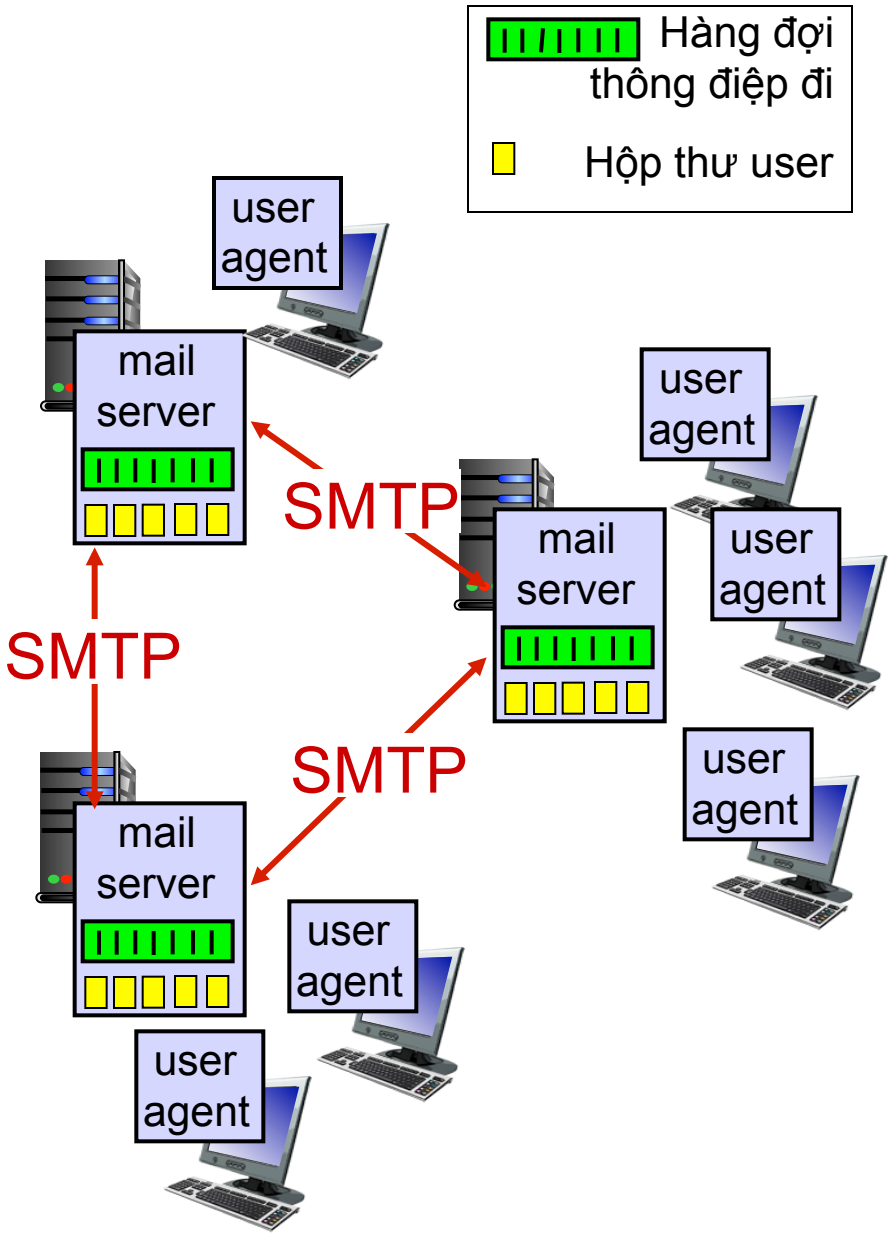
Thư điện tử

Ba thành phần chính:

- user agents
- mail servers
- simple mail transfer protocol: SMTP

User Agent

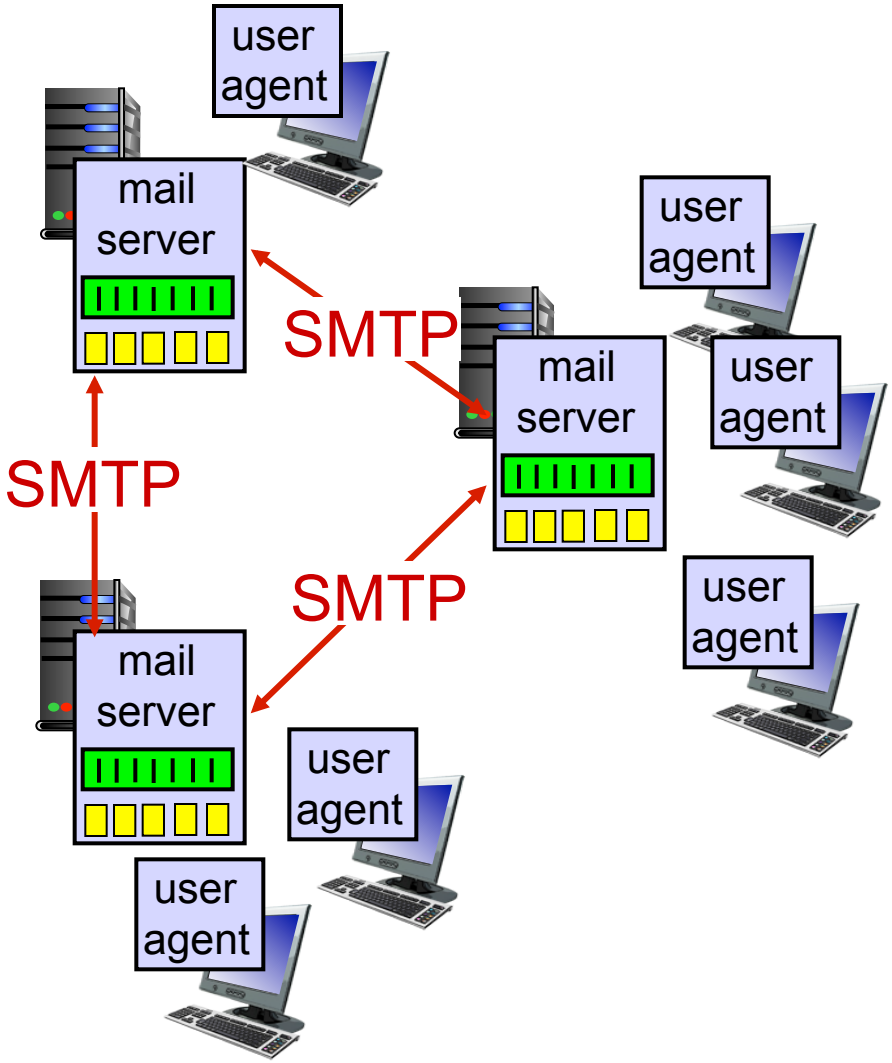
- Còn gọi là "mail reader"
- Soạn thảo, sửa đổi, đọc các thông điệp email
- Ví dụ Outlook, Thunderbird, iPhone mail client
- Các thông điệp đi và đến được lưu trên server



Thư điện tử: mail servers

Mail servers:

- *Hộp thư (mailbox)* chứa thông điệp đến user
- *Hàng thông điệp (message queue)* của các thông điệp mail ra ngoài (chuẩn bị gửi)
- *Giao thức SMTP* giữa các mail server để gửi các thông điệp email
  - client: gởi mail đến server.
  - "server": nhận mail từ server.

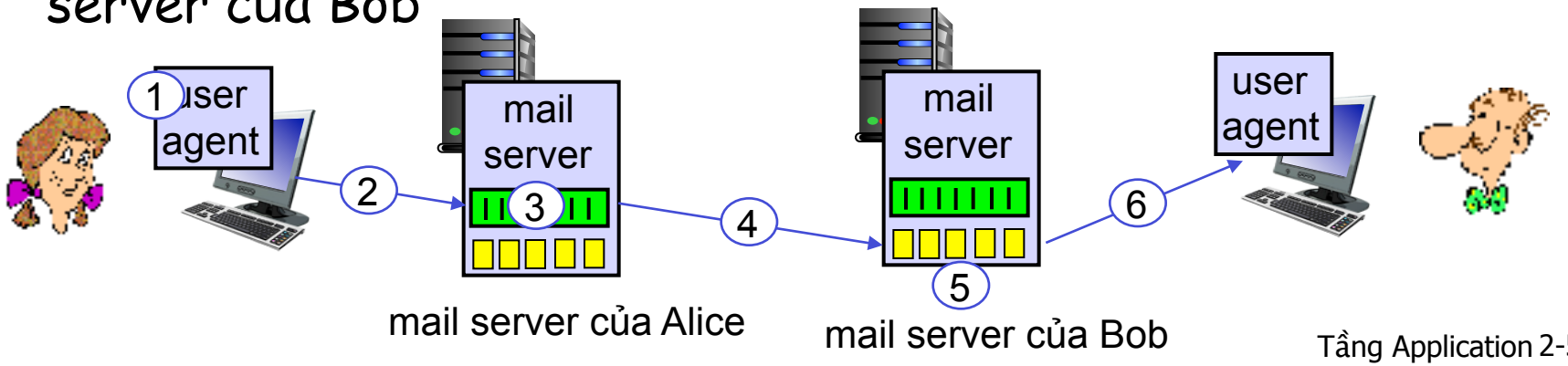


Thư điện tử: SMTP [RFC 2821]

- Sử dụng TCP để truyền thông điệp email một cách tin cậy từ client đến server, port 25
- Truyền trực tiếp: server gửi đến server nhận
- 3 giai đoạn truyền
  - Thiết lập kết nối.
  - Truyền thông điệp.
  - Đóng kết nối.
- Tương tác lệnh/phản hồi (như HTTP, FTP)
  - Lệnh: văn bản ASCII
  - Phản hồi: mã và cụm trạng thái
- Thông điệp phải ở dạng mã ASCI 7 bit

Tình huống: Alice gởi thông điệp đến Bob

1) Alice dùng một máy trạm để soạn thảo thông điệp "gởi đến" bob@someschool.edu
2) Máy trạm của Alice gởi thông điệp đến mail server của cô ta; thông điệp được đặt trong hàng đợi
3) phía client của SMTP mở kết nối TCP với mail server của Bob
4) SMTP client gửi thông điệp của Alice trên kết nối TCP
5) Mail server của Bob đặt thông điệp đó trong hộp thư của Bob
6) Bob kích hoạt user agent của anh ta để đọc thông điệp



Ví dụ tương tác SMTP

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250  Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

Thử nghiệm tương tác SMNP:

- **telnet servername 25**
- see 220 reply from server
- enter HELO, MAIL FROM, RCPT TO, DATA, QUIT commands

Lệnh ở trên cho phép bạn gửi email không cần dùng email client (reader)

SMTP: kết luận

- SMTP dùng kết nối bền vững
- SMTP yêu cầu thông điệp (header & body) phải ở dạng ASCII 7-bit
- SMTP server dùng uses CRLF.CRLF để xác định kết thúc thông điệp

So sánh với HTTP:

- HTTP: kéo
- SMTP: đẩy
- Cả hai đều có tương tác lệnh/phản hồi, các mã trạng thái dạng ASCII
- HTTP: mỗi đối tượng được đóng gói trong thông điệp phản hồi của nó
- SMTP: nhiều đối tượng được gửi trong thông điệp nhiều phần

