

Trường Đại Học Bách Khoa Tp.HCM
Hệ Đào Tạo Từ Xa
Khoa Khoa Học và Kỹ Thuật Máy Tính

Mạng máy tính căn bản

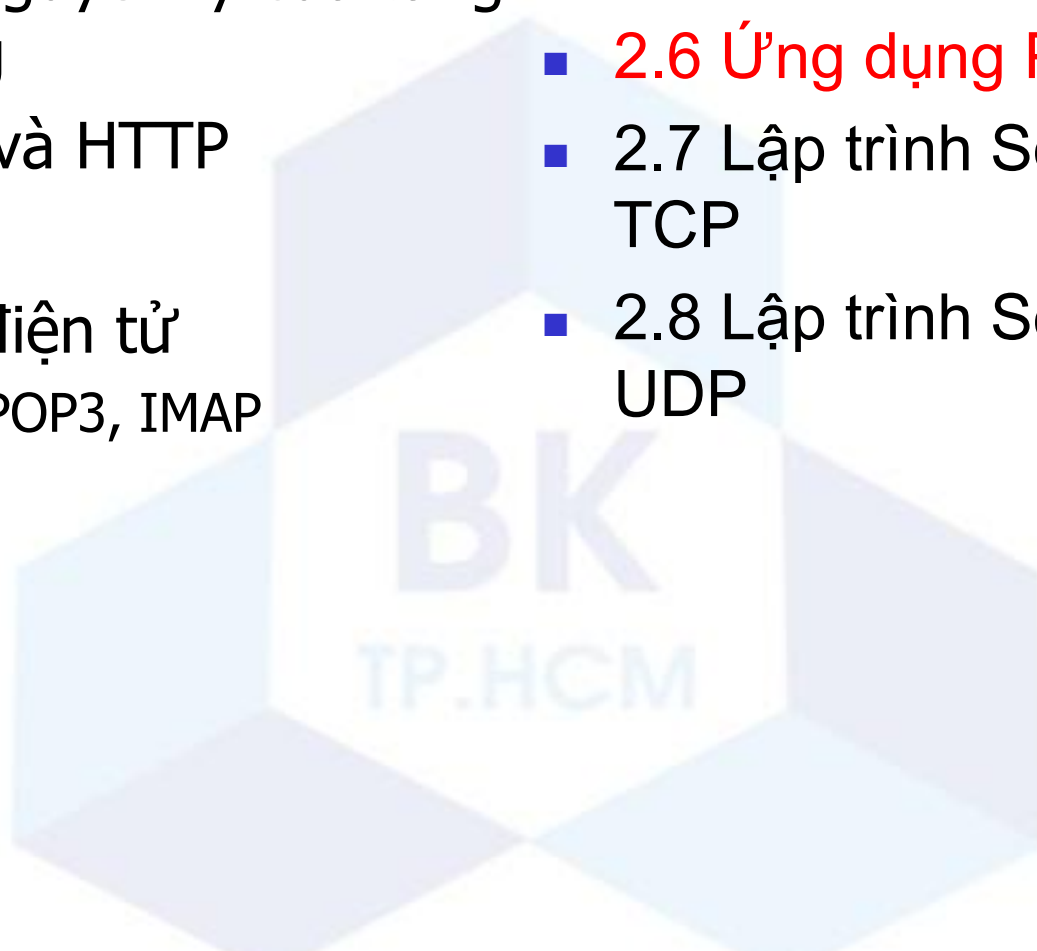
Bài giảng 5: Tầng ứng dụng (tt)

Tham khảo:

Chương 2: “Computer Networking – A top-down approach”
Kurose & Ross, 5th ed., Addison Wesley, 2010.

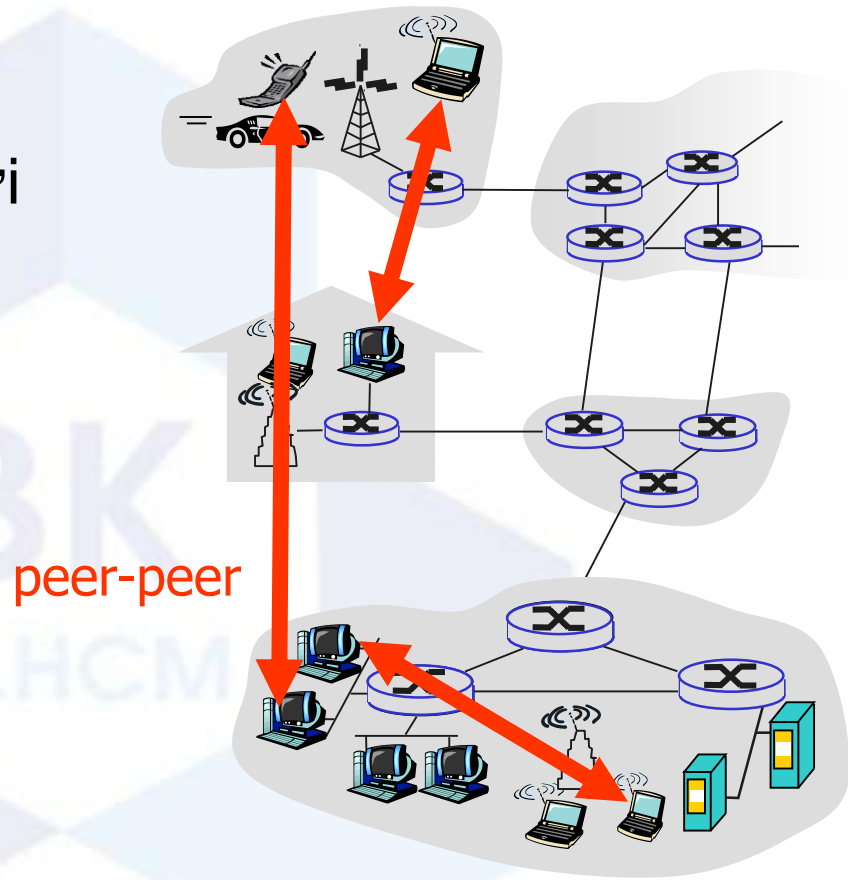
Chương 2: Tầng ứng dụng

- 2.1 Các nguyên lý của tầng ứng dụng
- 2.2 Web và HTTP
- 2.3 FTP
- 2.4 Thư điện tử
 - SMTP, POP3, IMAP
- 2.5 DNS
- 2.6 Ứng dụng P2P
- 2.7 Lập trình Socket với TCP
- 2.8 Lập trình Socket với UDP



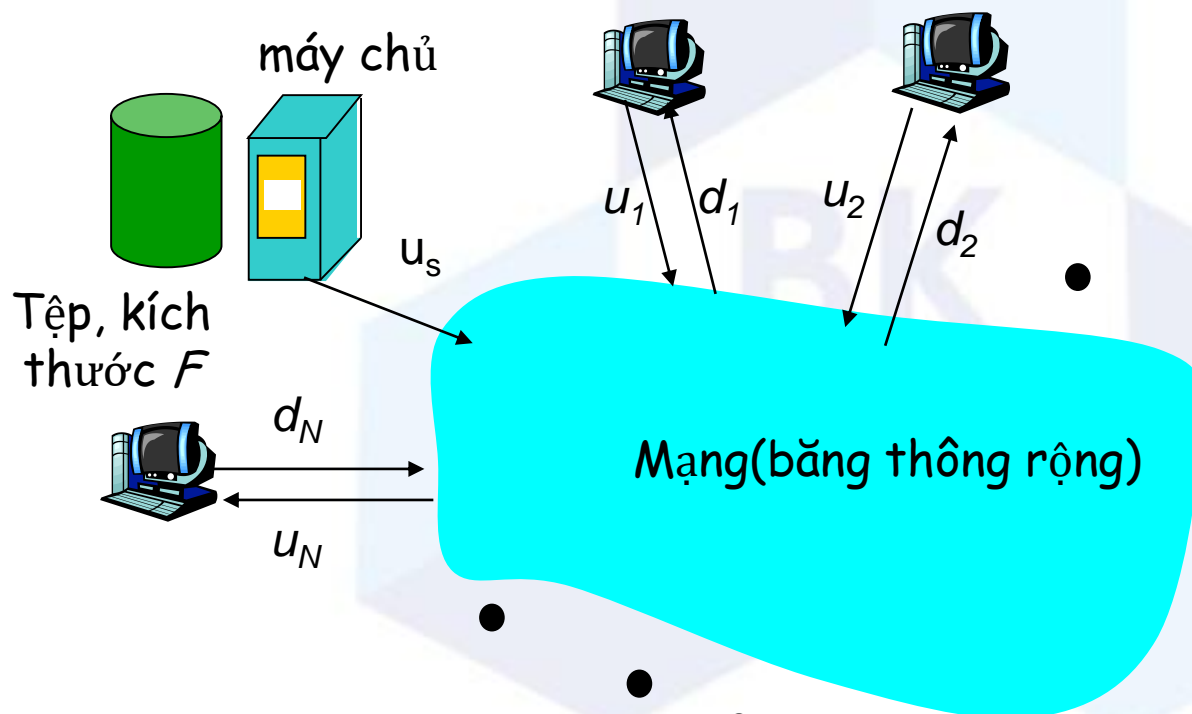
Cấu trúc P2P thuần túy

- máy chủ *không* luôn luôn mở
- nhiều máy đầu cuối khác nhau giao tiếp trực tiếp với nhau
- các bên thực hiện kết nối không liên tục và thay đổi địa chỉ IP
- Ba chủ đề:
 - Phân phối tệp tin
 - Tìm kiếm thông tin
 - Tình huống nghiên cứu: Skype



Phân phối tệp tin: Chủ-khách so với P2P

Câu hỏi: Cần bao nhiêu thời gian để phân phối tệp từ 1 máy chủ tới N người dùng?



