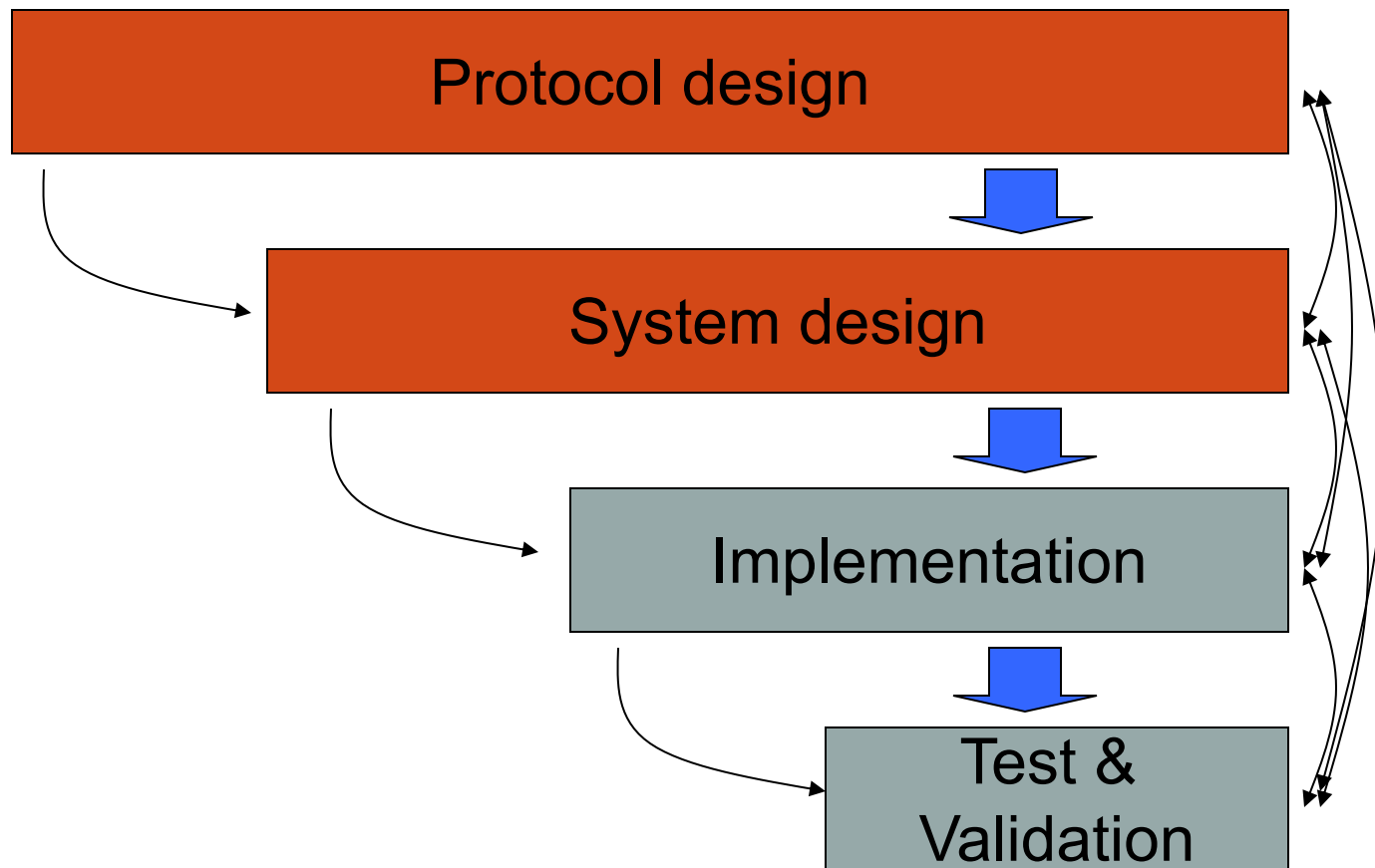


# Thiết kế ứng dụng mạng

# Nội dung bài học

- Mô hình truyền thông
- Thiết kế giao thức
- Đánh giá giao thức

# Các bước thiết kế, xây dựng ứng dụng mạng



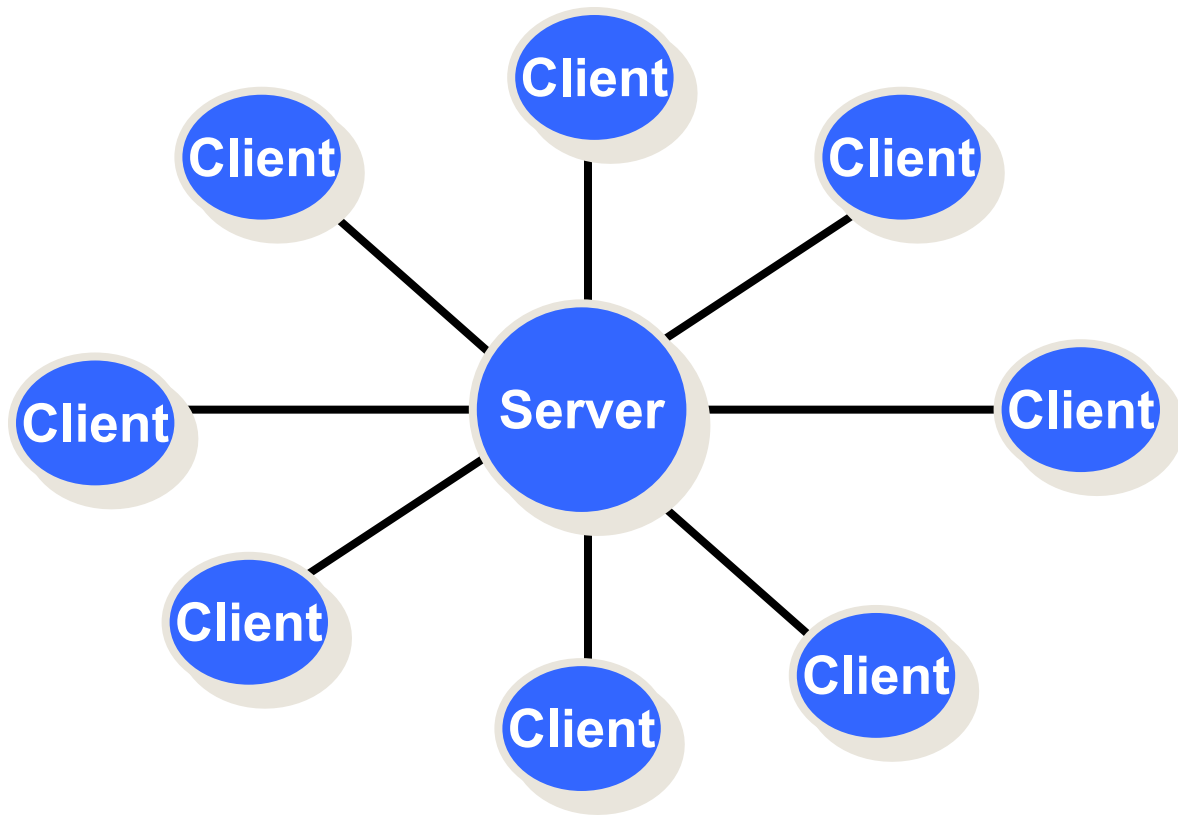
# Các bước thiết kế ứng dụng mạng

- Thiết lập mục tiêu
- Lựa chọn mô hình truyền thông
- Lựa chọn định dạng thông báo
- Thiết kế cấu trúc thông báo: định dạng, các trường, kiểu thông báo, ...
- Thiết kế các luật (bước) truyền thông

