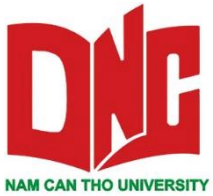
BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC NAM CẦN THƠ



MẠNG MÁY TÍNH

Chương 2:

TẦNG ỨNG DỤNG – APPLICATION LAYER

Giảng viên: ThS. Nguyễn Minh Triết

Tầng ứng dụng

2.1. Các nguyên tắc của ứng dụng mạng

2.2. Web và HTTP

2.3. FTP

2.4. Email

2.5. DNS

2.6. Ứng dụng P2P

2.7. Lập trình Socket với TCP

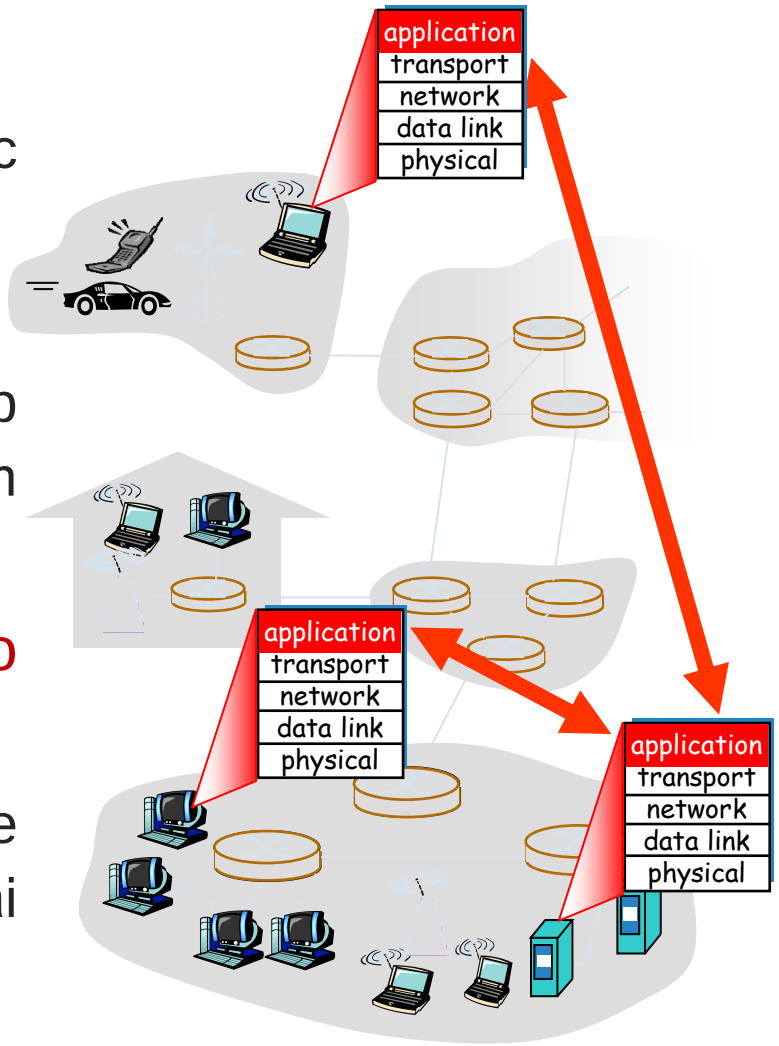
2.8. Lập trình Socket với UDP

Một số ứng dụng mạng

- e-mail
- web
- instant messaging
- remote login
- P2P file sharing
- multi-user network games
- streaming stored video clips (YouTube, NetFlix…)
- voice over IP
- real-time video conferencing
- grid computing

Xây dựng một ứng dụng mạng

- Viết chương trình
  - Chạy trên các end system khác nhau
  - Giao tiếp qua mạng
  - Ví dụ Web: phần mềm Web server giao tiếp với phần mềm Web browser
- Không phải phần mềm viết cho các thiết bị trong Network core
  - Các thiết bị trong Network core không thực hiện chức năng tại lớp ứng dụng



Tầng ứng dụng

2.1. **Các nguyên tắc của ứng dụng mạng**

2.2. Web và HTTP

2.3. FTP

2.4. Email

2.5. DNS

2.6. Ứng dụng P2P

2.7. Lập trình Socket với TCP

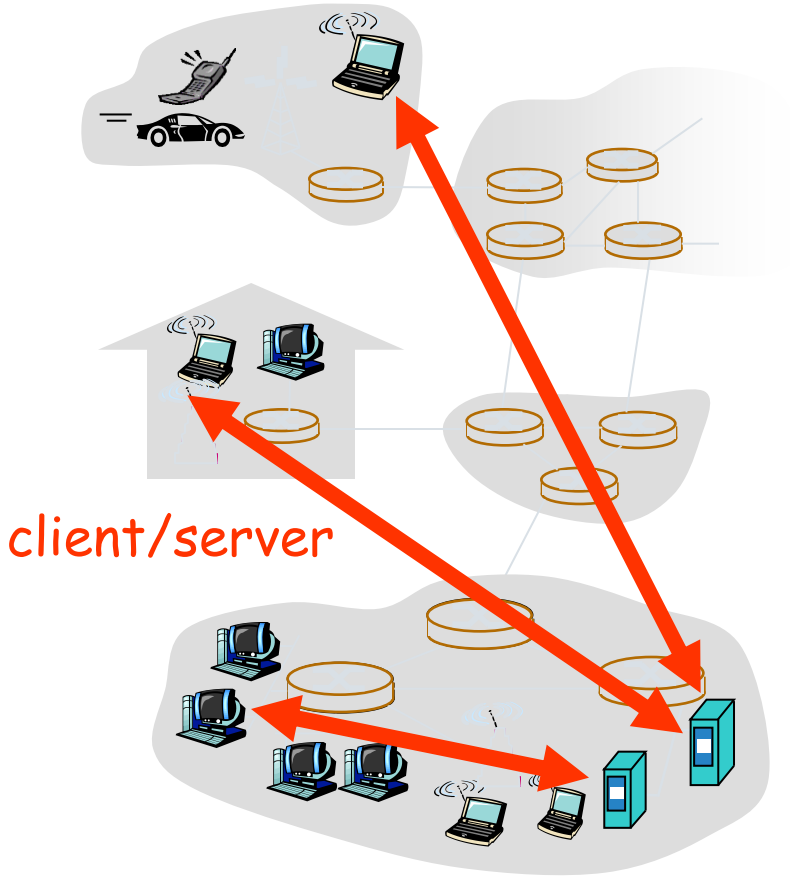
2.8. Lập trình Socket với UDP

Các nguyên tắc của ứng dụng mạng

- Các kiến trúc ứng dụng
  - Client-Server
  - Peer-to-Peer (P2P)
  - Hybrid of Client-Server and P2P

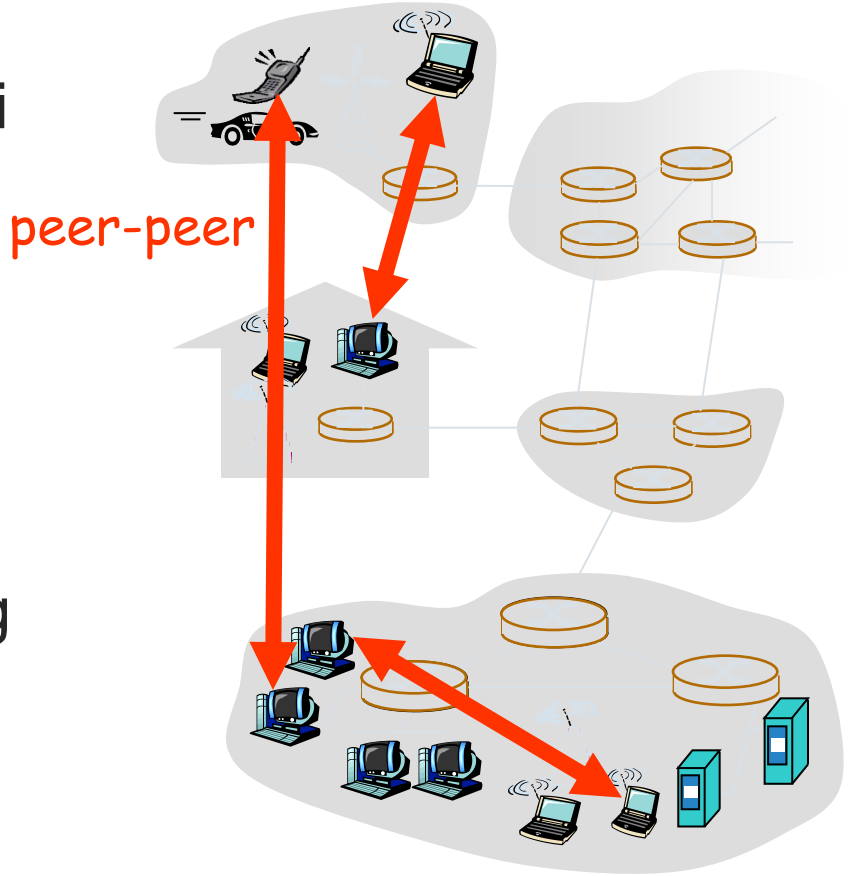
Các nguyên tắc của ứng dụng mạng (tt)

- Kiến trúc Client-Server
  - **Server**
    - Luôn ở trạng thái hoạt động
    - Địa chỉ IP cố định
    - Có khả năng mở rộng
  - **Client**
    - Giao tiếp với server
    - Có thể không kết nối liên tục
    - Có thể có địa chỉ IP động
    - Không giao tiếp trực tiếp với nhau



Các nguyên tắc của ứng dụng mạng (tt)

- Kiến trúc P2P
  - Server không ở trạng thái luôn hoạt động
  - Hệ thống cuối giao tiếp trực tiếp với nhau
  - Các Peer kết nối không liên tục và địa chỉ IP động
  - Dễ co giãn nhưng khó quản lý



Các nguyên tắc của ứng dụng mạng (tt)

- Hybrid of Client-Server and P2P
  - Instant messaging / Skype
    - Ứng dụng VoIP P2P
    - Server tập trung: quản lý trạng thái, địa chỉ client đăng nhập vào
    - Client – client kết nối trực tiếp

Các nguyên tắc của ứng dụng mạng (tt)

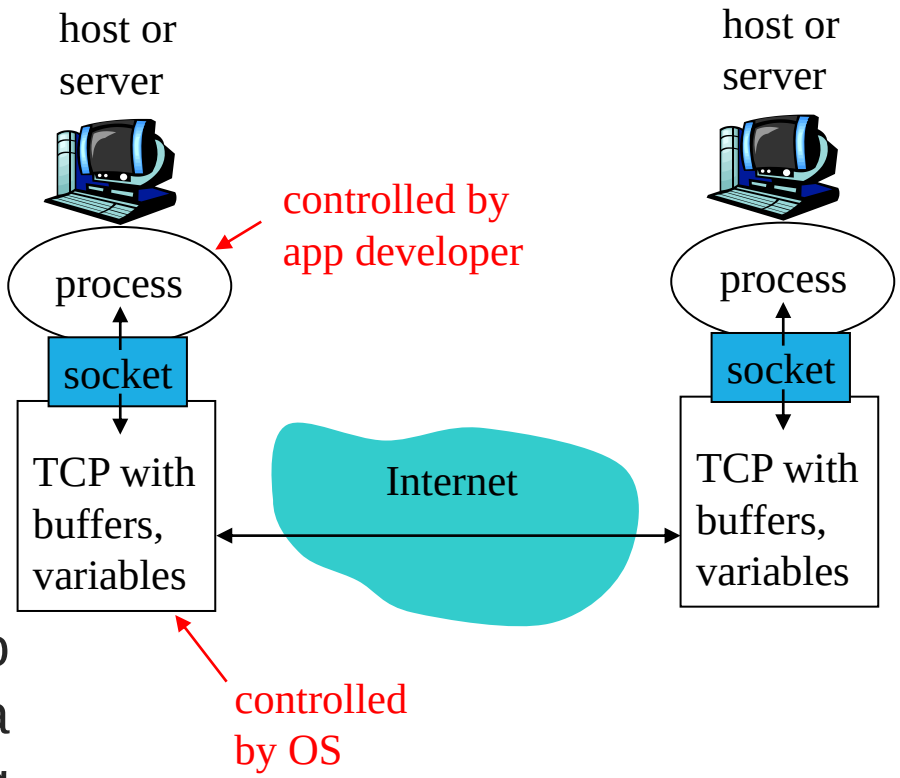
- Giao tiếp giữa các tiến trình
  - **Tiến trình (process):** chương trình chạy trên một host
  - Trong cùng host, 2 tiến trình giao tiếp sử dụng giao tiếp liên quá trình (inter-process communication), do hệ điều hành định nghĩa
  - Các tiến trình trên các host khác nhau giao tiếp bằng cách trao đổi các thông điệp (message)

Tiến trình client: tiến trình khởi đầu quá trình giao tiếp

Tiến trình server: tiến trình đợi kết nối

Các nguyên tắc của ứng dụng mạng (tt)

- Socket
  - Tiến trình gửi nhận message tới/từ socket của nó
  - Socket tương tự cửa ra vào
    - Tiến trình gửi đẩy message ra ngoài cửa
    - Tiến trình gửi dựa vào hạ tầng transport phía bên kia của cửa, mang message tới socket của tiến trình nhận



Các nguyên tắc của ứng dụng mạng (tt)

Định danh tiến trình

- Để nhận được thông điệp, tiến trình phải được định danh (identifier)
- Một host có một địa chỉ IP duy nhất
- *Câu hỏi:* có thể dùng địa chỉ IP của host chứa tiến trình để định danh tiến trình?
  - *Trả lời:* không, nhiều tiến trình có thể chạy trên cùng host

Các nguyên tắc của ứng dụng mạng (tt)

- Định danh tiến trình bao gồm cả địa chỉ IP và số hiệu cổng (port number) của tiến trình trên host
- Ví dụ: giá trị cổng của một số ứng dụng:
  - HTTP server: 80
  - Mail server: 25
- Để gửi thông điệp HTTP đến web server *nctu.edu.vn* cần có
  - Địa chỉ IP: 210.211.108.153
  - Số hiệu cổng: 80

Các nguyên tắc của ứng dụng mạng (tt)

- Giao thức tầng ứng dụng định nghĩa
  - Kiểu của thông điệp trao đổi
    - Ví dụ: thông điệp yêu cầu (request), thông điệp trả lời (response)
  - Cú pháp của kiểu thông điệp
    - Các trường trong thông điệp và mô tả các trường trong thông điệp
  - Ngữ nghĩa của các trường
  - Quy tắc các tiến trình gửi/nhận thông điệp (khi nào và như thế nào)

Các nguyên tắc của ứng dụng mạng (tt)

- Giao thức công khai (Public-domain protocols)
  - Được định nghĩa trong RFC
  - Ví dụ: HTTP, SMTP, …
- Giao thức riêng (Proprietary protocols)
  - Ví dụ: Skype

Các nguyên tắc của ứng dụng mạng (tt)

- Các dịch vụ Transport mà ứng dụng cần
  - **Mất mát dữ liệu (Data loss)**
    - Một số ứng dụng (audio,…) có thể chấp nhận một tỷ lệ mất dữ liệu nào đó
    - Một số ứng dụng khác (truyền file, telnet,…) đòi hỏi 100% dữ liệu truyền là tin cậy
  - **Độ trễ (Timing)**
    - Một số ứng dụng (điện thoại Internet, trò chơi tương tác,…) đòi hỏi độ trễ thấp
  - **Băng thông (Throughput)**
    - Một số ứng dụng (đa phương tiện,…) yêu cầu lượng băng thông tối thiểu
    - Một số ứng dụng khác sử dụng theo băng thông chúng nhận được
  - **Bảo mật (Security)**
    - Mã hóa, toàn vẹn dữ liệu, …