u_s : băng thông tải lên của máy chủ

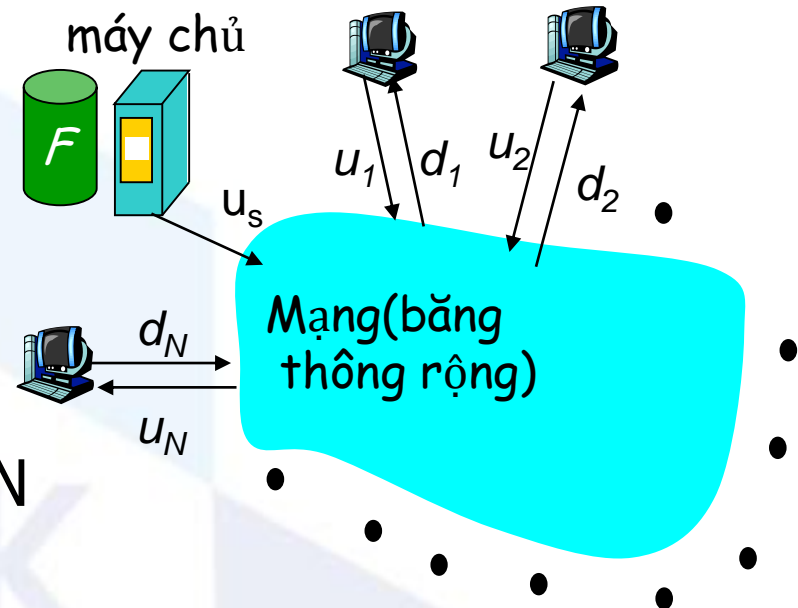
u_i : băng thông tải lên của mỗi khách

d_i : băng thông tải xuống của mỗi khách

Mạng(băng thông rộng)

Thời gian phân phối tệp: chủ-khách

- chủ lần lượt gửi N bản sao:
 - NF/u_s
- khách i cần F/d_i t/g để tải xuống

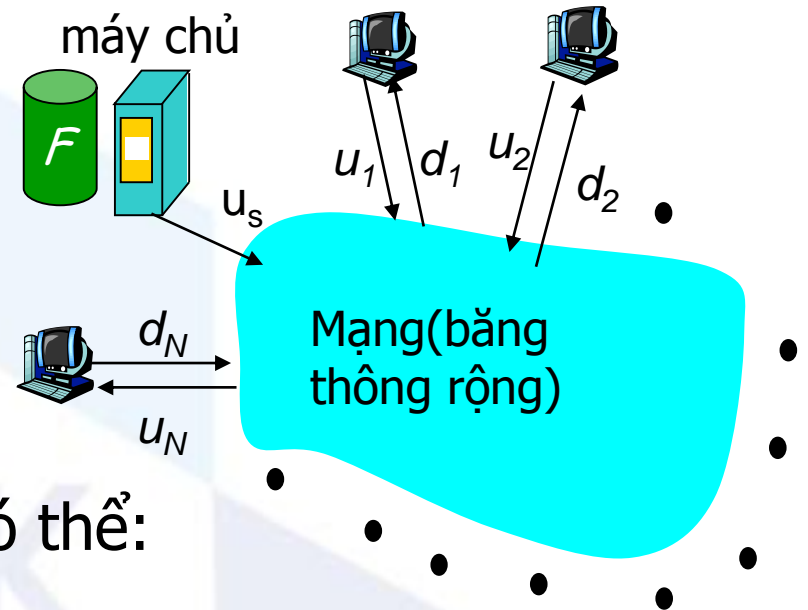


tăng tuyến tính theo N
(với N lớn)

Thời gian để phân phối F
tới N khách sử dụng
mô hình khách/chủ $= d_{cs} = \max \{ NF/u_s, F/\min_i(d_i) \}$

Phân phối tập tin: P2P

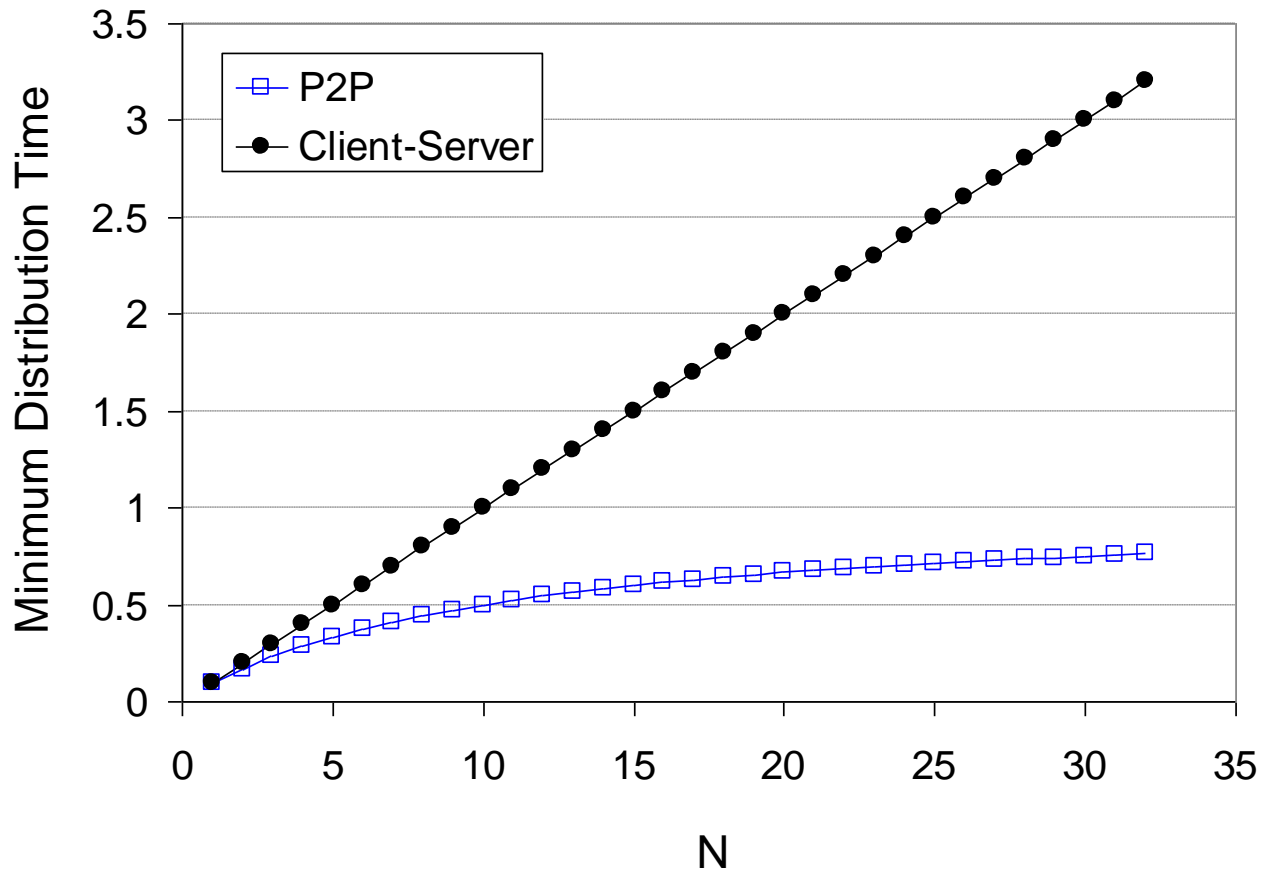
- máy chủ phải gửi một phản sao: F/u_s đvtg
- khách i cần F/d_i t/g để tải về
- NF bit phải được tải về
- Vận tốc tải lên nhanh nhất có thể:
 $u_s + Su_i$



$$d_{P2P} = \max \left\{ F/u_s, F/\min(d_i), NF/(u_s + Su_i) \right\}$$

Khách-chủ so với P2P: ví dụ

tốc độ tải lên của $n/d = u$, $F/u = 1h$, $u_s = 10u$, $d_{\min} \geq u_s$

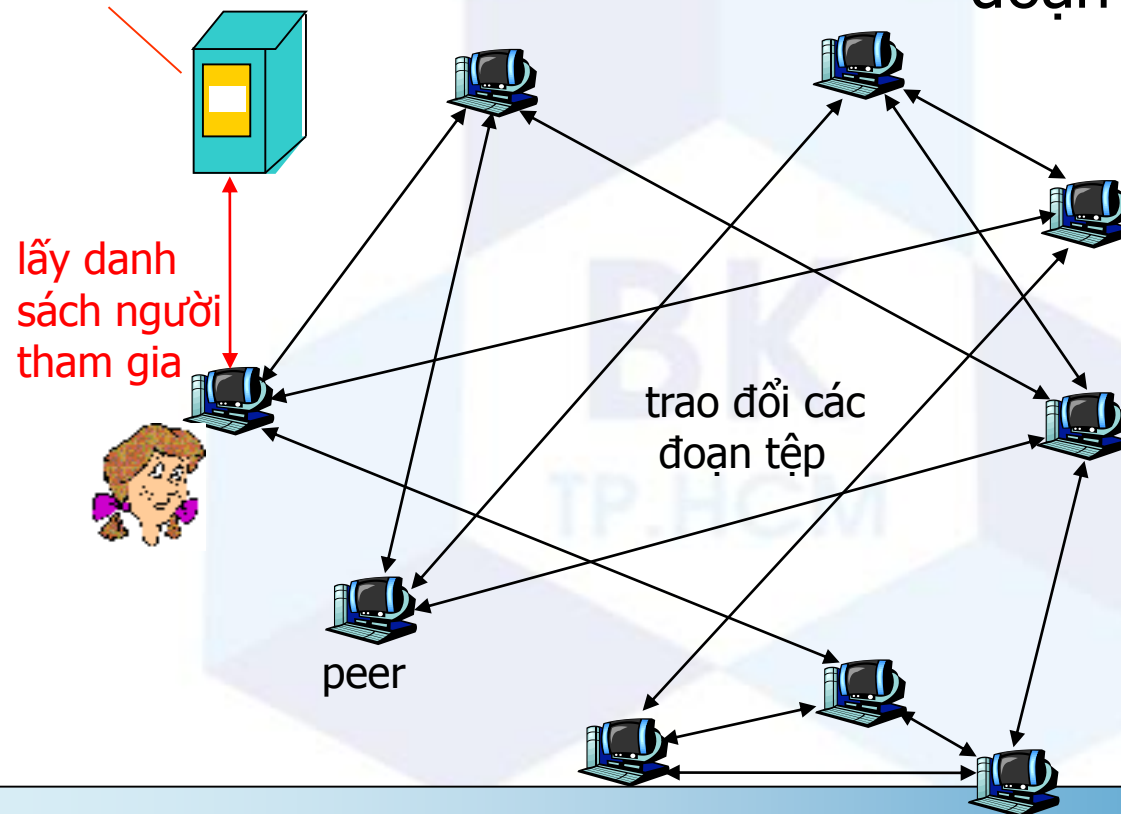


Phân phối tập tin: BitTorrent

- phân phối tập kiểu P2P

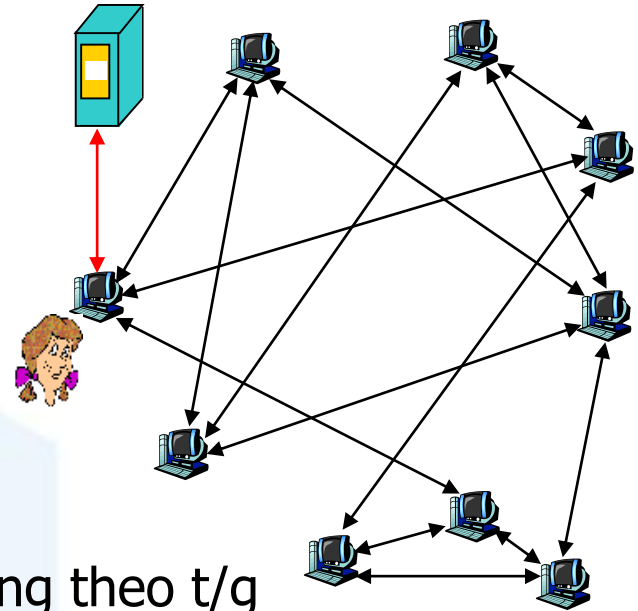
máy chủ (tracker): theo dõi các cá nhân tham gia vào torrent

