

# Chương 2

# Tầng Ứng dụng

# (Application Layer)

## A note on the use of these PowerPoint slides:

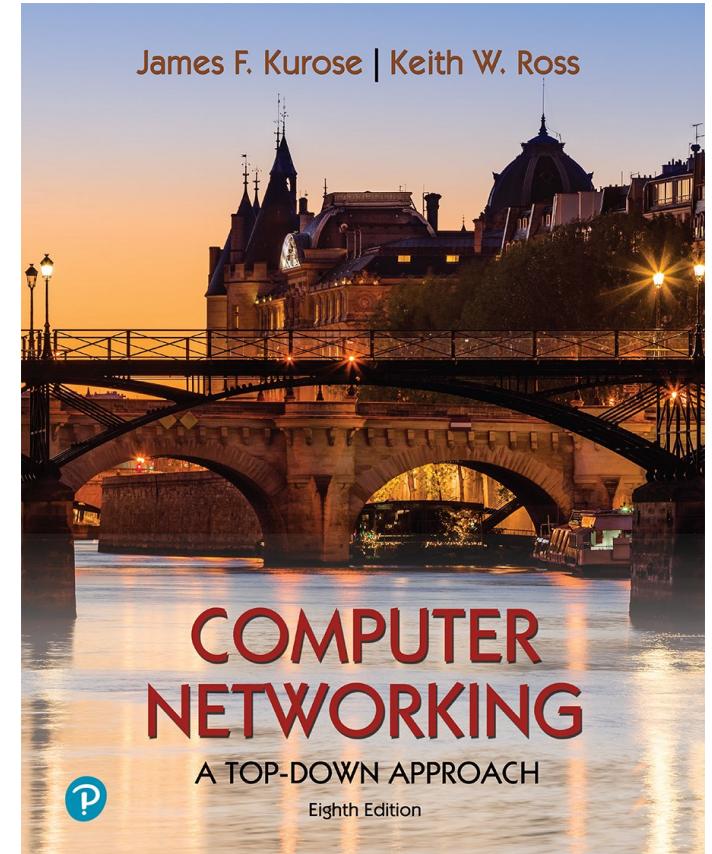
We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use,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

For a revision history, see the slide note for this page.

Thanks and enjoy! JFK/KWR

All material copyright 1996-2023  
J.F Kurose and K.W. Ross, All Rights Reserved



**Computer Networking:  
A Top-Down Approach**  
8<sup>th</sup> edition  
Jim Kurose, Keith Ross  
Pearson, 2020

# Tầng Ứng dụng – Tổng quan



- Nguyên lý của các ứng dụng mạng
- Web và HTTP
- E-mail, SMTP, IMAP
- Hệ thống phân giải tên miền DNS
- Các ứng dụng P2P
- Lập trình socket với UDP và TCP





# Tầng Ứng dụng: Tổng quan

## Mục tiêu:

- **Khái niệm** và khía cạnh **triển khai** của các giao thức tầng ứng dụng
  - Mô hình dịch vụ tầng vận chuyển
  - Mô hình client-server (máy khách – máy chủ)
  - Mô hình peer-to-peer (ngang hàng)

## Tìm hiểu

- Tìm hiểu về các giao thức thông qua việc xem xét các giao thức phổ biến của Tầng Ứng dụng:
  - HTTP
  - SMTP, IMAP
  - DNS
- Lập trình ứng dụng mạng
  - Socket API



# Một số ứng dụng mạng

- Mạng xã hội
- Web
- Nhắn tin, chat
- E-mail
- Trò chơi trực tuyến với nhiều người cùng tham gia
- Truyền hình trực tuyến (streaming stored video – Vd: YouTube, Hulu, Netflix)
- Chia sẻ file P2P
- Đàm thoại trên mạng IP (Ví dụ Skype)
- Hội thảo video thời gian thực (Ví dụ Zoom)
- Internet search
- Đăng nhập từ xa
- ...

Q: *Ứng dụng ưa thích của bạn?*



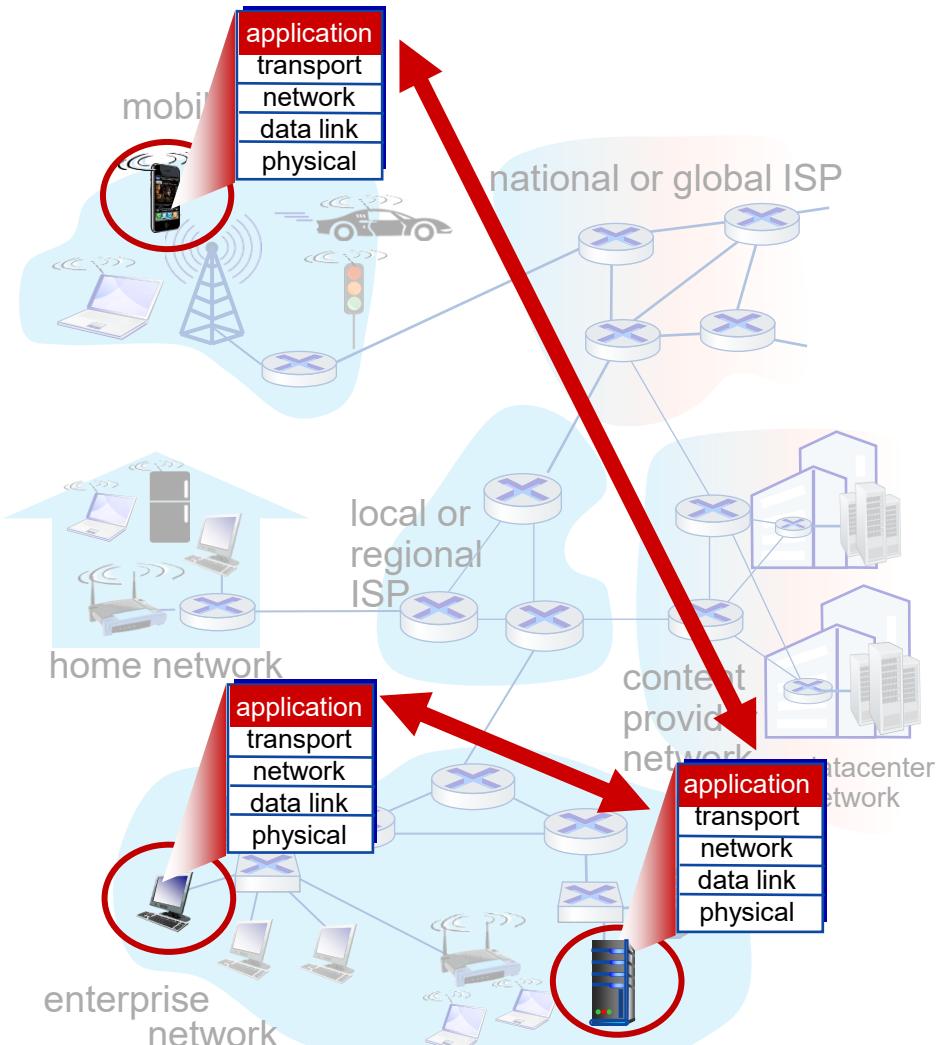
# Tạo một ứng dụng mạng

## Viết chương trình:

- Chạy trên các hệ thống đầu cuối (khác nhau)
- Giao tiếp qua mạng
- Ví dụ: phần mềm máy chủ web giao tiếp với phần mềm duyệt web

## Không cần viết phần mềm cho các thiết bị trong mạng lõi

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





# Mô hình Client – Server (Máy khách – máy chủ)

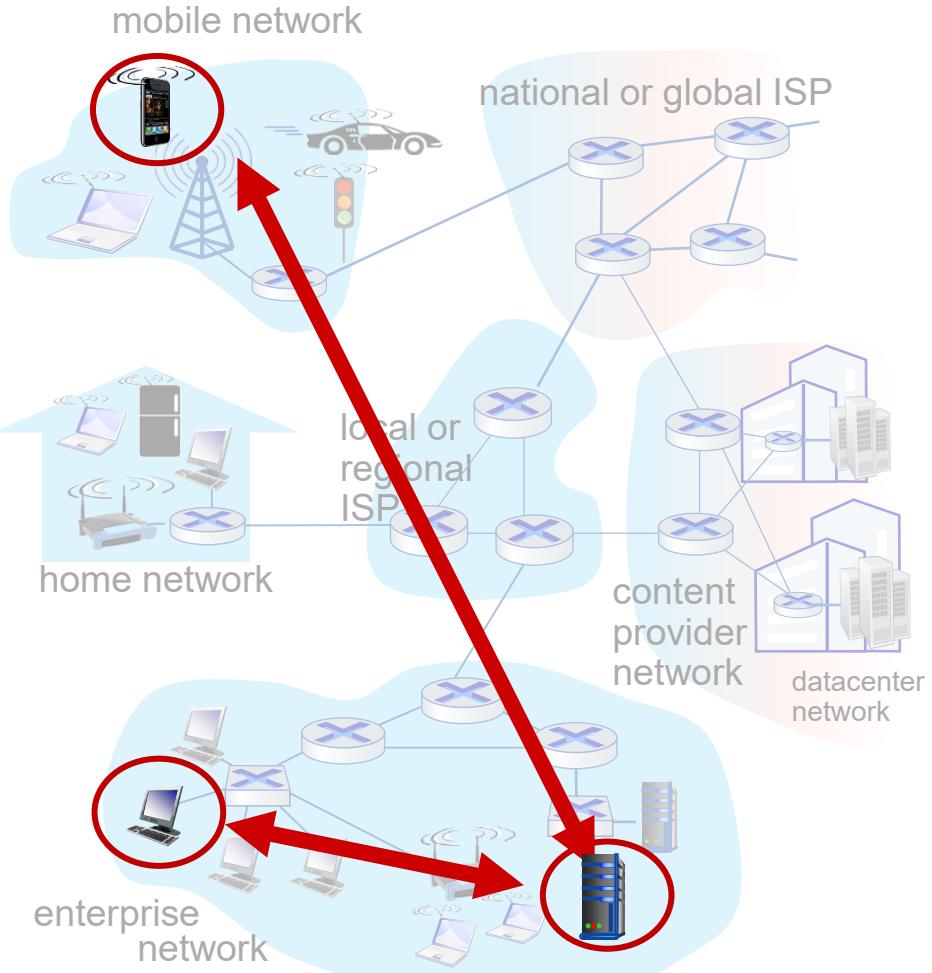
- **Máy chủ (Server):**

- Luôn luôn hoạt động
- Địa chỉ IP cố định
- Thường tổ chức thành các trung tâm dữ liệu để mở rộng quy mô

- **Máy khách (Client):**

- Liên lạc, giao tiếp với máy chủ
- Có thể kết nối không liên tục
- Có thể thay đổi địa chỉ IP
- Không giao tiếp trực tiếp với các máy khách khác

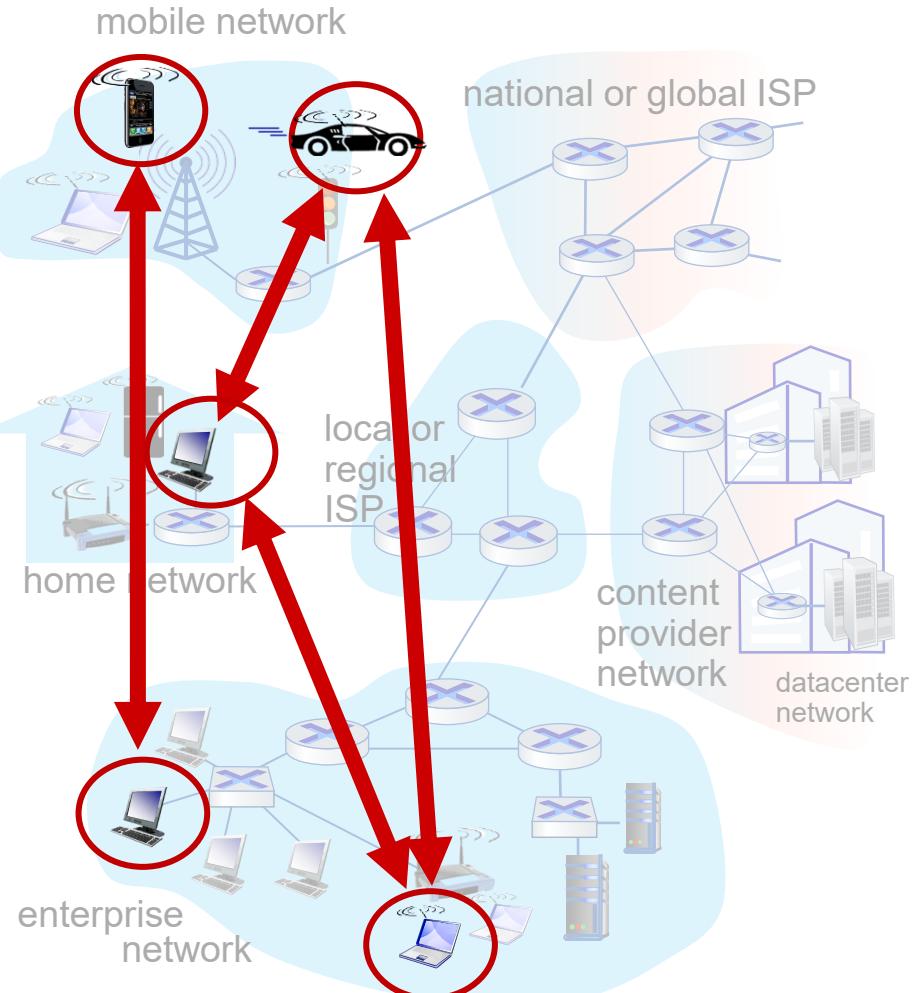
- **Ví dụ: HTTP, IMAP, FTP**





# Kiến trúc mạng ngang hàng (peer-to-peer)

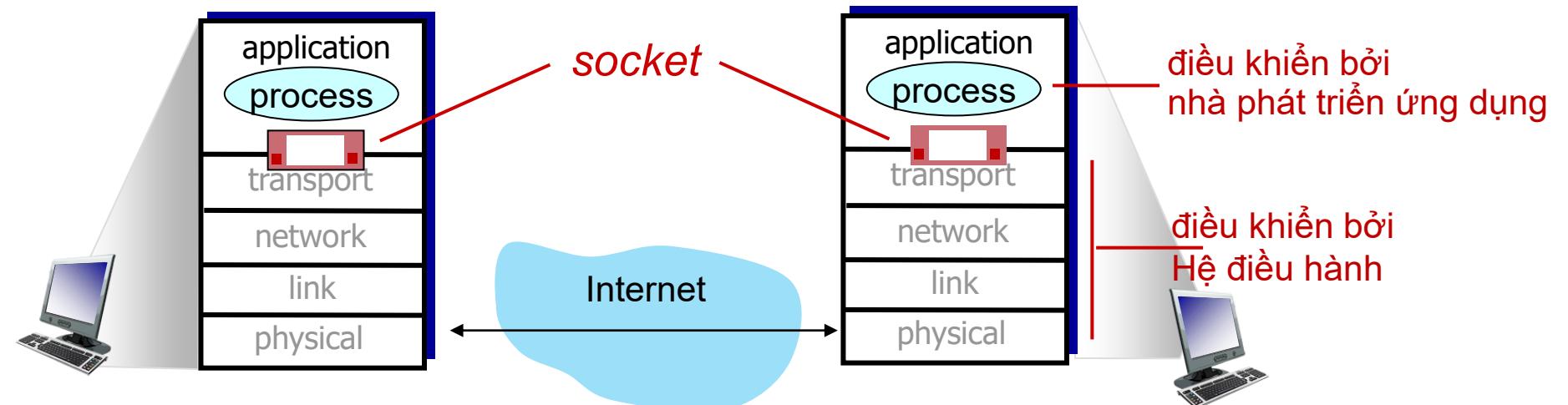
- **Không có** máy chủ luôn hoạt động
- Các hệ thống đầu cuối bất kỳ giao tiếp trực tiếp với nhau
- Các peer yêu cầu dịch vụ từ các peer khác và cung cấp dịch vụ ngược lại cho các peer khác
- **Khả năng mở rộng** – các peer mới cung cấp thêm dịch vụ mới, cũng như có thêm nhu cầu mới về dịch vụ
- Các peer có thể kết nối không liên tục và thay đổi địa chỉ IP
  - Quản lý phức tạp
- Ví dụ: Chia sẻ tập tin P2P [BitTorrent]





# Socket

- Tiến trình **gửi/nhận** thông điệp **đến/từ** socket của nó
- Socket hoạt động tương tựa như cánh cửa
  - Tiến trình gửi đẩy thông điệp ra khỏi cửa
  - Tiến trình gửi dựa trên hạ tầng vận chuyển bên kia của cánh cửa để phân phối thông điệp đến socket tại tiến trình nhận
- Hai socket có liên quan: một socket ở mỗi bên





# Xác định tiến trình

- Để nhận các thông điệp, tiến trình phải có định danh *identifier*
- Thiết bị đầu cuối có một địa chỉ IP 32-bit duy nhất
- Q: địa chỉ IP của hệ thống đầu cuối mà tiến trình chạy trên đó có đủ để xác định tiến trình đó hay không?
  - A: không, có nhiều tiến trình có thể được chạy trên cùng một hệ thống đầu cuối

- *Định danh (identifier)* bao gồm cả địa chỉ IP và số hiệu cổng (port) liên kết với tiến trình trên hệ thống đầu cuối.
- Ví dụ về số hiệu cổng:
  - Máy chủ HTTP (Web): 80
  - Máy chủ thư điện tử: 25
- Để gửi thông điệp HTTP đến máy chủ web gaia.cs.umass.edu:
  - Địa chỉ IP: 128.119.245.12
  - Số hiệu cổng: 80
- Còn nữa...



# Một giao thức tầng Ứng dụng định nghĩa:

- Các loại thông điệp được trao đổi
  - Ví dụ: yêu cầu (request), phản hồi (response)
- Cú pháp thông điệp:
  - Các trường trong thông điệp và cách 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 thức các tiến trình gửi và phản hồi các thông điệp
- Các giao thức (protocol) mở:
  - Được định nghĩa trong các RFCs, mọi người đều có quyền truy cập vào định nghĩa giao thức
  - Cho phép khả năng tương tác
  - Ví dụ: HTTP, SMTP
- Các giao thức độc quyền:
  - Ví dụ: Skype, Zoom



# Dịch vụ vận chuyển nào mà một ứng dụng cần?

- Toàn vẹn dữ liệu (Data integrity)

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

- Định thì (Timing)

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

- Thông lượng (Throughput)

- Một số ứng dụng (ví dụ: đa phương tiện - multimedia) yêu cầu thông lượng tối thiểu để đạt được hiệu quả
- Một số ứng dụng khác (“ứng dụng mềm dẻo - elastic apps”) sử dụng bất kỳ thông lượng nào mà nó nhận

- An ninh (Security)

- Mã hóa, toàn vẹn dữ liệu,...

# **Yêu cầu dịch vụ vận chuyển: các ứng dụng phổ biến**



<b>Ứng dụng</b>	<b>Mất dữ liệu</b>	<b>Thông lượng</b>	<b>Độ nhạy thời gian</b>
Truyền/tải tập tin	Không	Mềm dẻo	Không
Thư điện tử	Không	Mềm dẻo	Không
Tài nguyên web	Không	Mềm dẻo	Không
Audio/video thời gian thực	Chịu lỗi	audio: 5Kbps-1Mbps video: 10Kbps-5Mbps	Có, 10's msec
streaming audio/video	Chịu lỗi	Như trên	Có, few secs
Game tương tác	Chịu lỗi	Kbps+	Có, 10's msec
Nhắn tin	Không	Mềm dẻo	Có và không



# Các dịch vụ giao thức vận chuyển Internet

## Dịch vụ TCP:

- ✓ *Truyền tải tin cậy (reliable transport)* giữa các tiến trình gửi và nhận
- ✓ *Điều khiển luồng (flow control)*: bên gửi không áp đảo (gửi quá khả năng) bên nhận
- ✓ *Điều khiển tắc nghẽn (congestion control)*: điều tiết bên gửi khi mạng quá tải
- ✓ *Hướng kết nối (connection-oriented)*: thiết lập kết nối giữa bên gửi và bên nhận
- ✗ *Không cung cấp*: định thì, thông lượng tối thiểu, an ninh

## Dịch vụ UDP:

- ✓ *Truyền tải không tin cậy* giữa tiến trình gửi và nhận
- ✗ *Không cung cấp*: tính tin cậy, điều khiển luồng, điều khiển tắc nghẽn, định thì, thông lượng đảm bảo, an ninh hay thiết lập kết nối

Q: Tại sao phải quan tâm?

