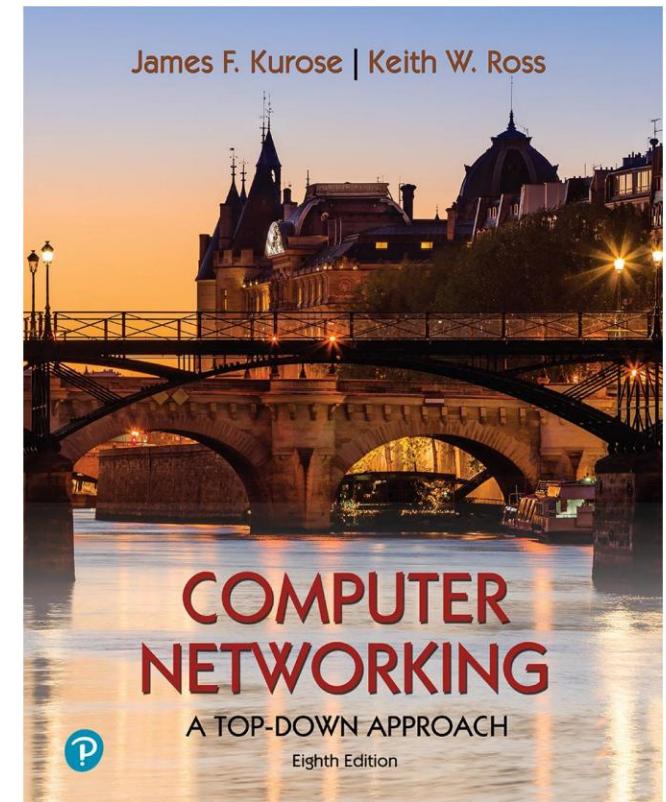


# chương 2

# Lớp ứng dụng



Mạng máy tính: A  
Cách tiếp cận từ trên xuống  
phiên bản thứ 8  
Jim KuroseKeith Ross  
Pearson, 2020

# Lớp Ứng dụng: tổng quan

Nguyên tắc ứng dụng  
mạng Web và HTTP

E-mail, SMTP, IMAP

Hệ thống tên miền (DNS)

Ứng dụng P2P

Mạng phân phối nội dung và  
truyền phát video

lập trình ổ cắm với  
UDP và TCP



# Lớp Ứng dụng: tổng quan

Mục tiêu

của chúng tôi:

khía cạnh khái niệm và triển khai của các giao thức tầng

ứng dụng • mô hình dịch vụ  
tầng vận chuyển

- mô hình máy khách-máy chủ
- mô hình ngang hàng

tìm hiểu về các giao thức  
bằng cách kiểm tra các giao  
thức tầng ứng dụng phổ biến •  
HTTP

- SMTP, IMAP
- DNS

lập trình các ứng  
dụng mạng • socket  
API

# Lớp Ứng dụng: tổng quan

Nguyên tắc ứng dụng  
mạng Web và HTTP

E-mail, SMTP, IMAP

Hệ thống tên miền (DNS)

Ứng dụng P2P

Mạng phân phối nội dung và  
truyền phát video

lập trình ổ cắm với  
UDP và TCP



# Một số ứng dụng mạng

mạng xã hội

thoại qua IP (ví dụ: Skype)

Web

hội nghị truyền hình thời gian thực

tin nhắn văn bản

Tìm kiếm trên Internet

e-mail

đăng nhập từ xa

trò chơi mạng nhiều người dùng

•

phát trực tuyến video được lưu trữ  
(YouTube, Hulu, Netflix)

Chia sẻ tệp P2P

Q: yêu thích của bạn?

# Tạo một ứng dụng mạng

viết các chương trình:

chạy trên các **hệ thống đầu cuối** (khác nhau)

**giao tiếp qua mạng**

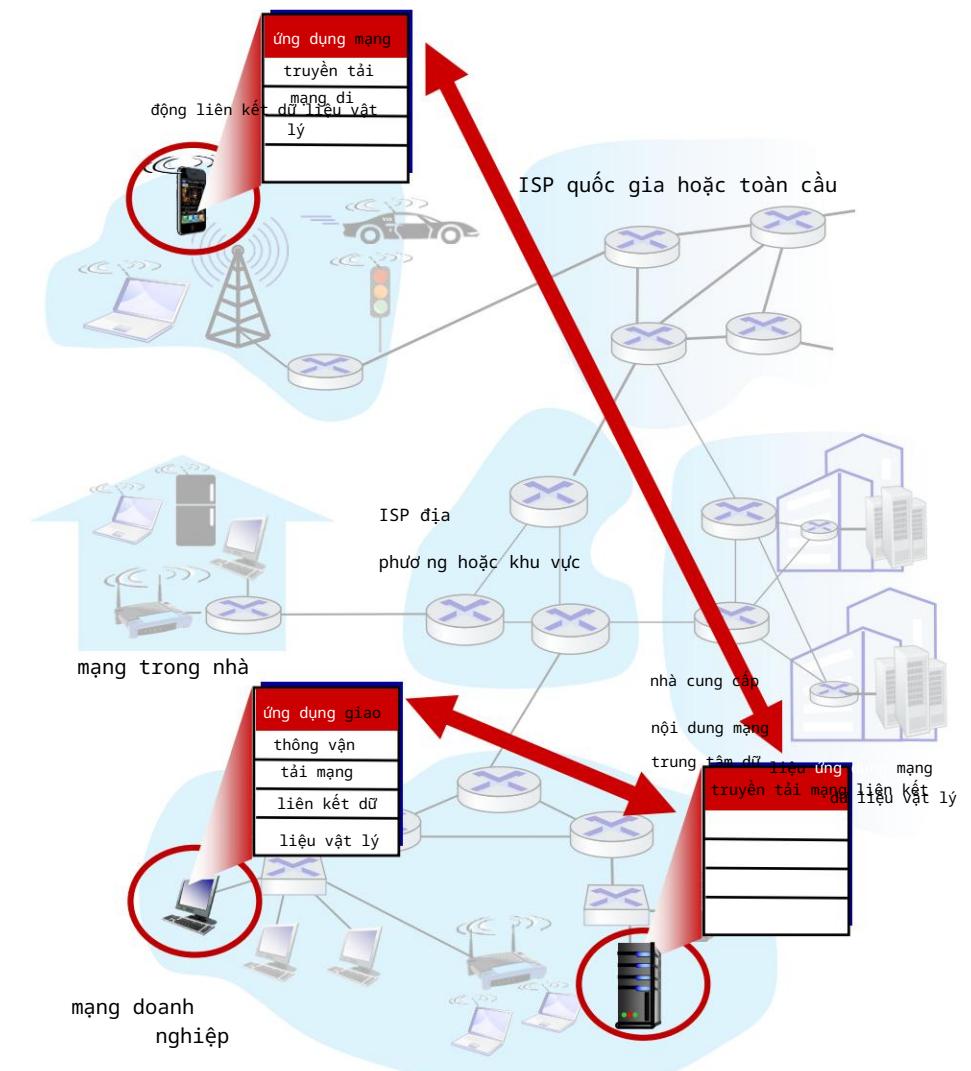
ví dụ: phần mềm máy chủ web

giao tiếp với phần mềm trình duyệt

không cần viết phần mềm cho các thiết bị lõi mạng

thiết bị lõi mạng không chạy người dùng các ứng dụng

ứng dụng trên hệ thống đầu cuối cho phép **phát triển**, phổ biến ứng dụng nhanh chóng



# Mô hình máy khách-máy chủ

người phục vụ:

máy chủ luôn bật

địa chỉ IP cố định

thường ở trung tâm dữ liệu, để mở rộng quy mô

khách hàng:

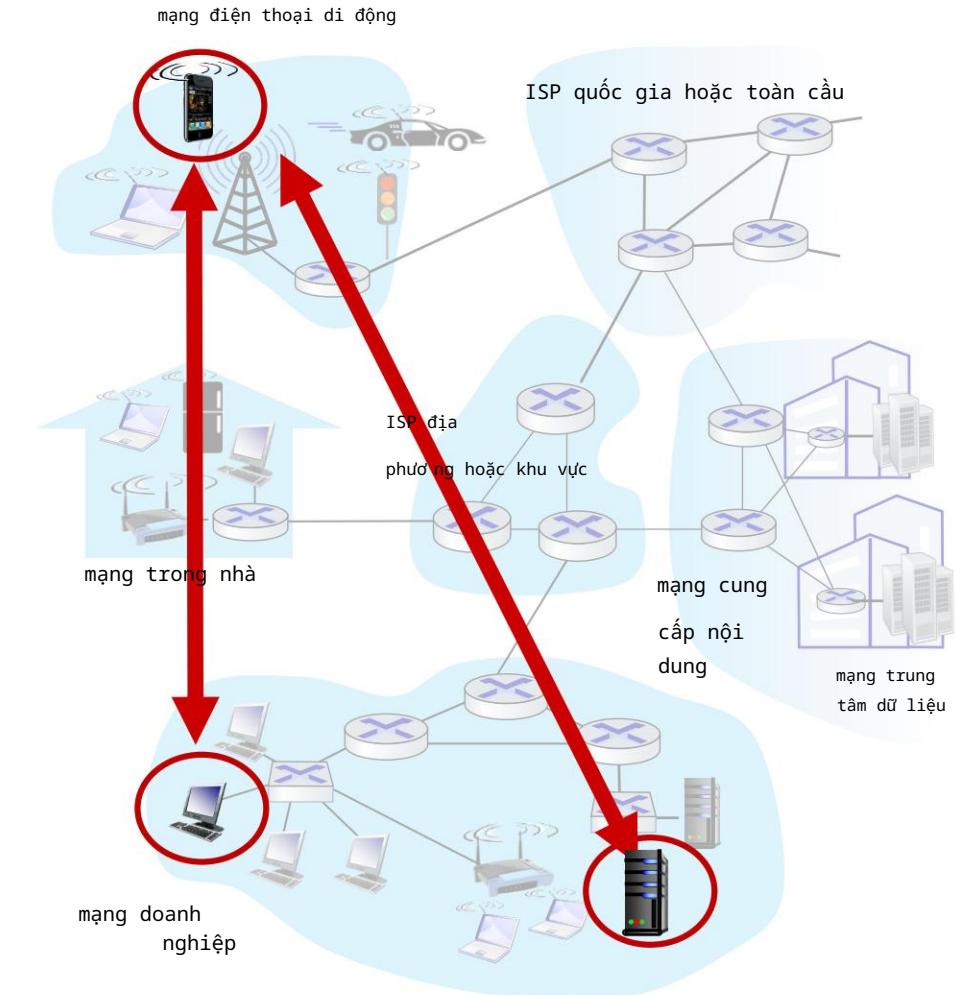
liên lạc, giao tiếp với máy chủ có thể

kết nối không liên tục có thể có địa chỉ

IP động không giao tiếp trực tiếp với

nhau

- ví dụ: HTTP, IMAP, FTP



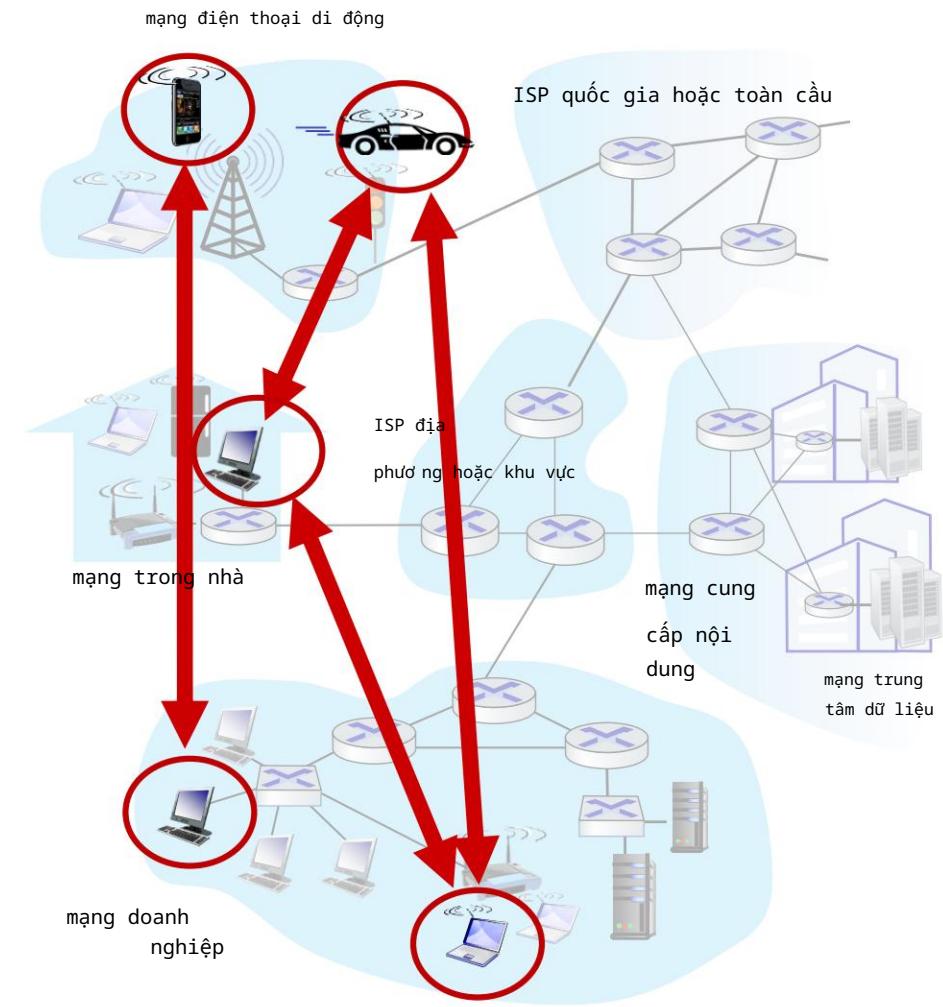
# kiến trúc ngang hàng

không có máy chủ luôn hoạt

động các hệ thống đầu cuối tùy ý giao  
tiếp trực tiếp

các đồng nghiệp yêu cầu dịch vụ từ các đồng  
nghiệp khác, cung cấp dịch vụ để đáp lại các  
đồng nghiệp khác • khả năng tự mở rộng - các  
đồng nghiệp mới mang lại khả năng dịch vụ mới,  
cũng như nhu cầu dịch vụ mới

các đồng nghiệp được kết nối không  
liên tục và thay đổi địa chỉ IP •  
quản lý phức tạp , ví dụ: Chia sẻ  
tệp P2P BitTorrent



# Quá trình giao tiếp

**tiến trình:** chươ ng trình chạy  
trong máy chủ

trong cùng một máy chủ, hai  
quy trình giao tiếp bằng cách  
sử dụng **giao tiếp giữa các  
quy trình** (được xác định bởi  
hệ điều hành)

các tiến trình trong các máy chủ  
khác nhau giao tiếp bằng cách **trao  
đổi thông điệp**

máy khách, máy chủ

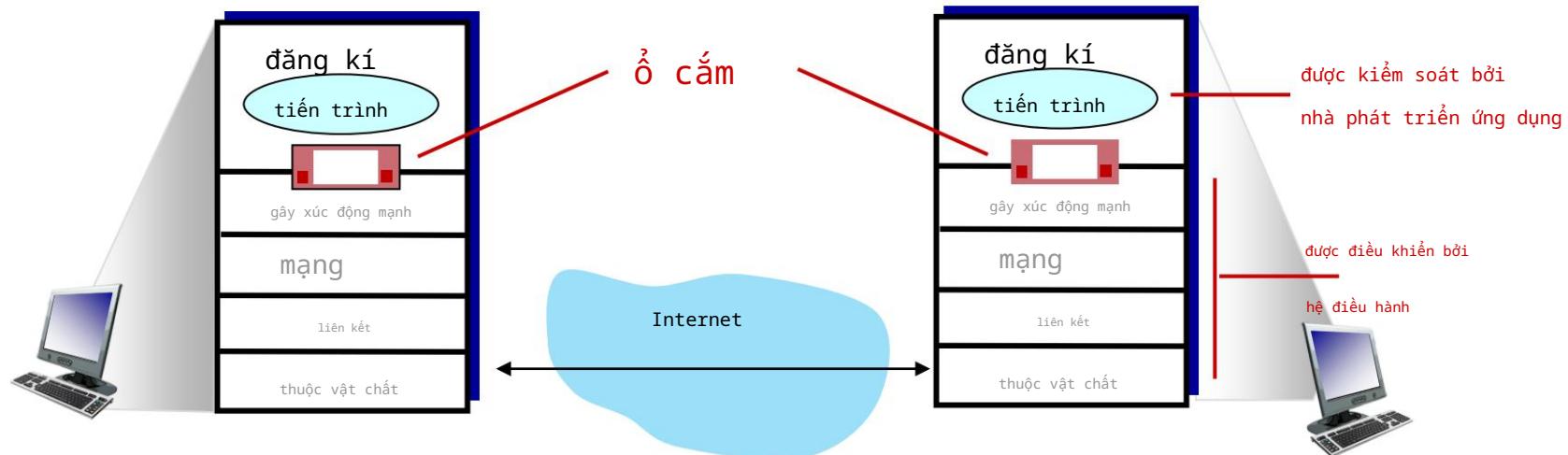
**quy trình máy khách:** quy trình đó  
bắt đầu giao tiếp

**quy trình máy chủ:** quy trình  
mà chờ đợi để được liên lạc

các ứng dụng ghi chú với  
Kiến trúc P2P có thể **có cả**  
quy trình máy khách và quy  
trình máy chủ

# Ổ cắm

tiến trình gửi/nhận thông điệp đến/từ **ổ cắm** của nó  
**ổ cắm** tương tự như một cánh cửa • **tiến trình gửi** sẽ  
 đẩy thông điệp ra khỏi cửa • **tiến trình gửi** dựa vào  
 cơ sở hạ tầng truyền tải ở phía bên kia của cánh cửa để chuyển thông  
 điệp đến **ổ cắm** ở **tiến trình nhận**  
 hai **ổ cắm** có liên quan: một ở mỗi bên



# xử lý địa chỉ

để nhận tin nhắn, quá trình nhận  
phải có mã **định danh** thiết bị  
**chủ** có 32-bit duy nhất  
địa chỉ IP

Hỏi: **địa chỉ IP** của máy chủ mà  
tiến trình chạy trên đó có **đủ** để  
xác định tiến trình không? **A:**  
**không**, **nhiều tiến trình** có thể  
chạy trên cùng một máy chủ

mã **định danh** bao gồm cả **địa chỉ IP** và  
**số cổng** được liên kết với quy trình  
trên máy chủ. **số cổng ví dụ:**

- Máy chủ HTTP: 80
- máy chủ thư: 25

để gửi thông báo HTTP đến  
máy chủ web “gaia.cs.umass.edu” :

- **Địa chỉ IP:** 128.119.245.12
- **số cổng:** 80

trong thời gian ngắn nữa.

# Một giao thức tầng ứng dụng định nghĩa:

các loại tin nhắn được trao đổi, ví dụ:

- yêu cầu, phản hồi

cú pháp tin nhắn: •

những trường nào trong tin nhắn  
& cách các trường được phân định

ngữ nghĩa thông điệp

- ý nghĩa của thông tin trong  
các trường

quy tắc về thời gian và cách thức

quy trình gửi và trả lời tin nhắn

giao thức mở: được

định nghĩa trong RFC, mọi người đều  
có quyền truy cập vào định nghĩa  
giao thức

cho phép khả năng tương tác ,

- ví dụ: HTTP, SMTP

giao thức độc quyền: ví

- dụ, Skype

# Ứng dụng cần dịch vụ vận tải nào?

tính toàn vẹn của

dữ liệu một số ứng dụng (ví dụ:  
truyền tệp, giao dịch web) yêu cầu  
**truyền dữ liệu đáng tin cậy 100%**

các ứng dụng khác (ví dụ: âm thanh) có  
thể **chịu một số mất mát**

thông lượng

một số ứng dụng (ví dụ: đa  
phươ ng tiện) yêu cầu **lượng  
thông lượng tối thiểu** để “hiệu quả”

các ứng dụng khác (“**ứng dụng đàm  
hồi**”) tận dụng bất kỳ thông lượng  
nào chúng nhận được

thời

**gian** một số ứng dụng (ví  
dụ: điện thoại Internet, trò chơi  
tươn tác) yêu cầu **độ trễ thấp** để “hiệu quả”

bảo mật

**mã hóa, toàn vẹn dữ liệu,**  
.

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

đăng kí	mất dữ liệu	thông lượng	thời điểm nhận yêu cầu?
chuyển tập tin/tải xuống e-mail	không mất mát	đàn hồi	không
tài liệu web	không mất mát	đàn hồi	không
âm thanh/video thời gian thực	chịu thua lỗ	âm thanh: 5Kbps-1Mbps video: 10Kbps-5Mbps	vâng, 10 mili giây
phát trực tuyến trò chơi	chịu thua lỗ	giống như trên	có, vài giây có,
tương tác âm thanh/video nhắn tin văn bản	chịu thua lỗ	Kbps+	10 giây có và không
	không mất mát	đàn hồi	

# Dịch vụ giao thức truyền tải Internet

## Các dịch vụ TCP:

vận chuyển đáng tin cậy giữa quá trình gửi và nhận kiểm soát luồng: người gửi sẽ không áp đảo người nhận

kiểm soát tắc nghẽn: điều tiết người gửi khi mạng quá tải

không cung cấp thời gian, đảm bảo thông lượng tối thiểu, bảo mật

hướng kết nối: yêu cầu thiết lập giữa các quy trình máy khách và máy chủ

## Dịch vụ UDP:

truyền dữ liệu không đáng tin cậy giữa quá trình gửi và nhận

không cung cấp độ tin cậy, kiểm soát luồng, kiểm soát tắc nghẽn, thời gian, đảm bảo thông lượng, bảo mật hoặc thiết lập kết nối.

Q: tại sao bạn tâm? Tại sao lại có UDP?

# Dịch vụ giao thức truyền tải Internet

đăng kí	giao thức lớp ứng dụng	giao thức vận chuyển
chuyển tập tin/tải xuống	FTP [RFC 959]	TCP
e-mail	SMTP [RFC 5321]	TCP
tài liệu web	HTTP 1.1 [RFC 7320]	TCP
điện thoại Internet	SIP [RFC 3261], RTP [RFC 3550] hoặc HTTP [RFC 7320]	TCP hoặc UDP
phát trực tuyến trò chơi	độc quyền , DASH WOW, FPS	TCP
tương tác âm thanh/video	(độc quyền)	UDP hoặc TCP

# Bảo mật TCP

Ổ cắm Vanilla TCP & UDP:

không mã hóa

mật khẩu văn bản rõ ràng được gửi vào ổ cắm đi qua Internet ở dạng văn bản rõ ràng (!)