Các nguyên tắc của ứng dụng mạng (tt)

- Yêu cầu của một số ứng dụng phổ biến

Application	Data loss	Throughput	Time Sensitive
file transfer	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, 100’s msec
stored audio/video	loss-tolerant	same as above	yes, few secs
interactive games	loss-tolerant	few kbps up	yes, 100’s msec
instant messaging	no loss	elastic	yes and no

Các nguyên tắc của ứng dụng mạng (tt)

Các dịch vụ giao thức Transport

- **Dịch vụ TCP**
  - Hướng kết nối: yêu cầu quá trình thiết lập giữa tiến trình client và tiến trình server
  - Truyền tin cậy giữa tiến trình gửi và tiến trình nhận
  - Điều khiển luồng: bên gửi sẽ không vượt quá khả năng bên nhận
  - Điều khiển tắc nghẽn: điều chỉnh bên gửi khi mạng quá tải
  - Không cung cấp: độ trễ, đảm bảo băng thông tối thiểu, bảo mật
- **Dịch vụ UDP**
  - Truyền dữ liệu không tin cậy giữa tiến trình gửi và tiến trình nhận
  - Không cung cấp: việc thiết lập kết, truyền tin cậy, điều khiển luồng, điều khiển tắc nghẽn, độ trễ, đảm bảo băng thông tối thiểu, bảo mật

Các nguyên tắc của ứng dụng mạng (tt)

- Một số ứng dụng trên Internet

Application	Application layer protocol	Underlying transport protocol
e-mail	SMTP [RFC 2821]	TCP
remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	HTTP (eg Youtube), RTP [RFC 1889]	TCP or UDP
Internet telephony	SIP, RTP, proprietary (e.g., Skype)	typically UDP

Tầng ứng dụng

2.1. Các nguyên tắc của ứng dụng mạng

2.2. **Web và HTTP**

2.3. FTP

2.4. Email

2.5. DNS

2.6. Ứng dụng P2P

2.7. Lập trình Socket với TCP

2.8. Lập trình Socket với UDP

Web và HTTP

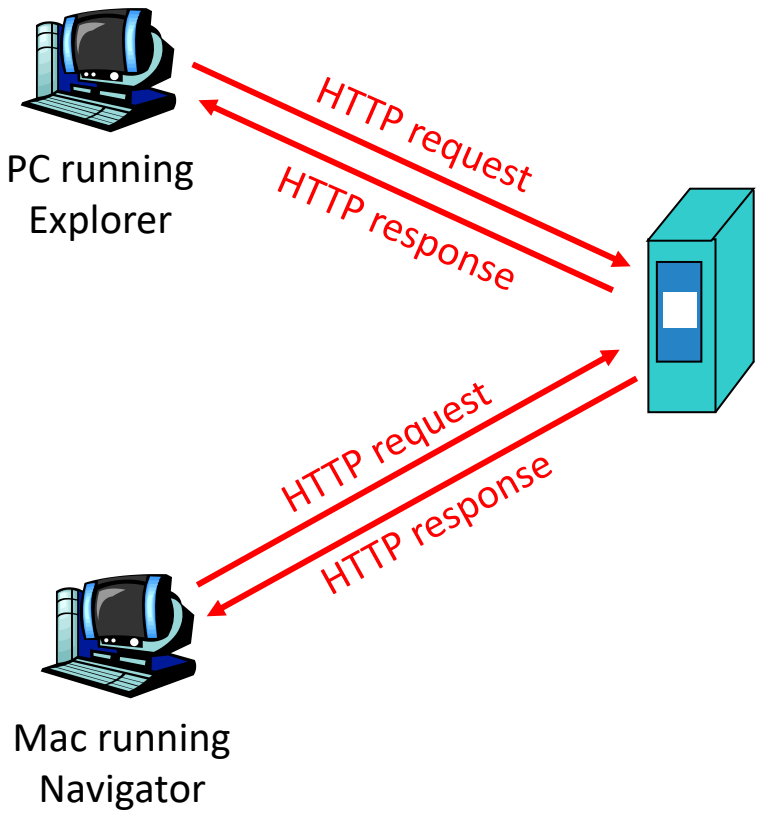
- Trang Web (web page) chứa các đối tượng (object)
- Đối tượng có thể là file HTML, ảnh JPEG, Java applet, audio,…
- Trang Web chứa file HTML, các file này có tham chiếu đến đối tượng khác
- Mỗi đối tượng được đánh địa chỉ bởi một URL
- Ví dụ URL:

`nctu.edu.vn/uploads/pic.gif`

host name path name

Web và HTTP (tt)

- HTTP: HyperText Transfer Protocol
  - Giao thức tầng ứng dụng của Web
  - Mô hình Client/Server
    - Client: trình duyệt yêu cầu, nhận và hiện thị các đối tượng
    - Server: Web server gửi các đối tượng trong thông điệp trả lời



Web và HTTP (tt)

- Sử dụng TCP
  - Client khởi đầu kết nối TCP (tạo socket) tới server, cổng 80
  - Server chấp nhận kết nối TCP từ client
  - Các thông điệp HTTP (thông điệp của giao thức tầng ứng dụng Web) trao đổi giữa trình duyệt (HTTP client) và Web server (HTTP server)
  - Kết nối TCP đóng

HTTP là giao thức không hướng trạng thái

Server không duy trì thông tin về các yêu cầu của client trong quá khứ

Các giao thức hướng trạng thái phức tạp hơn giao thức không hướng trạng thái

- Quá khứ (trạng thái) phải được duy trì
- Nếu server/client lỗi, các trạng thái có thể không thống nhất

Web và HTTP (tt)

- Kết nối HTTP
  - **Nonpersistent HTTP**

    Một đối tượng được gửi qua một kết nối TCP giữa client và server
  - **Persistent HTTP**

    Nhiều đối tượng có thể gửi qua một kết nối TCP giữa client và server

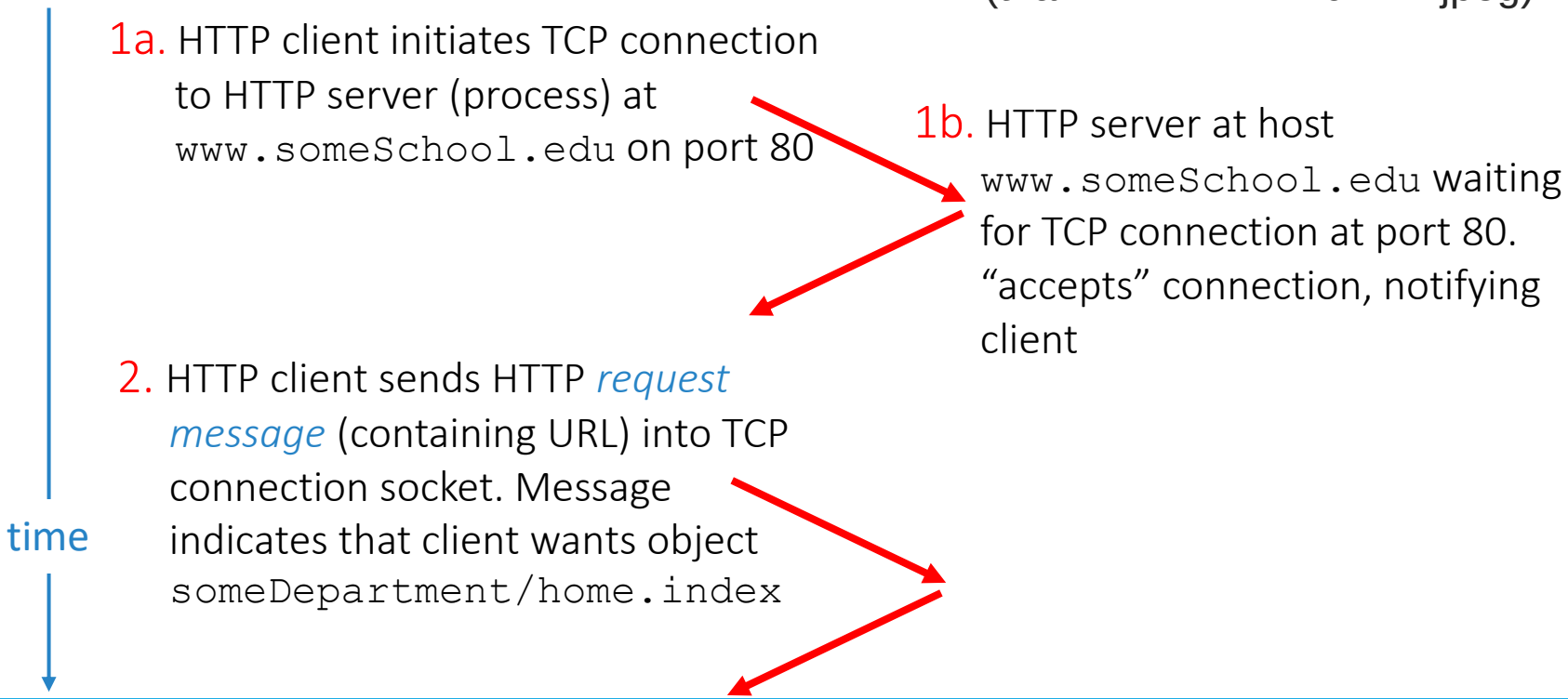
Web và HTTP (tt)

- Nonpersistent HTTP

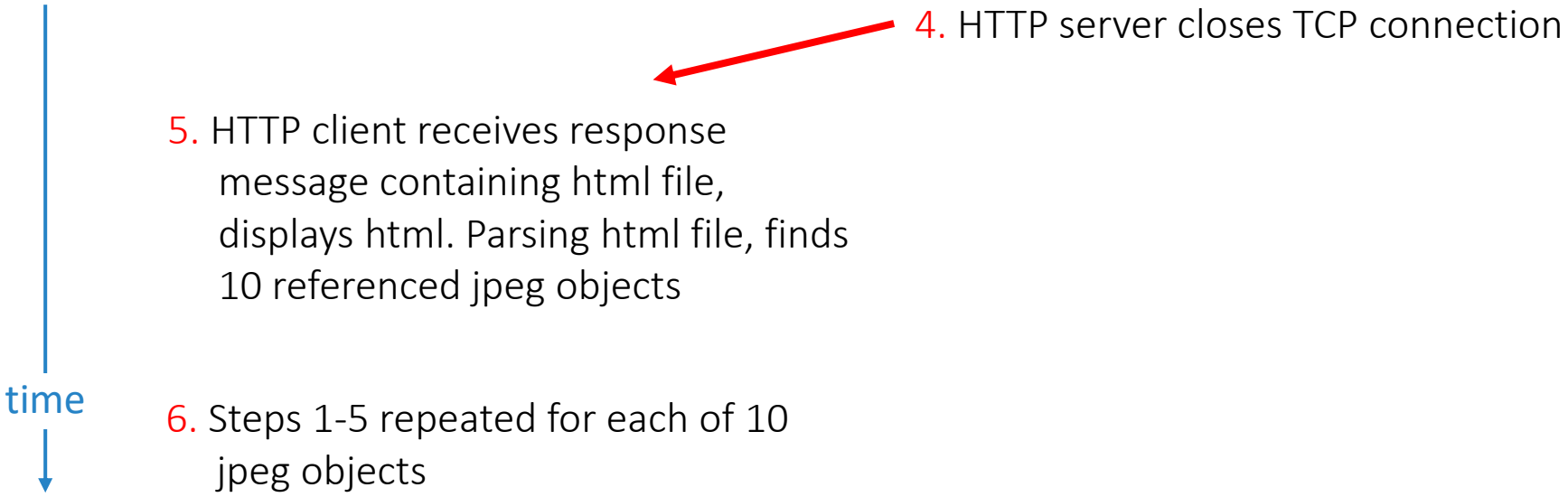
Giả sử người sử dụng truy cập URL

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

(tham chiếu đến 10 ảnh jpeg)

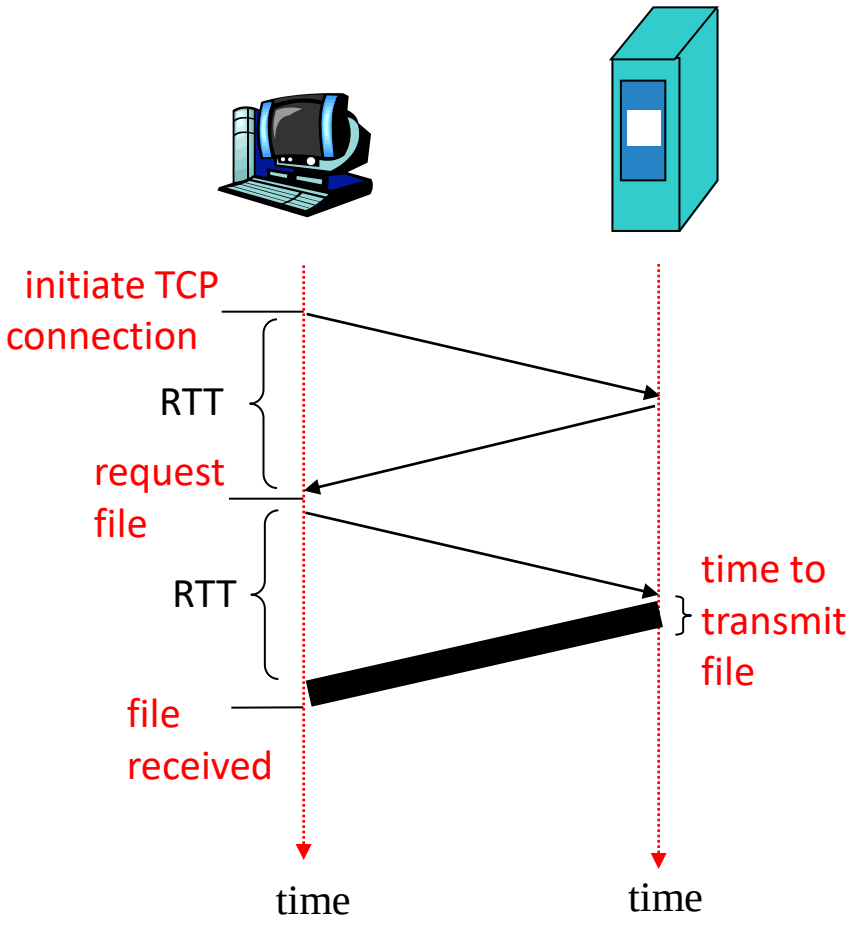


Web và HTTP (tt)



Web và HTTP (tt)

- RTT (Round Trip Time): thời gian để gửi một gói tin kích thước nhỏ từ client tới server và trở lại
- Thời gian trả lời (Response time):
  - 1 RTT để thiết lập kết nối TCP
  - 1 RTT cho yêu cầu HTTP và nhận byte đầu tiên của trả lời HTTP
  - Thời gian truyền file
- total = 2RTT + transmit time