Tại sao có UDP

# Ứng dụng Internet và các giao thức vận chuyển



<b>Ứng dụng</b>	<b>Giao thức Tầng Ứng dụng</b>	<b>Giao thức Vận chuyển</b>
Truyền/tải tập tin	FTP [RFC 959]	TCP
Thư điện tử	SMTP [RFC 5321]	TCP
Tài nguyên Web	HTTP [RFC 7230, 9110]	TCP
Điện thoại Internet	SIP [RFC 3261], RTP [RFC 3550], hoặc các giao thức độc quyền	TCP or UDP
Streaming audio/video	HTTP [RFC 7230], DASH	TCP
Game tương tác	WOW, FPS (proprietary)	UDP or TCP



- TCP và UDP Socket:
  - Không mã hóa
  - Dữ liệu chưa được mã hóa được gửi vào socket đi qua Internet dưới dạng nguyên bản (!)
- Transport Layer Security (TLS)
  - Cung cấp các kết nối TCP có mã hóa
  - Toàn vẹn dữ liệu
  - Chứng thực đầu cuối
- TLS được triển khai ở Tầng Ứng dụng
  - Các ứng dụng sử dụng các thư viện TLS, qua đó làm việc với TCP
  - Dữ liệu dạng nguyên bản được gửi vào các socket đi qua Internet ở dạng được mã hóa
  - Xem thêm ở các chương sau

# Tầng Ứng dụng – Tổng quan



- Nguyên lý của các ứng dụng mạng
- **Web và HTTP**
- E-mail, SMTP, IMAP
- Hệ thống phân giải tên miền DNS
- Các ứng dụng P2P
- Lập trình socket với UDP và TCP





## ○ Trước tiên, ôn tập lại...

- Trang web bao gồm các *đối tượng* (objects), có thể được lưu trữ ở các máy chủ Web khác nhau
- Đối tượng có thể là tập tin HTML, các hình ảnh JPEG, các tập tin audio,...
- Trang web bao gồm tập tin HTML cơ bản chứa tham chiếu đến nhiều đối tượng, mỗi đối tượng được đánh địa chỉ bởi một URL

www.someschool.edu/someDept/pic.gif

Tên máy

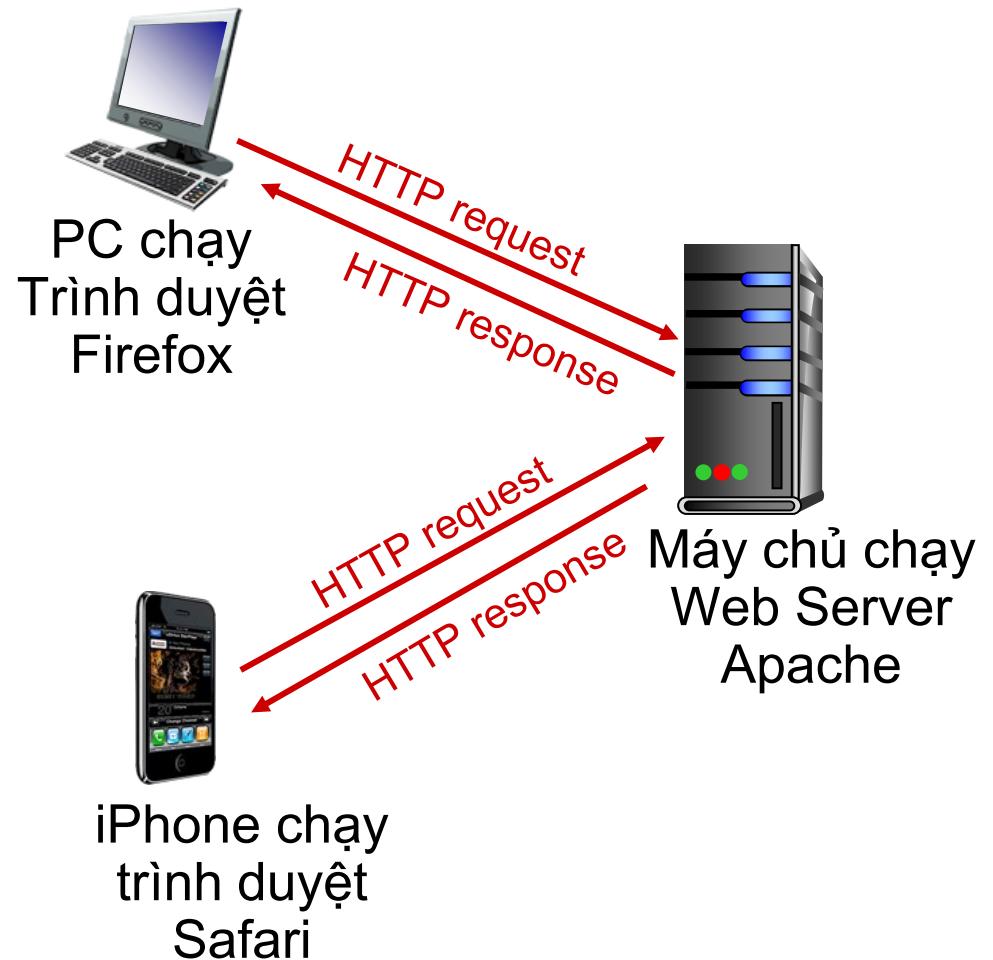
Đường dẫn



# Tổng quan về HTTP

- HTTP: hypertext transfer protocol

- Giao thức tầng ứng dụng cho Web
- Mô hình Máy khách – máy chủ (Client – Server)
  - **Client**: trình duyệt – yêu cầu, tiếp nhận, (sử dụng giao thức HTTP) và hiển thị các đối tượng Web
  - **Server**: máy chủ web gửi (sử dụng giao thức HTTP) các đối tượng trong phản hồi của các yêu cầu





# Tổng quan về HTTP (tt)

- Dùng TCP:
  - Máy khách khởi tạo kết nối TCP (tạo socket) đến cổng 80 của máy chủ
  - Máy chủ chấp nhận kết nối TCP từ máy khách
  - Các thông điệp HTTP (thông điệp thuộc giao thức Tầng ứng dụng) được trao đổi giữa trình duyệt (HTTP client) và máy chủ web (HTTP server)
  - Kết nối TCP được đóng
- HTTP “không lưu trạng thái”
  - Máy chủ không duy trì thông tin về các yêu cầu trước đó của máy khách

*Ngoài lề*

*Các giao thức nào duy trì “trạng thái” thì phức tạp!*

- Lịch sử trước đó (trạng thái) phải được duy trì
- Nếu máy chủ/máy khách bị sự cố, cách nhìn về “trạng thái” của nó có thể bị mâu thuẫn, phải được điều chỉnh



# Các kết nối HTTP: 2 loại

- HTTP không bền vững
  - 1. Kết nối TCP được mở
  - 2. Chỉ tối đa một đối tượng được gửi qua kết nối TCP
  - 3. Kết nối TCP đóng lại
- Tải nhiều đối tượng yêu cầu nhiều kết nối
- HTTP bền vững
  - Kết nối TCP được mở đến một máy chủ (server)
  - Nhiều đối tượng có thể được gửi thông qua *đa số* một kết nối TCP giữa máy khách (client) và máy chủ đó
  - Kết nối TCP đóng lại

# HTTP không bền vững: Ví dụ



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 với HTTP server tại `www.someSchool.edu` trên cổng 80



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

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

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

time ↓



# HTTP không bền vững: Ví dụ (tt)

- Giả sử người dùng vào URL như sau:

`www.someSchool.edu/someDepartment/home.index`

(chứa text, tham chiếu đến 10 hình jpeg)



5. HTTP client nhận được thông báo phản hồi có chứa file html, hiển thị html. Phân tích cú pháp tệp html, tìm 10 đối tượng jpeg được tham chiếu



4. HTTP server đóng kết nối TCP.



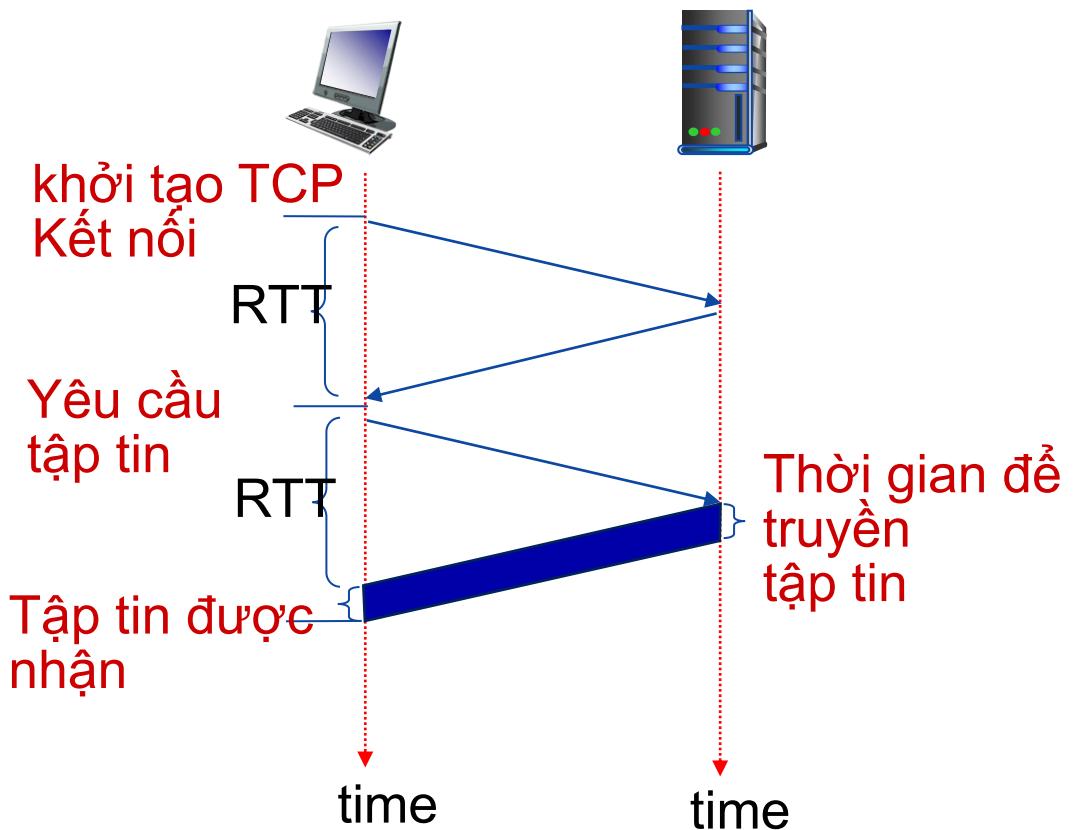
6. Các bước 1-5 lặp lại cho mỗi đối tượng trong số 10 đối tượng jpeg

time  
↓



# HTTP không bền vững: thời gian đáp ứng

- RTT – Round Trip Time (định nghĩa):
  - Thời gian cho một gói nhỏ di chuyển từ máy khách đến máy chủ và ngược lại
- Thời gian đáp ứng HTTP (trên mỗi đối tượng):
  - 1 RTT để bắt đầu kết nối TCP
  - 1 RTT cho yêu cầu HTTP và vài byte phản hồi HTTP đầu tiên trả về
  - Thời gian truyền đối tượng / tập tin



*Thời gian đáp ứng HTTP không bền vững = 2RTT + thời gian truyền tập tin*



# HTTP bền vững (HTTP 1.1)

## ○ Vấn đề với HTTP không bền vững

- Yêu cầu 2 RTT cho mỗi đối tượng
- Chi phí hệ điều hành cho mỗi kết nối TCP
- Trình duyệt thường mở nhiều kết nối TCP song song để tìm nạp song song các đối tượng được tham chiếu

## ○ HTTP bền vững (HTTP1.1):

- Máy chủ để kết nối mở sau khi gửi phản hồi
- Các thông điệp HTTP tiếp theo giữa cùng một máy khách/máy chủ được gửi qua kết nối đã mở
- Client gửi request ngay khi gặp một đối tượng được tham chiếu
- Chỉ cần một RTT cho tất cả các đối tượng được tham chiếu (cắt giảm một nửa thời gian phản hồi)



# Thông điệp yêu cầu HTTP

- Hai loại thông điệp HTTP: **yêu cầu, phản hồi**
- Thông điệp yêu cầu HTTP:
  - ASCII (định dạng con người có thể đọc được)

Dòng yêu cầu  
( Các lệnh GET,  
POST, HEAD)

Các dòng  
header

Ký tự xuống dòng,  
về đầu dòng mới chỉ  
điểm cuối cùng  
của thông điệp

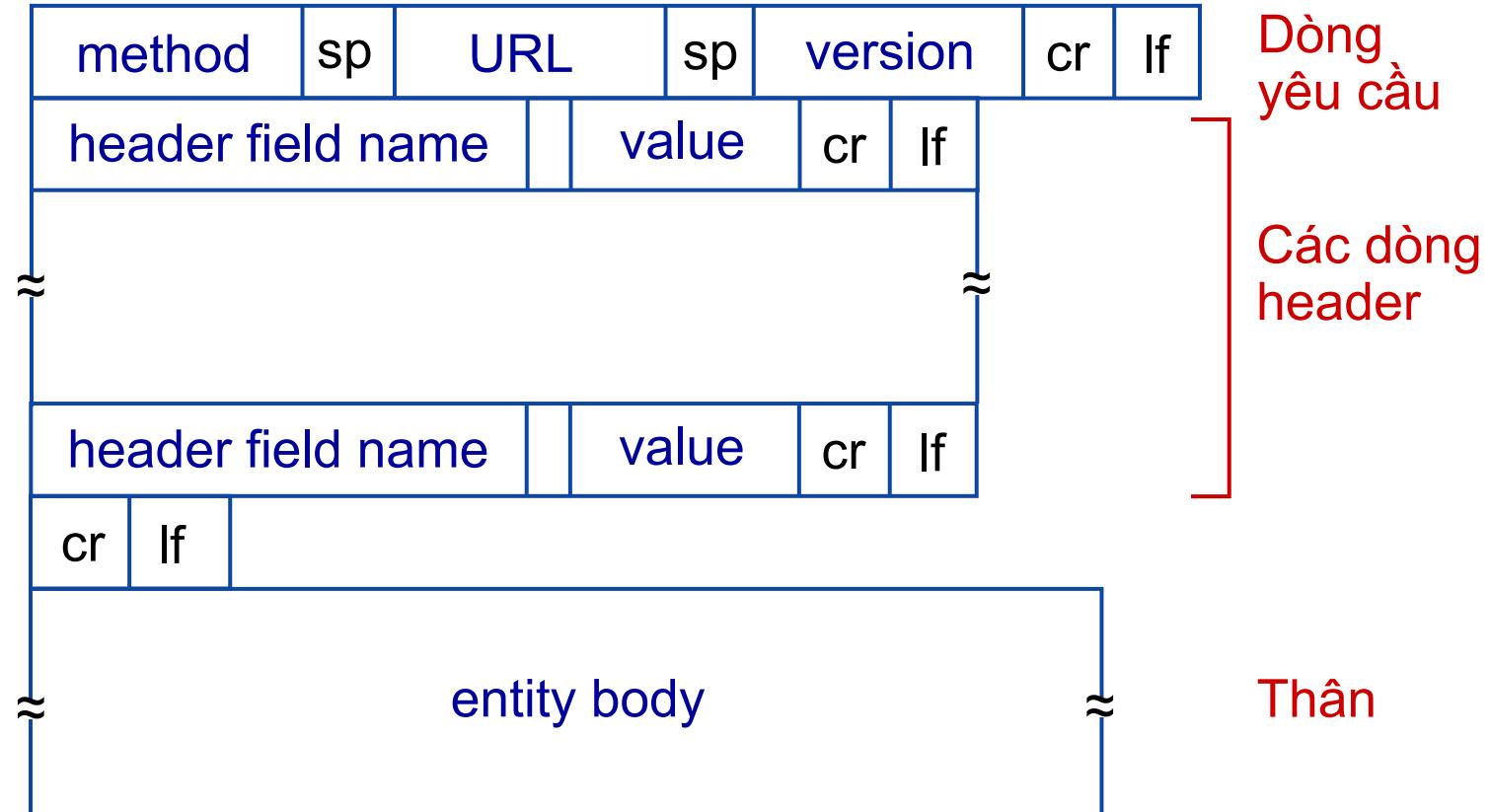
```
GET /index.html HTTP/1.1\r\n
Host: www-net.cs.umass.edu\r\n
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X
  10.15; rv:80.0) Gecko/20100101 Firefox/80.0 \r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Connection: keep-alive\r\n
\r\n
```

Ký tự xuống dòng  
Ký tự về đầu dòng

\* Check out the online interactive exercises for more  
examples: [http://gaia.cs.umass.edu/kurose\\_ross/interactive/](http://gaia.cs.umass.edu/kurose_ross/interactive/)



# Thông điệp yêu cầu HTTP: định dạng chung





# Các thông điệp yêu cầu HTTP khác

## ○ Phương thức POST:

- Trang web thường bao gồm form input
- Dữ liệu của người dùng được gửi từ máy khách đến máy chủ trong nội dung phần thân (body) của thông điệp yêu cầu HTTP POST

## ○ Phương thức GET (để gửi dữ liệu đến máy chủ):

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

[www.somesite.com/animalsearch?monkeys&banana](http://www.somesite.com/animalsearch?monkeys&banana)

## ○ Phương thức HEAD:

- Các dòng header sẽ được trả về nếu URL được chỉ định được yêu cầu bằng phương thức HTTP HEAD.

## ○ Phương thức PUT:

- Tải tập tin (đối tượng) mới lên máy chủ
- Thay thế hoàn toàn tập tin tồn tại tại URL được chỉ định bằng nội dung trong nội dung thực thể của thông điệp yêu cầu HTTP PUT



# Thông điệp phản hồi HTTP

Dòng trạng thái  
(Giao thức  
mã trạng thái  
Cụm từ trạng thái)

Các dòng  
Tiêu đề

Dữ liệu, ví dụ: Tập tin  
HTML được yêu cầu

HTTP/1.1 200 OK  
Date: Tue, 08 Sep 2020 00:53:20 GMT  
Server: Apache/2.4.6 (CentOS)  
OpenSSL/1.0.2k-fips PHP/7.4.9  
mod\_perl/2.0.11 Perl/v5.16.3  
Last-Modified: Tue, 01 Mar 2016 18:57:50 GMT  
ETag: "a5b-52d015789ee9e"  
Accept-Ranges: bytes  
Content-Length: 2651  
Content-Type: text/html; charset=UTF-8  
\r\n  
data data data data data ...

\* Check out the online interactive exercises for more examples: [http://gaia.cs.umass.edu/kurose\\_ross/interactive/](http://gaia.cs.umass.edu/kurose_ross/interactive/)



# Các mã trạng thái phản hồi HTTP

- Mã trạng thái xuất hiện ở dòng 1 trong thông báo phản hồi từ máy chủ đến máy khách.

Một số mã trạng thái:

- 200 OK
  - Yêu cầu thành công, đối tượng được yêu cầu kèm sau trong thông điệp này
- 301 Moved Permanently
  - Đối tượng được yêu cầu đã di chuyển, vị trí mới được xác định sau trong thông điệp này (trong trường Location: )
- 400 Bad Request
  - Máy chủ không hiểu thông điệp yêu cầu
- 404 Not Found
  - Tài liệu được yêu cầu không tìm thấy trên máy chủ này
- 505 HTTP Version Not Supported



# Thử nghiệm HTTP (phía máy khách)

## 1. Kết nối đến máy chủ Web yêu thích của bạn:

```
% nc -c -v gaia.cs.umass.edu 80  
hoặc  
telnet gaia.cs.umass.edu 80
```

- Mở kết nối TCP đến cổng 80 (cổng máy chủ HTTP mặc định) tại trang web yêu thích
- Bất cứ thứ gì được nhập sẽ được gửi đến cổng 80 tại tên miền đã gõ

## 2. Nhập yêu cầu GET HTTP:

```
GET /kurose_ross/interactive/index.php HTTP/1.1  
Host: gaia.cs.umass.edu
```

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

## 3. Xem thông điệp phản hồi được gửi bởi máy chủ HTTP!

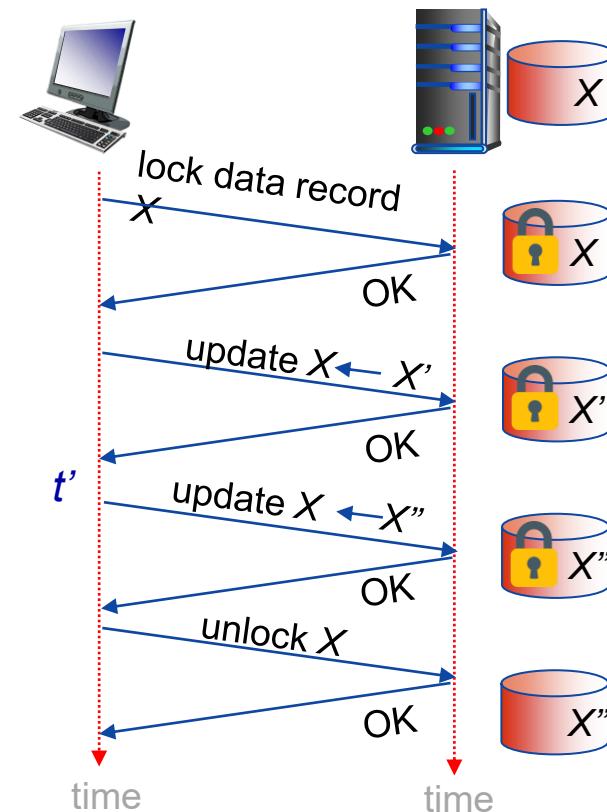
(hoặc sử dụng Wireshark để xem các thông điệp đã gửi/nhận)



# Duy trì trạng thái người dùng/máy chủ: cookie

- Nhắc lại: HTTP không lưu trữ trạng thái - stateless
- Không có khái niệm trao đổi nhiều bước của thông điệp HTTP để hoàn thành một "transaction - giao dịch" Web
  - Không cần máy khách/máy chủ theo dõi "trạng thái" của quá trình trao đổi nhiều bước
  - Tất cả các yêu cầu HTTP độc lập với nhau
  - Máy khách/máy chủ không cần phải khôi phục những giao dịch chưa hoàn thành

Giao thức lưu trạng thái: máy khách thực hiện hai thay đổi đối với X hoặc không có thay đổi nào cả



Câu hỏi: điều gì xảy ra nếu kết nối mạng hoặc máy khách gặp sự cố tại thời điểm t'?



# Duy trì trạng thái người dùng/máy chủ: cookie

Các trang web và trình duyệt của người dùng sử dụng **cookie** để duy trì một số trạng thái giữa các phiên làm việc

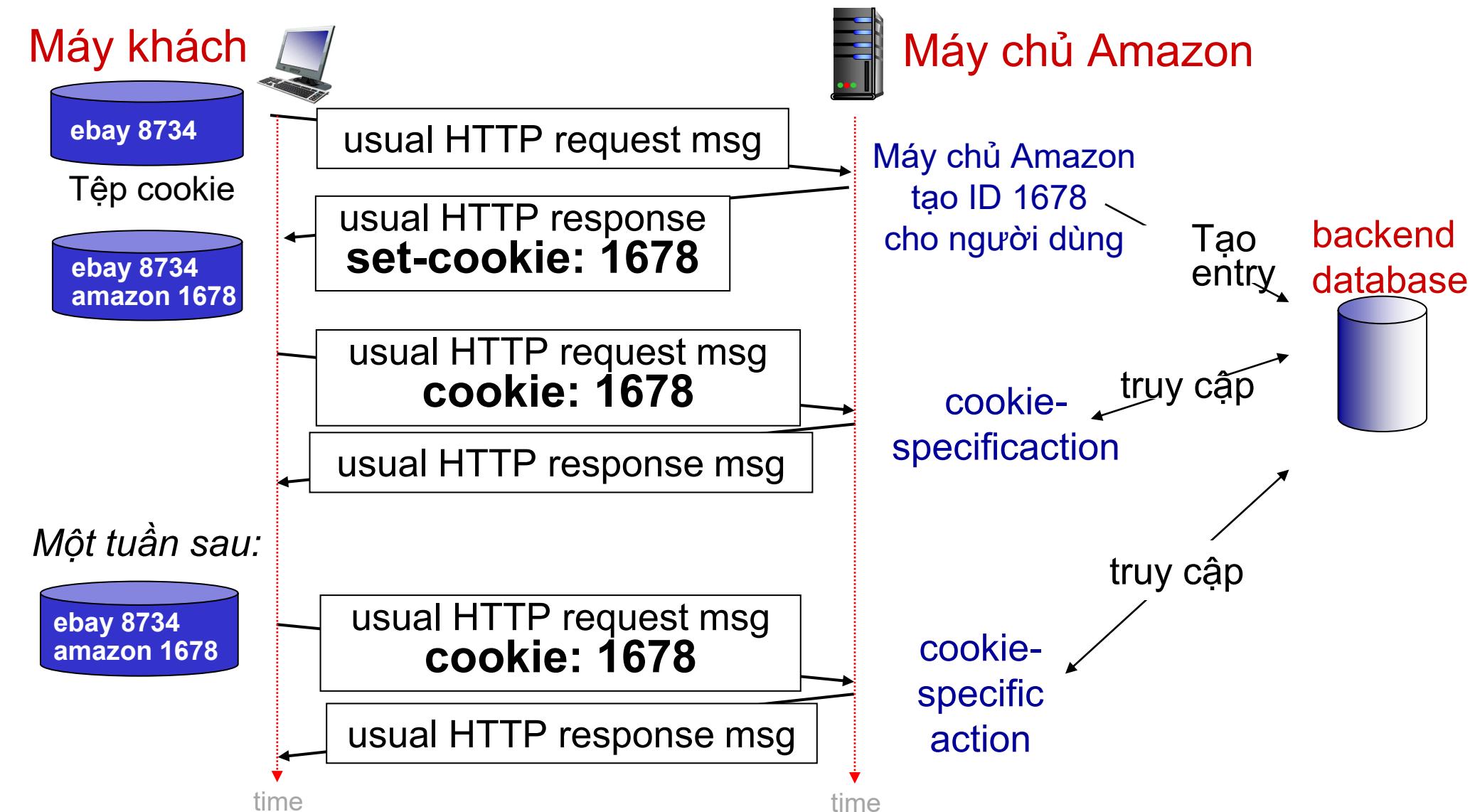
## ○ Bốn thành phần:

1. Dòng header cookie của thông điệp phản hồi HTTP
2. Dòng header cookie trong thông điệp yêu cầu HTTP tiếp theo
3. Tệp cookie được lưu giữ trên máy của người dùng, được quản lý bởi trình duyệt của người dùng
4. Cơ sở dữ liệu tại Web site

## ○ Ví dụ:

- Susan sử dụng trình duyệt trên máy tính xách tay, lần đầu tiên truy cập trang web thương mại điện tử cụ thể
- Khi yêu cầu HTTP ban đầu đến trang web, trang web sẽ tạo:
  - ID duy nhất (còn gọi là "cookie")
  - Một bản ghi trong cơ sở dữ liệu cho ID đó
- Các yêu cầu HTTP tiếp theo từ Susan đến trang web này sẽ chứa giá trị ID cookie, cho phép trang web "xác định" Susan

# Duy trì trạng thái người dùng/máy chủ: cookie





# HTTP cookies

- Cookie có thể được sử dụng cho:

- Authorization (Cấp phép)
- Giả hàng
- Khuyến nghị
- Trạng thái phiên người dùng (Web e-mail)

- Thách thức: Làm thế nào để giữ trạng thái?