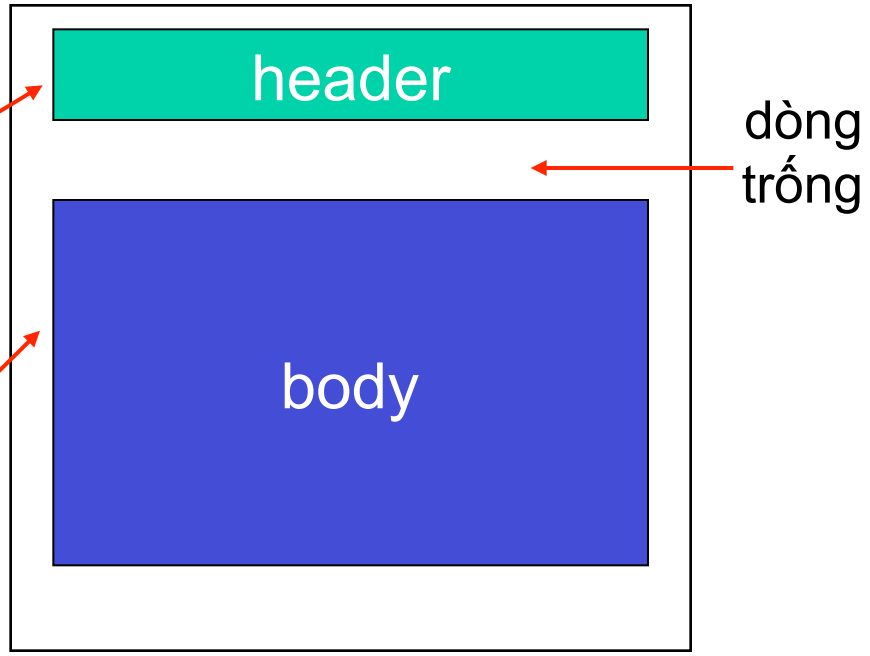
Định dạng thông điệp Mail

SMTP: giao thức dùng cho trao đổi thông điệp email

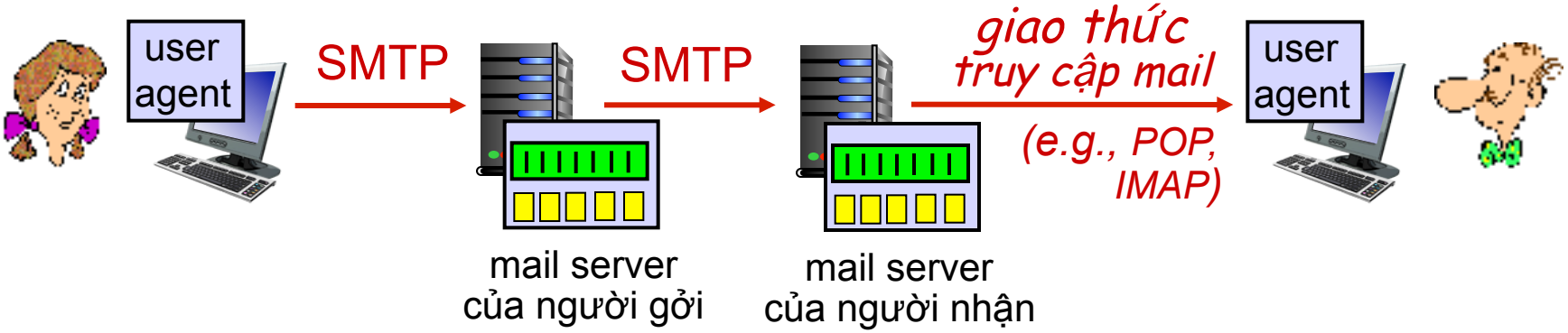
RFC 822: chuẩn cho định dạng thông điệp văn bản:

- Các dòng header, ví dụ
  - To:
  - From:
  - Subject:

  Khác với các lệnh SMTP MAIL FROM, RCPT TO!
- Body: "thông điệp"
  - Chỉ có các ký tự ASCII



Các giao thức truy cập Mail



- SMTP: truyền/lưu trữ vào server của người nhận
- Giao thức truy cập mail: từ server của người nhận
  - POP: Post Office Protocol [RFC 1939]: ủy quyền, download
  - IMAP: Internet Mail Access Protocol [RFC 1730]: nhiều tính năng hơn, bao gồm cả thao tác các thông điệp đã được lưu trên server
  - HTTP: gmail, Hotmail, Yahoo! Mail…

Giao thức POP3

giai đoạn ủy quyền (authorization)

- Các lệnh phía client:
  - **user**: khai báo username
  - **pass**: password
- Đáp ứng phía server
  - **+OK**
  - **-ERR**

```
S: +OK POP3 server ready
C: user bob
S: +OK
C: pass hungry
S: +OK user successfully logged on
```

Giai đoạn giao dịch, client:

- **list**: liệt kê các số thông điệp
- **retr**: trích xuất thông điệp theo số
- **dele**: xóa
- **quit**

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

POP3 và IMAP

Tìm hiểu thêm về POP3

- Sử dụng chế độ "tải xuống và xóa".
  - Bob không thể đọc lại e-mail nếu anh ta thay đổi client
- POP3 "tải xuống-và-giữ": sao chép các thông điệp trên các client khác nhau.
- POP3 không giữa trạng thái của các phiên làm việc.

IMAP

- Giữ tất cả các thông điệp ở một nơi: tại server.
- Cho phép người dùng tổ chức các thông điệp trong các thư mục.
- Giữ trạng thái của người dùng trong suốt phiên làm việc:
  - Các tên của các thư mục và ánh xạ giữa các ID của thông điệp và tên của thư mục

Chương 2: Nội dung

2.1 Các nguyên lý của các ứng dụng mạng

2.2 Web và HTTP

2.3 FTP

2.4 thư điện tử

- SMTP, POP3, IMAP

2.5 DNS

2.6 các ứng dụng P2P

2.7 lập trình socket với UDP và TCP

DNS: domain name system

Con người: nhiều cách nhận dạng:
- Số an sinh xã hội, tên, số hộ chiếu

Internet hosts, routers:
- Địa chỉ IP (32 bit) - được dùng cho định danh datagrams
- "tên", ví dụ www.yahoo.com - được dùng bởi con người

Q: làm cách nào để ánh xạ giữa địa chỉ IP và tên, và ngược lại?

Domain Name System:
- *Cơ sở dữ liệu phân tán* được thực hiện theo tổ chức phân cấp của nhiều *name server*
- *Giao thức tầng application:* các host, các name server truyền thông để *phân giải* tên (địa chỉ/ dịch tên)
  - Lưu ý: chức năng lõi Internet, được thực hiện như là giao thức tầng application
  - Sự phức tạp ở "biên" của mạng"

DNS: các dịch vụ, cấu trúc

Các dịch vụ DNS

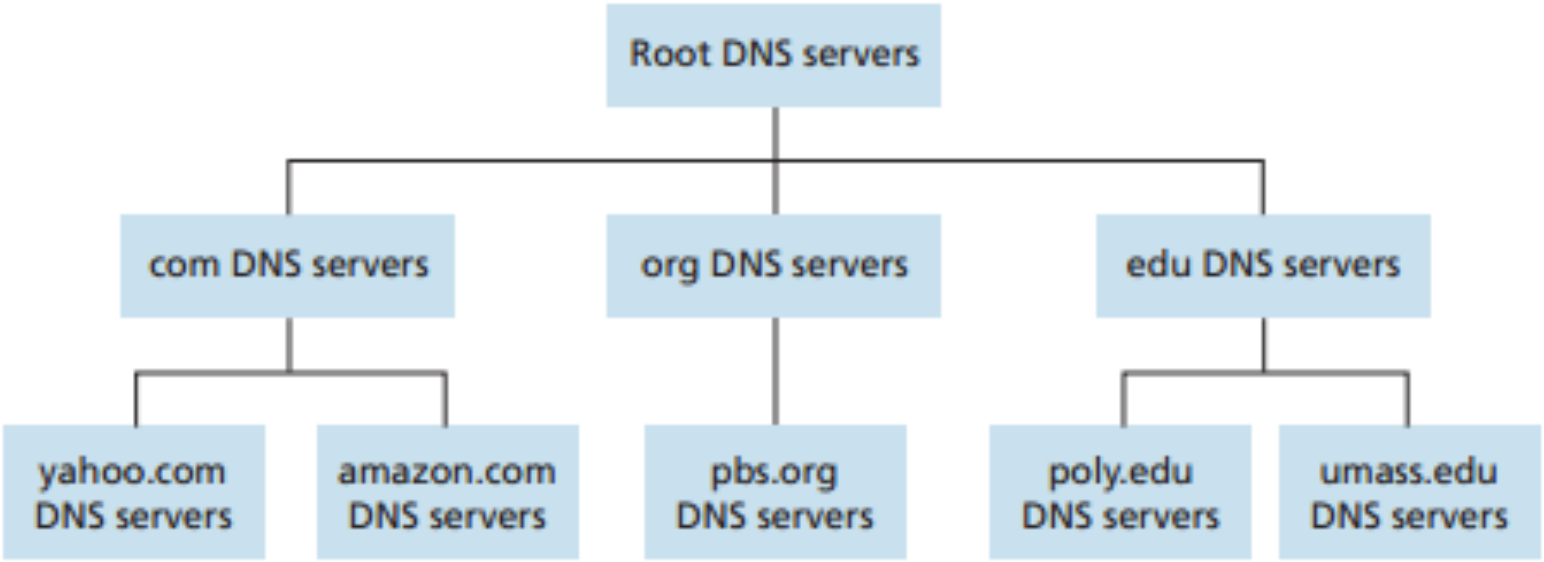
- Dịch tên host ra địa chỉ IP
- Bí danh host
  - Các tên đúng, bí danh
- Bí danh mail server
- Phân phối tải
  - Các web server: nhiều địa chỉ IP tương ứng cho 1 tên đúng chuẩn

Tại sao không tập trung hóa DNS?

- Một điểm chịu lỗi
- Lưu lượng
- Cơ sở dữ liệu tập trung
- Bảo trì

A: không có khả năng mở rộng!

DNS: cơ sở dữ liệu phân cấp, phân tán



client muốn địa chỉ IP của www.amazon.com:

- client truy vấn server gốc (root) để tìm DNS server com
- client truy vấn DNS server .com tìm DNS server amazon.com
- client truy vấn DNS server amazon.com để lấy địa chỉ IP của www.amazon.com

TLD, server có thẩm quyền

Các top-level domain (TLD) server :

- Chịu trách nhiệm cho tên miền com, org, net, edu, aero, jobs, museums, và tất cả các tên miền cao nhất của quốc gia, như là: uk, fr, ca, jp
- Network Solutions duy trì máy chủ cho .com TLD
- Lĩnh vực giáo dục cho .edu TLD

Các DNS server có thẩm quyền:

- DNS server của riêng tổ chức cung cấp các tên host có thẩm quyền để ánh xạ địa chỉ IP cho các host được đặt tên của tổ chức đó.
- Có thể được duy trì bởi tổ chức hoặc nhà cung cấp dịch vụ.

DNS name server cục bộ

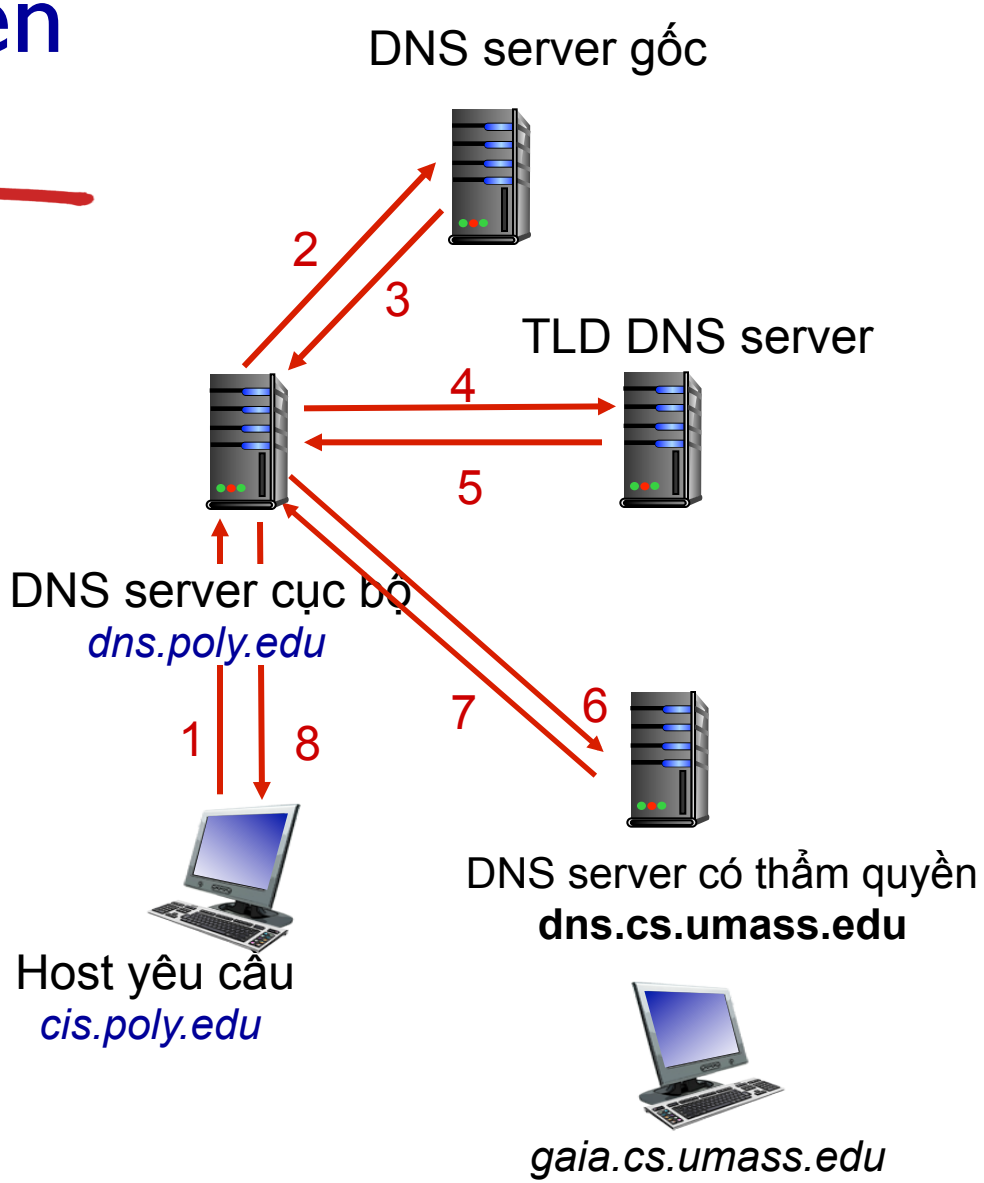
- Không hoàn toàn theo cấu trúc phân cấp
- Mỗi ISP (ISP cá nhân, công ty, trường đại học) có một server cục bộ.
  - Còn được gọi là "name server mặc định"
- Khi một host tạo một truy vấn DNS, truy vấn được gởi đến DNS server cục bộ của nó
  - Truy vấn vào bộ nhớ đệm (nhưng có thể hết hạn sử dụng).
  - Chuyển truy vấn vào trong tổ chức phân cấp.