torrent: nhóm các cá nhân chia sẻ những đoạn tập



BitTorrent (1)

- tệp được chia thành *đoạn* 256KB.
- cá nhân tham gia torrent:
 - ko có đoạn nào, nhưng sẽ tích lũy chúng theo t/g
 - đăng kí với tracker để lấy danh sách thành viên, liên kết tới một nhóm nhỏ thành viên ("hàng xóm")
- trong khi tải về, người tải đồng thời chia sẻ đoạn tệp cho những người khác.
- cá nhân có thể tham gia hoặc từ bỏ torrent
- một khi n/d tải xong tệp, họ có thể rời mạng torrent hoặc ở lại để chia sẻ cho người khác



BitTorrent (2)

Kéo các đoạn tệp

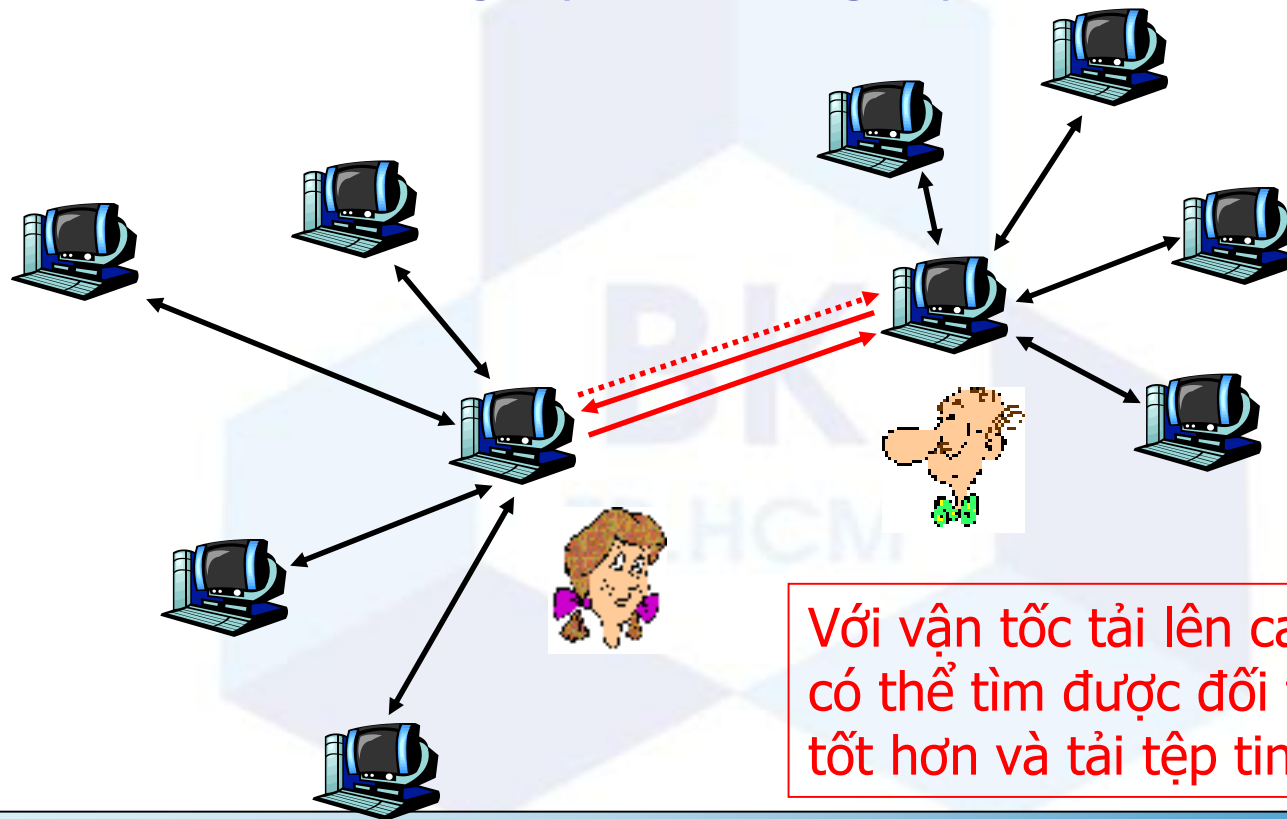
- tại bất kì thời điểm nào, các thành viên khác nhau sẽ có những đoạn khác nhau của một tệp
- một cách định kì, một thành viên (Alice) sẽ yêu cầu từ các hàng xóm danh sách các đoạn mà họ có.
- Alice gửi yêu cầu tới các đoạn mà cô ta thiếu
 - đoạn hiếm nhất trước

Gửi các đoạn tệp: tit-for-tat

- Alice gửi các đoạn cho 4 hàng xóm mà đang gửi đoạn cho cô ta ở vận tốc cao nhất
- đánh giá lại tốp 4 sau mỗi 10s
- sau mỗi 30 s: chọn ngẫu nhiên một thành viên khác, và gửi đoạn cho nó
- thành viên mới này có thể vào tốp 4

BitTorrent: Tit-for-tat

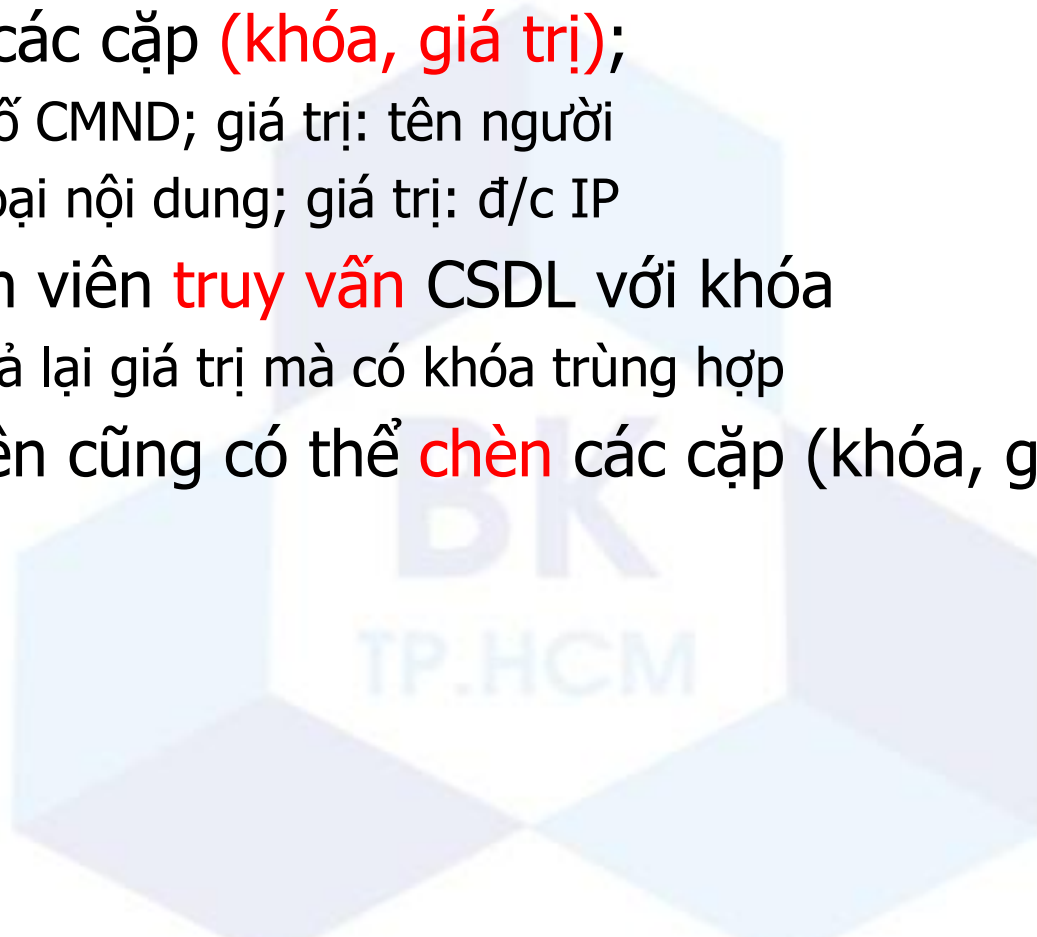
- (1) Alice “gửi thử nghiệm – ngẫu nhiên” cho Bob
- (2) Alice trở thành 1 trong топ 4 nhà cung cấp của Bob; Bob trả ơn
- (3) Bob trở thành 1 trong топ 4 nhà cung cấp của Alice



Với vận tốc tải lên cao hơn,
có thể tìm được đối tác truyền tải
tốt hơn và tải tệp tin nhanh hơn!