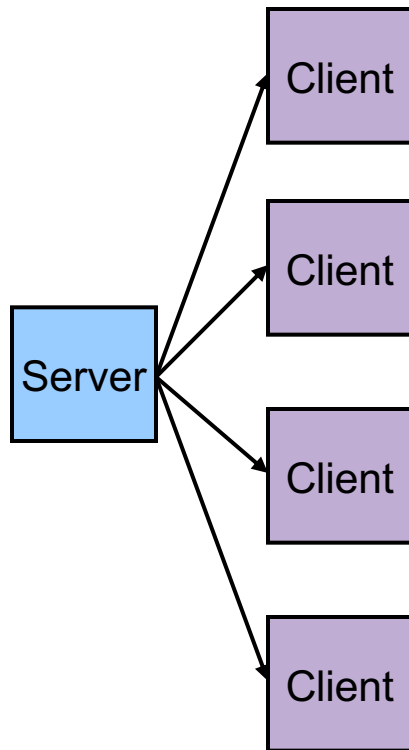
# Lựa chọn mô hình truyền thông

- Mô hình khách chủ
- Mô hình ngang hàng
- Mô hình lai

# Mô hình Client/Server



# Mô hình truyền tin khách/chủ



- Truyền tin bất đối xứng
  - ❑ Máy khách gửi yêu cầu
  - ❑ Máy chủ gửi trả lời
- Máy chủ
  - ❑ Sử dụng định dạng đã biết (e.g., IP address + port)
  - ❑ Đợi kết nối đến
  - ❑ Xử lý yêu cầu, gửi trả lời
- Máy khách
  - ❑ Bắt đầu một kết nối
  - ❑ Đợi trả lời từ máy chủ

# Mô hình truyền tin khách chủ (2)

- Mô hình dịch vụ
  - Xử lý tuần tự:
    - Máy chủ chỉ tạo một kết nối và xử lý yêu cầu của một máy khách tại mỗi thời điểm
      - E.g. Time server
  - Xử lý đồng thời:
    - Máy chủ tạo nhiều kết nối với các máy khách và xử lý các yêu cầu của các máy khách khác nhau cùng một lúc
      - E.g. Web server
  - Lai:
    - Máy chủ tạo ra nhiều kết nối với nhiều máy khách, nhưng xử lý yêu cầu của các máy khách một cách tuần tự
- Không có ranh giới rõ ràng giữa hai khái niệm máy khách và máy chủ
  - Một máy chủ có thể là máy khách của một máy chủ khác
  - Một máy chủ có thể là máy khách của chính máy khách của nó

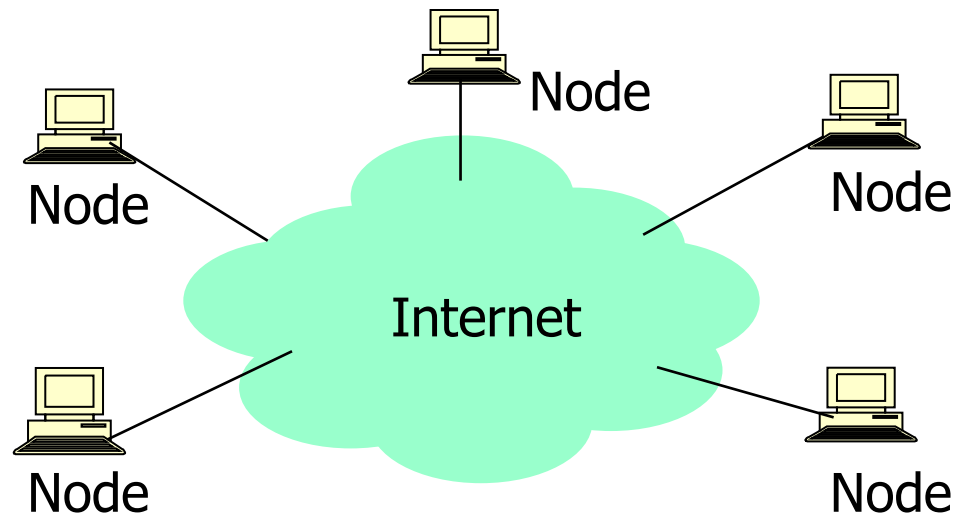


# Đặc điểm của mô hình truyền tin khách/chủ

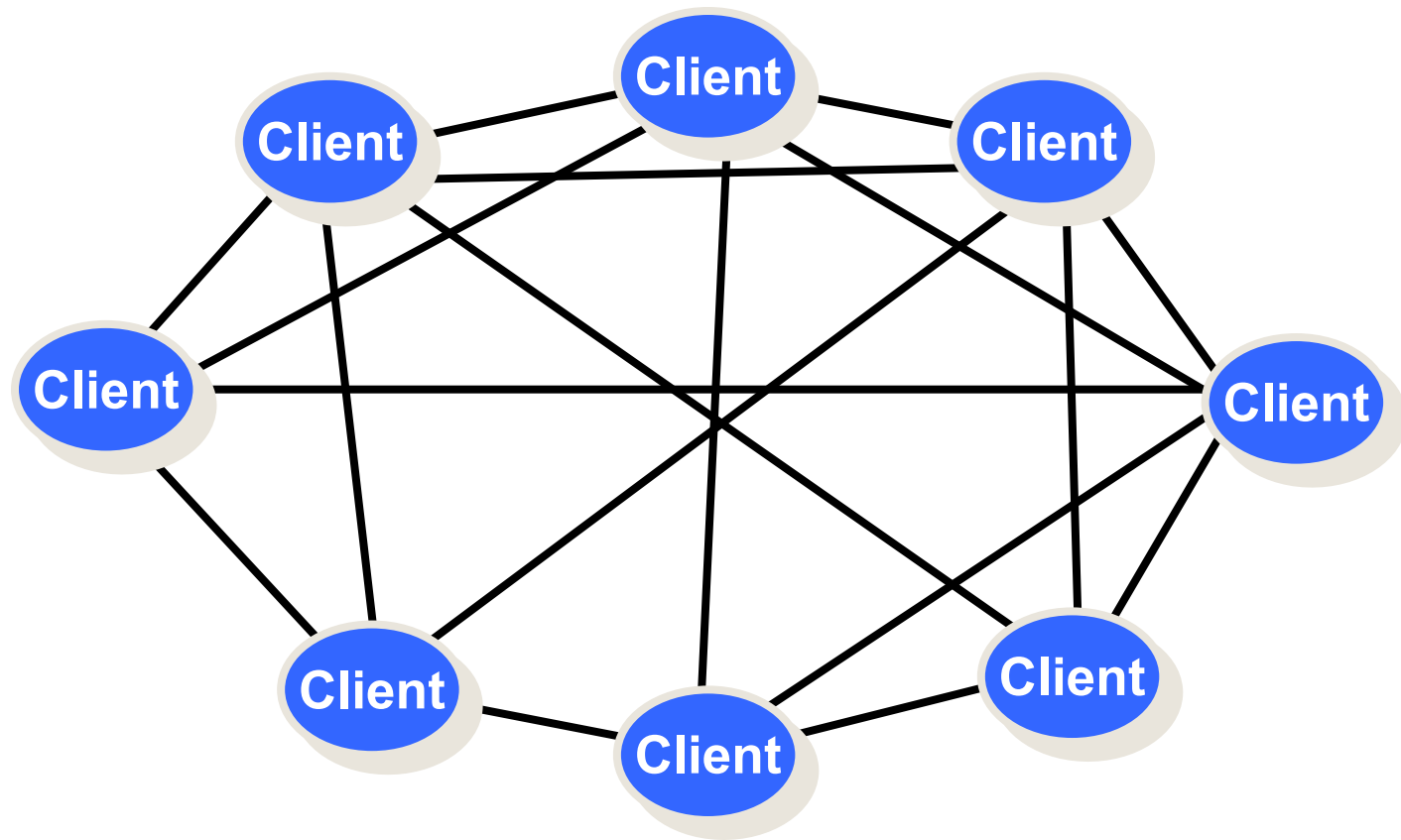
- Dễ dàng trong việc thiết kế
  - máy khách không cần biết tất cả các máy khách còn lại
  - quản lý bảo mật tập trung
- Máy chủ có thể bị quá tải
  - Ví dụ: với 10.000 truy cập một lúc
  - Khả năng mở rộng bị hạn chế -> Data center, cloud
  - Chi phí đầu tư máy chủ cao

# Mô hình truyền tin ngang hàng

- Các node trong mạng đóng cả hai vai trò máy chủ/máy khách
  - Cung cấp và sử dụng tài nguyên
  - Bất cứ node nào cũng có thể khởi tạo kết nối
- Không có máy chủ dữ liệu trung tâm
  - “The ultimate form of democracy on the Internet”
  - “The ultimate threat to copy-right protection on the Internet”



