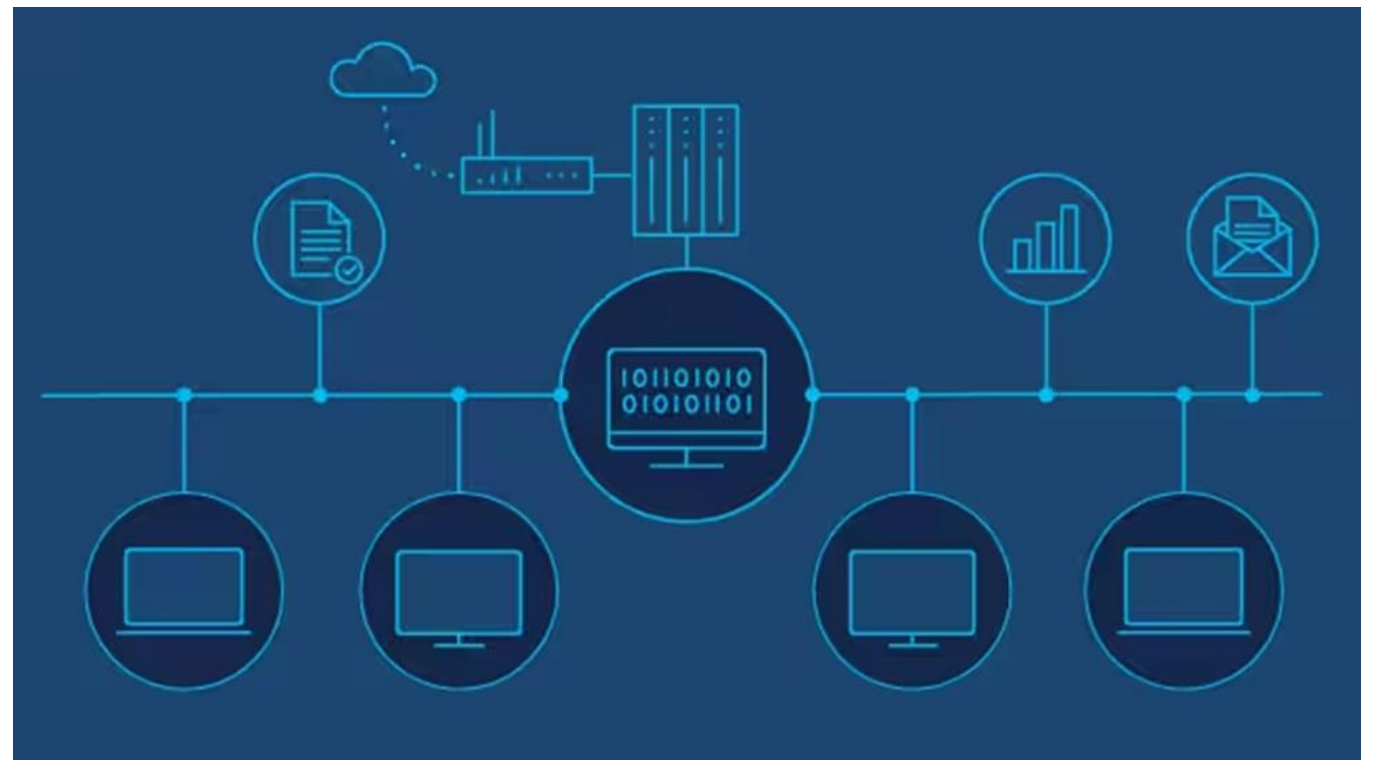


Bài 4 – 6

Lập trình mạng với Web và HTTP



Bài 4 – 6: Lộ trình

- Web và HTTP
 - Bản tin yêu cầu
 - Hoạt động nhóm
- Bộ đệm Web
 - Hoạt động nhóm

Bài 4 – 6: Lộ trình

- Web và HTTP
 - Bản tin yêu cầu
 - Hoạt động nhóm
- Bộ đệm Web
 - Hoạt động nhóm

Web và HTTP

Tổng quát...