Bảng băm phân tán (DHT)

- DHT = cơ sở dữ liệu P2P phân tán
- CSDL có các cặp (khóa, giá trị);
 - khóa: số CMND; giá trị: tên người
 - khóa: loại nội dung; giá trị: đ/c IP
- Các thành viên truy vấn CSDL với khóa
 - CSDL trả lại giá trị mà có khóa trùng hợp
- Thành viên cũng có thể chèn các cặp (khóa, giá trị) vào CSDL

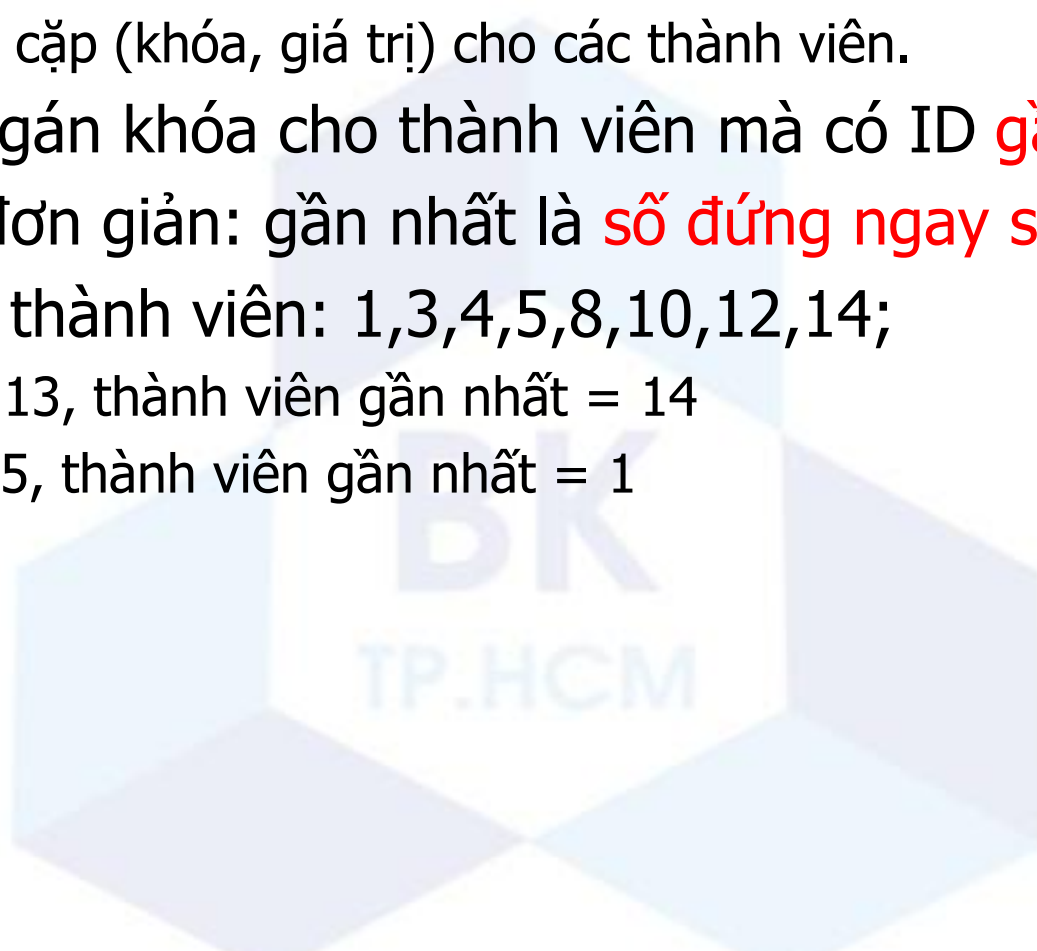


Định danh DHT

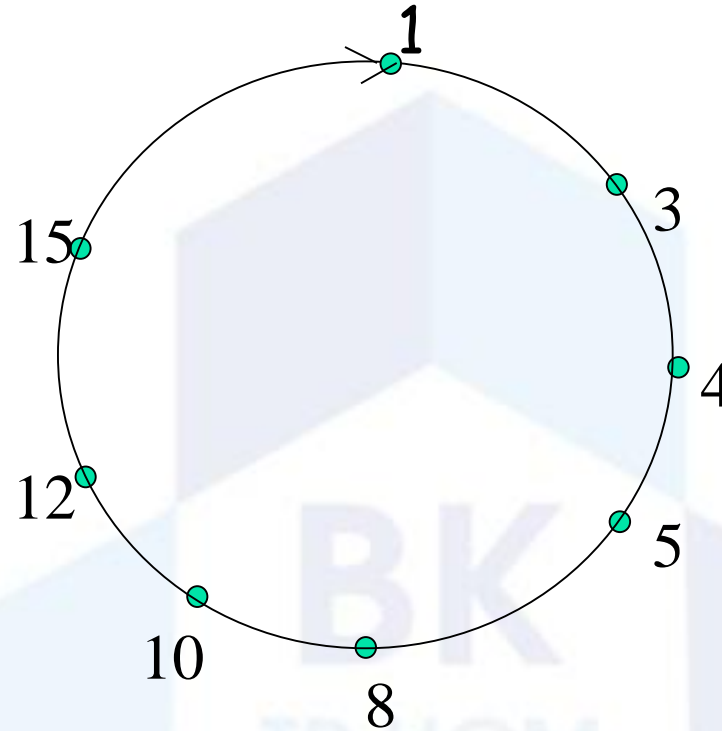
- Gán cho mỗi thành viên một số nguyên định danh trong khoảng $[0, 2^n - 1]$.
 - Mỗi định danh có thể được biểu diễn bằng n bit.
- Yêu cầu mỗi khóa cũng là một số nguyên trong cùng **khoảng trên**.
- Để tạo ra khóa số nguyên ta băm khóa nguyên thủy.
 - vd: $\text{key} = h(\text{"Led Zeppelin IV"})$
 - Vì vậy gọi là bảng "băm" phân tán

Làm sao để gán khóa cho các thành viên?

- Vấn đề trọng tâm:
 - gán các cặp (khóa, giá trị) cho các thành viên.
- Quy luật: gán khóa cho thành viên mà có ID gần nhất.
- Quy ước đơn giản: gần nhất là số đứng ngay sau của khóa.
- Vd: $n=4$; thành viên: 1,3,4,5,8,10,12,14;
 - khóa = 13, thành viên gần nhất = 14
 - key = 15, thành viên gần nhất = 1



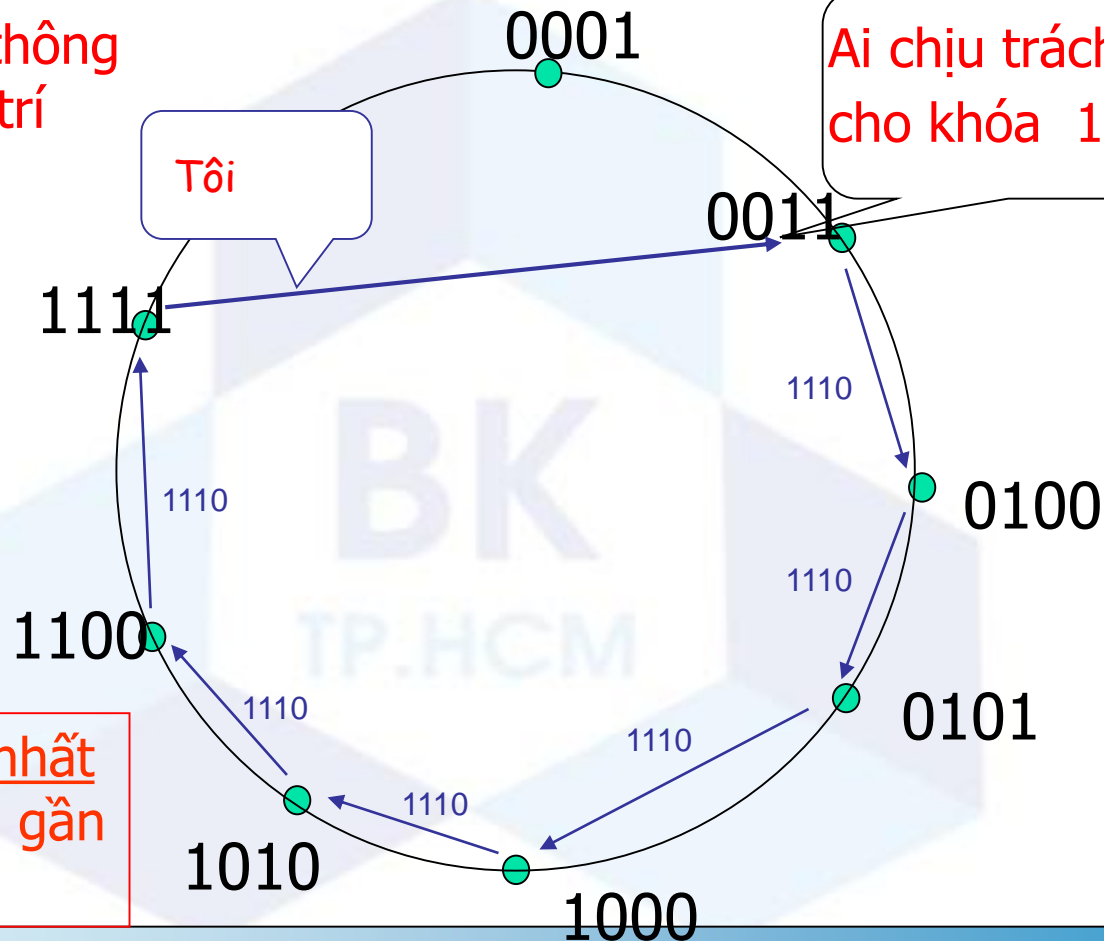
DHT xoay vòng (1)



- Mỗi thành viên *chỉ* nắm thông tin của người đứng ngay trước hoặc ngay sau nó.
- “Mạng bao phủ”

