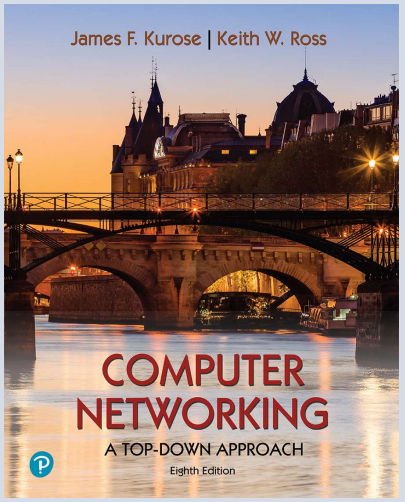
MẠNG MÁY TÍNH
(Computer Networking)

Tầng Ứng Dụng
(Application Layer)



Computer Networking: A Top-Down Approach
8th edition
Jim Kurose, Keith Ross
Pearson, 2020

TS. Nguyễn Mạnh Cương
Khoa CNTT, ĐH Nha Trang

Tầng Ứng dụng: Tổng quan

- Nguyên lý của ứng dụng mạng
- Web và HTTP
- E-mail, SMTP, IMAP
- Hệ thống tên miền DNS

Tầng Ứng dụng: Tổng quan

Mục tiêu:

- Các khía cạnh về khái niệm và thực thi của các giao thức tầng ứng dụng.
  - các mô hình dịch vụ tầng vận chuyển
  - mô hình client-server
  - mô hình peer-to-peer
- Tìm hiểu về các giao thức bằng cách xem xét các giao thức tầng ứng dụng và cơ sở hạ tầng phổ biến.
  - HTTP
  - SMTP, IMAP
  - DNS

Một số ứng dụng mạng

- Mạng xã hội (FB, Twitter)
- Web
- Tin nhắn văn bản (messenger, …)
- e-mail (Gmail, Yahoo mail, …)
- Trò chơi mạng nhiều người dùng
- Phát video được lưu trữ dưới dạng luồng (streaming). (YouTube, Hulu, Netflix, Apple TV)
- Chia sẽ tệp tin ngang hàng (P2P)
- Âm thanh qua IP (Skype)
- Hội nghị truyền hình thời gian thực (Zoom, Google meet, …)
- Tìm kiếm Internet (Google, Bing)
- Đăng nhập từ xa (Teamviewer, Ultraviewer)
- …

Q: yêu thích của bạn?

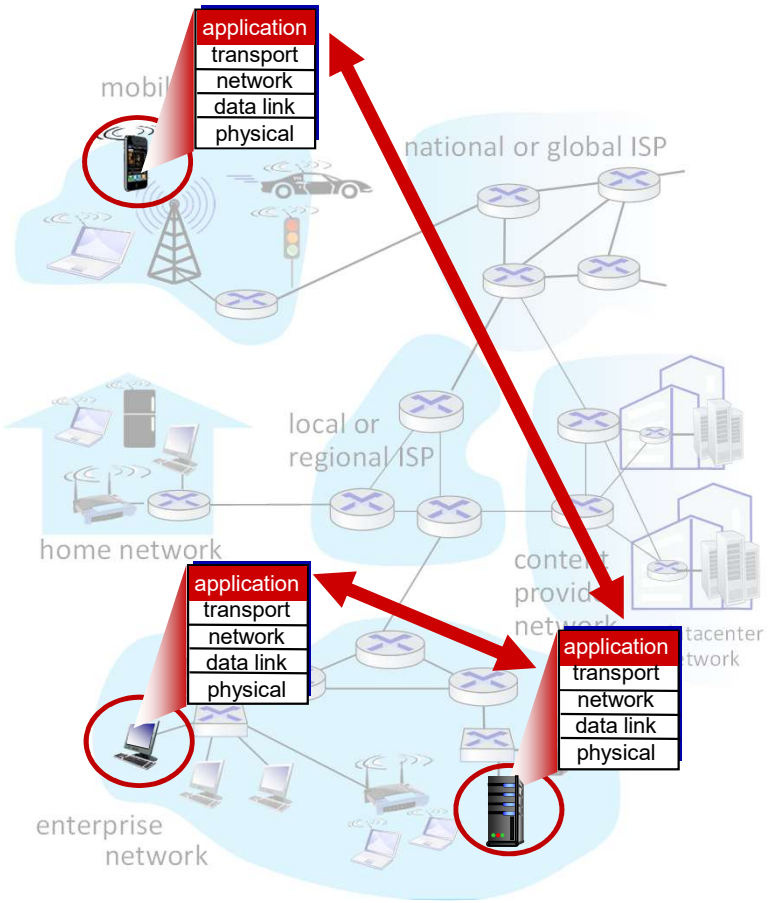
Tạo một ứng dụng mạng

viết chương trình để:

- 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/network-core

- thiết bị lõi mạng không chạy ứng dụng người dùng
- các ứng dụng trên hệ thống đầu cuối cho phép phát triển, phổ biến ứng dụng nhanh chóng



Mô hình client-server

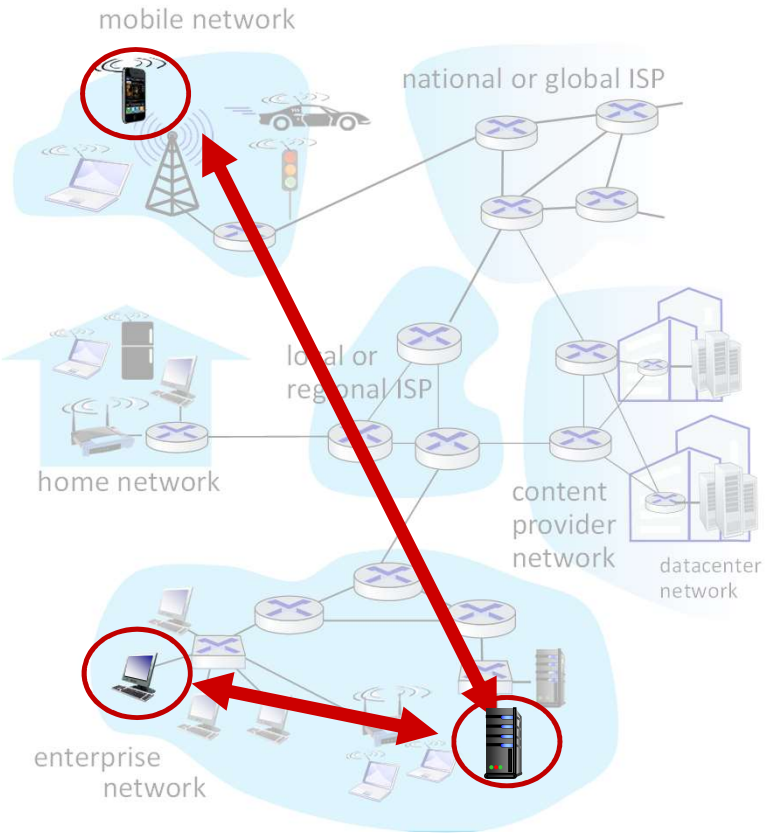
server/máy chủ:

- máy chủ luôn được bật
- địa chỉ IP cố định
- thường đặt tại các trung tâm dữ liệu để tăng khả năng mở rộng

clients/máy khách:

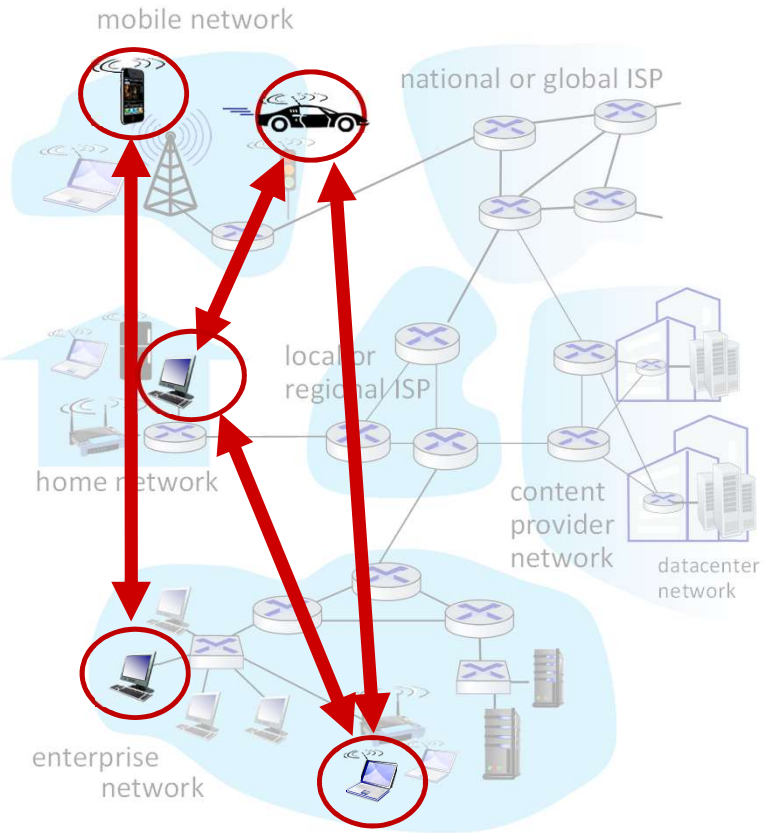
- liên hệ, giao tiếp với máy chủ
- có thể được kết nối không liên tục
- có thể có địa chỉ IP động
- không giao tiếp trực tiếp với nhau

ví dụ: HTTP, IMAP, FTP



Kiến trúc ngang hàng (Peer-peer architecture)