Ví dụ phân giải tên miền DNS

- host tại cis.poly.edu muốn địa chỉ IP của gaia.cs.umass.edu

Truy vấn lặp:

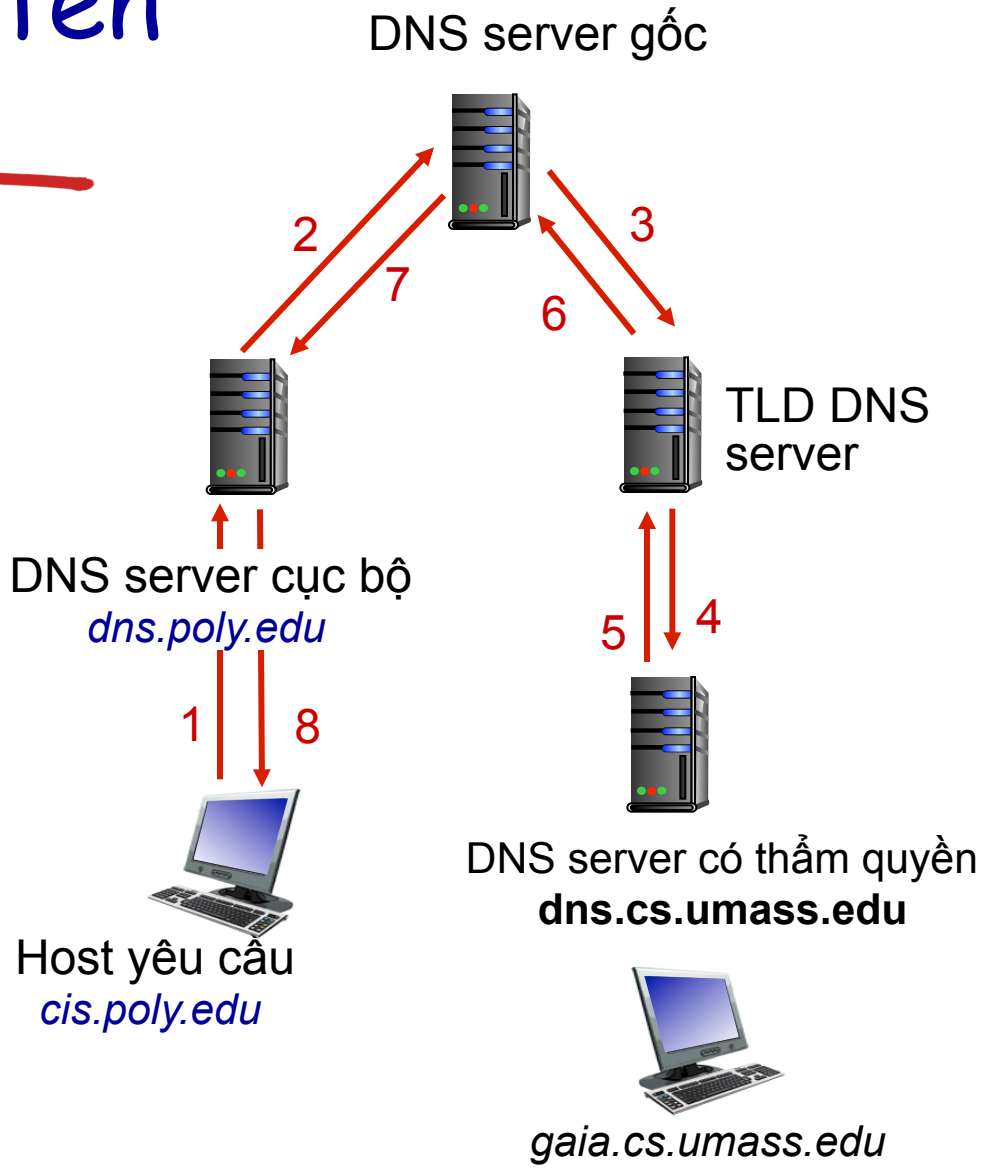
- **Server được liên lạc** sẽ trả lời với tên của **server để liên lạc**.
- "tôi không biết tên này, nhưng yêu cầu máy chủ này"



Ví dụ phân giải tên DNS

Truy vấn đệ quy:

- Đẩy trách nhiệm phân giải tên cho name server đã được tiếp xúc
- Tải nặng tại các tầng trên của hệ thống phân cấp?



DNS: caching, cập nhật các record

- Một khi name server học cách ánh xạ, nó sẽ *caches* ánh xạ đó
  - Các mục cache sẽ biến mất sau một vài lần TTL.
  - TLD servers thường được cache trong các name server cục bộ
    - Do đó các name server gốc không thường xuyên được truy cập
- Các mục được cache có thể hết hạn sử dụng (chuyển đổi tên-đến-địa chỉ nỗ lực nhất!)
  - Nếu tên host thay đổi địa chỉ IP, có thể không được biết đến trên Internet cho đến khi tất cả TTL hết hạn.
- Cơ chế cập nhật/thông báo được đề xuất bởi chuẩn IETF
  - RFC 2136

Các DNS record

DNS: cơ sở dữ liệu phân tán lưu trữ các record tài nguyên (RR)

Định dạng RR: **(name, value, type, ttl)**

type=A

- **Name** là tên host
- **Value** là địa chỉ IP

type=NS

- **Name** là tên miền (e.g., foo.com)
- **Value** là tên host của name server có thẩm quyền cho tên miền này

type=CNAME

- **Name** là bí danh của một số tên "chuẩn" (tên thực) **www.ibm.com** is really **servereast.backup2.ibm.com**
- **Value** là tên chuẩn (tên thật)

type=MX

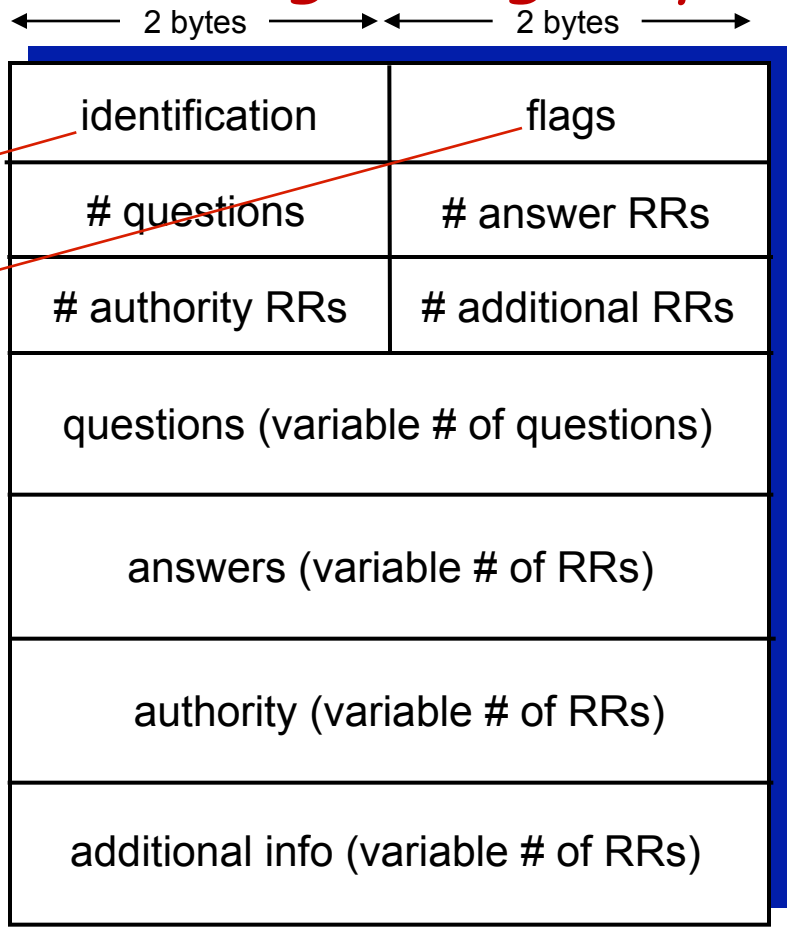
- **Value** là tên của mail server được liên kết với **name**

Giao thức và các thông điệp DNS

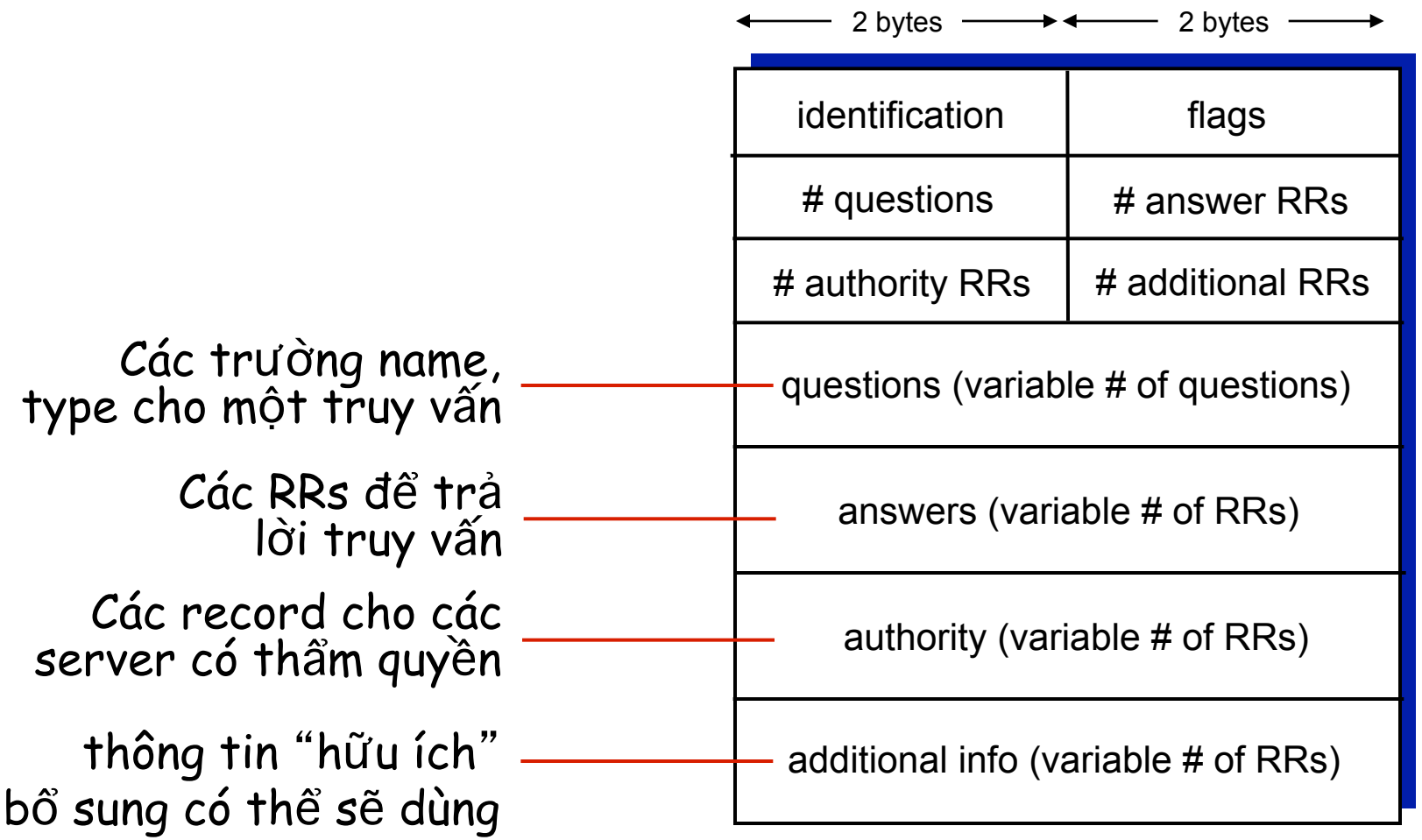
- Các thông *điệp truy vấn (query)* và *trả lời (reply)*, đều có cùng *định dạng thông điệp*

msg header

- identification: 16 bit # cho truy vấn, trả lời cho truy vấn dùng cùng #
- flags:
  - Truy vấn hoặc trả lời
  - Đệ quy mong muốn
  - Đệ quy sẵn sàng
  - Trả lời có thẩm quyền