# Mô hình mạng ngang hàng



# Mô hình mạng ngang hàng

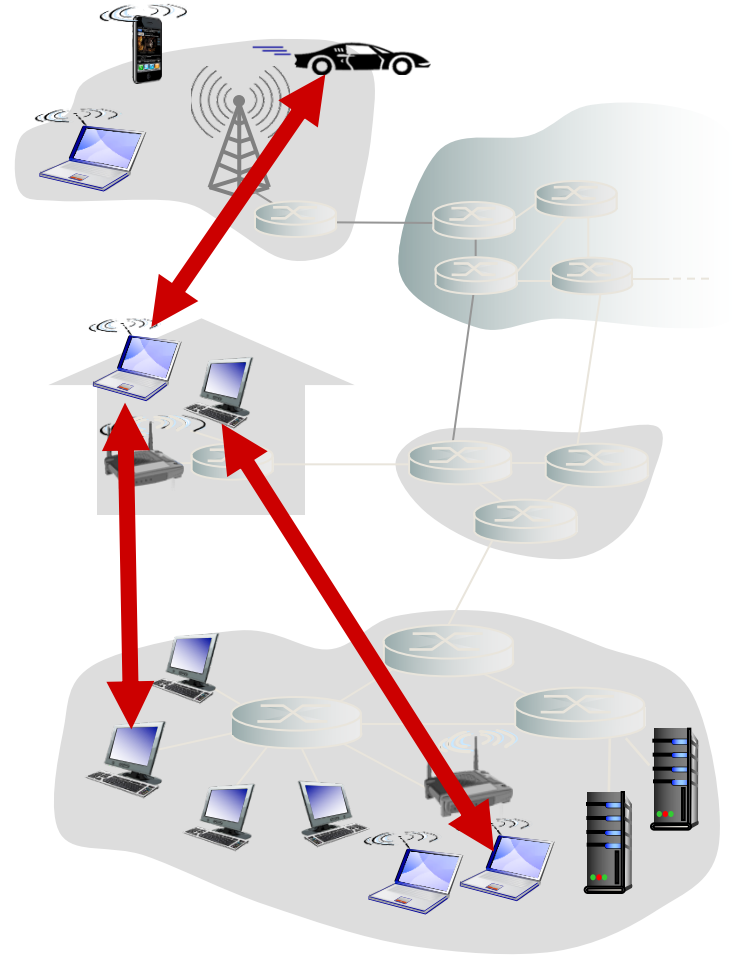
- No server, no client
  - Tất cả đều có vai trò như nhau
  - Không có quả tải ở server
- Không có điểm yếu
  - Nhưng có nhiều điểm yếu
- Có thể triển khai trên diện rộng
- Chi phí thấp
  - Xử lý chủ yếu tại máy tính người dùng
- Khó quản lý

# Mạng ngang hàng thuần túy

- Không có always-on server
- Các máy tính truyền dữ liệu trực tiếp cho nhau một cách tùy ý
- Các máy tính có thể ra vào mạng và thay đổi địa chỉ IP

## Ví dụ:

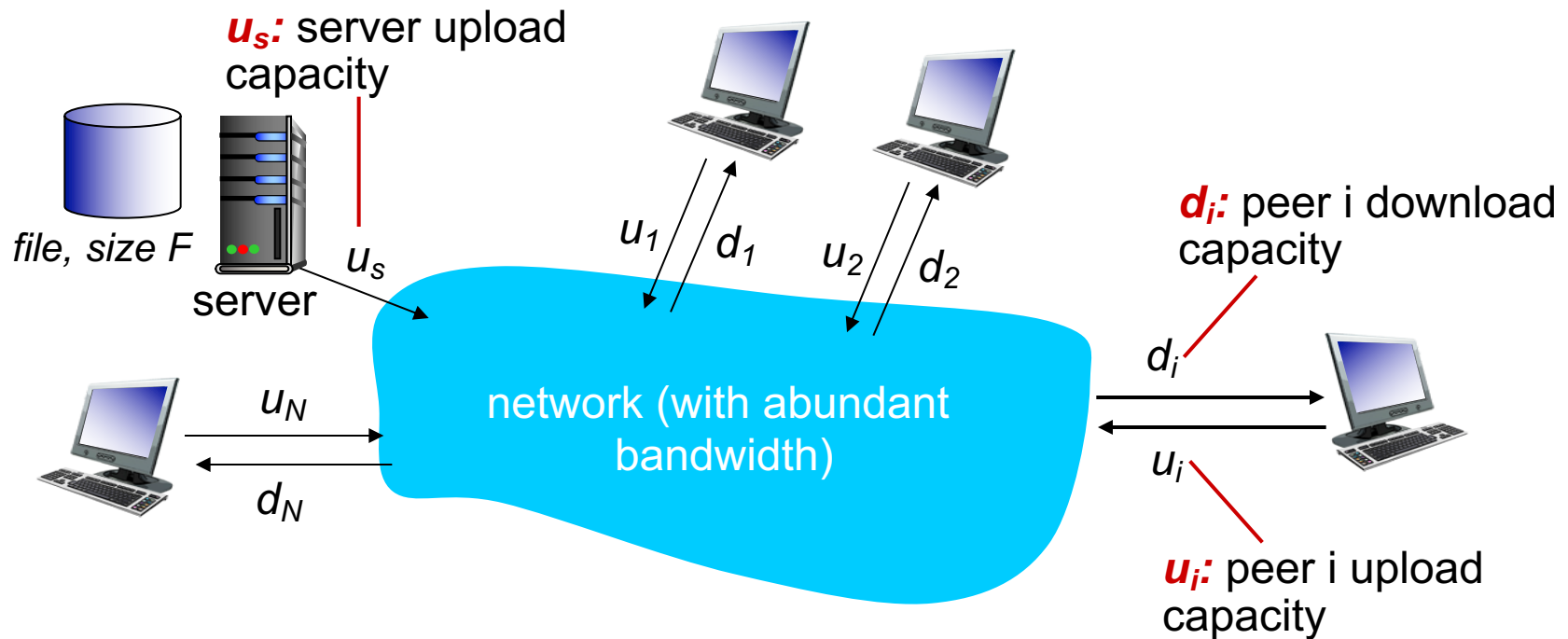
- file distribution (BitTorrent)
- Streaming (Sopcast)
- VoIP (Skype)



# Chia sẻ file: client-server vs P2P

**Question:** Mất bao nhiêu thời gian để chia sẻ 1 file (kích thước  $F$ ) từ 1 máy chủ đến  $N$  máy tính?

- Băng thông gửi/nhận của các máy tính là có giới hạn



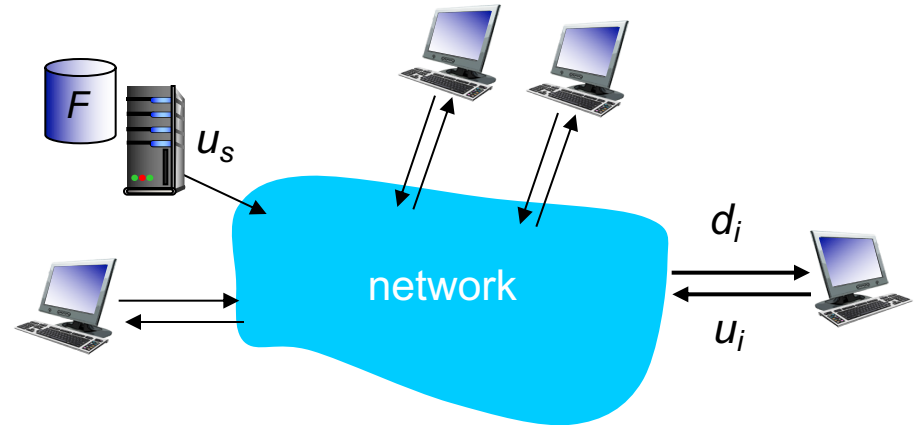
# Thời gian truyền file: mô hình client-server

- **Máy chủ truyền file:** truyền lần lượt  $N$  bản copy của file :

- Thời gian gửi 1 bản copy:  $F/u_s$
- Thời gian gửi  $N$  bản copies:  $NF/u_s$

- ❖ **client:** mỗi client phải download file copy

- $d_{\min}$  = min client download rate
- min client download time:  $F/d_{\min}$