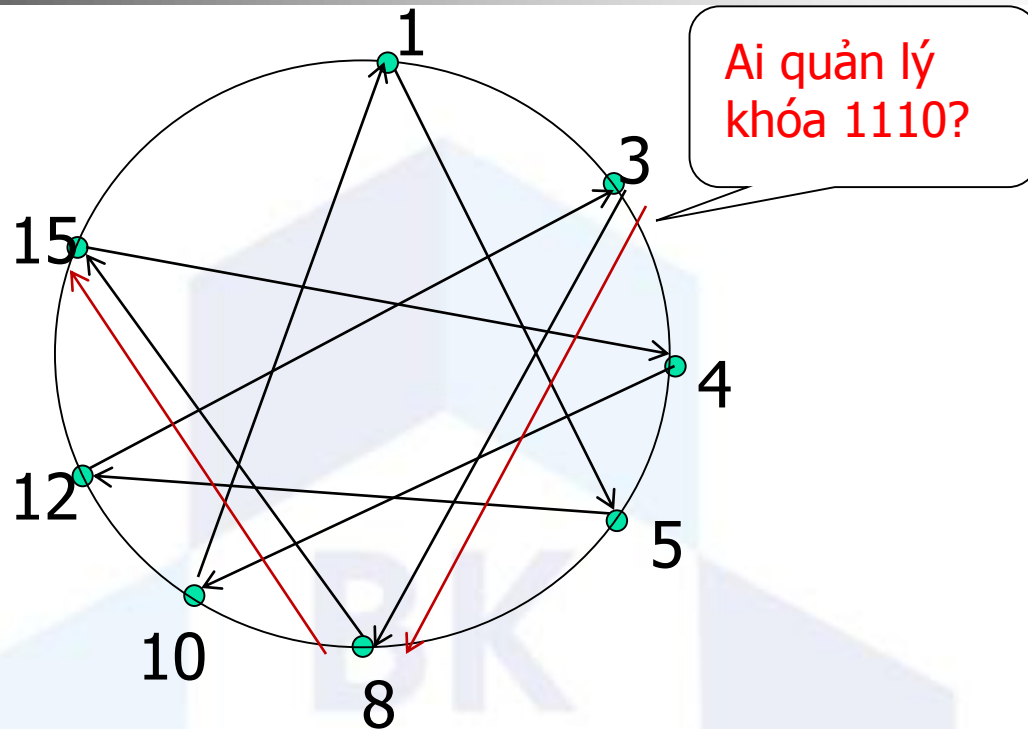
DHT xoay vòng (2)

trung bình $O(N)$ thông
điệp để tìm ra vị trí
thành viên



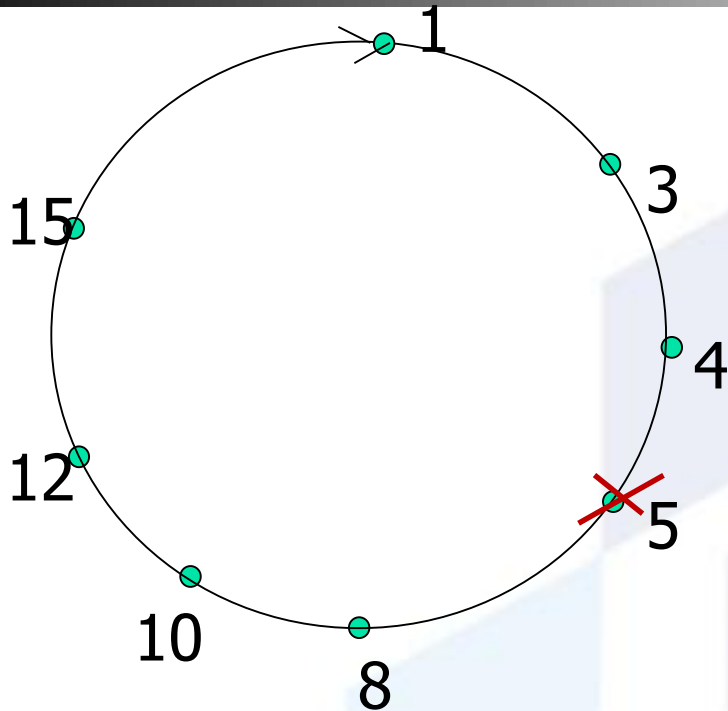
Định nghĩa gần nhất
là người liền sau gần
nhất

DHT xoay vòng với liên kết tắt



- Mỗi thành viên lưu dấu của địa chỉ IP của người liền trước, liền sau và vài liên kết tắt.
- Giảm từ 6 xuống còn 2 thông điệp.
- Có thể thiết kế liên kết tắt sao cho có $O(\log N)$ hàng xóm, $O(\log N)$ thông điệp cho mỗi truy vấn

Thành viên rời khỏi mạng (Peer Churn)



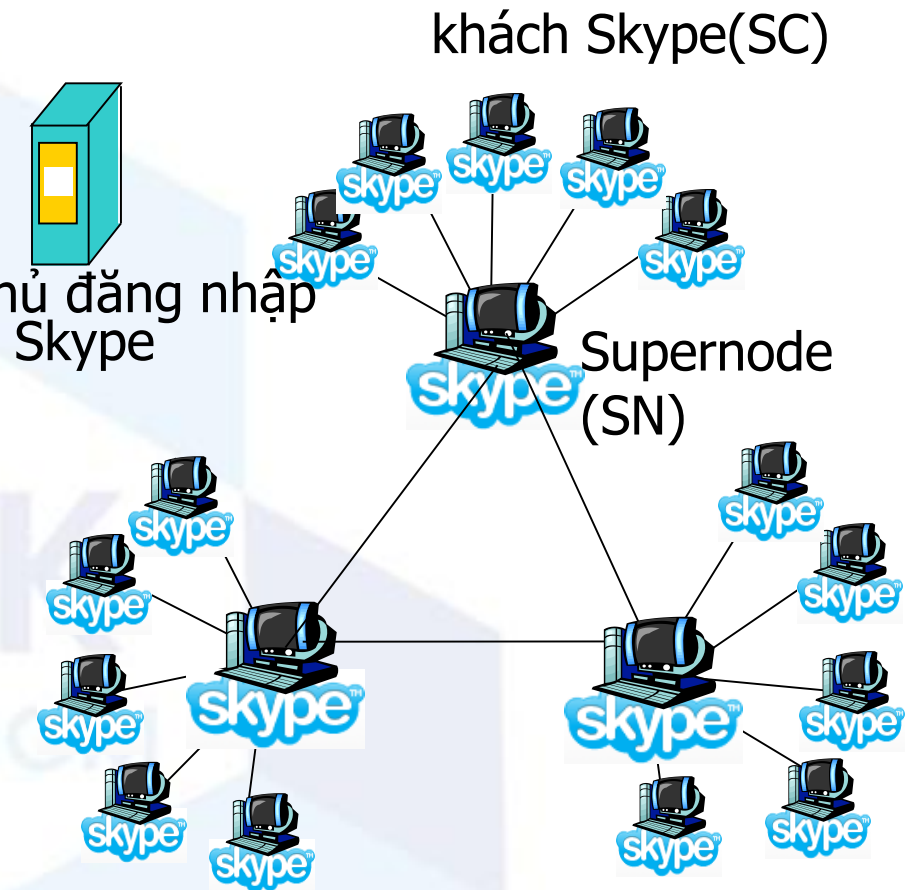
- Để xử lý peer churn, yêu cầu mỗi t/viên phải biết địa chỉ của 2 người liền sau nó.
- Mỗi t/viên theo định kì ping 2 người liền kề nó để xem họ còn trên mạng ko.

1. Thành viên số 5 đột nhiên rời khỏi mạng

- Thành viên 4 nhận ra; nhận 8 làm người liền sau chính thức; hỏi 8 ai là người liền sau chính thức của nó; nhận người liền sau chính thức của 8 làm người liền sau thứ 2.
- Chuyện gì xảy ra nếu 13 muốn gia nhập?

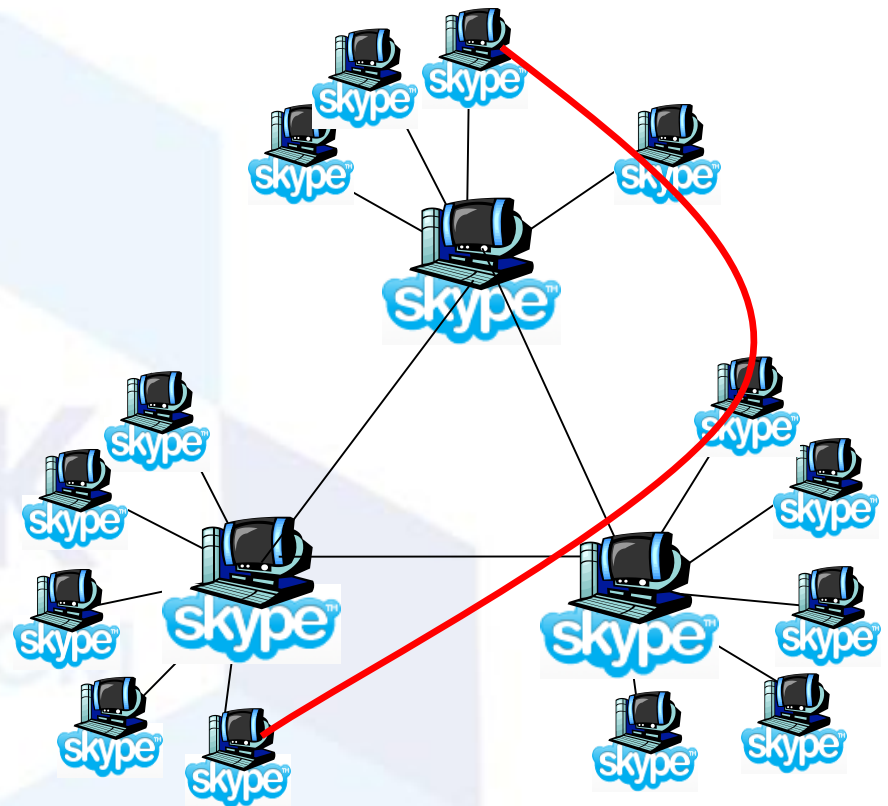
Trường hợp nghiên cứu P2P: Skype

- có tính chất P2P: người dùng liên lạc với nhau.
- giao thức tầng ứng dụng sở hữu (tài sản riêng của Skype)
- hệ thống bao phủ phân cấp với Supernode (SN)
- chỉ mục ánh xạ tên người dùng với địa chỉ IP; phân tán thông qua SN