Bảo mật tầng vận chuyển (TLS) cung cấp các kết nối TCP được mã hóa tính toàn vẹn của dữ liệu xác thực điểm cuối

TLS được triển khai trong các ứng dụng lớp ứng dụng

sử dụng các thư viện TLS, lần lượt sử dụng TCP

API ổ cắm TLS

văn bản rõ ràng được gửi vào ổ cắm đi qua Internet được mã hóa (xem thêm ở Chương 8)

# Lớp Ứng dụng: tổng quan

Nguyên tắc ứng dụng  
mạng Web và HTTP

E-mail, SMTP, IMAP

Hệ thống tên miền  
DNS

Ứng dụng P2P

Mạng phân phối nội dung và  
truyền phát video

lập trình ổ cắm với  
UDP và TCP



# Web và HTTP

Đầu tiên, xem xét nhanh.

trang web bao gồm các **đối tượng**, mỗi đối tượng có thể được lưu trữ trên các máy chủ Web khác nhau

đối tượng có thể là tệp HTML, hình ảnh JPEG, ứng dụng Java, tệp âm thanh,... trang web bao gồm một **tệp HTML cơ sở** bao gồm **một số đối tượng được tham chiếu**, mỗi đối tượng có thể định địa chỉ bằng một **URL**, ví dụ:

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

tên máy chủ

tên đường dẫn

# tổng quan về HTTP

**HTTP: giao thức truyền tải siêu văn bản**

Giao thức tầng ứng dụng của Web mô hình máy khách/máy chủ:

- **máy khách:** trình duyệt yêu cầu, nhận (sử dụng giao thức HTTP) và “hiển thị” các đối tượng Web
- **máy chủ:** Máy chủ Web gửi (sử dụng giao thức HTTP) để đáp ứng các yêu cầu



# Tổng quan về HTTP (tiếp theo)

HTTP sử dụng TCP:

máy khách bắt đầu kết nối TCP

(tạo ổ cắm) đến máy chủ, cổng 80

máy chủ chấp nhận kết nối TCP từ máy  
khách

Thông báo HTTP (thông báo giao thức tầng  
ứng dụng ) được trao đổi giữa trình duyệt  
(máy khách HTTP) và

Máy chủ web (máy chủ HTTP)

Kết nối TCP đã đóng

HTTP là "không quốc tịch"  
máy chủ không lưu giữ  
thông tin về các yêu cầu trước đây  
của máy khách

các  
giao thức duy trì "trạng thái" rất  
phức tạp! lịch sử (trạng thái)  
trong quá khứ phải được duy trì  
  
nếu máy chủ/máy khách gặp sự cố, lượt xem của chúng  
của "nhà nước" có thể không nhất quán, phải được  
dung hòa

# Kết nối HTTP: hai loại

## HTTP không liên tục 1.

Đã mở kết nối TCP 2. tối đa **một** đối tượng được gửi **qua kết nối TCP**

3. Kết nối TCP đã đóng  
tải xuống nhiều đối  
tượng yêu cầu nhiều kết  
nối

## HTTP liên tục

Kết nối TCP được mở tới  
Một máy chủ

Nhiều đối tượng có thể được  
gửi **qua một kết nối TCP** giữa  
máy khách và máy chủ đó

Kết nối TCP đã đóng

# HTTP không liên tục: ví dụ

Người dùng nhập URL: `www.someSchool.edu/someDepartment/home.index` (chứa văn bản, tham chiếu đến  
10 hình ảnh jpeg)



**1a.** Máy khách HTTP bắt đầu kết nối TCP với máy chủ HTTP (tiến trình) tại `www.someSchool.edu` trên cổng 80

thời gian

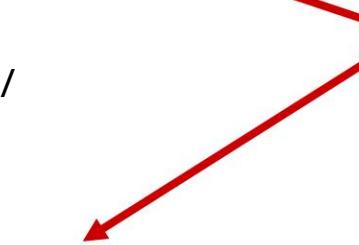


**1b.** Máy chủ HTTP tại máy chủ `www.someSchool.edu` đang chờ kết nối TCP tại cổng 80 "chấp nhận" kết nối, thông báo cho máy khách

**2.** Máy khách HTTP gửi HTTP thông báo yêu cầu (chứa URL) vào ổ cắm kết nối TCP. Thông báo cho biết rằng khách hàng muốn đối tượng `some Department/home.index`



**3.** Máy chủ HTTP nhận thông báo yêu cầu, tạo thông báo phản hồi chứa đối tượng được yêu cầu và gửi thông báo vào ổ cắm của nó



# HTTP không liên tục: ví dụ (tiếp)

Người dùng nhập URL: `www.someSchool.edu/someDepartment/home.index` (chứa văn bản, tham chiếu đến 10 hình ảnh jpeg)



5. Máy khách HTTP nhận được thông báo phản hồi chứa tệp html, hiển thị html. Phân tích tệp html, tìm thấy 10 đối tượng jpeg được tham chiếu

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



4. Máy chủ HTTP đóng TCP

sự liên quan.

thời gian

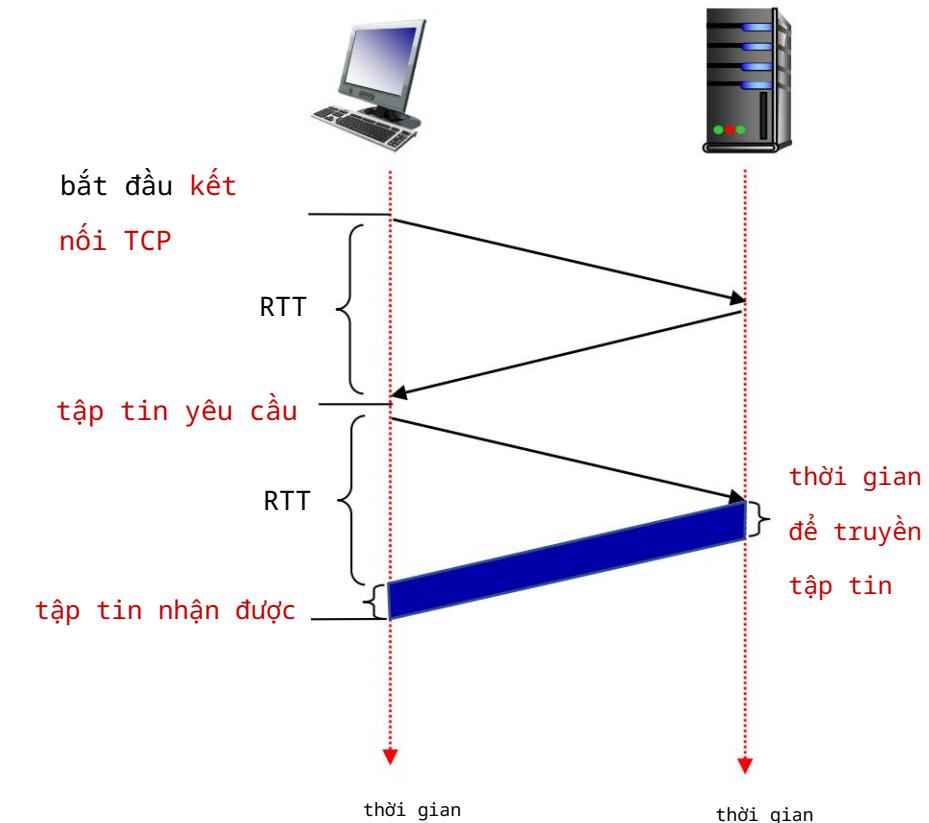
# HTTP không liên tục: thời gian phản hồi

RTT (định nghĩa): thời gian để một gói nhỏ di chuyển từ máy khách đến máy chủ và ngược lại

Thời gian phản hồi HTTP (mỗi đối tượng):

một RTT để bắt đầu kết nối TCP

một RTT cho yêu cầu HTTP và vài byte đầu tiên của phản hồi HTTP để trả về thời gian truyền đối tượng/tập tin



Thời gian phản hồi HTTP không liên tục (mỗi đối tượng) = 2RTT + thời gian truyền tệp

# HTTP liên tục (HTTP 1.1)

Các sự cố HTTP không liên tục: 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 liên tục (HTTP 1.1): máy

chủ để mở kết nối sau khi gửi phản hồi

Các tin nhắn HTTP tiếp theo giữa cùng một  
máy khách/máy chủ được gửi qua kết nối đã  
mở máy khách gửi yêu cầu ngay khi gấp  
đối tượng được tham chiếu

chỉ cần một RTT cho tất cả các  
đối tượng được tham chiếu (giảm  
một nửa thời gian phản hồi)

# Thông báo yêu cầu HTTP

hai loại thông báo HTTP: yêu cầu, phản hồi Thông

báo yêu cầu HTTP: • ASCII (định dạng con người có thể đọc được)

dòng yêu cầu (GET, POST,  
HEAD lệnh)

xuống dòng, xuống dòng  
ở đầu dòng cho biết kết  
thúc dòng tiêu đề

dòng

tiêu đề

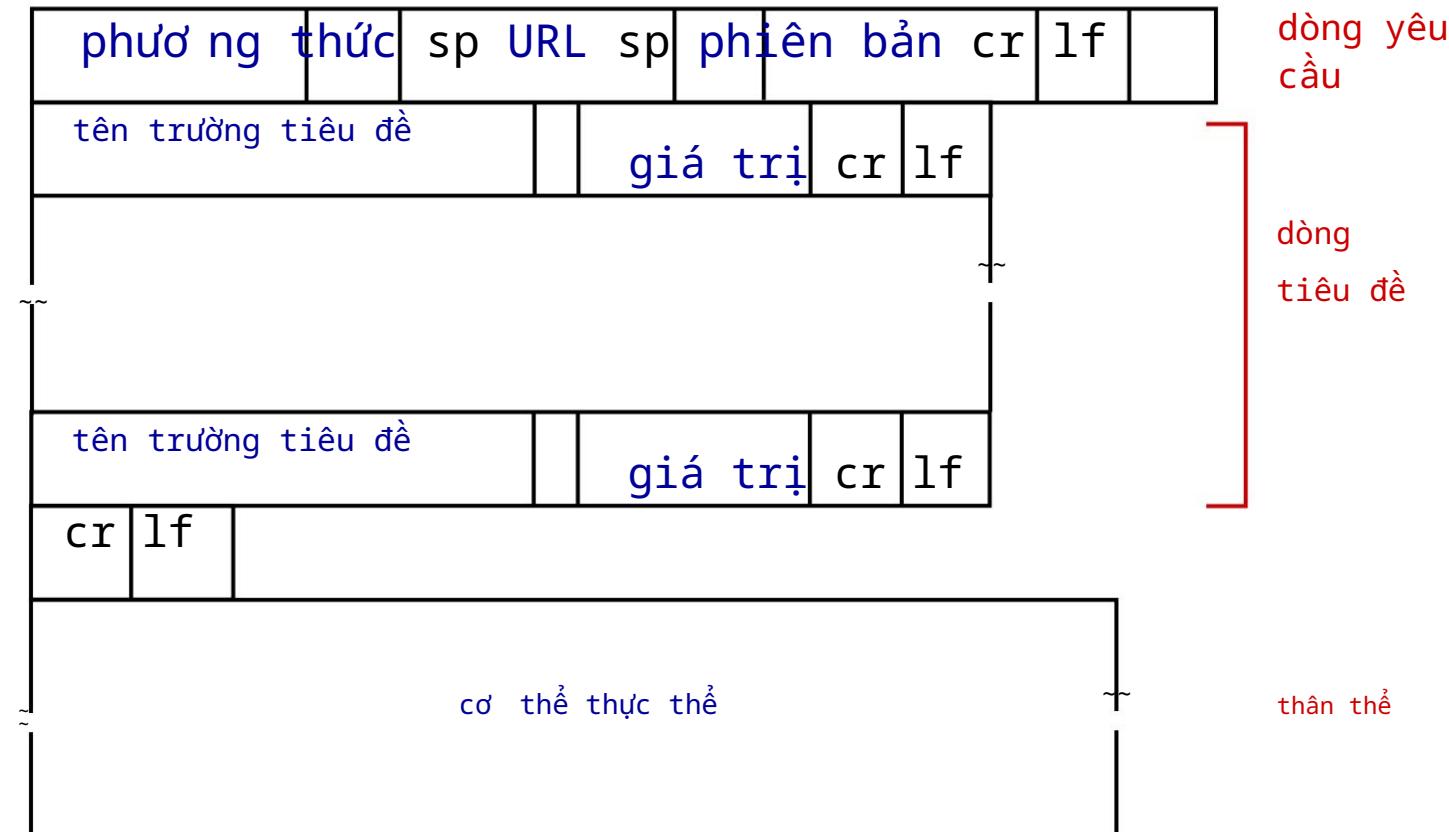
NHẬN /index.html HTTP/1.1\r\n Máy chủ:  
www-net.cs.umass.edu\r\n Tác nhân người  
dùng: Firefox/3.6.10\r\n Chấp nhận: text/  
html,application/xhtml+xml+xml\r\n Ngôn ngữ chấp nhận: en-  
us,en;q=0.5\r\n Chấp nhận mã hóa: gzip,deflate\r\n Chấp  
nhận bộ ký tự: ISO-8859-1,utf-8;q=0.7 \r\n Keep-Alive:  
115\r\n Kết nối: keep-alive\r\n \r\n

ký tự xuống dòng ký tự  
xuống dòng

\* Xem các bài tập tương tác trực tuyến để biết thêm

ví dụ: [http://gaia.cs.umass.edu/kurose\\_ross/interactive/](http://gaia.cs.umass.edu/kurose_ross/interactive/)

# Thông báo yêu cầu HTTP: định dạng chung



# Thông báo yêu cầu HTTP khác

## Phương thức POST:

trang web thường bao gồm đầu vào **biểu mẫu** đầu vào của người dùng được gửi từ máy khách đến máy chủ **trong phần thân thực thể** của HTTP

POST thông báo yêu cầu

Phương thức GET (để gửi dữ liệu đến máy chủ): đưa dữ liệu người dùng vào **trường URL** của thông báo yêu cầu HTTP GET (theo sau dấu '?'):

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

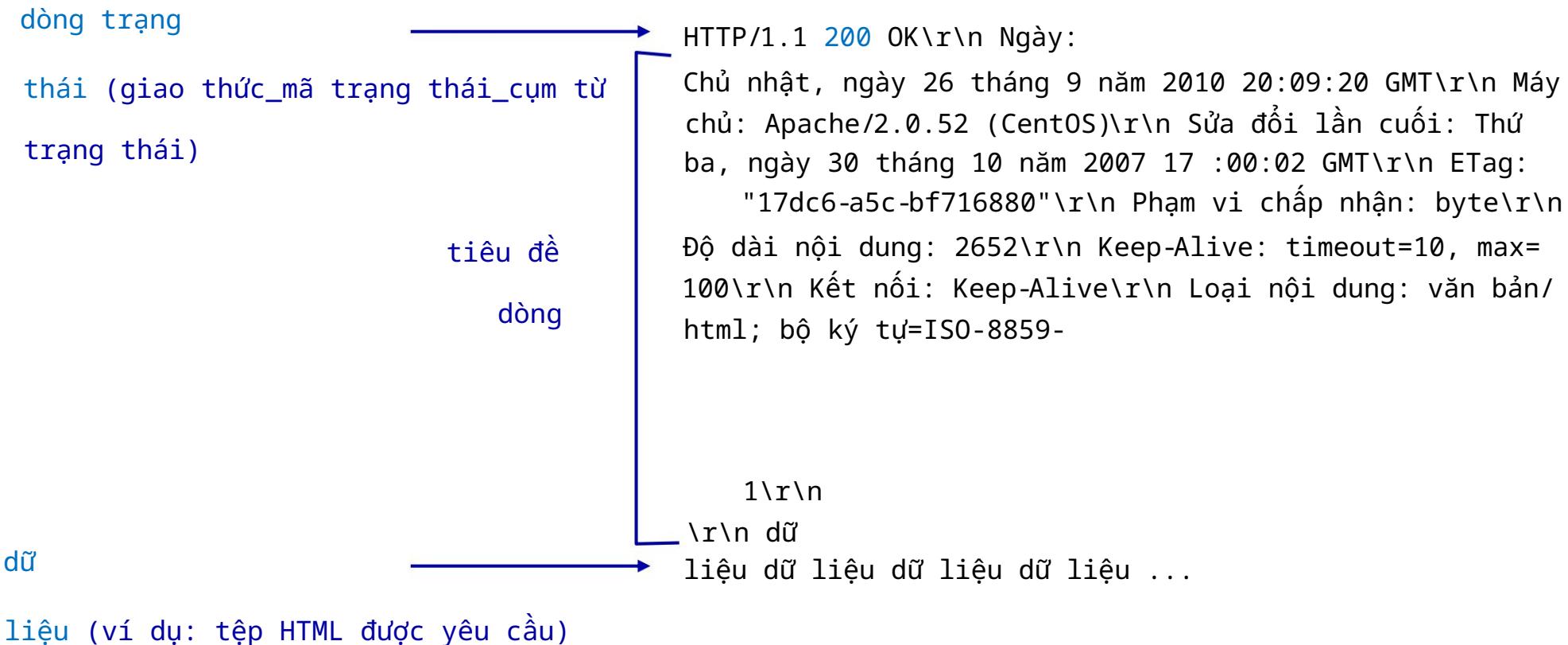
## Phương thức HEAD:

yêu cầu (chỉ) **các tiêu đề** mà sẽ được trả lại nếu URL được chỉ định được yêu cầu bằng phương thức HTTP GET.

## Phương thức PUT:

tải tệp (đối tượng) mới lên máy chủ thay thế hoàn toàn tệp tồn tại tại URL đã chỉ định bằng nội dung **trong phần thân thực thể** của thông báo yêu cầu POST HTTP

# Thông báo phản hồi HTTP



\* Xem các bài tập tương tác trực tuyến để biết thêm ví dụ: [http://gaia.cs.umass.edu/kurose\\_ross/interactive/](http://gaia.cs.umass.edu/kurose_ross/interactive/)

# Mã trạng thái phản hồi HTTP

mã trạng thái xuất hiện ở dòng đầu tiên trong thông báo phản hồi từ máy chủ đến máy khách. một số mẫu mã:

## 200 được

- yêu cầu đã thành công, đối tượng được yêu cầu sau trong thông báo này

## 301 Đã di chuyển vĩnh viễn • đối tượng

được yêu cầu đã di chuyển, vị trí mới được chỉ định sau trong thông báo này (trong  
Vị trí: hiện trường)

## 400 yêu cầu sai

- thông điệp yêu cầu không được máy chủ hiểu

## 404 không tìm thấy

- không tìm thấy tài liệu được yêu cầu trên máy chủ này

## Phiên bản HTTP 505 không được hỗ trợ

# Tự mình dùng thử HTTP (phía máy khách)

1. Telnet tới máy chủ Web yêu thích của bạn: mở [kết](#)

`telnet gaia.cs.umass.edu 80`

nối TCP tới cổng 80 (cổng máy chủ HTTP mặc định) tại gaia.cs.umass.edu.

bất kỳ thứ gì được nhập vào sẽ được gửi đến cổng 80 tại gaia.cs.umass.edu

2. nhập yêu cầu GET HTTP:

[NHẬN /kurose\\_ross/interactive/index.php](#) Máy chủ HTTP/1.1: gaia.cs.umass.edu

bằng cách nhập nội dung này (nhấn xuống dòng hai lần), bạn gửi [yêu cầu NHẬN](#) tối thiểu (nhưng đầy đủ) này tới máy chủ HTTP

3. xem thông báo phản hồi được gửi bởi máy chủ HTTP!

(hoặc sử dụng [Wireshark](#) để xem yêu cầu/phản hồi HTTP đã chụp)

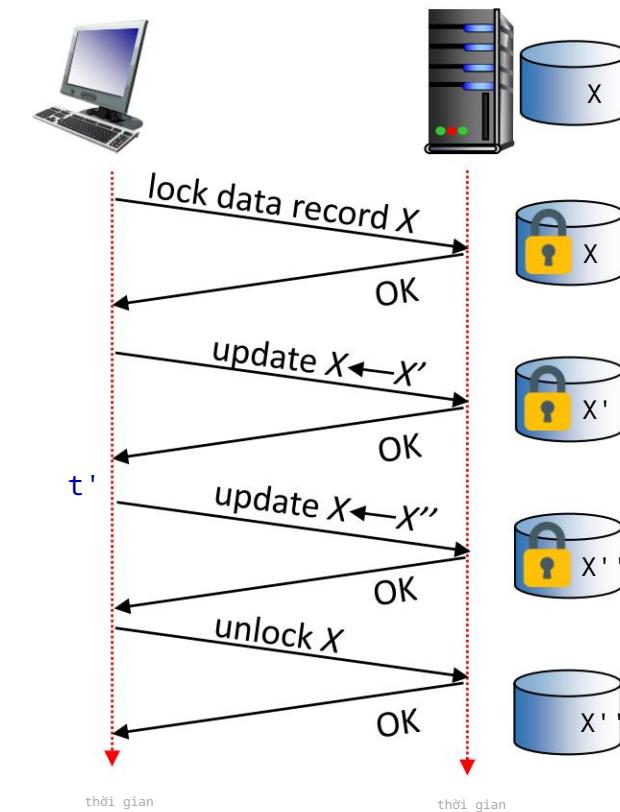
# Duy trì trạng thái người dùng/máy chủ: cookie

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

không có khái niệm trao đổi nhiều bước của  
Thông báo HTTP để hoàn tất một “giao  
dịch” Web

- **không cần** máy khách/máy chủ theo  
dõi “trạng thái” của trao đổi nhiều  
bước • tất cả các yêu cầu HTTP đều **độc  
lập** với nhau
- **không cần** máy khách/máy chủ “khôi phục”  
từ một giao dịch hoàn thành một phần  
nhưng không bao giờ hoàn thành

một giao thức có trạng thái: máy khách thực hiện  
hai thay đổi đối với X hoặc không thay đổi gì cả



**H:** điều gì xảy ra nếu kết nối mạng hoặc ứng dụng  
khách gặp sự cố tại  $t'$  ?

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

Các trang web và trình duyệt của khách hàng sử dụng **cookie** để duy trì một số trạng thái giữa các giao dịch bốn thành phần:

- 1) dòng tiêu đề cookie của thông báo phản hồi HTTP
- 2) dòng tiêu đề cookie trong HTTP tiếp theo

thông báo yêu cầu 3)

,

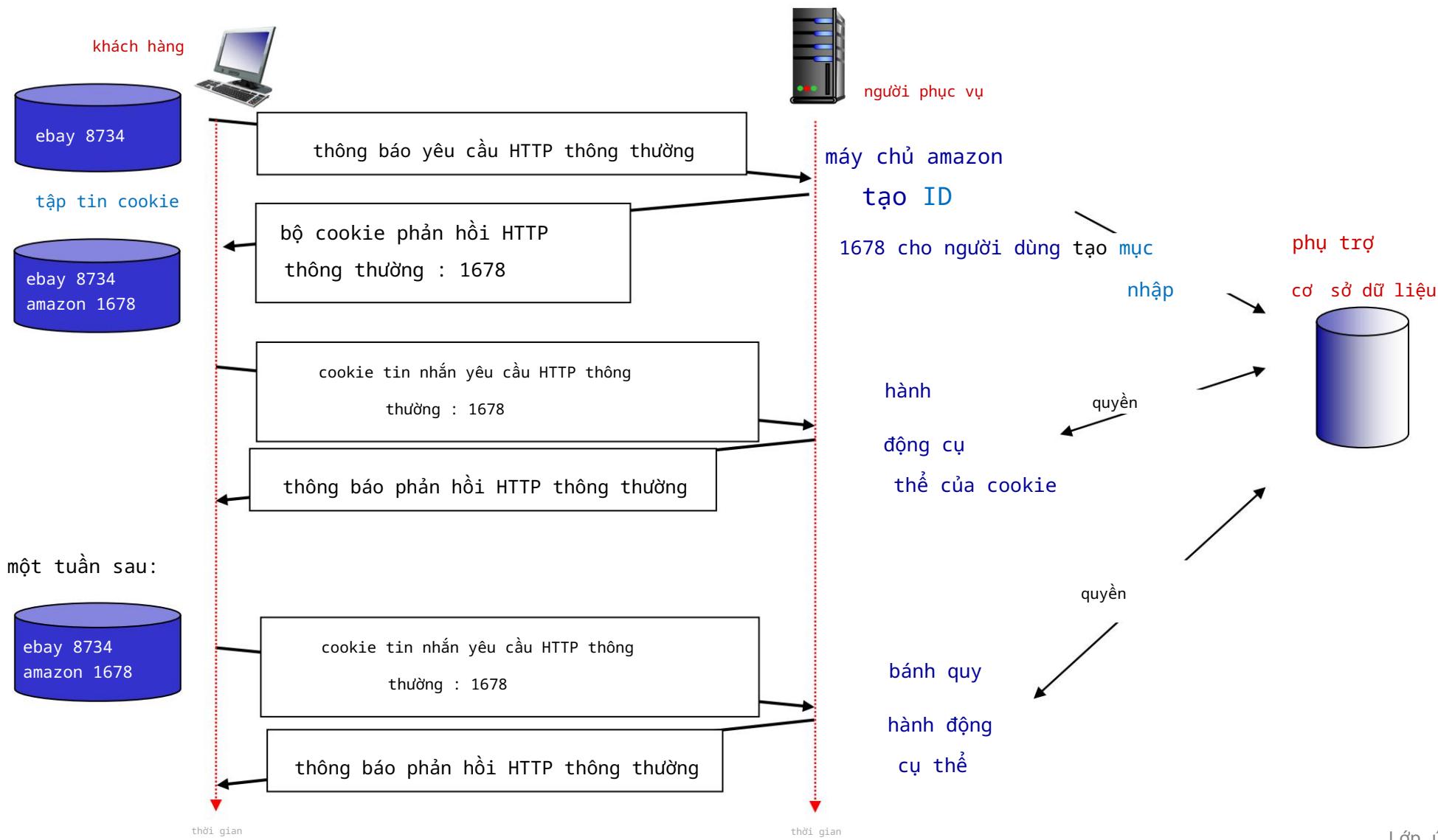
tệp cookie được lưu trên máy chủ của người dùng, được quản lý bởi trình duyệt của người dùng, dữ liệu phụ trợ tại trang Web

Ví dụ:

Susan sử dụng trình duyệt trên máy tính xách tay, truy cập trang web thư ng mại điện tử cụ thể lần đầu tiên khi yêu cầu HTTP ban đầu đến trang web, trang web sẽ tạo:

- ID duy nhất (còn gọi là "cookie")
- mục nhập trong cơ sở dữ liệu phụ trợ cho ID
- các yêu cầu HTTP tiếp theo từ Susan tới 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



# Cookie HTTP: nhận xét

Những cookie nào có thể được sử dụng để:

Ủy quyền

Giả hàng

Khuyến nghị

Trạng thái phiên người dùng (Web e-mail)

Thách thức: Làm thế nào để giữ trạng thái?