- 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 tập tin HTML, ảnh JPEG, tập tin âm thanh,...
- Trang Web chứa *tập tin cơ sở HTML* bao gồm *một số đối tượng được tham chiếu, mỗi* đối tượng có thể được xác định bằng một *URL*, ví dụ:

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

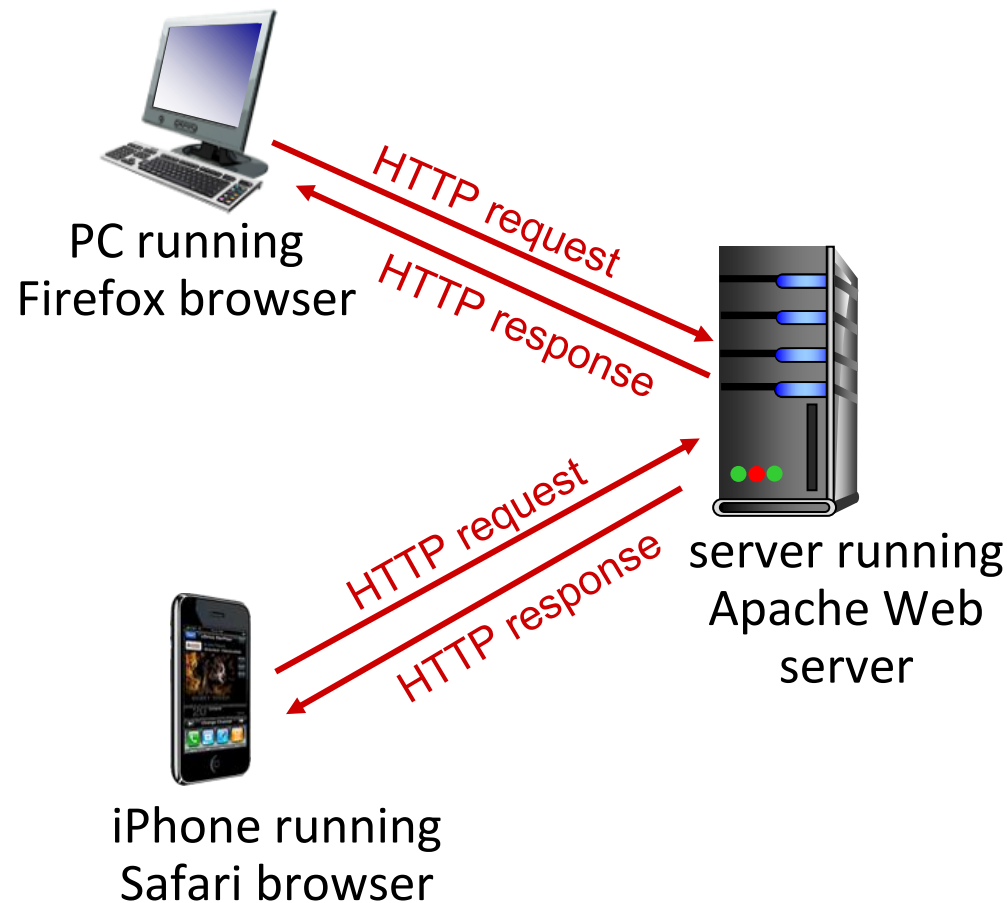
host name

path name

Tổng quan HTTP

HTTP: hypertext transfer protocol

- Giao thức lớp ứng dụng của Web
- Mô hình client/server:
 - **client**: trình duyệt gửi yêu cầu, nhận phản hồi (sử dụng giao thức HTTP) và “hiển thị” các đối tượng Web
 - **server**: máy chủ Web gửi (sử dụng giao thức HTTP) các đối tượng để phản hồi lại Client



Tổng quan HTTP (tiếp)

HTTP sử dụng TCP:

- Client khởi tạo kết nối TCP (tạo socket) tới server, cổng 80
- Server chấp nhận kết nối TCP từ client
- Bản tin HTTP (bản tin giao thức lớp ứng dụng) được trao đổi giữa trình duyệt (HTTP client) và máy chủ Web (HTTP server)
- Kết nối TCP được đóng (ngắt)

HTTP là “không trạng thái”