Giao thức và các thông điệp DNS



Chèn các record vào trong DNS

- Ví dụ: khởi động mới "Network Utopia"
- Đăng ký tên miền networkuptopia.com tại một *DNS registrar* (như là Network Solutions)
  - Cung cấp tên, địa chỉ IP của server có thẩm quyền (primary and secondary)
  - Registrar chèn hai RR vào trong server .com TLD :
    (networkutopia.com, dns1.networkutopia.com, NS)
    (dns1.networkutopia.com, 212.212.212.1, A)
- Tạo record type A trong server có thẩm quyền cho www.networkuptopia.com; type MX record cho networkutopia.com

Tấn công DNS

Tấn công DDoS

- Các server gốc Bombard với lưu lượng
  - Không thành cho đến nay
  - Lọc lưu lượng
  - Các DNS server cục bộ cache các địa chỉ IP của TLD servers, cho phép bỏ qua server gốc
- Các server Bombard TLD
  - Nguy hiểm hơn

Tấn công chuyển hướng

- Man-in-middle
  - Ngăn chặn các truy vấn
- Đầu độc DNS
  - Gửi các trả lời giả tạo đến DNS server

Khai thác DNS cho tấn công DDoS

- Gởi các truy vấn với địa chỉ nguồn giả tạo: địa chỉ IP mục tiêu
- Yêu cầu khuếch đại

Chương 2: Nội dung

2.1 Các nguyên lý của các ứng dụng mạng

2.2 Web và HTTP

2.3 FTP

2.4 thư điện tử

- SMTP, POP3, IMAP

2.5 DNS

2.6 các ứng dụng P2P

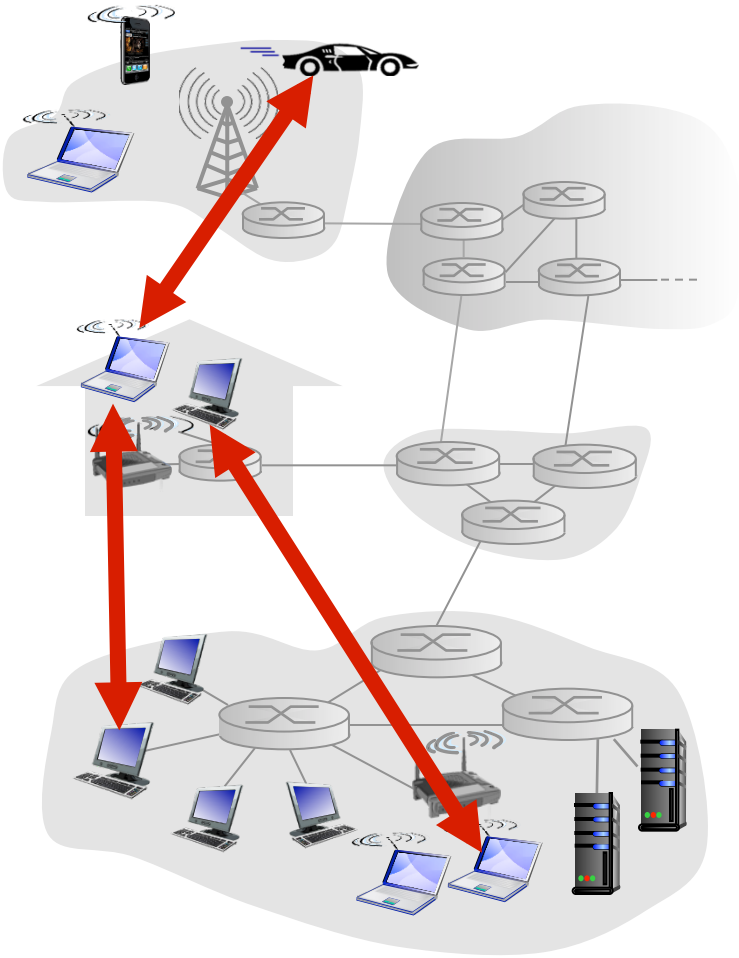
2.7 lập trình socket với UDP và TCP

Kiến trúc P2P thuần túy

- Server không hoạt động liên tục
- Các hệ thống đầu cuối bất kỳ giao tiếp trực tiếp với nhau
- Các peer được kết nối liên tục và thay đổi địa chỉ IP

Ví dụ:

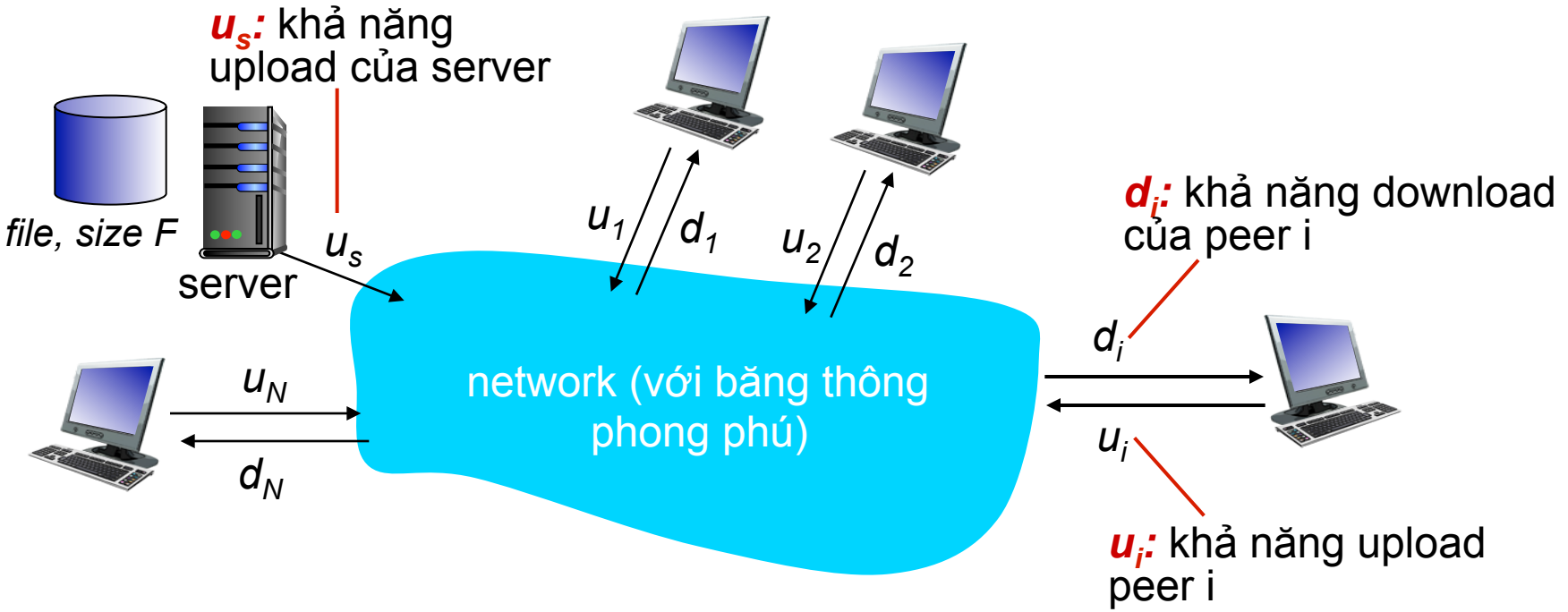
- Phân phối file (BitTorrent)
- Streaming (KanKan)
- VoIP (Skype)



Phân phối file: so sánh giữa client-server và P2P

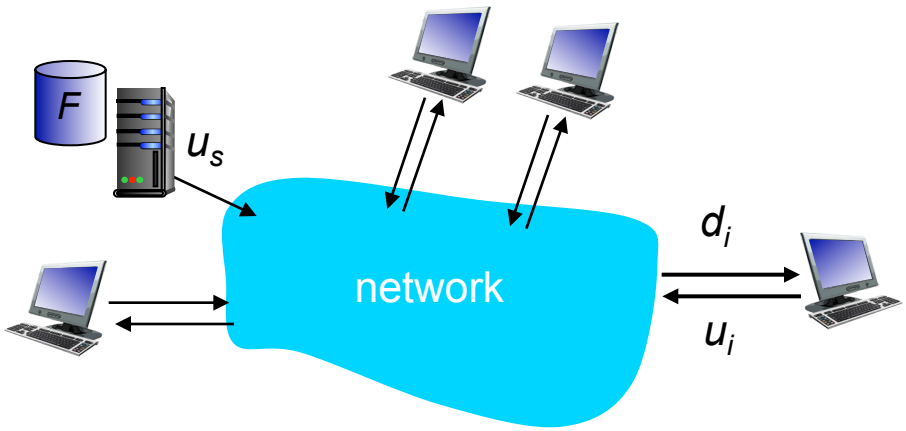
Câu hỏi: mất bao lâu để phân phối file (how much time to distribute file) (kích thước *F*) từ một server đến N peer?