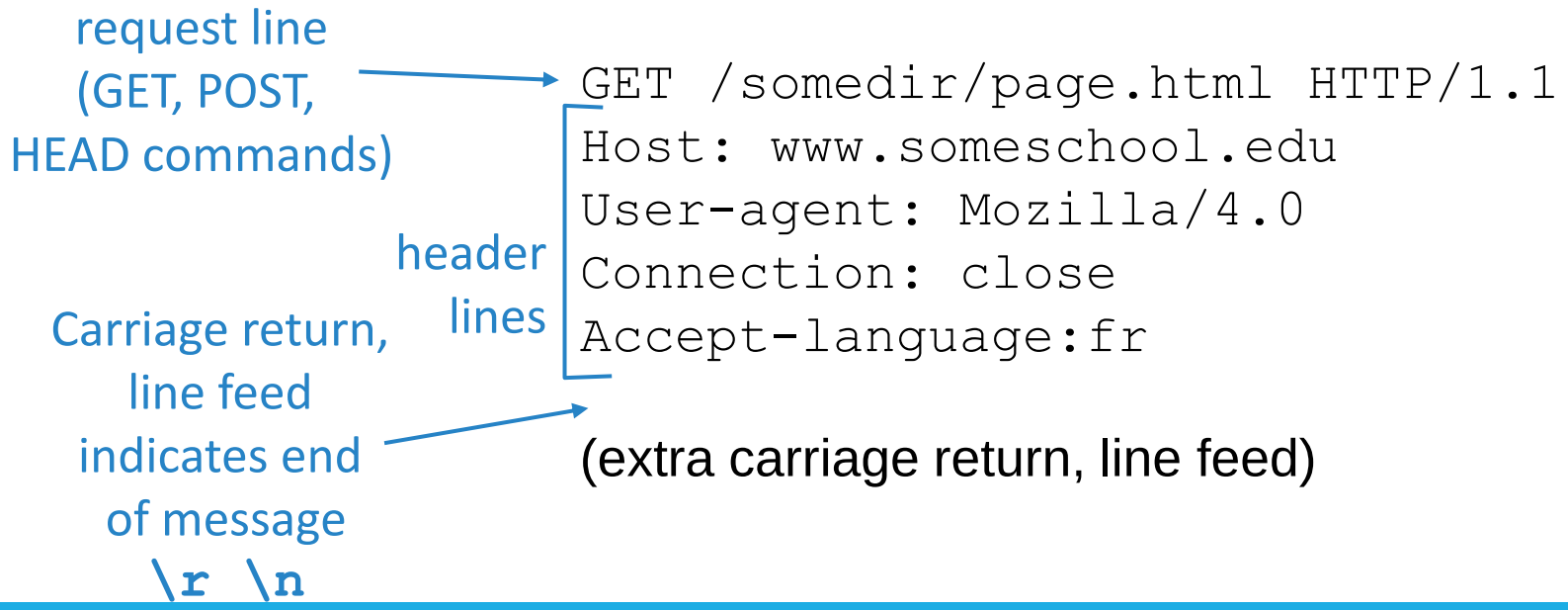


Web và HTTP (tt)

- Nonpersistent HTTP
  - Cần 2 RTT cho 1 đối tượng
  - Hệ điều hành phải cấp phát tài nguyên cho mỗi kết nối TCP
  - Trình duyệt phải mở song song nhiều kết nối TCP để lấy các đối tượng tham chiếu
- Persistent HTTP
  - Server giữ lại kết nối sau khi gửi trả lời cho client
  - Các thông điệp HTTP sau đó giữa client - server được gửi qua kết nối đó

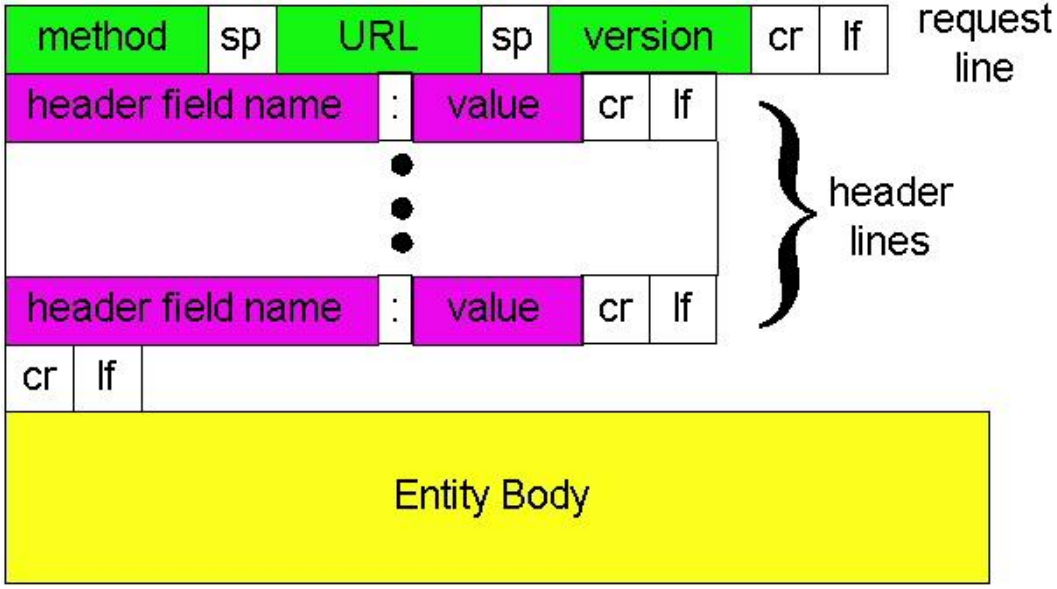
Web và HTTP (tt)

- Thông điệp HTTP
  - Hai kiểu thông điệp HTTP: yêu cầu (request), trả lời (response)
  - **Thông điệp HTTP request:**
    - ASCII (định dạng con người có thể đọc được)



Web và HTTP (tt)

- Định dạng chung của thông điệp HTTP request



Web và HTTP (tt)

- Kiểu Method:

HTTP/1.0

+ GET
+ POST
+ HEAD

  Yêu cầu server đặt đối tượng đã yêu cầu ra khỏi trả lời

HTTP/1.1

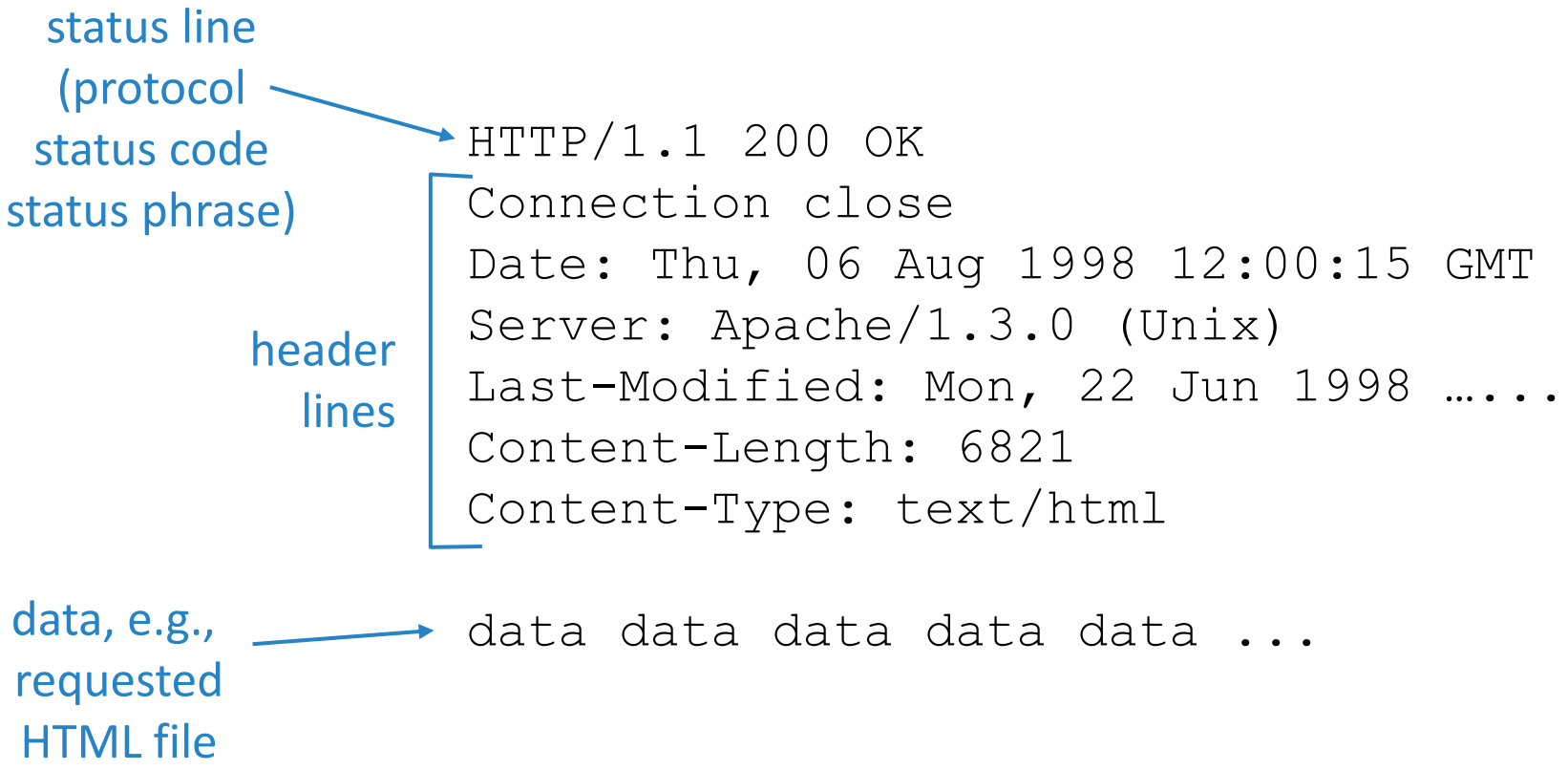
+ GET, POST, HEAD
+ PUT

  Upload file trong phần body lên đường dẫn chỉ trong URL
+ DELETE

  Xóa file trong trường URL

Web và HTTP (tt)

- **Thông điệp HTTP response:**



Web và HTTP (tt)

- Mã trạng thái HTTP response (dòng đầu tiên của thông điệp)

200 OK
Yêu cầu thực hiện thành công, đối tượng yêu cầu trong thông điệp

301 Moved Permanently
Đối tượng yêu cầu đã di chuyển vị trí, ví trí mới được chỉ ra trong thông điệp (Location:)

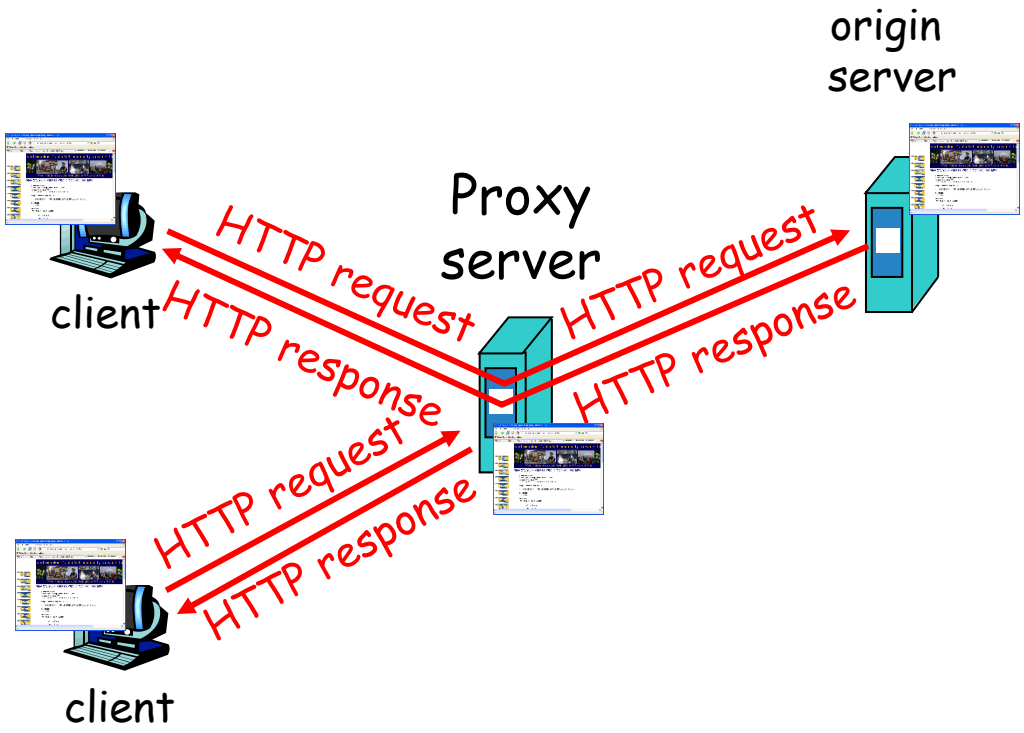
400 Bad Request
Server không hiểu thông điệp yêu cầu

404 Not Found
Không tìm thấy đối tượng yêu cầu

505 HTTP Version Not Supported

Web và HTTP (tt)

- Web caches (proxy server)
  - Mục đích: client không cần phải yêu cầu đến server gốc (origin server)
  - Được thiết lập thông qua trình duyệt Web
  - Trình duyệt gửi mọi yêu cầu HTTP qua Cache
    - Đối tượng trong cache: sẽ được trả về
    - Nếu không: yêu cầu được chuyển tới server gốc



Web và HTTP (tt)

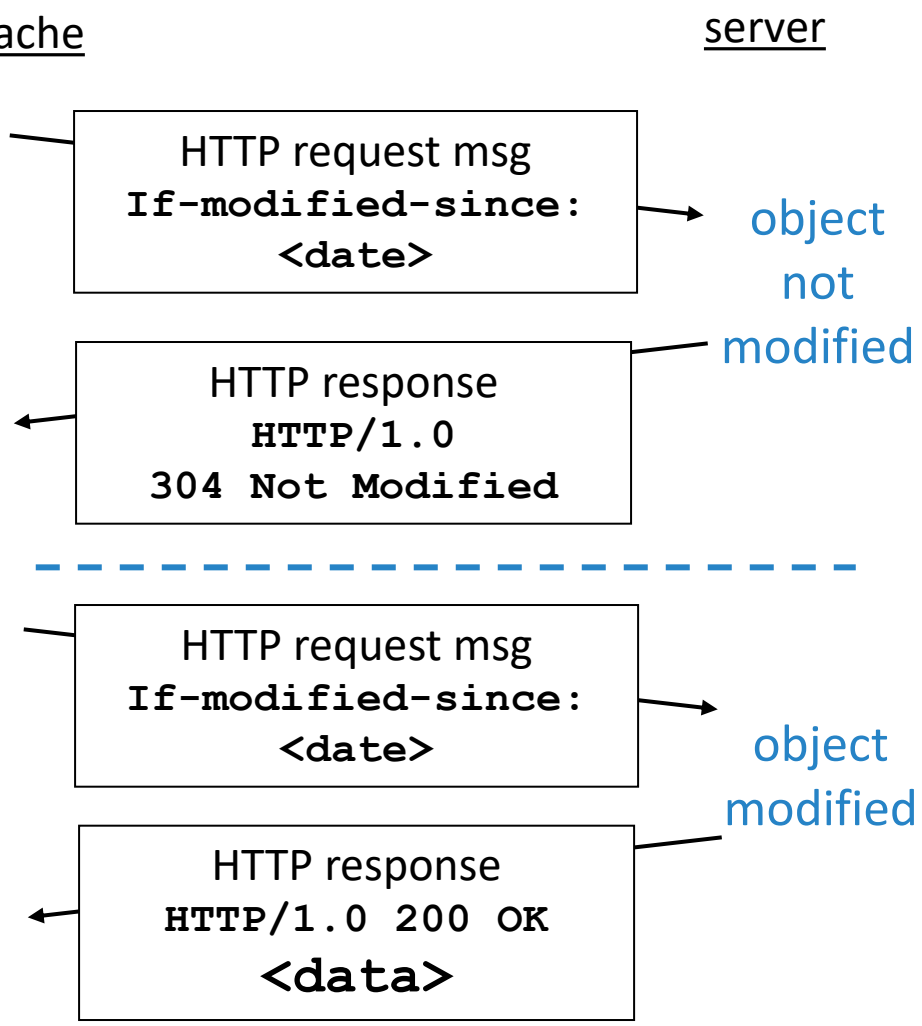
- Web cache hoạt động như cả client lẫn server
- Thông thường Cache được cài đặt bởi ISP (trường đại học, công ty, nhà cung cấp dịch vụ cho gia đình)
- Tại sao dùng Web cache?
  - Giảm thời gian trả lời cho yêu cầu của client
  - Giảm lưu lượng trên đường truy cập của tổ chức