- không có máy chủ luôn bật
- hệ thống đầu cuối tùy ý giao tiếp trực tiếp
- peers yêu cầu dịch vụ từ các peers khác, đồng thời cung cấp dịch vụ cho peers khác
  - *tự mở rộng* - peers mới mang lại khả năng dịch vụ mới, cũng như nhu cầu dịch vụ mới
- các người dùng kết nối không liên tục và thay đổi địa chỉ IP
  - quản lý phức tạp
- ví dụ: Chia sẻ tệp P2P



Giao tiếp giữa các tiến trình

Process/tiến trình: chương trình chạy bên trong máy chủ/host

- trong cùng một host, hai tiến trình giao tiếp bằng cách sử dụng giao tiếp giữa các tiến trình (được định nghĩa bởi HĐH)
- các tiến trình trong các host khác nhau giao tiếp bằng cách trao đổi thông điệp/messages

clients, servers

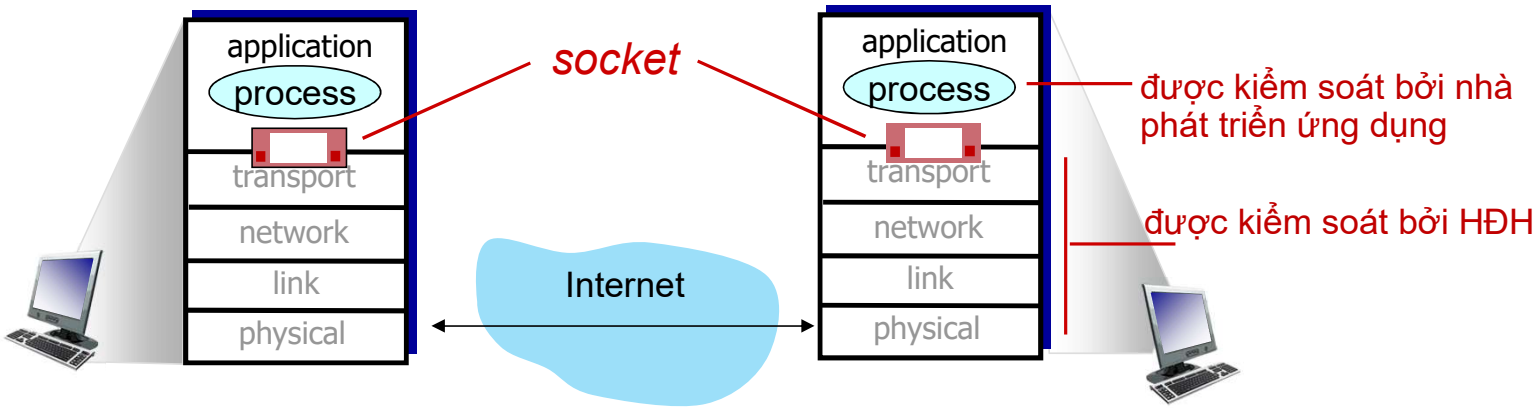
client process: tiến trình khởi tạo giao tiếp

server process: tiến trình đợi được liên hệ

- Ghi chú: các ứng dụng có kiến trúc P2P có các tiến trình client và các tiến trình server

Sockets/Ổ cắm

- tiến trình gửi/nhận thông điệp đến/từ socket của nó
- ổ cắm tương tự như cửa
  - tiến trình gửi đẩy tin nhắn ra khỏi cửa
  - tiến trình gửi thông điệp phụ thuộc vào cơ sở hạ tầng vận chuyển ở phía bên kia cửa để gửi thông điệp đến socket của tiến trình nhận
  - Hai socket có liên quan: một socket ở mỗi bên



Địa chỉ hóa các tiến trình

- để nhận thông điệp, tiến trình phải có *định danh*
- thiết bị chủ có địa chỉ IP 32 bit duy nhất
- *Q:* địa chỉ IP của host 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 host

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

Một giao thức tầng ứng dụng định nghĩa:

- các loại thông điệp được trao đổi,
  - ví dụ: yêu cầu, phản hồi
- cú pháp thông điệp:
  - các trường trong thông điệp và cách phân cách các trường.
- ngữ nghĩa thông điệp
  - ý nghĩa của thông tin trong các trường dữ liệu
- quy tắc cho việc khi nào và cách các tiến trình gửi và phản hồi thông điệp thế nào

các giao thức mở:

- được định nghĩa trong các tài liệu RFC, tất cả mọi người đều có quyền truy cập vào định nghĩa giao thức
- cho phép khả năng tương kết/tương thích
- ví dụ: HTTP, SMTP

các giao thức độc quyền:

- ví dụ: Skype, Zoom

Ứng dụng cần dịch vụ vận chuyển nào?

tính toàn vẹn dữ liệu:

- một số ứng dụng (ví dụ như truyền tập tin, giao dịch web) yêu cầu truyền dữ liệu tin cậy 100%
- Các ứng dụng khác (ví dụ như âm thanh) có thể chấp nhận mất mát một số dữ liệu.

timing/định thời

- một số ứng dụng (ví dụ như điện thoại Internet, trò chơi tương tác) yêu cầu độ trễ thấp để có "hiệu quả"

throughput/thông lượng

- một số ứng dụng (ví dụ: đa phương tiện) yêu cầu lượng thông lượng tối thiểu nào đó để "hiệu quả"
- các ứng dụng khác ("ứng dụng mềm dẽo") tận dụng bất kỳ thông lượng nào mà chúng nhận được

security/bảo mật

- mã hóa, tính toàn vẹn dữ liệu,...

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

ứng dụng	mất dữ liệu	thông lượng	nhạy cảm thời gian?
file transfer/download	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	no loss	elastic	no
real-time audio/video	loss-tolerant	audio: 5Kbps-1Mbps video:10Kbps-5Mbps	yes, 10's msec
streaming audio/video	loss-tolerant	same as above	yes, few secs
interactive games	loss-tolerant	Kbps+	yes, 10's msec
text messaging	no loss	elastic	yes and no

Các dịch vụ giao thức vận chuyển Internet

dịch vụ TCP:

- *vận chuyển* tin cậy giữa tiến trình gửi và nhận
- *kiểm soát luồng:* bên gửi sẽ không làm ngập bên nhận
- *kiểm soát tắc nghẽn:* điều tiết bên gửi khi mạng quá tải
- *hướng kết nối:* yêu cầu thiết lập kết nối giữa các tiến trình client và server
- *không cung cấp:* định thời, đảm bảo thông lượng tối thiểu, bảo mật

dịch vụ UDP:

- *truyền dữ liệu không tin cậy* giữa tiến trình gửi và nhận
- *không cung cấp:* độ tin cậy, kiểm soát luồng, kiểm soát tắc nghẽn, định thời, đảm bảo thông lượng, bảo mật hoặc thiết lập kết nối.

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

Các ứng dụng Internet và các giao thức vận chuyển

ứng dụng	giao thức tầng ứng dụng	giao thức vận chuyển
file transfer/download	FTP [RFC 959]	TCP
e-mail	SMTP [RFC 5321]	TCP
Web documents	HTTP 1.1 [RFC 7320]	TCP
Internet telephony	SIP [RFC 3261], RTP [RFC 3550], or proprietary	TCP or UDP
streaming audio/video	HTTP [RFC 7320], DASH	TCP
interactive games	WOW, FPS (proprietary)	UDP or TCP

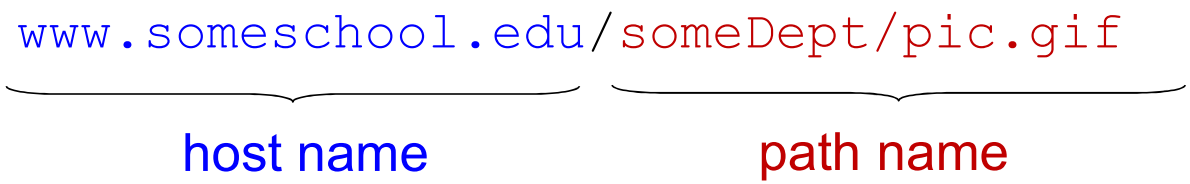
Application layer: overview

- Principles of network applications
- **Web and HTTP**
- E-mail, SMTP, IMAP
- The Domain Name System DNS