- Khả năng tải lên/tải xuống của peer bị giới hạn



Thời gian phân phối file: client-server

- *Truyền máy chủ:* phải gửi (tải lên) tuần tự N bản sao file:
  - Thời gian để gởi một bản sao: F/u_s
  - Thời gian để gởi N bản sao: NF/u_s
    - *client: mỗi* client phải tải xuống bản sao của file
      - d_{min} = tốc độ tối thiểu client download
      - Thời gian thối thiểu client download: F/d_{min}



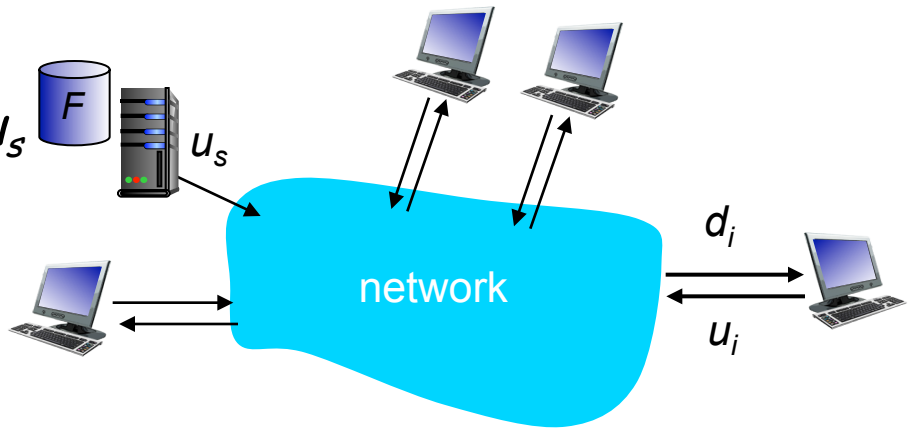
Thời gian để phân phối F đến N client dùng phương pháp client-server

$$D_{c\text{-}s} \geq max\{NF/u_s, F/d_{min}\}$$

Tăng tuyến tính trong N

Thời gian phân phối file: P2P

- *Truyền server:* phải upload ít nhất một bản sao
  - Thời gian gởi một bản sao: F/u_s
- *client:* mỗi client phải download bản sao file
  - Thời gian tối thiểu client tải xuống: F/d_{min}
- *clients:* trong khi tổng thể cần phải tải về NF bit
  - Tốc độ upload tối đa (tốc độ tải về giới hạn tối đa) là $u_s + Su_i$



Thời gian phân phối F đến N client dùng phương pháp P2P

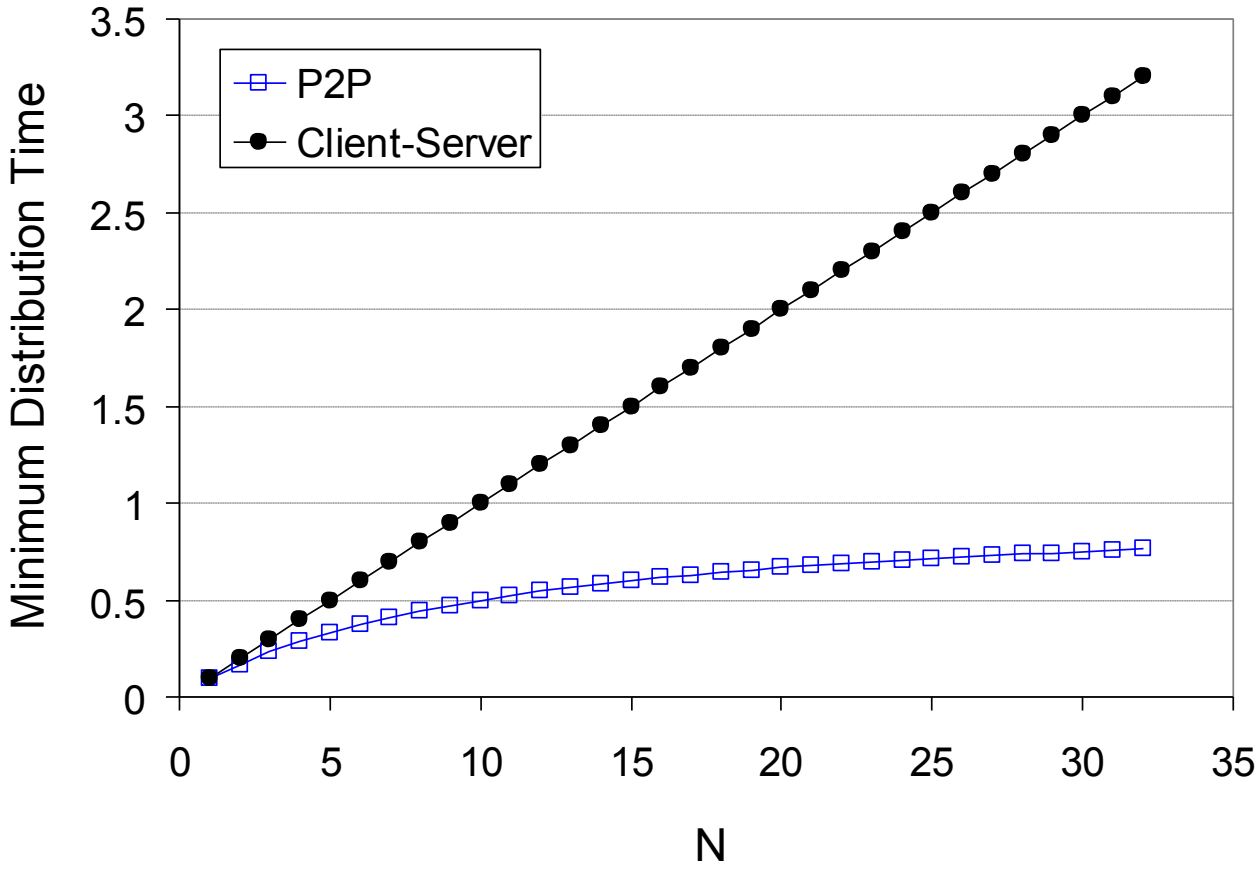
$$D_{P2P} \geq max\{F/u_s, F/d_{min}, NF/(u_s + \Sigma u_i)\}$$

Tăng tuyến tính trong *N* …

… nhưng để thực hiện điều này, mỗi khi mỗi peer mang lại năng lực dịch vụ

So sánh Client-server với P2P: ví dụ

Tốc độ client upload = u, F/u = 1 hour, $u_s = 10u$, $d_{min} \geq u_s$

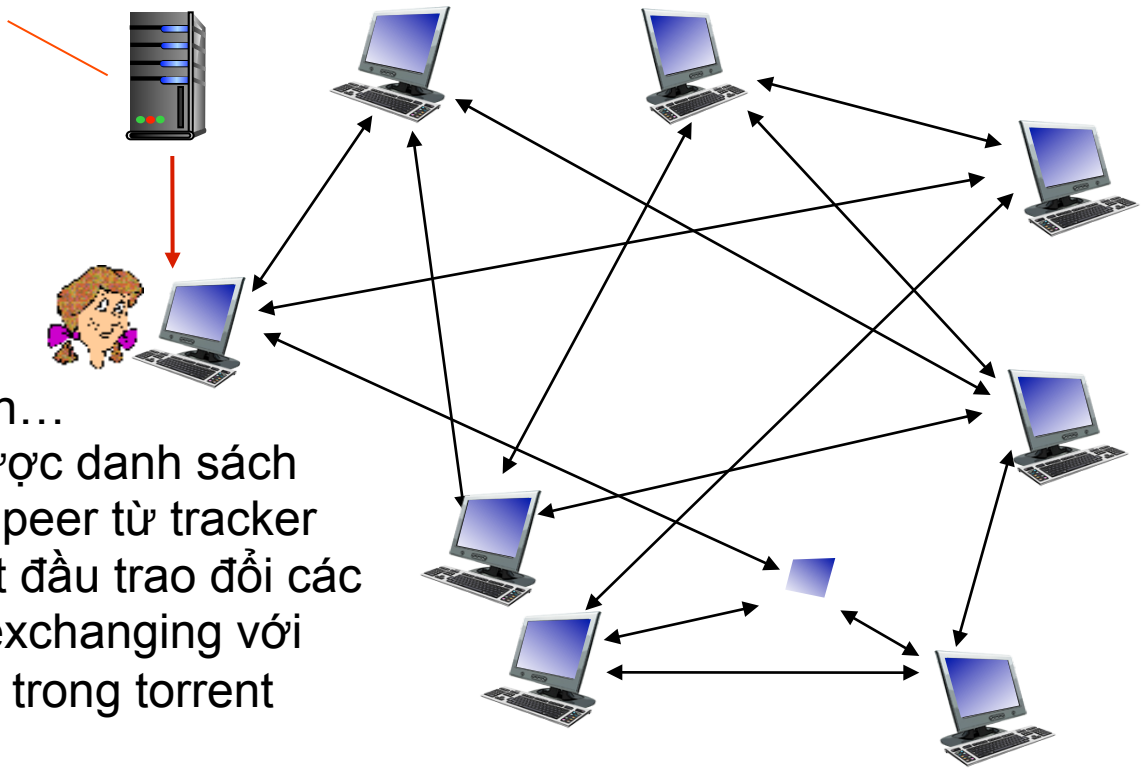


Phân phối file P2P: BitTorrent

- File được chia thành các khối 256Kb
- Các peer trong in torrent gởi/nhận các khối file

tracker: theo dõi các peers tham gia trong torrent

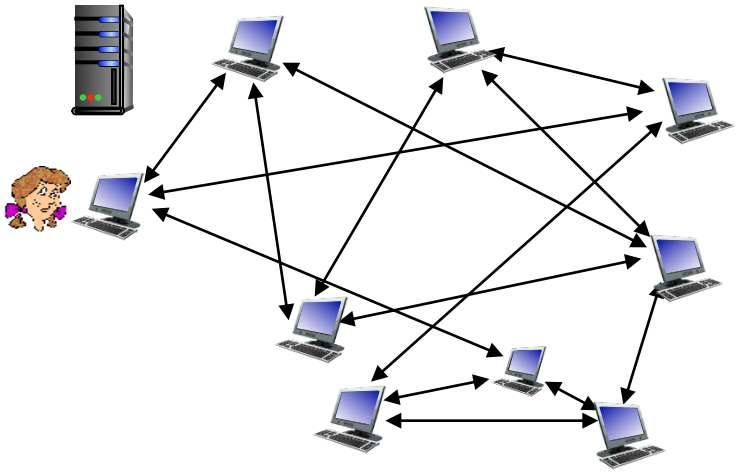
torrent: nhóm các peer trao đổi các khối file



Alice đến…
… có được danh sách của các peer từ tracker
… và bắt đầu trao đổi các khối fileexchanging với các peer trong torrent

Phân phối file P2P: BitTorrent

- Peer tham gia torrent:
  - Không có các khối, nhưng sẽ tích lũy chúng theo thời gian từ các peer khác
  - Đăng ký với tracker để lấy danh sách các peer, kết nối với tập hợp của các peer ("các láng giềng")



- Trong khi tải xuống, peer tải lên các khối dữ liệum mà mình đang có tới các peer khác
- Peer có thể thay đổi các peer mà nó đang trao đổi các khối dữ liệu
- *churn:* peers có thể đến và đi
- Một khi peer có toàn bộ file, nó có thể rời khỏi hoặc ở lại trong torrent

BitTorrent: yêu cầu, gởi các khối file

Yêu cầu các khối:

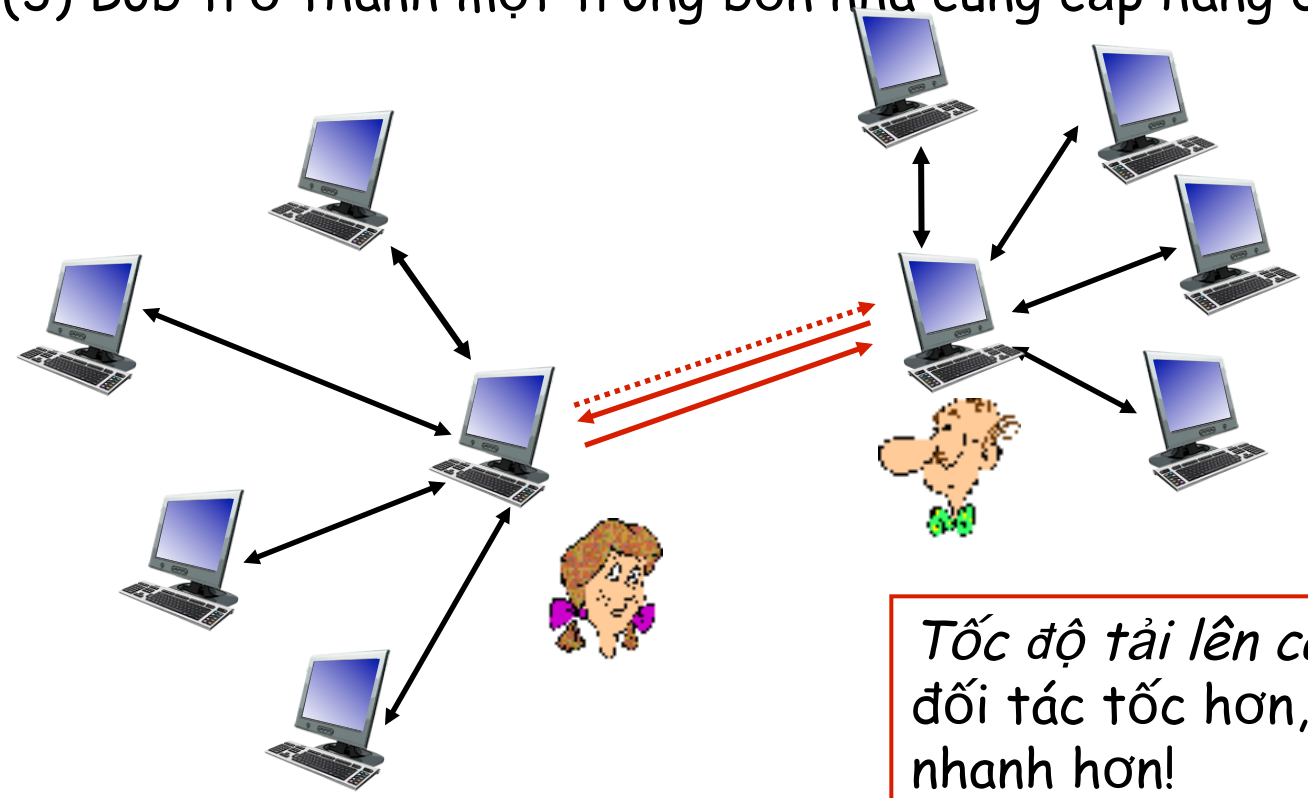
- Tại bất kỳ thời điểm, các peer khác nhau có các tập con khác nhau của các khối file
- Định kỳ, Alice yêu cầu mỗi peer cho danh sách các khối mà các peer có
- Alice yêu cầu các khối đang thiếu từ các peer, hiếm trước