Web và HTTP (tt)

- Conditional GET
  - Mục đích: không cần cập nhật nếu dữ liệu của Cache là mới nhất
  - Cache: cho biết ngày của dữ liệu cache trong cache trong HTTP request

    `If-modified-since: <date>`
  - Server: trả lời không chứa đối tượng nếu bản cache là bản cập nhật mới nhất

    `HTTP/1.0 304 Not Modified`



Tầng ứng dụng

2.1. Các nguyên tắc của ứng dụng mạng

2.2. Web và HTTP

2.3. **FTP**

2.4. Email

2.5. DNS

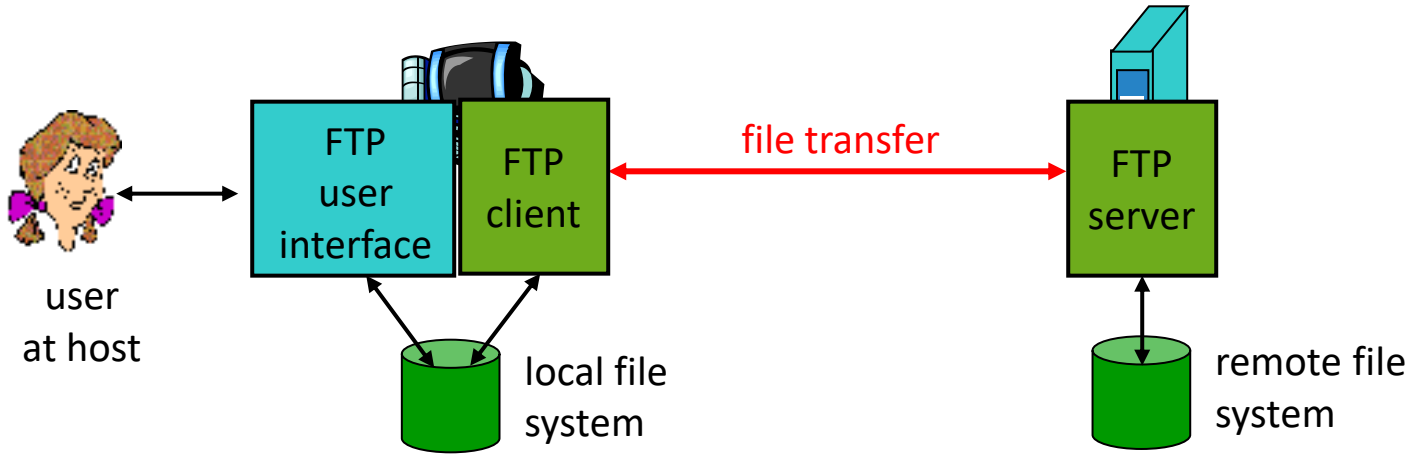
2.6. Ứng dụng P2P

2.7. Lập trình Socket với TCP

2.8. Lập trình Socket với UDP

FTP

- FTP: File Transfer Protocol

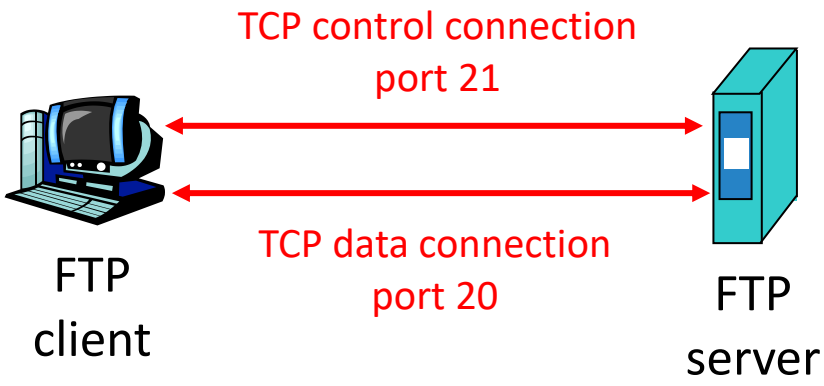


  - Truyền file tới/từ host ở xa
  - Mô hình Client/Server
    - Client: khởi đầu việc truyền (tới hoặc từ host ở xa)
    - Server: host ở xa
  - FTP: RFC 959
  - FTP server: port 21

FTP (tt)

- Điều khiển riêng biệt với truyền dữ liệu
  - FTP client giao tiếp với FTP server tại cổng 21, dùng TCP làm giao thức vận chuyển
  - Client chứng thực thông qua kết nối điều khiển (control connection)
  - Client xem thư mục từ xa bằng cách gửi lệnh (command) qua kết nối điều khiển
  - Khi nhận được lệnh truyền file, server mở một kết nối TCP thứ 2 đến client (dùng để truyền file)
  - Sau khi truyền file, server đóng kết nối này

FTP (tt)



- Server mở kết nối TCP thứ 2 để truyền file
- Kết nối điều khiển: “out of band”
- FTP server duy trì trạng thái (state): client đã chứng thực, thư mục hiện hành

FTP (tt)

- Lệnh (command) và phản hồi (respond)
  - *Ví dụ một số lệnh*

    Gửi văn bản mã ASCII qua kênh điều khiển

    - **USER** `username`
    - **PASS** `password`
    - **LIST** trả về một danh sách các file trong thư mục hiện tại
    - **RETR** `filename` lấy file
    - **STOR** `filename` đưa file lên remote host
  - *Ví dụ phản hồi từ server*

    Status code và status phrase (giống HTTP)

    - **331** `Username OK, password required`
    - **125** `data connection already open, transfer starting`
    - **425** `Can't open data connection`
    - **452** `Error writing file`

Tầng ứng dụng

2.1. Các nguyên tắc của ứng dụng mạng

2.2. Web và HTTP

2.3. FTP

2.4. **Email**

2.5. DNS

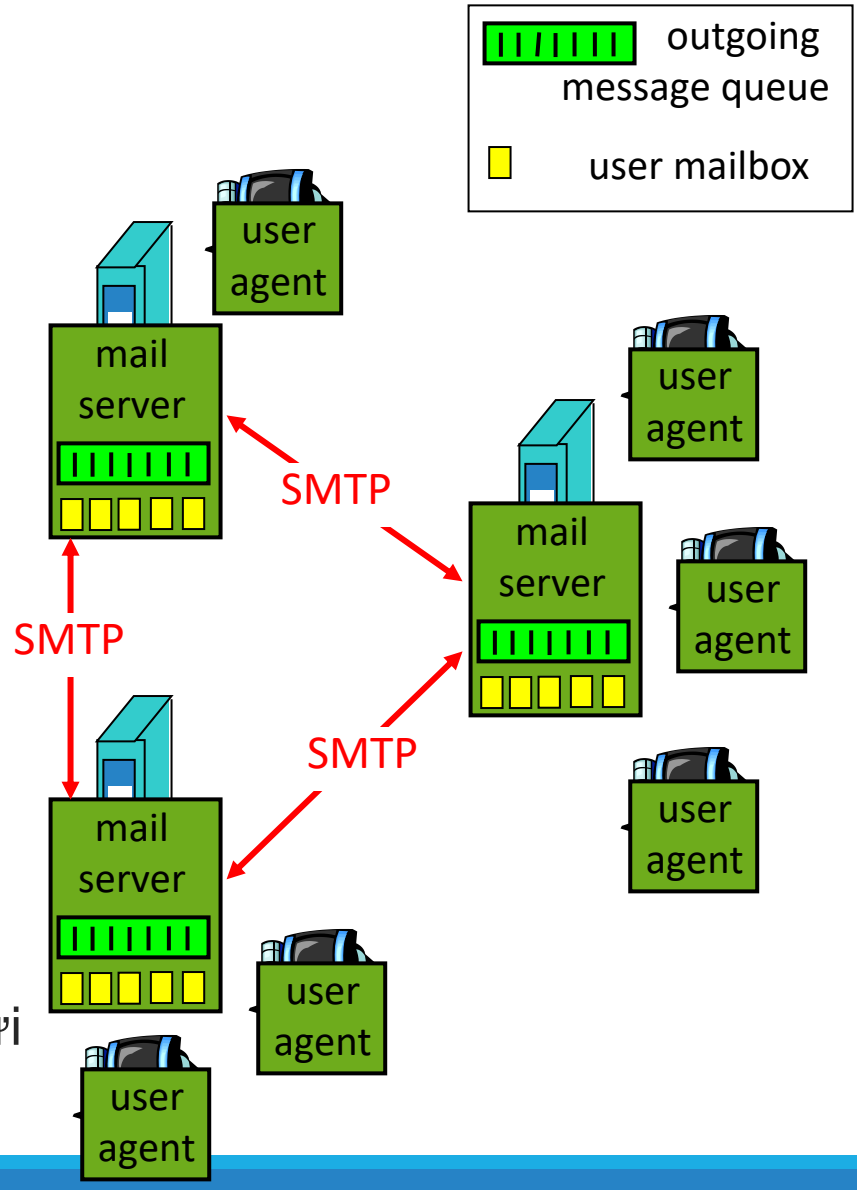
2.6. Ứng dụng P2P

2.7. Lập trình Socket với TCP

2.8. Lập trình Socket với UDP

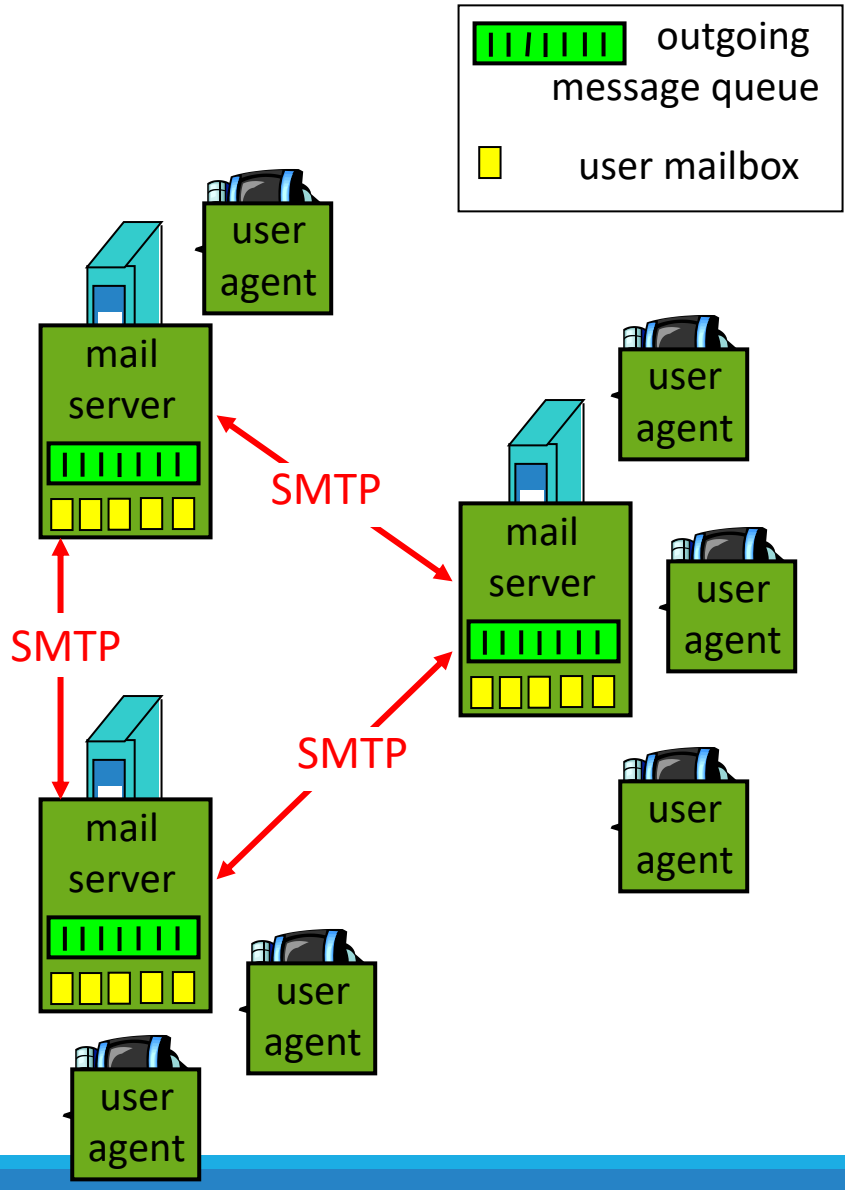
Email

- Ba thành phần chính
  - User agent (UA)
  - Mail server
  - SMTP (Simple Mail Transfer Protocol)
- User agent
  - Còn gọi là Mail Reader
  - Soạn, sửa, đọc thông điệp mail (Mail message)
  - Ví dụ: Eudora, Outlook, Thunderbird, …
  - Các thông điệp (gửi đến và gửi đi) chứa trên server



Email (tt)

- Mail server
  - Mailbox chứa các thông điệp thư điện tử gửi đến cho người sử dụng
  - Message queue của các thông điệp thư điện tử gửi đi
  - Giao thức SMTP giữa các mail server để gửi các thông điệp thư điện tử
    - Client: mail server gửi
    - "Server": mail server nhận



Email (tt)

- SMTP [RFC 2821]
  - Sử dụng TCP để truyền tin cậy các thông điệp thư điện tử từ client tới server, cổng 25
  - Gửi trực tiếp: server gửi tới server nhận
  - Ba bước của việc truyền thông điệp
    - Bắt tay (handshaking)
    - Truyền các thông điệp (transfer of messages)
    - Kết thúc (closure)
  - Lệnh (Command) / Trả lời (Response)
    - Lệnh: văn bản mã ASCII
    - Trả lời: status code và status phrase
  - Các thông điệp sử dụng mã ASCII 7-bit

Email (tt)

- Alice gửi thông điệp email cho Bob

1) Alice dùng UA để soạn email và gửi cho bob@someschool.edu

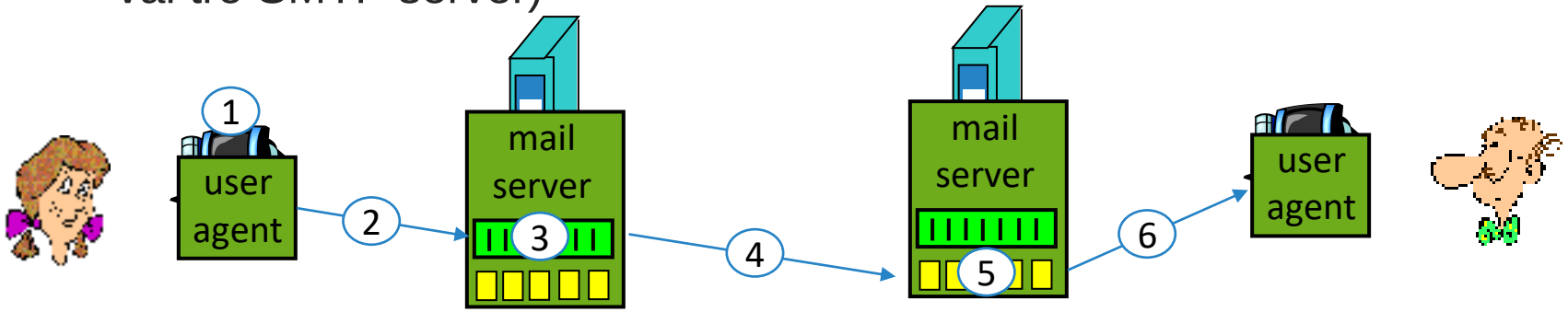
2) UA của Alice gửi thông điệp đến mail server của cô ta; email được đặt trong message queue