- Các thời điểm kết thúc giao thức: duy trì trạng thái tại người gửi/nhận thông qua nhiều giao dịch
- Trong các thông điệp: cookie trong các thông điệp HTTP mang trạng thái

Ngoài lề

*Cookie và quyền riêng tư:*

- Cookie cho phép các trang web biết nhiều về bạn trên trang web của họ.
- Cookie liên tục của bên thứ ba (cookie theo dõi) cho phép theo dõi danh tính chung (giá trị cookie) trên nhiều trang web



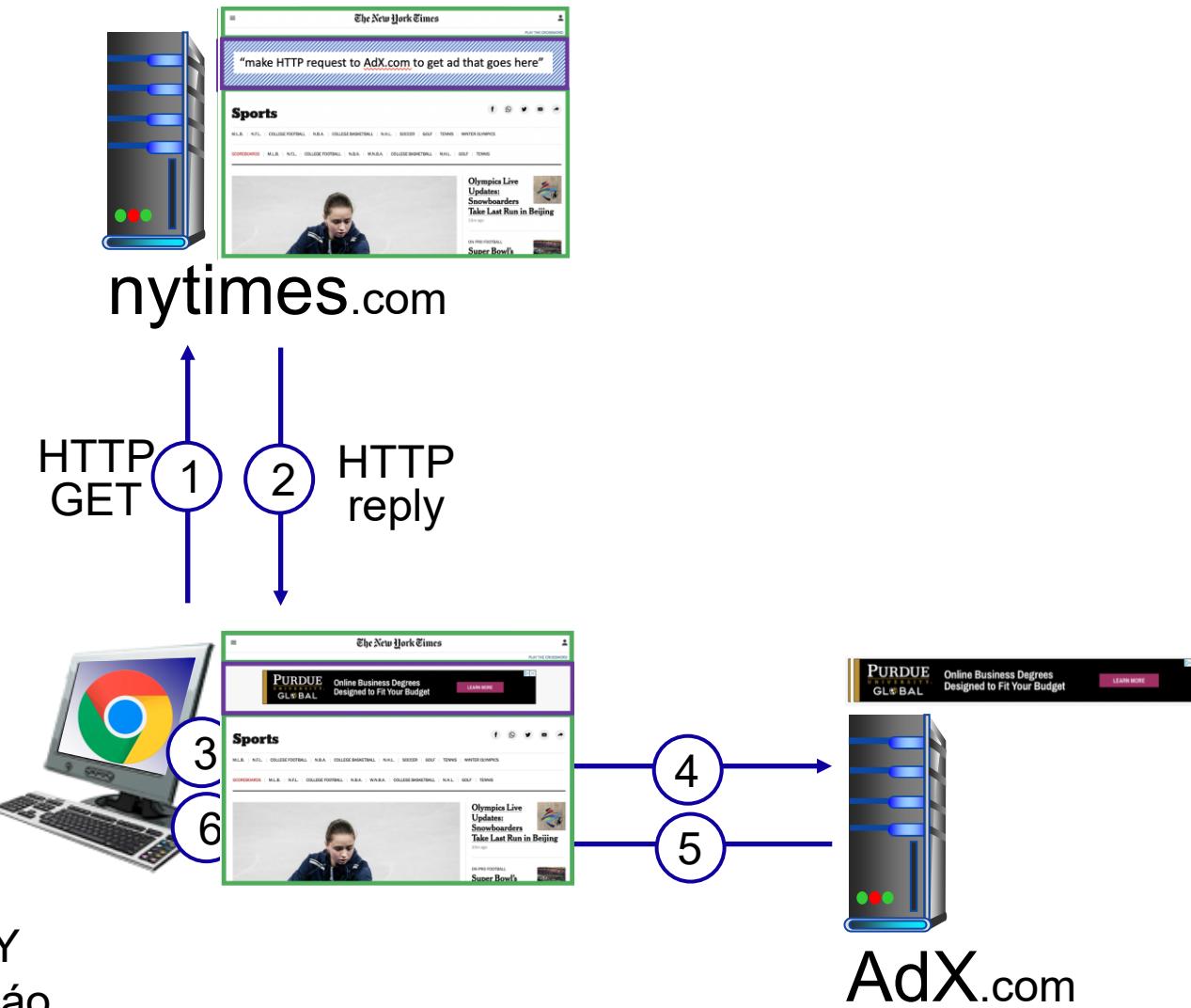
# Ví dụ: hiển thị trang web NY Times

1 GET tệp html cơ sở  
từ nytimes.com

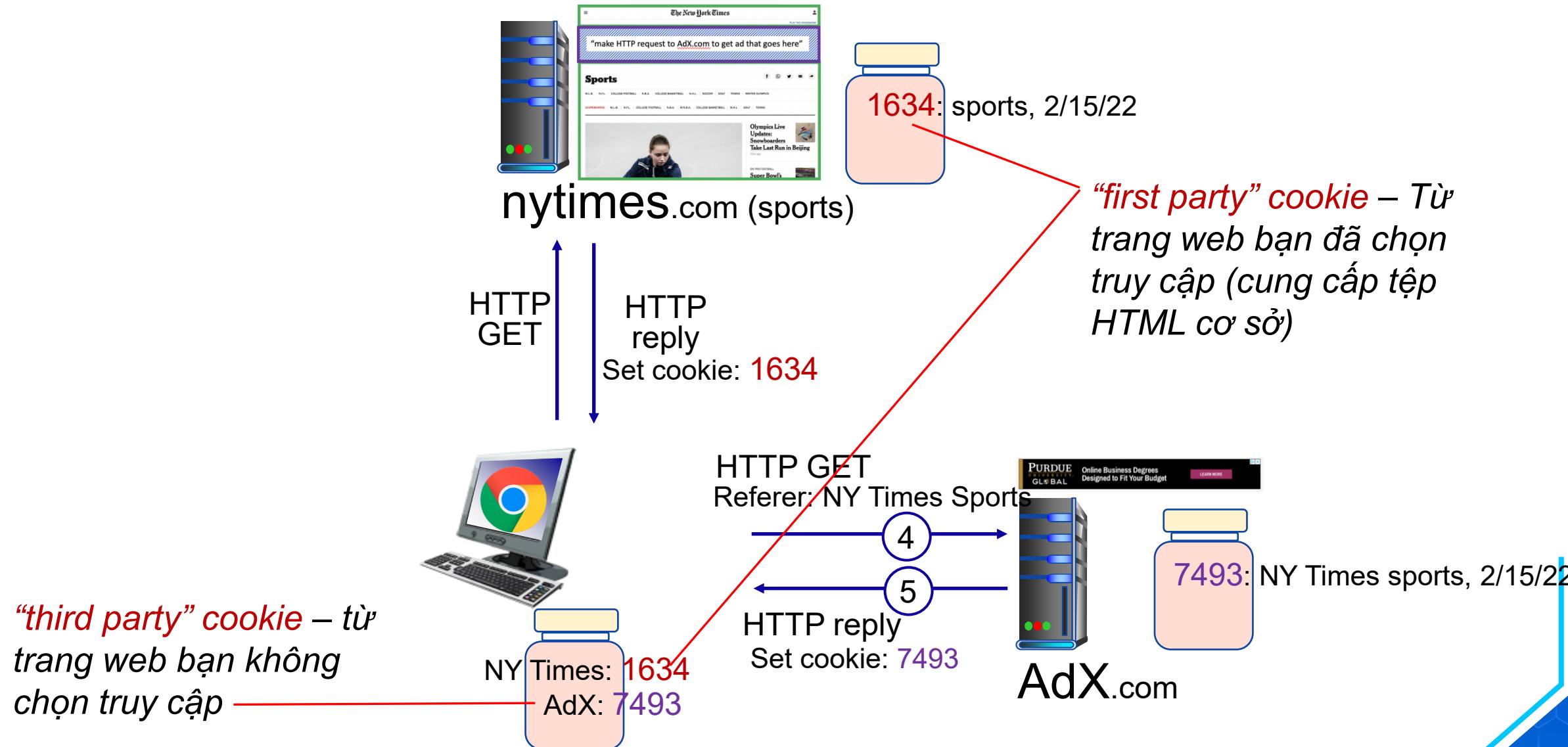
4 Tìm nạp quảng  
cáo từ AdX.com

7 Hiển thị trang

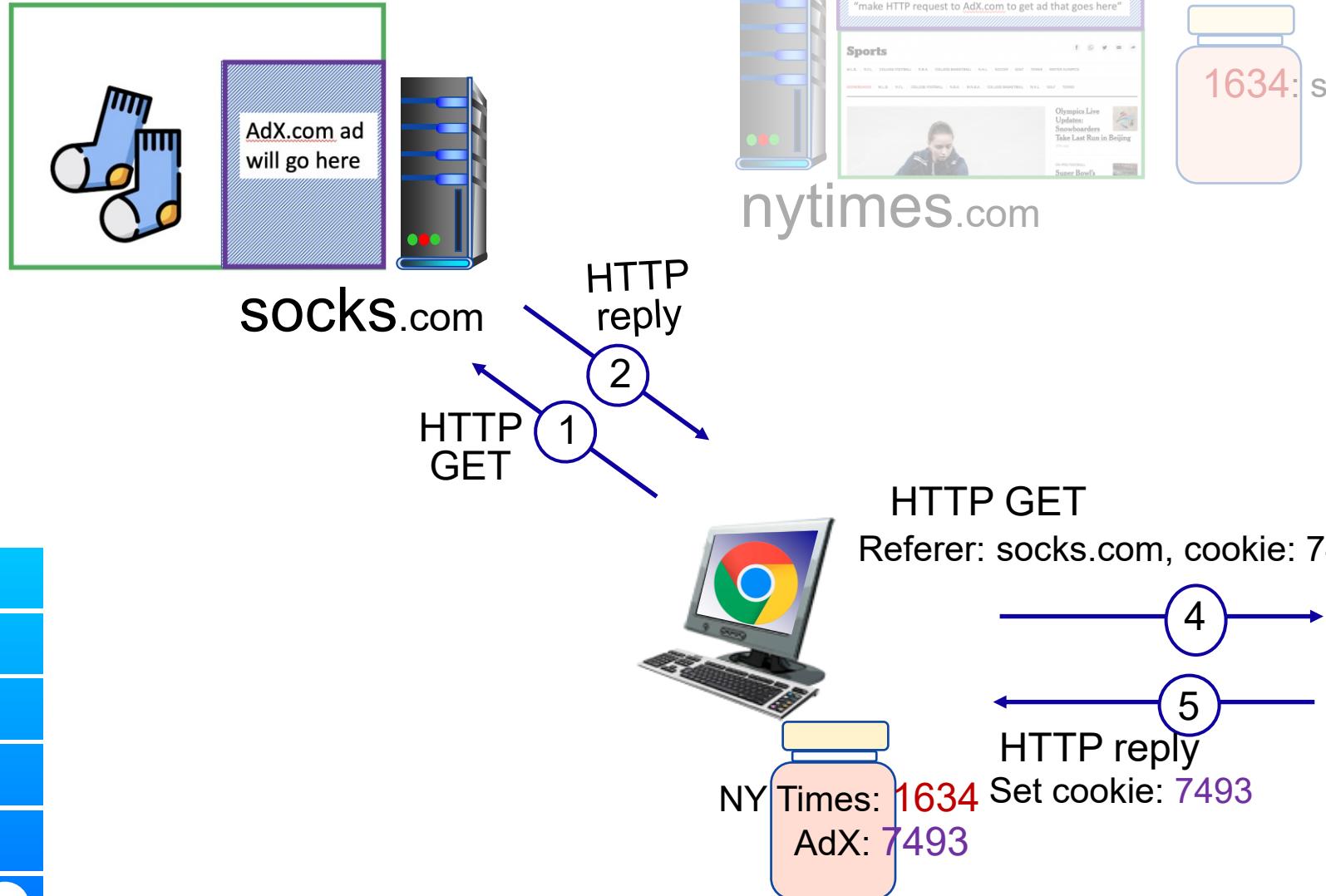
Hiển thị trang NY  
times với quảng cáo  
được nhúng



# Cookie: theo dõi hành vi duyệt web của người dùng

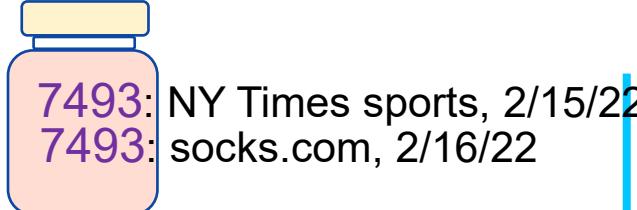


# Cookie: theo dõi hành vi duyệt web của người dùng



AdX:

- *theo dõi quá trình duyệt web* của tôi trên các trang web có quảng cáo AdX
- Có thể trả về quảng cáo được nhắm mục tiêu dựa trên lịch sử duyệt web

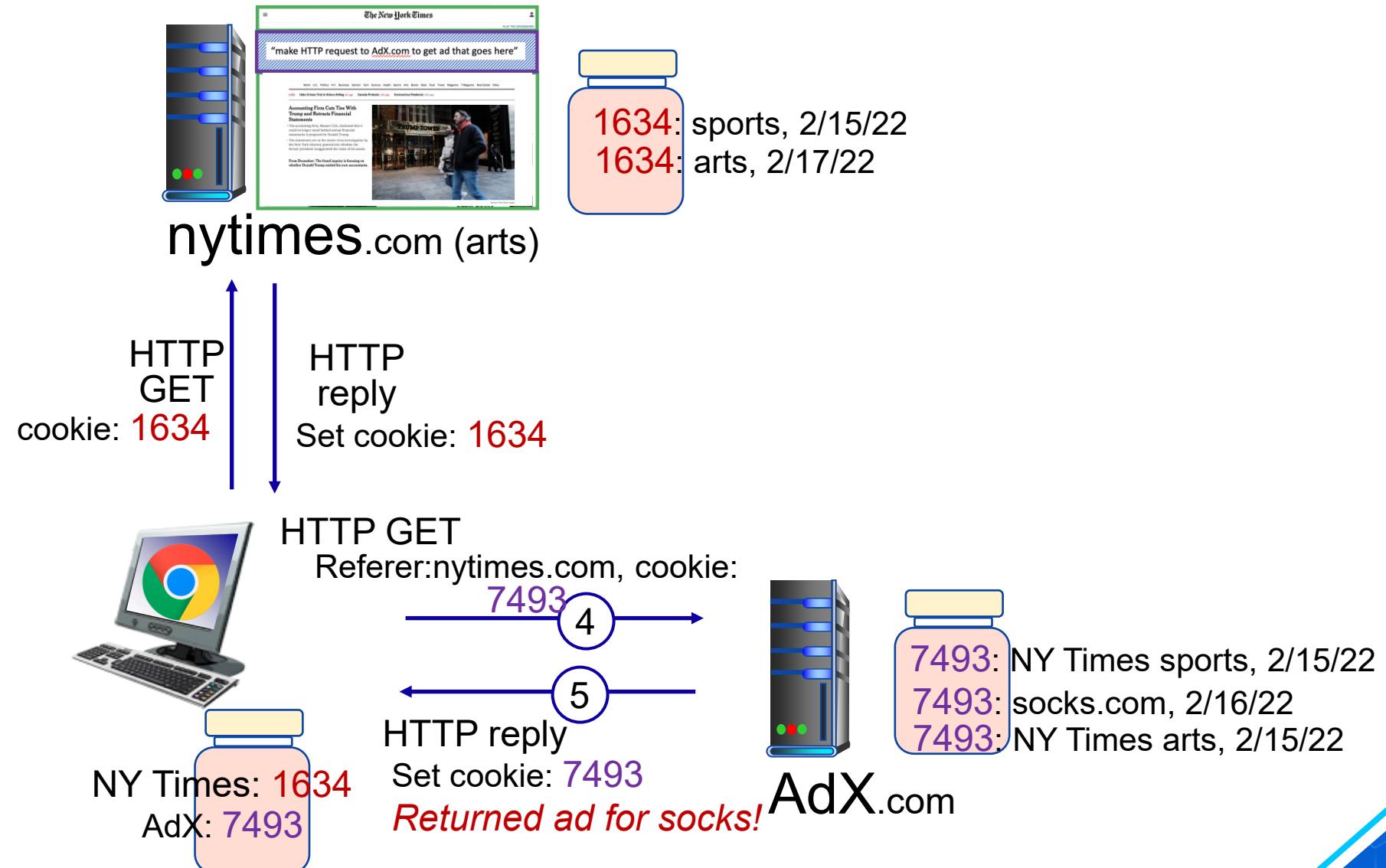


AdX.com



# Cookie: theo dõi hành vi duyệt web của người dùng (tt)

Một ngày sau đó ...





# Cookie: theo dõi hành vi duyệt web của người dùng

- Cookie có thể được sử dụng để:
  - Theo dõi hành vi của người dùng trên một trang web nhất định (cookie của bên thứ nhất)
  - Theo dõi hành vi của người dùng trên nhiều trang web (cookie của bên thứ ba) mà người dùng không truy cập trang web theo dõi (!)
- Theo dõi có thể vô hình đối với người dùng:
  - Thay vì hiển thị quảng cáo kích hoạt HTTP GET để theo dõi, có thể là một liên kết vô hình
- Theo dõi bên thứ ba (Third party tracking) qua cookie:
  - Bị tắt theo mặc định trong trình duyệt Firefox, Safari
  - Sẽ bị vô hiệu hóa trong trình duyệt Chrome vào năm 2023

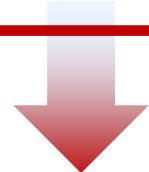
# GDPR (Quy định chung về bảo vệ dữ liệu của EU) và cookie



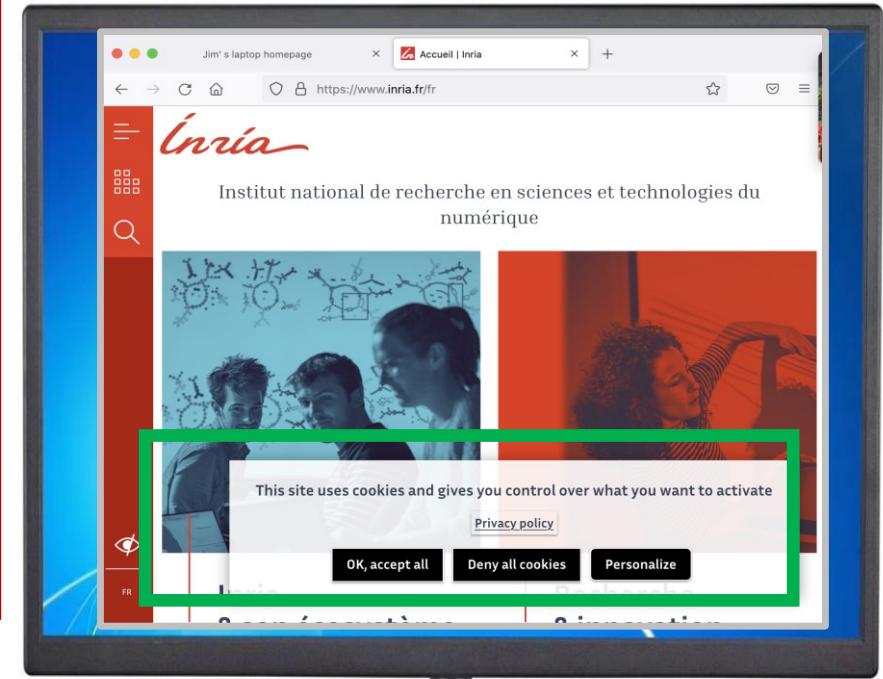
“Natural persons may be associated with online identifiers [...] such as internet protocol addresses, cookie identifiers or other identifiers [...].

This may leave traces which, in particular when combined with unique identifiers and other information received by the servers, may be used to create profiles of the natural persons and identify them.”

GDPR, recital 30 (May 2018)



Khi cookie có thể xác định một cá nhân, cookie được coi là dữ liệu cá nhân, tuân theo các quy định về dữ liệu cá nhân GDPR



*Người dùng có quyền kiểm soát rõ ràng về việc cookie có được phép hay không*

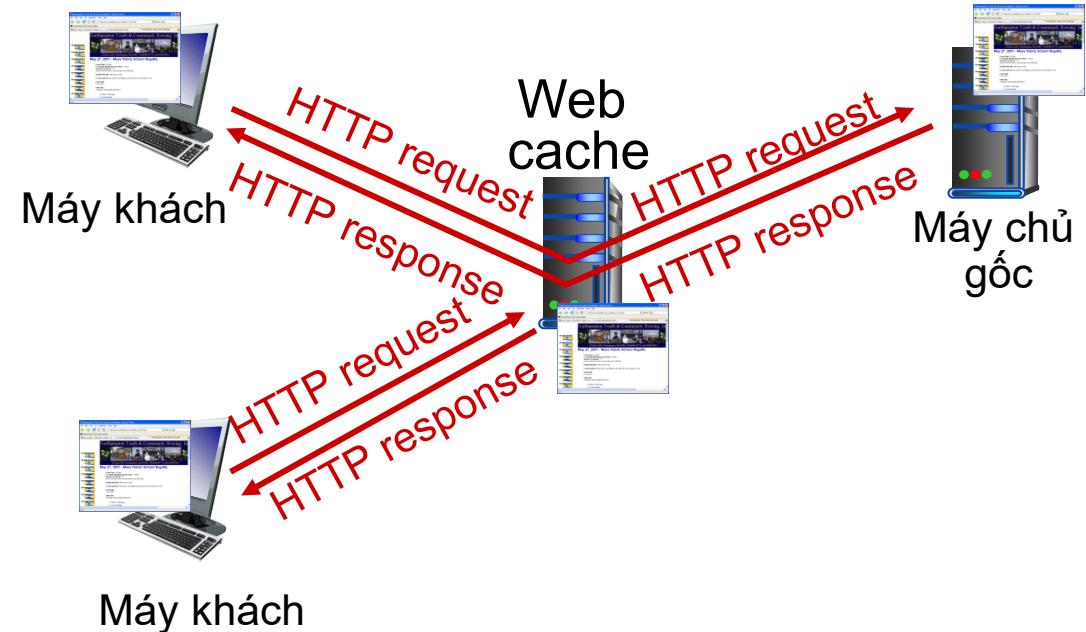


# Web caches

- **Mục tiêu:** đáp ứng các yêu cầu của khách hàng mà không liên quan đến máy chủ gốc
- User thiết lập trình duyệt: truy cập Web thông qua **Web cache**
- Trình duyệt gửi tất cả các yêu cầu

HTTP đến bộ nhớ cache

- **Nếu** đối tượng trong cache:  
cache trả về đối tượng cho máy khách
- **Ngược lại** cache yêu cầu  
đối tượng từ máy chủ gốc, lưu vào cache  
đối tượng đã nhận, sau đó trả  
về đối tượng cho máy khách





# Web cache (còn gọi là máy chủ proxy)

- Web cache hoạt động như cả máy khách và máy chủ
  - Máy chủ cho máy khách yêu cầu ban đầu
  - Máy khách đến máy chủ gốc
- Server cho Cache biết về bộ nhớ đệm cho phép của đối tượng trong tiêu đề phản hồi:
- Tại sao sử dụng Web caching?
  - Giảm thời gian đáp ứng cho yêu cầu của client
    - Bộ nhớ cache gần với máy khách hơn
  - Giảm lưu lượng trên liên kết truy cập ra Internet của tổ chức
  - Internet dày đặc với bộ nhớ cache
    - Cho phép các nhà cung cấp nội dung “kém” vẫn phân phối nội dung một cách hiệu quả

Cache-Control: max-age=<seconds>

Cache-Control: no-cache



# Ví dụ Caching:

- Ngữ cảnh:

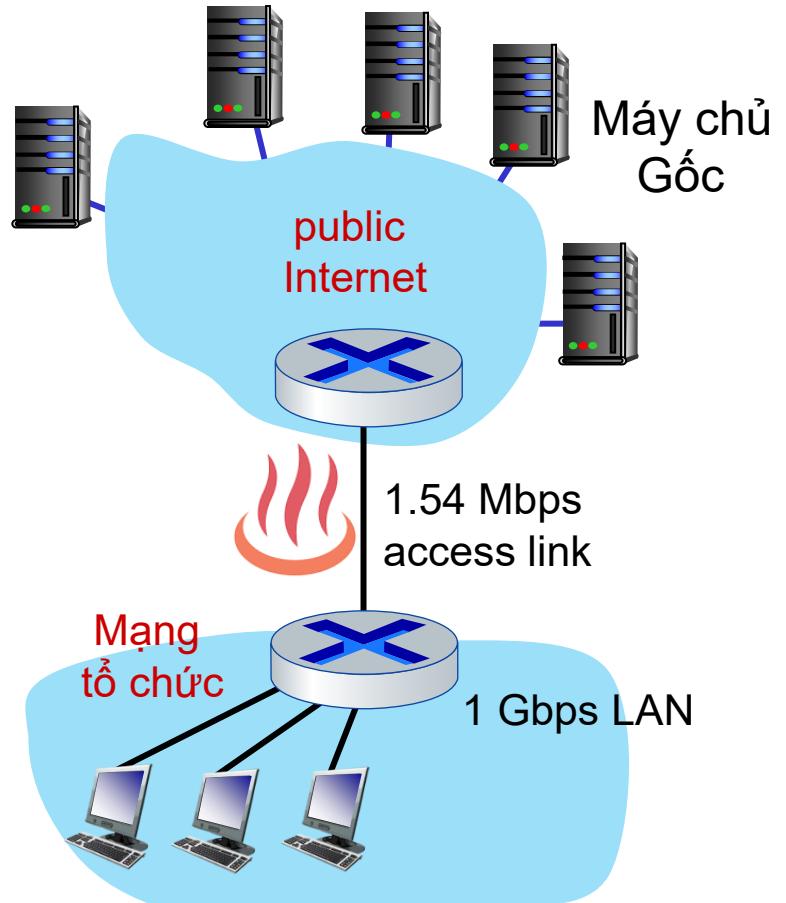
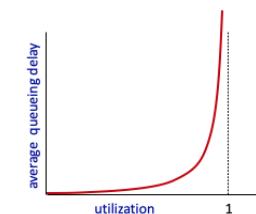
- Tốc độ liên kết truy cập: 1,54 Mbps
- RTT từ bộ định tuyến tổ chức đến máy chủ: 2 giây
- Kích thước đối tượng web: 100K bit
- Tỷ lệ yêu cầu trung bình từ trình duyệt đến máy chủ gốc: 15/giây
- Tốc độ dữ liệu trung bình cho trình duyệt: 1,50 Mbps

- Hiệu suất:

- Độ khả dụng của liên kết truy cập = .97
- Độ khả dụng của LAN = .0015

*Vấn đề: Sự chậm trễ  
xếp hàng lớn khi sử  
dụng cao!*

- Độ trễ đầu cuối = Độ trễ Internet +  
độ trễ liên kết truy cập + độ trễ mạng LAN  
= 2 giây + phút + USECS





# Tùy chọn 1: mua liên kết truy cập nhanh hơn

- Ngữ cảnh:

- Tốc độ liên kết truy cập: 1,54 Mbps
- RTT từ bộ định tuyến tổ chức đến máy chủ: 2 giây
- Kích thước đối tượng web: 100K bit
- Tỷ lệ yêu cầu trung bình từ trình duyệt đến máy chủ gốc: 15/giây
- Tốc độ dữ liệu trung bình cho trình duyệt: 1,50 Mbps

- Hiệu suất:

- Độ khả dụng của liên kết truy cập = .97

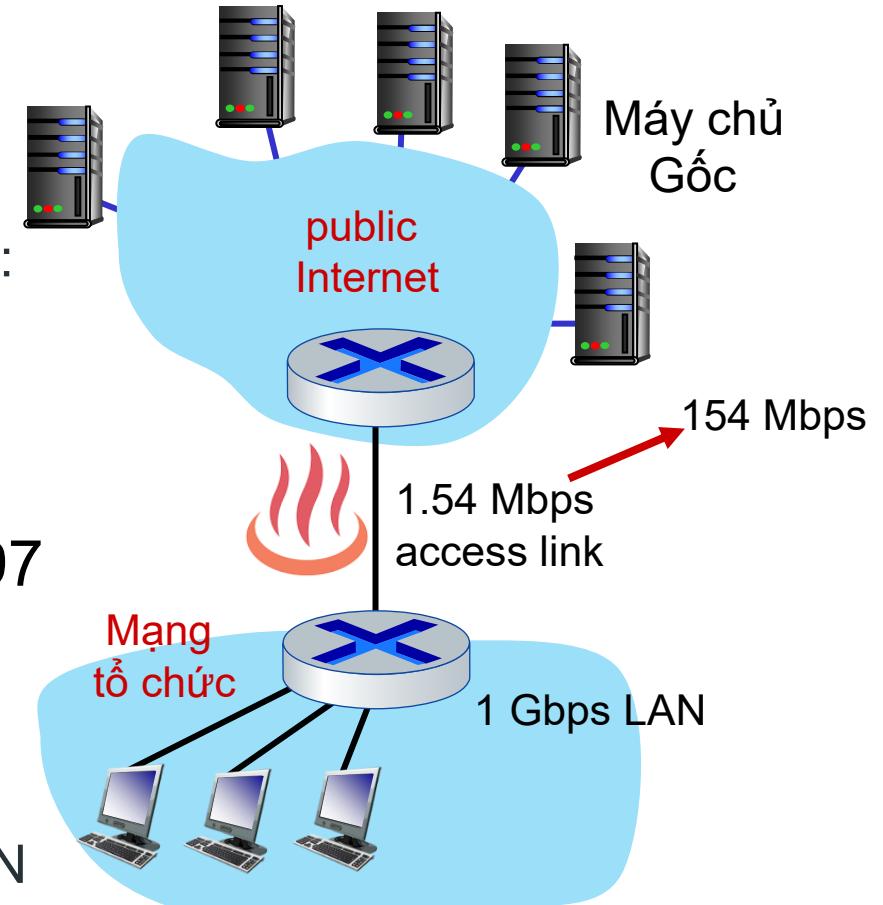
- Độ khả dụng của LAN = .0015

- Độ trễ đầu cuối = Độ trễ Internet +

độ trễ liên kết truy cập + độ trễ mạng LAN

= 2 giây + phút + USECS

154 Mbps



**Chi phí:** liên kết truy cập nhanh hơn (đắt!)

msecs



## Tùy chọn 2: cài đặt Web cache

- Ngữ cảnh:

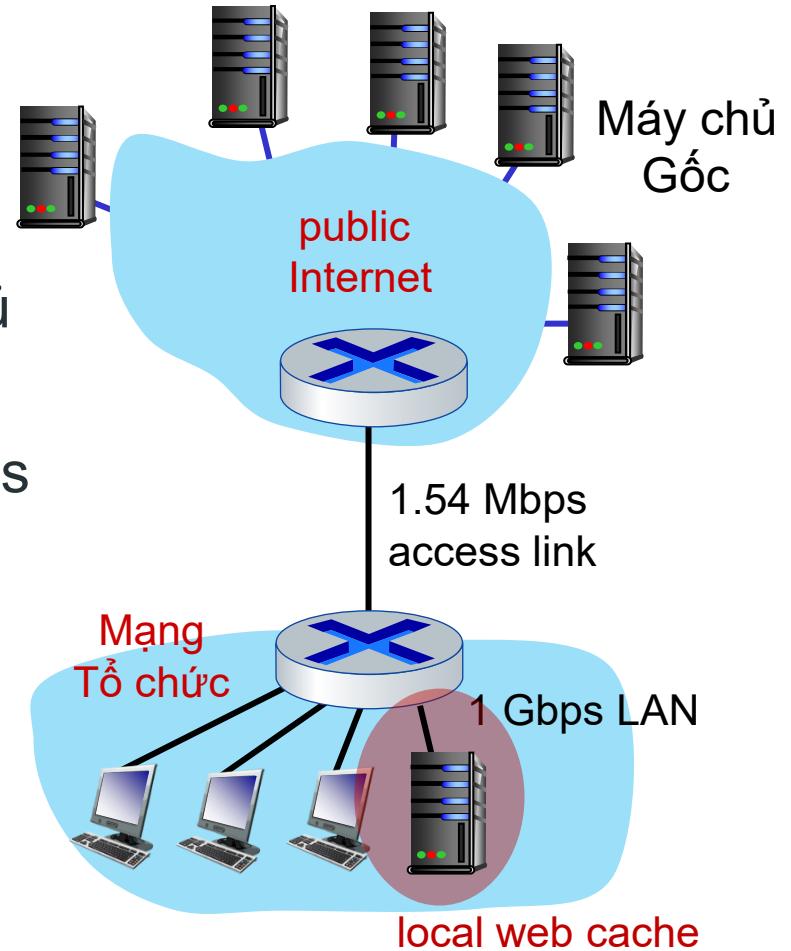
- Tốc độ liên kết truy cập: 1,54 Mbps
- RTT từ bộ định tuyến tổ chức đến máy chủ: 2 giây
- Kích thước đối tượng web: 100K bit
- Tỷ lệ yêu cầu trung bình từ trình duyệt đến máy chủ gốc: 15/giây
- Tốc độ dữ liệu trung bình cho trình duyệt: 1,50 Mbps

- Chi phí: web cache (rẻ)

- Hiệu suất:

- Độ khả dụng của liên kết truy cập = ?
- Độ khả dụng của LAN = ?
- Độ trễ = ?

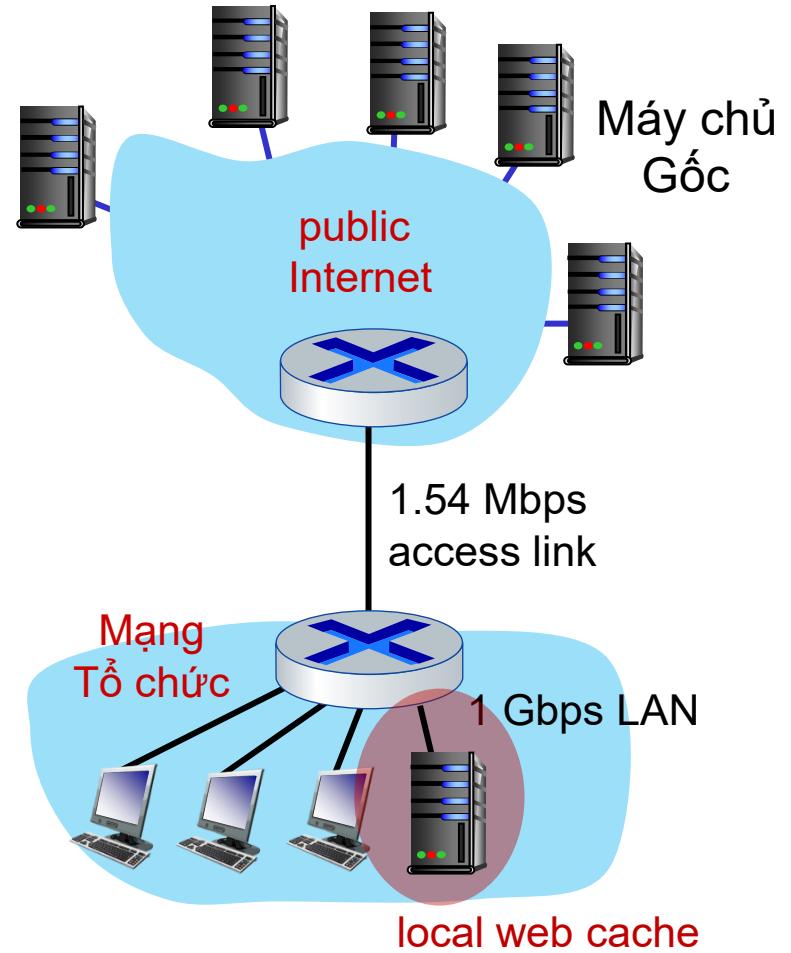
*Làm cách nào để tính  
độ khả dụng đường liên kết, độ trễ?*





# Tính toán độ khả dụng, độ trễ với cache

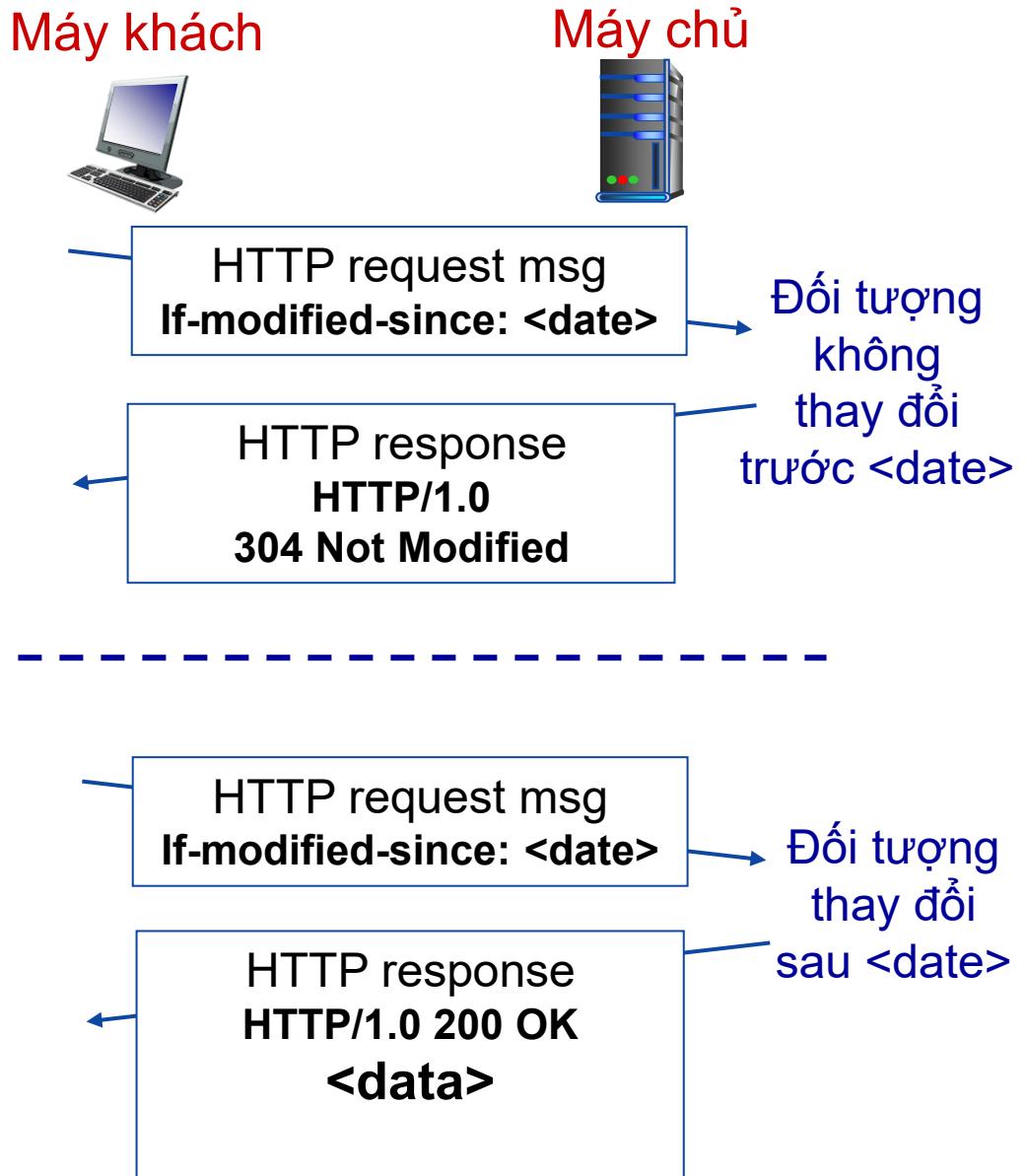
- Giả sử tỷ lệ truy cập bộ nhớ cache là 0,4:
    - 40% yêu cầu được phục vụ bởi cache, với độ trễ thấp (milisec), 60% yêu cầu được máy chủ gốc đáp ứng
    - Tốc độ truyền dữ liệu đến trình duyệt trên liên kết truy cập =  $0,6 * 1,50 \text{ Mbps} = 0,9 \text{ Mbps}$
    - Độ khả dụng liên kết truy cập =  $0,9 / 1,54 = 0,58$  có nghĩa là độ trễ hàng đợi thấp (MCSEC) tại liên kết truy cập
  - Tổng độ trễ cuối cùng:
    - =  $0,6 * (\text{độ trễ từ máy chủ gốc})$
    - +  $0,4 * (\text{độ trễ khi đáp ứng bởi cache})$
    - =  $0,6 * (2,01) + 0,4 * (\sim \text{mili giây}) = \sim 1,2 \text{ giây}$
- Độ trễ đầu cuối trung bình thấp hơn so với liên kết 154 Mbps (và rẻ hơn!)*





# Cache của trình duyệt: GET có điều kiện

- **Mục tiêu:** không gửi đối tượng nếu trình duyệt có phiên bản cập nhật được lưu trong cache
  - Không có độ trễ truyền đối tượng (hoặc sử dụng tài nguyên mạng)
  - *Client*: xác định thời gian của bản sao được đếm trong thông điệp yêu cầu HTTP
    - **If-modified-since: <date>**
  - *Server*: Phản hồi không chứa đối tượng nếu bản sao được lưu trong bộ nhớ cache của trình duyệt được cập nhật:
    - **HTTP/1.0 304 Not Modified**



# HTTP/2 [RFC 7540, 2015]



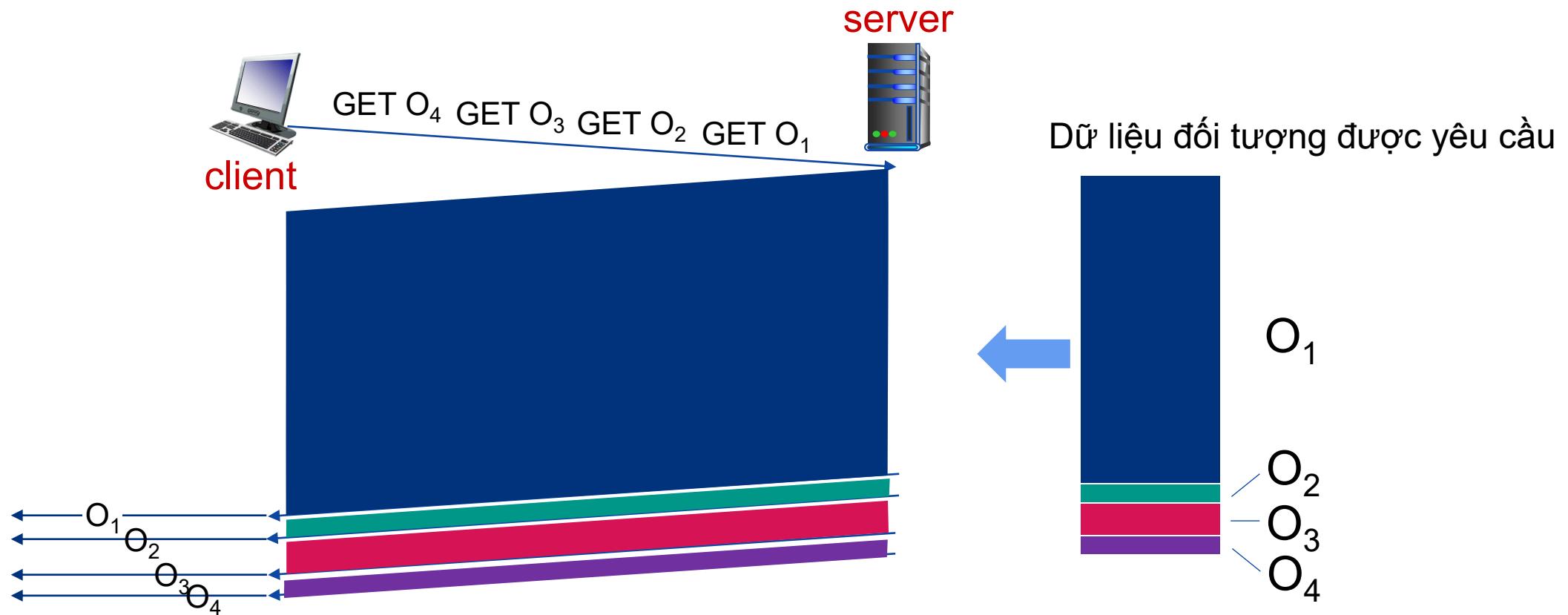
- HTTP1.1: giới thiệu nhiều GET pipelined qua một kết nối TCP
  - Máy chủ phản hồi theo thứ tự (FCFS: lập lịch ai đến trước được phục vụ trước) cho các yêu cầu GET
  - Với FCFS, đối tượng nhỏ có thể phải đợi truyền (chặn đầu dòng (HOL) phía sau (các) đối tượng lớn
  - Phục hồi mất mát (truyền lại các phân đoạn TCP bị mất) bị trì hoãn truyền đối tượng
- HTTP/2: tăng tính linh hoạt tại server trong việc gửi các đối tượng đến client
  - Phương thức, mã trạng thái, hầu hết các header không thay đổi so với HTTP 1.1
  - Thứ tự truyền của các đối tượng được yêu cầu dựa trên mức độ ưu tiên đối tượng do client chỉ định (không nhất thiết là FCFS)
  - Chia đối tượng thành các frame, lên lịch các frame để giảm thiểu HOL blocking

*Mục tiêu chính:* giảm độ trễ trong các yêu cầu HTTP đa đối tượng



# HTTP/2: giảm thiểu HOL blocking

HTTP 1.1: máy khách yêu cầu 1 đối tượng lớn (ví dụ: tệp video) và 3 đối tượng nhỏ hơn

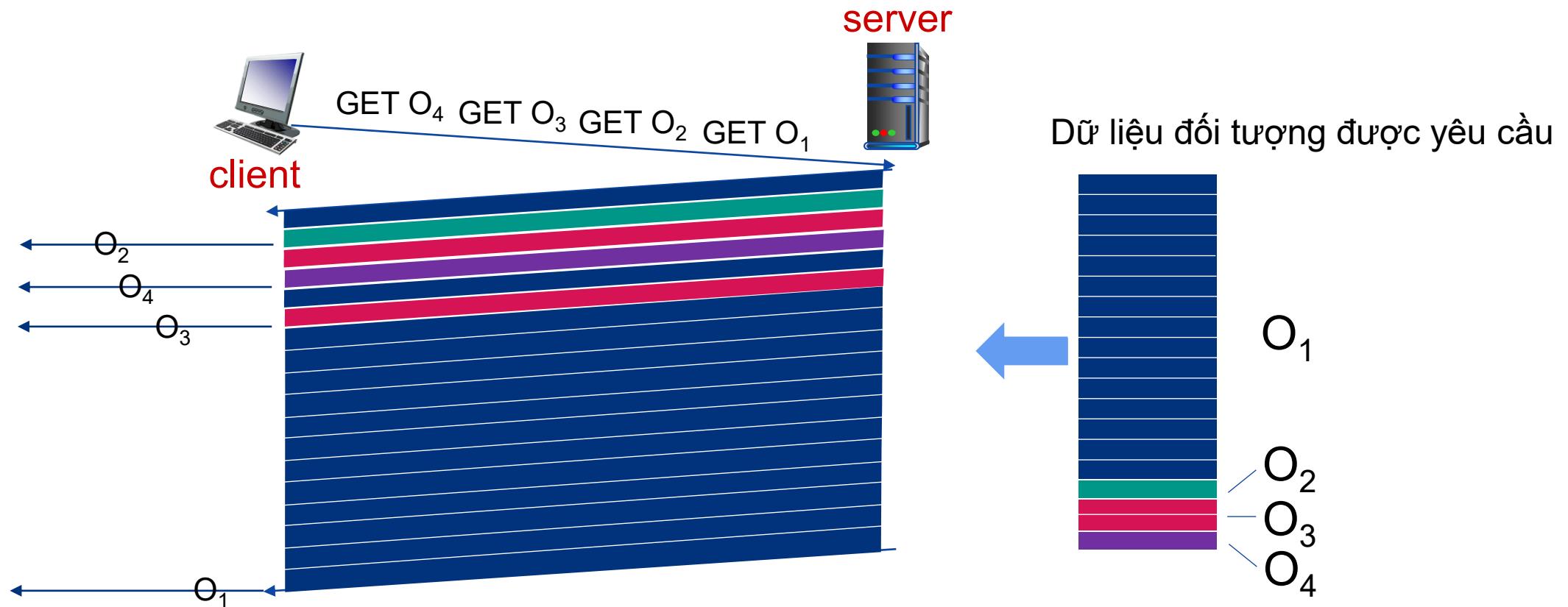


Các đối tượng được chuyển theo thứ tự yêu cầu: O<sub>2</sub>, O<sub>3</sub>, O<sub>4</sub> chờ phía sau O<sub>1</sub>