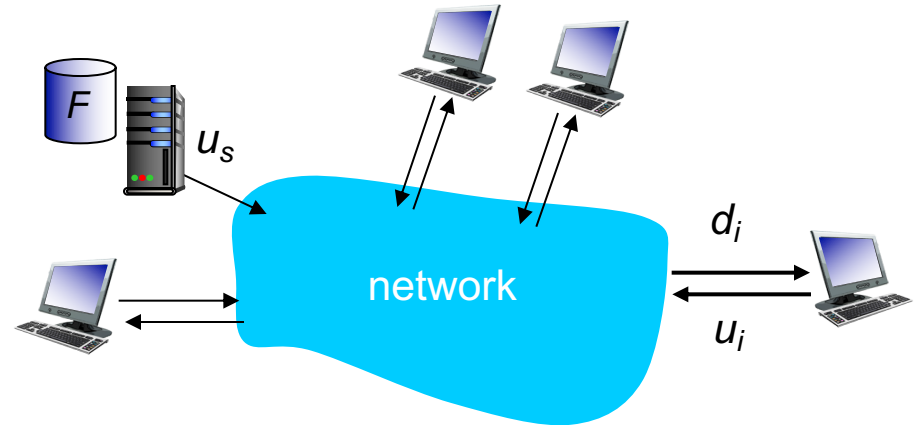
*time to distribute  $F$   
to  $N$  clients using  
client-server approach*

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

increases linearly in  $N$

# Thời gian truyền file: mô hình P2P

- **Truyền dữ liệu từ máy chủ:** Phải tải ít nhất một bản copy
  - Thời gian để gửi một bản copy:  $F/u_s$
- **client:** mỗi client phải download file copy
  - Thời gian tải tối thiểu của client:  $F/d_{\min}$



- ❖ **clients:** phải tải tổng dữ liệu là  $NF$  bits
  - Tốc độ upload tối đa là  $u_s + \sum u_i$

Thời gian để phân phối  
 $F$  đến  $N$  clients  
sử dụng P2P

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

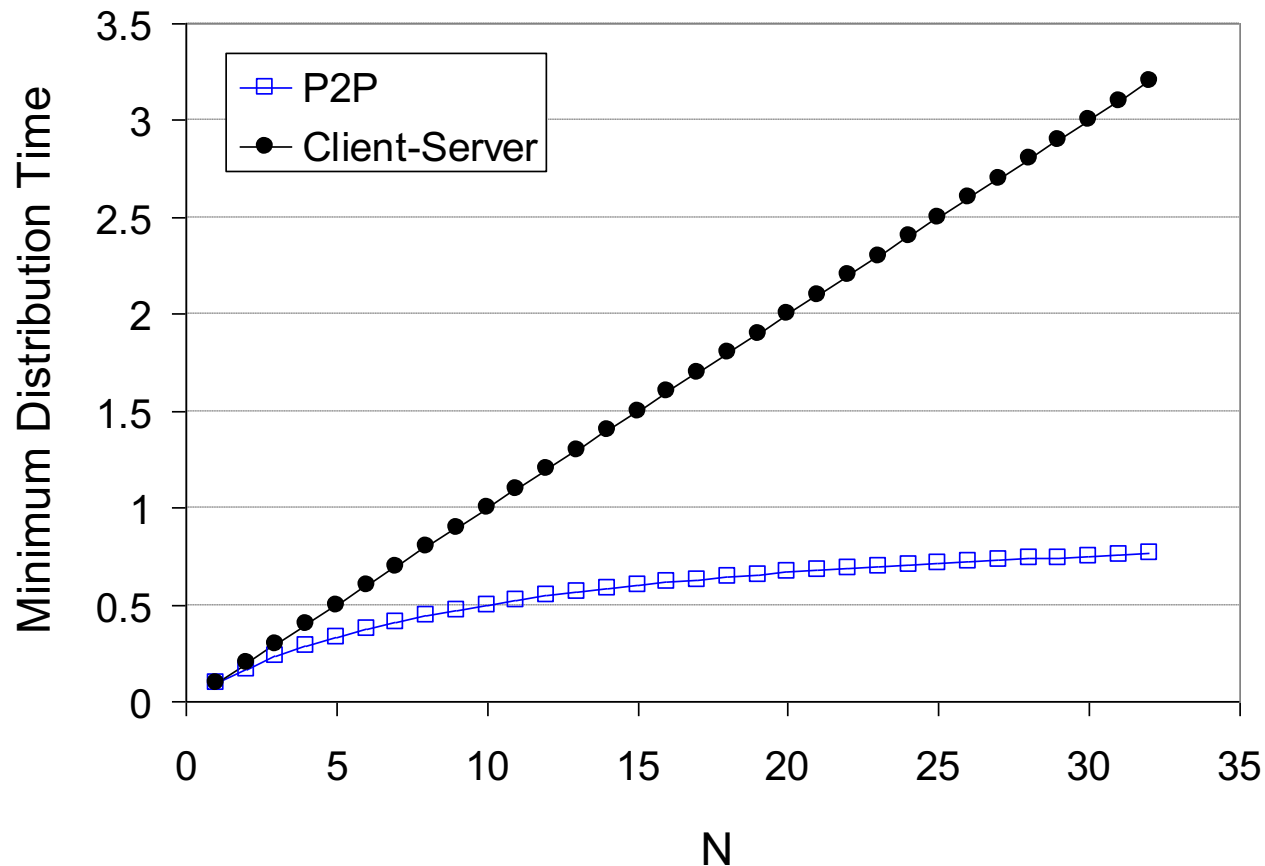
Tăng tuyến tính với  $N$  ...