- Server *không* duy trì thông tin về các yêu cầu trước đó của client

Kết nối HTTP: Hai loại

HTTP không bền vững

1. Kết nối TCP được mở
2. Nhiều nhất **một** đối tượng được gửi qua một kết nối TCP
3. Kết nối TCP được đóng

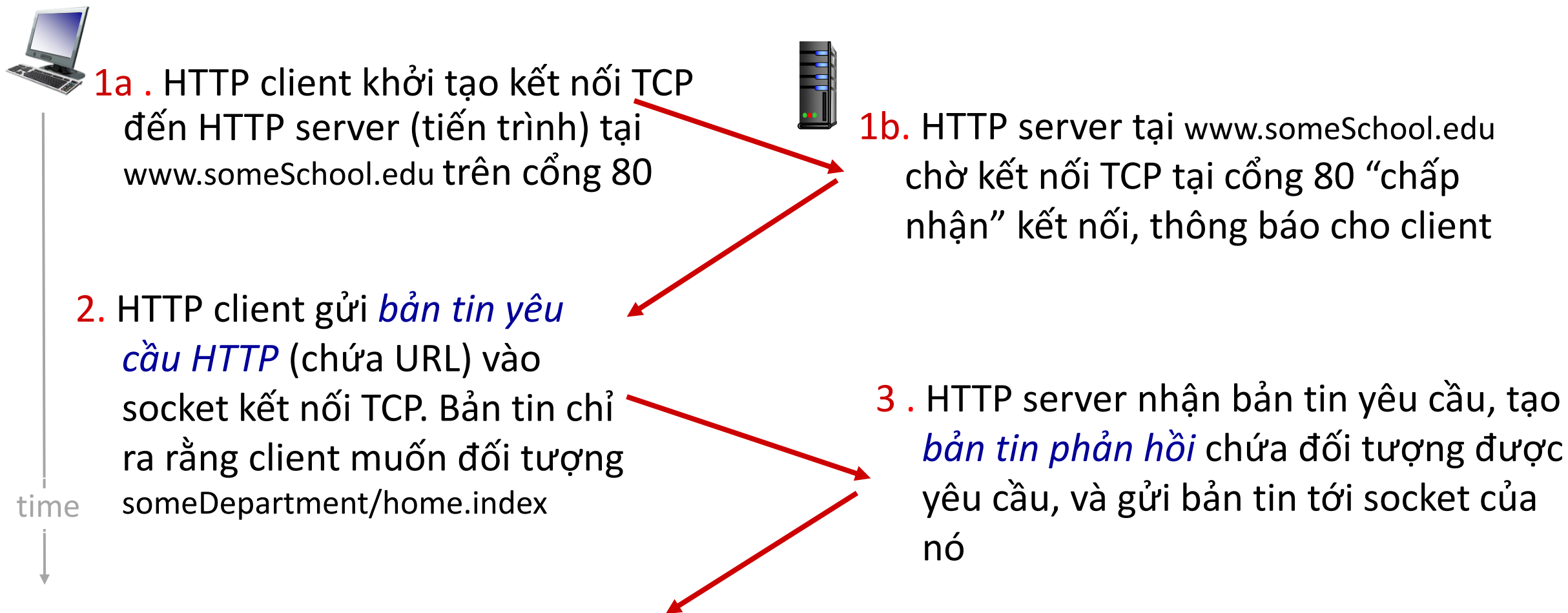
Tải xuống **nhiều** đối tượng yêu cầu **nhiều** kết nối TCP

HTTP bền vững

- Kết nối TCP được mở
- Nhiều đối tượng có thể được gửi qua **một** kết nối TCP duy nhất giữa client và server
- Kết nối TCP được đóng

HTTP không bền vững: Ví dụ

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



HTTP không bền vững: Ví dụ (tiếp)