Điểm cuối giao thức: duy trì trạng thái ở  
người gửi/người nhận qua nhiều giao dịch

Cookie: Trạng thái mang thông điệp HTTP

qua một bên

Cookie và quyền riêng

Tư: cookie cho phép các trang tìm hiểu nhiều điều về bạn trên trang của họ.

bên thứ ba liên tục

Cookie (cookie theo dõi) cho phép theo dõi danh tính chung (giá trị cookie) trên nhiều trang web

# Bộ đệm web (máy chủ proxy)

**Mục tiêu:** đáp ứng yêu cầu của khách hàng mà không liên quan đến máy chủ gốc

người dùng **định cấu hình** trình duyệt

để **trỏ đến** bộ **đệm Web**

trình duyệt gửi tất cả các yêu

cầu HTTP tới bộ nhớ cache • nếu

đối tượng **trong** bộ nhớ cache: bộ nhớ

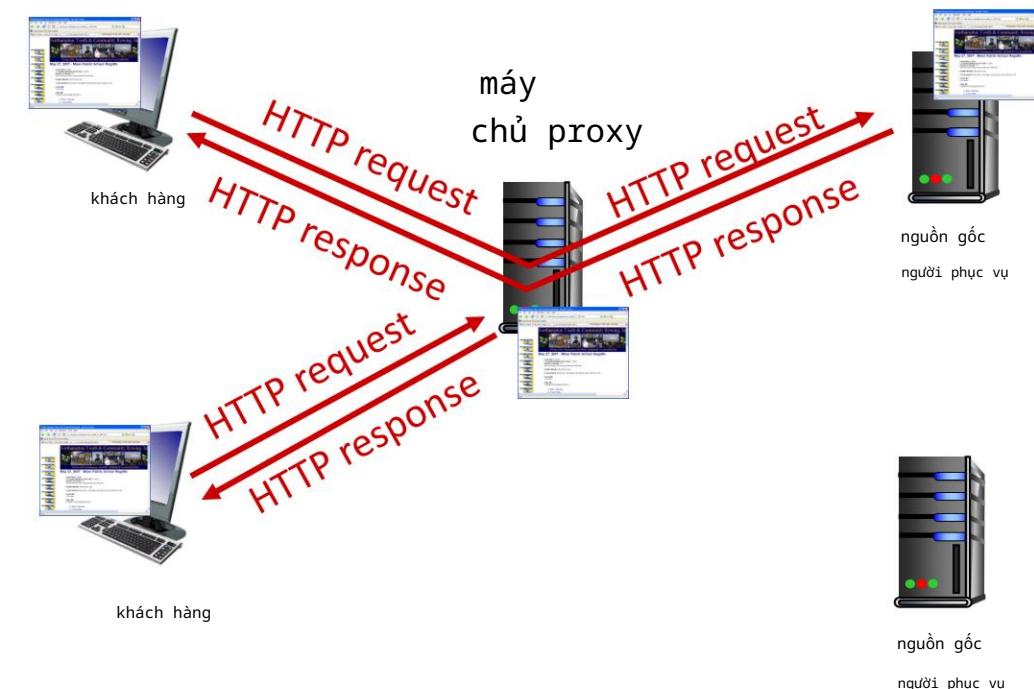
cache trả đối tượng về máy khách •

nếu **không** bộ nhớ cache yêu cầu đối tượng

từ máy chủ gốc, **lưu trữ** đối tượng nhận

được trong bộ nhớ cache, sau đó **trả**

đối tượng về máy khách



# Bộ đệm web (máy chủ proxy)

Web cache đóng vai trò vừa là máy khách vừa là máy chủ • máy chủ dành cho máy khách yêu cầu ban đầu • máy khách đến máy chủ gốc thông thường, bộ nhớ cache được cài đặt bởi ISP (trường đại học, công ty, ISP khu dân cư)

Tại sao phải lưu vào bộ

nhớ đệm Web? giảm thời gian phản hồi yêu cầu của máy khách • bộ đệm gần máy khách hơn

giảm lưu lượng trên liên kết truy cập của tổ chức

Internet dày đặc cache

- cho phép các nhà cung cấp nội dung “nghèo nàn” phân phối nội dung hiệu quả hơn

# Ví dụ về bộ nhớ đệm

Kịch bản:

Tốc độ liên kết truy cập: 1,54 Mbps RTT

từ bộ định tuyến của tổ chức đến máy chủ: 2 giây Kích thước đống

tương ứng web: 100K bit Tốc độ yêu cầu trung bình từ trình duyệt đến nguồn gốc

máy chủ: 15 yêu cầu/giây tốc độ dữ

liệu trung bình cho trình duyệt: 1,50 Mbps

Hiệu suất: Mức

sử dụng mạng LAN: 0,0015

sử dụng liên kết truy cập = .97

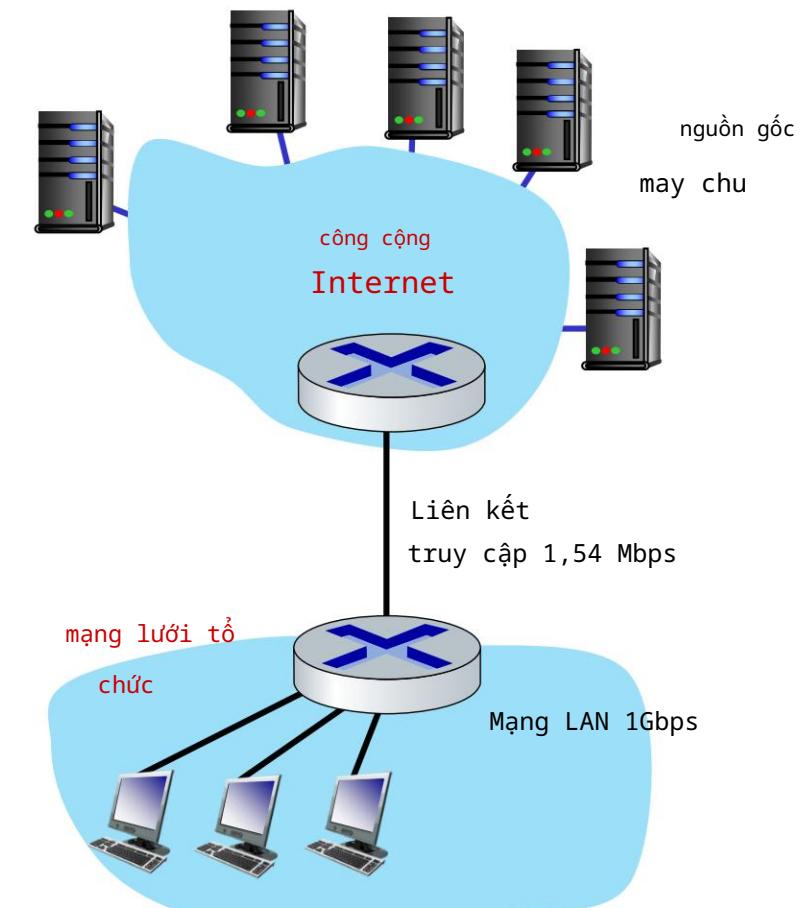
vấn đề: sự chậm  
trễ lớn ở mức  
sử dụng cao!

độ trễ đầu cuối = độ trễ Internet +

độ trễ liên kết truy cập +

Độ trễ mạng

LAN = 2 giây + phút + usec



# Caching ví dụ: mua link truy cập nhanh hơn

Kích bản: 154 Mbps Tốc độ liên kết truy cập: 1,54 Mbps RTT từ bộ định tuyến của tổ chức đến máy chủ: 2 giây Kích thước đối tượng web: 100K bit Tốc độ yêu cầu trung bình từ trình duyệt đến máy chủ gốc: 15 yêu cầu/giây Tốc độ dữ liệu trung bình đến trình duyệt: 1,50 Mbps

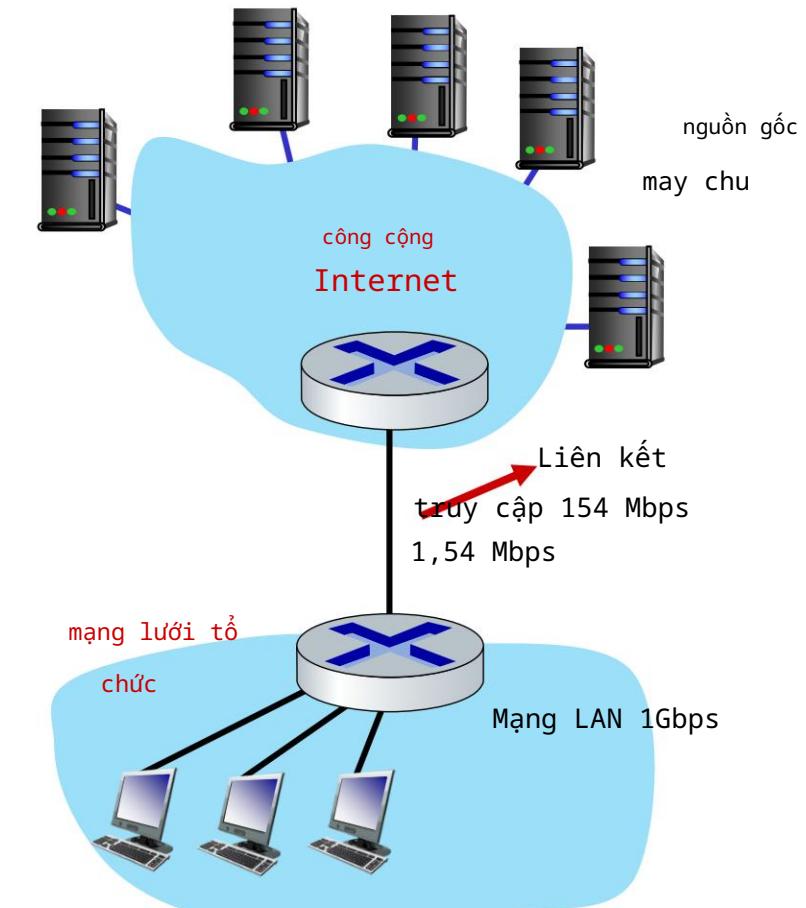
## Hiệu suất:

Mức sử dụng mạng LAN: 0,0015

sử dụng liên kết truy cập = .97  $\longrightarrow$  .0097

$$\begin{aligned} \text{độ trễ đầu cuối} &= \text{độ trễ Internet} + \text{độ trễ} \\ &\quad \text{liên kết truy cập} + \text{độ trễ LAN} \\ &= 2 \text{ giây} + \text{phút} + \text{lần sử dụng} \end{aligned}$$

Chi phí: liên kết truy cập nhanh hơn (đắt!)  $\longrightarrow$  mili giây



# Ví dụ về bộ đệm: cài đặt bộ đệm web

Kịch bản:

Tốc độ liên kết truy cập: 1,54 Mbps

RTT từ bộ định tuyến của tổ chức đến máy chủ: 2 giây Kích

thước đối tượng web: 100K bit Tốc độ yêu cầu trung bình từ

trình duyệt đến máy chủ gốc: Tốc độ dữ liệu trung bình 15/ giây đến trình duyệt: 1,50 Mbps

Hiệu suất: Sử

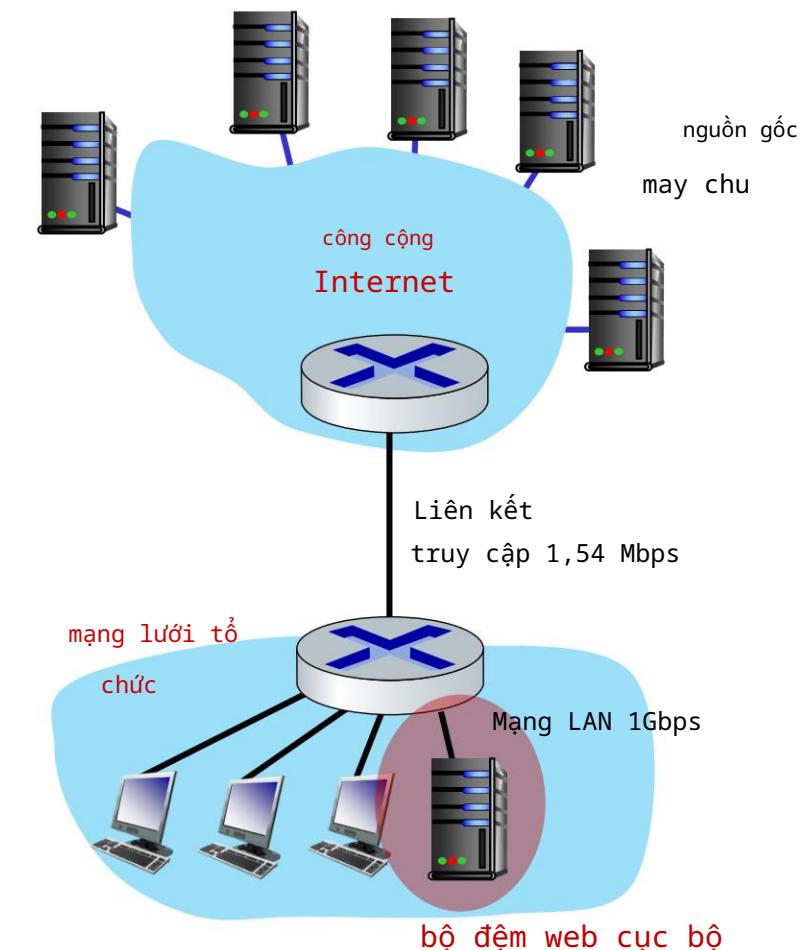
dụng mạng LAN: .?

sử dụng liên kết truy cập = ?

độ trễ đầu cuối trung bình = ?

Chi phí: bộ đệm web (rẻ!)

Làm thế nào để tính toán  
sử dụng liên kết, độ trễ?



# Ví dụ về bộ đệm: cài đặt bộ đệm web

Tính toán mức sử dụng liên kết truy cập, độ trễ  
kết thúc với bộ đệm: giả sử **tỷ lệ trúng bộ đệm**  
là 0,4: **40%** yêu cầu được thỏa mãn tại bộ đệm;  
**60%** yêu cầu hài lòng tại nguồn gốc

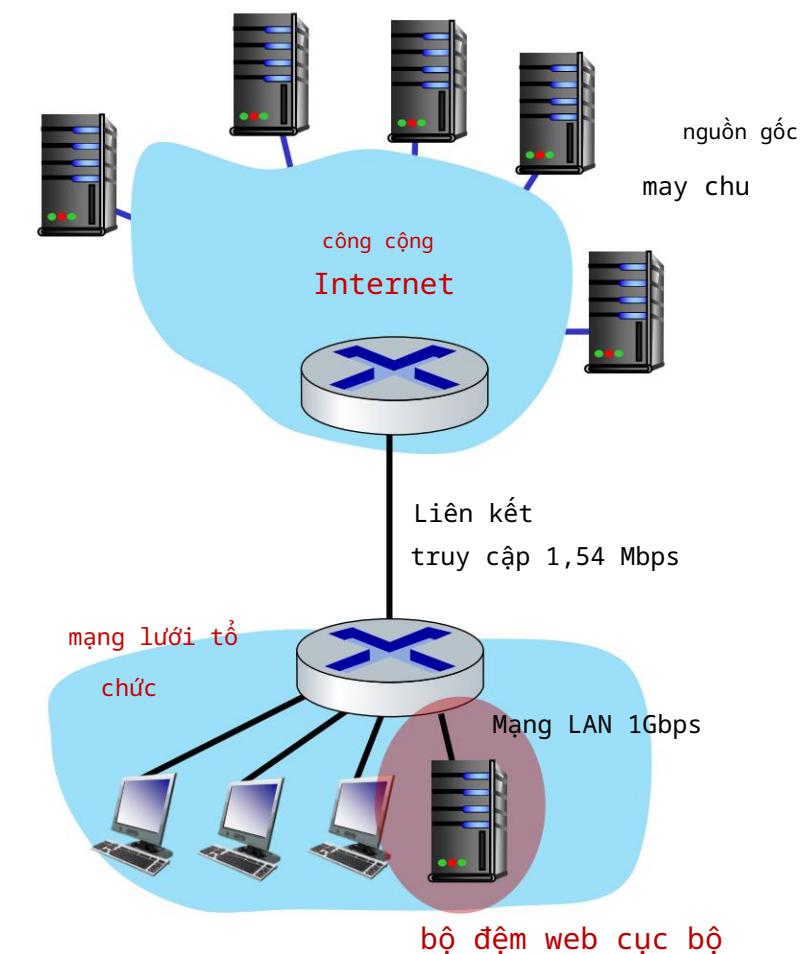
liên kết truy cập: **60%** yêu cầu sử dụng liên kết truy cập  
tốc độ dữ liệu tới trình duyệt qua liên kết truy cập  
 $= 0,6 * 1,50 \text{ Mbps} = 0,9 \text{ Mbps}$

$$\text{sử dụng} = 0,9/1,54 = 0,58$$

độ trễ đầu cuối trung

$$= 0,6 \text{ (bình thường)} * t_{\text{trung}} + t_{\text{không trung}} + t_{\text{bộ đệm}}$$

$$(\sim \text{msec}) \sim + 1024 \text{ giây} + 0,6 (2,01) + 0,4$$



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

# NHẬN có điều kiện

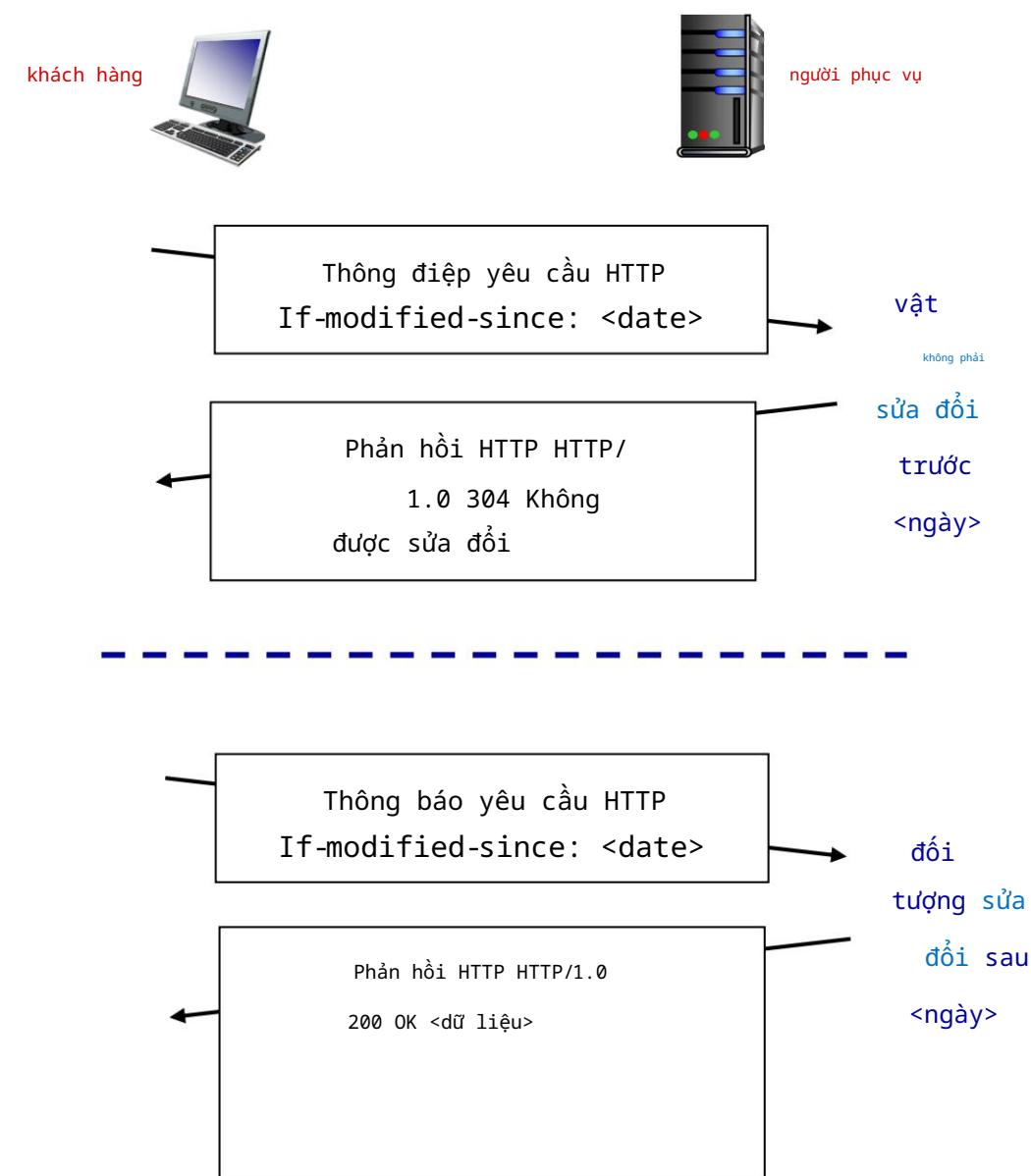
**Mục tiêu:** không gửi đối tượng nếu bộ đệm có phiên bản được lưu trong bộ nhớ cache cập nhật

- không có độ trễ truyền đối tượng
- sử dụng liên kết thấp hơn

**bộ đệm:** chỉ định ngày sao chép được lưu trong bộ đệm trong yêu cầu HTTP

If-modified-since: <date>

**máy chủ:** phản hồi không chứa đối tượng nếu bản sao được lưu trong bộ nhớ cache được cập nhật: HTTP/1.0 304 Không được sửa đổi



# HTTP/2

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

HTTP1.1: đã giới thiệu nhiều GET theo đường ống qua một kết nối TCP

máy chủ phản hồi theo thứ tự (FCFS: lập lịch đến trước phục vụ trước) tới NHẬN yêu cầu

với FCFS, đối tượng nhỏ có thể phải chờ truyền (**chặn đầu dòng (HOL)**) phía sau (các) đối tượng lớn

**khôi phục tổn thất** (truyền lại các phân đoạn TCP bị mất) làm đối tượng bị treo truyền tải

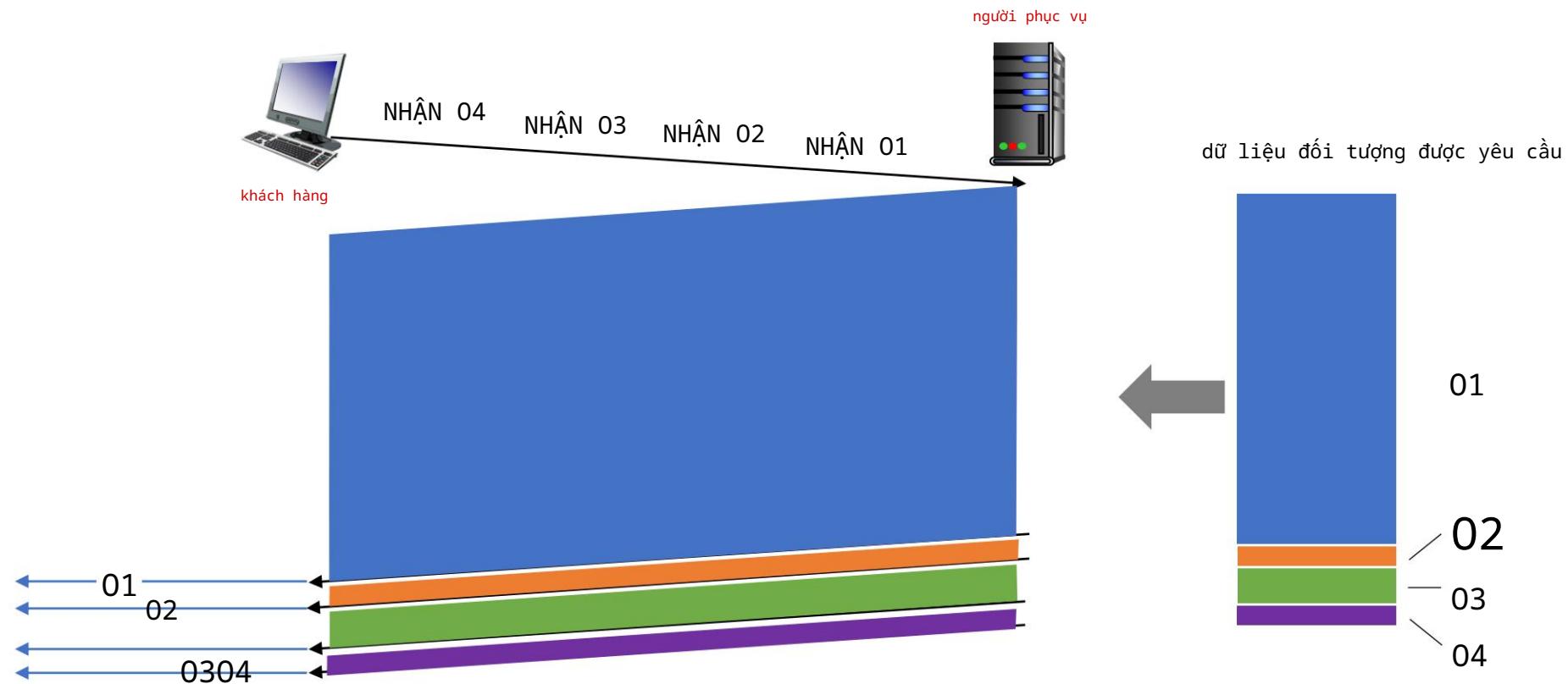
# HTTP/2

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

HTTP/2: [RFC 7540, 2015] tăng **tính linh hoạt** tại máy chủ trong việc gửi các đối tượng  
tới máy khách: các phươ ng thức, mã trạng thái, hầu hết các trường tiêu đề **không**  
**thay đổi so** với HTTP 1.1      **Thứ tự truyền** của các đối tượng được yêu cầu dựa trên mức độ ưu  
tiên của đối tượng do máy khách chỉ định (không nhất thiết phải là FCFS )      **đẩy các đối**  
**tượng không** được yêu cầu đến máy khách      **chia các đối tượng** thành các khung, lên lịch  
các khung để **giảm thiểu chặn HOL**