... nhưng cũng tăng với  $N$ , vì mỗi peer đều cung cấp băng thông



# So sánh mô hình Client-server và P2P

client upload rate =  $u$ ,  $F/u = 1$  hour,  $u_s = 10u$ ,  $d_{min} \geq u_s$

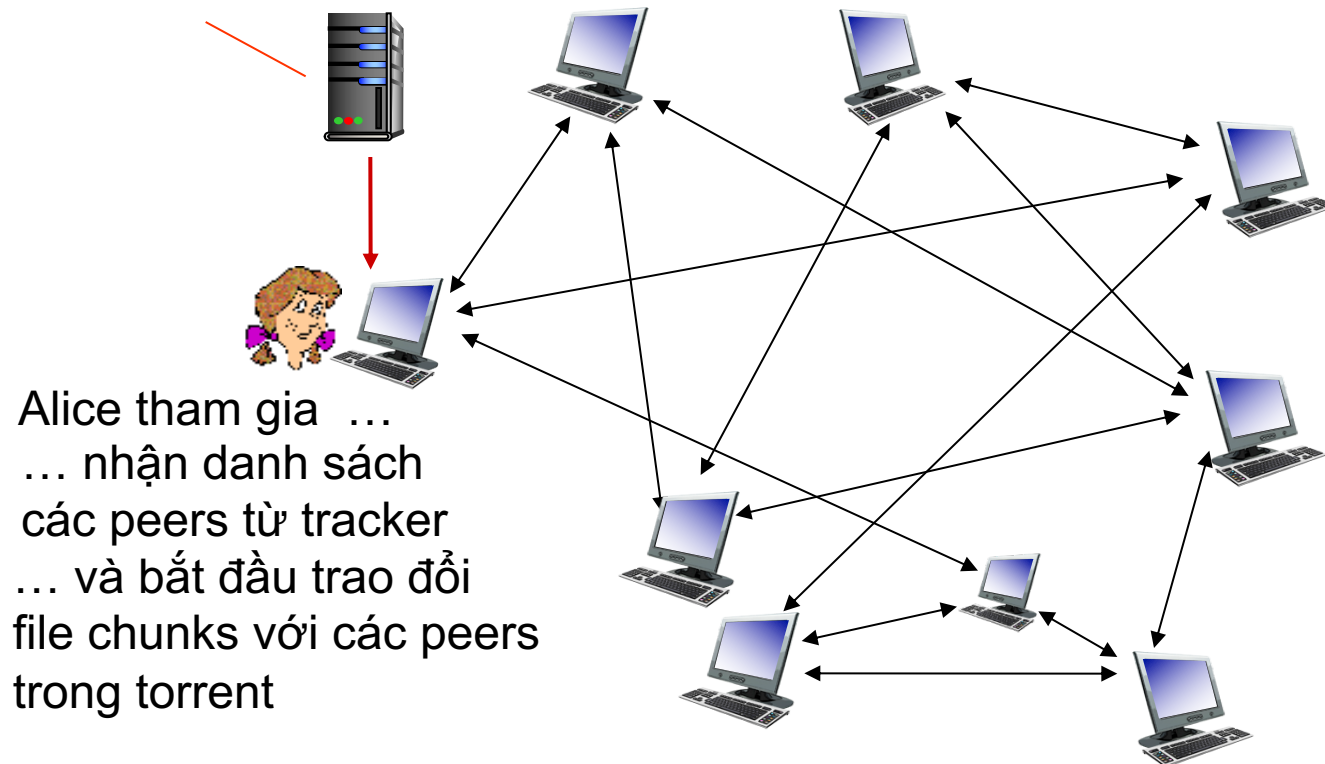


# Phân phối file P2P: BitTorrent

- ❖ file được chia làm các mảnh (chunks) 256Kb
- ❖ peers gửi/nhận các file chunks

**tracker:** theo dõi các peers tham gia vào một torrent

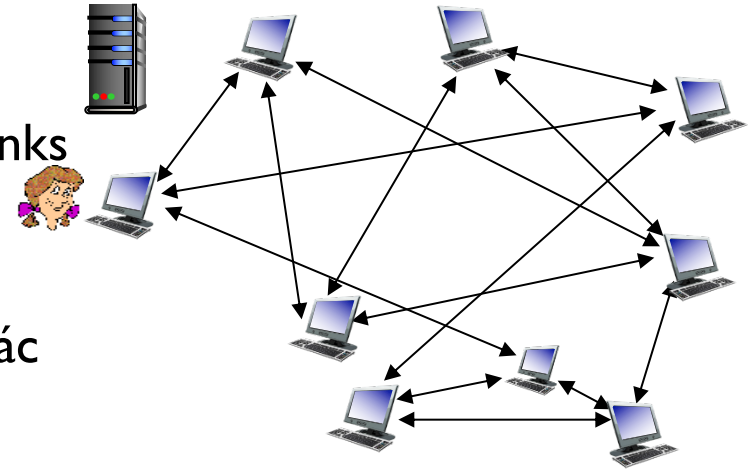
**torrent:** nhóm các peers trao đổi chunks của 1 file



# Phân phối file P2P: BitTorrent

- peer tham gia torrent:

- Không có chunks, nhưng tích lũy chunks theo thời gian từ các peer khác
- Đăng ký với tracker để lấy danh sách các peers và kết nối với một nhóm các peer khác (“các nút hàng xóm”)



- ❖ Trong khi download, peer uploads chunks cho các peer khác
- ❖ peer có thể thay đổi peers mà nó trao đổi chunks
- ❖ **churn**: peers có thể tham gia hoặc rời khỏi mạng
- ❖ Một khi peer có toàn bộ file, nó có thể rời mạng (người dùng ích kỷ) hoặc lưu lại trong torrent

# BitTorrent: Yêu cầu, gửi file chunks

## *Yêu cầu chunks:*

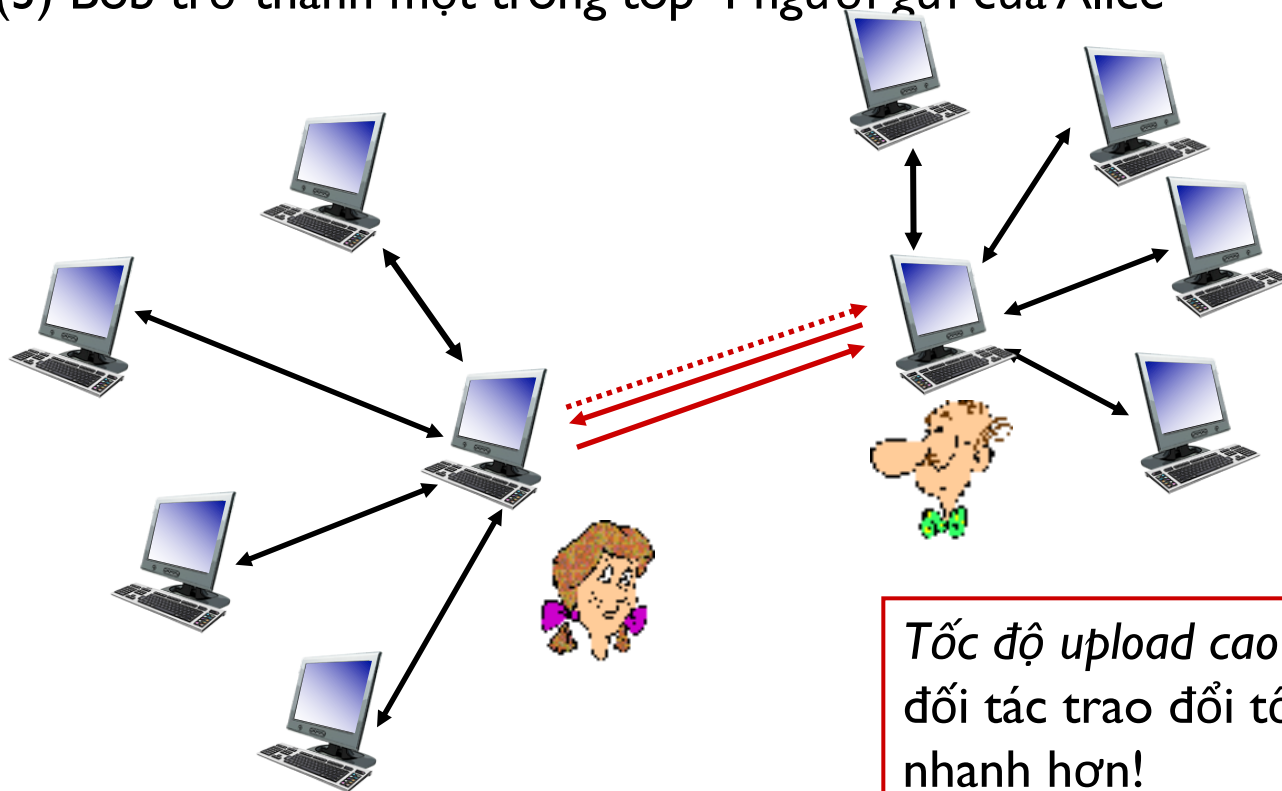
- Tại mỗi thời điểm, các peers khác nhau có tập các file chunks khác nhau
- Một cách định kỳ, Alice hỏi mỗi peer danh sách các chunks mà họ có
- Alice yêu cầu các chunks còn thiếu từ các peers có, bắt đầu từ chunks hiếm trước

## *Gửi chunks: tit-for-tat*

- ❖ Alice gửi chunks cho bốn peers hiện tại gửi cho Alice với *tốc độ cao nhất*
  - Các peer khác bị chặn, không nhận được dữ liệu từ Alice
  - Đánh giá lại top 4 sau mỗi 10 secs
- ❖ Trong chu kỳ 30 secs: chọn một cách ngẫu nhiên một peer và bắt đầu gửi chunks
  - "Gửi dữ liệu một cách lạc quan" với peer này
  - Peer được chọn mới có thể tham gia top 4