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



time
↓

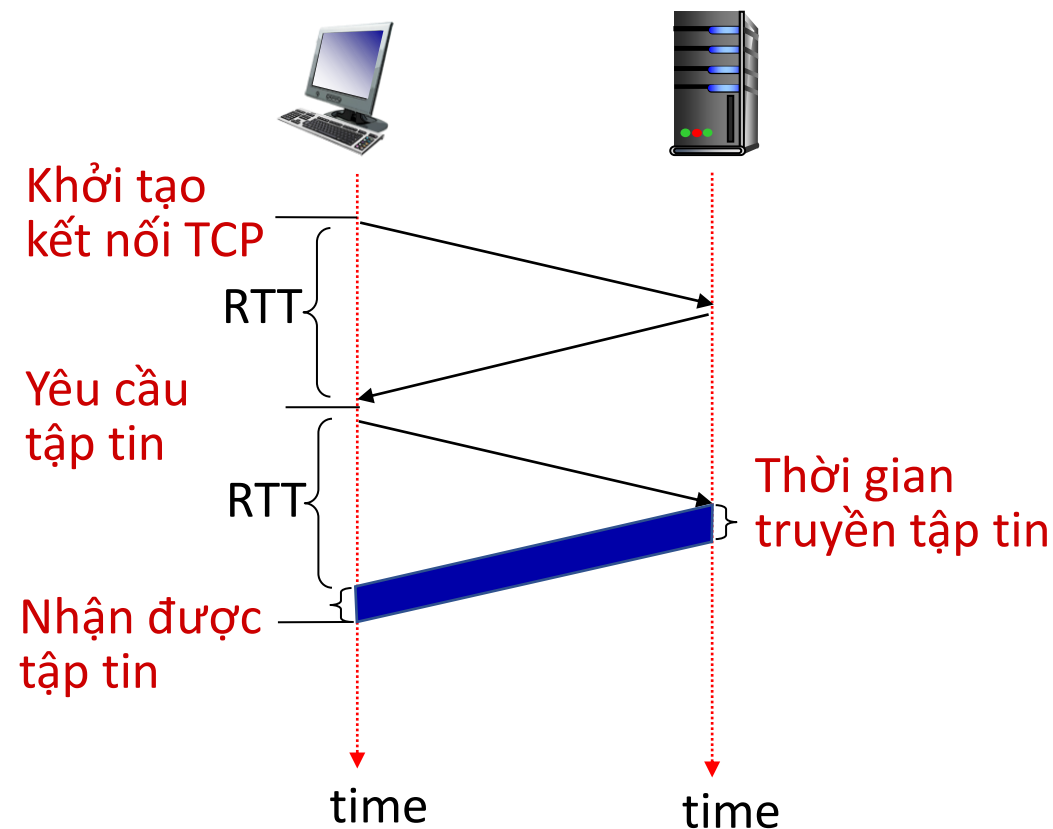
5. HTTP client nhận *bản tin phản hồi* chứa tập tin html, hiển thị tập tin html. Phân tích tập tin html, tìm ra 10 đối tượng hình ảnh jpeg được tham chiếu
 6. Các bước 1-5 được lặp lại đối với từng đối tượng để có thể lấy được 10 hình ảnh jpeg
4. HTTP server đóng kết nối TCP

HTTP không bền vững: Thời gian phản hồi

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

Thời gian phản hồi HTTP (trên 1 đối tượng):

- Một RTT để khởi tạo kết nối TCP
- Một RTT cho yêu cầu HTTP và một vài byte đầu tiên của phản hồi HTTP được trả về
- Thời gian truyền đối tượng/file



Thời gian phản hồi HTTP không bền vững = $2RTT +$ thời gian truyền tập tin

HTTP bền vững (HTTP 1.1)