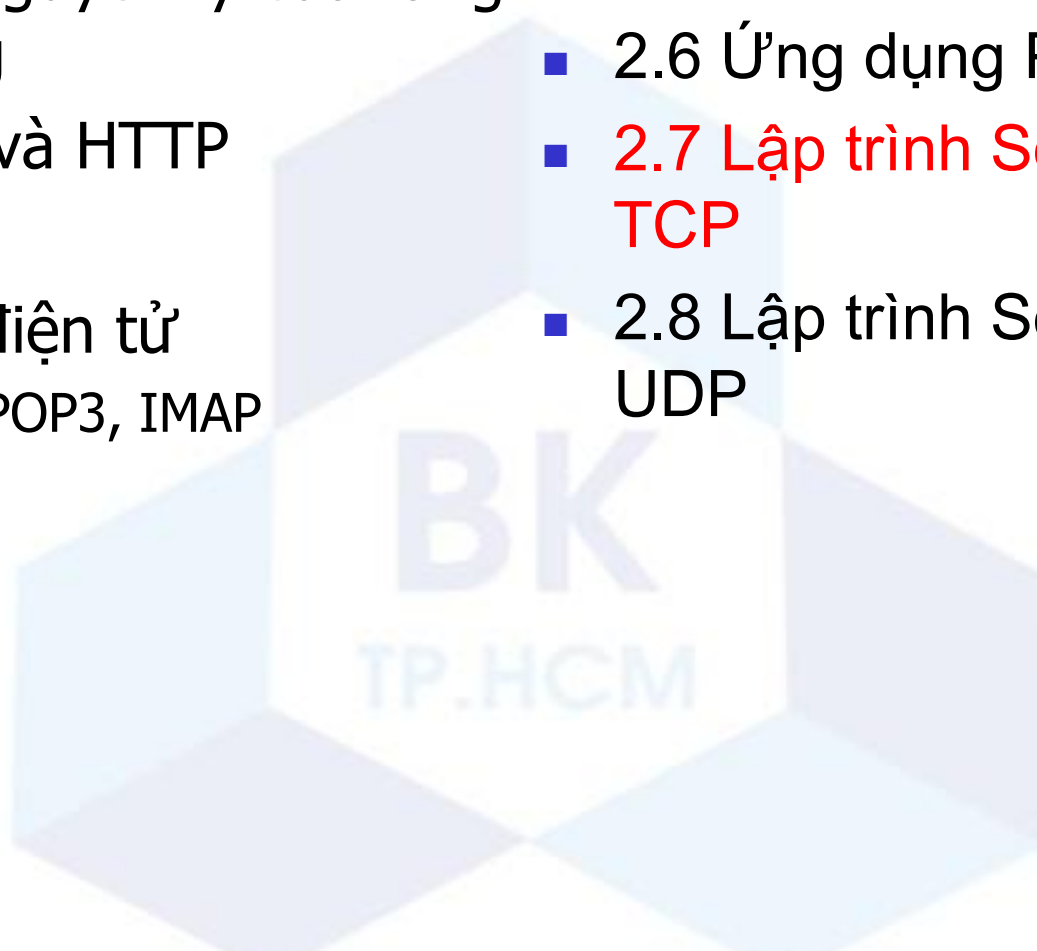
Các thành viên như là trạm chuyển tiếp

- Vấn đề khi cả Alice và Bob đứng sau các "NAT".
 - NAT ngăn cản những t/viên bên ngoài khởi đầu cuộc gọi vào t/viên bên trong
- Giải pháp:
 - Sử dụng SN của Alice và Bob, Chế độ chuyển tiếp được chọn
 - Mỗi thành viên khởi đầu phiên làm việc với SN "chuyển tiếp".
 - Các t/viên có thể liên lạc xuyên qua NAT bằng "trạm chuyển tiếp"



Chương 2: Tầng ứng dụng

- 2.1 Các nguyên lý của tầng ứng dụng
- 2.2 Web và HTTP
- 2.3 FTP
- 2.4 Thư điện tử
 - SMTP, POP3, IMAP
- 2.5 DNS
- 2.6 Ứng dụng P2P
- 2.7 Lập trình Socket với TCP
- 2.8 Lập trình Socket với UDP



Lập trình hốc kết nối (HKN)

Mục tiêu: học cách xây dựng ứng dụng khách/chủ sử dụng hốc kết nối (HKN - socket)

Giao diện lập trình ứng dụng (GDLTƯD - API) HKN

- được giới thiệu trong BSD4.1 UNIX, 1981
- được khởi tạo, sử dụng, và giải phóng bởi các ứng dụng
- mô hình khách/chủ
- hai loại dịch vụ truyền tải thông qua GDLTƯD HKN:
 - không tin cậy
 - tin cậy, hướng kết nối

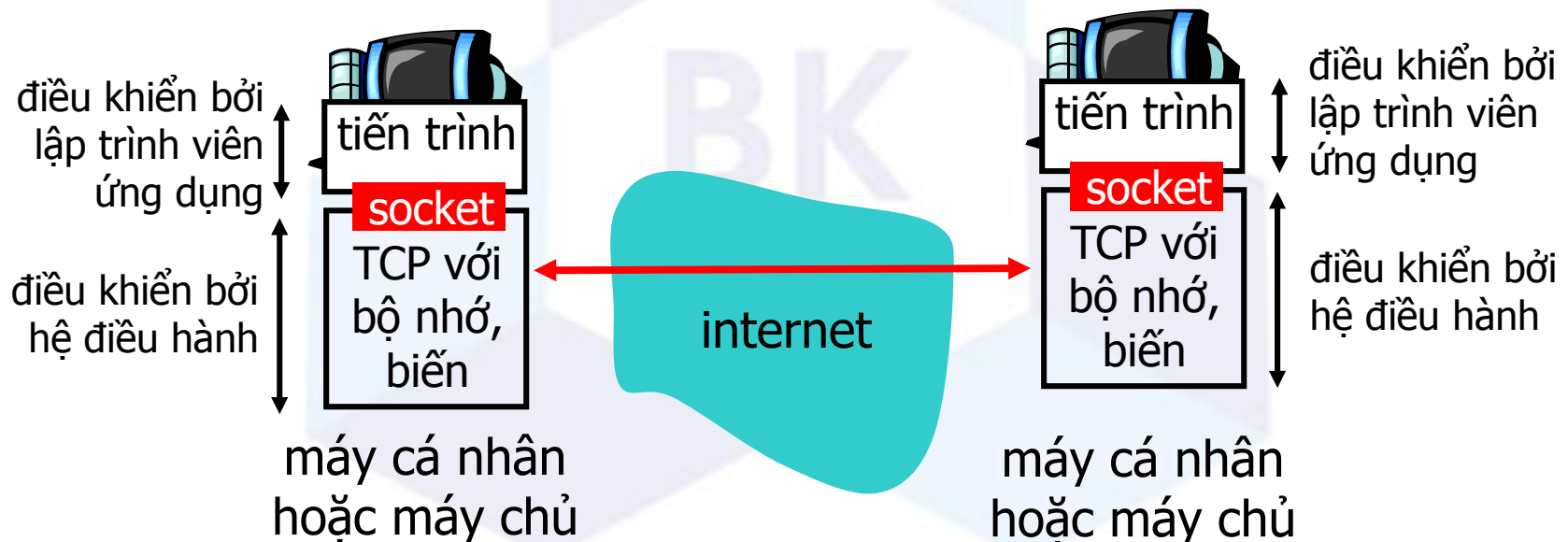
HKN

một giao diện *trên máy cục bộ, tạo bởi ứng dụng, điều khiển bởi OS* (một “cánh cửa”) thông qua đó tiến trình ứng dụng có thể **vừa gửi và nhận** tin nhắn tới/từ một tiến trình ứng dụng khác

Lập trình Socket sử dụng TCP

Socket: một cánh cửa giữa tiến trình ứng dụng và giao thức truyền tải đầu cuối-đầu cuối (UCP hoặc TCP)

Dịch vụ TCP: truyền tải tin cậy của **bytes (bai)** từ một tiến trình tới tiến trình khác



Lập trình Socket *với TCP*

Khách phải liên hệ chủ

- tiến trình chủ phải khởi chạy từ đầu
- máy chủ phải khởi tạo socket và sẵn sàng nhận sự liên hệ từ khách

Khách liên hệ chủ bằng cách:

- tạo ra một socket TCP cục bộ trên máy khách
- chỉ rõ địa chỉ IP, số cổng của tiến trình chủ
- khi **khách tạo socket**: khách đồng thời thiết lập một kết nối tới máy chủ TCP

- Khi được liên hệ bởi khách, **máy chủ TCP tạo một socket mới** để tiến hành liên lạc với khách
 - cho phép chủ có thể nói chuyện với nhiều khách
 - số cổng nguồn được dùng để phân biệt người dùng