Web and HTTP

First, a quick review…

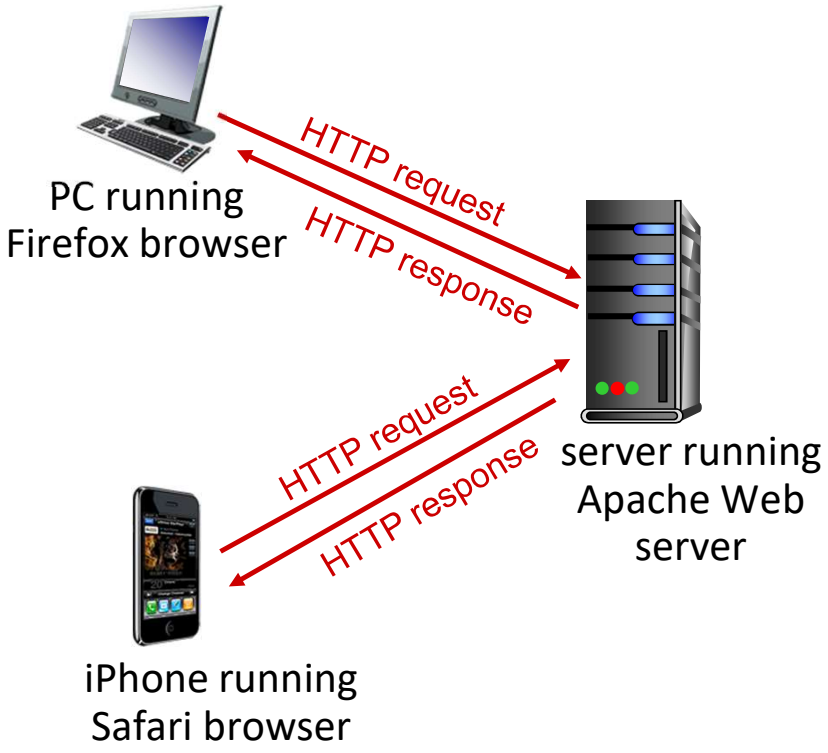
- web page bao gồm các objects, và có thể được lưu trữ trên nhiều Web servers khác nhau.
- object có thể là HTML file, JPEG image, Java applet, audio file, video, form… và được thể hiện trên cơ sở ngôn ngữ HTML qua địa chỉ gọi là URL



HTTP overview

HTTP: hypertext transfer protocol

- Một giao thức của dịch vụ Web, hoạt động ở tầng ứng dụng
- client/server model:
  - client: browser gửi yêu cầu, nhận, (sử dụng giao thức HTTP) và "displays" Web objects trên webbrowser.
  - server: Web server gửi (sử dụng giao thức HTTP) các objects trong response cho các yêu cầu từ clients



HTTP overview (continued)

HTTP uses TCP:

- client khởi tạo kết nối TCP (socket) đến server, port 80
- server chấp nhận kết nối TCP từ client
- HTTP messages: được browser (HTTP client) và Web server (HTTP server) sử dụng để giao tiếp với nhau.
- Đóng kết nối TCP

HTTP is "stateless"

- server không giữ lại các thông tin về các yêu cầu đã thực hiện từ các client.

aside

Nếu protocols thực hiện việc lưu lại các "state"! thì sẽ rất phức tạp

- Các trang thái kết nối trước đó cần được lưu lại
- Nếu server/client xảy ra sự cố, state 2 bên có thể có khác biệt. Thực hiện việc đồng bộ gặp khó khăn

HTTP connections: two types

Non-persistent HTTP (không liên tục)

1. Mở kết nối TCP
2. Tối đa một object được gửi qua kết nối TCP.
3. Đóng kết nối TCP.

Nếu client cần nhiều hơn một objects thì sẽ có nhiều yêu cầu kết nối đến server.

Persistent HTTP (liên tục)

- Mở kết nối TCP đến server
- Nhiều objects có thể được gửi qua một kết nối TCP connection giữa client và server
- Đóng kết nối TCP

Non-persistent HTTP: example

User enters URL: **`www.someSchool.edu/someDepartment/home.index`**
(containing text, references to 10 jpeg images)

1a. HTTP client initiates TCP connection to HTTP server (process) at www.someSchool.edu on port 80

1b. HTTP server at host www.someSchool.edu waiting for TCP connection at port 80 "accepts" connection, notifying client

2. HTTP client sends HTTP *request message* (containing URL) into TCP connection socket. Message indicates that client wants object someDepartment/home.index

3. HTTP server receives request message, forms *response message* containing requested object, and sends message into its socket

time

Non-persistent HTTP: example

User enters URL:

(containing text, references to 10 jpeg images)

4. HTTP server closes TCP connection.

5. HTTP client receives response message containing html file, displays html. Parsing html file, finds 10 referenced jpeg objects

6. Steps 1-5 repeated for each of 10 jpeg objects

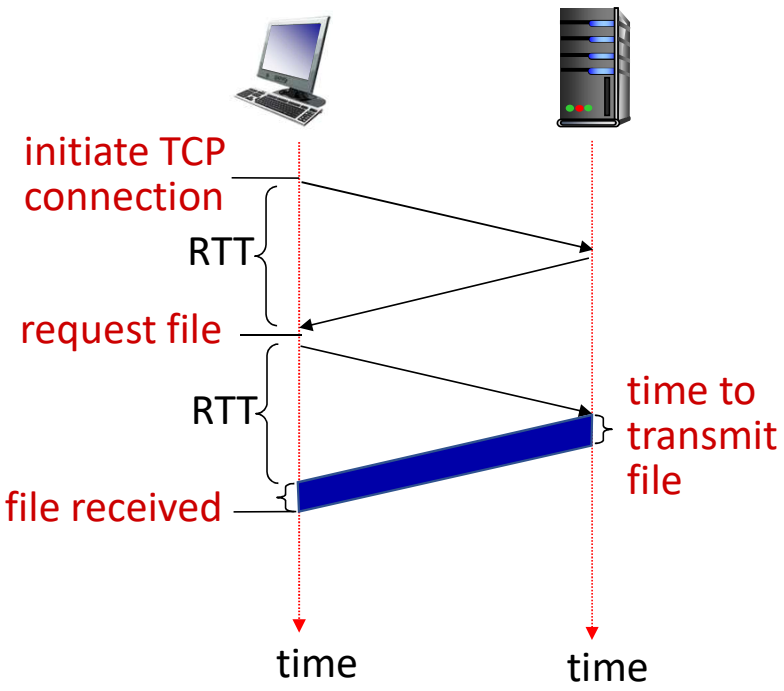
time

Non-persistent HTTP: response time

RTT (round trip time): thời gian của một packet được truyền tải từ client đến server và quay lại (hình).

HTTP response time (object/file):

- 1 RTT khở tạo kết nối TCP
- 1 RTT cho yêu cầu HTTP và phản hồi HTTP từ server.
- Là thời gian truyền tải một obect/file



Non-persistent HTTP response time = 2RTT+ file transmission time

Persistent HTTP (HTTP 1.1)

Non-persistent HTTP issues:

- Cần 2 RTTs cứ mỗi object
- OS cần nhiều tài nguyên cho các kết nối TCP
- browsers thường mở rất nhiều tiến trình kết nối TCP song song

Persistent HTTP (HTTP1.1):

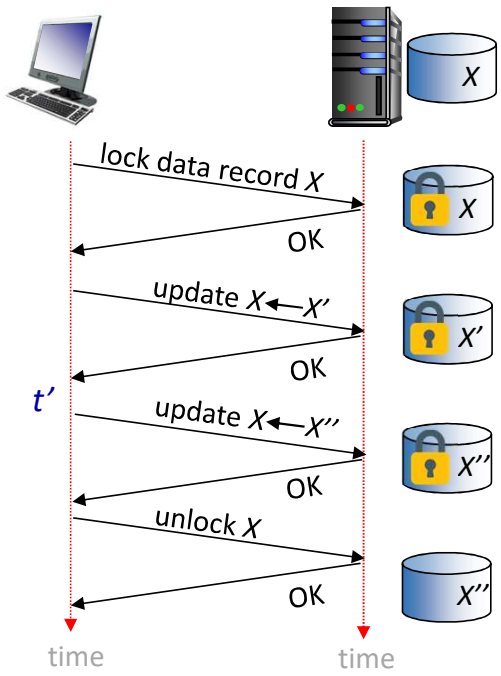
- server duy trì kết nối sau khi phản cho hồi các objects.
- Các giao tiếp HTTP messages giữa client/server tiếp tục được thực hiện bởi kết nối TCP trước đó
- client gửi requests liên tiếp cho các objects liên quan
- 1 RTT cho tất cả các objects liên quan (giảm response time còn một nửa)

Duy trì state giao tiếp giữa user/server: cookies

Recall: Các tương tác HTTP GET/response là *stateless*

- Các trao đổi bằng HTTP messages để hoàn thành một Web "transaction" không có khái niệm:
  - Client/server phải theo dõi "state" của các lần trao đổi.
  - Các HTTP requests phụ thuộc vào các phiên trao đổi trước
  - Client/server phải "recover" phần nội dung đã hoàn thành trước đó nhưng chưa đầy đủ nội dung.

a stateful protocol: client makes two changes to X, or none at all



Q: what happens if network connection or client crashes at *t'* ?