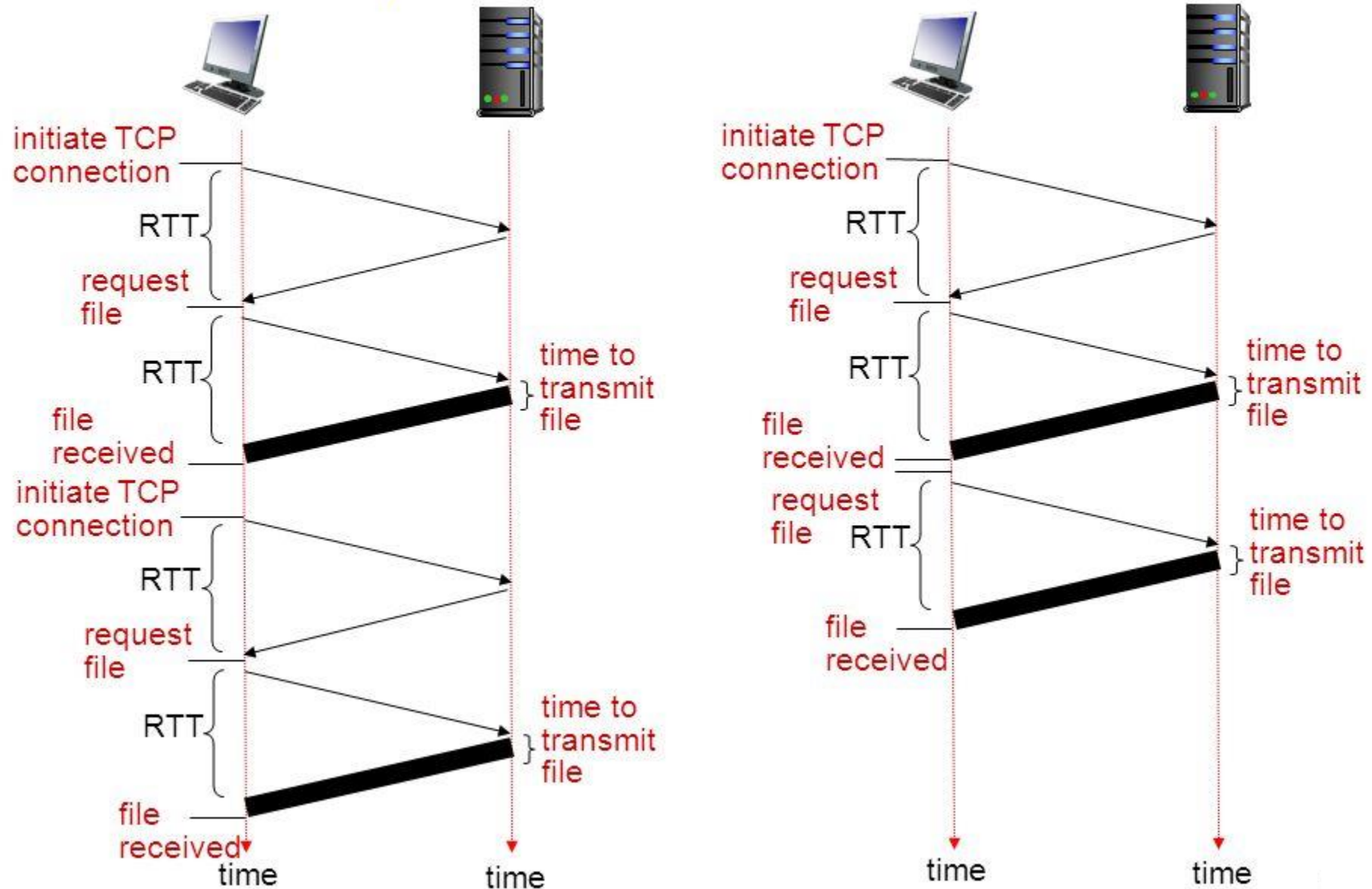
Vấn đề của HTTP không bền vững

- Yêu cầu 2 RTT cho mỗi đối tượng
- Tốn tài nguyên của hệ điều hành khi xử lý mỗi kết nối TCP
- Các trình duyệt thường mở nhiều kết nối TCP song song để yêu cầu các đối tượng được tham chiếu song song

HTTP bền vững (HTTP 1.1):

- Server để lại kết nối mở sau khi gửi phản hồi tới Client
- Bản tin HTTP tiếp theo giữa client và server được gửi qua kết nối đã mở ở trên
- Client gửi phản hồi ngay khi nó nhận đối tượng được tham chiếu
- Chỉ cần một RTT cho tất cả đối tượng được tham chiếu (giảm một nửa thời gian phản hồi)

Non-persistent HTTP v.s. Persistent HTTP for Two Objects



Bài 4 – 6: Lộ trình

- Web và HTTP
 - Bản tin yêu cầu
 - Hoạt động nhóm
- Bộ đệm Web
 - Hoạt động nhóm

Bản tin yêu cầu HTTP

- Có 2 kiểu bản tin HTTP: *yêu cầu (request), phản hồi (response)*
- **Bản tin yêu cầu HTTP:**
 - ASCII (human-readable format)