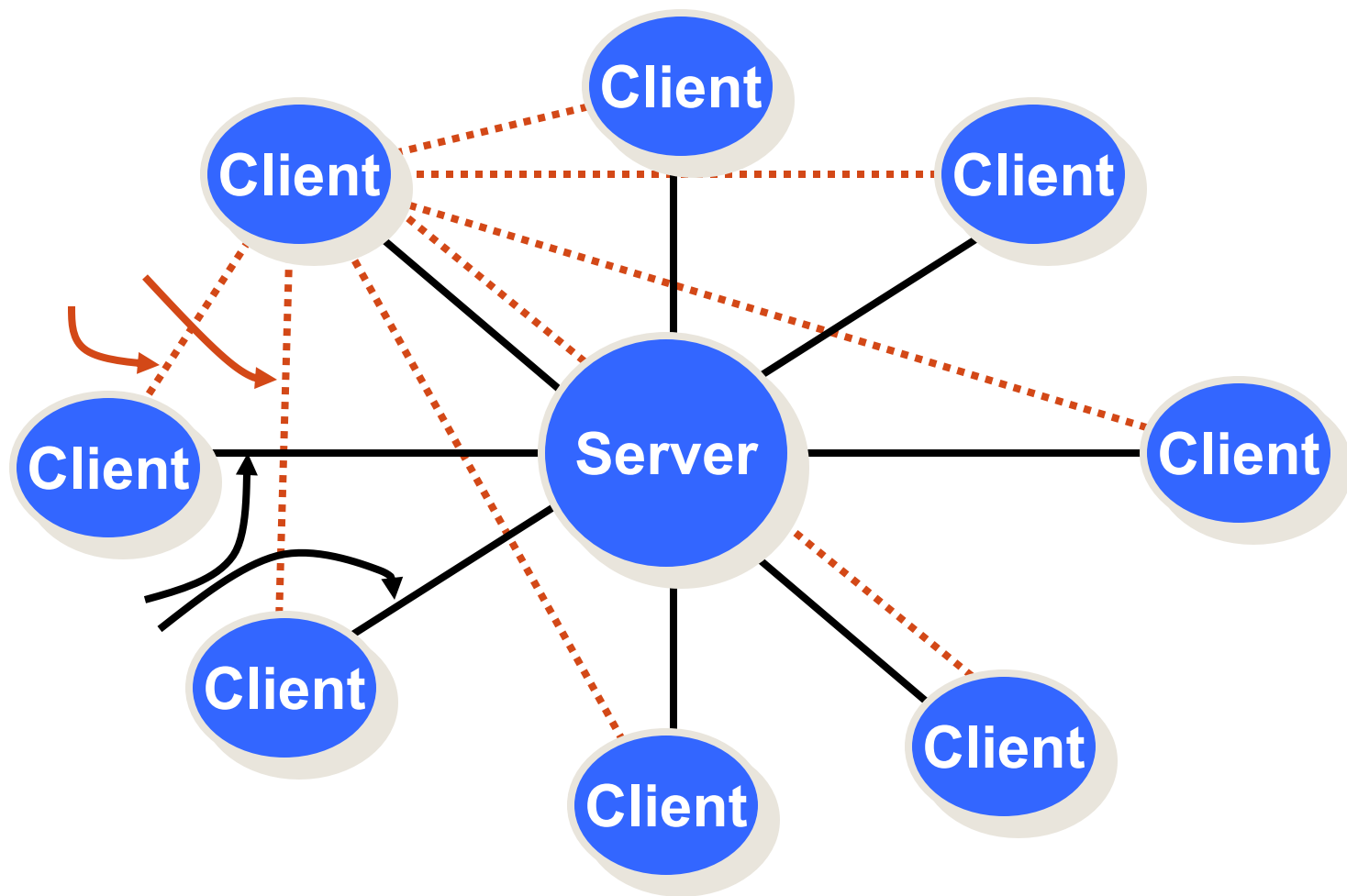
# BitTorrent: tit-for-tat

- (1) Alice “gửi dữ liệu một cách lặt vặt” tới Bob
- (2) Alice trở thành một trong top-4 người gửi của Bob; Bob đáp lại bằng cách gửi dữ liệu cho Alice
- (3) Bob trở thành một trong top-4 người gửi của Alice



*Tốc độ upload cao hơn: tìm được đối tác trao đổi tốt hơn, nhận file nhanh hơn!*

# Mô hình lai



# Giao thức là gì?

- “*protocols define format, order of msgs sent and received among network entities, and actions taken on msg transmission, receipt*”
- Giao thức rất quan trọng trong thiết kế ứng dụng mạng, ảnh hưởng trực tiếp đến
  - Khả năng xử lý
  - Hiệu năng của chương trình
  - Khả năng mở rộng

# Mục tiêu xây dựng giao thức

- Giao thức cần truyền tin nhanh hay không?
- Có cần truyền tin tin cậy không?
- Có cần xác thực giữa các bên không?
- Có cần mã hoá không?
- Có những bên tham gia nào? Tất cả các bên có cần truyền tin cho nhau không?
- Có giới hạn về băng thông và kết nối không?
- Có cần duy trì phiên làm việc không? Có thể dùng mô hình truyền tin không trạng thái không?
- Có cần cơ chế xử lý lỗi không?
- ...



# Yêu cầu khi thiết kế giao thức

- Đơn giản
  - Không phức tạp hoá vấn đề đơn giản
  - Không cung cấp 2 cách để thực hiện cùng 1 thứ
- Có khả năng mở rộng
  - Ước lượng số lượng clients trên một server (hoặc số lượng các nút tham gia)
  - Thiết kế giao thức để nó cân bằng trách nhiệm giữa mỗi bên (chuyển bớt chức năng sang phía client để giảm tải cho server)
  - Dành chỗ cho các chức năng sẽ mở rộng sau này
    - Nhưng không để dành nhiều quá
- Hiệu quả
  - Tối thiểu dư thừa điều khiển
  - Tối thiểu lưu lượng mạng

# Thiết kế thông báo

- Vấn đề rất quan trọng trong thiết kế giao thức
  - Ảnh hưởng lớn đến các đặc tính truyền tin: khả năng mở rộng, hiệu quả, tính đơn giản
- Gồm hai phần
  - Kiểu thông báo
  - Cấu trúc thông báo

# Kiểu thông báo

- Ba loại thông báo
  - Thông báo lệnh
  - Thông báo truyền dữ liệu
  - Thông báo điều khiển
- Mỗi loại thông báo có thể có nhiều thông báo khác nhau

# Thông báo lệnh

- Định nghĩa các giai đoạn truyền tin giữa các bên
- Định nghĩa các khía cạnh truyền tin khác nhau
  - Khởi tạo và kết thúc phiên làm việc
  - Đặc tả các giai đoạn truyền tin (e.g. xác thực, yêu cầu trạng thái, truyền dữ liệu)
  - Thay đổi trạng thái (e.g. yêu cầu chuyển chế độ truyền dữ liệu)
  - Thay đổi tài nguyên (e.g. yêu cầu kênh truyền mới)
  - ...

# Thông báo truyền dữ liệu

- Thông báo truyền dữ liệu được truyền khi trả lời thông báo lệnh
- Dữ liệu có thể bị phân mảnh trong nhiều thông báo
- Ngoài dữ liệu thực tế còn có các đặc tả
  - Định dạng dữ liệu nhị phân
  - Cách thức sắp xếp của dữ liệu có cấu trúc (khi dữ liệu có cấu trúc động)
  - độ lớn dữ liệu, offset của thông tin
  - Kiểu của khối dữ liệu: cuối/trung gian

# Thông báo điều khiển

- Điều khiển quá trình truyền tin giữa hai bên
  - Hợp tác giữa nhiều bên
  - Ngắt hoặc huỷ bỏ phiên làm việc
  - Kiểm tra tính sẵn sàng
  - ...