Duy trì state giao tiếp giữa user/server: cookies

Web sites và client browser sử *cookies* để duy trì một số state trong quá trình transactions

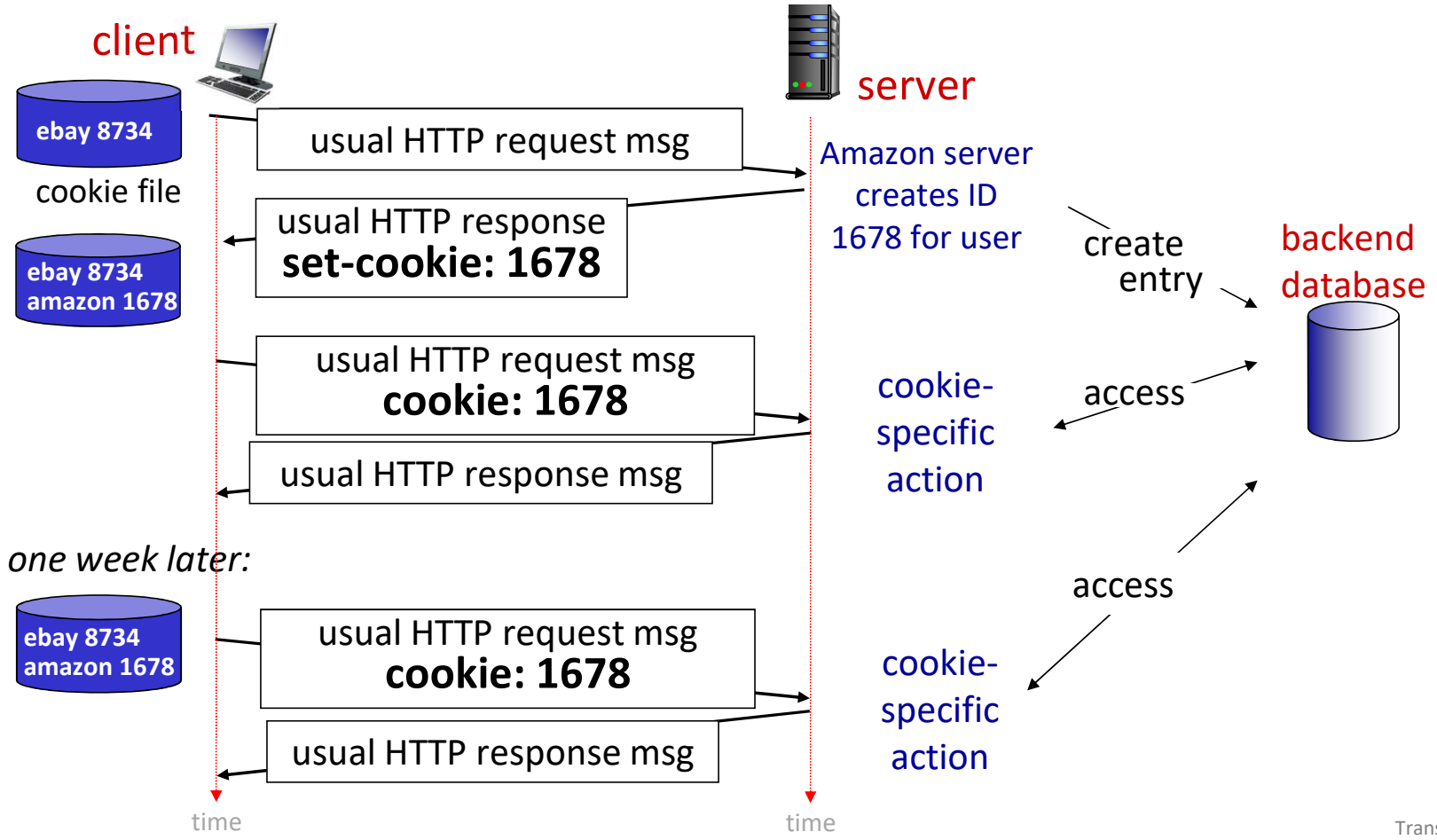
4 thành phần:

1) cookie header line của HTTP *response* message
2) cookie header line trong HTTP *request* message kế tiếp
3) cookie file được lưu trữ trong user's host, được quản lý bởi user's browser
4) back-end database tại Web site

Example:

- A sử dụng browser on laptop, truy cập đến 1 website trong lần đầu tiên
- Khi HTTP requests ban đầu đến máy chủ web, máy chủ tạo:
  - unique ID (aka "cookie")
  - Quyền truy cập vào backend database của ID
- Những HTTP requests kế tiếp từ A đến site này sẽ chứa cookie ID, cho phép máy chủ có thể "identify" A

Duy trì state giao tiếp giữa user/server: cookies



HTTP Cookies

What cookies can be used for:

- Authorization (Xác thực, chứng thực)
- shopping carts (Mua sắm, giỏ hàng)
- Recommendations (Gợi ý)
- user session state (Web e-mail)

Challenge: How to keep state:

- protocol endpoints: duy trì các state tại sender/receiver trên nhiều transactions
- cookies: Các trao đổi chứa thêm data của state trong HTTP messages

aside

cookies and privacy:

- cookies cho phép sites biết các thông tin của bạn khi truy cập đến site. VD: (Ads, gợi ý…)
- Các cookies cho phép định danh, theo dõi người dùng trên nhiều web sites khác nhau.

Application layer: overview

- Principles of network applications
- Web and HTTP
- E-mail, SMTP, IMAP
- The Domain Name System DNS

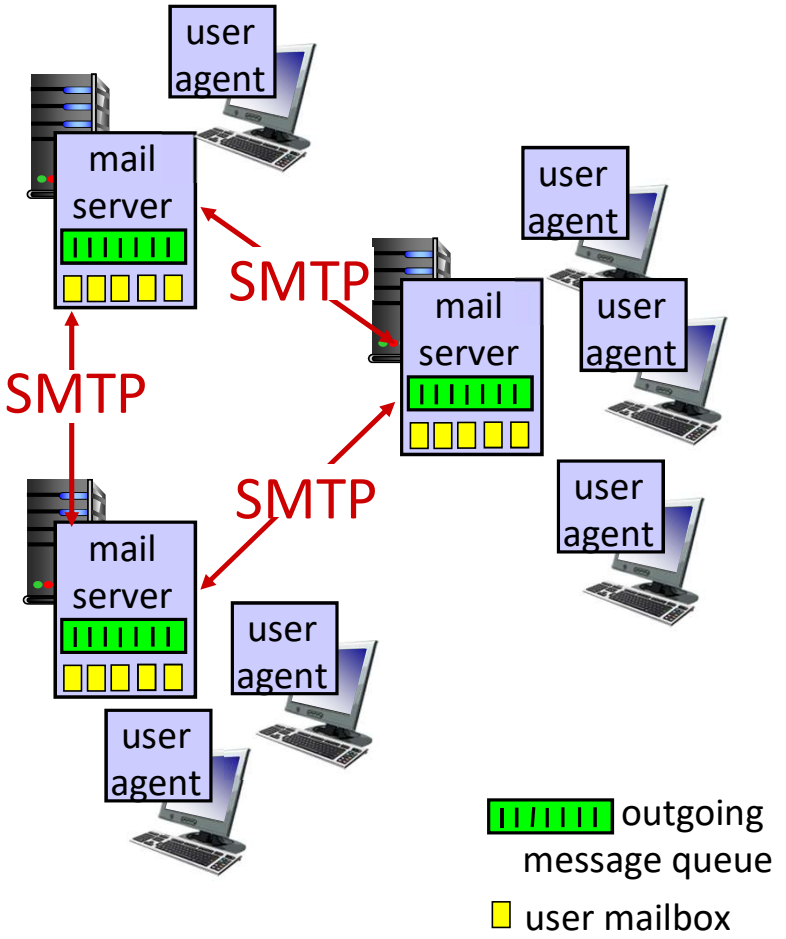
E-mail

Three major components:

- user agents (Người dùng)
- mail servers (Nhà cung cấp dịch vụ mail)
- simple mail transfer protocol: SMTP (các giao thức)