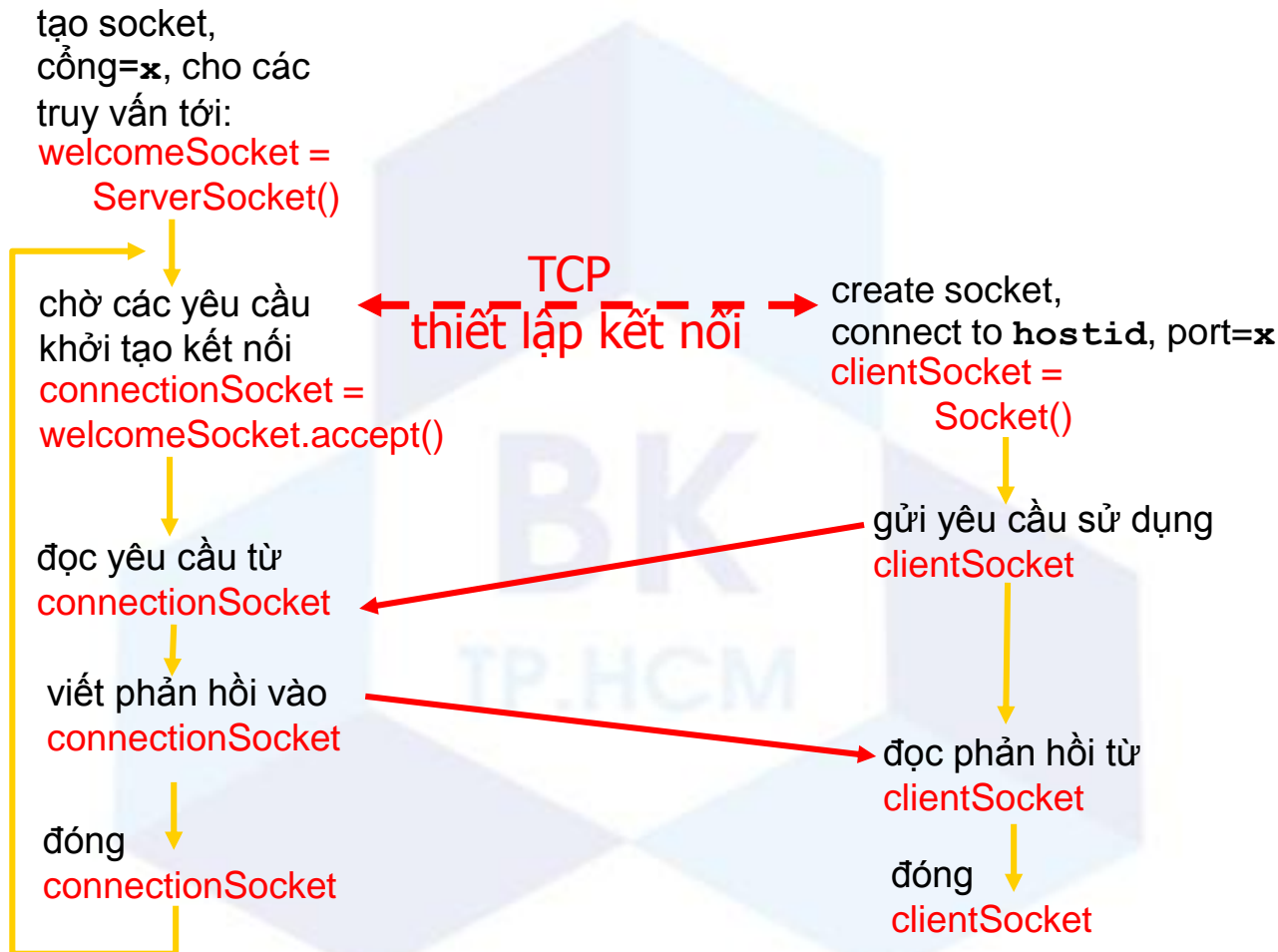
từ góc nhìn của ứ/d

TCP cung cấp dịch vụ truyền tải tin cậy, theo thứ tự của các byte giữa khách và chủ

Tương tác socket Khách/Chủ: TCP

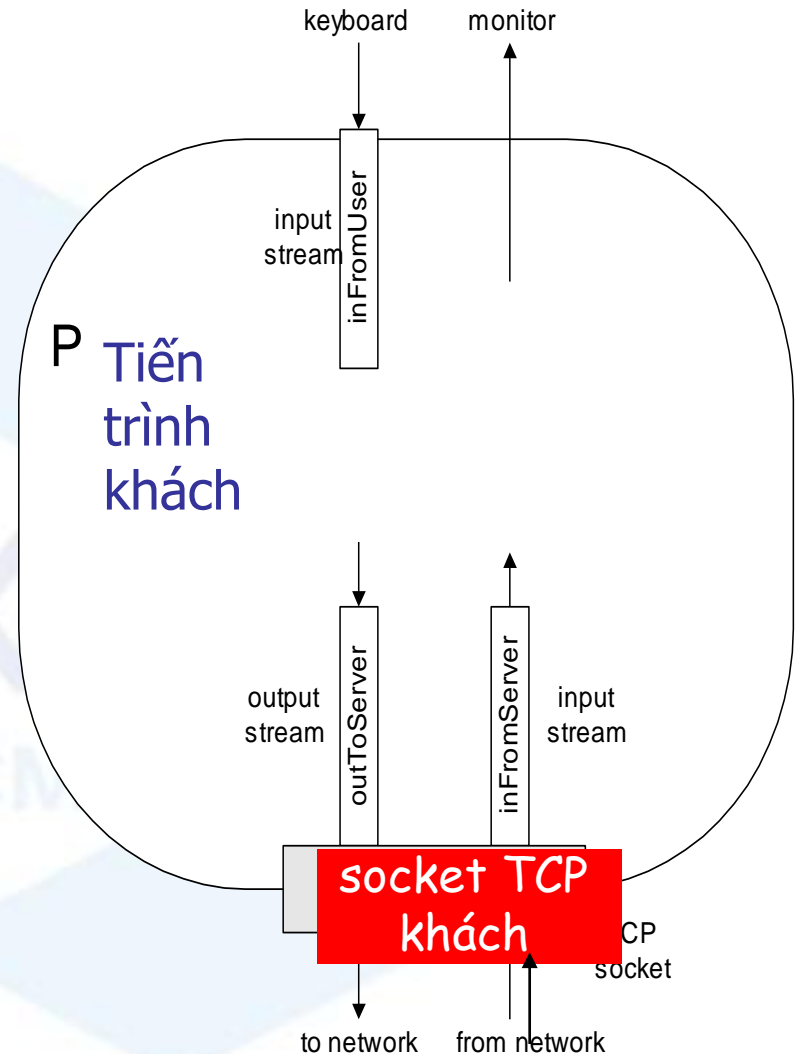
Chủ (chạy trên **hostid**)

Khách



Thuật ngữ Stream – luồng

- Một **luồng** là một chuỗi các kí tự chảy (đi) vào hoặc ra từ một tiến trình.
- Một **luồng đầu vào** được gắn vào *nguồn đầu vào* nào đó của tiến trình, vd: bàn phím hoặc socket
- Một **luồng đầu ra** được gắn vào một *nguồn đầu ra*, vd: màn hình hoặc socket.



Lập trình Socket với TCP – Ví dụ

Ví dụ ứng dụng khách-chủ:

- 1) khách đọc từng dòng từ đầu vào chuẩn (luồng **inFromUser**) , gửi cho chủ thông qua socket (luồng **outToServer**)
- 2) chủ đọc từng dòng từ socket
- 3) chủ chuyển từng dòng sang dạng viết HOA, gửi lại cho khách
- 4) khách đọc, in dòng đã được chỉnh sửa từ socket (luồng **inFromServer**)

Ví dụ: khách Java (TCP)

```
import java.io.*;
import java.net.*;
class TCPClient {
```

```
    public static void main(String argv[]) throws Exception
    {
        String sentence;
        String modifiedSentence;
```

Tạo
luồng đầu vào

```
        BufferedReader inFromUser =
            new BufferedReader(new InputStreamReader(System.in));
```

Tạo
socket khách,
kết nối tới chủ

```
        Socket clientSocket = new Socket("hostname", 6789);
```

Tạo
luồng đầu ra
gắn vào Socket

```
        DataOutputStream outToServer =
            new DataOutputStream(clientSocket.getOutputStream());
```

Ví dụ: Khách Java (TCP), tt.

Tạo
luồng đầu vào
gắn với socket

```
BufferedReader inFromServer =  
    new BufferedReader(new  
        InputStreamReader(clientSocket.getInputStream()));
```

Gửi dòng
tới chủ

```
sentence = inFromUser.readLine();  
  
outToServer.writeBytes(sentence + '\n');
```

Đọc dòng
từ chủ

```
modifiedSentence = inFromServer.readLine();  
  
System.out.println("FROM SERVER: " + modifiedSentence);  
  
clientSocket.close();
```

```
}  
  
}
```

Ví dụ: chủ Java (TCP)

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

```
class TCPServer {
```

```
    public static void main(String argv[]) throws Exception  
    {
```

```
        String clientSentence;  
        String capitalizedSentence;
```

Tạo
Socket đón khách
tại cổng 6789

```
        ServerSocket welcomeSocket = new ServerSocket(6789);
```

Chờ khách tới
liên hệ tại Socket

```
        while(true) {
```

```
            Socket connectionSocket = welcomeSocket.accept();
```

Tạo
luồng đầu vào,
gắn với socket

```
            BufferedReader inFromClient =  
                new BufferedReader(new  
                    InputStreamReader(connectionSocket.getInputStream()));
```

Ví dụ: máy chủ Java (TCP), tt

Tạo luồng đầu ra,
gắn vào socket

```
DataOutputStream outToClient =  
    new DataOutputStream(connectionSocket.getOutputStream());
```

Đọc vào 1
dòng từ socket

```
clientSentence = inFromClient.readLine();  
  
capitalizedSentence = clientSentence.toUpperCase() + '\n';
```

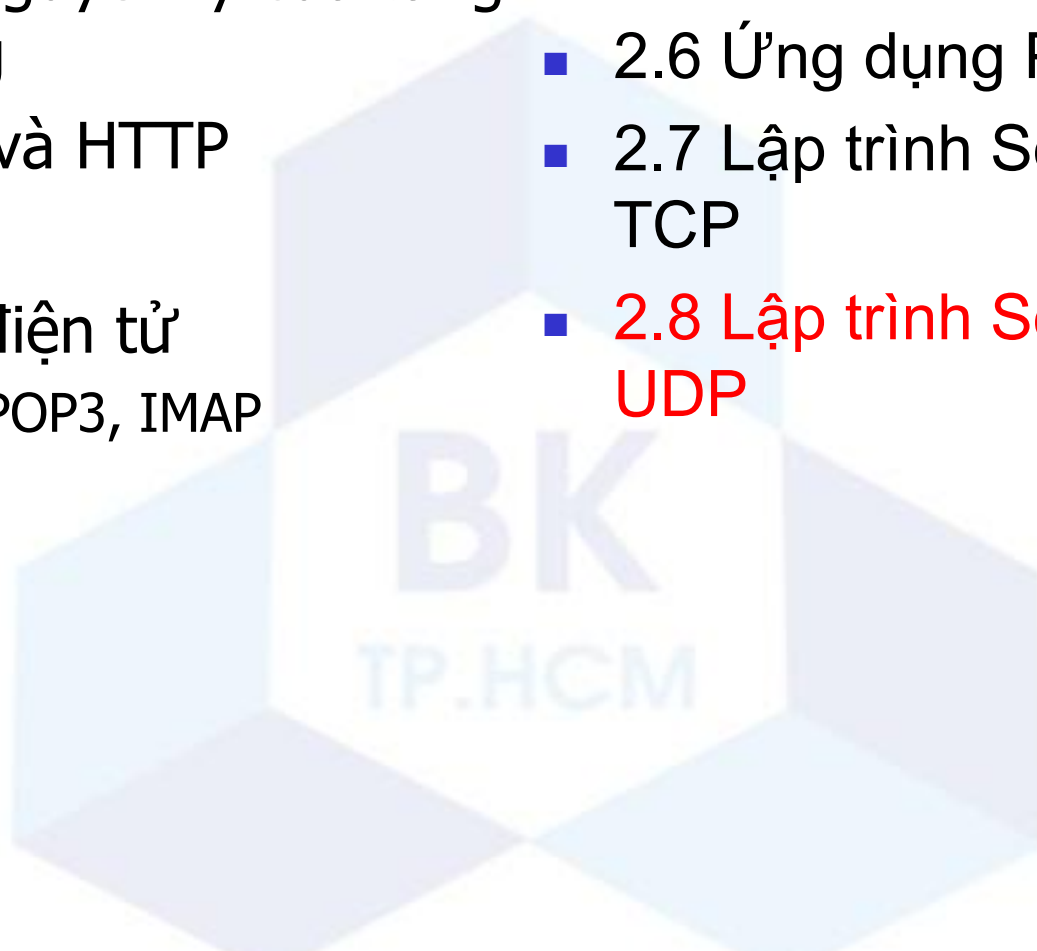
Viết một
dòng ra socket

```
outToClient.writeBytes(capitalizedSentence);  
    }  
    }  
}
```