# HTTP/2: giảm thiểu chặng HOL

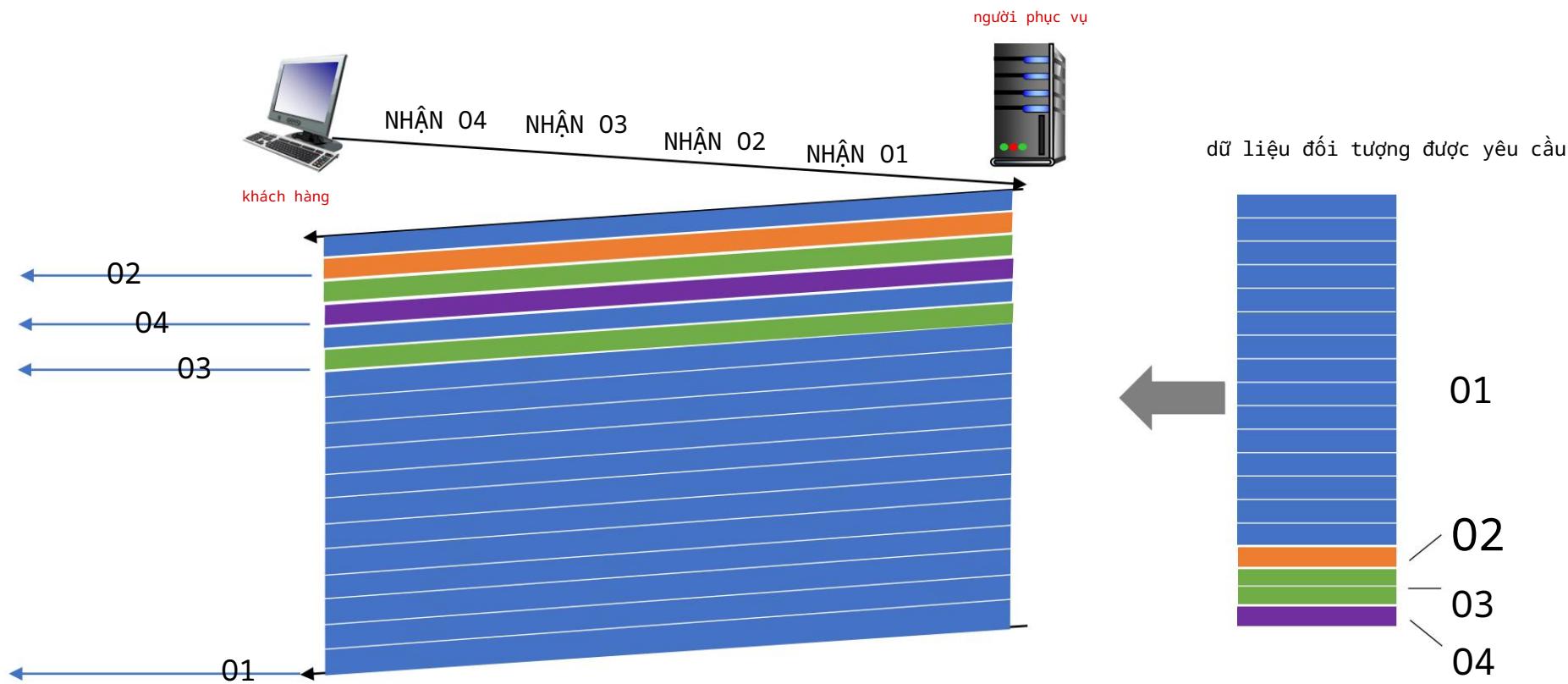
HTTP 1.1: máy khách yêu cầu 1 đối tượng lớn (ví dụ: tệp video và 3 đối tượng nhỏ hơn)



vật được giao theo thứ tự yêu cầu: 02 , 03 , 04 đợi sau 01

# HTTP/2: giảm thiểu chặng HOL

HTTP/2: các đối tượng được chia thành các khung; truyền khung xen kẽ



02 , 03 , 04 giao nhanh, 01 hời chậm

# HTTP/2 đến HTTP/3

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

HTTP/2 trên một kết nối TCP có nghĩa là: khôi phục từ

mất gói vẫn làm ngừng tất cả các quá trình truyền đối tượng • như trong HTTP 1.1, các trình duyệt có động cơ mở nhiều kết nối TCP song song để giảm tình trạng ngừng hoạt động, **tăng thông lượng tổng thể**

**không có bảo mật** đối với kết nối TCP cơ bản

**HTTP/3** bổ sung tính **bảo mật, kiểm soát tắc nghẽn và lỗi** trên mỗi đối tượng (nhiều đường ống hơn) **so với UDP** • nhiều hơn về **HTTP/3 trong lớp truyền tải**

# Lớp Ứng dụng: tổng quan

Nguyên tắc ứng dụng  
mạng Web và HTTP

E-mail, SMTP, IMAP

Hệ thống tên miền  
DNS

Ứng dụng P2P

Mạng phân phối nội dung và  
truyền phát video

lập trình ổ cắm với  
UDP và TCP



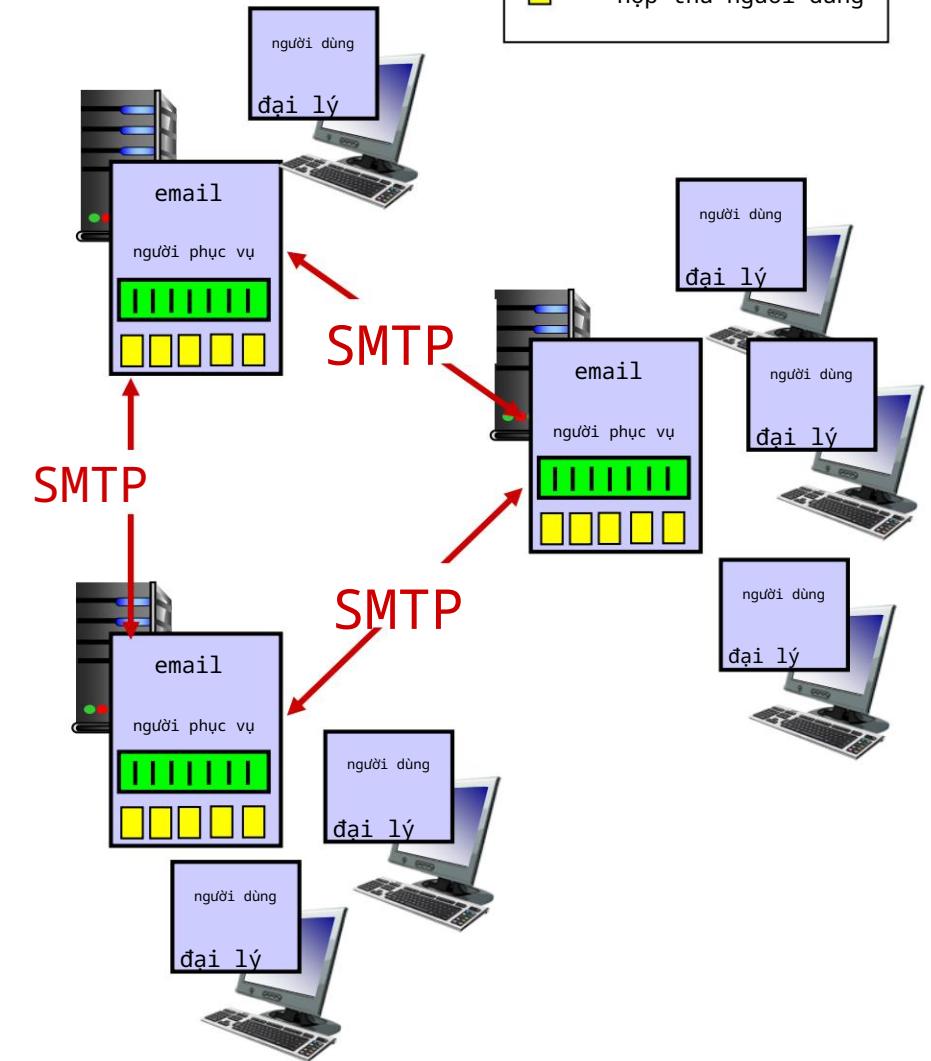
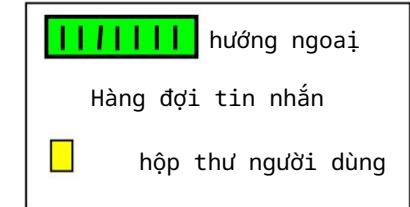
# E-mail

Ba thành phần chính: tác  
nhân người dùng (UA) máy  
chủ thư (MA) giao thức  
truyền thư đơn giản: SMTP

Đại lý người dùng

hay còn gọi là  
“trình đọc thư” soạn, chỉnh sửa, đọc thư ,  
ví dụ: Outlook, ứng dụng thư iPhone thư  
đi, thư đến được lưu trữ trên

người phục vụ



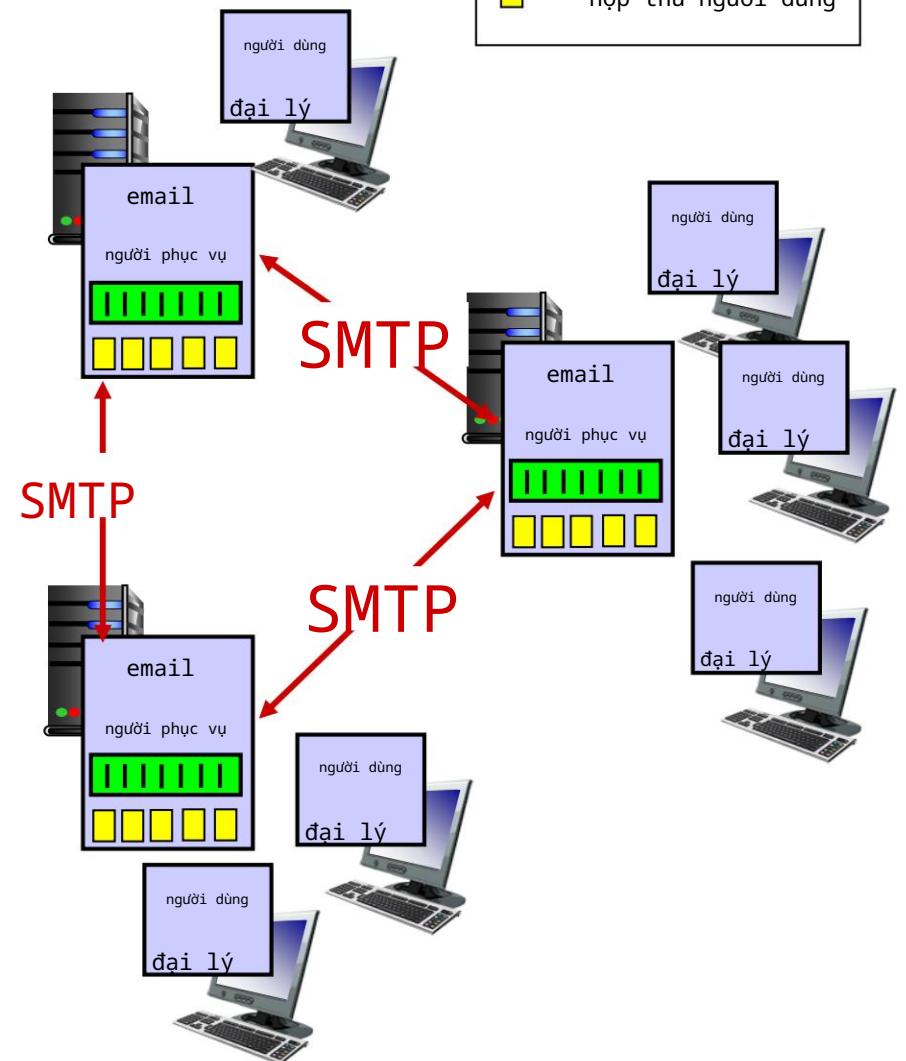
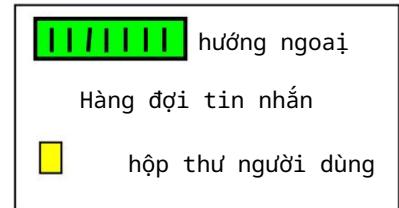
# E-mail: máy chủ thư

máy chủ thư:

hộp thư chứa thư đến cho hàng đợi thư người dùng

của thư đi (sẽ được gửi)

Giao thức SMTP giữa các máy chủ thư để gửi thư điện tử • máy khách: máy chủ gửi thư • “máy chủ”: máy chủ nhận thư



# E-mail: RFC (5321)

sử dụng **TCP** để truyền tin nhắn email **từ máy khách** (máy chủ thư bắt đầu kết nối) **đến máy chủ**, cổng **25**

**chuyển giao trực tiếp:** máy chủ gửi (hoạt động như máy khách) đến máy chủ nhận    **ba giai đoạn chuyển giao** • bắt tay (chào hỏi) • chuyển giao thông điệp  
• kết thúc

**tương tác lệnh/phản hồi** (như HTTP)

- **lệnh:** văn bản ASCII
- **phản hồi:** mã trạng thái và

**thông báo** cụm từ phải ở **dạng ASCII 7 bit**

# Tình huống: Alice gửi e-mail cho Bob

1) Alice sử dụng UA để **soạn** e-mail  
gửi tin nhắn “tới” bob@someschool.edu

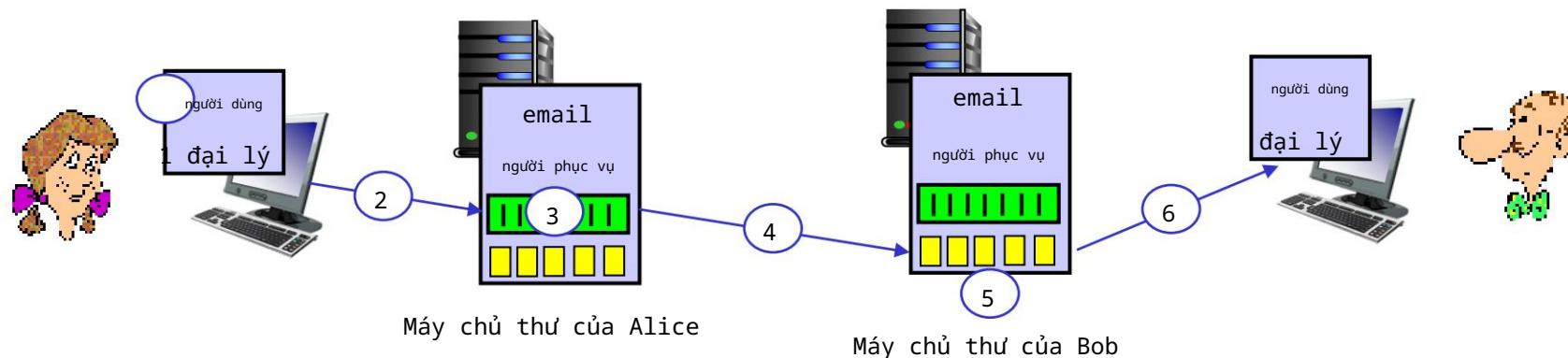
2) UA của Alice **gửi** tin nhắn đến máy chủ  
thư của cô ấy; tin nhắn được đặt trong  
hàng đợi tin nhắn

3) phía máy khách của SMTP **mở** TCP  
kết nối với máy chủ thư của Bob

4) Máy khách SMTP **gửi** tin nhắn của  
Alice qua kết nối TCP

5) Máy chủ thư của Bob **đặt**  
thư vào hộp thư của Bob

6) Bob gọi người dùng của mình  
đại lý để **đọc** tin nhắn



# Tương tác SMTP mẫu

S: 220 hamburger.edu C: HELO

crepes.fr S: 250 Xin chào

crepes.fr, rất vui được gặp bạn C: MAIL TỪ: <alice@crepes.fr> S: 250

alice@crepes.fr... Người gửi ok C : RCPT ĐẾN: <bob@hamburger.edu> S: 250

bob@hamburger.edu ... Người nhận ok C: DATA

S:354 Nhập thư, kết thúc bằng dấu "." trên một dòng của chính nó C: Do you like ketchup?

C: Còn dưa muối thì sao?

C: .

S: 250 Tin nhắn được chấp nhận gửi C: QUIT S: 221

hamburger.edu đóng kết nối

## Hãy thử tương tác SMTP cho chính bạn:

telnet <tên máy chủ> 25

xem 220 trả lời từ máy chủ

nhập HELO, MAIL FROM:, RCPT TO:, DATA, QUIT Các lệnh trên cho phép bạn gửi email mà không cần sử dụng ứng dụng e-mail (trình đọc)

Lưu ý: điều này sẽ chỉ hoạt động nếu <tên máy chủ> cho phép kết nối telnet đến cổng 25 (điều này ngày càng hiếm do lo ngại về bảo mật)

# SMTP: đóng quan sát

## so sánh với HTTP:

HTTP: kéo

SMTP: đẩy

cả hai đều có tương tác lệnh/phản hồi ASCII , mã trạng thái

HTTP: mỗi đối tượng được gói gọn trong thông báo phản hồi của chính nó

SMTP: gửi nhiều đối tượng trong thông điệp nhiều phần

SMTP sử dụng liên tục kết nối

SMTP yêu cầu thông báo (tiêu đề & nội dung) ở dạng ASCII 7 bit

Máy chủ SMTP sử dụng CRLF.CRLF để xác định phần cuối của thông báo

# Định dạng tin nhắn thư

SMTP: giao thức trao đổi thư điện tử,  
được định nghĩa trong RFC 531 (như HTTP)

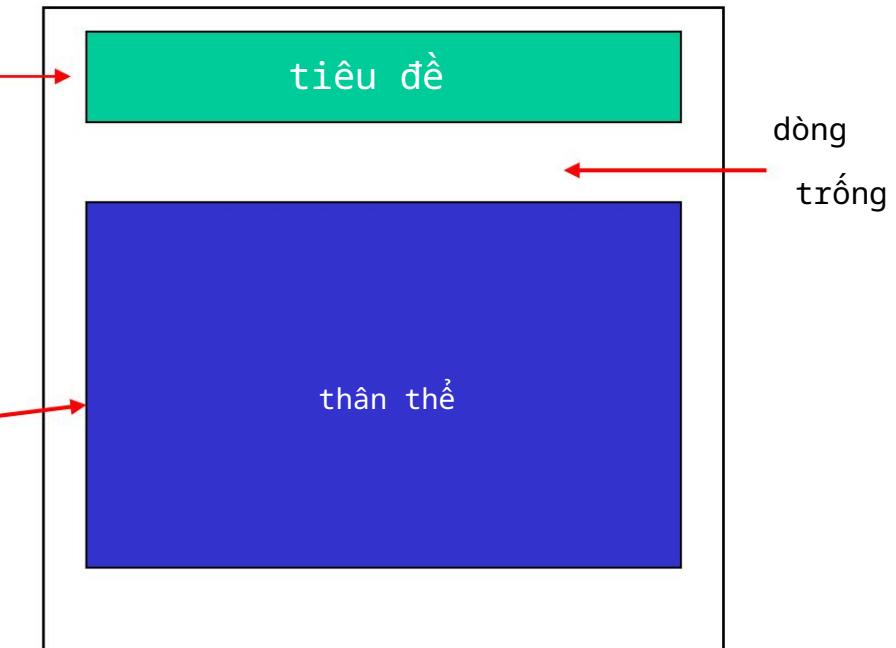
RFC 822 xác định cú pháp cho chính thông điệp  
e-mail (như HTML)

dòng tiêu đề, ví dụ,

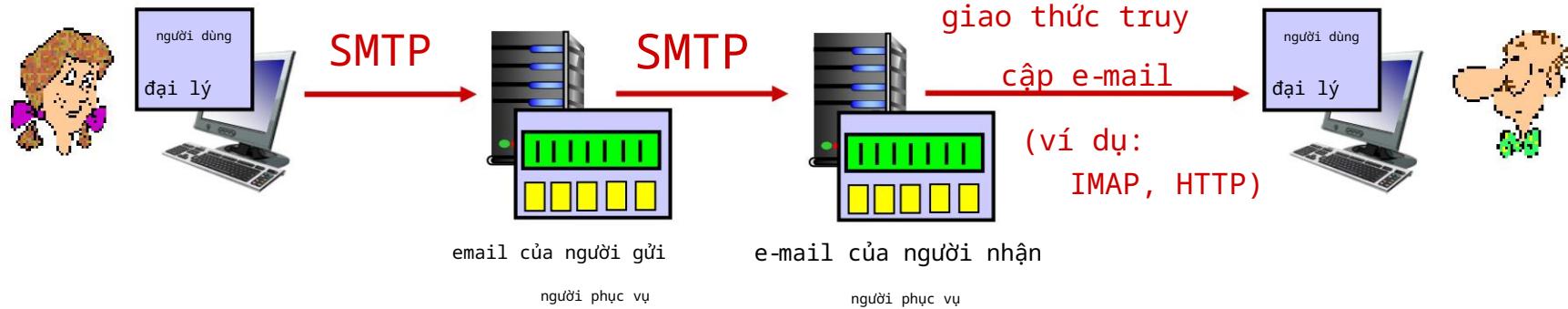
- Đến:
- Từ:
- Chủ đề:

những dòng này, trong phần nội dung của khu vực  
thông báo email khác với các lệnh SMTP MAIL TỪ:, RCPT ĐẾN :!