User Agent

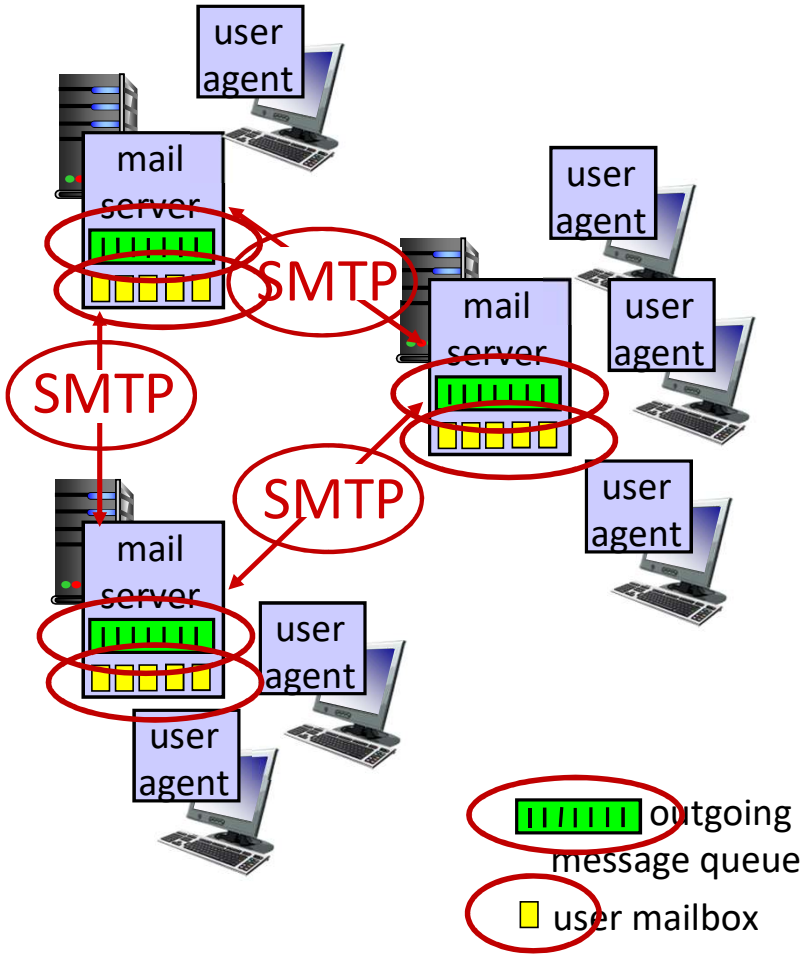
- Người sử dụng mail
- Nhận mail, soạn thảo mail…
- Thông qua sử dụng các ứng dụng mail client Outlook, Mail, Webmail
- Mail gửi và nhận mail được lưu trên server



E-mail: mail servers

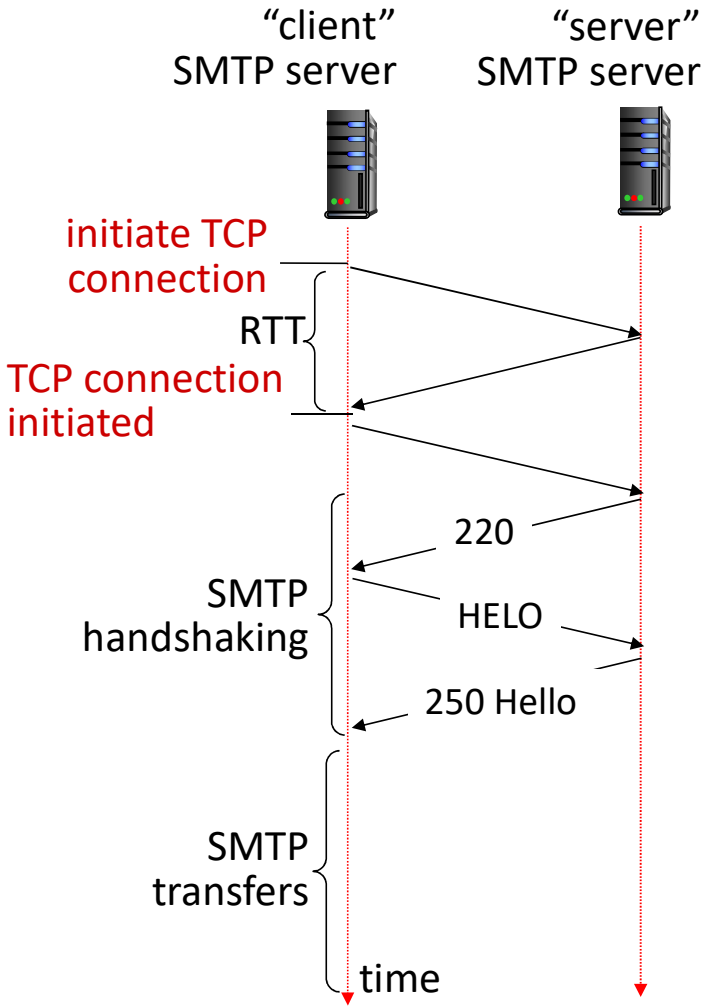
mail servers:

- *mailbox* chứa các thư đến của user
- *message queue* Chứa các thư sẽ được gửi đi (theo thứ tự)
- *SMTP protocol* giao thức giúp mail server có thể giao tiếp với client/server và ngược lại.
  - client: gửi mail đến server
  - "server": nhận mail và chuyển tiếp đến client nhận



SMTP RFC (5321)

- Sử dụng TCP để đảm bảo độ tin cậy cho việc truyền tải email (client thực hiện kết nối đến mail server port 25)
- Chuyển phát trực tiếp: sending server đến receiving server (Người dùng gửi mail có địa chỉ @gmail.com sang mail @yahoo.com, Gmail server gửi trực tiếp thư đến yahoo mail)
- 3 quá trình của sự truyền tải
  - handshaking (tạo kết nối)
  - transfer of messages (truyền tải)
  - Closure (Đóng kết nối)
- Có các command/response tương tự như HTTP
  - commands: ASCII text
  - response: status code and phrase



Scenario: Alice sends e-mail to Bob

1) Alice sử dụng Yahoo maill để soạn mail gửi đến Bob với địa chỉ bob@ntu.edu.vn

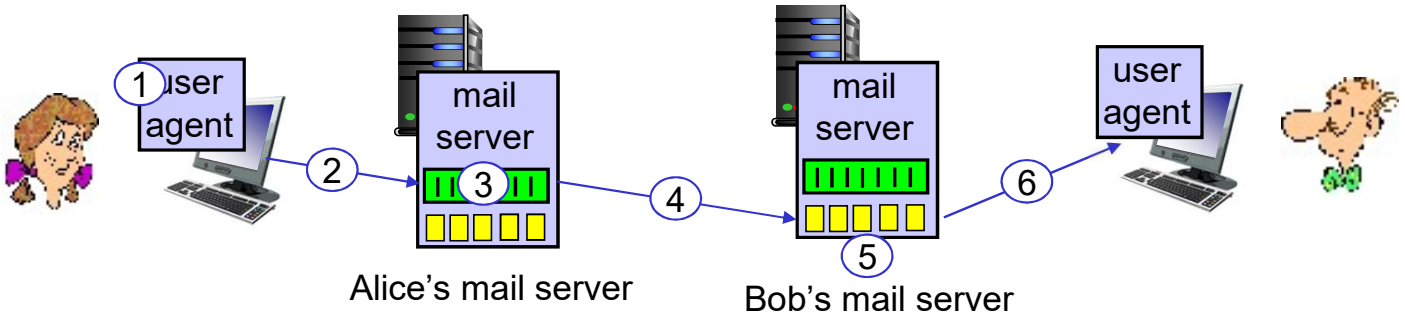
2) Alice gửi thư đến Yahoo mail server; thư được lưu trữ trong message queue và sẵn sàng gửi đi

3) Yahoo mail server sử dụng SMTP mở kết nối TCP với Bob's mail server (ntu mail)

4) SMTP gửi thư của Alice thông qua kết nối TCP đến server mail của bob

5) Bob's mail server nhận thư, và chuyển vào Bob's mailbox (thư đến)

6) Bob xác thực quyền của mình và sử dụng các mail client để đọc thư



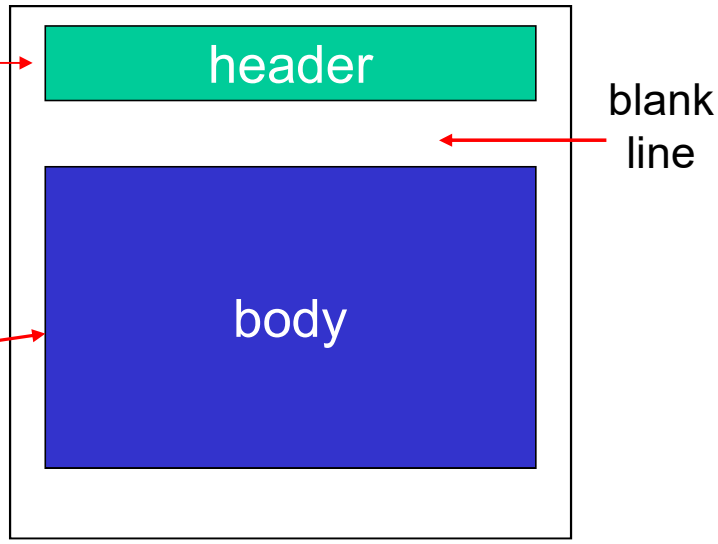
Mail message format

SMTP: là một giao thức hỗ trợ việc trao đổi e-mail, được định nghĩa tại RFC 531 (tương tự như HTTP)