3) Mail server của Alice (đóng vai trò SMTP client) mở kết nối TCP đến mail server của Bob (đóng vai trò SMTP server)

4) SMTP client gửi thông điệp của Alice qua kết nối TCP

5) Mail server của Bob chuyển thông điệp vào mailbox của Bob

6) Bob sử dụng user agent để đọc thông điệp email



Email (tt)

- Ví dụ tương tác SMTP đơn giản

```
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: ?
C: How do you like ketchupabout pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

Email (tt)

- Thử nghiệm gửi email bằng dòng lệnh
  - **`telnet servername 25`**
  - Sử dụng các lệnh HELO, MAIL FROM, RCPT TO, DATA, QUIT để gửi email

Email (tt)

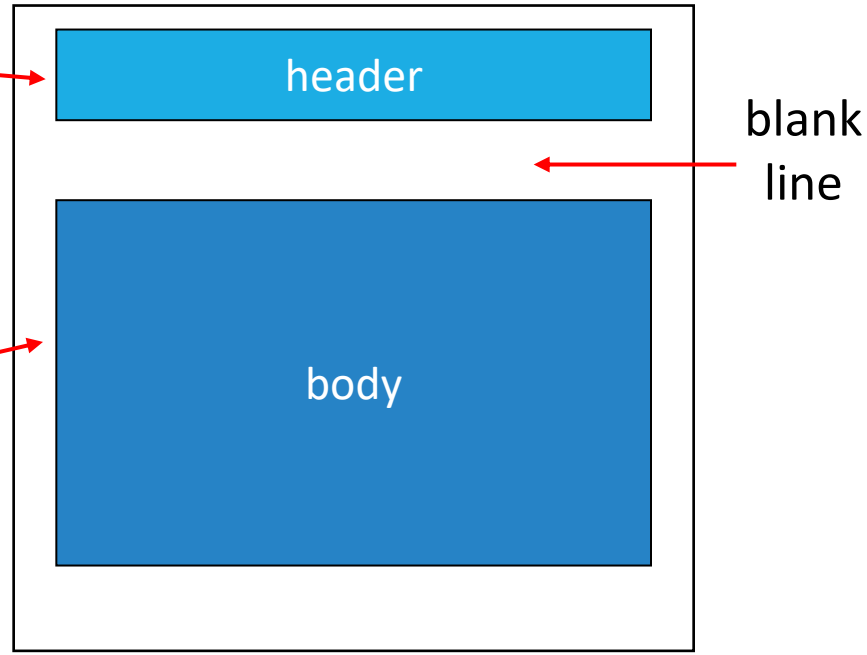
- SMTP sử dụng persistent connection
- SMTP server sử dụng `CRLF.CRLF` để xác định kết thúc thông điệp

Email (tt)

- So sánh SMTP và HTTP
  - HTTP: pull
  - SMTP: push
  - Cả hai có tương tác Lệnh/ Trả lời dạng mã ASCII, status code
  - HTTP: mỗi đối tượng được đóng gói trong chính thông điệp trả lời
  - SMTP: nhiều đối tượng được gửi trong thông điệp có nhiều phần (multipart msg)

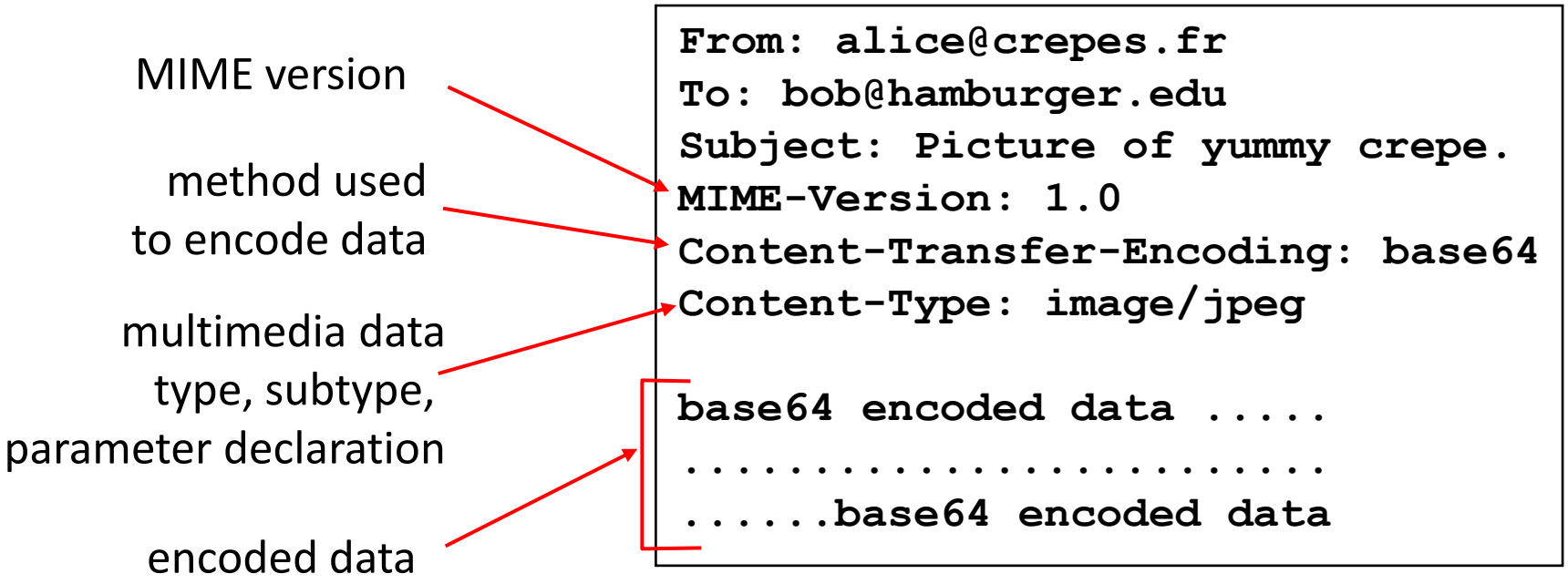
Email (tt)

- Định dạng của thông điệp
  - Header
    - To:
    - From:
    - Subject:
  - Body
    - Nội dung thông điệp (ký tự ASCII)



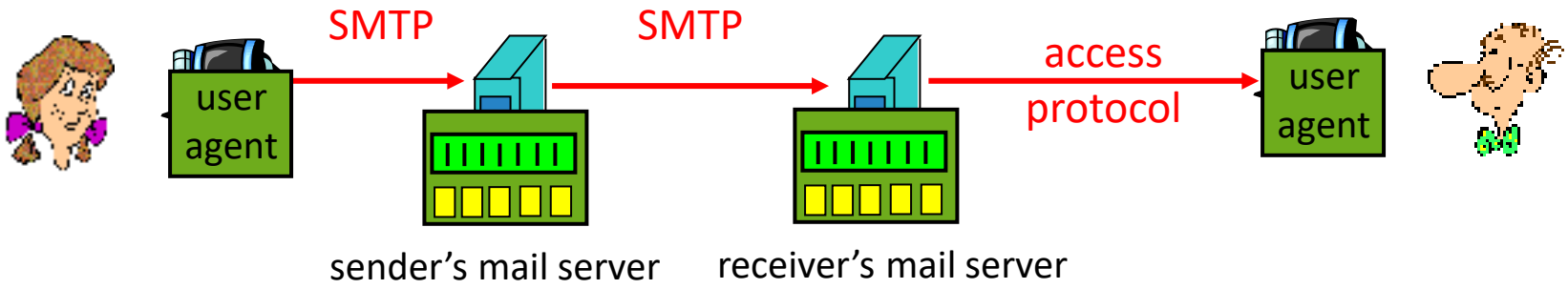
Email (tt)

- Định dạng của thông điệp – các mở rộng đa phương tiện
  - MIME: Multimedia Mail Extension, RFC 2045, 2046
  - Thêm các dòng trong header của thông điệp, khai báo kiểu nội dung MIME



Email (tt)

- Các giao thức truy cập thư điện tử



- SMTP: chuyển/lưu trữ thư tới server bên nhận
- Giao thức truy cập thư: lấy thư từ server
  - POP: Post Office Protocol [RFC 1939]
    - Chứng thực (agent <--> server) và tải thư
  - IMAP: Internet Mail Access Protocol [RFC 1730]
    - Nhiều tính năng hơn (phức tạp hơn)
    - Thao tác trên các thông điệp lưu trên server
  - HTTP: Hotmail , Yahoo! Mail, …

Email (tt)

- Giao thức POP3

Bước chứng thực

- Command của client:
  - **user**: declare username
  - **pass**: password
- Response của server
  - **+OK**
  - **-ERR**

Bước giao dịch, client:

- **list**: liệt kê số lượng thông điệp
- **retr**: lấy thông điệp dựa vào số hiệu
- **dele**: xóa
- **quit**

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

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

Email (tt)

- POP3 và IMAP

POP3

- Ví dụ trước sử dụng chế độ “Download and delete”
- Bob không thể đọc lại các thư điện tử nếu Bob chuyển sang client khác
- Chế độ “Download and keep”: sao chép thông điệp trên các client khác nhau
- POP3 không lưu trạng thái giữa các phiên

IMAP

- Giữ tất cả các thông điệp trên server
- Cho phép người sử dụng tổ chức thông điệp vào các thư mục
- IMAP giữ trạng thái người sử dụng qua các phiên:
  - Ánh xạ giữa ID của thông điệp và tên thư mục

Tầng ứng dụng

2.1. Các nguyên tắc của ứng dụng mạng

2.2. Web và HTTP

2.3. FTP

2.4. Email

2.5. **DNS**

2.6. Ứng dụng P2P

2.7. Lập trình Socket với TCP

2.8. Lập trình Socket với UDP

DNS

- DNS: Domain Name System
  - Con người: có nhiều định danh:
    - Tên, CMND, hộ chiếu, …
  - Host, router trên Internet:
    - Địa chỉ IP
    - Tên, ví dụ: www.yahoo.com sử dụng bởi con người
  - Câu hỏi: ánh xạ giữa địa chỉ IP và tên?

DNS (tt)

- Hệ thống tên miền (DNS)
  - Cơ sở dữ liệu phân tán được thực hiện bởi nhiều name server phân cấp
  - Là giao thức tầng ứng dụng để host, router,… phân giải tên thành địa chỉ IP

DNS (tt)

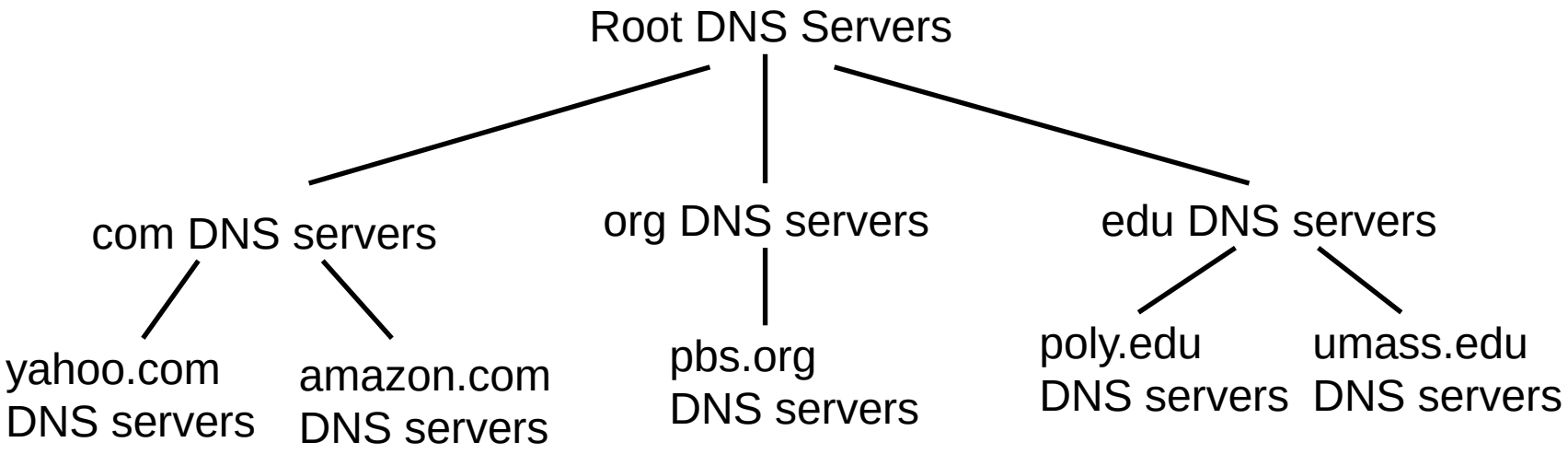
- Các dịch vụ DNS
  - Dịch tên host sang địa chỉ IP
  - Bí danh cho host (host alias)
    - Canonical name và alias name