Nội dung: "thông điệp", chỉ các ký tự ASCII



# Giao thức truy cập thư



**SMTP:** gửi/lưu trữ thư điện tử đến máy chủ của người nhận

**giao thức truy cập thư:** truy xuất từ máy chủ

- **IMAP:** Internet Mail Access Protocol [RFC 3501]: thư được lưu trữ trên máy chủ, IMAP cung cấp **truy xuất, xóa, thư mục** của các tin nhắn được lưu trữ trên máy chủ

**HTTP:** Gmail, Hotmail, Yahoo!Mail, v.v. cung cấp giao diện dựa trên web dựa trên STMP (để gửi), IMAP (hoặc POP3) để truy xuất thư điện tử

# Lớp Ứng dụng: Tổng quan

Nguyên tắc ứng dụng  
mạng Web và HTTP

E-mail, SMTP, IMAP

Hệ thống tên miền (DNS)

Ứng dụng P2P

Mạng phân phối nội dung và  
truyền phát video

lập trình ổ cắm với  
UDP và TCP



# DNS: Hệ thống tên miền

người: nhiều định danh:

- SSN, tên, hộ chiếu #

Máy chủ, bộ định tuyến Internet: • Địa chỉ

IP (32 bit) - được sử dụng để đánh địa chỉ các datagram • "tên" (ví dụ: cs.umass.edu) -

được sử dụng bởi con người

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

Hệ thống tên miền: cơ sở

dữ liệu phân tán được triển khai trong hệ thống phân cấp của nhiều máy chủ định danh giao thức tầng ứng dụng: máy chủ, máy chủ định danh giao tiếp để phân giải tên (bản dịch địa chỉ/tên) • lưu ý: chức năng Internet cốt lõi, được triển khai dưới dạng giao thức tầng ứng dụng • độ phức tạp ở "biên" của mạng

# DNS: dịch vụ, cấu trúc

## dịch vụ DNS

dịch tên máy chủ sang địa chỉ IP

bí danh máy

chủ • chính tắc, tên bí danh

bí danh máy chủ thư

phân bổ tải

- máy chủ Web nhân rộng: **nhiều địa chỉ IP** tương ứng với một Tên

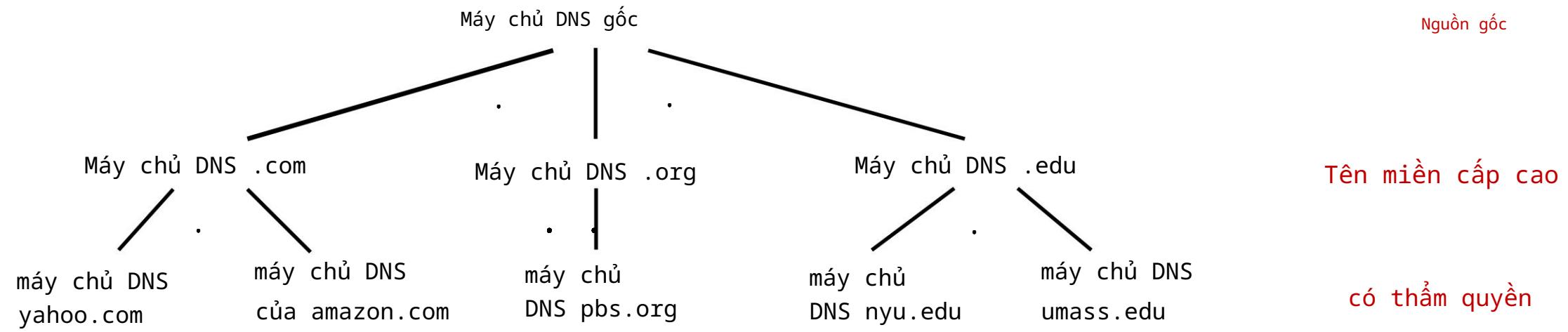
Hỏi: Tại sao không tập trung hóa

DNS? điểm lỗi duy nhất lưu lượng truy cập cơ sở dữ liệu tập trung từ xa  
bảo trì

A: không mở rộng quy

mô! ví dụ : riêng máy chủ Comcast DNS : 600B truy vấn DNS mỗi ngày

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



Khách hàng muốn có địa chỉ IP cho [www.amazon.com](http://www.amazon.com); xấp xỉ 1 :

client truy vấn **root** server để tìm .com DNS server

client truy vấn .com DNS server để lấy [amazon.com](http://amazon.com) DNS server

client truy vấn [amazon.com](http://amazon.com) DNS server để lấy địa chỉ IP cho [www.amazon.com](http://www.amazon.com)

# DNS: máy chủ tên gốc

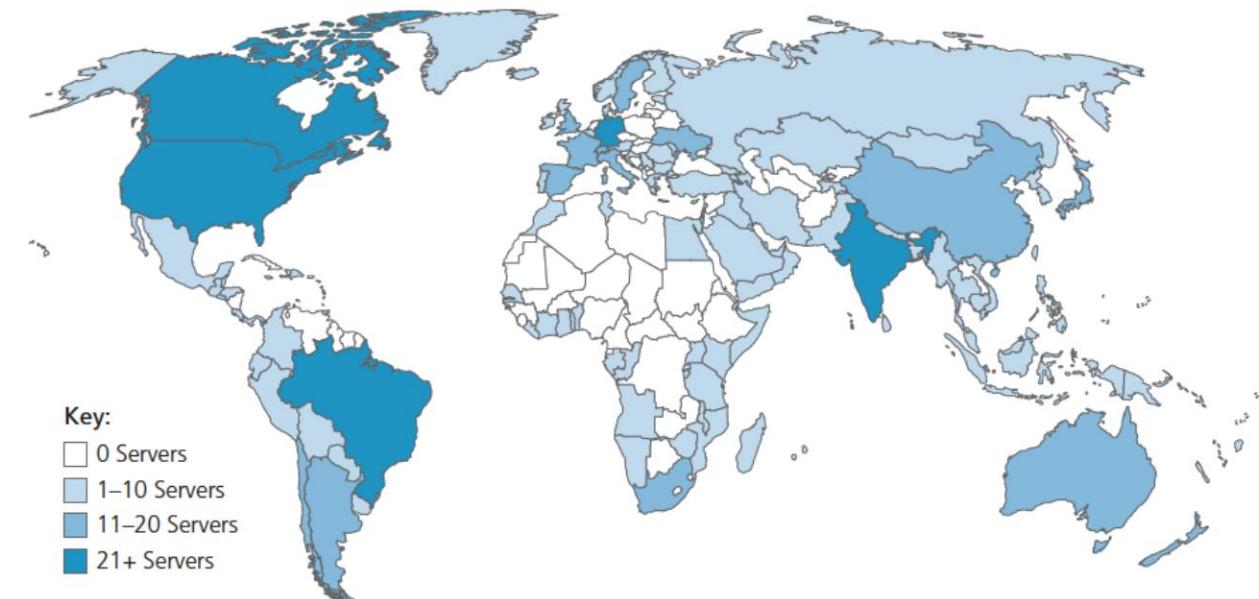
chính thức, [liên hệ cuối cùng](#) của máy chủ tên không thể giải quyết tên

chức năng Internet **cực kỳ** quan trọng

- Internet không thể hoạt động nếu không có nó!
- [DNSSEC](#) - cung cấp bảo mật (xác thực và tính toàn vẹn của thông báo)

[ICANN](#) (Tập đoàn Internet cho Tên và số được gán) quản lý [miền DNS gốc](#)

13 “máy chủ” tên gốc hợp lý trên toàn thế giới, mỗi “máy chủ” được sao chép nhiều lần (~200 máy chủ ở Hoa Kỳ)



## TLD: máy chủ có thẩm quyền

Máy chủ miền cấp cao nhất (TLD): chịu

trách nhiệm về .com, .org, .net, .edu, .aero, .jobs, .museums và tất cả các miền quốc gia cấp cao nhất, ví dụ: .cn, .uk, .fr, .ca, .jp Giải pháp mạng: cơ quan đăng ký có thẩm quyền cho .com, .net TLD Nguyên nhân giáo dục: .edu TLD

Máy chủ DNS có thẩm quyền:

(các) máy chủ DNS riêng của tổ chức, cung cấp tên máy chủ có thẩm quyền tới ánh xạ IP cho các máy chủ được đặt tên của tổ chức có thể được duy trì bởi tổ chức hoặc nhà cung cấp dịch vụ

# Máy chủ tên DNS cục bộ

không hoàn toàn thuộc về hệ thống phân cấp mõi

ISP (ISP khu dân cư, công ty, trường đại học) có một

- còn được gọi là “máy chủ định danh mặc định”

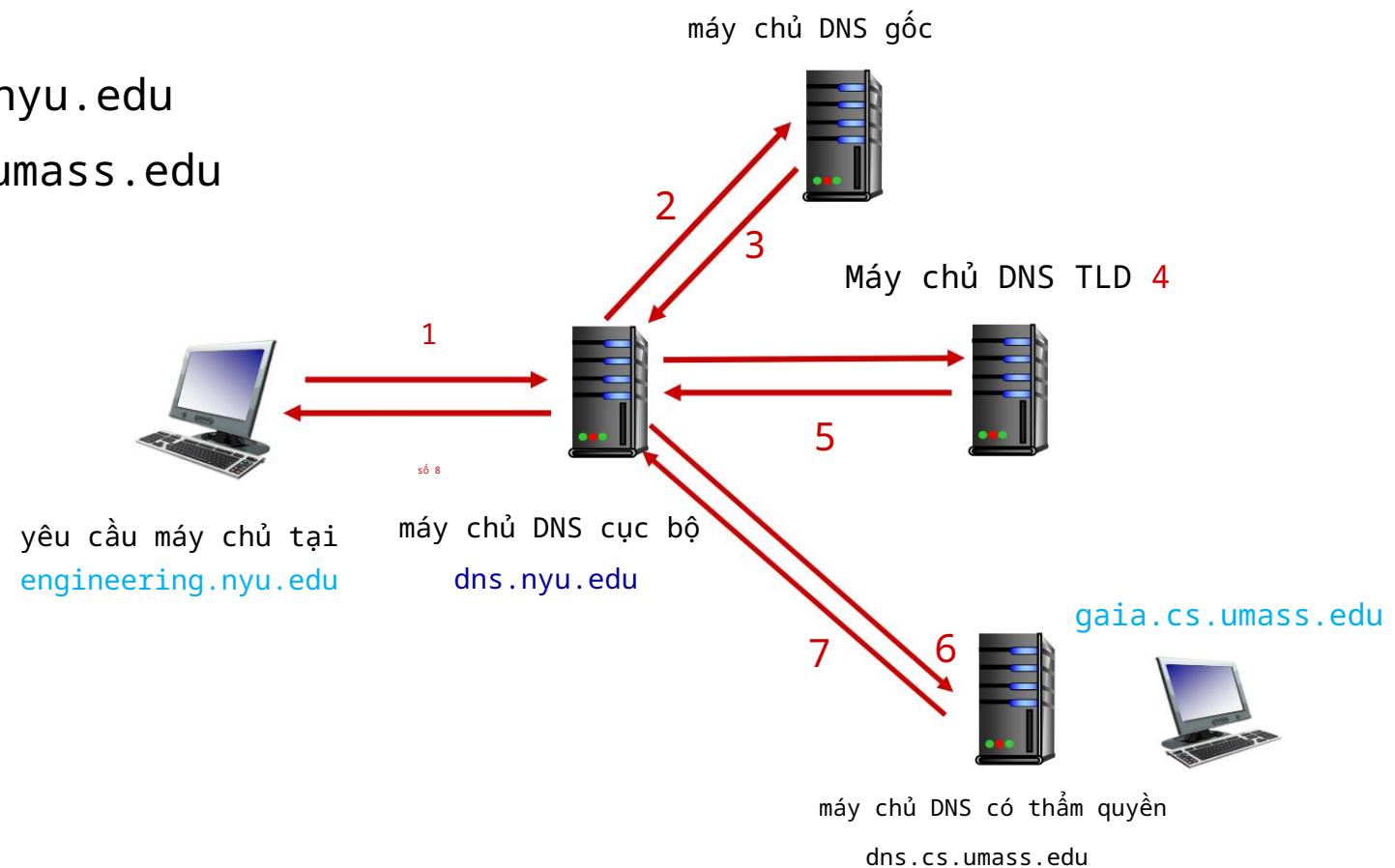
khi máy chủ thực hiện truy vấn DNS, truy vấn được gửi đến DNS cục bộ của nó  
người phục vụ

- có bộ đệm cục bộ của các cặp dịch tên-địa chỉ gần đây (nhưng có thể đã lỗi thời!)
- hoạt động như proxy, chuyển tiếp truy vấn vào hệ thống phân cấp

# Độ phân giải tên DNS: truy vấn lặp lại

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

Truy vấn lặp đi lặp  
lại: Máy chủ đã liên hệ trả  
lời với tên của máy chủ cần  
liên hệ “Tôi không biết  
tên này, nhưng hãy hỏi máy chủ  
này”



# Phân giải tên DNS: truy vấn đệ quy

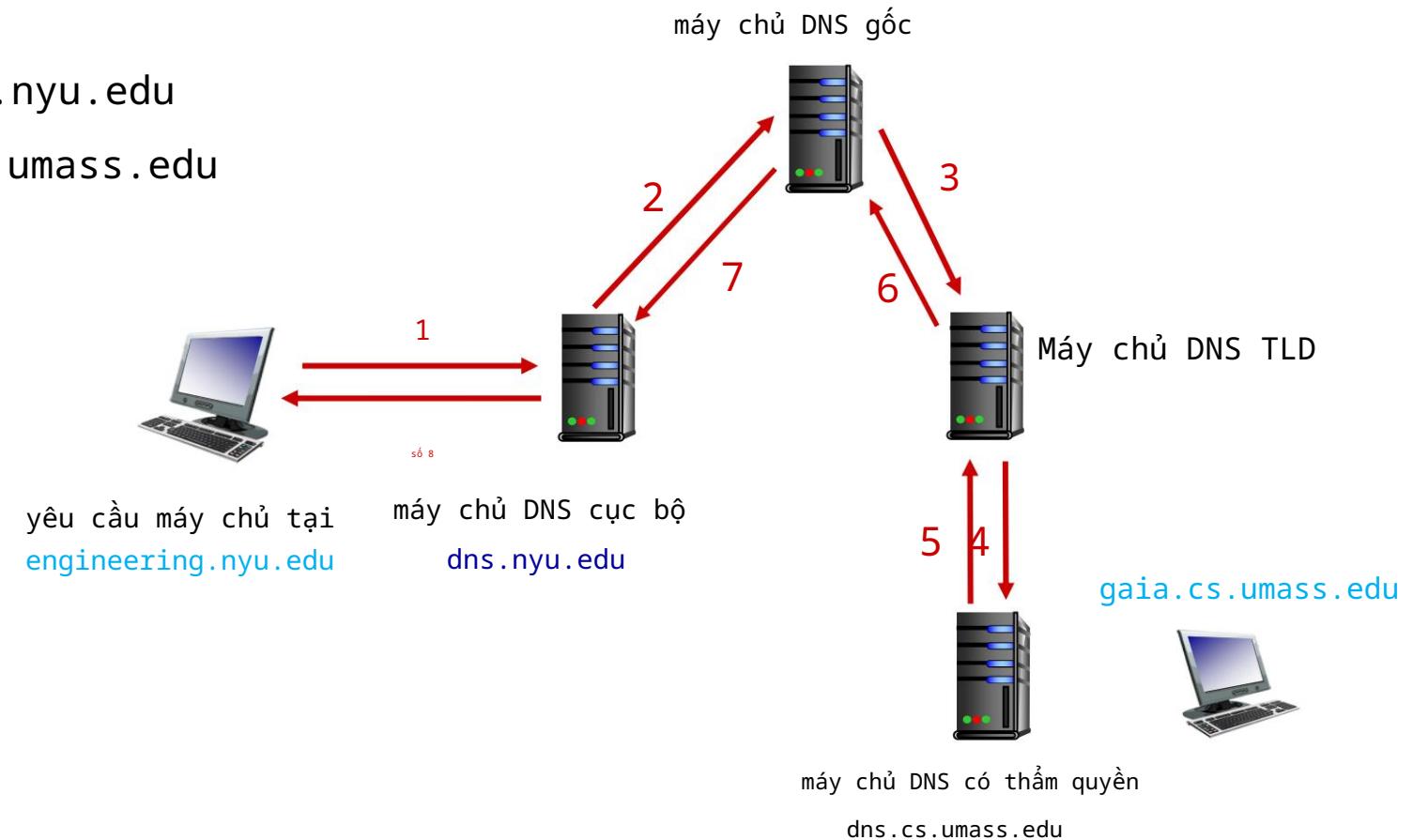
Ví dụ: máy chủ tại engineering.nyu.edu  
muốn có địa chỉ IP cho gaia.cs.umass.edu

Truy vấn đệ quy:

đặt gánh nặng phân giải tên  
lên tên được liên hệ

người phục vụ

gánh nặng ở cấp trên  
của hệ thống phân cấp?



# Bộ nhớ đệm, cập nhật bản ghi DNS

một khi (bất kỳ) máy chủ định danh nào học cách lập bản đồ, nó sẽ lưu bản đồ vào bộ đệm • **hết thời gian chờ** các mục trong bộ đệm (biến mất) sau một thời gian (TTL) • **Máy chủ TLD thường được lưu vào bộ đệm trong máy chủ định danh cục bộ** • do đó, máy chủ định danh gốc không thường xuyên được truy cập

các mục được lưu trong bộ nhớ cache có thể **đã lỗi thời** (tên cố gắng tốt nhất để dịch địa chỉ!) • nếu máy

chủ được đặt tên thay đổi địa chỉ IP, nó có thể không được biết đến trên toàn Internet **cho đến khi tất cả các TTL hết hạn!** cập nhật/thông báo

cơ chế đề xuất tiêu chuẩn IETF  
• RFC 2136

# Bản ghi DNS

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

Định dạng RR: (tên, giá trị, loại, ttl)

loại = A

tên là tên máy chủ

giá trị là địa chỉ IP

type=NS

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

gõ=CNAME

tên là tên bí danh của một số tên

“chuẩn” (tên thật) www.ibm.com thực sự  
là

servereast.backup2.ibm.com

giá trị là tên chuẩn

loại = MX

giá trị là tên của mailserver  
liên kết với tên miền

# Thông báo giao thức DNS

Tin nhắn trả lời và truy vấn DNS , cả hai đều có cùng định dạng:

tiêu đề thông báo:

nhận dạng: 16-bit # cho truy vấn, trả

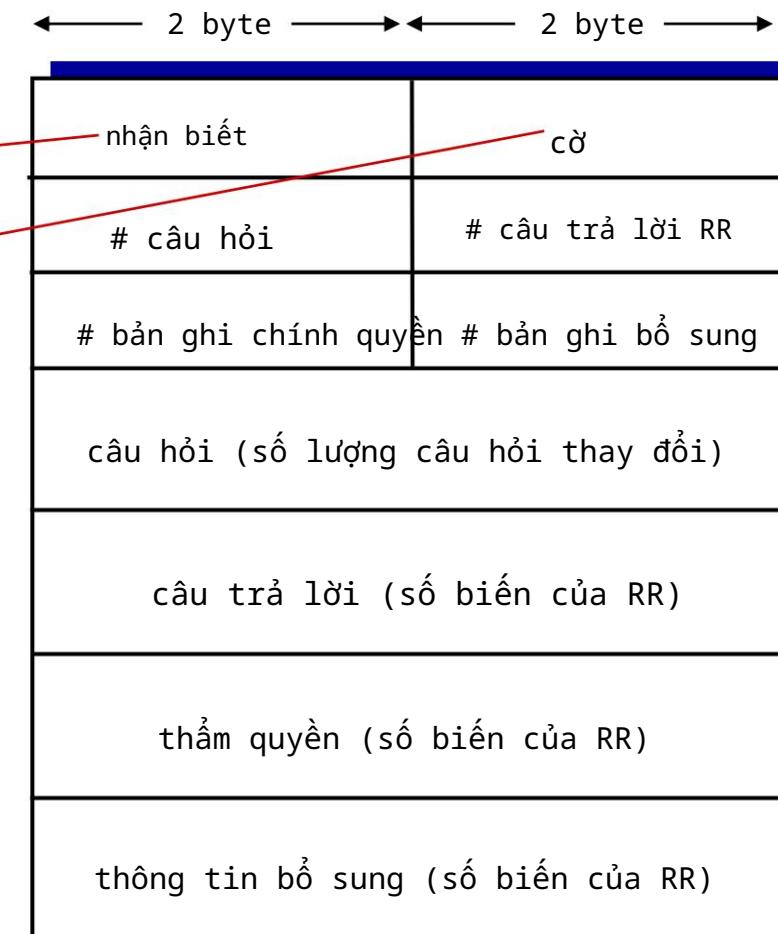
lời truy vấn sử dụng cùng một #

cờ: truy vấn hoặc trả lời • đệ quy mong

- muốn

- đệ quy có sẵn • trả

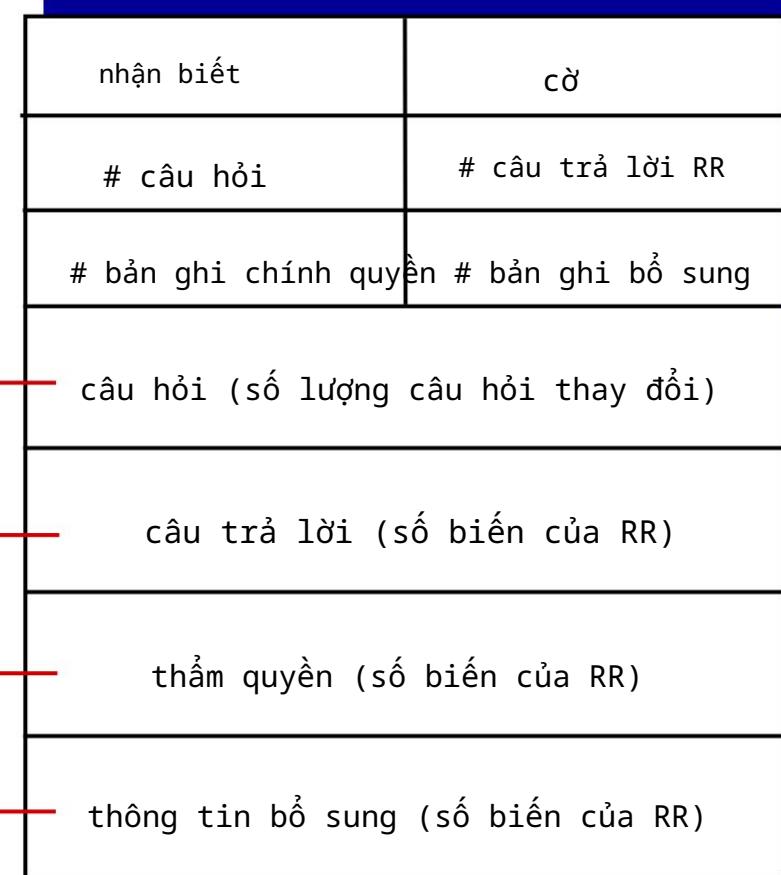
lời là có thẩm quyền



# Thông báo giao thức DNS

Tin nhắn trả lời và truy vấn DNS , cả hai đều có cùng định dạng:

← 2 byte → ← 2 byte →



tên, loại trường cho một truy vấn

→ câu hỏi (số lượng câu hỏi thay đổi)

RR để đáp ứng với truy vấn

→ câu trả lời (số biến của RR)

hồ sơ cho các máy chủ có thẩm quyền

→ thảm quyền (số biến của RR)

thông tin “hữu ích” bổ sung có thể được sử dụng

→ thông tin bổ sung (số biến của RR)

# Chèn bản ghi vào DNS

Ví dụ: startup mới “Network Utopia”

đăng ký tên networkutopia.com tại công ty đăng ký DNS (ví dụ: Network  
Giải pháp)

- cung cấp tên, địa chỉ IP của máy chủ tên có thẩm quyền (chính và  
phụ) •

công ty đăng ký chèn RR (ví dụ: NS, A ) vào máy chủ .com TLD:

(networkutopia.com, dns1.networkutopia.com, NS)  
(dns1.networkutopia.com, 212.212.212.1, A)

tạo cục bộ máy chủ có thẩm quyền với địa chỉ IP 212.212.212.1 • nhập bản  
ghi A cho www.networkutopia.com • nhập bản ghi MX cho networkutopia.com

# bảo mật DNS

## tấn công DDoS

tấn công **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**

oanh tạc **các máy chủ TLD**

- có khả năng **nguy hiểm** hơn

## Chuyển hướng tấn công

**người trung gian**

- chặn các truy vấn DNS

Đầu **độc DNS** • gửi tin cậy

không có thật đến máy chủ DNS,  
nơi i lưu trữ bộ đệm

Khai thác DNS để **DDoS** gửi truy  
vấn với **địa chỉ nguồn giả mạo** làm IP  
mục tiêu

yêu cầu khuếch đại

DNSSEC

[RFC 4033]