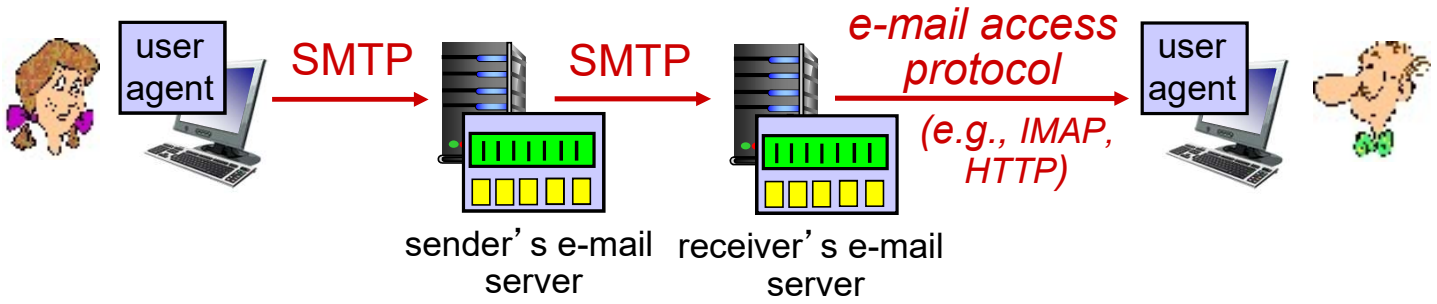
RFC 822 định nghĩa *syntax* cho e-mail message (tương tự HTML)

- header lines, e.g.,
  - To:
  - From:
  - Subject:

  these lines, within the body of the email message area different from SMTP MAIL FROM:, RCPT TO: commands!
- Body: the "message", ASCII characters only



Retrieving email: mail access protocols



- SMTP: delivery/storage e-mail messages đến receiver's server
- mail access protocol: giúp client tải các massage từ server về mail client
  - IMAP: Internet Mail Access Protocol [RFC 3501]: messages được lưu trữ trên server, IMAP cung cấp khả năng retrieval, deletion, phân loại messages trên server
- HTTP: gmail, Hotmail, Yahoo!Mail, etc. cung cấp mail client trên nền tảng/giao diện web-based sử dụng STMP (to send), IMAP (or POP) để retrieve e-mail messages

Application Layer: Overview

- Principles of network applications
- Web and HTTP
- E-mail, SMTP, IMAP
- The Domain Name System DNS

DNS: Domain Name System

people: ID Card, Tên, passport để định danh.

Internet hosts, routers:

- IP address (32 bit) - Sử dụng để phân biệt nhau
- Sử dụng các tên miền (vd ntu.edu.vn) để con người dễ nhận biết, dễ nhớ.

Q: how to map between IP address and name, and vice versa?

Domain Name System:

- *distributed database* được tổ chức theo cấu trúc phân cấp của nhiều *name servers (Why? Q)*
  - Xảy ra lỗi (nếu chỉ có 1 máy chủ)
  - Phục vụ tốt hơn (Load Balancing)
  - Khoảng cách địa lý -> Tốc độ phản hồi
- *application-layer protocol:* là một hosts, các name servers giao tiếp để *resolve* names (IP address/Tên)
  - *note:* core Internet function, implemented as application-layer protocol

DNS: services, structure

DNS services:

- Chuyển đổi hostname sang IP address các “bí danh” – sinhvien.ntu.edu.vn
- Mail server aliasing
- Load distribution
  - Web servers cân bằng tải: Nhiều IP addresses đại diện cho một name

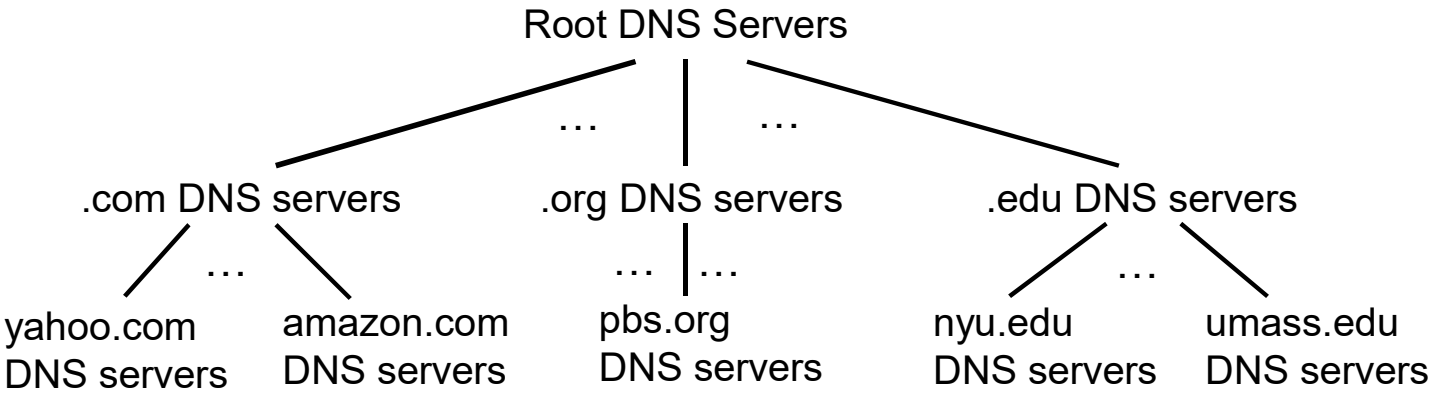
Q: Why not centralize DNS?

- single point of failure
- traffic volume
- distant centralized database
- maintenance

A: doesn‘t scale!

- Comcast DNS servers alone: Có đến 600 tỷ DNS queries trong 1 ngày đến một máy chủ DNS
- Akamai DNS servers alone: 2.2T DNS queries/day

DNS: a distributed, hierarchical database



Root

Top Level Domain

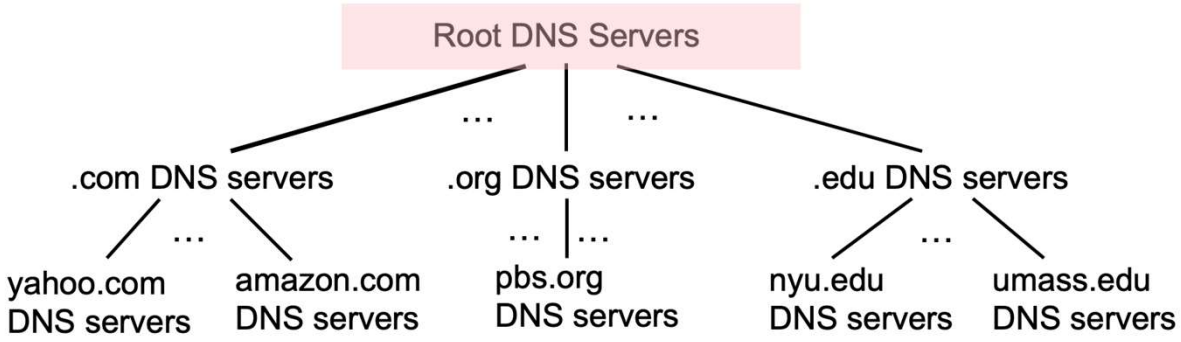
Authoritative

Ví dụ người dùng muốn truy cập đến "ntu.edu.vn":

- client hỏi root server để tìm .vn DNS server
- client hỏi .vn DNS server để tìm đến .edu DNS server
- client hỏi .edu DNS server để tìm đến ntu DNS server và sau đó hỏi ntu.edu.vn DNS server để lấy IP address -> truy cập bằng địa chỉ IP

DNS: root name servers

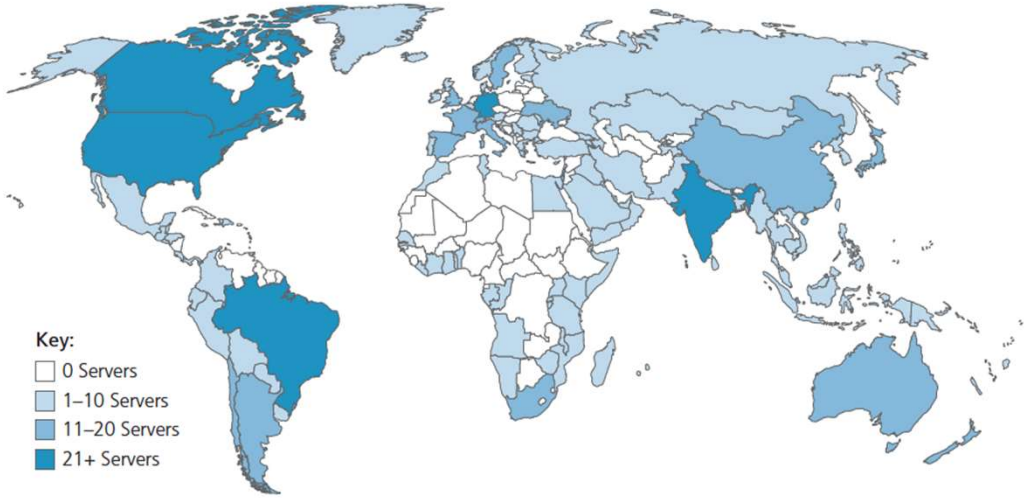
- Đảm nhiệm việc kết nối đến các name servers, để chỉ nơi có thể tìm ra bản ghi phân giải (không phản hồi bản ghi cho client)