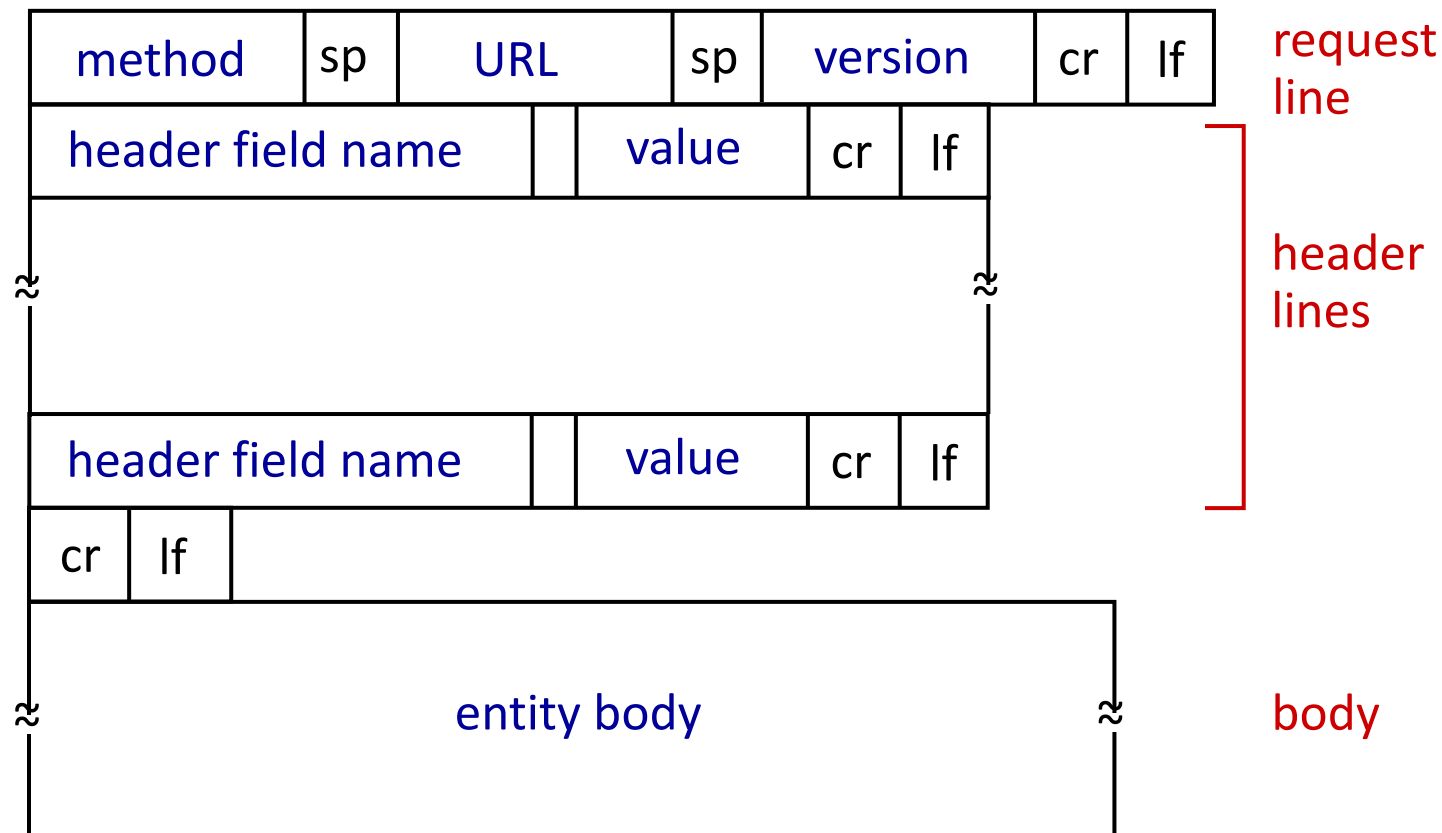
Dòng yêu cầu (lệnh GET,
POST, HEAD) →

/ ký tự xuống dòng (carriage return)
/ ký tự về đầu dòng (line-feed)

Ký tự xuống dòng, về đầu →
dòng mới để chỉ điểm cuối
cùng của bản tin

* Check out the online interactive exercises for more
examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

Bản tin yêu cầu HTTP: Định dạng chung



Bản tin yêu cầu HTTP (tiếp)

Phương thức POST:

- Trang Web thường bao gồm biểu mẫu để nhập dữ liệu đầu vào
- Dữ liệu vào của người dùng được gửi từ client đến server trong phần thân của bản tin yêu cầu HTTP POST

Phương thức GET (gửi dữ liệu tới server):

- Gồm dữ liệu người dùng trong trường URL của bản tin yêu cầu HTTP GET (sau dấu '?'):
`www.somesite.com / animalsearch?monkeys&banana`


Phương thức HEAD:

- Tiêu đề yêu cầu sẽ được trả về nếu URL được chỉ định với phương thức HTTP GET.

Phương thức PUT:

- Tải tập tin mới (đối tượng) đến server
- Thay thế hoàn toàn tập tin tồn tại tại URL chỉ định với nội dung trong phần thân của bản tin yêu cầu POST HTTP

Bản tin phản hồi HTTP

Dòng trạng thái (giao thức,  HTTP/1.1 200 OK
mã trạng thái)

Mã trạng thái phản hồi HTTP

- Mã trạng thái xuất hiện ở dòng đầu tiên trong bản tin phản hồi
- Một số mã trạng thái mẫu:

200 OK

- Yêu cầu thành công, đối tượng được yêu cầu ở trong bản tin này

301 Moved Permanently

- Đối tượng yêu cầu đã được di chuyển, vị trí mới được xác định sau trong bản tin này (in Location: field)

400 Bad Request

- Server không hiểu bản tin yêu cầu

404 Not Found

- Thông tin được yêu cầu không tìm thấy trên server này

505 HTTP Version Not Supported

Bài 4 – 6: Lộ trình

- Web và HTTP
 - Bản tin yêu cầu
 - Hoạt động nhóm
- Bộ đệm Web
 - Hoạt động nhóm