đoạn cuối vòng lặp "while",
quay trở lại và chờ kết nối của khách khác

Chương 2: Tầng ứng dụng

- 2.1 Các nguyên lý của tầng ứng dụng
- 2.2 Web và HTTP
- 2.3 FTP
- 2.4 Thư điện tử
 - SMTP, POP3, IMAP
- 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*

UDP: không có “kết nối” giữa khách và chủ

- không bắt tay
- người gửi gán địa chỉ IP và cổng của người nhận vào mỗi gói tin
- máy chủ phải tách địa chỉ IP và cổng của người gửi từ gói tin nhận được

UDP: dữ liệu truyền tải có thể bị lộn xộn trật tự hoặc là bị mất

từ góc nhìn của Ứ/D

UDP cung cấp sự truyền tải không tin cậy của một nhóm byte (“datagrams”) giữa khách và chủ

Tương tác socket khách/chủ: UDP

Máy chủ (chạy trên **hostid**)

Khách

tạo socket,
port= x.
serverSocket =
DatagramSocket()

đọc gói tin từ
serverSocket

viết phản hồi tới
serverSocket
chỉ rõ địa chỉ,
số cổng của khách

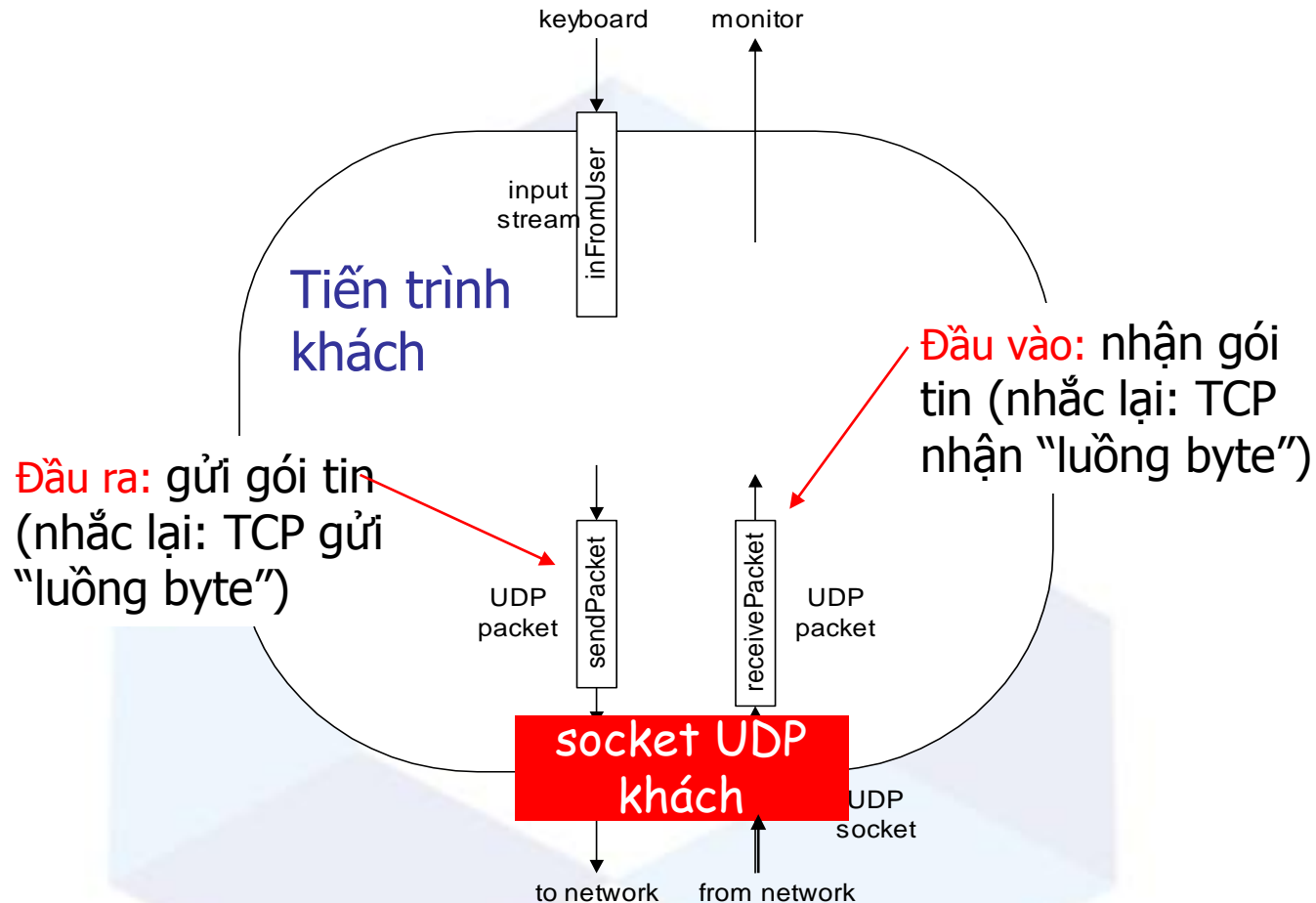
tạo socket,
clientSocket =
DatagramSocket()

tạo gói tin với IP máy chủ và
port=x; gửi gói tin thông qua
clientSocket

đọc gói tin từ
clientSocket

đóng
clientSocket

Ví dụ: Khách Java (UDP)



Ví dụ: khách Java (UDP)

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

```
class UDPClient {  
    public static void main(String args[]) throws Exception  
    {
```

Tạo
luồng đầu vào

```
        BufferedReader inFromUser =  
            new BufferedReader(new InputStreamReader(System.in));
```

Tạo
socket khách

```
        DatagramSocket clientSocket = new DatagramSocket();
```

Dịch tên máy
sang địa chỉ IP
sử dụng DNS

```
        InetAddress IPAddress = InetAddress.getByName("hostname");
```

```
        byte[] sendData = new byte[1024];  
        byte[] receiveData = new byte[1024];
```

```
        String sentence = inFromUser.readLine();  
        sendData = sentence.getBytes();
```

Ví dụ: khách Java client (UDP), tt.

Tạo gói tin
với dữ liệu,
độ dài, IP, cổng

```
DatagramPacket sendPacket =  
    new DatagramPacket(sendData, sendData.length, IPAddress, 9876);
```

Gửi gói tin
tới máy chủ

```
clientSocket.send(sendPacket);
```

```
DatagramPacket receivePacket =  
    new DatagramPacket(receiveData, receiveData.length);
```

Đọc gói tin
từ máy chủ

```
clientSocket.receive(receivePacket);
```

```
String modifiedSentence =  
    new String(receivePacket.getData());
```

```
System.out.println("FROM SERVER:" + modifiedSentence);  
clientSocket.close();  
}
```

```
}
```

Ví dụ: máy chủ Java (UDP)

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

```
class UDPServer {  
    public static void main(String args[]) throws Exception  
    {
```

Tạo
socket UDP
tại cổng 9876



```
DatagramSocket serverSocket = new DatagramSocket(9876);
```

```
byte[] receiveData = new byte[1024];  
byte[] sendData = new byte[1024];
```

```
while(true)  
{
```

Tạo bộ nhớ cho
gói tin đến



```
DatagramPacket receivePacket =  
    new DatagramPacket(receiveData, receiveData.length);
```

Nhận
gói tin



```
serverSocket.receive(receivePacket);
```

Ví dụ: Máy chủ Java (UDP), tt

```
String sentence = new String(receivePacket.getData());
```

Lấy địa chỉ IP
số cổng, của
người gửi

```
InetAddress IPAddress = receivePacket.getAddress();
```

```
int port = receivePacket.getPort();
```

```
String capitalizedSentence = sentence.toUpperCase();
```

```
sendData = capitalizedSentence.getBytes();
```

Tạo ra gói tin
để gửi tới khách

```
DatagramPacket sendPacket =  
    new DatagramPacket(sendData, sendData.length, IPAddress,  
        port);
```

Viết gói tin
ra socket

```
serverSocket.send(sendPacket);
```

```
}  
}  
}
```

Đoạn cuối vòng lặp, quay lại và
chờ gói tin khác

Chương 2: Tổng kết

Kết thúc tìm hiểu về các ứng dụng mạng!

- kiến trúc ứng dụng
 - khách-chủ
 - P2P
 - lai
- các yêu cầu dịch vụ của ứng dụng:
 - sự tin cậy, băng thông, độ trễ
- mô hình dịch vụ truyền tải Internet
 - hướng kết nối, tin cậy: TCP
 - ko tin cậy, gói tin: UDP
- các giao thức cụ thể:
 - HTTP
 - FTP
 - SMTP, POP, IMAP
 - DNS
 - P2P: BitTorrent, Skype
- lập trình hốc kết nối

Chương 2: Tổng kết

Quan trọng nhất: hiểu về các giao thức

- sự trao đổi truy vấn/phản hồi:
 - khách yêu cầu thông tin hoặc dịch vụ
 - máy chủ phản hồi với dữ liệu, mã trạng thái
- các định dạng t/điệp:
 - mào đầu: các trường mang thông tin về dữ liệu
 - dữ liệu: thông tin được trao đổi

Các chủ đề quan trọng:

- t/điệp điều khiển vs. dữ liệu
 - nội tuyến, ngoại tuyến
- tập trung vs. phân tán
- không trạng thái vs. có trạng thái
- truyền tải t/điệp tin cậy vs. ko tin cậy