DNS: root name servers

- Đảm nhiệm việc kết nối đến các name servers, để chỉ nơi có thể tìm ra bản ghi phân giải (không phản hồi bản ghi cho client)
- *Là một thành phần quan trọng*
  - Mạng Internet có thể không hoặc động được nếu thiếu!
  - DNSSEC – cung cấp khả năng security (Xác thực và đảm bảo cho các thông điệp trao đổi giữa các thành phần trong dịch vụ DNS)
- ICANN (*Internet Corporation for Assigned Names and Numbers*) quản lý Root DNS domain

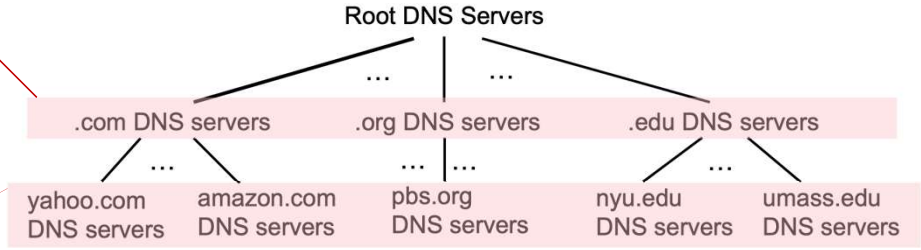
13 logical root name "servers" worldwide each "server" replicated many times (~200 servers in US)



Top-Level Domain, and authoritative servers

Top-Level Domain (TLD) servers:

- Đảm nhiều cho các tên miền .com, .org, .net, .edu…, và các tên miền cho quốc gia .cn, .uk, .fr, .ca, .jp, .vn…
- Các TLD mang ý nghĩa riêng theo lĩnh vực của đối tượng sở hữu tên miền đang hoạt động (ví dụ .com -> commercial, edu -> education)



Authoritative DNS servers:

- Tổ chức sở hữu DNS server(s), cung cấp, xác thực Tên miền sang địa IP mà tổ chức sở hữu (đăng ký từ nhà cung cấp dịch vụ mạng ISP)
- Có thể được duy trì bởi tổ chức hoặc nhà cung cấp dịch vụ (đơn vị bán tên miền).

Local DNS name servers

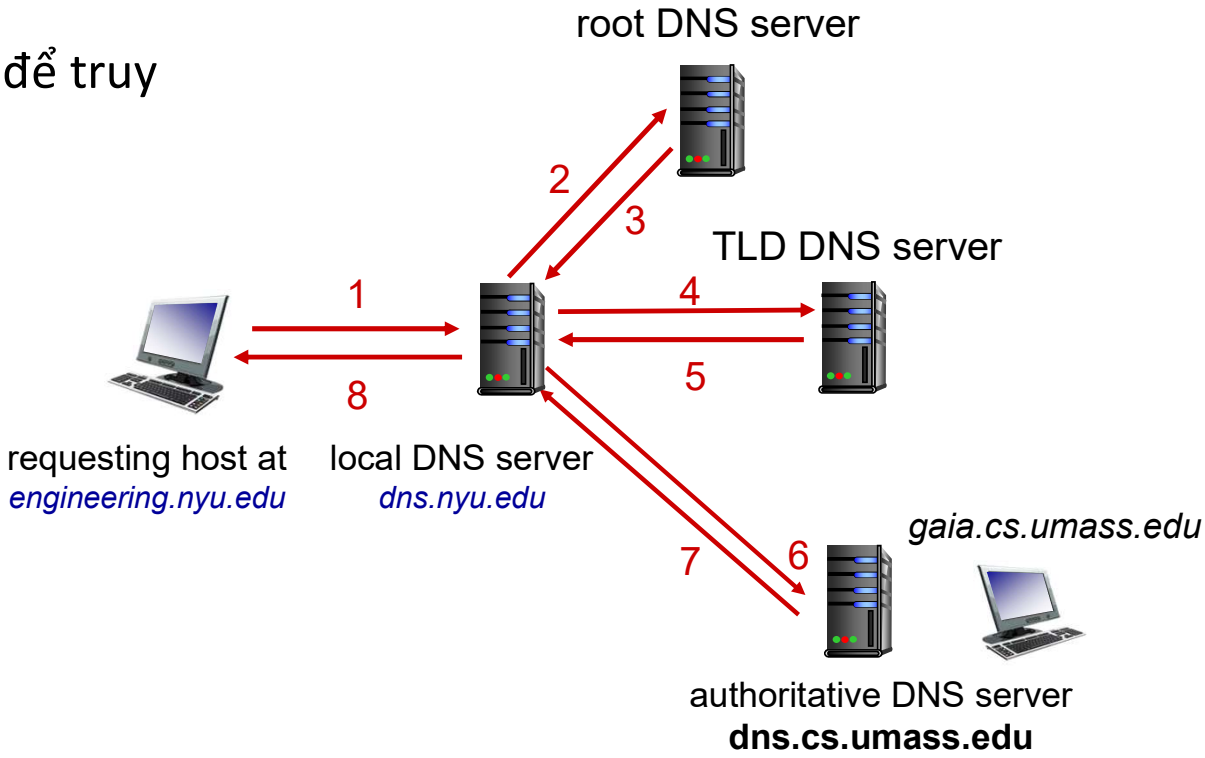
- Khi host request một DNS query, nó được gởi tới DNS server cục bộ (Local DNS server)
  - Local DNS server returns reply, answering:
    - Có local cache của các bản ghi name-to-address (có thể out-of-day)
    - Hoạt động giống như 1 proxy, chuyển tiếp các query đến DNS toàn cầu.
  - Mỗi một ISP có một local DNS name server; có thể tìm thấy:
    - MacOS: `% scutil --dns`
    - Windows: `>ipconfig /all`
- Local DNS server không thuộc vào hệ thống DNS toàn cầu

DNS name resolution: iterated query

Example: Alice cần có IP address để truy cập đến ntu.edu.vn (gửi 1)

Iterated query:

- Trả về thông tin tốt nhất, có thể là IP address cần hoặc thông tin của máy chủ DNS kế tiếp cho truy vấn tìm kiếm
- "I don't know this name, but ask this server"

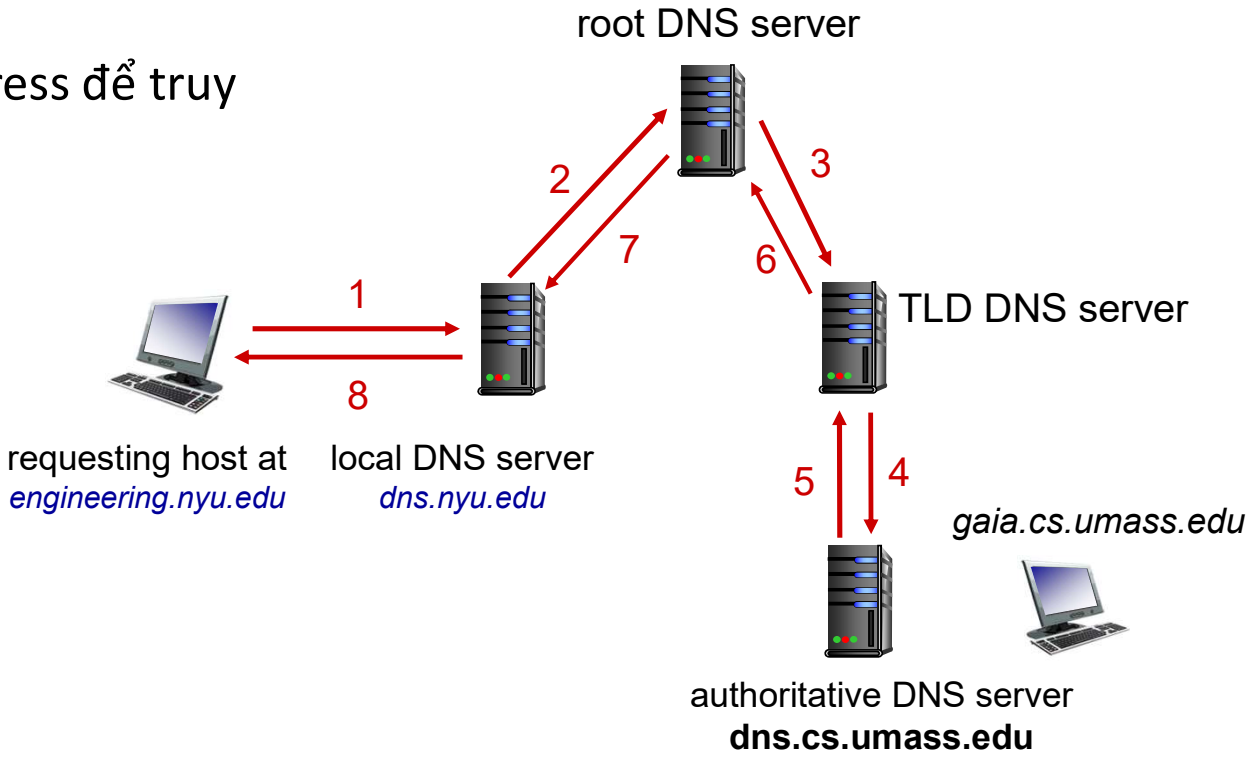


DNS name resolution: recursive query

Example: Alice cần có IP address để truy cập đến ntu.edu.vn (gửi 1)

Recursive query:

- Trả về thông tin "IP address cần" từ server nhận được truy vấn
- Sẽ gây quá tải cho cấu trúc phân cấp mức cao?