# HTTP/2: giảm thiểu HOL blocking

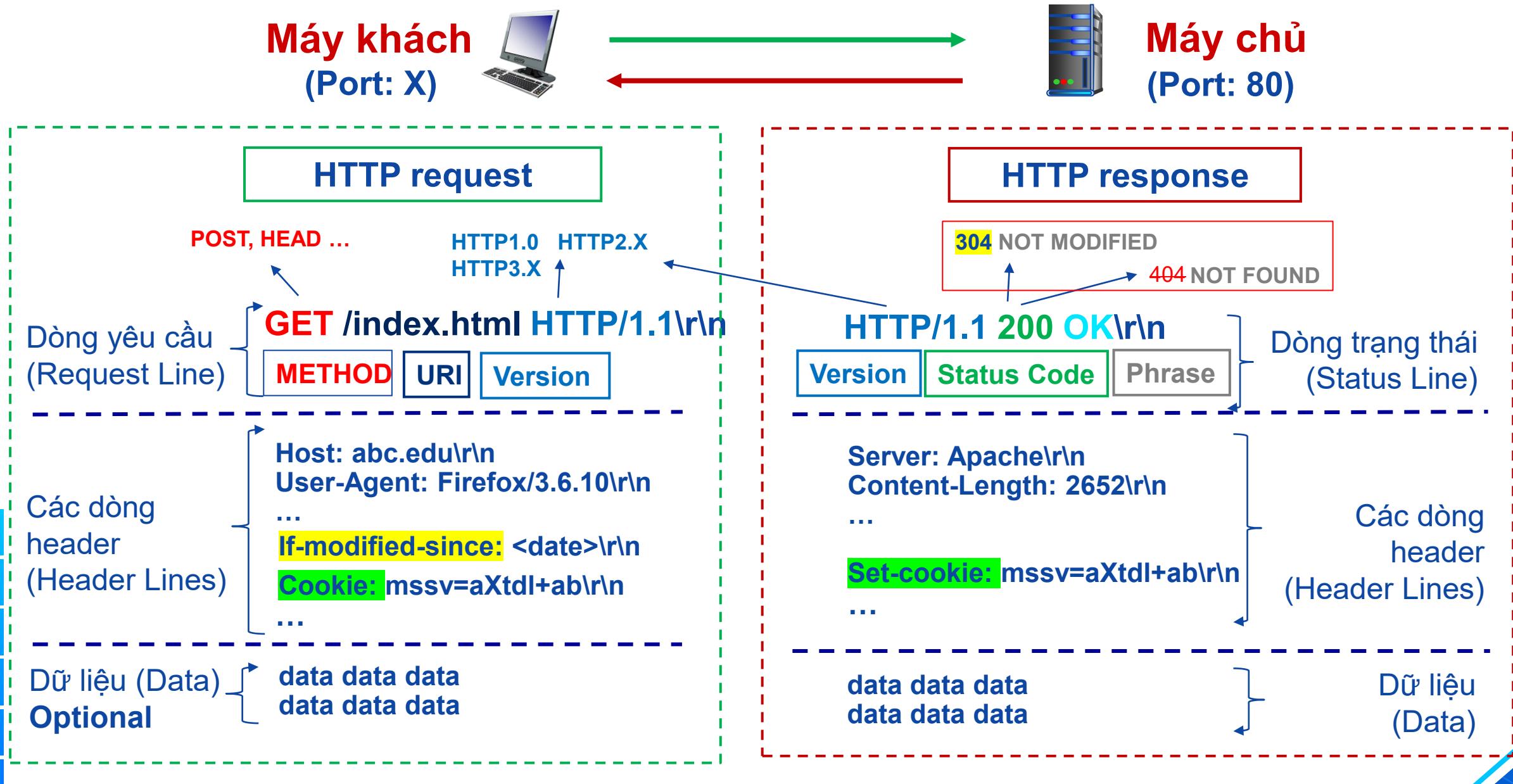
HTTP/2: các đối tượng được chia thành các frame, truyền frame xen kẽ



O<sub>2</sub>, O<sub>3</sub>, O<sub>4</sub> được phân phối nhanh, O<sub>1</sub> hơi chậm trễ



# Tổng quan HTTP





# Tầng Ứng dụng – Tổng quan

- Nguyên lý của các ứng dụng mạng
- Web và HTTP
- **E-mail, SMTP, IMAP**
- Hệ thống phân giải tên miền DNS
- Các ứng dụng P2P
- Lập trình socket với UDP và TCP





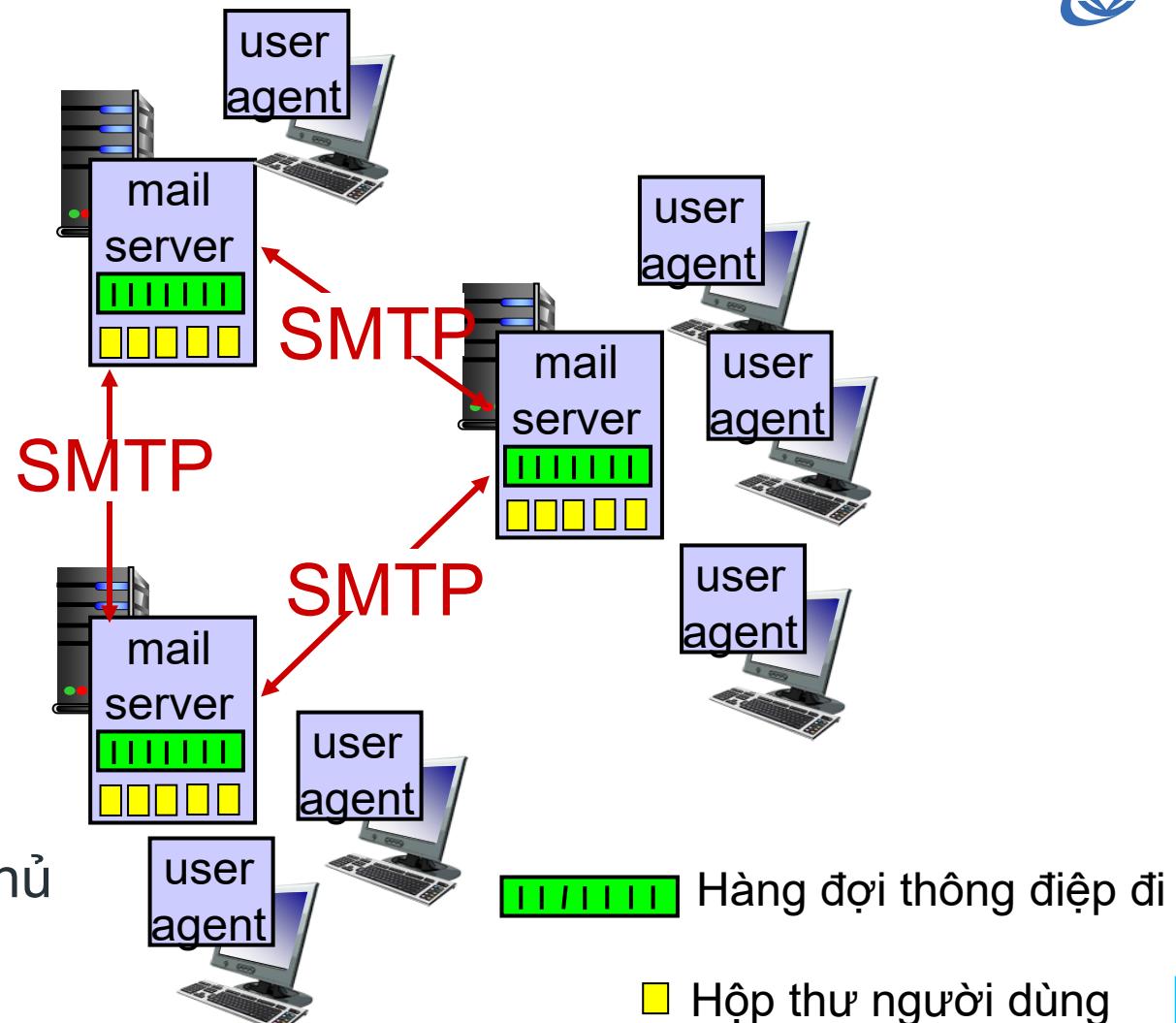
# E-mail – Thư điện tử

- Ba thành phần chính

- User agents
- Máy chủ thư điện tử (mail server)
- Simple mail transfer protocol: SMTP

- User Agent

- Còn gọi là “mail reader”
- Soạn thảo, chỉnh sửa, đọc thư điện tử
- Ví dụ: Outlook, ứng dụng mail iPhone
- Thư đi, thư đến được lưu trữ trên máy chủ

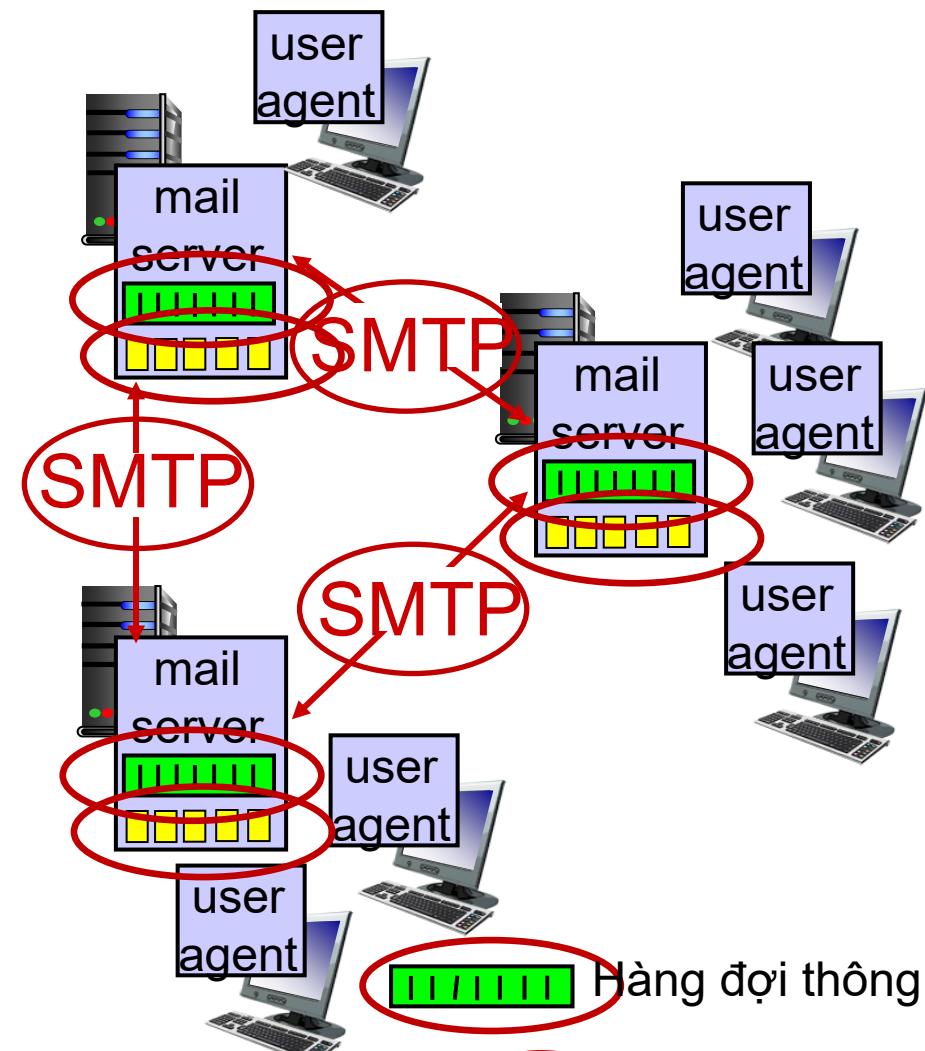


# E-mail: mail servers



## ○ **Máy chủ thư điện tử:**

- **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** được dùng để gửi các thông điệp thư điện tử giữa các mail server
    - Client là máy chủ thư điện tử gửi
    - Server là máy chủ thư điện tử nhận

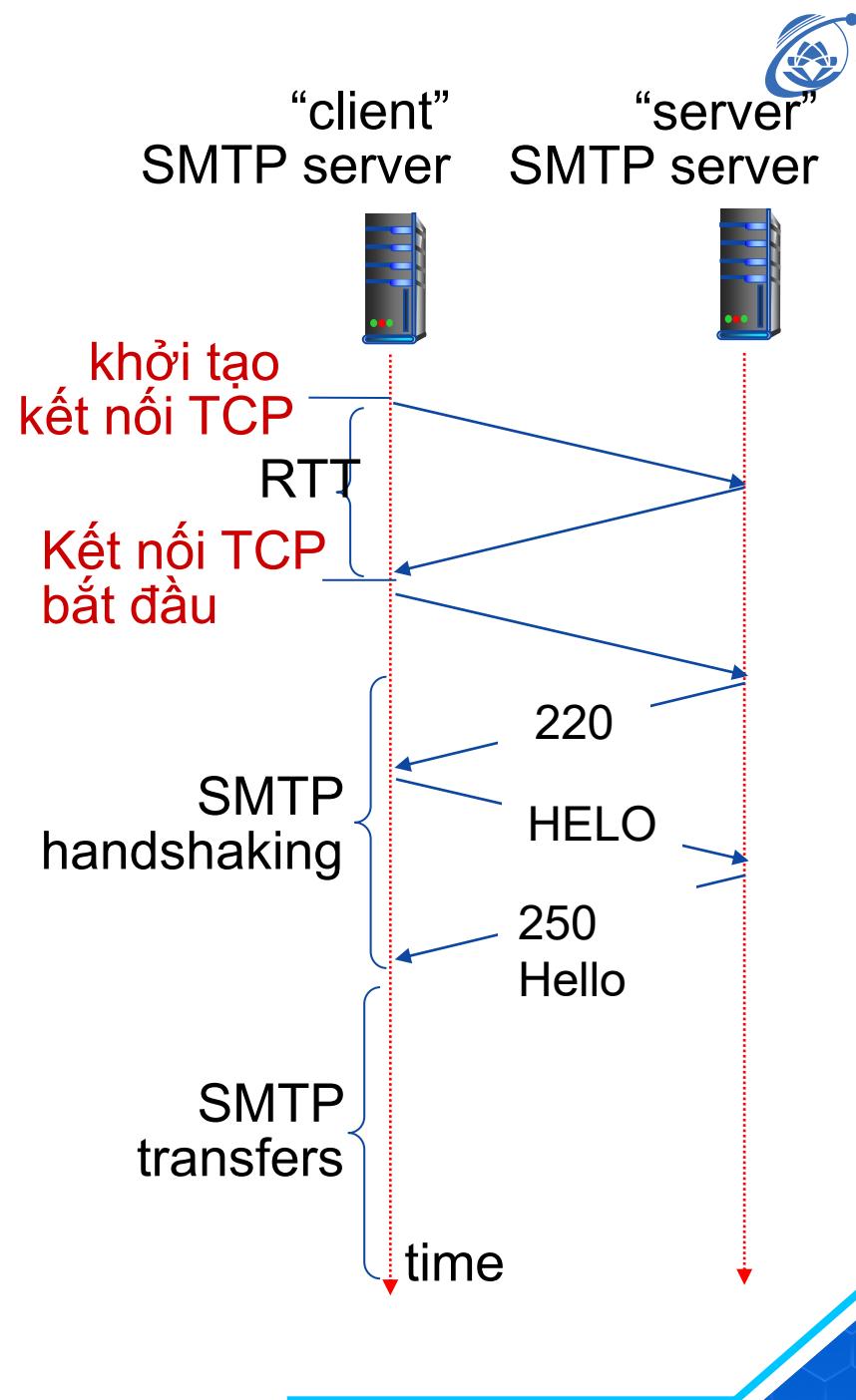


## Hàng đợi thông điệp đi

## Hộp thư người dùng

# SMTP RFC (5321)

- Sử dụng TCP để truyền thông điệp thư điện tử một cách tin cậy từ máy khách đến cổng 25 của máy chủ
  - Truyền trực tiếp: máy chủ gửi thư đến máy chủ nhận
- Ba giai đoạn truyền
  - Bắt tay (lời chào)
  - Truyền thông điệp
  - Đóng
- Tương tác lệnh/phản hồi (như HTTP)
  - Lệnh: Văn bản ASCII
  - Phản hồi: Mã trạng thái và cụm từ

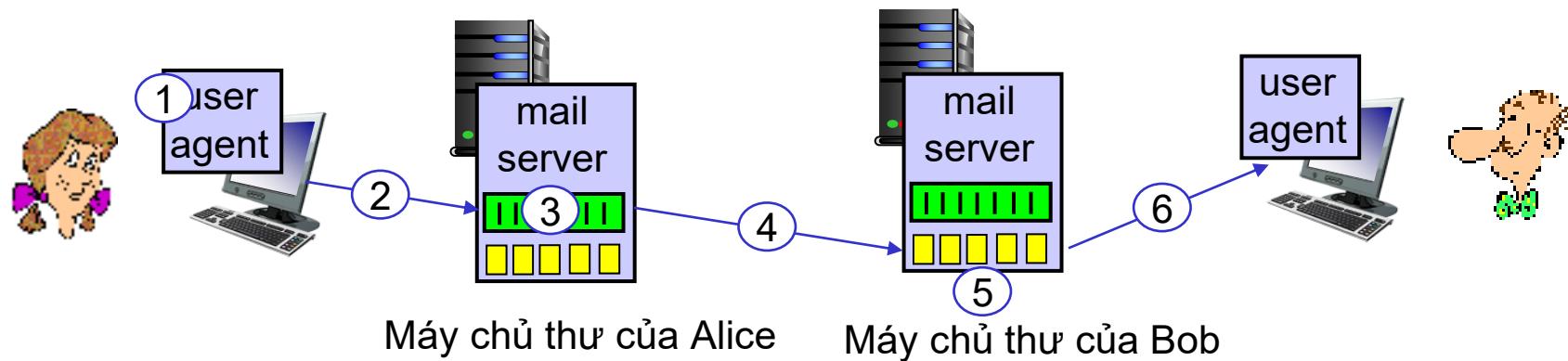




# Tình huống: Alice gửi e-mail cho Bob

- 1) Alice sử dụng UA để soạn e-mail gửi đến bob@someschool.edu
- 2) UA của Alice gửi tin nhắn đến máy chủ thư của cô ấy bằng SMTP; Thư được đặt trong hàng đợi thư
- 3) Phía máy khách của SMTP tại máy chủ thư mở kết nối TCP với máy chủ thư điện tử của Bob

- 4) Máy khách SMTP gửi tin nhắn của Alice qua kết nối TCP
- 5) Máy chủ thư của Bob đặt thư trong hộp thư của Bob
- 6) Bob dùng UA của mình để đọc tin nhắn





# Tương tác SMTP mẫu

```
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
```



# SMTP: Kết luận

## ○ So sánh với HTTP:

- HTTP: pull (kéo)
- SMTP: push (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 chứa nhiều phần

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



# Định dạng thư - email

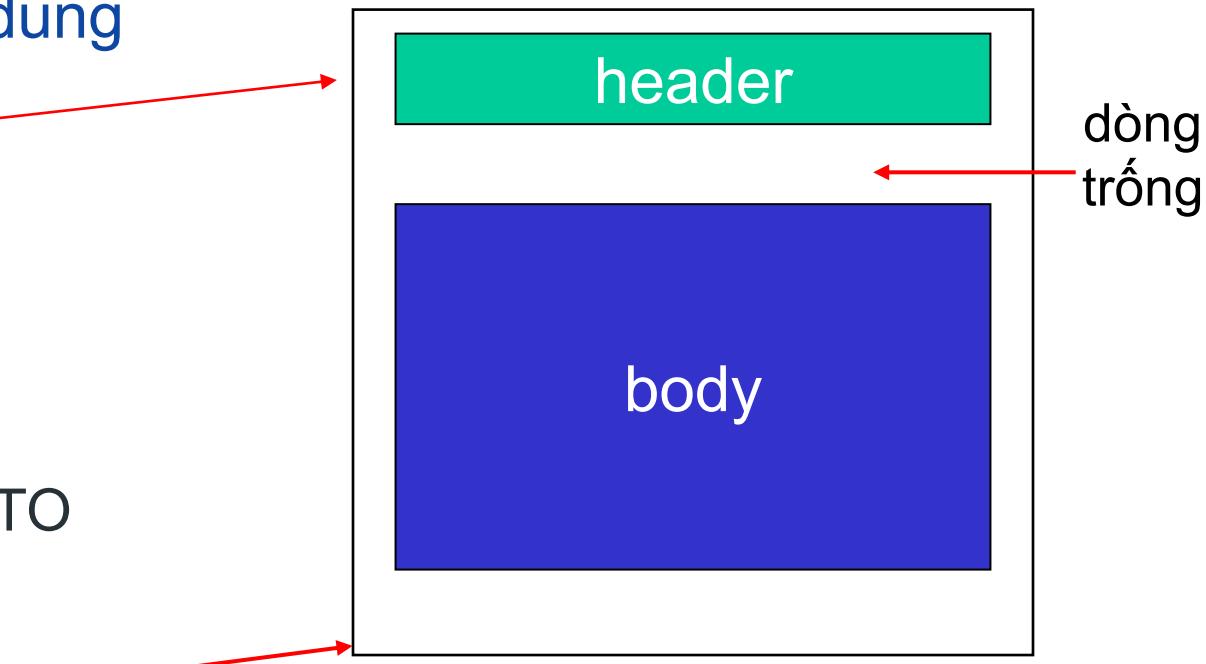
- SMTP: giao thức trao đổi thư e-mail, được định nghĩa trong RFC 5321
- RFC 2822 định nghĩa cú pháp cho thông điệp email
- Các dòng header, nằm trong phần nội dung

DATA của thư, ví dụ

- To:
- From:
- Subject:

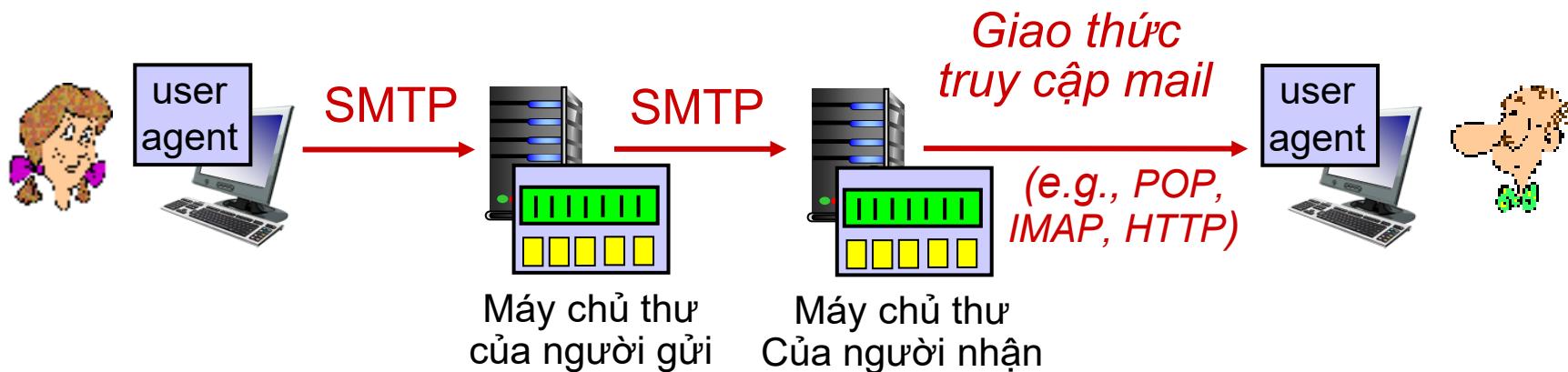
Khác với lệnh SMTP MAIL FROM:, RCPT TO

- Body: "thông điệp", chỉ các ký tự ASCII





# Truy xuất email: Các giao thức truy cập thư



- **SMTP**: gửi/lưu trữ thư e-mail đến máy chủ của người nhận
- Giao thức truy cập thư: Truy xuất từ máy chủ
  - **POP**: Post Office Protocol [RFC 1939]: xác thực, tải thư về
  - **IMAP**: Internet Mail Access Protocol [RFC 3501]: thư được lưu trữ trên máy chủ, IMAP cung cấp truy xuất, xóa, thư mục thư được lưu trữ trên máy chủ
  - **HTTP**: gmail, Hotmail, Yahoo!Mail, v.v. cung cấp giao diện web trên STMP (để gửi), IMAP (hoặc POP) để truy xuất thư e-mail