Gởi các khối: tit-for-tat

- Alice gởi các khối cho bốn peer mà hiện tại đang gởi các khối của cô ở tốc độ cao nhất
  - Các peer khác bị thắt lại bởi Alice (không nhận khối từ cô ta)
  - Đánh giá lại top 4 mỗi 10 giây
- Mỗi 30 giây: chọn ngẫu nhiên một peer khác, bắt đầu gởi các khối
  - "optimistically unchokes" peer này
  - Peer mới được chọn có thể tham gia và top 4

BitTorrent: tit-for-tat

(1) Alice "optimistically unchokes" Bob
(2) Alice trở thành một trong bốn nhà cung cấp hàng đầu của Bob's top-four providers; Bob đáp lại
(3) Bob trở thành một trong bốn nhà cung cấp hàng đầu của Alice



Tốc độ tải lên cao hơn: tìm đối tác tốc hơn, lấy file nhanh hơn!

Distributed Hash Table (DHT)

- Bảng Hash
- Mô hình DHT
- Circular DHT and overlay networks
- Peer churn

Cơ sở dữ liệu đơn giản

Cơ sở dữ liệu đơn giản với cặp(key, value):

- key: tên người: số an sinh xã hội

Key	Value
John Washington	132-54-3570
Diana Louise Jones	761-55-3791
Xiaoming Liu	385-41-0902
Rakesh Gopal	441-89-1956
Linda Cohen	217-66-5609
…….	……
Lisa Kobayashi	177-23-0199

- key: tên phim; value: địa chỉ IP

Bảng Hash

- thuận tiện hơn để lưu trữ và tìm kiếm trên đại diện số của key
- key = hash(original key)

Original Key	Key	Value
John Washington	8962458	132-54-3570
Diana Louise Jones	7800356	761-55-3791
Xiaoming Liu	1567109	385-41-0902
Rakesh Gopal	2360012	441-89-1956
Linda Cohen	5430938	217-66-5609
…….		…….
Lisa Kobayashi	9290124	177-23-0199

Distributed Hash Table (DHT)

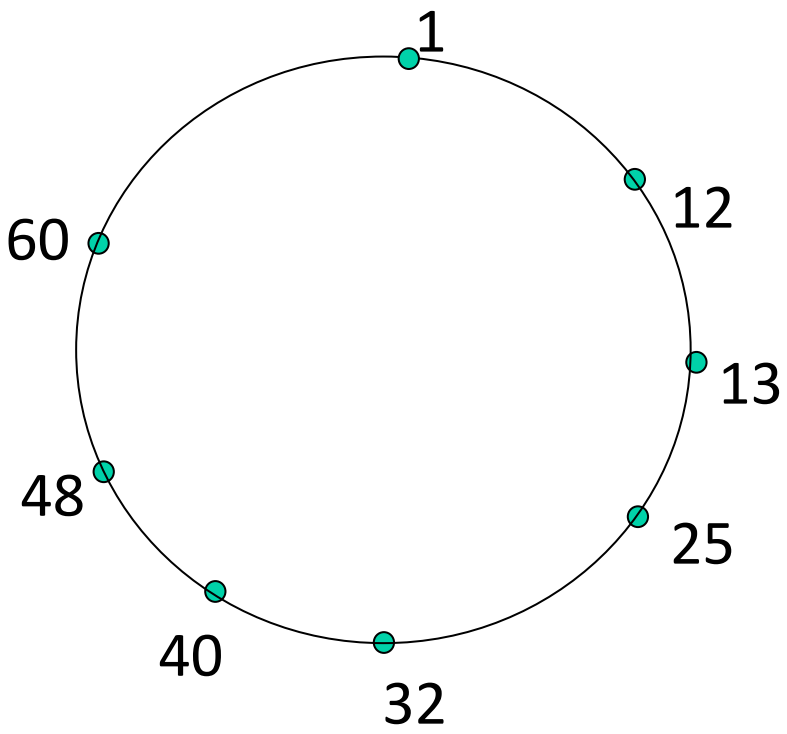
- Phân phối các cặp (key, value) qua hàng triệu các peer
  - Các cặp được phân bố đều trên các peer
- Bất kỳ peer nào cũng có thể truy vấn cơ sở dữ liệu của một key
  - Cơ sở dữ liệu trả về giá trị cho key đó
  - Để giải quyết truy vấn, số lượng nhỏ các thông điệp được trao đổi giữa các peer
- Mỗi peer chỉ biết một số nhỏ các peer khác

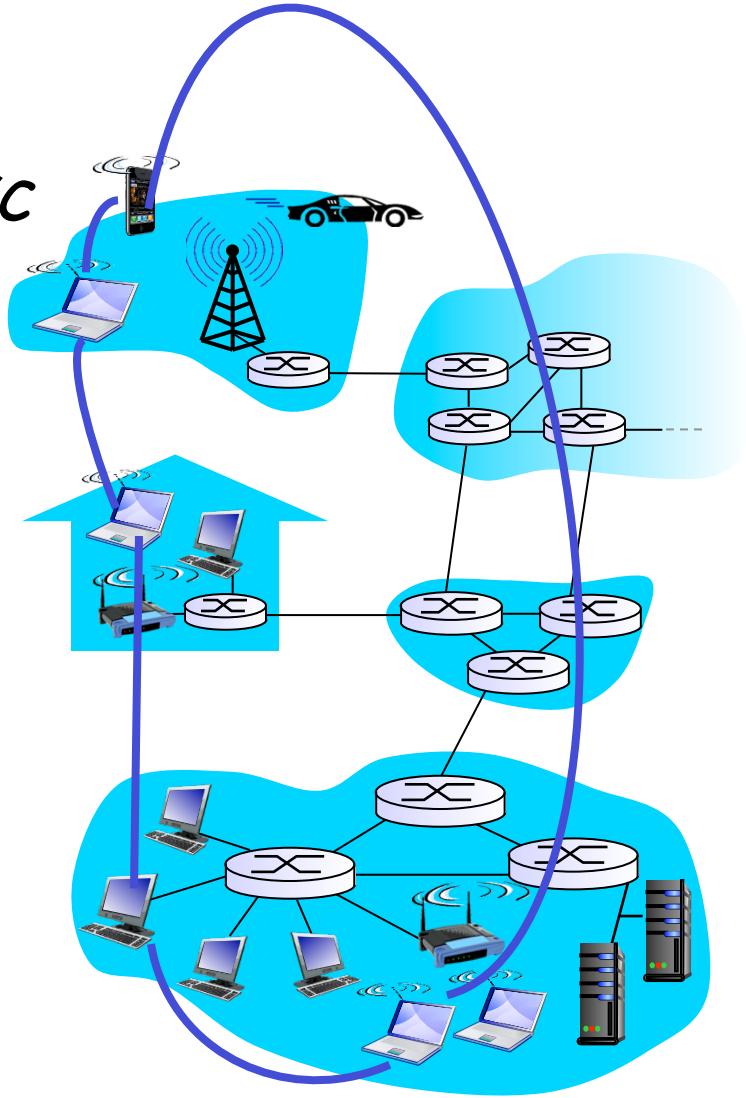
Chỉ định các cặp key-value cho các peer

- Quy tắc: chỉ định cặp key-value đến peer mà có ID *gần nhất (closest)*.
- Quy ước: gần nhất(closest) is *sự kế thừa ngay lập tức (immediate successor)* của khóa (key) đó.
- Ví dụ: không gian ID {0,1,2,3,…,63}
- Giả sử 8 peer: 1,12,13,25,32,40,48,60
  - Nếu key = 51, thì được chỉ định cho peer 60
  - Nếu key = 60, thì được chỉ định cho peer 60
  - Nếu key = 61, thì được chỉ định cho peer 1

DHT vòng tròn

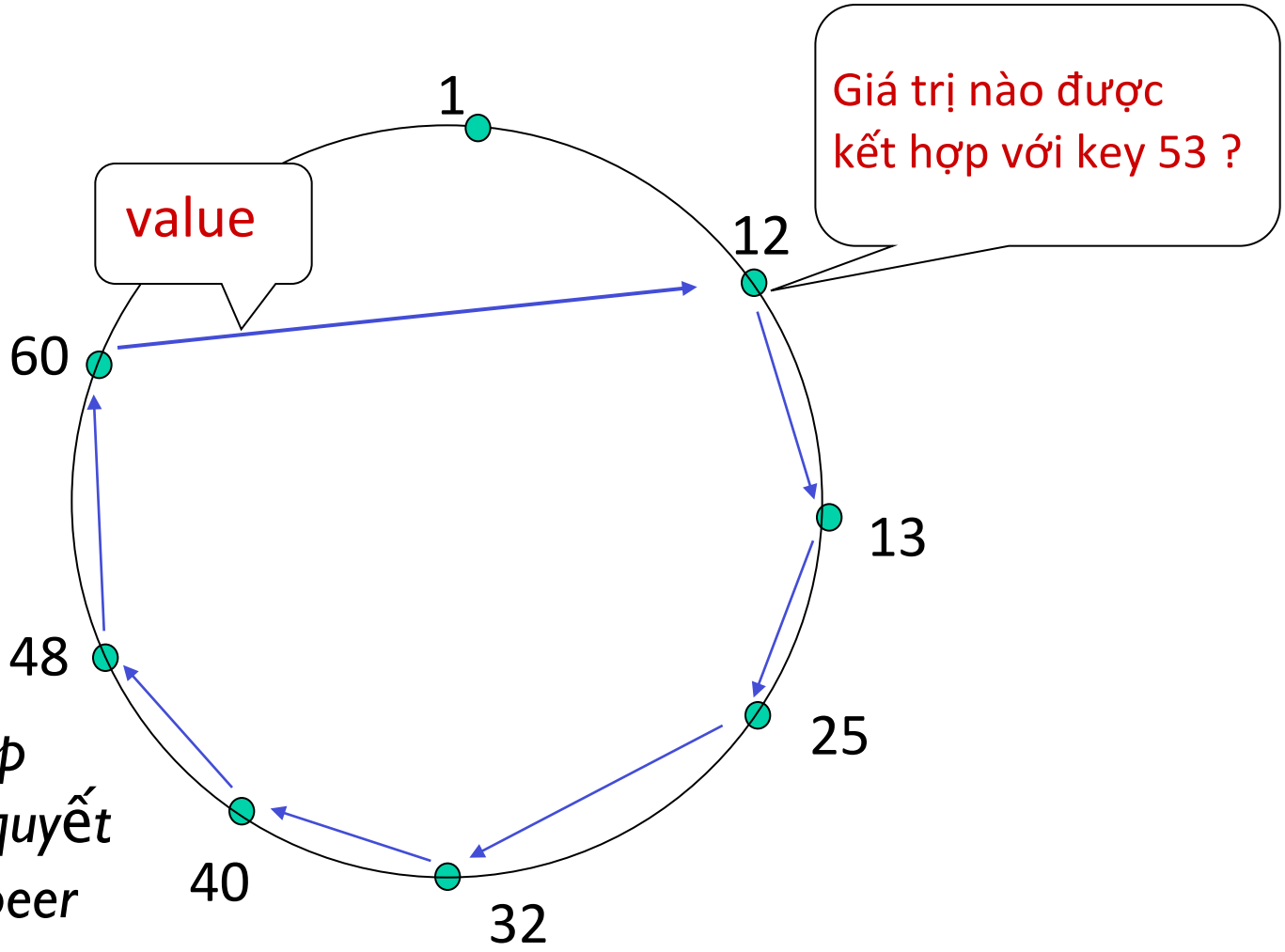
- Mỗi peer *chỉ nhận thức được người lập tức kế nhiệm và người tiền nhiệk*