Caching DNS Information

- Các máy chủ DNS có bản *caches* lưu lại những thông tin DNS ở những lần truy vấn trước
  - cache sẽ mất đi sau một khoảng thời gian cụ thể (TTL)
  - Các TLD servers thông thường được lưu tại local name servers
    - Vì thế giúp các root name server không bị quá tải
- cached có thể *out-of-date*
  - Nếu máy chủ name server thay đổi IP address, các máy khác trên internet có thể không biết cho đến khi hết khoảng thời gian TTL!
- Hoạt động trên chuẩn của IETF được quy định tại RFC 2136

DNS records

DNS: Là các bản ghi lưu trữ thông tin cho việc phản hồi đến client
resource records (RR)

RR format: (`name, value, type, ttl`)

type=A

- `name` is hostname
- `value` is IP address

type=NS

- `name` is domain (e.g., foo.com)
- `value` is hostname of authoritative name server for this domain

type=CNAME

- `name` is alias name
- www.ntu.edu.vn tên thật ntu.edu.vn
- `value` is canonical name

type=MX

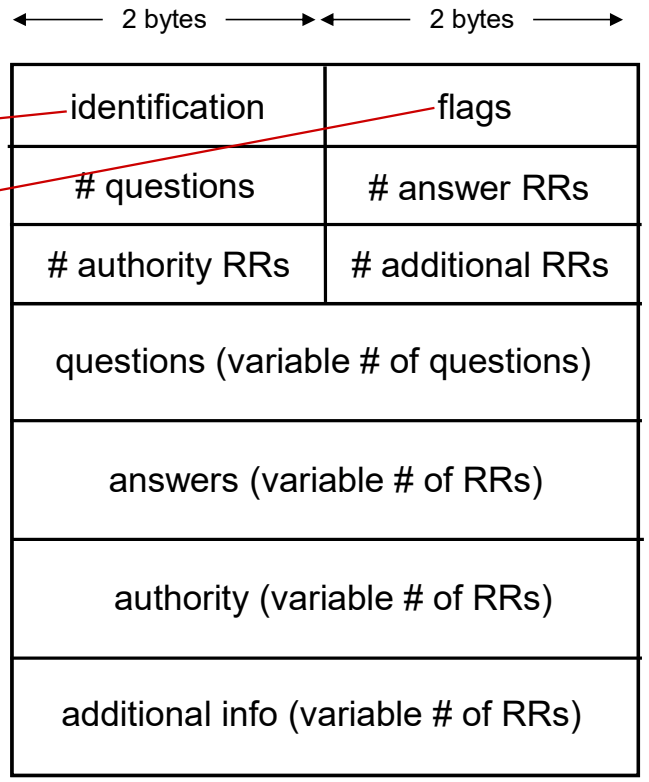
- `value` is name of mailserver associated with `name`

DNS protocol messages

DNS *query* and *reply* messages, có chung 1 *format:*

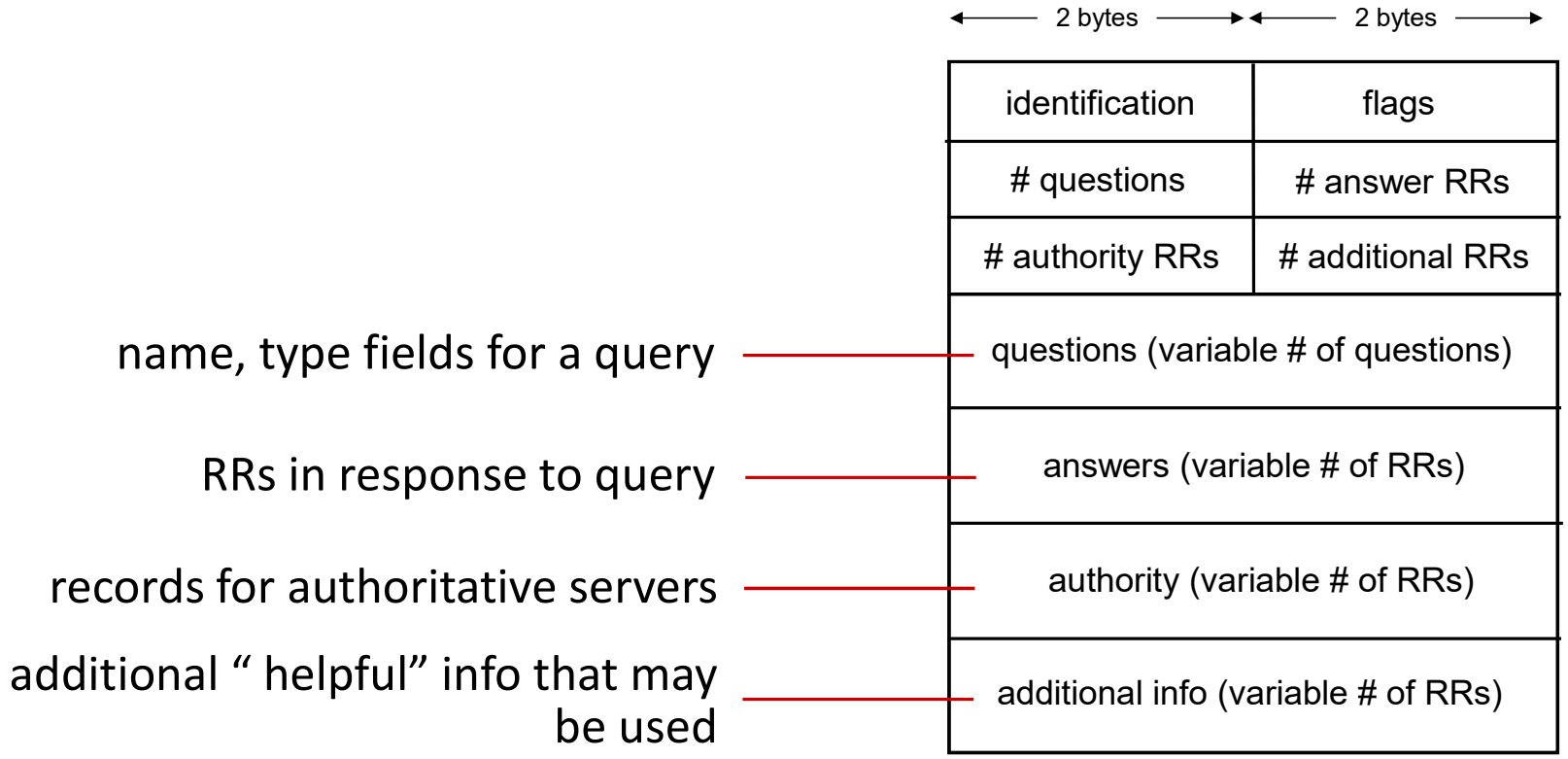
message header:

- identification: 16 bit # for query, reply to query uses same #
- flags:
  - query or reply
  - recursion desired
  - recursion available
  - reply is authoritative



DNS protocol messages

DNS *query* and *reply* messages, có chung 1 *format:*



Getting your info into the DNS

example: new startup “Network Utopia”

- register name networkuptopia.com at *DNS registrar* (e.g., Network Solutions)
  - provide names, IP addresses of authoritative name server (primary and secondary)
  - registrar inserts NS, A RRs into .com TLD server:
    `(networkutopia.com, dns1.networkutopia.com, NS)`
    `(dns1.networkutopia.com, 212.212.212.1, A)`
- create authoritative server locally with IP address `212.212.212.1`
  - type A record for www.networkuptopia.com
  - type MX record for networkutopia.com

DNS security

DDoS attacks

- bombard root servers with traffic
  - not successful to date
  - traffic filtering
  - local DNS servers cache IPs of TLD servers, allowing root server bypass
- bombard TLD servers
  - potentially more dangerous

Spoofing attacks

- intercept DNS queries, returning bogus replies
  - DNS cache poisoning
  - RFC 4033: DNSSEC authentication services