# Tầng Ứng dụng – Tổng quan



- Nguyên lý của các ứng dụng mạng
- Web và HTTP
- E-mail, SMTP, IMAP
- **Hệ thống phân giải tên miền DNS**
- Các ứng dụng P2P
- Lập trình socket với UDP và TCP



# DNS: Domain Name System - Hệ thống tên miền



- Con người: nhiều định danh:
  - SSN, Họ tên, Hộ chiếu #
- Internet hosts, routers:
  - Địa chỉ IP address (32 bit) – được dùng để định địa chỉ gói tin
  - “tên”, ví dụ uit.edu.vn – được dùng bởi con người
- Q: làm sao để ánh xạ giữa địa chỉ IP và tên, và ngược lại?
- Hệ thống tên miền - Domain Name System (DNS):
  - **Cơ sở dữ liệu phân tán** được triển khai trong hệ thống phân cấp của nhiều máy chủ DNS
  - **Giao thức Tầng Ứng dụng:** các hệ thống đầu cuối, các máy chủ DNS servers giao tiếp để **phân giải** tên (địa chỉ ⇔ tên)
    - Lưu ý: chức năng trong phần lõi Internet, được thực hiện như là giao thức Tầng ứng dụng
    - Sự phức tạp ở “biên” của mạng

# DNS: các dịch vụ, cấu trúc



## ○ Các dịch vụ DNS

- Dịch tên máy ra địa chỉ IP
- Bí danh máy
  - Lưu các tên gốc, bí danh tương ứng
- Bí danh máy chủ thư điện tử
- Cân bằng tải
  - Các bản sao cho web server: nhiều địa chỉ IP tương ứng cho 1 tên miền

## Q: Tại sao không tập trung hóa DNS?

- Một điểm chịu lỗi
- Lưu lượng
- Cơ sở dữ liệu tập trung cách xa nơi yêu cầu
- Bảo trì

## A: Không biến đổi được quy mô!

- Chỉ riêng máy chủ DNS Comcast: 600 tỷ truy vấn DNS/ngày
- Chỉ riêng máy chủ DNS Akamai: 2.2 nghìn tỷ truy vấn DNS/ngày



# Suy nghĩ về DNS

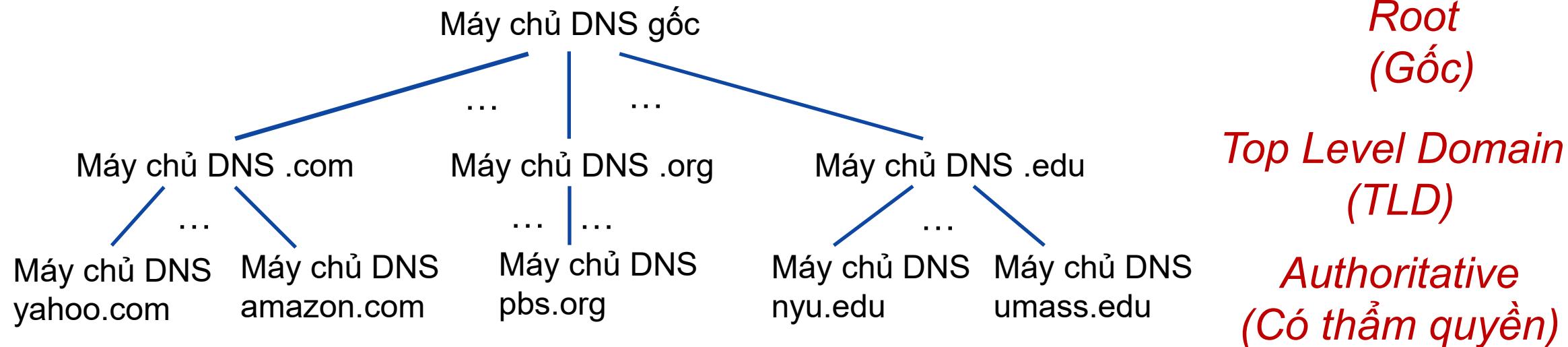
- Cơ sở dữ liệu phân tán khổng lồ:
  - ~ Hàng tỷ bản ghi, mỗi bản ghi đơn giản
- Xử lý hàng nghìn tỷ truy vấn/ngày:
  - Đọc nhiều hơn ghi
  - Hiệu suất quan trọng: hầu hết mọi giao dịch Internet đều tương tác với DNS - số mili giây!
- Về mặt tổ chức, phân cấp vật lý:
  - Hàng triệu tổ chức khác nhau chịu trách nhiệm về thông tin của họ



“Vấn đề”: độ tin cậy, bảo mật



# DNS: cơ sở dữ liệu phân tán, phân cấp

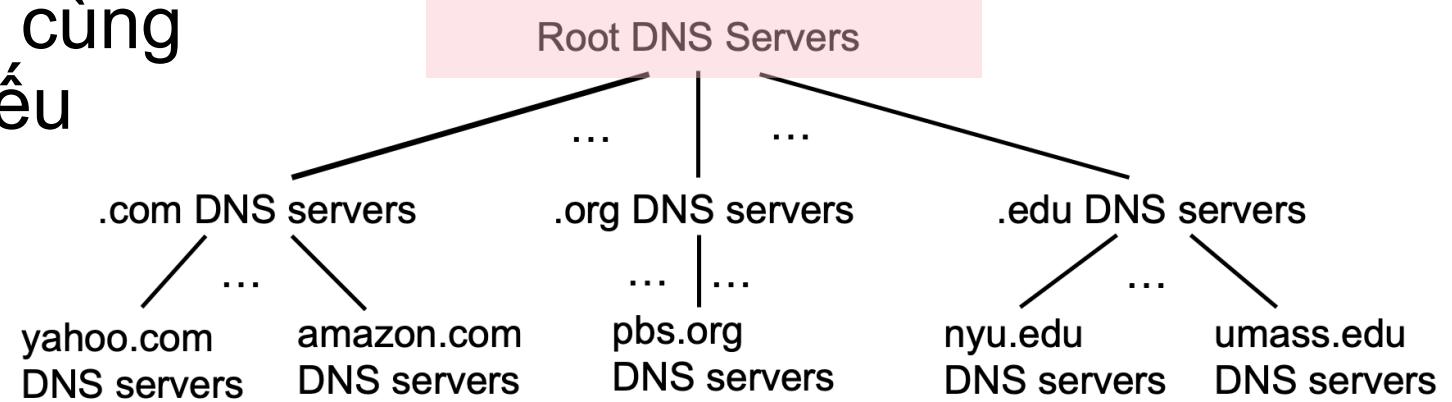


- Client muốn địa chỉ IP của [www.amazon.com](http://www.amazon.com); lần đầu tiên sẽ:
  - Client truy vấn máy chủ gốc (root) để tìm máy chủ DNS .com
  - Client truy vấn máy chủ DNS .com để tìm máy chủ DNS amazon.com
  - Client truy vấn máy chủ DNS amazon.com để lấy địa chỉ IP của [www.amazon.com](http://www.amazon.com)

# DNS: Máy chủ DNS gốc



- Chính thức, liên hệ cuối cùng bởi máy chủ tên miền nếu không thể phân giải tên

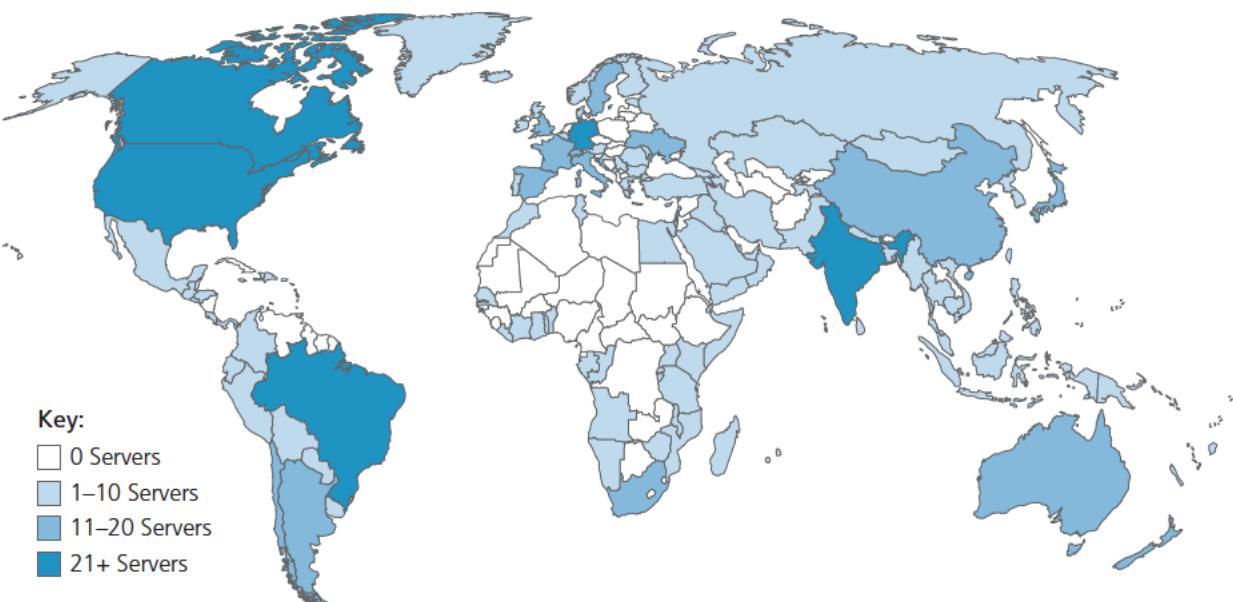




# DNS: Máy chủ DNS gốc

- Chính thức, liên hệ cuối cùng bởi máy chủ tên miền nếu không thể phân giải tên
- Chức năng Internet **cực kỳ quan trọng**
  - Internet không thể hoạt động mà không có nó!
  - DNSSEC - cung cấp bảo mật (xác thực, tính toàn vẹn của tin nhắn)
  - ICANN (Internet Corporation for Assigned Names and Numbers) quản lý miền DNS gốc

13 máy chủ gốc trên toàn thế giới  
Mỗi máy chủ được sao chép nhiều lần (~200 máy chủ ở US)

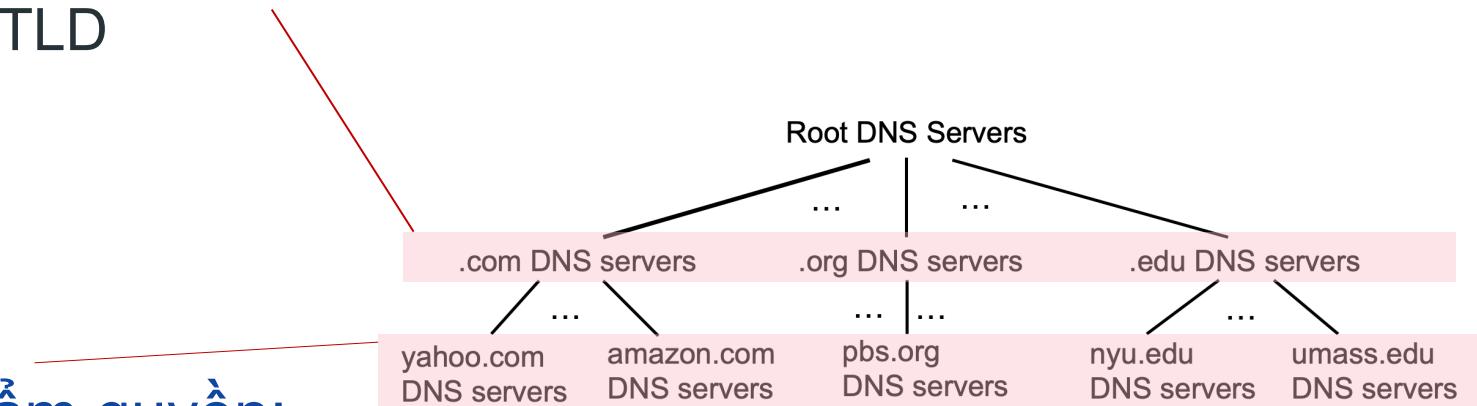




# Top-Level Domain (TLD), và Máy chủ có thẩm quyền

## ○ Các máy chủ miền cấp cao nhất (TLD):

- Chịu trách nhiệm cho tên miền .com, .org, .net, .edu, .aero, .jobs, .museums và tất cả các miền quốc gia cấp cao nhất, ví dụ: .cn, .uk, .fr, .ca, .jp
- Network Solutions: đăng ký có thẩm quyền cho .com, .net TLD
- Tổ chức Educause: .edu TLD



## ○ Các máy chủ DNS có thẩm quyền:

- (Các) máy chủ DNS riêng của tổ chức, cung cấp tên được cấp phép và ánh xạ địa chỉ IP cho các hệ thống đầu cuối được đặt tên của tổ chức
- Có thể được quản lý bởi tổ chức hoặc nhà cung cấp dịch vụ



# Máy chủ DNS cục bộ

- Khi host (hệ thống đầu cuối) thực hiện truy vấn DNS, truy vấn sẽ được gửi đến máy chủ DNS cục bộ của nó
  - Máy chủ DNS cục bộ trả về lời đáp, trả lời:
    - Từ bộ nhớ đệm cục bộ (local cache): các cặp dịch *tên-địa chỉ* gần đây (nhưng có thể hết hạn)
    - Chuyển tiếp yêu cầu vào hệ thống phân cấp DNS để phân giải
  - Mỗi ISP có máy chủ DNS cục bộ; cách tìm:
    - MacOS: % scutil --dns
    - Windows: >ipconfig /all
- Máy chủ DNS cục bộ không hoàn toàn thuộc về hệ thống phân cấp

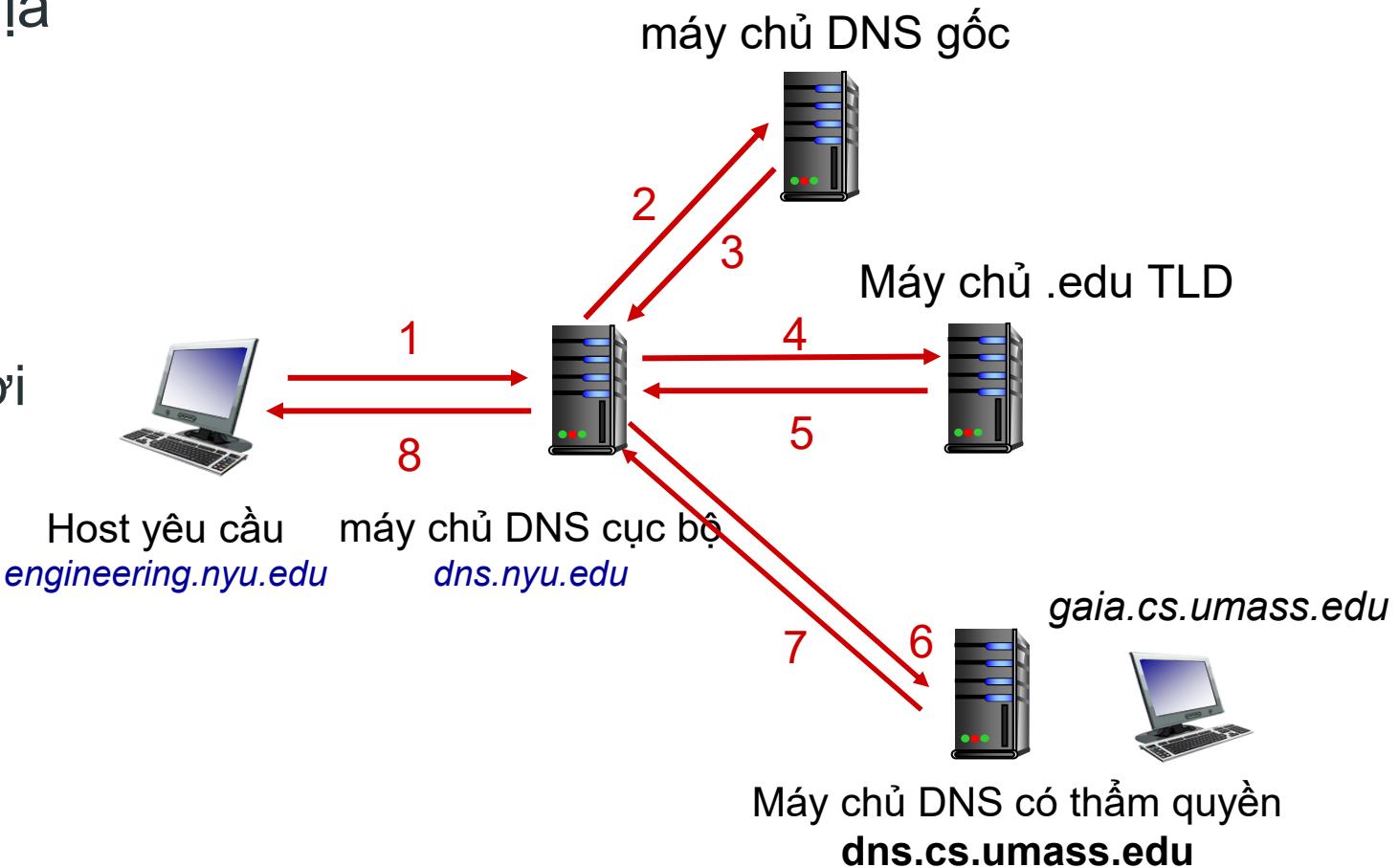


# Phân giải tên DNS: truy vấn tuần tự

Ví dụ: máy tại  
*engineering.nyu.edu* muốn có địa  
chỉ IP cho *gaia.cs.umass.edu*

## ○ Truy vấn tuần tự:

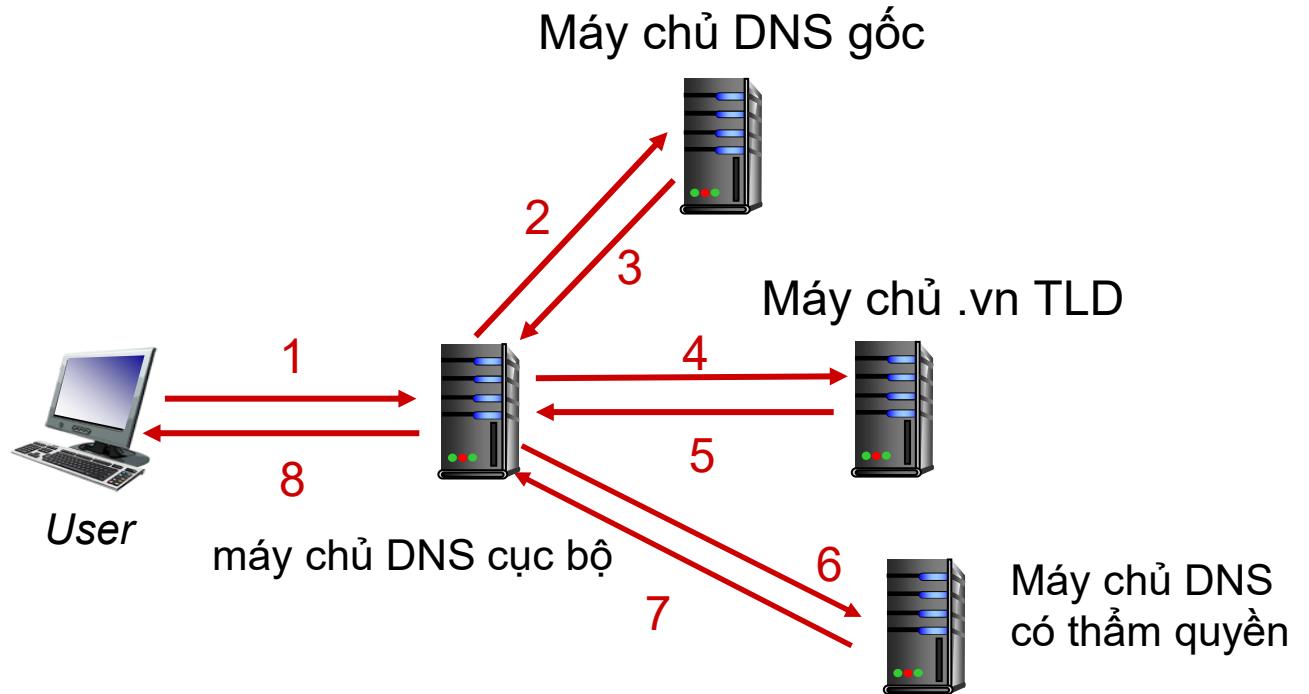
- Máy chủ được hỏi sẽ trả lời với tên của máy chủ quản lý vùng liên quan
- “tôi không biết tên này, nhưng hãy hỏi thêm thông tin từ máy chủ này”





# Ví dụ Truy vấn DNS

STT	Query/Answer
1	<b>Q:</b> who is nc.uit.edu.vn?
2	<b>Q:</b> who is nc.uit.edu.vn?
3	<b>A:</b> Ask TLD .vn
4	<b>Q:</b> who is nc.uit.edu.vn?
5	<b>A:</b> Ask name server
6	<b>Q:</b> who is nc.uit.edu.vn?
7	<b>A:</b> 118.69.123.140
8	<b>A:</b> 118.69.123.140



nc.uit.edu.vn  
118.69.123.140

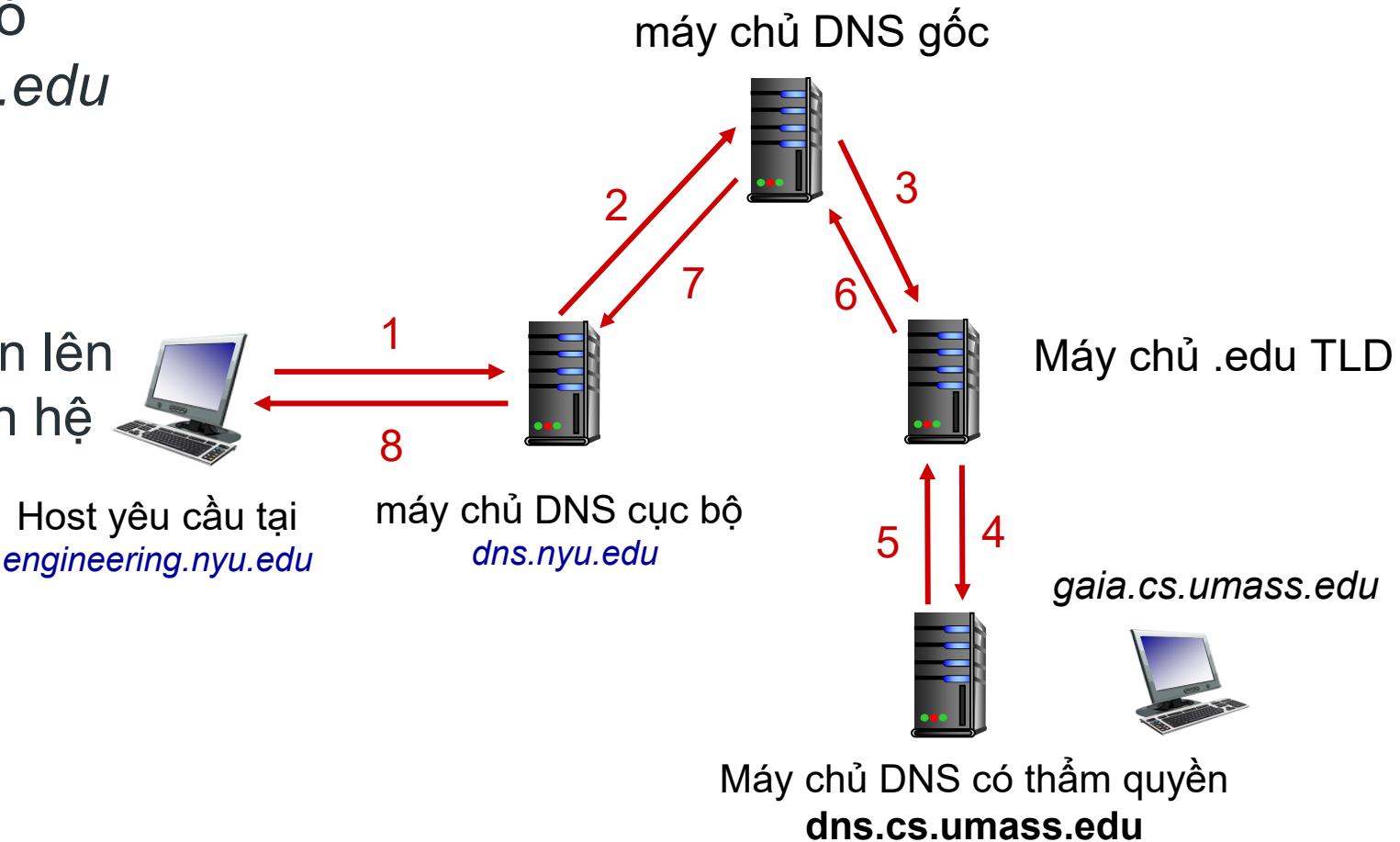


# Phân giải tên DNS: truy vấn đệ quy

Ví dụ: máy tại  
*engineering.nyu.edu* muốn có  
địa chỉ IP cho *gaia.cs.umass.edu*

## ○ Truy vấn đệ quy:

- Đặt gánh nặng phân giải tên lên máy chủ tên miền được liên hệ
- Tải nặng ở cấp trên của hệ thống phân cấp?