Hoạt động nhóm số 1



- Truy xuất nội dung trang Web tại địa chỉ sau:

<http://gaia.cs.umass.edu/wireshark-labs/INTRO-wireshark-file1.html>

- Giải pháp 1: Sử dụng trình duyệt Web, ví dụ Google Chrome
- Giải pháp 2: Viết chương trình Client dựa trên lập trình mạng
- Sử dụng Wireshark để bắt và phân tích gói tin

Lời giải: Hoạt động nhóm số 1



```
import socket
serverIP = "128.119.245.12"
serverPort = 80
maxBytes = 4096

request_message = """\
GET /wireshark-labs/INTRO-
wireshark-file1.html
HTTP/1.1\r\n\
Host: gaia.cs.umass.edu\r\n\
User-Agent: Group work
1\r\n\
Connection: keep-alive\r\n\
Accept-Language: vn\r\n\
\r\n\
"""
```

```
sock = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
sock.connect((serverIP, serverPort))

sock.send(request_message.encode())
responseMessage = sock.recv(maxBytes)
sock.close()

responseMessage =
responseMessage.decode()
print(responseMessage)
```

Hoạt động nhóm số 2



- Viết **chương trình Web Server** chạy trên localhost để xử lý yêu cầu HTTP của Client.
- Client là trình duyệt Web. Client hiển thị nội dung trong tập tin “Helloworld.html” chứa văn bản “**Lap trình mạng**”.
- Socket của Server gửi nhận tối đa **4096** bytes tại một thời điểm.
- **Nếu không có** tập tin “Helloworld.html” trong bộ nhớ lưu trữ của Server thì Client hiển thị “Khong tim thay du lieu trong bo nho cua Server”.
- Sử dụng Wireshark để bắt và phân tích gói tin

Lời giải: Hoạt động nhóm số 2



Trình duyệt nhập địa chỉ sau:
127.0.0.10:1000/Helloworld.html

```
import socket

serverIP =
"127.0.0.10"
serverPort = 10000
maxBytes = 4096

sock =
socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
sock.bind((serverIP,
serverPort))
sock.listen()
```

```
while True:
    connectionSocket, address = sock.accept()
    message = connectionSocket.recv(maxBytes)
    try:
        filename = message.decode().split()[1]
        with open(filename[1:], mode='rt',
encoding='utf-8') as f:
            data = f.read()
            responseMessage = f"HTTP/1.1 200
OK\r\n\r\n{data}"
    except:
        data = "<p>Khong tim thay du lieu trong bo
nho cua Server</p>"
        responseMessage = f"HTTP/1.1 404 Not
Found\r\n\r\n{data}"
    pass
    connectionSocket.send(responseMessage.encode())
    connectionSocket.close()
```

Hoạt động nhóm số 3



- Viết **chương trình Web Server** như Hoạt động nhóm số 3
- Viết **chương trình Client** thay vì sử dụng trình duyệt Web
- Sử dụng Wireshark để bắt và phân tích gói tin

Lời giải: Hoạt động nhóm số 3



```
import socket
serverIP = "127.0.0.10"
serverPort = 1000
maxBytes = 4096
```

```
request_message = """\
GET /HelloWorld.html
HTTP/1.1\r\n\
Host: localhost\r\n\
User-Agent: Group work
3\r\n\
\r\n\
"""
```

```
sock =
socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
sock.connect((serverIP,
serverPort))

sock.send(request_message.encode(
))
responseMessage =
sock.recv(maxBytes)
sock.close()
```

```
responseMessage =
responseMessage.decode()
print(responseMessage)
```