# Lớp Ứng dụng: Tổng quan

Nguyên tắc ứng dụng  
mạng Web và HTTP

E-mail, SMTP, IMAP

Hệ thống tên miền  
DNS

## Ứng dụng P2P

Mạng phân phối nội dung và  
truyền phát video

lập trình ổ cắm với  
UDP và TCP



# Kiến trúc ngang hàng (P2P)

không có máy chủ luôn hoạt động

các hệ thống đầu cuối tùy ý giao tiếp trực tiếp

các máy ngang hàng yêu cầu dịch vụ từ các máy ngang

hàng khác, cung cấp dịch vụ đáp lại các máy ngang

hàng khác • khả năng tự mở rộng - các máy ngang hàng

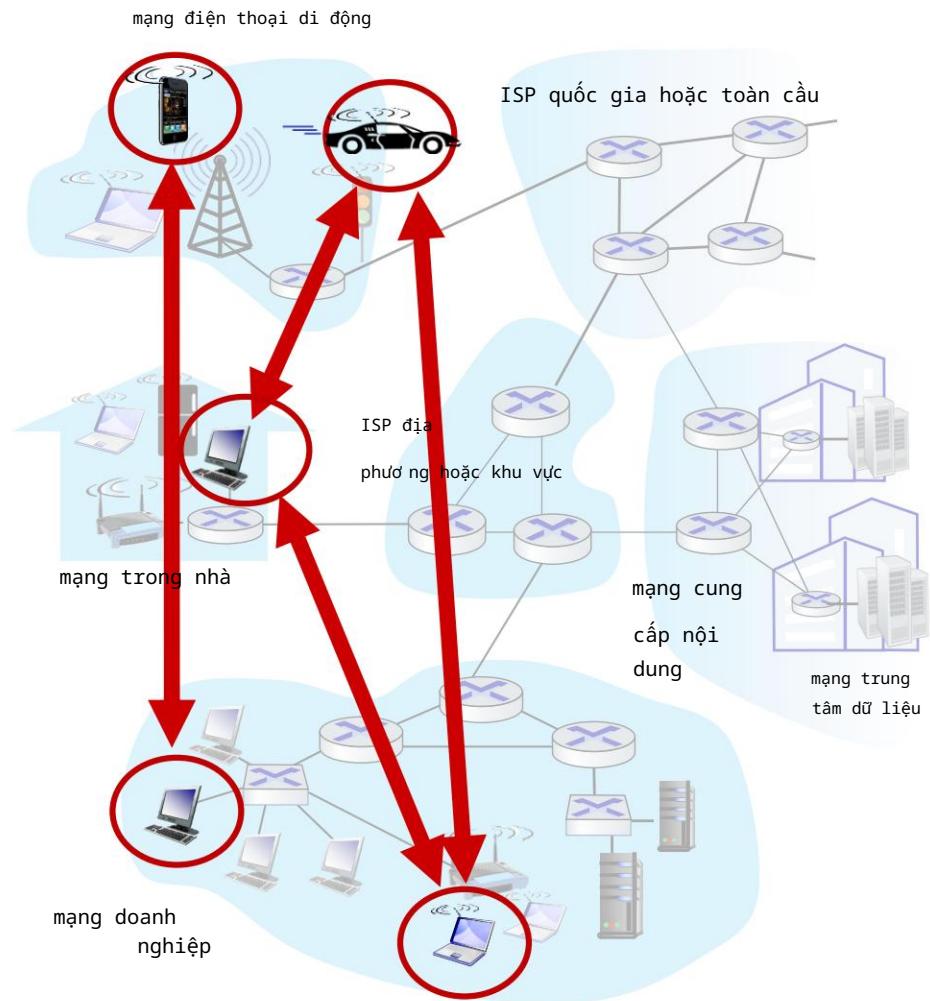
mới mang lại cái mới

**năng lực phục vụ và nhu cầu dịch vụ mới**

các đồng nghiệp được kết nối không liên  
tục và thay đổi địa chỉ IP • quản lý

**phức tạp** ví dụ: chia sẻ tệp P2P

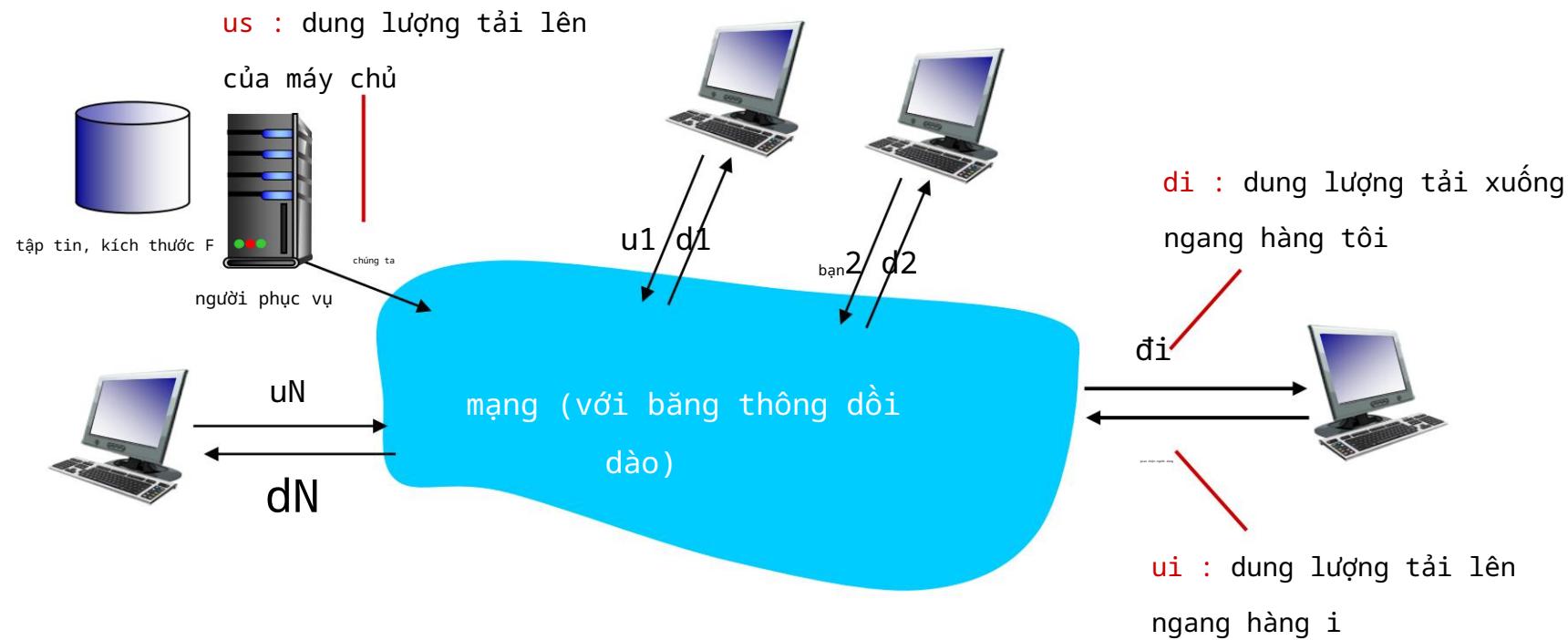
(BitTorrent), truyền trực tuyến (KanKan),  
VoIP (Skype)



# Phân phối tệp: máy khách-máy chủ so với P2P

Hỏi: mất bao nhiêu **thời gian** để phân phối tệp (cỡ F) từ một máy chủ đến N đồng trang

- lứa? dung lượng tải lên/tải xuống hàng là **tài nguyên hạn chế**



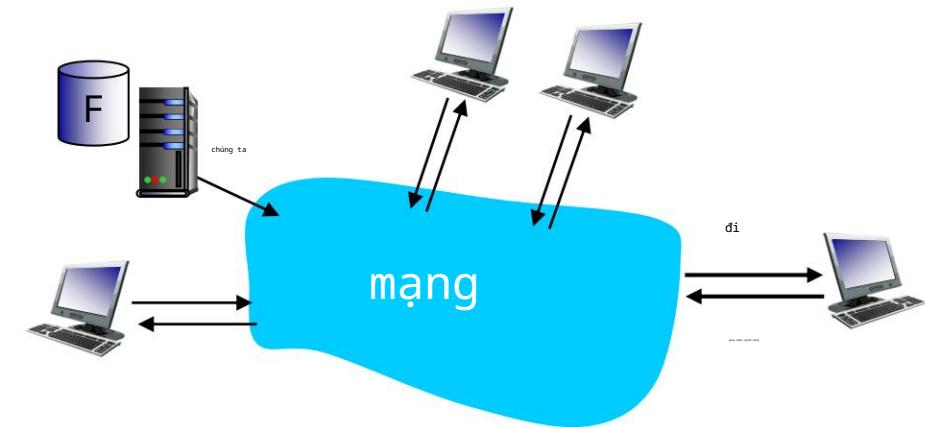
# Thời gian phân phối tệp: máy khách-máy chủ

**truyền máy chủ:** phải gửi (tải lên) liên tục N bản sao

- thời gian gửi một bản sao:  $F/\text{us}$
- thời gian gửi N bản sao:  $NF/\text{us}$

**máy khách:** mỗi máy khách phải tải xuống bản sao

- tệp •  $d_{\min} =$  tốc độ tải xuống máy khách tối thiểu
- thời gian tải xuống máy khách tối thiểu:  $F/d_{\min}$



thời gian để phân phối F cho

N máy khách bằng cách sử

dụng phươ ng pháp máy khách-máy chủ

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

tăng tuyến tính theo N

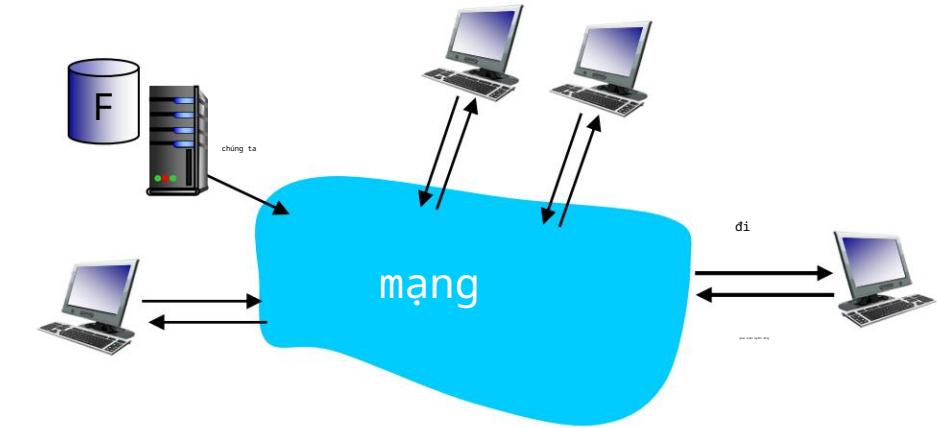
# Thời gian phân phối tệp: P2P

đường truyền máy chủ: phải tải lên ít nhất một bản: • thời gian gửi một bản:  $F/us$

máy khách: mỗi máy khách phải tải xuống bản sao tệp • thời gian tải xuống tối thiểu của máy khách:  $F/dmin$       **máy khách:** dưới dạng

tổng hợp phải tải xuống các bit NF

- tốc độ tải lên tối đa (giới hạn tốc độ tải xuống tối đa) là của chúng tôi +



thời gian để phân phối F

cho N khách hàng sử dụng  
phương pháp P2P

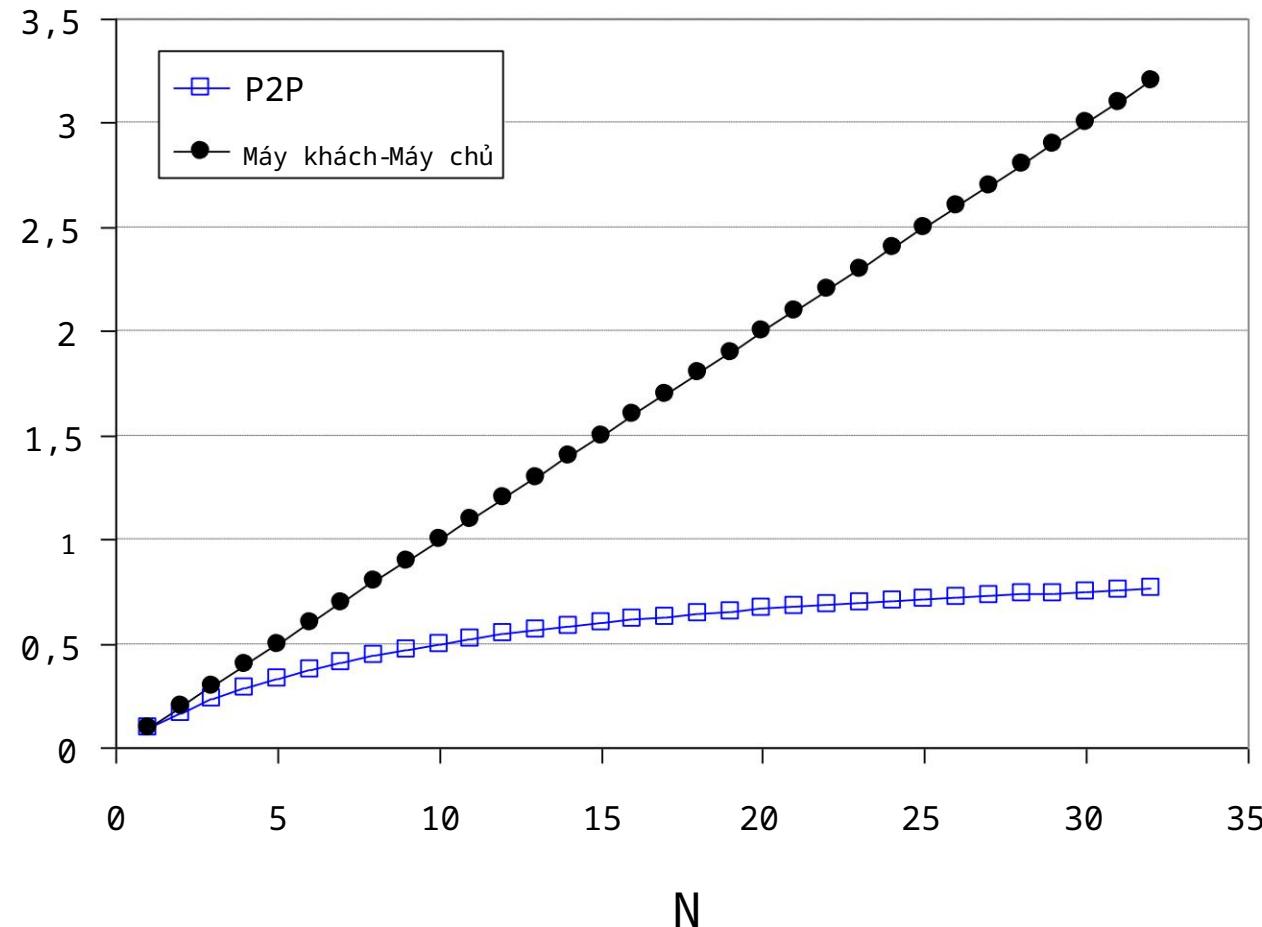
$$DP2P \geq \max\{F/us, , F/dmin, , NF/(us + S_{ui})\}$$

tăng tuyến tính trong N . .

nhưng điều này cũng vậy, vì mỗi đồng đẳng mang lại khả năng phục vụ

# Máy khách-máy chủ so với P2P: ví dụ

tốc độ tải lên của khách hàng =  $u$ ,  $F/u = 1$  giờ, chúng tôi =  $10u$ ,  $d_{min} \geq$  chúng tôi



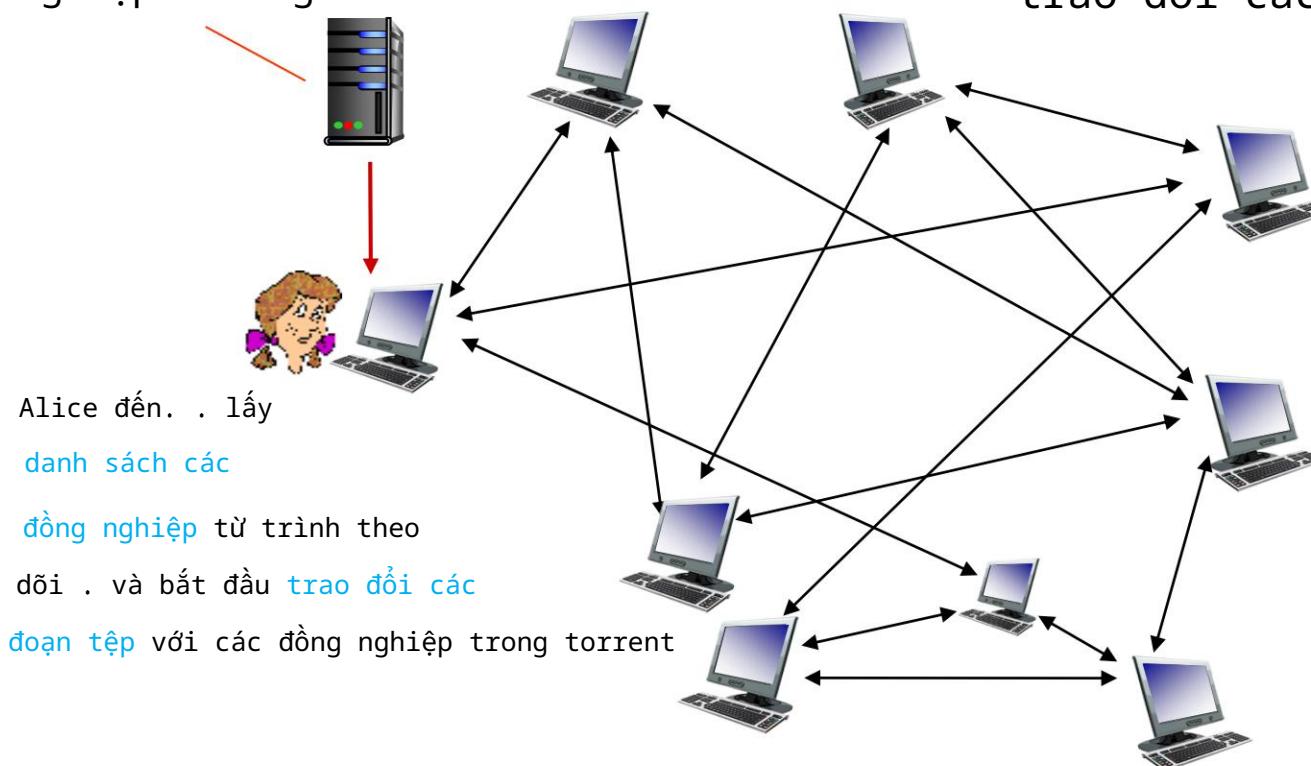
# Phân phối tệp P2P: BitTorrent

tệp được chia thành các khối 256Kb ngang

hàng trong các khối tệp gửi/nhận torrent

**tracker:** theo dõi các đồng nghiệp tham gia torrent

**torrent:** nhóm đồng nghiệp trao đổi các đoạn của tệp



# Phân phối tệp P2P: BitTorrent

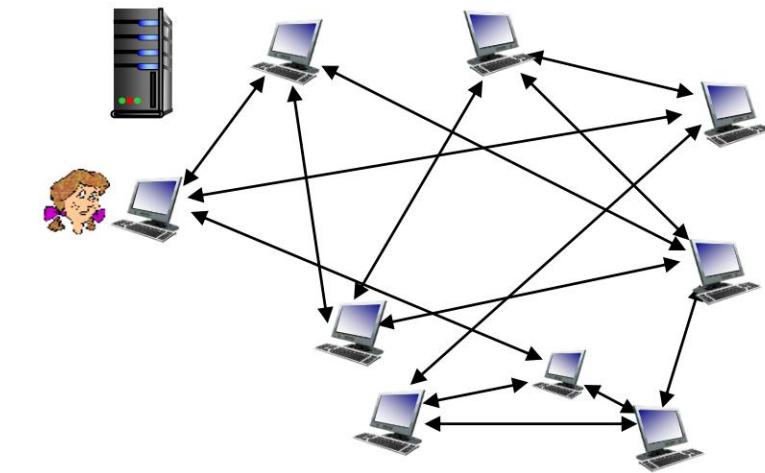
torrent tham gia ngang hàng :

- không có khói, nhưng sẽ tích lũy

chúng theo thời gian từ các ngang hàng khác

- đăng ký với trình theo dõi để nhận danh sách ngang hàng, kết nối với tập hợp con của ngang hàng ("làng giềng") trong khi tải

xuống, ngang hàng tải các khối lên ngang hàng khác ngang hàng có thể thay đổi đồng nghiệp mà nó trao đổi khối với rời bỏ: đồng nghiệp có thể đến và đi một khi đồng đồng có toàn bộ tệp, nó có thể (ích kỷ) rời đi hoặc (vị tha) ở lại trong torrent



# BitTorrent: yêu cầu, gửi khối tệp

Yêu cầu các khối: tại bất kỳ

thời điểm nào, các đồng nghiệp khác nhau

có các tập hợp con khác nhau của các  
khối tệp theo định kỳ, Alice hỏi

từng ngang hàng để biết danh sách các  
khối mà chúng có Yêu cầu Alice bị  
thiếu

khối từ các đồng nghiệp, hiếm nhất  
đầu tiên

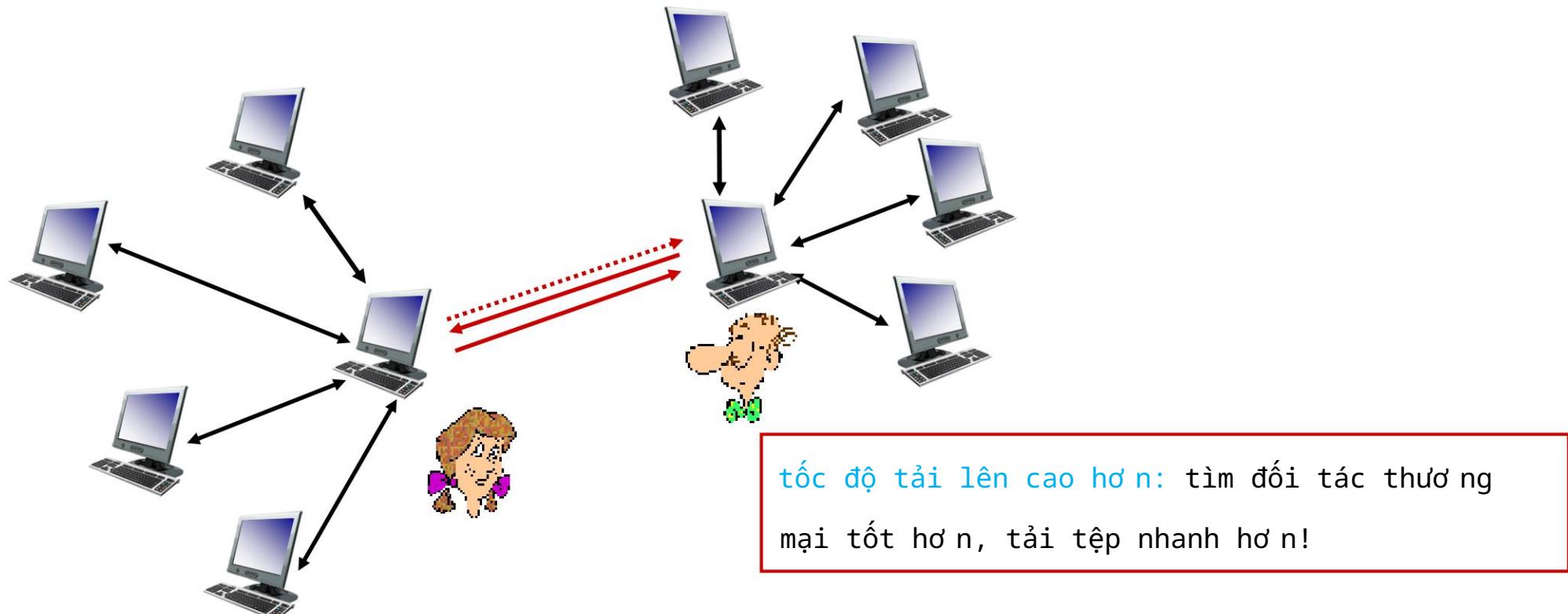
Gửi tin nhắn: ăn miếng trả miếng Alice

gửi tin nhắn cho bốn người đó

các đồng nghiệp hiện đang gửi khối của cô ấy  
với tốc độ cao nhất • các đồng nghiệp khác  
bị Alice bóp cổ (không nhận được khối từ cô  
ấy) • đánh giá lại top 4 cứ sau 10 giây  
và cứ sau 30 giây: chọn ngẫu nhiên một đồng  
nghiệp khác, bắt đầu gửi các khối • “mở một  
cách lạc quan” đồng đẳng này • đồng đẳng  
mới được chọn có thể tham gia top 4

# BitTorrent: ăn miếng trả miếng

- (1) Alice “tháo cuộn” Bob một cách lạc quan
- (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



# Lớp Ứng dụng: tổng quan

Nguyên tắc ứng dụng  
mạng Web và HTTP

E-mail, SMTP, IMAP

Hệ thống tên miền  
DNS

Ứng dụng P2P

Mạng phân phối nội dung và  
truyền phát video

lập trình ổ cắm với  
UDP và TCP



# Truyền phát video và CDN: ngũ cảnh

lưu lượng truy cập video trực tuyến: người sử dụng băng thông

Internet chính , chẳng hạn như Netflix, YouTube, Amazon Prime: 80%



- lưu lượng truy cập ISP dân cư (2020)      **thách thức: quy mô** - làm cách nào để tiếp cận ~1 tỷ người dùng? • máy chủ mega-video duy nhất không hoạt động (tại sao?)      **thách thức: tính không đồng nhất** - làm thế nào để thỏa mãn?