# Caching thông tin DNS

- Khi máy chủ tên miền học về 1 ánh xạ, nó sẽ lưu ánh xạ vào bộ nhớ cache và ngay lập tức trả về ánh xạ được lưu trong bộ nhớ cache để đáp ứng truy vấn
  - Bộ nhớ đệm cải thiện thời gian phản hồi
  - Các mục cache hết hạn (biến mất) sau một thời gian (TTL)
  - Thông tin trong các máy chủ TLD thường được lưu trong bộ nhớ cache trong các máy chủ DNS cục bộ
- Các mục được lưu trong bộ nhớ cache có thể lỗi thời (hết hạn)
  - Nếu máy được đặt tên thay đổi địa chỉ IP, có thể không được biết đến trên toàn Internet cho đến khi tất cả các TTL hết hạn!
  - Dịch từ tên đến địa chỉ nỗ lực tốt nhất!



**DNS: Cơ sở dữ liệu phân tán lưu trữ bản ghi tài nguyên (RR)**

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

## **type=A**

- name là tên máy( hostname)
- value là địa chỉ IP

## **type=NS**

- name là tên miền - domain (ví dụ: foo.com)
- value là tên của máy chủ tên miền có thẩm quyền quản lý tên miền này

## **type=CNAME**

- name là bí danh của một tên “gốc” (tên thực)
- www.ibm.com có tên thực là servereast.backup2.ibm.com
- value là tên gốc

## **type=MX**

- value là tên máy chủ thư SMTP liên kết với name

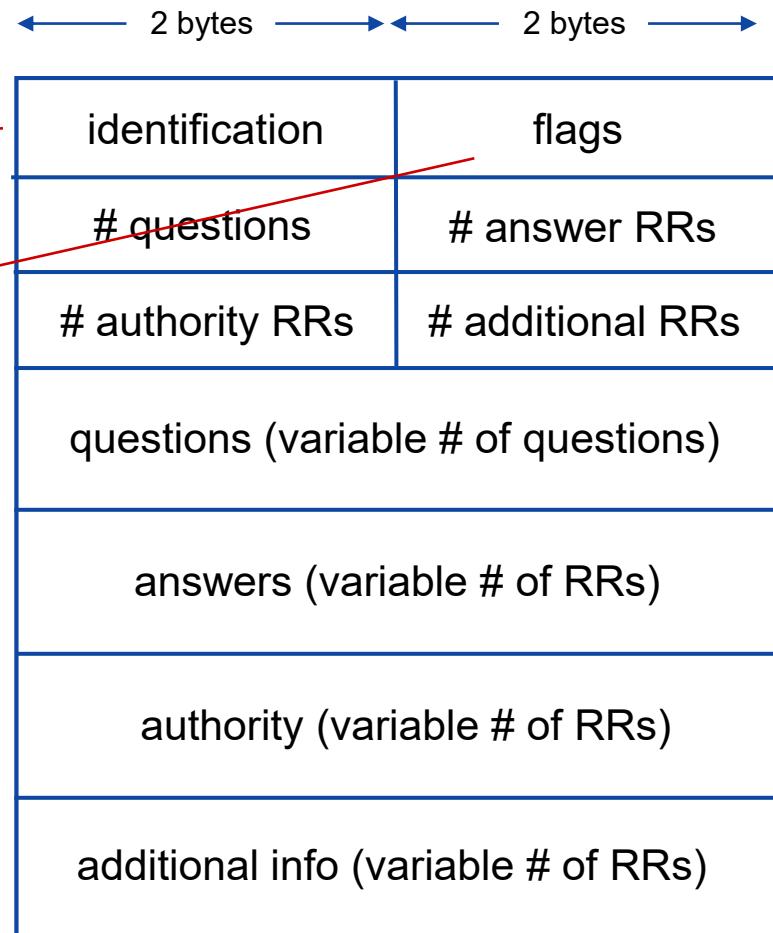
# Các thông điệp của giao thức DNS



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

## Phần header của thông điệp:

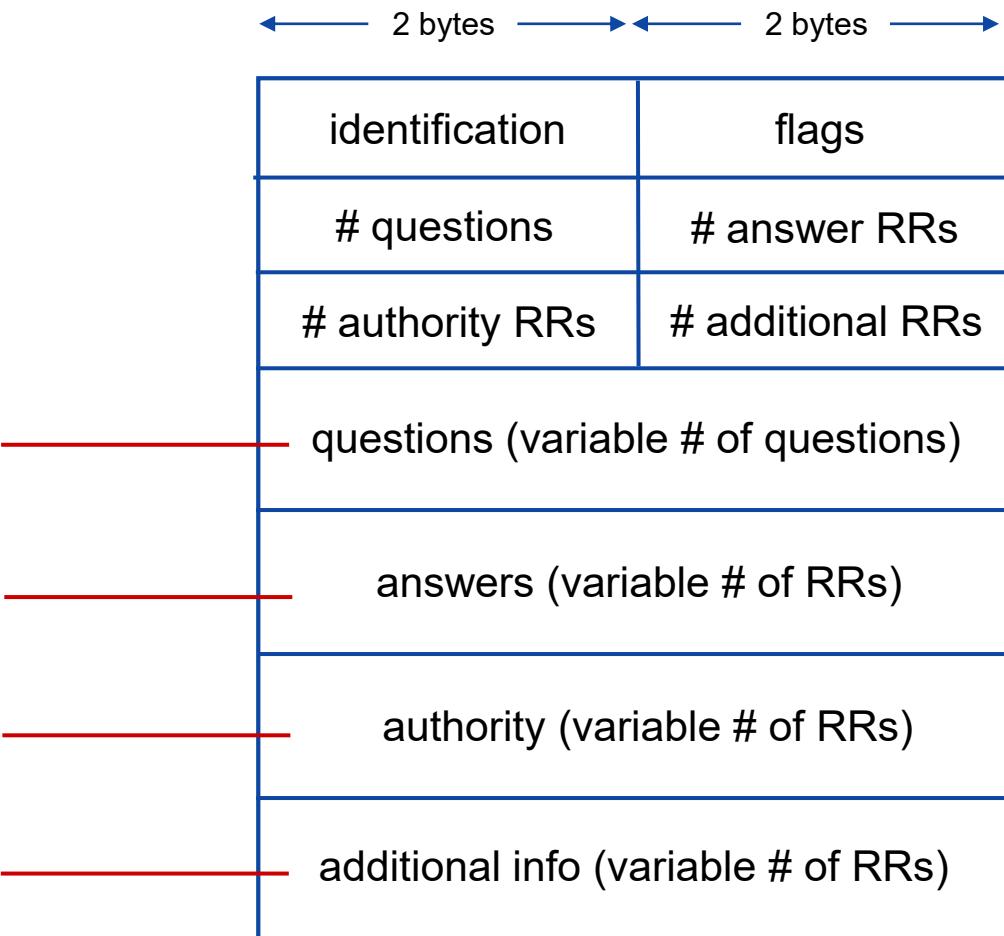
- identification: số 16 bit xác định 1 truy vấn, hoặc trả lời cho truy vấn có cùng số này
- Các cờ (flags):
  - Truy vấn hoặc trả lời
  - Mong muốn đệ quy
  - Đệ quy có sẵn
  - Trả lời là có thẩm quyền





# Các thông điệp của giao thức DNS

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



Các trường **name, type** của một truy vấn

Các RRs để trả lời truy vấn

Các bản ghi thông tin về các máy chủ có thẩm quyền

Thông tin “hữu ích” bổ sung có thể sẽ dùng



# Thêm các bản ghi vào trong DNS

- Ví dụ: khởi tạo mới công ty “Network Utopia”
  - Đăng ký tên miền networkuptopia.com tại một *DNS registrar – tổ chức nhận đăng ký tên miền* (như là Network Solutions)
    - Cung cấp tên, địa chỉ IP của máy chủ tên miền có thẩm quyền quản lý tên miền này (primary và secondary)
    - Tổ chức quản lý tên miền thêm hai bản ghi vào trong máy chủ quản lý vùng .com:
      - (networkutopia.com, dns1.networkutopia.com, NS)
      - (dns1.networkutopia.com, 212.212.212.1, A)
  - Tạo máy chủ có thẩm quyền cục bộ với địa chỉ IP 212.212.212.1
    - Bản ghi type A cho www.networkuptopia.com
    - Bản ghi type MX cho máy chủ thư điện tử thuộc networkutopia.com



# Tấn công DNS

## Tấn công DDoS

- Bắn phá các máy chủ gốc bằng lưu lượng truy cập
  - Không thành công cho đến nay
  - Lọc lưu lượng
  - Máy chủ DNS cục bộ lưu trữ IP của máy chủ TLD, cho phép bỏ qua máy chủ gốc
- Bắn phá các máy chủ TLD
  - Tiềm ẩn nhiều nguy hiểm hơn

## Các cuộc tấn công giả mạo

- Chặn truy vấn DNS, trả về câu trả lời không có thật
  - DNS cache poisoning (Tấn công đầu độc DNS cache)
  - RFC 4033: Dịch vụ xác thực DNSSEC



# Tầng Ứng dụng – Tổng quan

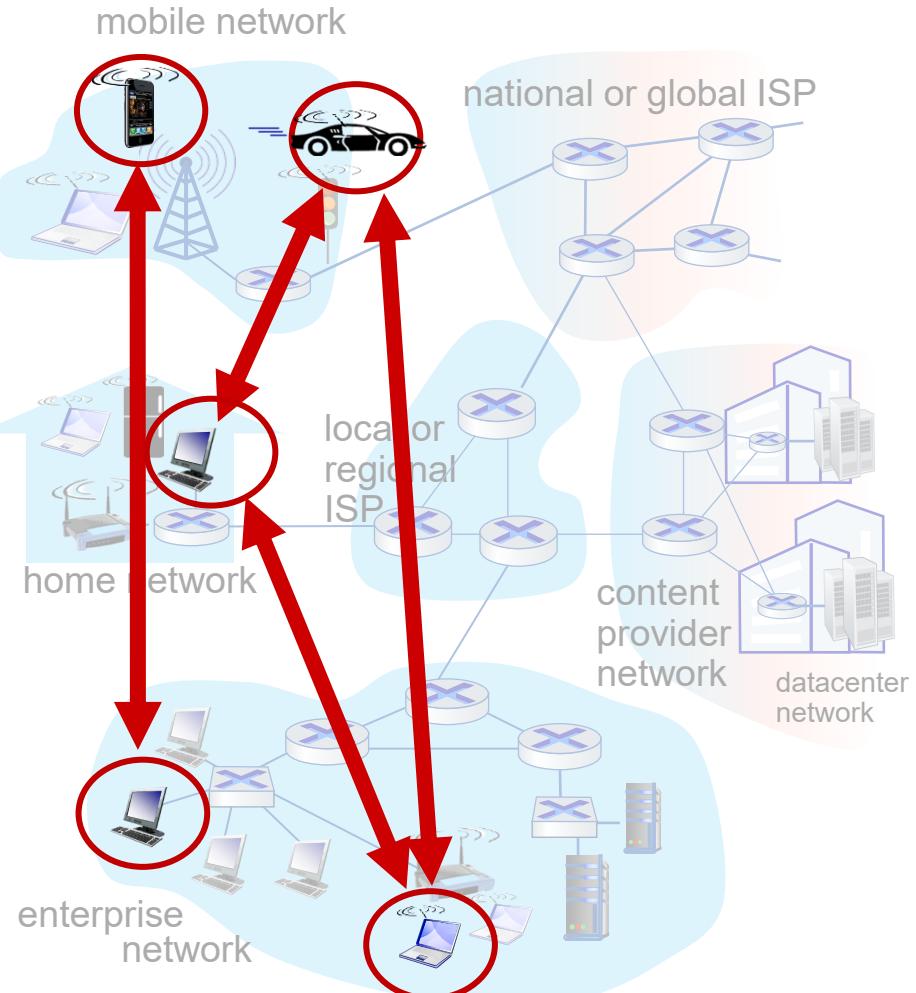
- Nguyên lý của các ứng dụng mạng
- Web và HTTP
- E-mail, SMTP, IMAP
- Hệ thống phân giải tên miền DNS
- **Các ứng dụng P2P**
- Lập trình socket với UDP và TCP





# Kiến trúc Peer-to-peer (P2P)

- **Không có** máy chủ luôn hoạt động
- Các hệ thống đầu cuối bất kỳ giao tiếp 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
- **Khả năng tự mở rộng** – các peer mới cung cấp thêm dịch vụ mới, cũng như có thêm nhu cầu mới về dịch vụ
- Các peer có thể kết nối không liên tục và thay đổi địa chỉ IP
  - Quản lý phức tạp
- Ví dụ: Chia sẻ tập tin P2P [BitTorrent]

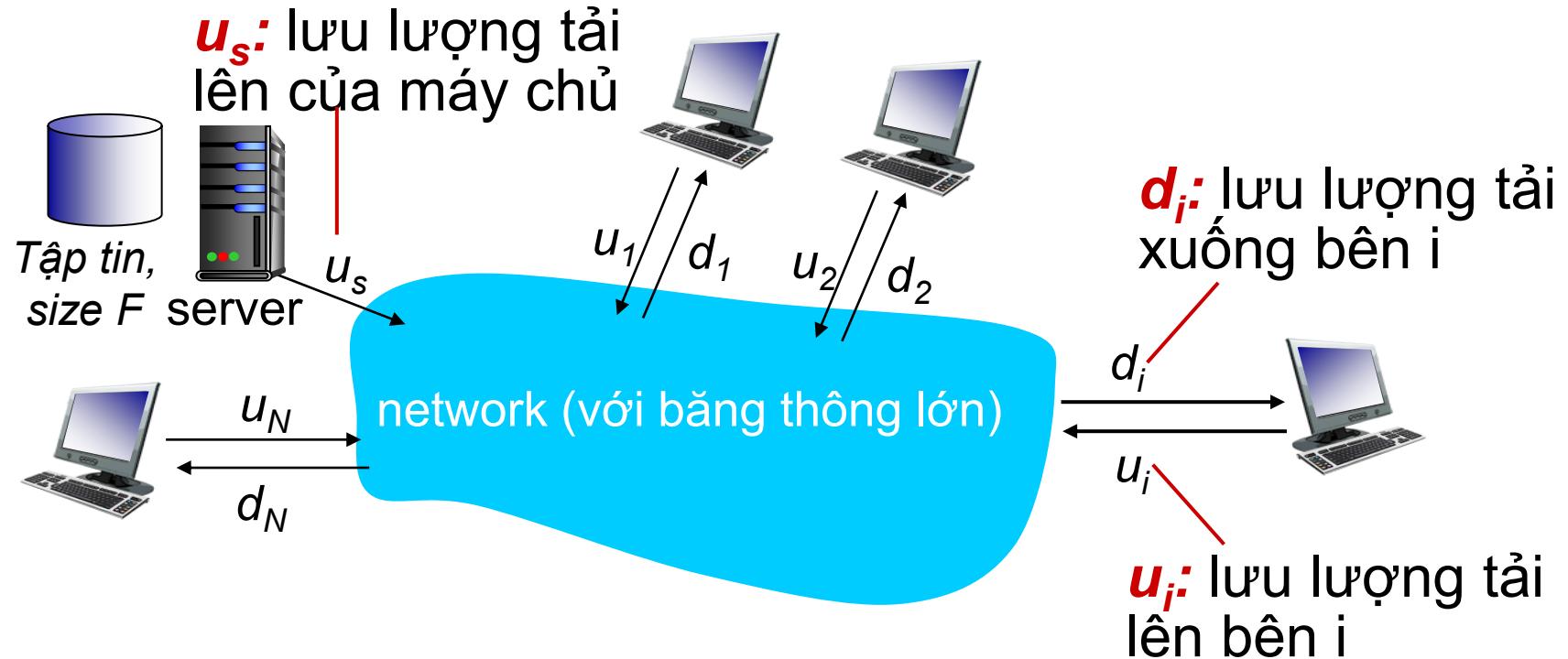


# Phân phối tập tin: Máy khách – máy chủ và P2P



Câu hỏi: mất bao lâu để phân phối tập tin (kích thước  $F$ ) từ một máy chủ đến  $N$  máy?

- Lưu lượng tải lên/tải xuống của bên (peer) là tài nguyên giới hạn





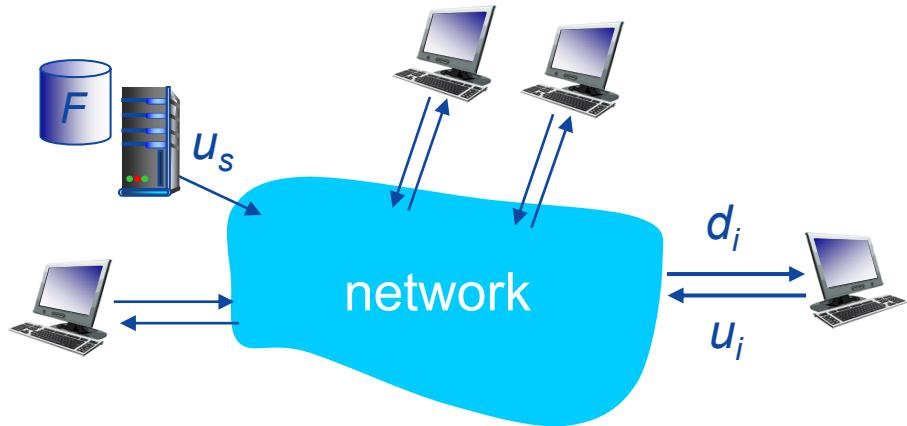
# Thời gian phân phối tập tin: máy khách-máy chủ

- **Máy chủ truyền:** phải gửi (tải lên) tuần tự N bản sao tập tin:

- Thời gian để gửi một bản sao:  $F/u_s$
- Thời gian để gửi N bản sao:  $NF/u_s$

- **Máy khách:** mỗi máy khách phải tải xuống bản sao của tập tin

- $d_{min}$  = tốc độ tải xuống của máy khách tải chậm nhất
- Thời gian để máy khách tải chậm nhất tải xong:  $F/d_{min}$



Thời gian để phân phối  
tập tin F đến N máy khách  
dùng phương pháp  
máy khách-máy chủ

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

Tăng tuyễn tính trong N



# Thời gian phân phối tập tin: P2P

- **Máy chủ:** chỉ cần tải lên ít nhất một bản sao

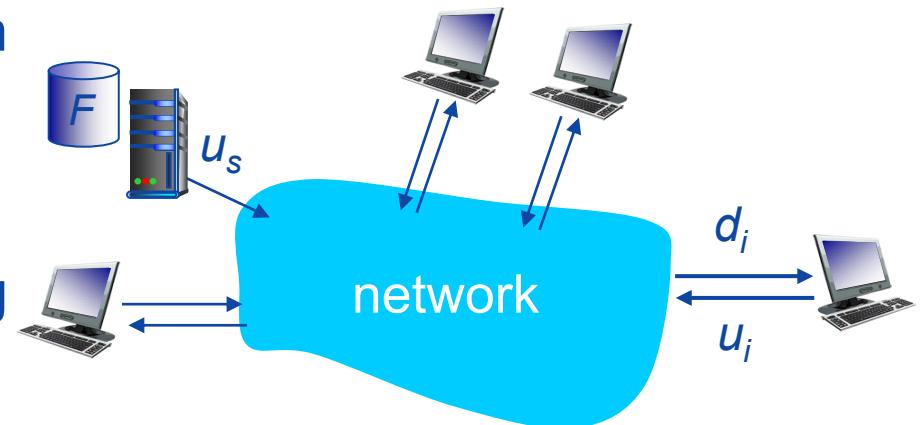
■ Thời gian gửi một bản sao:  $F/u_s$

- **Một máy khách:** máy khách phải tải xuống bản sao tập tin

■ Thời gian tối thiểu để máy khách tải xuống:  $F/d_{min}$

- **Nhiều máy khách:** với  $N$  máy thì tổng dung lượng tải về là  $NF$  bit

■ Tốc độ upload tối đa (không chế tốc độ tải về tối đa) là  $u_s + \sum u_i$



Tăng tuyễn tính trong  $N$  ...