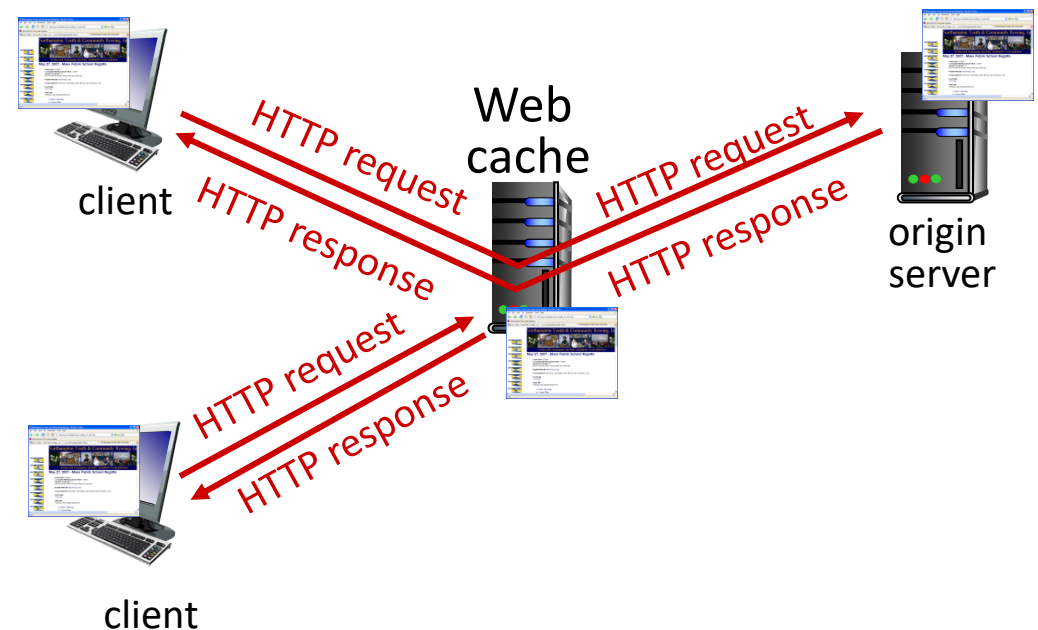
Bài 4 – 6: Lộ trình

- Web và HTTP
 - Bản tin yêu cầu
 - Hoạt động nhóm
- Bộ đệm Web
 - Hoạt động nhóm

Bộ đệm Web (Web caches)

Mục tiêu: đáp ứng các yêu cầu của client mà không liên quan đến server nguồn

- Người dùng cấu hình trình duyệt để trở đến **bộ đệm Web (cục bộ)**
- Trình duyệt gửi tất cả các yêu cầu HTTP đến bộ đệm Web
 - *Nếu* đối tượng **không có** trong bộ đệm: Web cache yêu cầu đối tượng từ server nguồn, sau đó gửi đối tượng cho client
 - *Nếu* đối tượng **có** trong bộ đệm: Web cache gửi đối tượng cho client



Bộ đệm Web (aka Proxy servers)

- Bộ đệm Web hoạt động với hai vai trò:
 - Server: phản hồi yêu cầu của client
 - Client yêu cầu đến server nguồn
- Server nguồn thông báo cho Web cache về bộ nhớ đệm được phép của đối tượng's trong tiêu đề phản hồi

```
Cache-Control: max-age=<seconds>
```

```
Cache-Control: no-cache
```

Tại sao phải dùng bộ đệm Web?

- Giảm thời gian phản hồi cho yêu cầu của client
 - Bộ đệm gần client hơn
- Giảm lưu lượng trên đường liên kết truy cập của tổ chức tới server nguồn
- Internet có rất nhiều bộ đệm Web
 - Cho phép các nhà cung cấp nội dung với lượng tài nguyên “nghèo nàn” vẫn cung cấp nội dung một cách hiệu quả

Bài 4 – 6: Lộ trình

- Web và HTTP
 - Bản tin yêu cầu
 - Hoạt động nhóm
- Bộ đệm Web
 - Hoạt động nhóm

Hoạt động nhóm số 4



- Viết **chương trình Web Server** và **Proxy Server** chạy trên localhost để xử lý yêu cầu HTTP của Client.
- Client là trình duyệt Web. Client hiển thị nội dung trong tập tin “Helloworld.html” chứa văn bản “**Lap trình mạng**”.
- Socket của Server gửi nhận tối đa **4096** bytes tại một thời điểm.
- **Nếu Proxy Server và WebServer không có** tập tin “Helloworld.html” thì Client hiển thị “Khong tim thay du lieu tu may chu goc va tam thoi”.

Hoạt động nhóm số 5



- Viết **chương trình Web Server, Proxy Server** và **Client** chạy trên localhost để xử lý yêu cầu HTTP của Client.
- Client là trình duyệt Web. Client hiển thị nội dung trong tập tin “Helloworld.html” chứa văn bản “**Lap trình mạng**”.
- Socket của Server gửi nhận tối đa **4096** bytes tại một thời điểm.
- **Nếu Proxy Server và WebServer không có** tập tin “Helloworld.html” thì Client hiển thị “Khong tim thay du lieu tu may chu goc va tam thoi”.

Bài 4 – 6: Tổng kết

- Web và HTTP
 - Bản tin yêu cầu
 - Hoạt động nhóm
- Bộ đệm Web
 - Hoạt động nhóm

Thank you