# FTP commands, responses

## *sample commands:*

- sent as ASCII text over control channel
- **USER *username***
- **PASS *password***
- **LIST** return list of file in current directory
- **RETR *filename*** retrieves (gets) file
- **STOR *filename*** stores (puts) file onto remote host


## *sample return codes*

- status code and phrase (as in HTTP)
- **331 Username OK, password required**
- **125 data connection already open; transfer starting**
- **425 Can't open data connection**
- **452 Error writing file**

# POP3 protocol

## *authorization phase*


- client commands:
  - **user**: declare username
  - **pass**: password
- server responses
  - **+OK**
  - **-ERR**



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

## *transaction phase, client:*

- **list**: list message numbers
- **retr**: retrieve message by number
- **dele**: delete
- **quit**

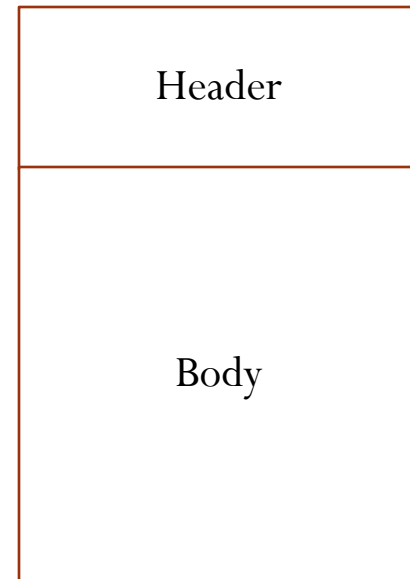


```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```



# Cấu trúc thông báo

- Tiêu đề:
  - chứa thông tin giúp bên nhận hiểu được phần còn lại của thông báo và chi tiết về dữ liệu trong phần thân
  - VD về các trường trong tiêu đề
    - Kiểu thông báo
    - Kiểu lệnh
    - Kích thước phần thân
    - Thông tin bên nhận
    - Thông tin chuỗi
    - Số lần gửi lại
    - Etc
- Thân:
  - Các tham số của lệnh
  - dữ liệu gửi



# Định dạng thông báo

- Thông báo có dạng text
- Thông báo nhị phân
- Thông báo có các trường cố định
- Thông báo có các trường thay đổi

# Thông báo dạng text

- Thông báo bao gồm một chuỗi ký tự đọc được
- Ưu điểm
  - dễ hiểu và dễ theo dõi
  - Mềm dẻo và dễ mở rộng
  - dễ test
- Nhược điểm
  - dễ bị đọc trộm
  - Phức tạp và khó khăn trong việc phân tích nội dung thông báo khi code
  - Kích thước thông báo lớn

# Thông báo dạng text

- Tag header

- XML-based

`<messageID>000</messageID><name>xxx</named><passwd>yyy</passwd>`

- Untagged header

- Sử dụng các ký tự đặc biệt để phân chia các từ khóa

- E.g. `\r\n` in SMTP

- `messageID:000\r\nname:xxx\r\npasswd:yyy`

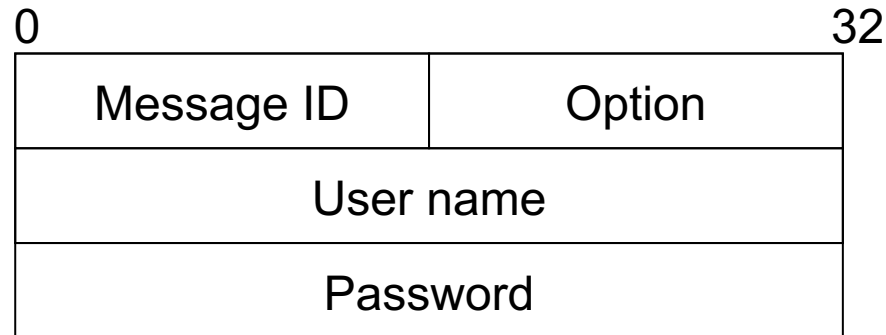
- json

- `{"messageID":100,"option":1000,"userName":"hoaison","password":"12345"}`

# Thông báo nhị phân

- Thông báo là các khối dữ liệu nhị phân có cấu trúc
- Ưu điểm
  - Phù hợp với việc truyền dữ liệu lớn và phức tạp
  - Thông báo có kích thước nhỏ
  - Cấu trúc dữ liệu chặt chẽ
- Nhược điểm
  - Khó đọc, debug và test
  - Cần lưu ý đến cách biểu diễn dữ liệu nhị phân trong các hệ thống máy tính khác nhau (big endian vs little endian)

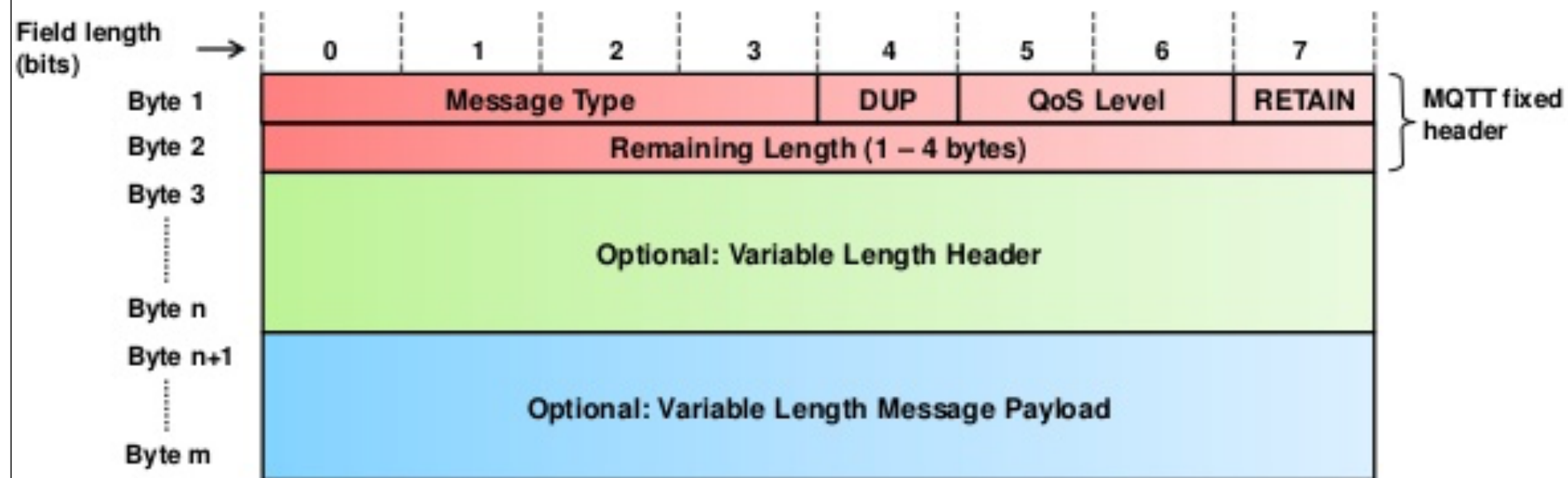
# Thông báo nhị phân