Thời gian phân phối  
F đến  $N$  máy khách  
dùng phương pháp P2P

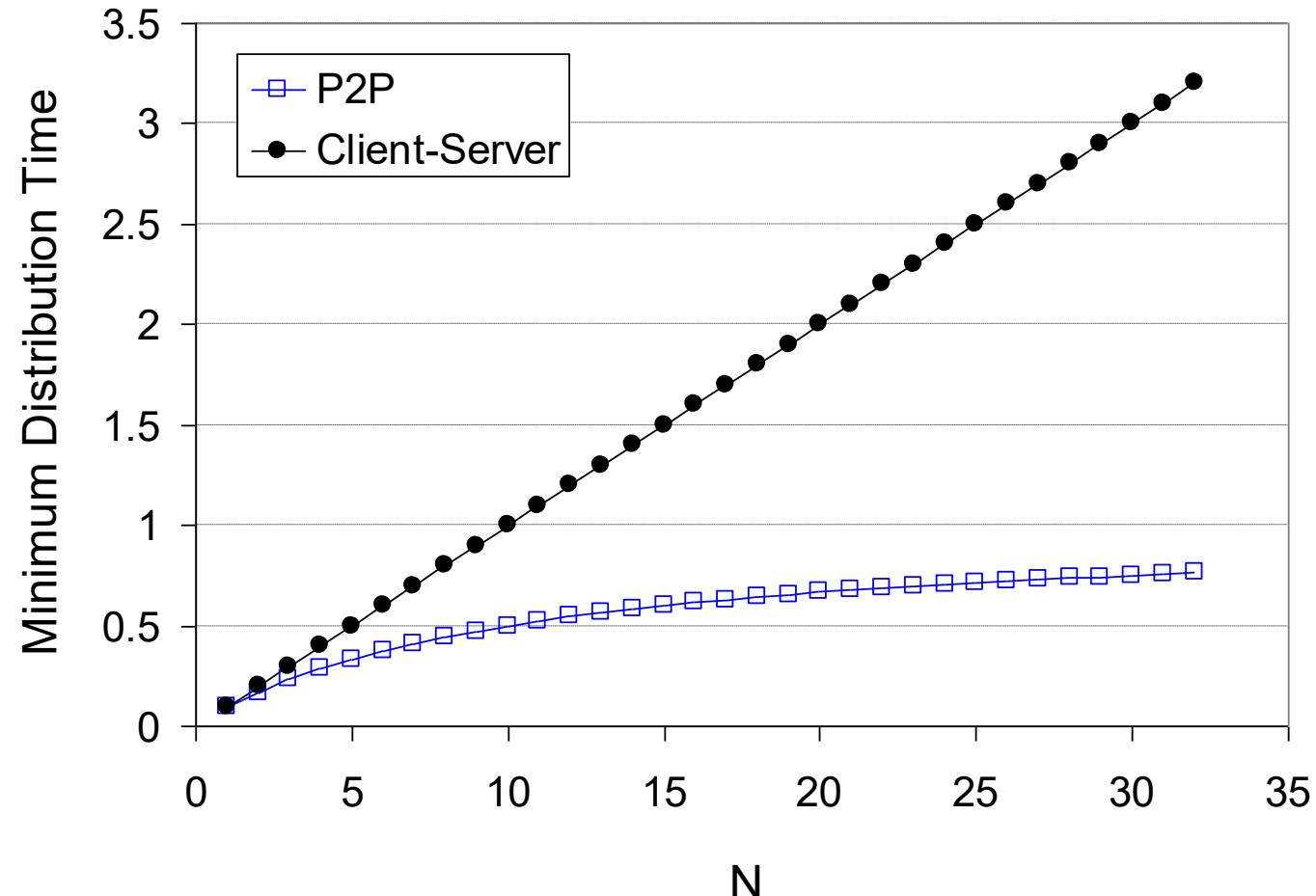
$$D_{P2P} > \max\{F/u_s, F/d_{min}, NF/(u_s + \sum u_i)\}$$

...nhưng mỗi khi thêm bên (peer) tham gia sử dụng dịch vụ,  
chính nó lại gia tăng năng lực dịch vụ

# So sánh máy khách-máy chủ với P2P: ví dụ



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

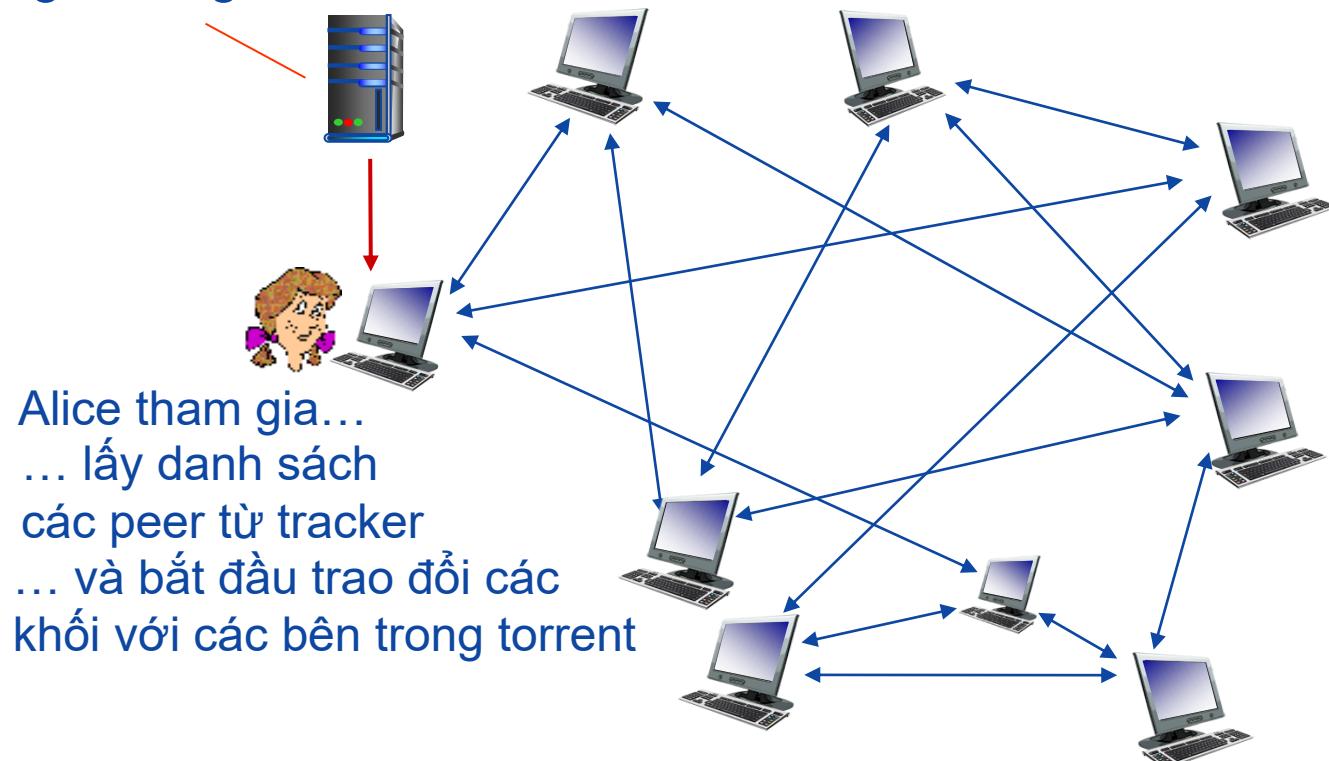




# Phân phối tập tin P2P: BitTorrent

- Tập tin được chia thành các khối 256Mb (chunks)
- Các bên (peer) trong torrent gửi/nhận các khối

*tracker:* theo dõi các bên (peers)  
tham gia trong torrent

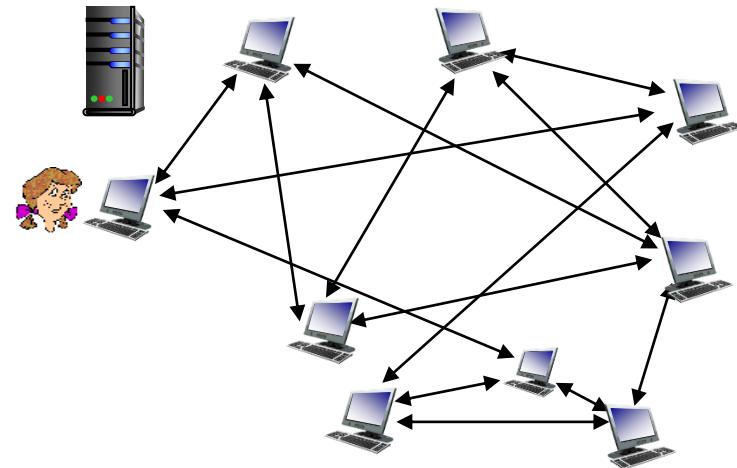


*torrent:* nhóm các bên trao đổi  
các khối



# Phân phối tập tin P2P: BitTorrent

- Bên (peer) tham gia torrent:
  - Không có các khói, nhưng sẽ lắn lượt tích lũy chúng từ các bên (peer) khác
  - Đăng ký với tracker để lấy danh sách các bên (peer) khác, kết nối với peer khác và cả “láng giềng” của các bên (peer) khác
- Trong khi tải xuống, bên (peer) tải các khói dữ liệu nó đã có tới các bên (peer) khác
- Bên (peer) có thể thay đổi các bên (peer) mà nó đang trao đổi các khói dữ liệu
- Các bên (peer) có thể tham gia hay rời bỏ nhóm phân phối
- Một khi bên (peer) có toàn bộ tập tin, nó có thể (ích kỷ) rời khỏi hoặc (vị tha) ở lại trong nhóm



# BitTorrent: requesting, sending file chunks



## ○ Yêu cầu các khối:

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

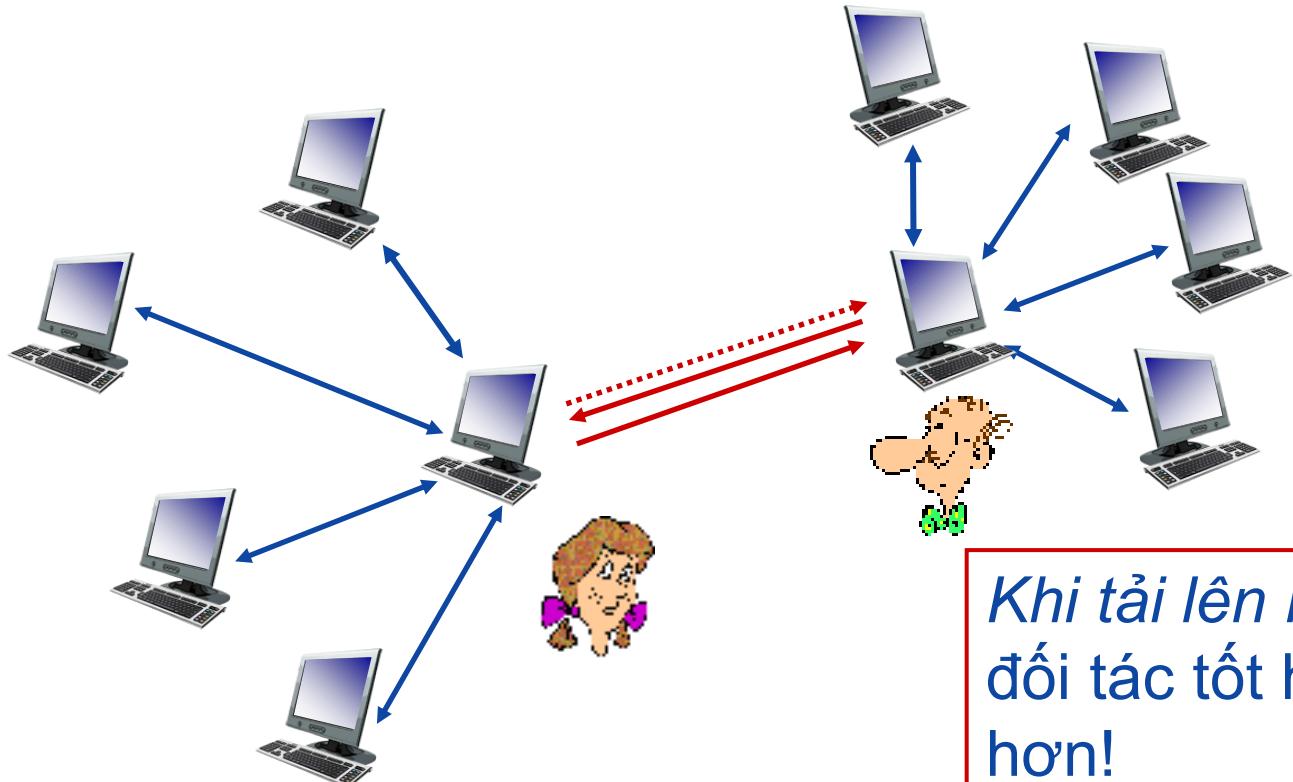
## ○ Gửi các khối: tit-for-tat

- Alice gửi các khối cho bốn bên (peer) nào hiện tại đang gửi các khối cho mình ở tốc độ cao nhất
  - Alice xiết các yêu cầu từ các peer khác (sẽ không nhận được khối từ Alice)
  - Đánh giá lại top 4 mỗi 10 giây
- Mỗi 30 giây: chọn ngẫu nhiên một peer khác, bắt đầu gửi các khối
  - Không xiết các yêu cầu của bên (peer) này
  - Bên (peer) mới được chọn có thể tham gia và top 4

# BitTorrent: tit-for-tat



- (1) Alice cho Bob kết nối
- (2) Alice trở thành một trong bốn nhà cung cấp hàng đầu của Bob; Bob đáp lại
- (3) Bob trở thành một trong bốn nhà cung cấp hàng đầu của Alice



*Khi tải lên nhanh hơn: dễ tìm được  
đối tác tốt hơn, lấy tập tin nhanh  
hơn!*

# Tầng Ứng dụng – Tổng quan



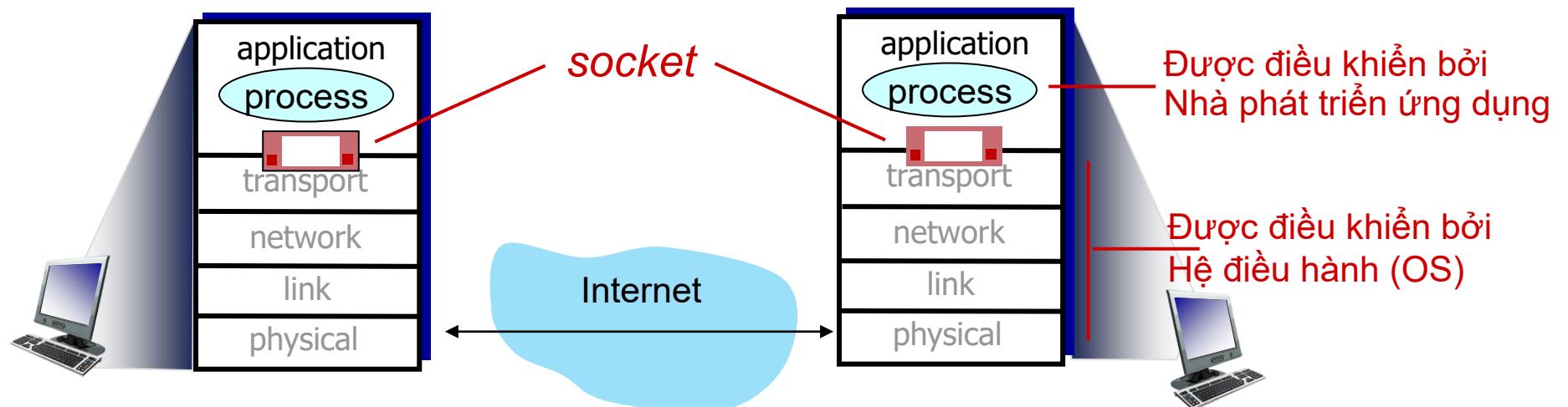
- Nguyên lý của các ứng dụng mạng
- Web và HTTP
- E-mail, SMTP, IMAP
- Hệ thống phân giải tên miền DNS
- Các ứng dụng P2P
- **Lập trình socket với UDP và TCP**



# Lập trình Socket



- Mục tiêu: tìm hiểu cách xây dựng các ứng dụng máy khách/máy chủ liên lạc bằng sockets
- Socket: một cánh cửa giữa tiến trình ứng dụng và giao thức vận chuyển giữa 2 đầu cuối





# Lập trình Socket

- Hai loại socket cho hai dịch vụ vận chuyển:

- UDP: chuyển gói tin không đảm bảo
- TCP: chuyển luồng tin có đảm bảo (stream-oriented)

## Ví dụ ứng dụng:

1. Máy khách đọc một dòng ký tự (dữ liệu) từ bàn phím của nó và gửi dữ liệu đến máy chủ
2. Máy chủ nhận dữ liệu và chuyển đổi các ký tự thành chữ hoa
3. Máy chủ gửi dữ liệu đã sửa đổi đến máy khách
4. Máy khách nhận dữ liệu đã sửa đổi và hiển thị dòng trên màn hình của nó



# Lập trình Socket với UDP

- UDP: không có "kết nối" giữa máy khách và máy chủ
  - Không bắt tay trước khi gửi dữ liệu
  - Bên gửi đính kèm rõ ràng địa chỉ IP đích và số cổng vào mỗi gói (packet)
  - Bên nhận trích xuất địa chỉ IP của người gửi và số cổng từ gói đã nhận

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

Quan điểm ứng dụng:

- UDP cung cấp cơ chế truyền các nhóm bytes ("datagrams") không tin cậy giữa các tiến trình của máy khách và máy chủ



# Tương tác socket máy khách/máy chủ: UDP

Máy chủ (chạy với địa chỉ IP máy chủ)



Tạo socket, port= x:  
`serverSocket =  
socket(AF_INET,SOCK_DGRAM)`

Đọc dữ liệu từ  
`serverSocket`

Ghi phản hồi đến  
`serverSocket`  
chỉ định địa chỉ IP và số cổng

Máy khách



Tạo socket:  
`clientSocket =  
socket(AF_INET,SOCK_DGRAM)`

Tạo datagram với địa chỉ IP máy chủ  
Và port = x; Gửi datagram qua  
`clientSocket`

Đọc dữ liệu từ  
`clientSocket`  
Đóng  
`clientSocket`



# Ví dụ: UDP client

## Python UDPClient

```
Bao gồm thư viện socket của Python → from socket import *
serverName = 'hostname'
serverPort = 12000
Tạo UDP socket → clientSocket = socket(AF_INET,
                                         SOCK_DGRAM)
Nhận thông tin nhập bằng bàn phím người dùng → message = input('Input lowercase sentence:')
Đính kèm tên máy chủ, cổng vào → clientSocket.sendto(message.encode(),
                                                        (serverName, serverPort))
tin nhắn; gửi vào Socket
Đọc dữ liệu trả lời (byte) từ Socket → modifiedMessage, serverAddress =
                                                clientSocket.recvfrom(2048)
In ra chuỗi nhận được và đóng Socket → print(modifiedMessage.decode())
                                                clientSocket.close()
```

# Ví dụ: UDP server



## Python UDPServer

```
from socket import *
serverPort = 12000
Tạo UDP socket → serverSocket = socket(AF_INET, SOCK_DGRAM)
Gắn socket với số cổng 12000 → serverSocket.bind(("", serverPort))
                                         print('The server is ready to receive')
Lặp vô hạn → while True:
Đọc từ UDP Socket vào tin nhắn, lấy địa chỉ
của máy khách (IP máy khách và cổng) →     message, clientAddress = serverSocket.recvfrom(2048)
                                                modifiedMessage = message.decode().upper()
Gửi chuỗi chữ hoa trả lại cho máy khách →     serverSocket.sendto(modifiedMessage.encode(),
                                         clientAddress)
```



# Lập trình Socket với TCP

- Máy khách phải liên lạc với máy chủ
  - Tiến trình máy chủ phải được chạy trước
  - Máy chủ phải tạo socket để mời máy khách đến liên lạc
- Máy khách tiếp xúc máy chủ bằng:
  - Tạo socket TCP, xác định địa chỉ IP, số cổng của tiến trình máy chủ
  - Khi máy khách tạo socket: máy khách TCP thiết lập kết nối đến máy chủ TCP
- Khi được máy khách liên lạc, *máy chủ TCP tạo socket mới* cho tiến trình máy chủ để trao đổi với máy khách đó
  - Cho phép máy chủ nói chuyện với nhiều máy khách
  - Số cổng nguồn được dùng để phân biệt các máy khách (xem tiếp chương 3)

## Quan điểm ứng dụng:

TCP cung cấp việc truyền các byte đáng tin cậy và theo thứ tự giữa máy khách và máy chủ



# Tương tác socket máy khách/máy chủ: TCP



Máy chủ (chạy trên HostID)



Máy khách

Tạo socket,  
port=x, cho các yêu cầu được gửi đến:

`serverSocket = socket()`

Chờ các yêu cầu kết nối  
được gửi đến  
`connectionSocket = connection setup`  
`serverSocket.accept()`

Đọc yêu cầu từ  
`connectionSocket`

Viết phản hồi đến  
`connectionSocket`

Đóng  
`connectionSocket`

Tạo socket,  
Kết nối đến hostid, port=x  
`clientSocket = socket()`

Gửi yêu cầu sử dụng  
`clientSocket`

Đọc phản hồi từ  
`clientSocket`

Đóng  
`clientSocket`

TCP

# Ví dụ: TCP client



## *Python TCPClient*

Tạo Socket TCP đến máy chủ, cổng từ xa 12000

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

Không cần đính kèm tên máy chủ, cổng



# Ví dụ: TCP server

## Python TCP Server

- Tạo Socket TCP chào đón
- Máy chủ bắt đầu lắng nghe các yêu cầu TCP đến
- Lặp vô hạn
- Máy chủ 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 máy khách này (nhưng không đóng socket chào đón)

```
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 True:
→     connectionSocket, addr = serverSocket.accept()
→     sentence = connectionSocket.recv(1024).decode()
→     capitalizedSentence = sentence.upper()
→     connectionSocket.send(capitalizedSentence.
→                           encode())
→     connectionSocket.close()
```



## Chương 2: Bảng chú giải – Glossary

Tiếng Anh	Tiếng Việt	Chú giải
Application Layer	Tầng Ứng dụng	
Client	Máy khách	
Server	Máy chủ	
P2P (Peer-to-peer)	Mạng ngang hàng	
Process	Tiến trình	
Socket		
Protocol	Giao thức	
Message	Thông điệp	
Request	Yêu cầu	
Round-Trip-Time (RTT)		Thời gian cho một gói nhỏ di chuyển từ máy khách đến máy chủ và ngược lại
Response	Phản hồi	
Header	Phần mào đầu	
IP Address	Địa chỉ IP	
Port	Cổng	



## Chương 2: Bảng chú giải – Glossary (tt)

Tiếng Anh	Tiếng Việt	Chú giải
HTTP		HyperText Transfer Protocol
Cookie		
Web cache		
Proxy		
Non-persistent	Không bền vững	
Persistent	Bền vững	
Utilization	Độ hiệu dụng/khả dụng	
Stateless	Không trạng thái	
Browser	Trình duyệt	
URL		Uniform Resource Locator
HTML		HyperText Markup Language
User agent		
Email	Thư điện tử	
SMTP		Simple Mail Transfer Protocol



## Chương 2: Bảng chú giải – Glossary (tt)

Tiếng Anh	Tiếng Việt	Chú giải
POP		Post Office Protocol
IMAP		Internet Mail Access Protocol
Mailbox	Hộp thư	
DNS	Hệ thống phân giải tên miền	Domain Name System
Top Level Domain (TLD)		
Authoritative Server	Máy chủ có thẩm quyền	
Local Name Server	Máy chủ tên miền cục bộ	
TTL		
Query	Truy vấn	
Reply	Trả lời	
Name Resolution	Phân giải tên miền	



## Chương 2: Bảng chú giải – Glossary (tt)

Tiếng Anh	Tiếng Việt	Chú giải
Transport Layer	Tầng Vận chuyển	
UDP		
TCP		
Connection-oriented	Kết nối có hướng	
Reliable	Tin cậy	
Unreliable	Không tin cậy	
Datagram		



## Chương 2: Tóm tắt

- Kiến trúc ứng dụng
  - Client-server (Máy khách – máy chủ)
  - P2P (Mạng ngang hàng)
- Các yêu cầu dịch vụ của ứng dụng:
  - Độ tin cậy, băng thông, độ trễ
- Mô hình dịch vụ vận chuyển Internet
  - Hướng kết nối, đáng tin cậy: TCP
  - Không tin cậy, datagrams: UDP
- Giao thức cụ thể:
  - HTTP
  - SMTP, IMAP
  - DNS
- Lập trình Socket:
  - Socket UDP, TCP



## Chương 2: Tóm tắt

- Quá trình trao đổi thông điệp yêu cầu/phản hồi điển hình:
  - Máy khách yêu cầu thông tin hoặc dịch vụ
  - Máy chủ phản hồi bằng dữ liệu, mã trạng thái
  - Định dạng thông điệp:
  - Phần headers: Các trường cung cấp thông tin về dữ liệu
  - Dữ liệu: Thông tin (payload) đang được truyền đạt

- Chủ đề quan trọng:
  - Tập trung so với phi tập trung
  - Không trạng thái và có trạng thái
  - Khả năng mở rộng
  - Truyền tin cậy so và không tin cậy
  - "Sự phức tạp tại mạng biên"

**Quan trọng nhất: đã học về các giao thức!**