  - Bí danh cho Mail server
  - Phân tải
    - Web server: nhiều địa chỉ IP cho một canonical name

DNS (tt)

- Tại sao không sử dụng DNS tập trung?
  - Một điểm lỗi
  - Lưu lượng tập trung
  - Cơ sở dữ liệu tập trung ở xa
  - Bảo trì

DNS (tt)

- Cơ sở dữ liệu phân cấp và phân tán

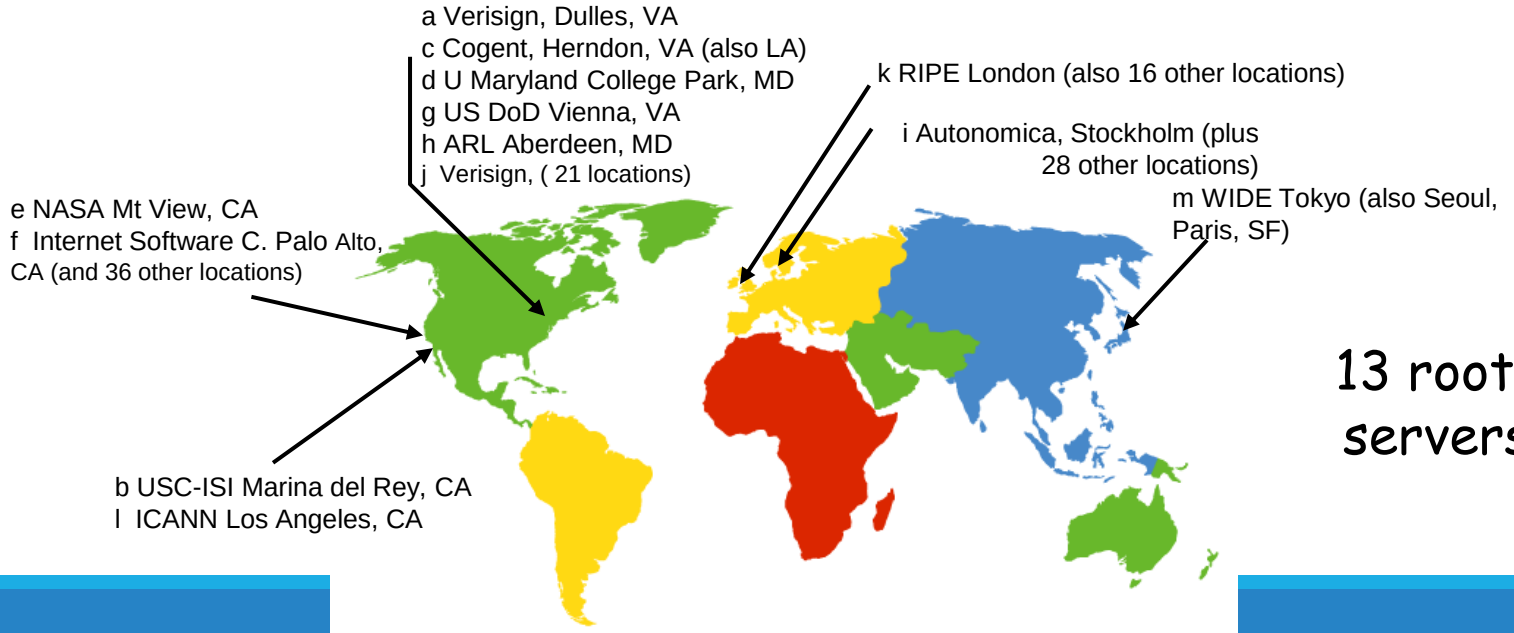


Client muốn biết địa chỉ IP của www.amazon.com:

- Client yêu cầu root server để tìm com DNS server
- Client yêu cầu com DNS server để xác định amazon.com DNS server
- Client yêu cầu amazon.com DNS server để lấy địa chỉ IP của www.amazon.com

DNS (tt)

- DNS – Root name server
  - Local name server không trả lời được thì nó sẽ liên lạc với Root name server
  - Root name server:
    - Liên lạc với authoritative name server nếu nó không biết ánh xạ tên miền
    - Lấy ánh xạ
    - Trả ánh xạ về cho Local name server



13 root name servers worldwide

DNS (tt)

- TLD và Authoritative server
  - **Top-level domain (TLD) server:** có vai trò đối với tên miền com, org, net, edu… và tất cả các miền quốc gia mức trên cùng uk, fr, ca, jp…
  - **Authoritative DNS server:** DNS server của các tổ chức cung cấp ánh xạ authoritative hostname thành địa chỉ IP cho server của tổ chức
    - Có thể được triển khai bởi tổ chức hoặc nhà cung cấp dịch vụ

DNS (tt)

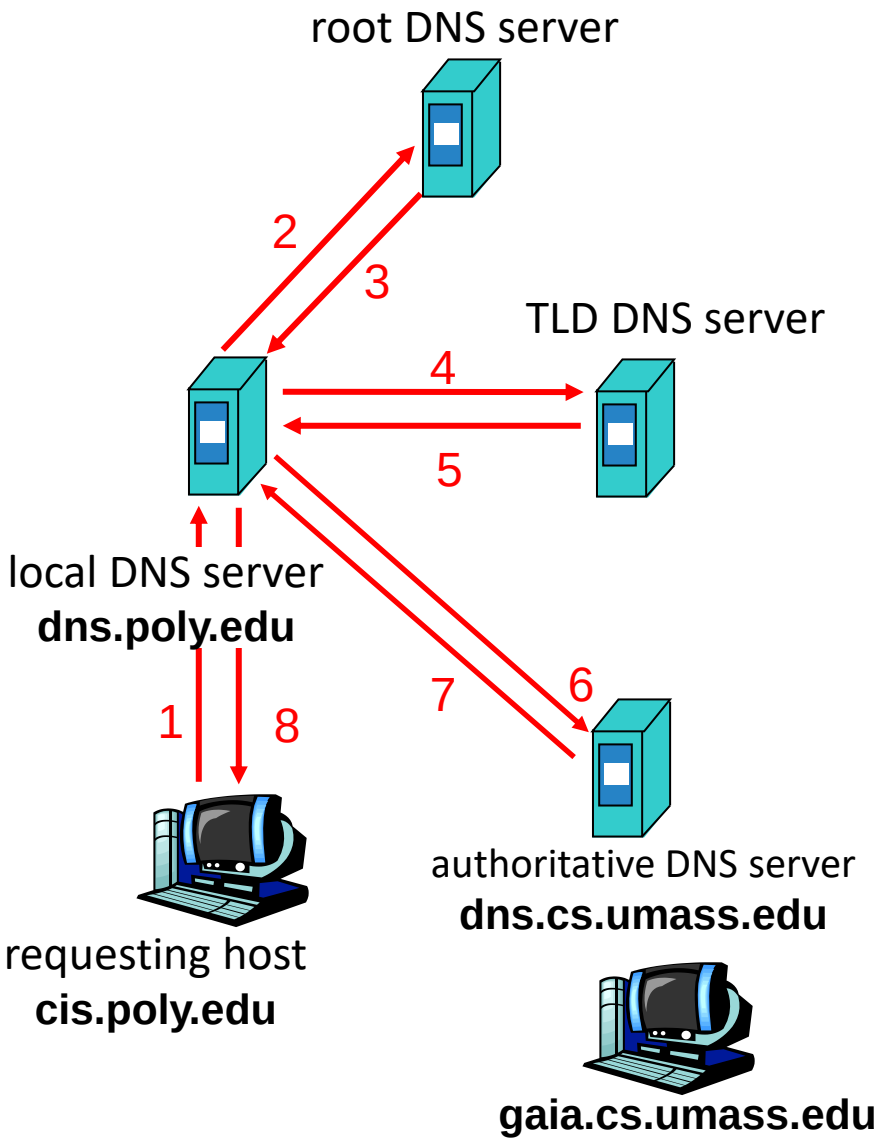
- Local name server
  - Không hoàn toàn thuộc vào hệ thống phân cấp
  - Có mặt ở ISP (residential ISP, công ty, tổ chức)
    - Còn gọi là “Default name server”
  - Khi một host tạo truy vấn DNS, truy vấn được gửi tới Local DNS server của nó
    - Hoạt động như một proxy, chuyển tiếp truy vấn vào hệ thống phân cấp

DNS (tt)

- Ví dụ DNS phân giải tên

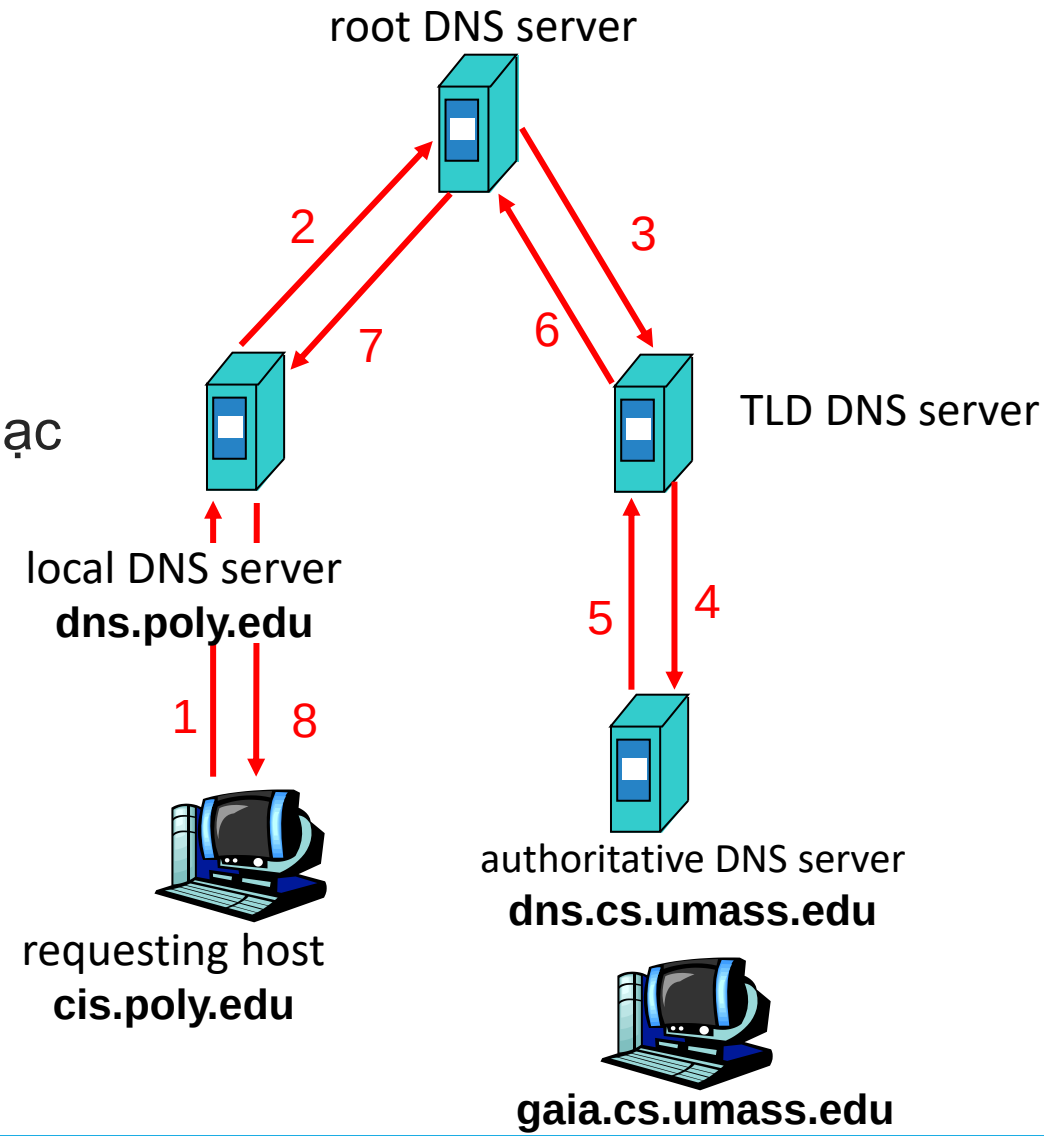
  Host tại cis.poly.edu muốn biết địa chỉ IP của gaia.cs.umass.edu

  - **Truy vấn lặp (iterated query):**
    - Server liên lạc và trả về tên của server cần liên lạc tiếp theo
    - "Tôi không biết nhưng anh muốn biết thì đi hỏi server này"



DNS (tt)

- **Truy vấn đệ quy (recursive query):**
  - Giao toàn bộ việc phân giải tên cho name server liên lạc
  - Tải lớn?



DNS (tt)

- Lưu trữ tạm và cập nhật bản ghi
  - Name server nào đó học các ánh xạ, sẽ lưu trữ tạm các ánh xạ đó
  - Lưu trữ tạm quá hạn (biến mất) sau một khoảng thời gian
  - Thông thường TLD server lưu trữ tạm trong các Local name servers
    - Vì thế, Root name server không bị truy cập thường xuyên

DNS (tt)

Bản ghi DNS (DNS record)

DNS: cơ sở dữ liệu phân tán chứa các bản ghi tài nguyên (Resource record - RR)

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

- **Type=A**
  - *name* là hostname
  - *value* là địa chỉ IP
- **Type=NS**
  - *name* là domain
  - *value* là địa chỉ của authoritative name server cho domain đó
- **Type=CNAME**
  - *name* là tên bí danh (alias) cho tên thật (cannonical name)
  - *value* là tên thật
- **Type=MX**
  - *value* là tên của mail server

DNS (tt)

- Chèn bản ghi vào DNS

  Ví dụ: công ty mới thành lập "Network Utopia"

  - Đăng ký tên miền `networkuptopia.com` tại nhà cung cấp dịch vụ tên miền Registrar
    - Cần cung cấp cho Registrar tên và địa chỉ IP của authoritative name server (primary và secondary)
    - Registrar chèn thêm 2 RR vào trong com TLD server:
      - `(networkutopia.com, dns1.networkutopia.com, NS)`
      - `(dns1.networkutopia.com, 212.212.212.1, A)`
    - Thêm vào authoritative server bản ghi kiểu A cho www.networkuptopia.com (để chạy web) và bản ghi kiểu MX cho networkutopia.com (để chạy mail)
  - *Người khác xác định địa chỉ IP của Website này như thế nào?*

Tầng ứng dụng

2.1. Các nguyên tắc của ứng dụng mạng