những người dùng khác nhau có các khả năng khác nhau  
(ví dụ: có dây so với di động; giàu băng thông so với  
băng thông kém)      **giải pháp: cơ sở hạ tầng cấp ứng dụng, phân tán**

# Đa phươ ng ti&eth;en: video

**video:** chu&ocirc;i h&imacirc;m ảnh đ&ugrave;t được  
hi&egrave;n thi &ograve; tốc độ kh&ocirc;ng đổi ,

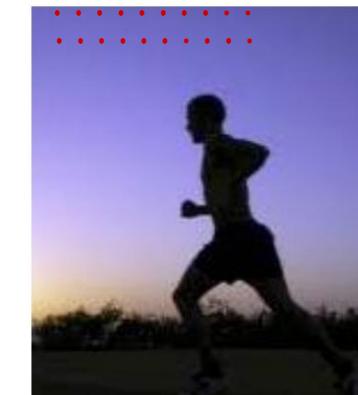
- ví dụ: 24 hình ảnh/giây

**hình ảnh kỹ thuật số:** mảng pixel

- mỗi pixel đ&ugrave;t đ&ugrave;t đại diện bởi các bit

**mã hóa:** sử dụng dự phòng **trong** và **giữa**  
**các** hình ảnh để giảm # bit đ&ugrave;t được sử dụng  
để mã hóa hình ảnh • **không gian** (trong  
hình ảnh) • **thời gian** (từ hình ảnh này  
sang hình ảnh tiếp theo)

ví dụ mã hóa không gian: thay  
vì gửi N giá trị cùng màu (tất  
cả đều màu tím), chỉ gửi hai giá  
trị: giá trị màu (màu tím) và số  
giá trị lặp lại (N)



khung t&ograve;i

ví dụ về mã hóa thời gian:  
thay vì gửi toàn bộ khung  
tại i+1, chỉ gửi các điểm  
khác biệt từ khung i



khung i+1

## Đa phươ ng ti&acute;n: video

CBR (tốc độ bit không đổi):

tốc độ mã hóa video cố định

VBR (tốc độ bit thay đổi): tốc  
độ mã hóa video thay đổi khi  
lượng mã hóa không gian, thời  
gian thay đổi      ví dụ: • MPEG

1 (CD-ROM) 1,5 Mbps • MPEG 2

(DVD) ) 3-6 Mbps • MPEG 4

(thường dùng trong Internet, 64  
Kbps - 12 Mbps)

ví dụ mã hóa không gian: thay  
vì gửi N giá trị cùng màu (tất  
cả đều màu tím), chỉ gửi hai giá  
trị: giá trị màu (màu tím) và số  
giá trị lặp lại (N)



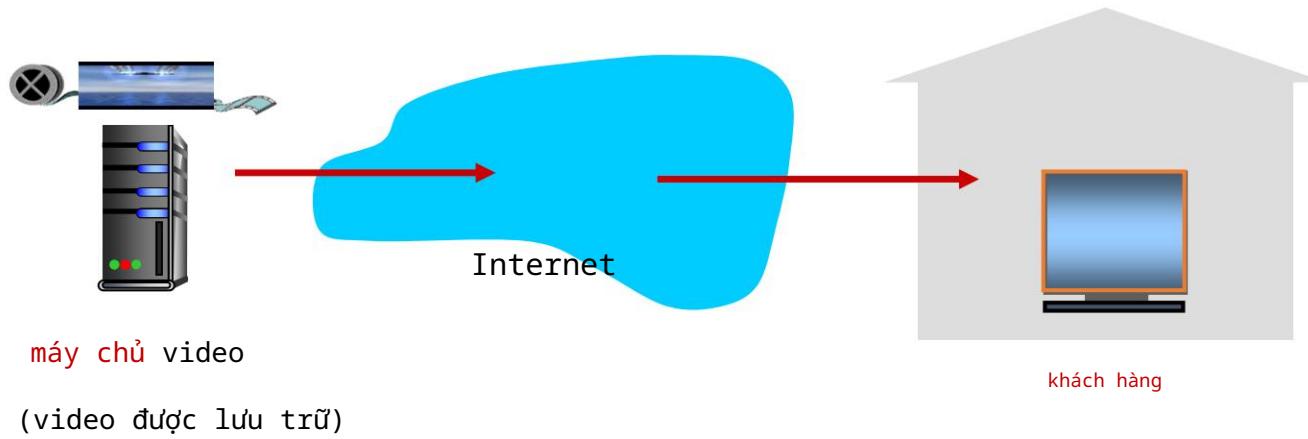
khung t&ocacute;i

ví dụ về mã hóa thời gian:  
thay vì gửi toàn bộ khung  
tại  $i+1$ , chỉ gửi các điểm  
khác biệt từ khung  $i$



khung  $i+1$

## Truyền phát video được lưu trữ kịch bản đơn giản:

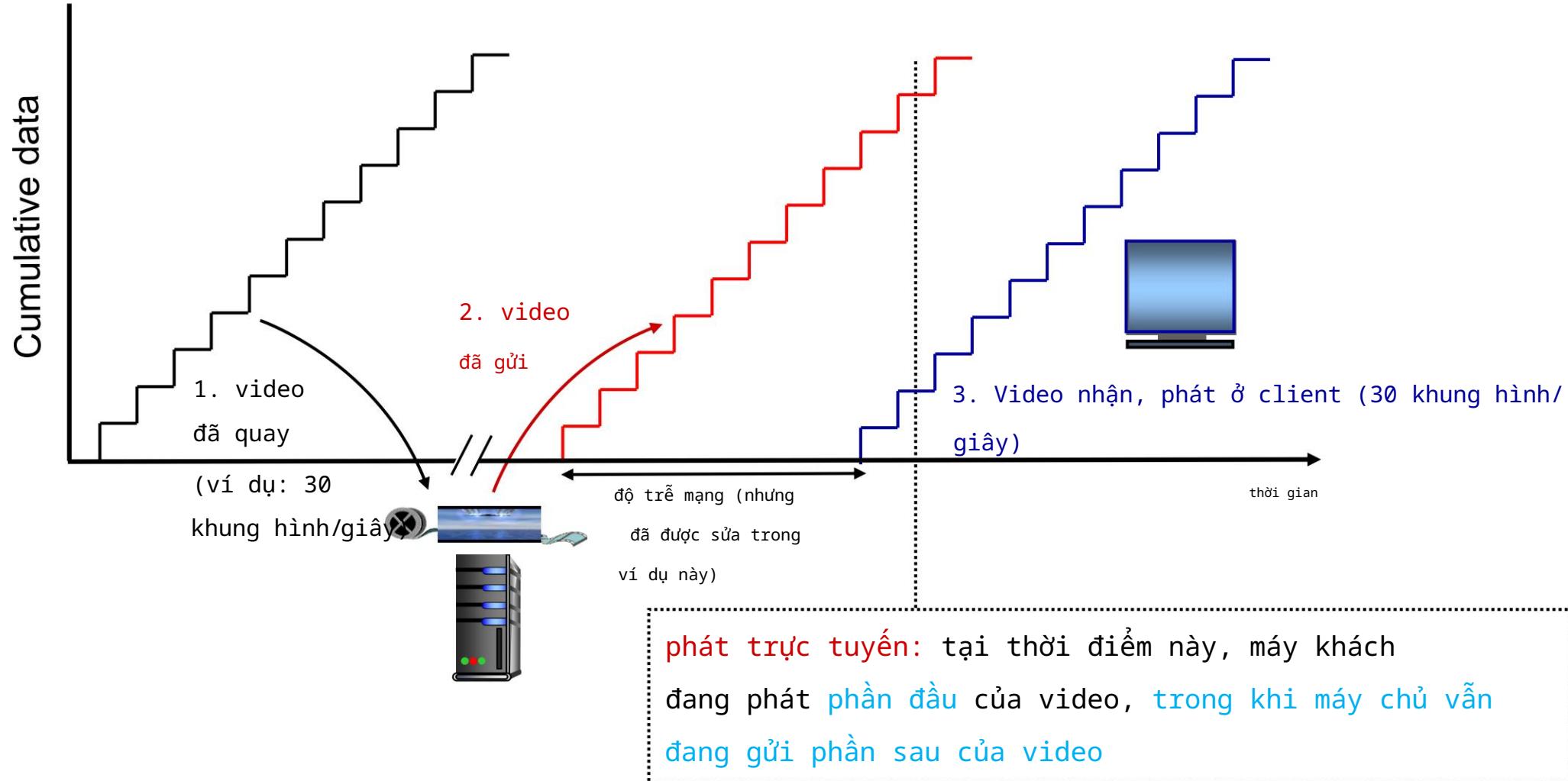


### Những thách thức

chính: **Băng thông** từ máy chủ đến máy khách sẽ **thay đổi** theo thời gian, cùng với việc thay đổi **mức độ tắc nghẽn mạng** (trong nhà, trong mạng truy cập, trong lõi mạng, tại máy chủ video)

**mất gói** và **chậm trễ** do tắc nghẽn sẽ làm chậm quá trình phát hoặc dẫn đến chất lượng video kém

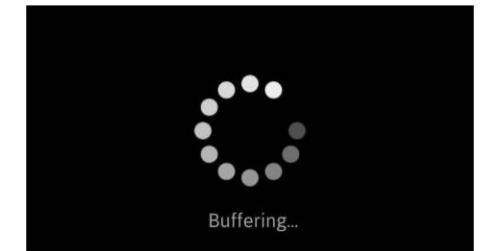
# Truyền phát video được lưu trữ



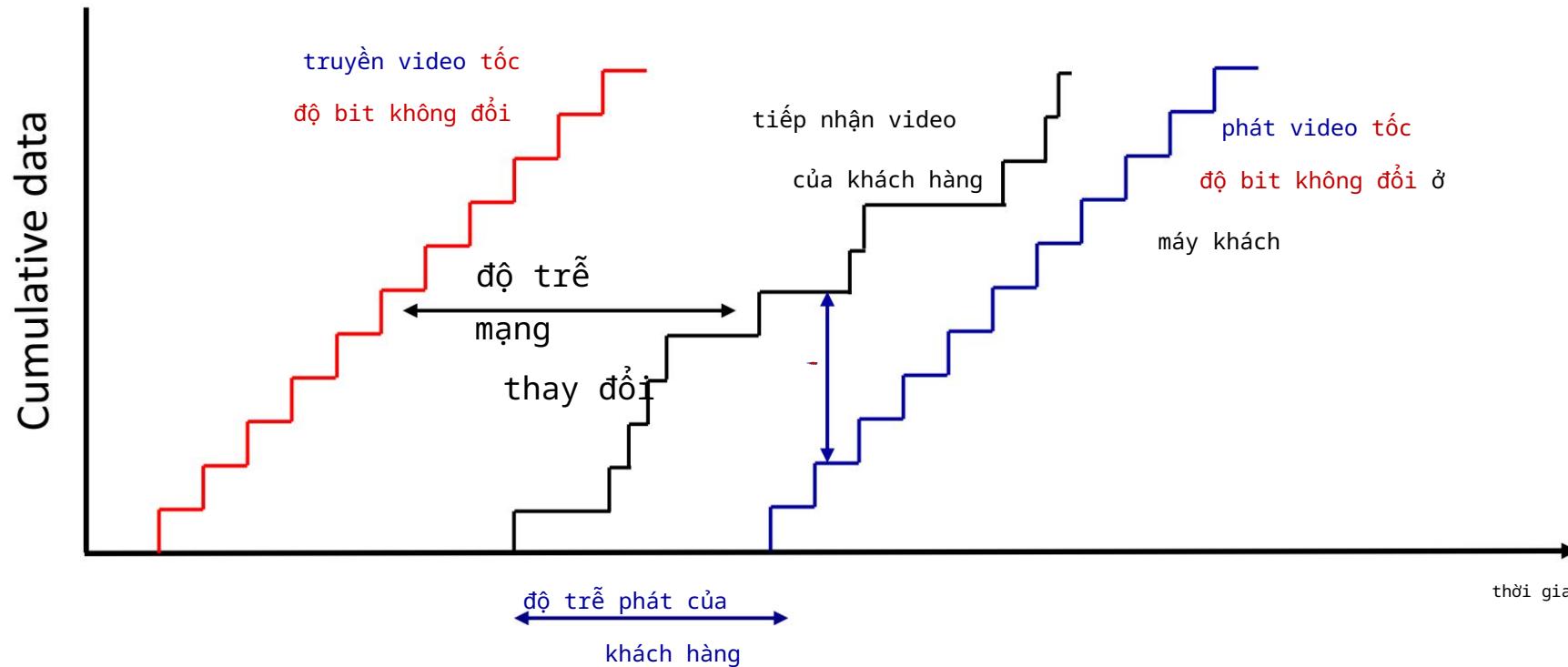
## Truyền trực tuyến video được lưu trữ: thách thức

ràng buộc phát liên tục: một khi khách hàng  
quá trình phát bắt đầu, quá trình phát lại phải khớp với thời  
gian ban đầu . nhưng độ trễ của mạng có thể thay đổi (jitter),

- do đó sẽ cần bộ đệm phía máy khách để phù hợp với các yêu cầu của quá trình phát
- các thách thức khác:
  - tương tác của máy khách: tạm dừng, tua đi nhanh, tua lại, chuyển qua video
  - video các gói có thể bị mất và được truyền lại



# Truyền trực tuyến video được lưu trữ: phát vào bộ đệm



Đệm phía máy khách và độ trễ phát: bù cho độ trễ do mạng thêm vào , độ trễ trễ

# Truyền phát đa phươ ng tiệm: DASH

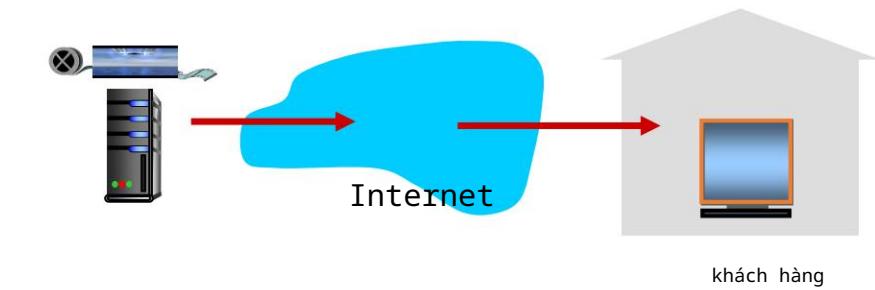
DASH: Truyền phát động, thích ứng qua HTTP

máy chủ:

- chia tệp video thành nhiều phần • mỗi phần được lưu trữ, mã hóa ở các tốc độ khác nhau
- tệp kê khai: cung cấp URL cho các phần khác nhau

khách hàng:

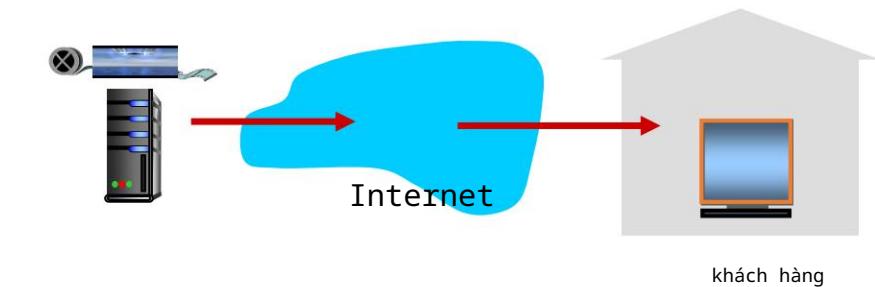
- định kỳ đo băng thông từ máy chủ đến máy khách • bảng kê khai tư vấn, yêu cầu từng đoạn một • chọn tốc độ mã hóa tối đa bền vững cho băng thông hiện tại • có thể chọn tốc độ mã hóa khác nhau tại các thời điểm khác nhau (tùy thuộc vào băng thông khả dụng tại thời điểm đó)



# Truyền phát đa phươ ng tiệm: DASH

“thông minh” tại khách hàng: khách hàng xác định

- **khi nào** yêu cầu đoạn dữ liệu (để không xảy ra tình trạng thiếu hoặc tràn bộ đệm)
- yêu cầu **tốc độ mã hóa** nào ( chất lượng cao hơn khi có nhiều băng thông hơn)
- **nơi** yêu cầu đoạn dữ liệu (có thể yêu cầu từ máy chủ URL “gần” với máy khách hoặc có băng thông khả dụng cao)



Truyền phát video = mã hóa + DASH + bộ đệm phát

# Mạng phân phối nội dung (CDN)

# Mạng phân phối nội dung (CDN)

thách thức: cách truyền phát nội dung (ví dụ: được chọn từ triệu video) đến hàng trăm nghìn (hoặc nhiều hơn) người dùng đồng thời?

tùy chọn 1: một “siêu máy chủ” lớn

- một điểm lỗi • điểm

tắc nghẽn mạng • đường dẫn dài

đến các máy khách ở xa • nhiều

bản sao video được gửi qua liên kết ra ngoài

..khá đơn giản: giải pháp này không mở rộng quy mô

# Mạng phân phối nội dung (CDN)

thách thức: làm cách nào để **truyền phát** nội dung (được chọn từ hàng triệu video) tới hàng trăm nghìn người dùng **đồng thời** ?

tùy chọn 2: lưu trữ/cung cấp nhiều bản sao video tại **nhiều** trang web được phân phối theo địa lý (CDN) • nhập sâu: đầy máy chủ CDN (các nút CDN) vào sâu **trong** nhiều mạng truy cập

- gần gũi với

người dùng • Akamai: 240.000 máy chủ được triển khai tại hơn 120 quốc gia (2015) • mang về nhà: số lượng nhỏ

hơn (10) cụm lớn hơn **trong** các POP gần (nhưng không nằm **trong**) mạng truy cập • được sử dụng bởi Limelight



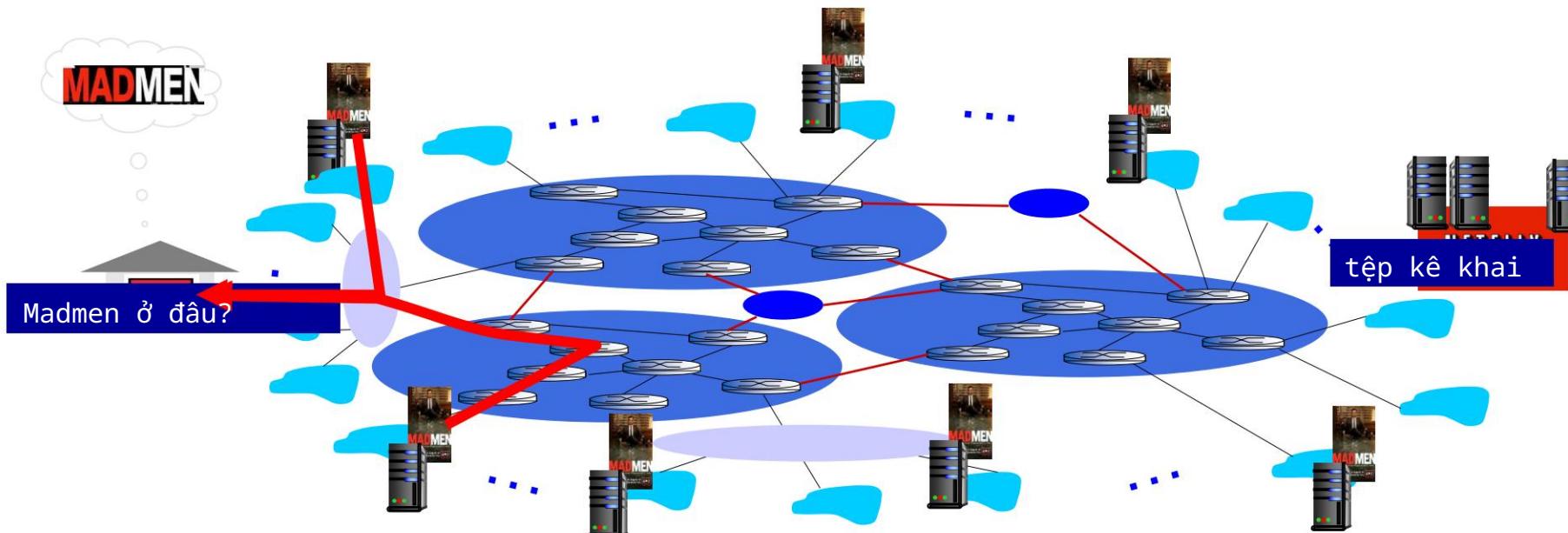
# Mạng phân phối nội dung (CDN)

CDN: lưu trữ các bản sao của nội dung tại các nút

- CDN , ví dụ: Netflix lưu trữ các bản sao của phim

MadMen người đăng ký yêu cầu nội dung từ CDN • chuyển  
đến bản sao gần đó, truy xuất nội dung

- có thể chọn bản sao khác nếu đường dẫn bị tắc nghẽn



# Mạng phân phối nội dung (CDN)



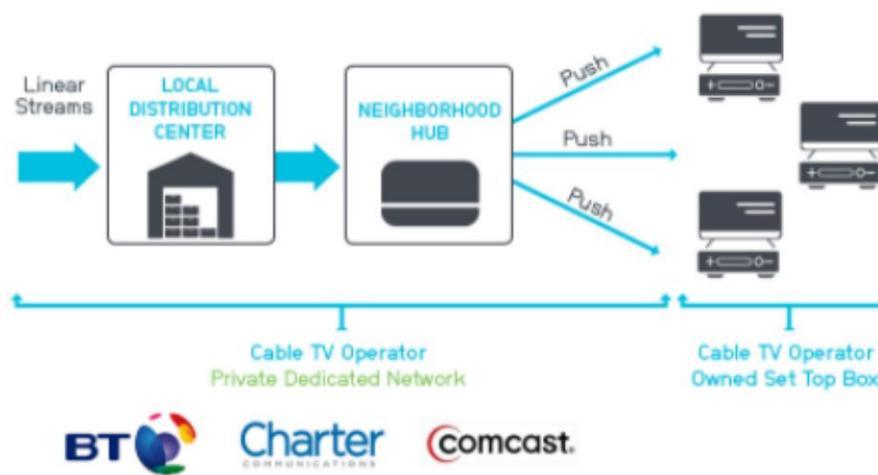
Thách thức OTT: đối phó với Internet tắc nghẽn

từ nút CDN nào để truy xuất nội dung?

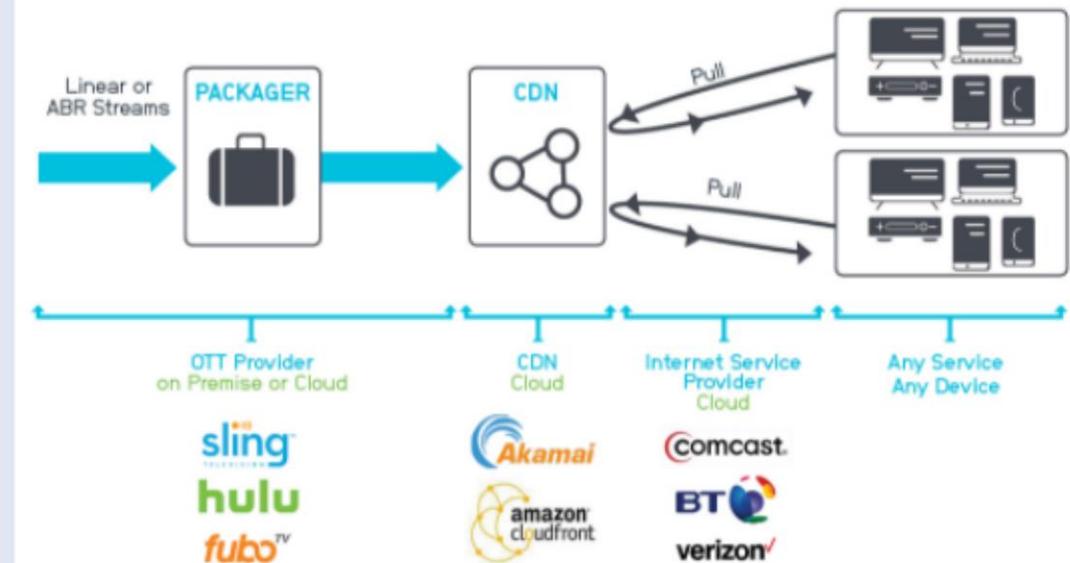
hành vi của người xem khi có tắc nghẽn? đặt  
nội dung nào trong nút CDN nào?