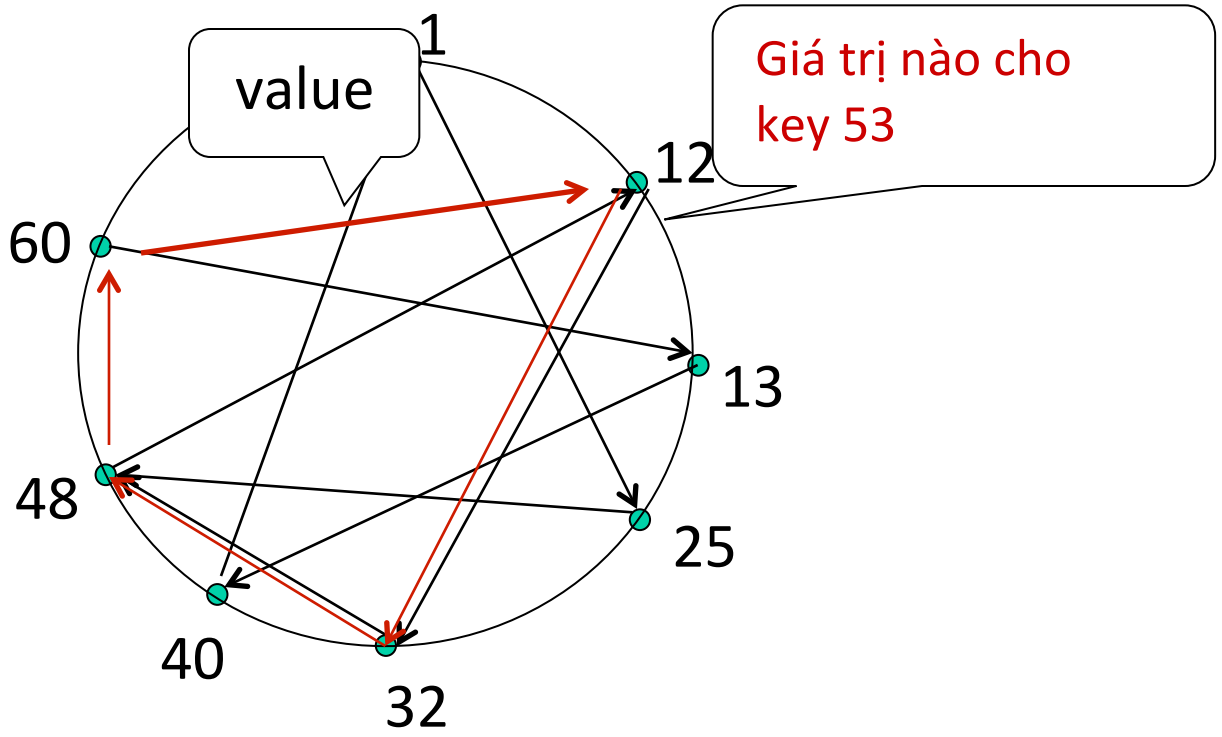
"overlay network"

Giải quyết một truy vấn



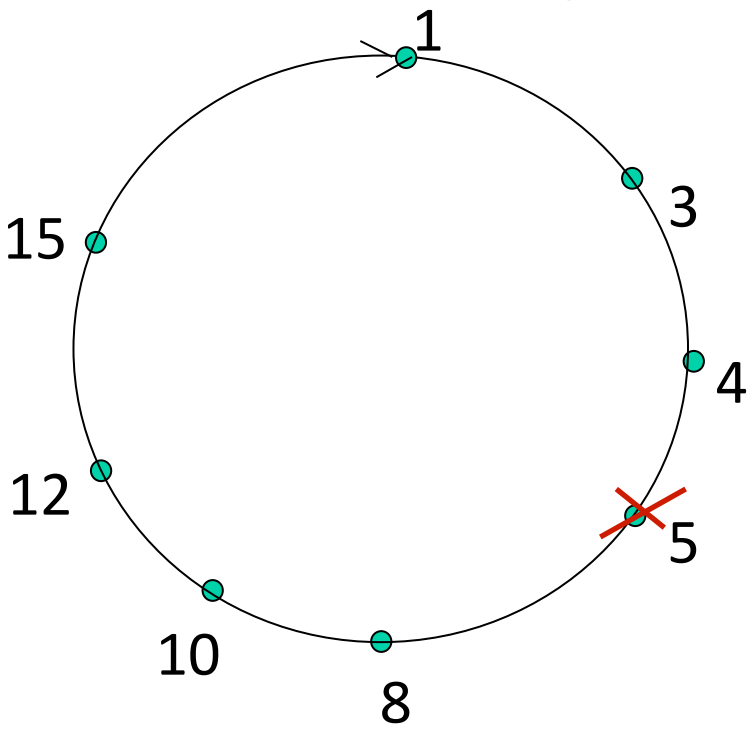
O(N) các thông điệp trung bình để giải quyết truy vấn, khi có N peer

DHT vòng tròn với đường tắt



- Mỗi peer theo dõi đại chỉ IP của người tiền nhiệm, người kế nhiệm, đường tắt.
- Giảm từ 6 còn 3 thông điệp.
- Có thể thiết kế các đường tắt với *O(log N)* láng giềng, *O(log N)* thông điệp trong truy vấn

Peer churn

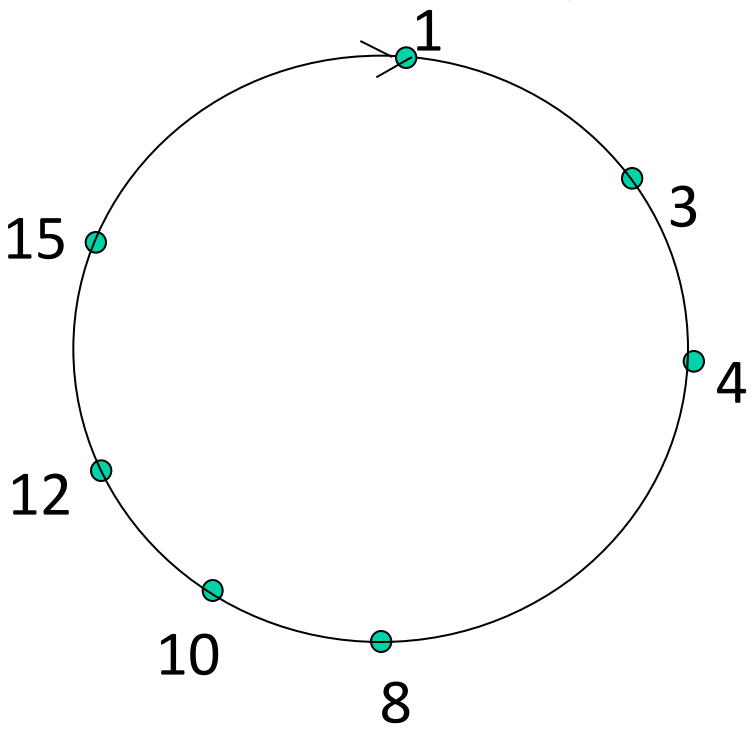


Ví dụ: peer 5 đột ngột rời khỏi

Xử lý peer churn:

- Các peer có thể đến và đi (churn)
- Mỗi peer biết địa chỉ của hai kế nhiệm của nó
- Mỗi peer định kỳ ping hai kế nhiệm của nó để kiểm tra sự tồn tại
- Nếu người vừa kế nhiệm bỏ đi, thì chọn kế nhiệm kế tiếp như là người kế nhiệm tức thời mới

Peer churn



Xử lý peer churn:

- Các peer có thể đến và đi (churn)
- Mỗi peer biết địa chỉ của hai kế nhiệm của nó
- Mỗi peer định kỳ ping hai kế nhiệm của nó để kiểm tra sự tồn tại
- Nếu người vừa kế nhiệm bỏ đi, thì chọn kế nhiệm kế tiếp như là người kế nhiệm tức thời mới

Ví dụ: peer 5 đột ngột rời khỏi

- peer 4 phát hiện sự rời khỏi của peer 5; peer 8 trở thành người kế nhiệm ngay lập tức của nó
- 4 yêu cầu 8 là người kế nhiệm tức thời của nó; người kế nhiệm tức thời của 8 trở thành người kế nhiệm thứ 2 của 4.

Chương 2: Nội dung

2.1 Các nguyên lý của các ứng dụng mạng

2.2 Web và HTTP

2.3 FTP

2.4 electronic mail

- SMTP, POP3, IMAP

2.5 DNS

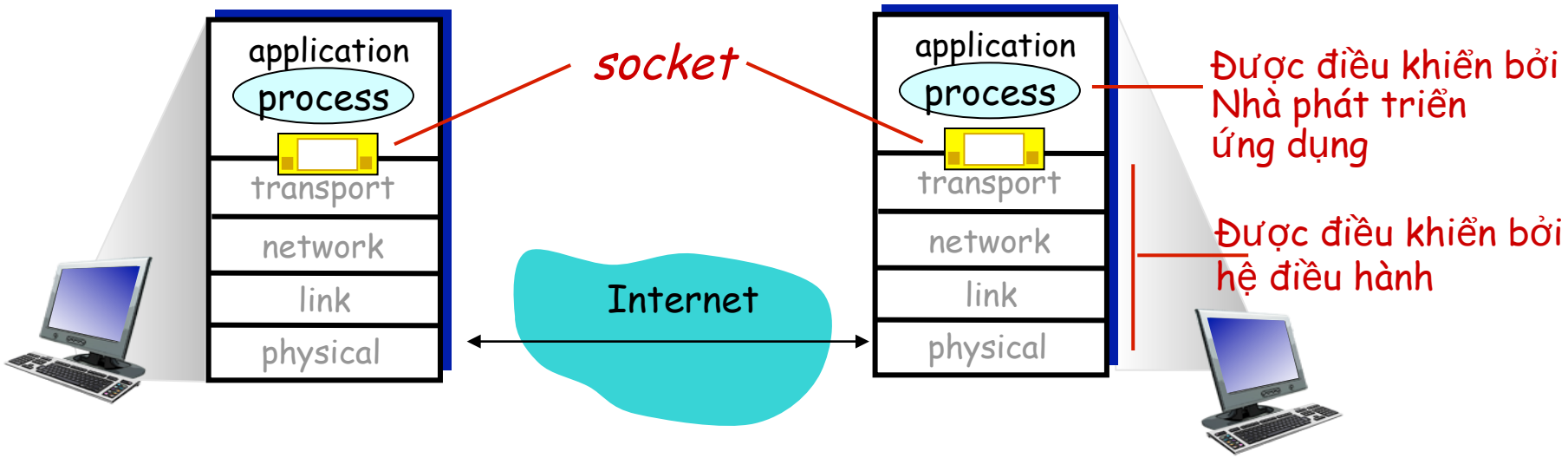
2.6 các ứng dụng P2P

2.7 lập trình socket với UDP và TCP

Lập trình Socket

Mục tiêu: tìm hiểu cách xây dựng các ứng dụng client/server cái mà truyền thông dùng sockets

socket: một cánh cửa giữa tiến trình ứng dụng và giao thức transport end-end



Lập trình Socket

Hai loại socket cho hai dịch vụ transport:

- *UDP:* datagram không tin cậy
- *TCP:* tin cậy, byte được định hướng dòng (stream-oriented)

Ví dụ ứng dụng:

1. Client đọc một dòng các ký tự (dữ liệu) từ bàn phím của nó và gởi dữ liêu đó đến server.
2. server nhận được dữ liệu đó và chuyển đổi các ký tự sang chữ hoa.
3. server gởi dữ liệu đã được sửa đổi cho client ở trên.
4. client này nhận được dữ liệu đã bị sửa đổi và hiển thị dòng đó lên màn hình của nó.