2.2. Web và HTTP

2.3. FTP

2.4. Email

2.5. DNS

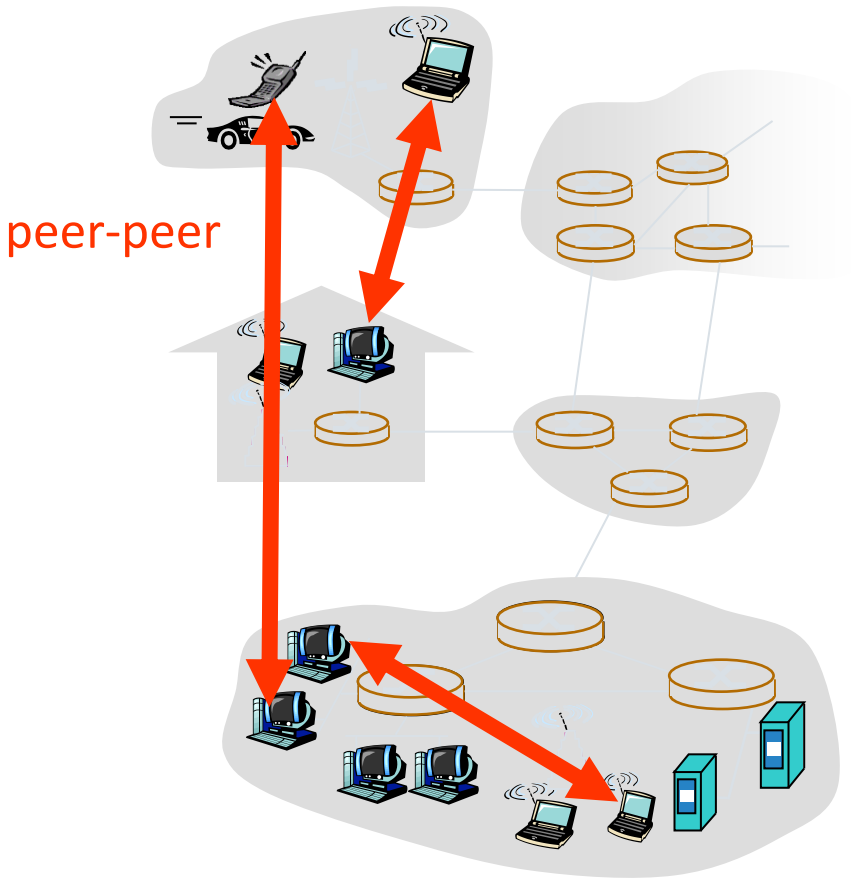
2.6. **Ứng dụng P2P**

2.7. Lập trình Socket với TCP

2.8. Lập trình Socket với UDP

Ứng dụng P2P

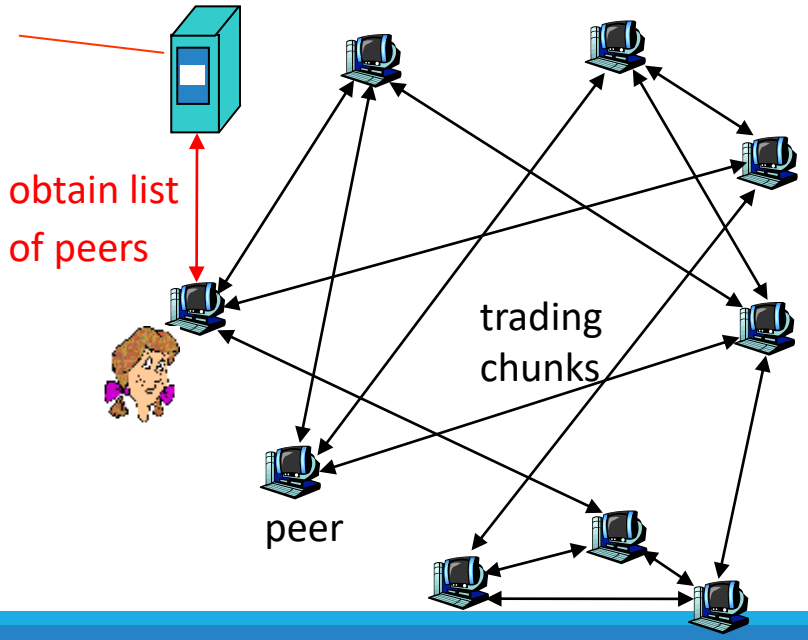
- Không cần phải có server hoạt động liên tục
- Các hệ thống đầu cuối kết nối trực tiếp với nhau
- Các peer (nút mạng) không cần kết nối liên tục vào hệ thống và địa chỉ IP có thể thay đổi
- Ứng dụng:
  - Chia sẻ file (BitTorrent)
  - VoIP (Skype)
  - …



Ứng dụng P2P (tt)

- BitTorrent
  - File được chia thành các chunk (phần) có kích thước 256 Kb
  - Các peer trong torrent gửi/nhận các chunk của file

tracker: kiểm tra các peer tham gia vào torrent



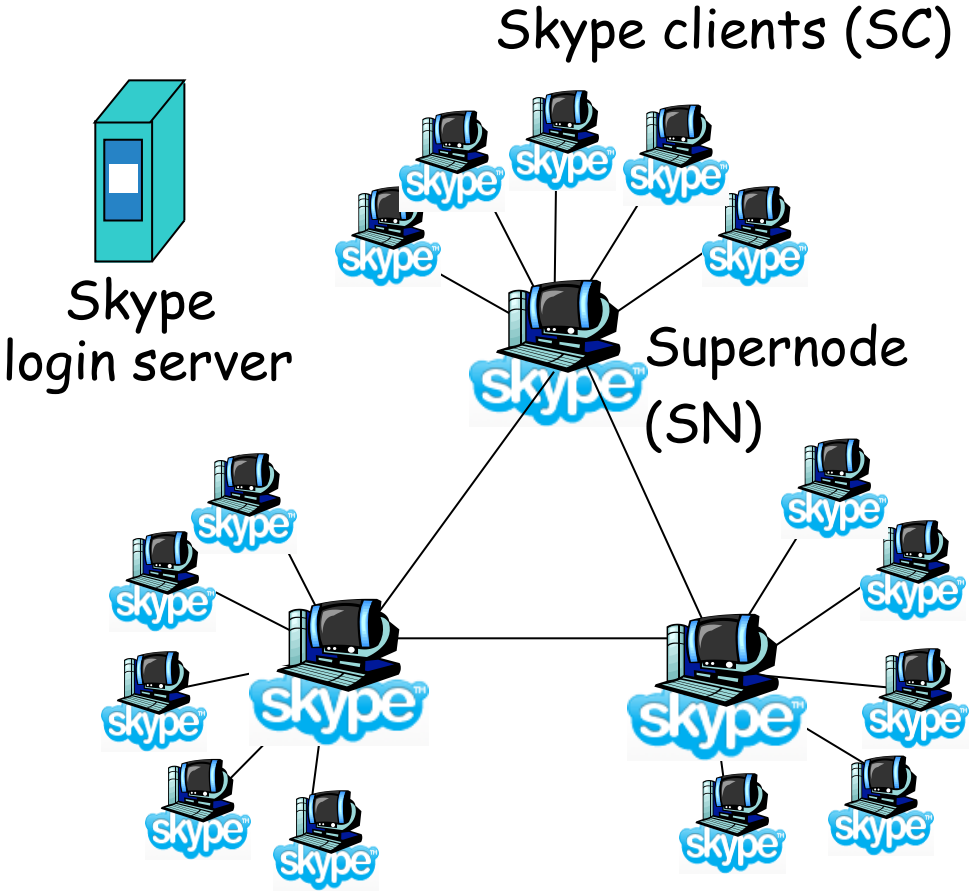
torrent: nhóm các peer trao đổi chunk của file

Ứng dụng P2P (tt)

- Peer tham gia vào torrent:
  - Không có các chunk, nhưng sẽ tích lũy chúng qua thời gian từ các peer khác
  - Đăng ký với tracker để nhận được danh sách các peer, kết nối với các peer "hàng xóm"
- Trong khi download, peer sẽ upload các chunk tới các peer khác
- peer có thể thay đổi các peer mà nó sẽ trao đổi chunk
- *churn:* các peer có thể đến hoặc đi
- Khi peer có được toàn bộ file, nó có thể rời đi (selfishly) hoặc ở lại (altruistically) trong torrent

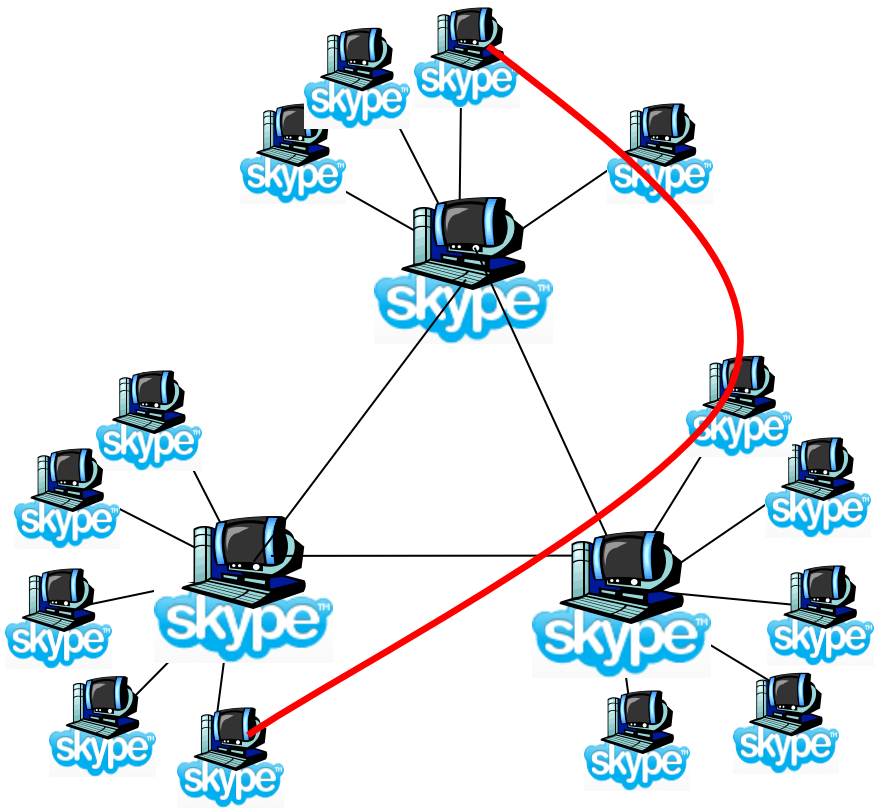
Ứng dụng P2P (tt)

- Skype
  - Từng cặp người dùng giao tiếp với nhau
  - Sử dụng giao thức tầng ứng dụng riêng
  - Phân cấp với SuperNode (SN)



Ứng dụng P2P (tt)

- Người dùng (peer) được đặt sau "NAT" – Network Address Translation
  - NAT ngăn chặn "bên ngoài" thiết lập kết nối vào "bên trong"
- Giải pháp:
  - Sử dụng SN như điểm trung gian (relay)
  - Các peer sẽ kết nối đến relay
  - Các peer có thể giao tiếp với nhau thông qua relay, vượt qua NAT



Tầng ứng dụng

2.1. Các nguyên tắc của ứng dụng mạng

2.2. Web và HTTP

2.3. FTP

2.4. Email

2.5. DNS

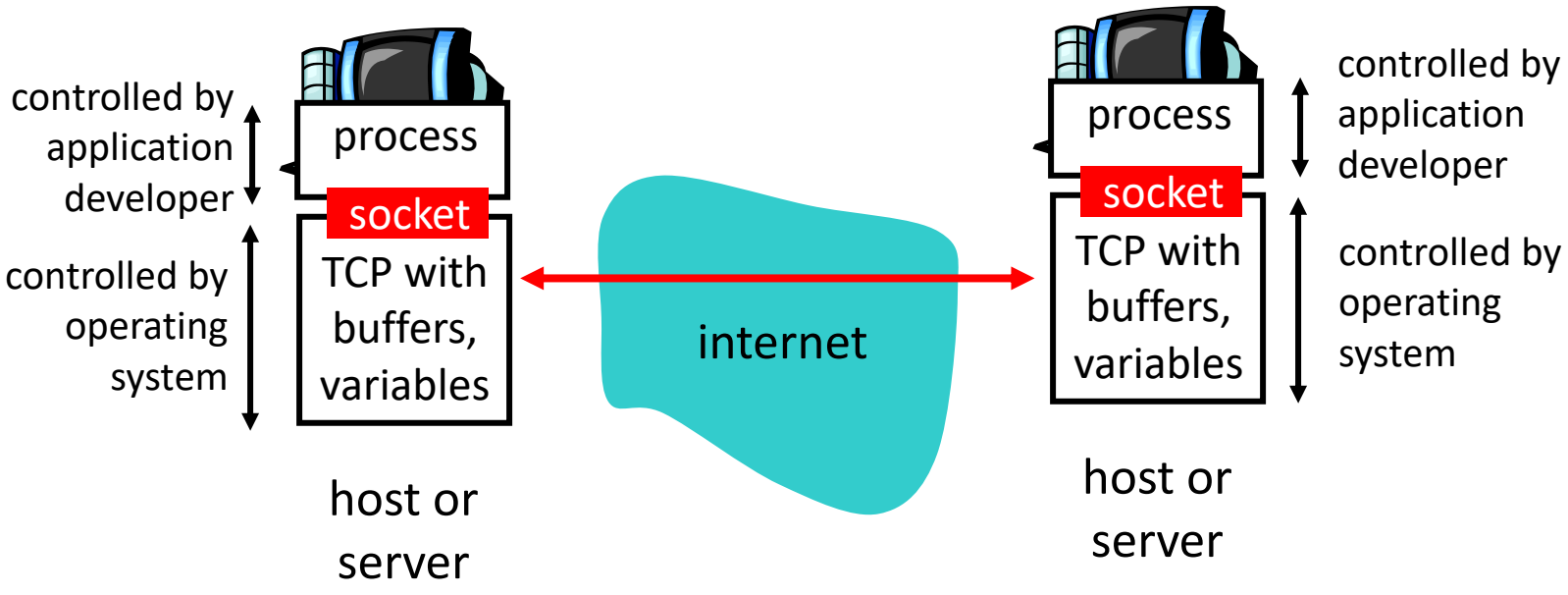
2.6. Ứng dụng P2P

2.7. **Lập trình Socket với TCP**

2.8. Lập trình Socket với UDP

Lập trình Socket với TCP

- Socket: “cánh cửa” giữa tiến trình ứng dụng và giao thức tầng vận chuyển (TCP hoặc UDP)
- Dịch vụ TCP: truyền dữ liệu tin cậy (các byte) giữa các tiến trình

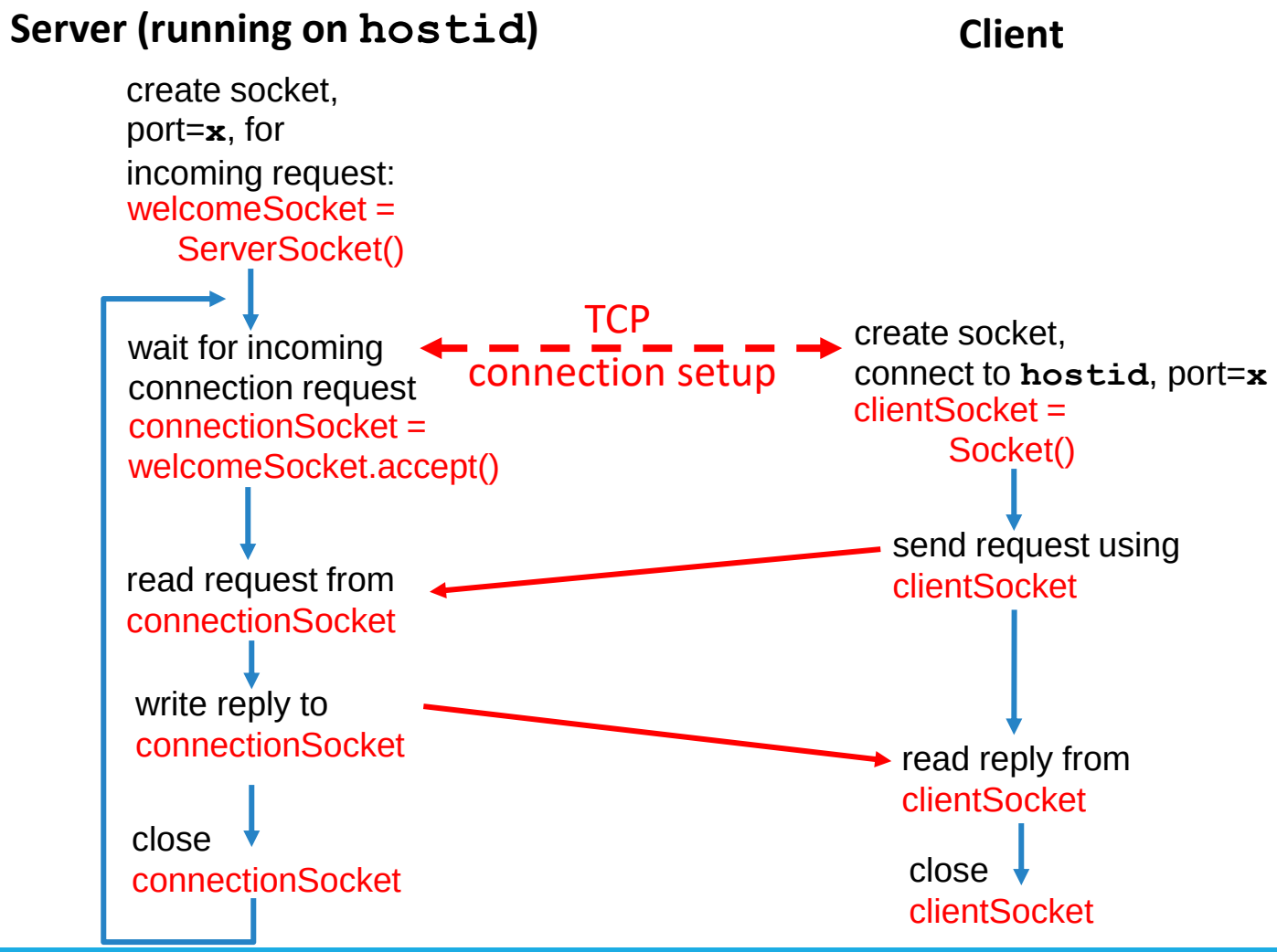


Lập trình Socket với TCP (tt)

- Client phải liên lạc với Server
  - Tiến trình server phải đang chạy
  - Server phải mở socket để Client liên lạc
- Client liên lạc với Server bằng cách:
  - Tạo client-local TCP socket
  - Gán địa chỉ IP, cổng của tiến trình Server
  - Khi client tạo socket: client TCP kết nối đến server TCP
- Khi nhận được sự liên lạc của client, Server tạo một socket mới cho tiến trình server để giao tiếp với client
  - Cho phép server giao tiếp với nhiều client
  - Giá trị source port dùng để phân biệt các client

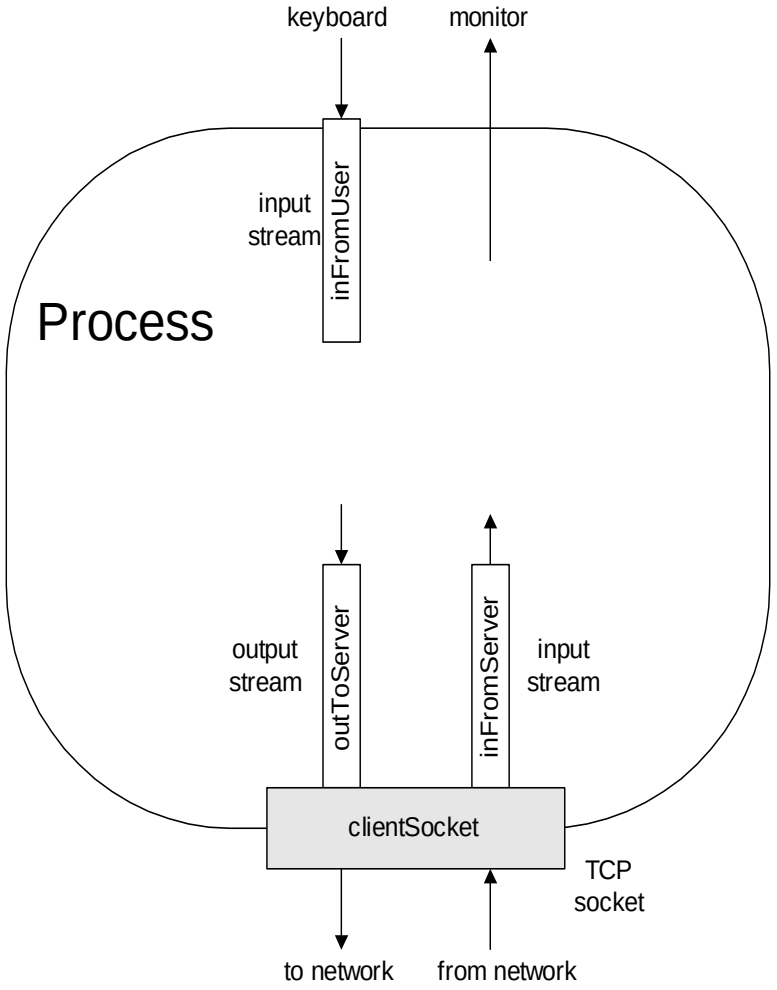
Lập trình Socket với TCP (tt)

- Tương tác giữa Server và Client



Lập trình Socket với TCP (tt)

- Stream: một luồng (chuỗi có trật tự) các ký tự vào hoặc ra một tiến trình
- Input stream: luồng dữ liệu vào tiến trình
- Output stream: luồng dữ liệu xuất ra từ tiến trình



Lập trình Socket với TCP (tt)

- Ví dụ lập trình Socket với TCP

  Xây dựng ứng dụng Client-Server như sau:

  1. Client đọc các ký tự từ thiết bị nhập (**`inFromUser`** stream), gửi dữ liệu sang Server (**`outToServer`** stream) thông qua Socket
  2. Server đọc dữ liệu từ Socket
  3. Server chuyển chuỗi ký tự (từ Client gửi qua) sang chữ in hoa (uppercase) và gửi lại cho Client
  4. Client đọc dữ liệu từ Socket (**`inFromServer`** stream), hiển thị kết quả lên màn hình.

Lập trình Socket với TCP (tt)

- **TCP Server java**