```
struct messageHeader{
    uint32 messageId;
    uint32 option;
    char userName[32];
    char password[32];
} *msgHeader;
char message[1024];
msgHeader = (struct messageHeader *)message;
```

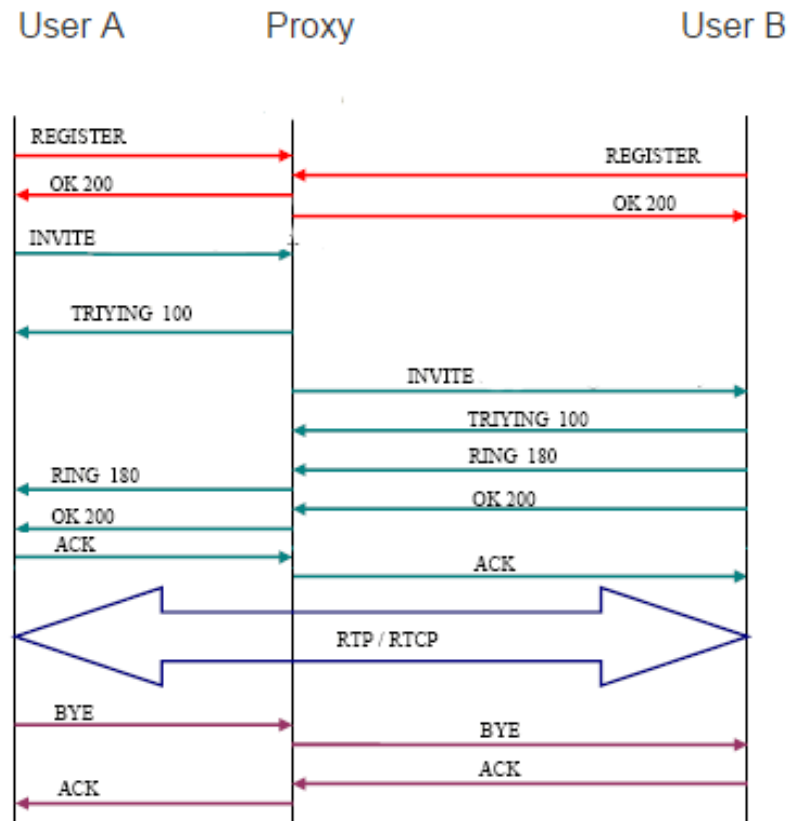
# Kiểu thông báo lai ghép

- Vừa có các trường cố định, vừa có các trường có độ dài thay đổi



# Cách thức truyền thông

- Mô tả thứ tự gửi nhận thông báo giữa các bên trong hệ thống
- Cần được mô tả rõ ràng và đầy đủ thông qua mô tả chi tiết về các kịch bản truyền thông có thể xảy ra
- Có thể sử dụng sơ đồ tuần tự để mô tả cách thức giao tiếp giữa các bên

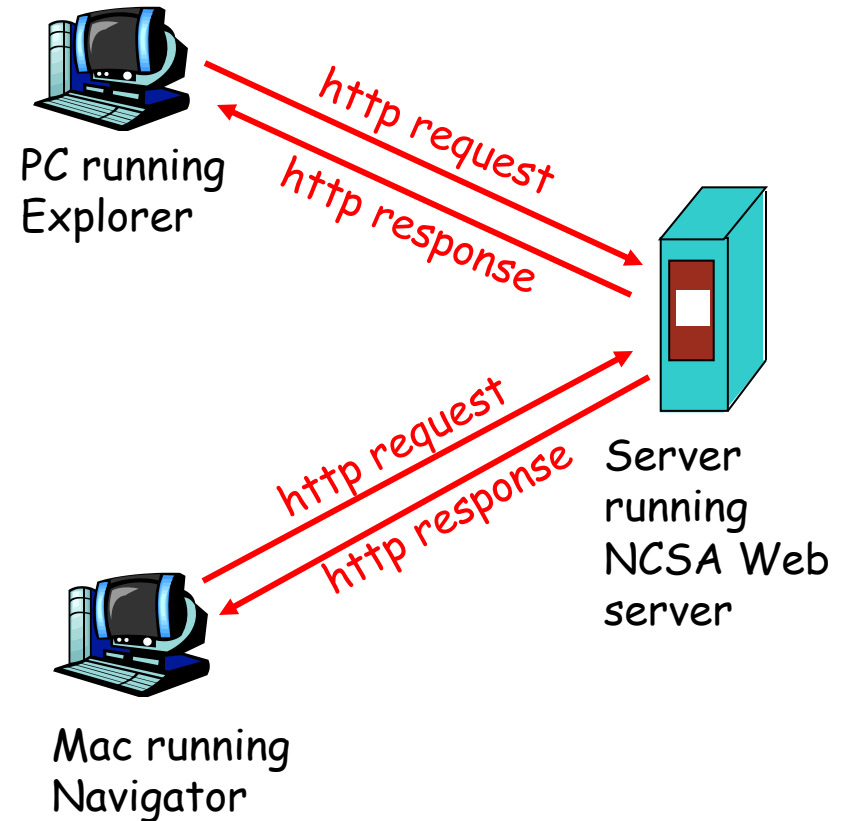




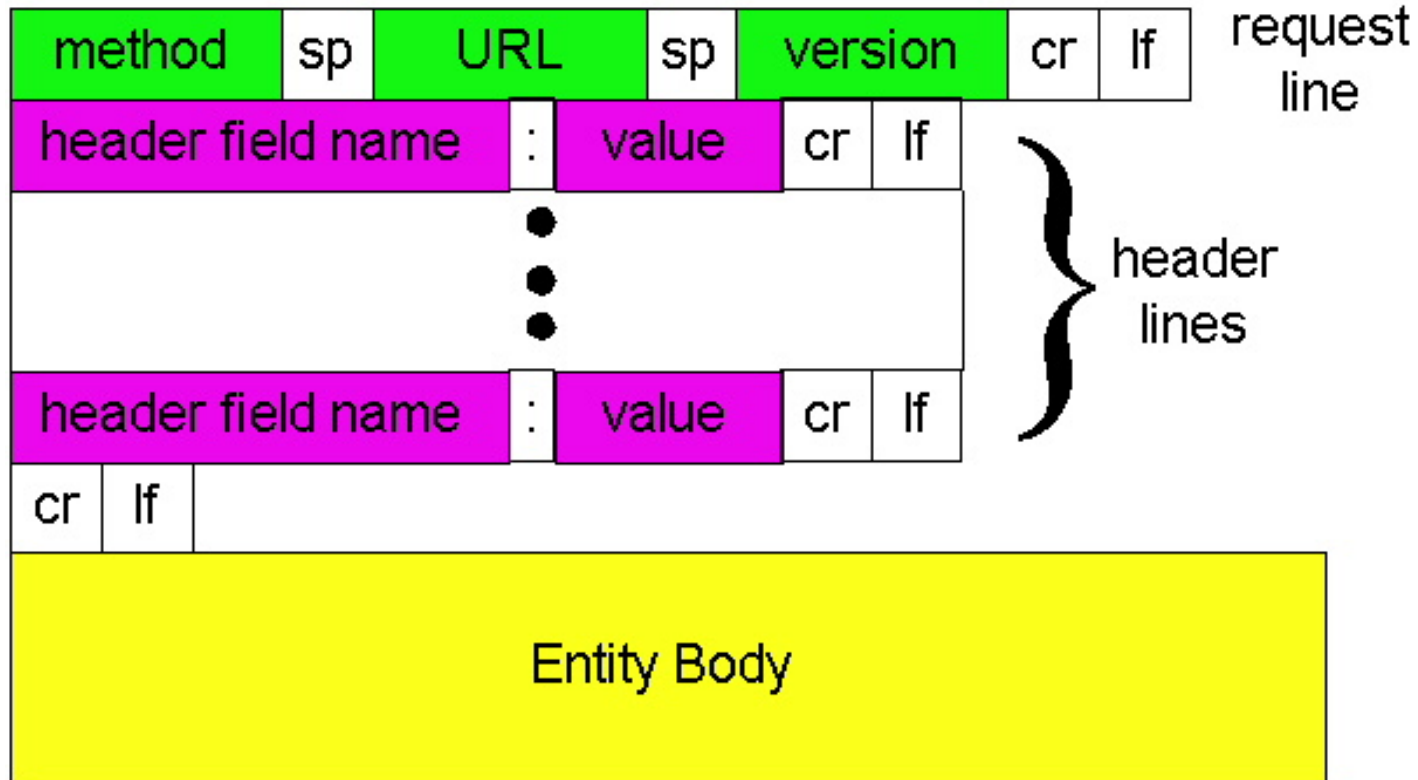
# Ví dụ về giao thức tầng ứng dụng HTTP

## http: hypertext transfer protocol

- Giao thức tầng ứng dụng của Web
- Mô hình khách chủ
  - *máy khách*: Trình duyệt gửi yêu cầu, nhận kết quả và hiển thị trang Web
  - *Máy khách*: Máy chủ Web gửi trang Web khi nhận được yêu cầu của máy khách
- http1.0: RFC 1945
- http1.1: RFC 2068



# Định dạng thông báo yêu cầu



# Ví dụ về thông báo yêu cầu

- GET /hello.html HTTP/1.0
- Host: www.abc.xyz.edu
- User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; ...
- Accept: text/xml,application/xml,application/xhtml+xml,...
- Accept-Language: en-us,en;q=0.5
- Accept-Encoding: gzip,deflate
- Accept-Charset: ISO-8859-1,utf-8;q=0.7,\*;q=0.7
- Keep-Alive: 300
- Connection: keep-alive
- Cookie: SignOnDefault=ATWOLF

# Định dạng thông báo trả lời

status line  
(protocol  
status code  
status phrase)

header  
lines

HTTP/1.0 200 OK

Date: Thu, 06 Aug 1998 12:00:15 GMT

Server: Apache/1.3.0 (Unix)

Last-Modified: Mon, 22 Jun 1998 .....

Content-Length: 6821

Content-Type: text/html

data, e.g.,  
requested  
html file

data data data data data ...

# Đánh giá giao thức HTTP

- Ưu điểm
  - Đơn giản, dễ cài đặt
  - Khả năng mở rộng cao
  - Dễ hiểu với người dùng
- Nhược điểm
  - Overhead lớn
  - Không phù hợp với ứng dụng thời gian thực