# Chuyển từ IPTV sang OTT

Live IPTV Delivery Environment



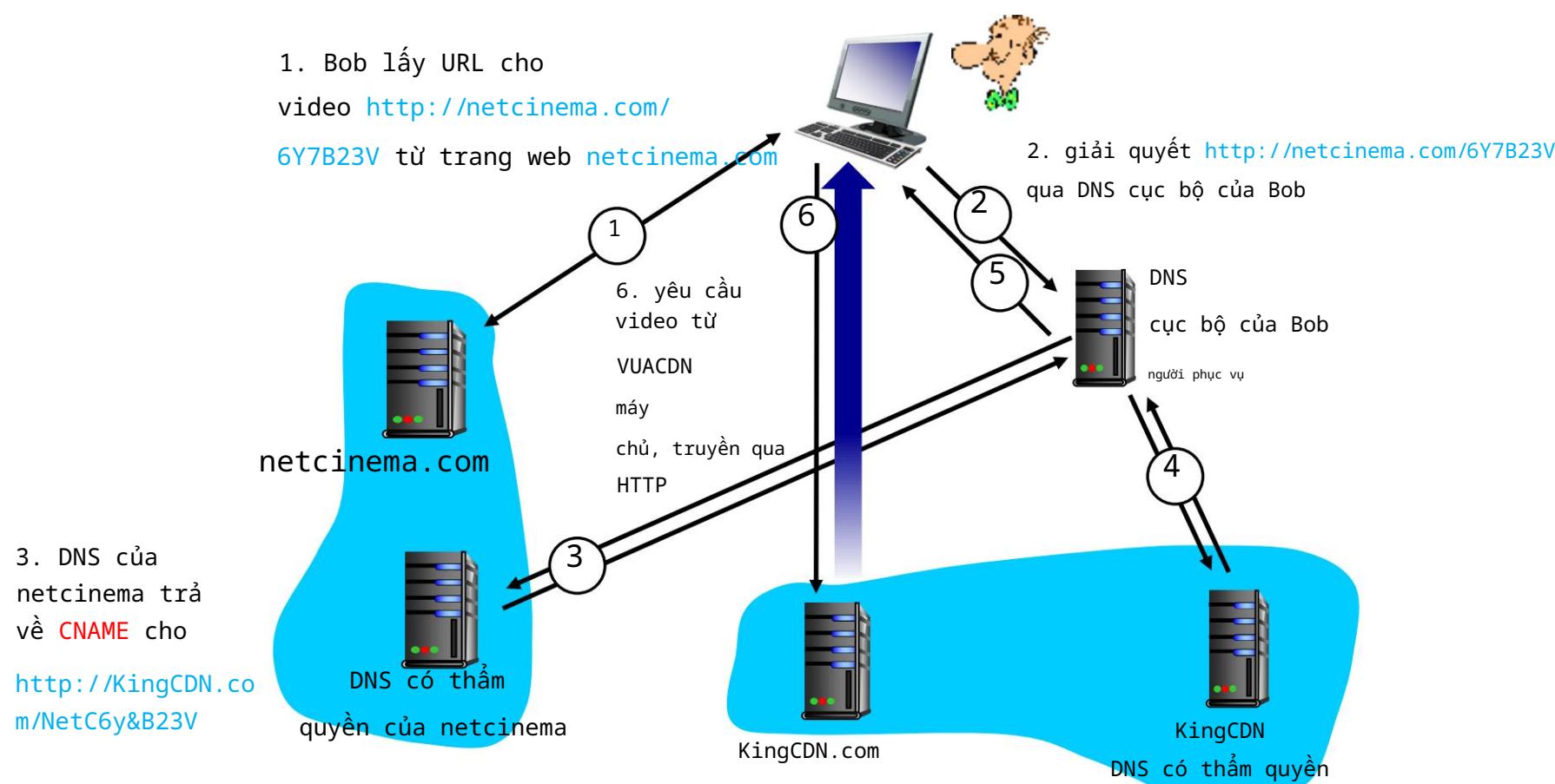
Live OTT Delivery Environment



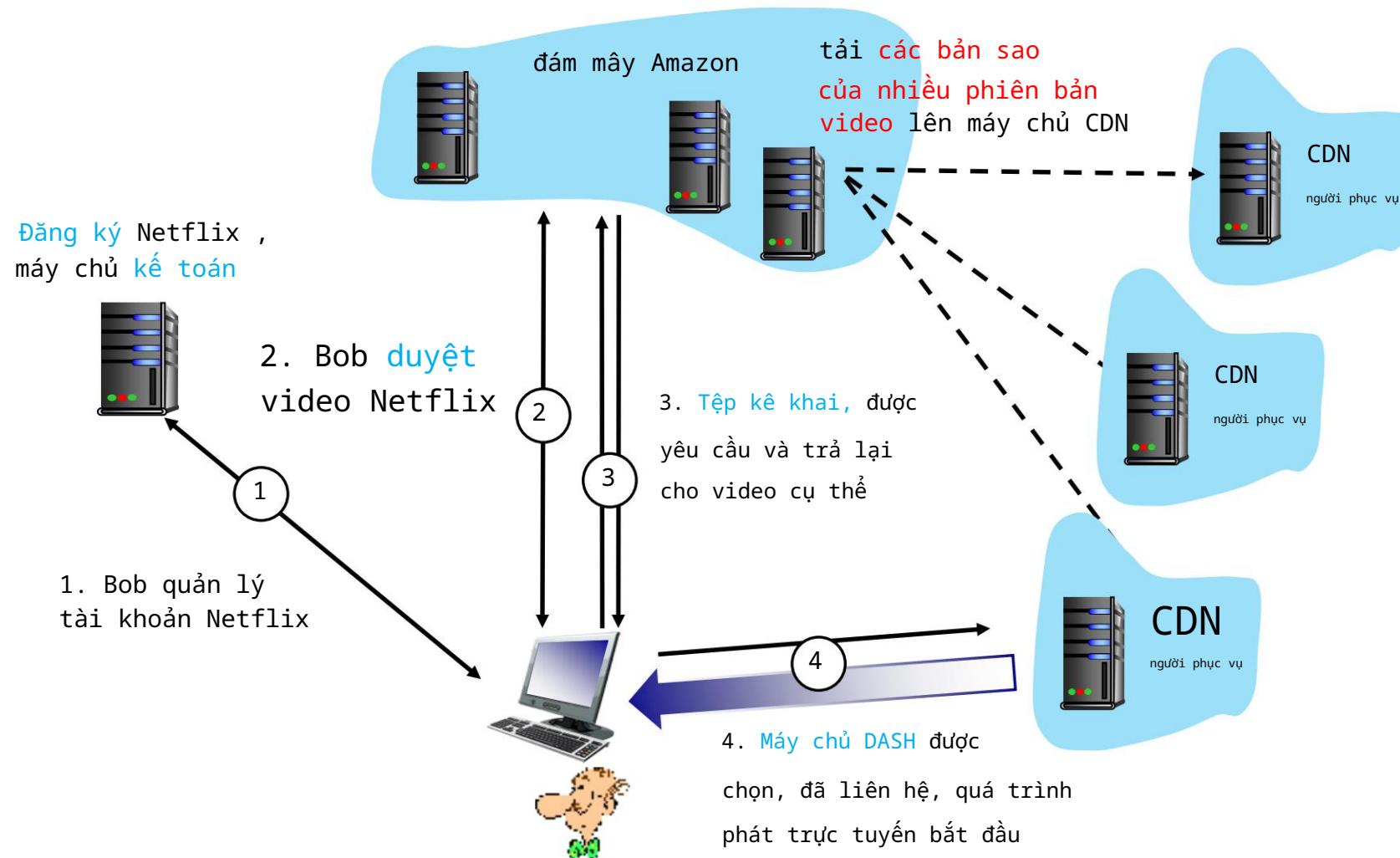
# Truy cập nội dung CDN: xem xét kỹ hơn

Bob (khách hàng) yêu cầu video <http://netcinema.com/6Y7B23V>

video được lưu trữ trong nút CDN tại <http://KingCDN.com/NetC6y&B23V>



## Nghiên cứu điển hình: Netflix



# Lớp Ứng dụng: Tổng quan

Nguyên tắc ứng dụng  
mạng Web và HTTP

E-mail, SMTP, IMAP

Hệ thống tên miền  
DNS

Ứng dụng P2P

Mạng phân phối nội dung và  
truyền phát video

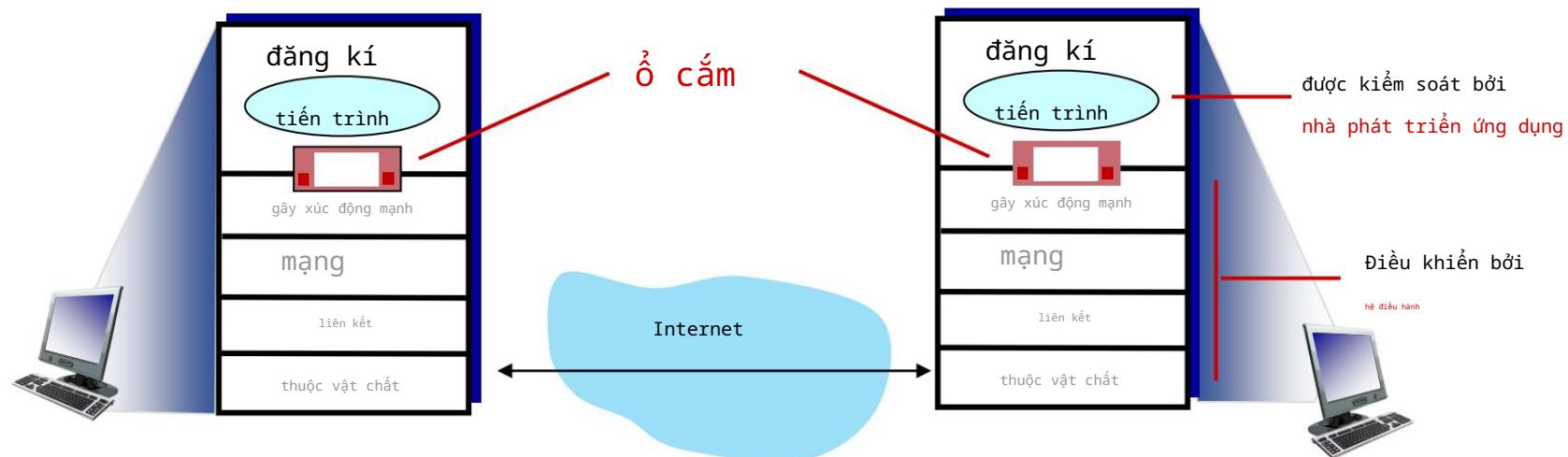
lập trình ổ cắm với  
**UDP và TCP**



# Lập trình ỗ cắm

mục tiêu: tìm hiểu cách xây dựng các ứng dụng máy khách/máy chủ giao tiếp sử dụng ỗ cắm • ỗ

cắm: cửa giữa quá trình ứng dụng và vận chuyển đầu cuối giao thức



# Lập trình ổ cắm

Hai loại ổ cắm cho hai dịch vụ truyền tải: **UDP**:

datagram không đáng tin cậy      **TCP**: đáng tin cậy,  
định hướng theo luồng byte

## Ví dụ ứng dụng đơn giản:

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 cho **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 **người gửi** đính kèm rõ ràng **địa chỉ IP đích** và **số cổng** cho mỗi gói

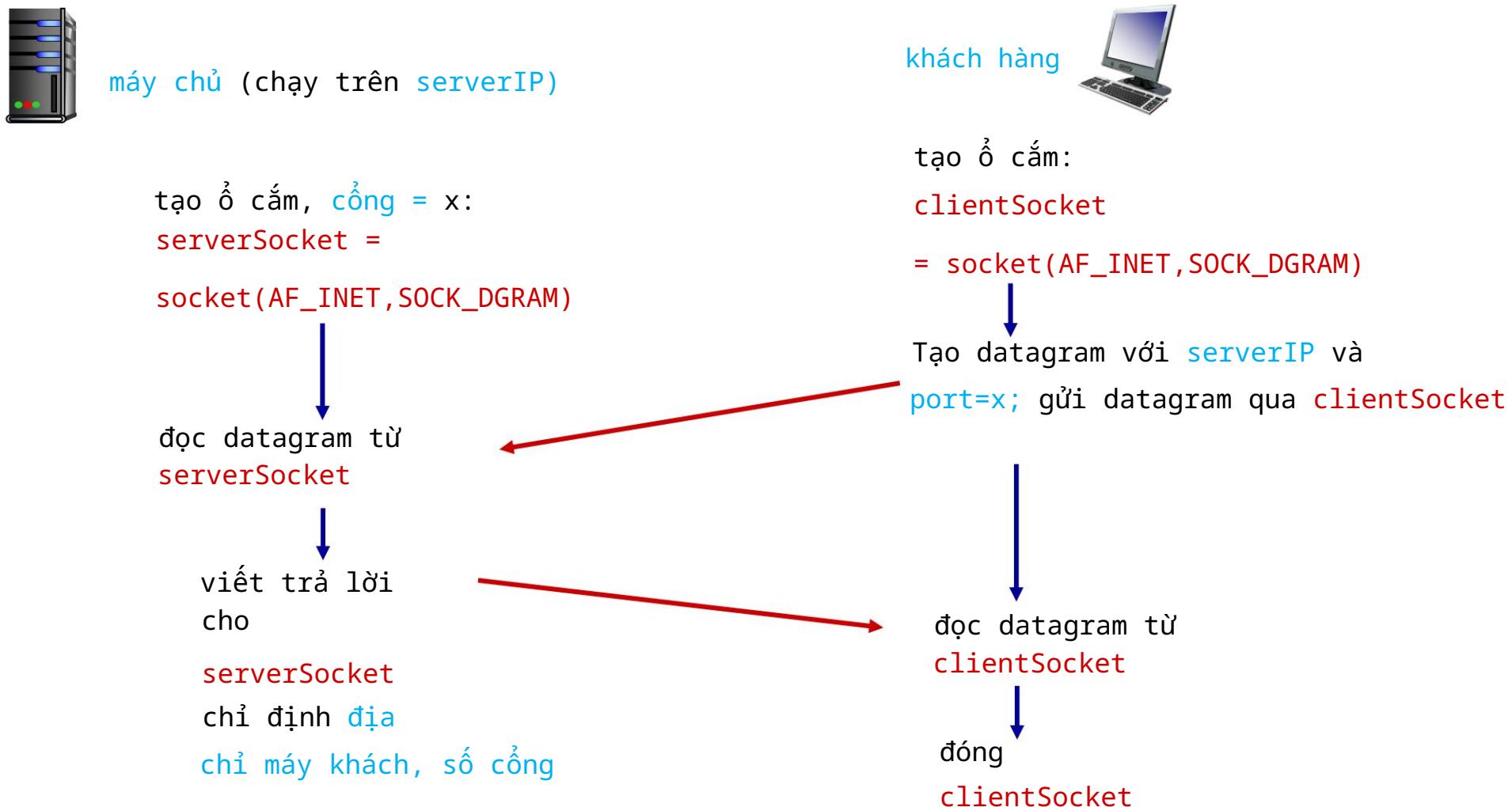
**người nhận** trích xuất **địa chỉ IP** của người gửi và **số cổng** từ gói đã nhận

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

**Quan điểm ứng dụng:** UDP

cung cấp khả năng truyền các nhóm byte (“datagram”) **không đáng tin cậy** giữa máy khách và máy chủ

# Tương tác ổ cắm máy khách/máy chủ: UDP



# Ứng dụng ví dụ: ứng dụng khách UDP

## Python UDPClient

```

* bao gồm thư viện ổ cắm của Python → từ ổ cắm nhập tên
    máy chủ = 'tên máy chủ'
    serverPort = 12000

tạo ổ cắm UDP cho máy chủ → clientSocket = socket(AF_INET,
    SOCK_DGRAM)

nhận đầu vào bàn phím của người dùng → message = raw_input('Nhập câu viết thường: ')
gắn tên máy chủ, cổng vào tin nhắn; gửi vào ổ cắm → clientSocket.sendto(message.encode(), (serverName,
    serverPort)) modMessage,
đọc các ký tự trả lời từ ổ cắm thành chuỗi → serverAddress = clientSocket.recvfrom(2048)

in ra chuỗi nhận được và đóng ổ cắm → in modMessage.decode()
                                         clientSocket.close()

```

## Ứng dụng ví dụ: máy chủ UDP

# Máy chủ UDP Python

		từ ổ cắm nhập
		serverPort = 12000
tạo ổ cắm UDP	→	serverSocket = socket(AF_INET, SOCK_DGRAM)
liên kết ổ cắm với số cổng cục bộ 12000	→	serverSocket.bind(('', serverPort)) print ("Máy chủ sẵn sàng nhận") trong khi True:
vòng lặp mãi mãi	→	
✓ ổ cắm UDP vào tin nhắn, nhận địa chỉ của máy (IP máy khách và cổng)	→	tin nhắn, clientAddress = serverSocket.recvfrom(2048) modMessage = message.decode().upper()
gửi lại chuỗi chữ hoa cho khách hàng này	→	serverSocket.sendto(modifiedMessage.encode(), clientAddress)

# Lập trình socket với TCP

Khách hàng phải liên hệ với máy chủ  
quy trình máy chủ trước tiên phải  
đang chạy  
máy chủ phải tạo **ổ cắm** (cửa) chào đón  
liên hệ của máy khách

Máy khách liên hệ với máy chủ bằng  
cách: tạo **ổ cắm** TCP, chỉ định  
**Địa chỉ IP, số cổng** của quá trình  
máy chủ  
khi máy khách tạo **ổ cắm**: thực  
thể TCP máy khách **thiết lập** kết  
nối với thực thể TCP máy chủ

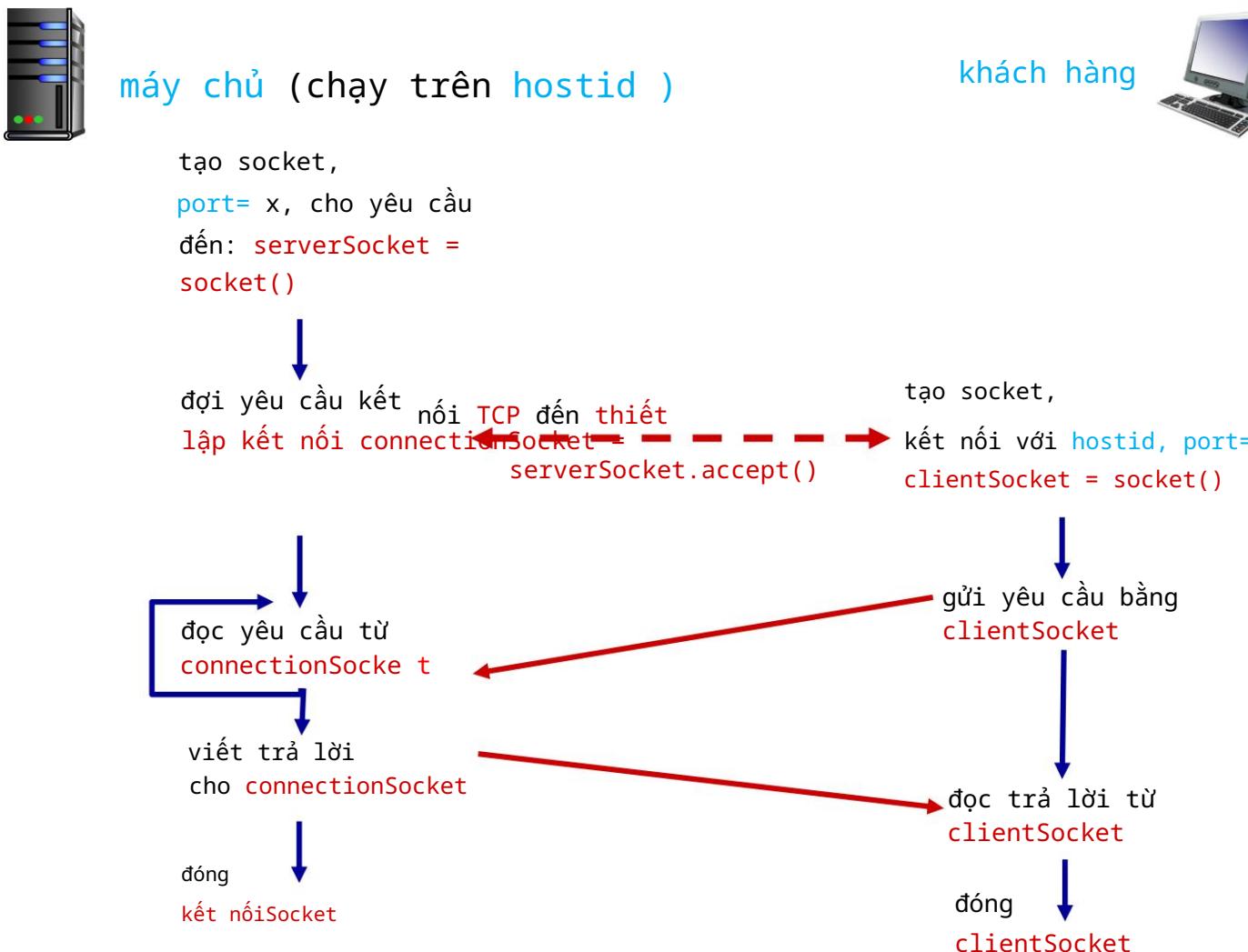
khi được liên lạc bởi máy khách, **máy chủ**  
**TCP** tạo **ổ cắm mới** cho quy trình máy chủ để giao  
tiếp với máy khách cụ thể đó • cho phép máy chủ  
nói chuyện với **nhiều máy khách** • **số cổng nguồn**  
được sử dụng để

phân biệt khách hàng (thêm trong Chap 3)

## quan điểm ứng dụng

TCP cung cấp khả **năng truyền luồng**  
**byte theo thứ tự, đáng tin cậy** ("đường  
ống") giữa máy khách và máy chủ

# Tương tác ổ cắm máy khách/máy chủ: TCP



# Ứng dụng ví dụ: Máy khách TCP

## Python TCPClient

tạo TCP socket cho server,  
remote port 12000

Không cần đính kèm tên máy chủ, cổng

```
*  
từ ổ cắm nhập tên  
máy chủ = 'tên máy chủ'  
máy chủPort = 12000  
  
clientSocket = socket(AF_INET, SOCK_STREAM)  
clientSocket.connect((serverName,serverPort)) câu =  
raw_input('Nhập câu viết thường: ')  
clientSocket.send(sentence.encode()) ModifiedSentence  
= clientSocket.recv(1024) print (' Từ Máy chủ: ', đã  
sửa đổisentence.decode()) clientSocket.close()
```

# Ứng dụng ví dụ: máy chủ TCP

```

Python TCPServer từ ồ
*
cắm nhập serverPort =
12000 serverSocket =
socket(AF_INET,SOCK_STREAM)
serverSocket.bind( '',serverPort)
serverSocket.listen(1) print 'Máy chủ đã sẵn sàng
nhận' trong khi True: connectionSocket, addr =
serverSocket .Chấp nhận()

mãi máy chủ chờ chấp nhận () cho các yêu cầu
đến, ổ cắm mới được tạo khi trả về

đọc byte từ ổ cắm (nhưng không
phải địa chỉ như trong UDP) →
câu = connectionSocket.recv(1024).decode() vốn
hóaSentence = câu.upper() connectionSocket.send(viết
hoaSentence. mã hóa())

đóng kết nối với máy khách này (nhưng không
chào đón ổ cắm) →
kết nốiSocket.close()

```

## Chương 2: Tổng kết

nghiên cứu của chúng tôi về lớp ứng dụng mạng hiện đã hoàn tất!

kiến trúc ứng dụng • máy  
khách-máy chủ

- P2P

yêu cầu dịch vụ ứng dụng: • độ tin cậy,  
băng thông, độ trễ, bảo mật

Mô hình dịch vụ truyền tải Internet •  
định hướng kết nối, đáng tin cậy: TCP •  
không đáng tin cậy, datagram: UDP

giao thức cụ thể:

- HTTP
- SMTP, IMAP •

DNS

- P2P: BitTorrent

truyền phát video, CDN

lập trình ổ cắm:

Ổ cắm TCP, UDP

## Chương 2: Tổng kết

Quan trọng nhất: đã học về các giao thức!

trao đổi bản tin yêu cầu/trả lời điển

hình : • máy khách yêu cầu thông tin  
hoặc dịch vụ • máy chủ phản hồi với dữ  
liệu, mã trạng thái

định dạng thông báo: •

tiêu đề: trường cung cấp thông tin về dữ  
liệu • dữ liệu: thông tin (tải trọng) được  
truyền đạt

các chủ đề quan trọng:

tập trung so với phi tập trung  
không trạng thái so với trạng thái  
khả năng mở rộng truyền tin nhắn  
đáng tin cậy so với không đáng tin cậy  
“sự phức tạp ở biên mạng”

Các slide Chương 2 bổ sung