```
import java.io.*;
import java.net.*;

public class TCPServer {
    public static void main(String argv[]) throws Exception
    {
        //Declare variables
        String clientSentence;
        String capitalizedSentence;

        //Create server welcoming socket (at port 6789)
        ServerSocket welcomeSocket = new ServerSocket(6789);
```

Lập trình Socket với TCP (tt)

```
while(true)
        {
            //Wait, on welcoming socket for contact by client
            Socket connectionSocket = welcomeSocket.accept();

            //Create input stream, attacked to socket
            BufferedReader inFromClient = new BufferedReader(new
InputStreamReader(connectionSocket.getInputStream()));

            //Create output stream, attacked to socket
            DataOutputStream outToClient = new
DataOutputStream(connectionSocket.getOutputStream());

            //Read in line from socket
            clientSentence = inFromClient.readLine();

            //Convert line to upper case
            capitalizedSentence = clientSentence.toUpperCase() +
'\n';

            //Write out line to socket
            outToClient.writeBytes(capitalizedSentence);
        }
    }
}
```

Lập trình Socket với TCP (tt)

- **TCP Client java**

```
import java.io.*;
import java.net.*;

public class TCPClient {
    public static void main(String argv[]) throws Exception
    {
        //Declare variables
        String sentence;
        String modifiedSentence;

        //Create input stream
        BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));

        //Create client socket
        Socket clientSocket = new Socket("localhost", 6789);

        //Create output stream attacked to socket
        DataOutputStream outToServer = new
DataOutputStream(clientSocket.getOutputStream());

        //Create input stream attacked to socket
        BufferedReader inFromServer = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
```

Lập trình Socket với TCP (tt)

```
        //Read characters from keyboard
        sentence = inFromUser.readLine();

        //Send line to server (byte)
        outToServer.writeBytes(sentence + '\n');

        //Read line from server
        modifiedSentence = inFromServer.readLine();

        //Display
        System.out.println("FROM SERVER: " + modifiedSentence);

        //Close client socket
        clientSocket.close();
    }
}
```

Tầng ứng dụng

2.1. Các nguyên tắc của ứng dụng mạng

2.2. Web và HTTP

2.3. FTP

2.4. Email

2.5. DNS

2.6. Ứng dụng P2P

2.7. Lập trình Socket với TCP

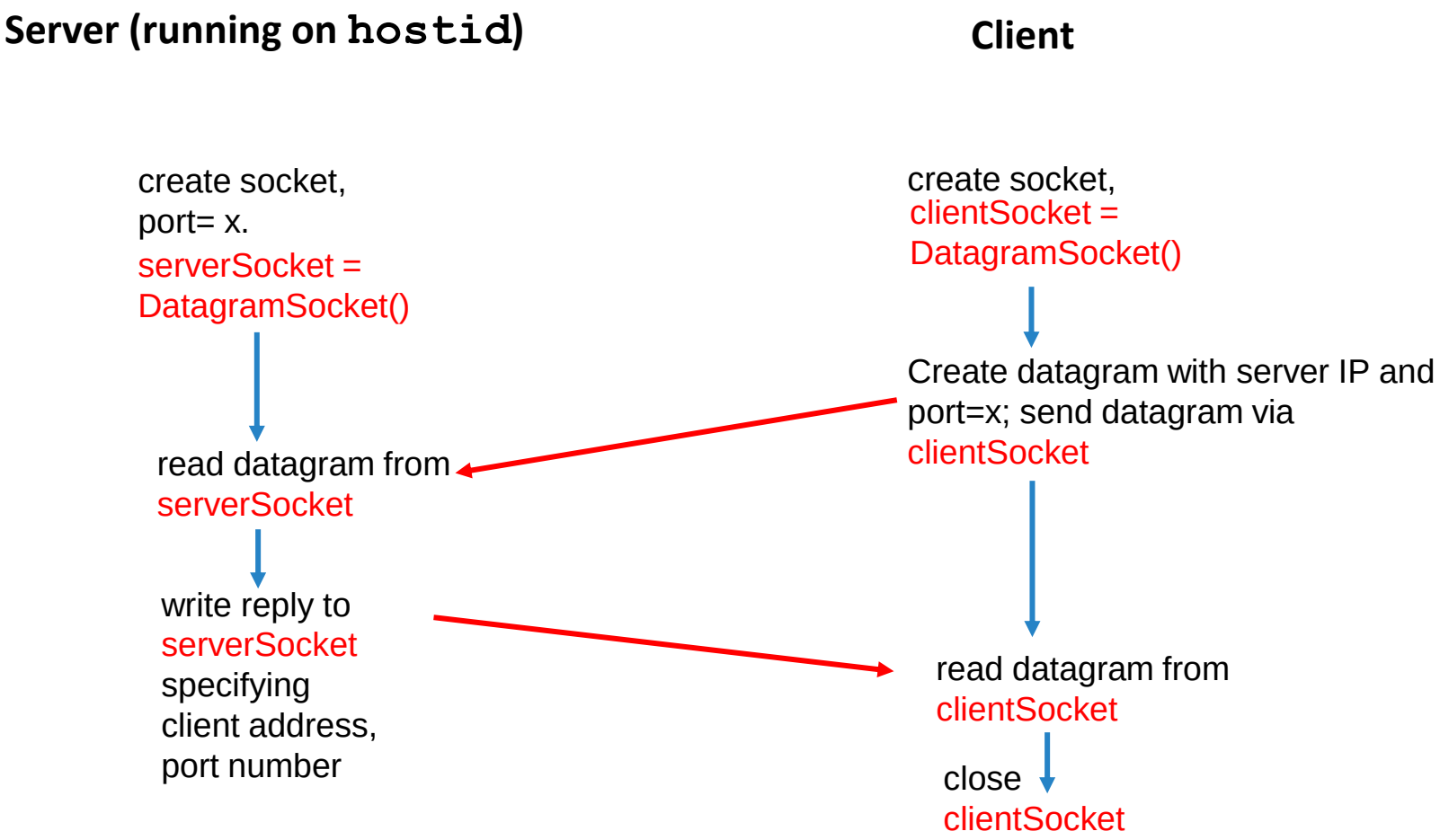
2.8. **Lập trình Socket với UDP**

Lập trình Socket với UDP

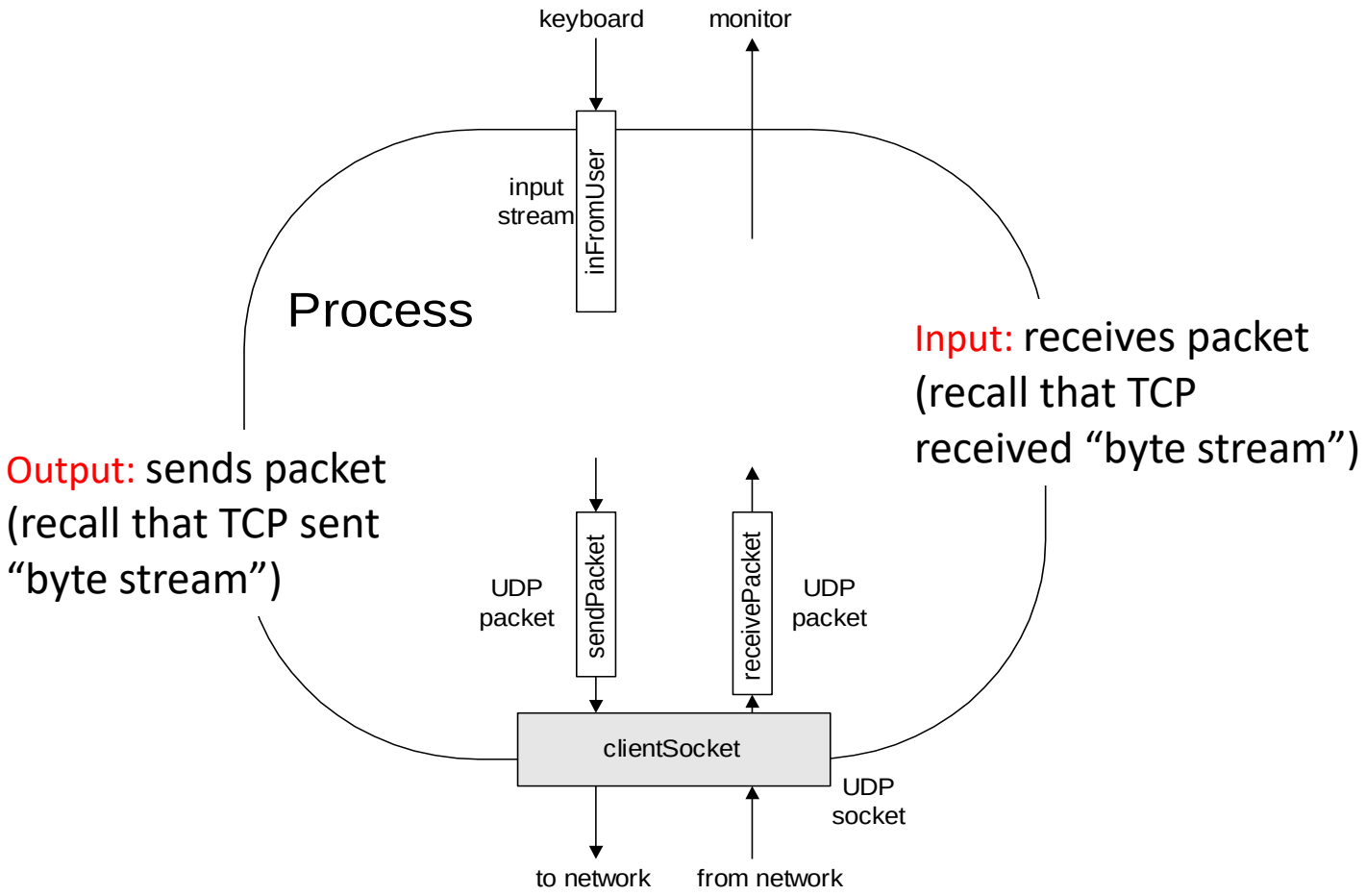
- Không có kết nối giữa Client và Server
  - Không có bước bắt tay
  - Bên gửi đính kèm thông tin về địa chỉ IP và số hiệu cổng trong mỗi gói tin
  - Bên nhận lấy ra địa chỉ IP, số hiệu cổng của người gửi từ mỗi gói tin
- Dữ liệu nhận được có thể không đầy đủ hoặc không đúng thứ tự

Lập trình Socket với UDP (tt)

- Tương tác giữa Server và Client



Lập trình Socket với UDP (tt)



Lập trình Socket với UDP (tt)

- **UDP Server java**

```
import java.io.*;
import java.net.*;

public class UDPServer {
    public static void main(String argv[]) throws Exception
    {
        //Create datagram socket (port 9876)
        DatagramSocket serverSocket = new DatagramSocket(9876);

        while(true)
        {
            //Declare variables
            byte[] receiveData = new byte[1024];
            byte[] sendData = new byte[1024];

            //Create space for received datagram
            DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);

            //Receive datagram
            serverSocket.receive(receivePacket);
```

Lập trình Socket với UDP (tt)

```
                //Get data of sender
                String sentence = new String(receivePacket.getData());

                //Get IP address of sender
                InetAddress IPAddress = receivePacket.getAddress();

                //Get port number of sender
                int port = receivePacket.getPort();

                //Convert line to upper case
                String capitalizedSentence = sentence.toUpperCase();

                //Convert string to byte type
                sendData = capitalizedSentence.getBytes();

                //Create datagram to send to client
                DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, IPAddress, port);

                //Write out datagram to socket
                serverSocket.send(sendPacket);
            } //End of while loop, loop back and wait for another datagram
        }
}
```

Lập trình Socket với UDP (tt)

- **UDP Client java**

```
import java.io.*;
import java.net.*;

public class UDPClient {
    public static void main(String argv[]) throws Exception
    {
        //Create input stream
        BufferedReader inFromServer = new BufferedReader(new
InputStreamReader(System.in));

        //Create client socket
        DatagramSocket clientSocket = new DatagramSocket();

        //Translate hostname to IP address
        InetAddress IPAddress = InetAddress.getByName("127.0.0.1");

        //Declare variables
        byte sendData[] = new byte[1024];
        byte receiveData[] = new byte[1024];

        //Read characters from keyboard
        String Sentence = inFromServer.readLine();
```

Lập trình Socket với UDP (tt)

```
        //Convert string to byte type
        sendData = Sentence.getBytes();

        //Create datagram with data-to-send, length, IP address, port number
        DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length,
IPAddress, 9876);

        //Send datagram to server
        clientSocket.send(sendPacket);

        //Create datagram with data-to-receive
        DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);

        //Read datagram from server
        clientSocket.receive(receivePacket);

        //Get data from datagram from server
        String modifiedSentence = new String(receivePacket.getData());

        //Display
        System.out.println("FROM SERVER: " + modifiedSentence);

        //Close client socket
        clientSocket.close();
    }
}
```