Lập trình Socket với *with UDP*

UDP: không "kết nối" giữa client và server

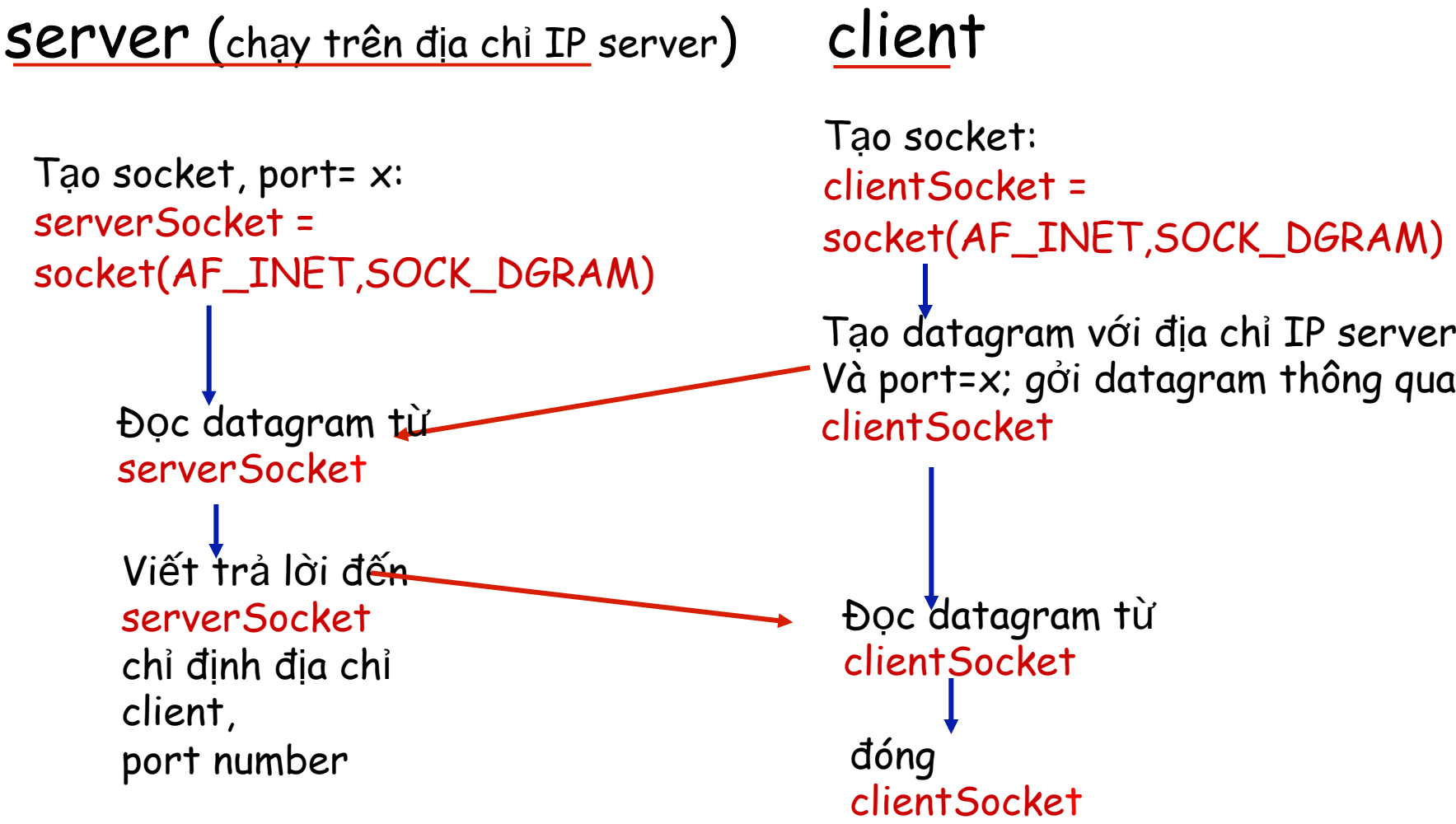
- Không bắt tay trước khi gởi dữ liệu
- Bên gửi chỉ rõ địa chỉ IP đích và số port cho mỗi packet
- Bên nhận lấy địa chỉ IP và số ort của người gởi từ packet được nhận

UDP: dữ liệu được truyền có thể bị mất hoặc được nhận không thứ tự

Quan điểm ứng dụng:

- UDP cung cấp truyền không tin cậy của các nhóm byte ("dâtgrams") giữa client và server

Sự tương tác socket Client/server: UDP



Ví dụ ứng dụng: UDP client

Python UDPClient

Bao gồm thư viện socket của Python'

```
from socket import *
serverName = 'hostname'
serverPort = 12000
```

Tạo socket UDP cho server

```
clientSocket = socket(socket.AF_INET,
                      socket.SOCK_DGRAM)
```

Nhận thông điệp từ bàn phím người dùng

```
message = raw_input('Input lowercase sentence:')
```

Đính kèm tên server, port đến thông điệp; gởi vào tron socket

```
clientSocket.sendto(message,(serverName, serverPort))
```

Đọc các ký tự trả lời từ socket vào chuỗi

```
modifiedMessage, serverAddress =
                      clientSocket.recvfrom(2048)
```

In ra chuỗi được nhận và đóng socket

```
print modifiedMessage
clientSocket.close()
```

Ví dụ ứng dung: UDP server

Python UDPServer

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', serverPort))
print "The server is ready to receive"
while 1:
    message, clientAddress = serverSocket.recvfrom(2048)
    modifiedMessage = message.upper()
    serverSocket.sendto(modifiedMessage, clientAddress)
```

Tạo UDP socket

Đính kèm socket đến số port cục bộ12000

Lặp mãi mãi

Đọc từ UDP socket vào trong thông điệp, lấy địa chỉ IP của client(địa chỉ IP client và port)

Gởi chuỗi chữ hoa trở lại cho client này

Lập trình Socket *với TCP*

client phải tiếp xúc với server

- Tiến trình server phải được chạy trước
- server phải tạo socket (cửa) để mời client đến liên lạc

client tiếp xúc server bằng:

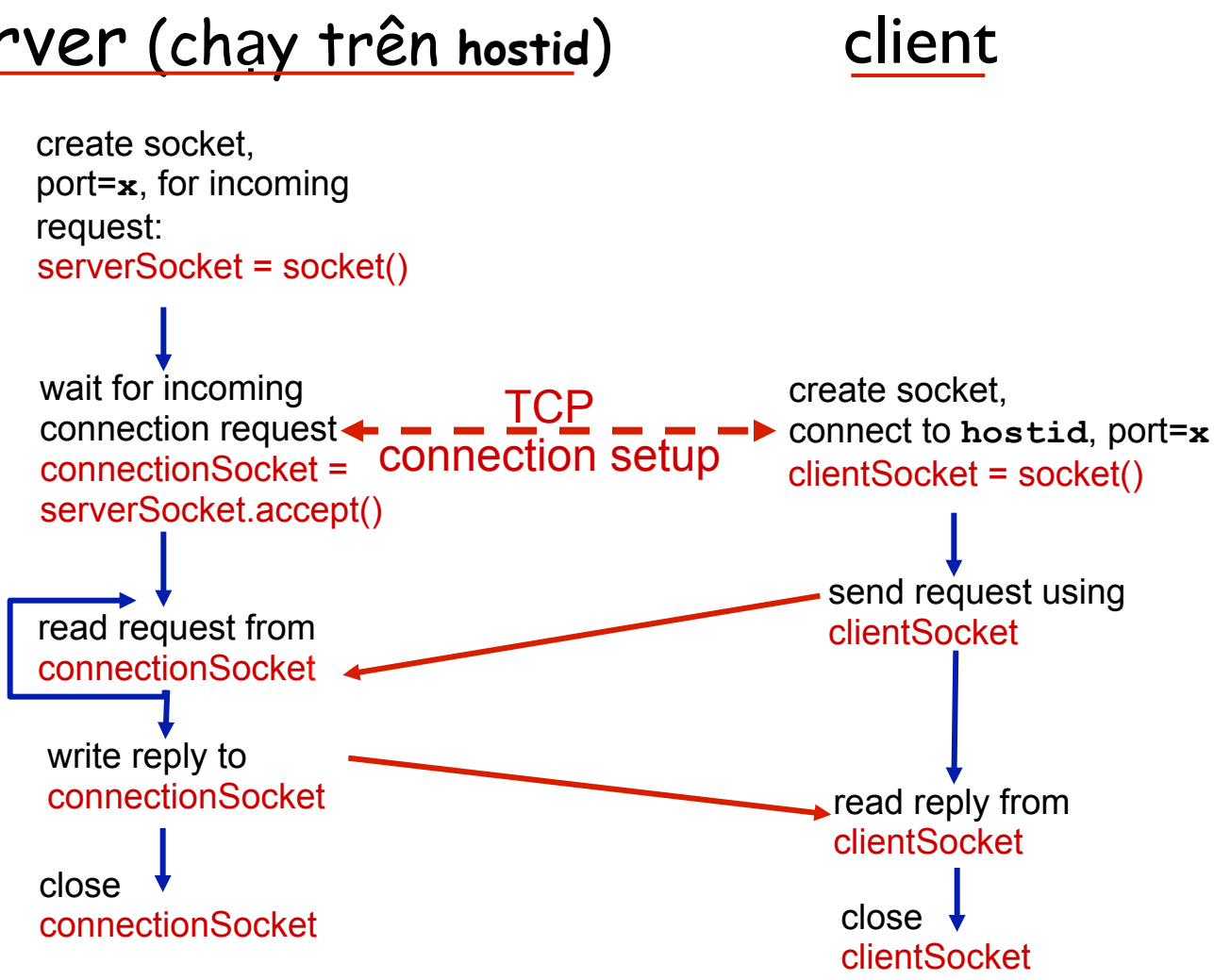
- Tạo socket TCP, xác định địa chỉ IP, số port của tiến trình server
- *Khi client tạo socket:* client TCP thiết lập kết nối đến server TCP

- Khi đã tiếp xúc với client, *server TCP tạo socket mới* cho tiến trình serverđể truyền thông với client đó
  - Cho phép server nói chuyện với nhiều client
  - Số source port được dùng để phân biệt các client (xem tiếp chương 3)

Nhìn dưới góc độ ứng dụng:

TCP cung cấp việc truyền các byte tin cậy và theo thứ tự giữa client và server

Tương tác socket Client/server: TCP



Ví dụ ứng dụng: TCP client

Python TCPClient

```
from socket import *
serverName = 'servername'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
sentence = raw_input('Input lowercase sentence:')
clientSocket.send(sentence)
modifiedSentence = clientSocket.recv(1024)
print 'From Server:', modifiedSentence
clientSocket.close()
```

Tạo TCP socket cho server, port 12000 ở xa

Không cần đính kèm tên server, port

Ví dụ ứng dụng: TCP server

Python TCPServer

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind(('',serverPort))
serverSocket.listen(1)
print 'The server is ready to receive'
while 1:
    connectionSocket, addr = serverSocket.accept()

    sentence = connectionSocket.recv(1024)
    capitalizedSentence = sentence.upper()
    connectionSocket.send(capitalizedSentence)
    connectionSocket.close()
```

Tạo socket TCP chào đón

server bắt đầu lắng nghe các yêu cầu TCP đến

Lặp mãi mãi

server đợi accept() cho yêu cầu đến, socket mới được tạo trở về

Đọc các byte từ socket nhưng không đọc địa chỉ như UDP)

Đóng kết nối đến client này(nhưng không đóng socket chào đón)

Chương 2: tóm tắt

- Các kiến trúc ứng dụng
  - client-server
  - P2P
- Các yêu cầu dịch vụ ứng dụng:
  - Độ tin cậy, băng thông, độ trễ
- Mô hình dịch vụ vận chuyển Internet
  - Kết nối định hướng, tin cậy: TCP
  - Không tin c, datagrams: UDP
- Các giao thức:
  - HTTP
  - FTP
  - SMTP, POP, IMAP
  - DNS
  - P2P: BitTorrent, DHT
- Lập trình socket : TCP, UDP sockets

Chương 2: tóm tắt

Quan trọng: tìm hiểu về các giao thức!

- trao đổi thông điệp yêu cầu /trả lời điển hình:
  - client yêu cầu thông tin hoặc dịch vụ
  - server đáp ứng với dữ liệu, mã trạng thái
- Các định dạng thông điệp:
  - headers: các trường cho biết thông tin về dữ liệu
  - data: thông tin để truyền thông

Các chủ đề quan trọng:

- Điều khiển với các thông điệp dữ liệu
  - in-band, out-of-band
- Tập trung và không tập trung
- Không trạng thái và có trạng thái
- Truyền tin cậy và không tin cậy
- "sự phức tạp tại mạng biên"