Thẻ ghi DNS

DNS: CSDL phân bố chứa thẻ tài nguyên (RR)

định dạng RR: **(tên, giá trị, loại, tgs)**

- Loại A (Type=A)
  - **tên** là tên máy
  - **giá trị** là địa chỉ IP
- Type=NS
  - **tên** là tên miền (vd: foo.com)
  - **giá trị** là tên máy của máy chủ DNS có thẩm quyền cho miền này
- Type=CNAME
  - **tên** là tên thay thế cho tên chính thống

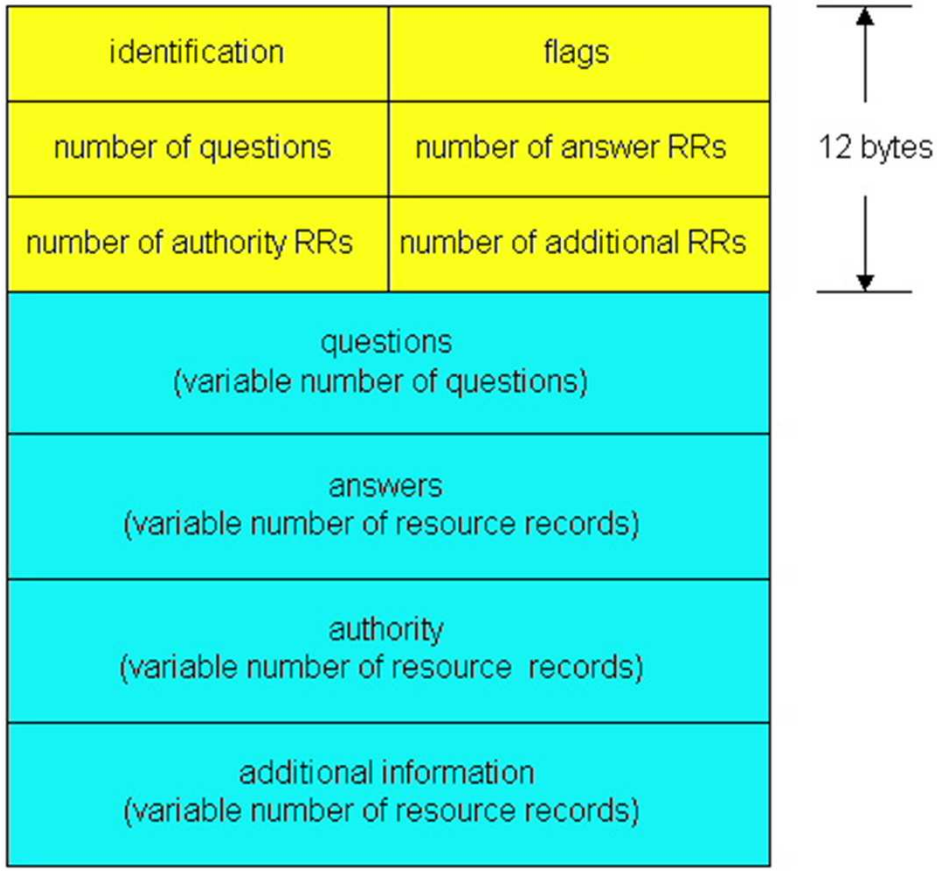
    `www.ibm.com` thực chất là `servereast.backup2.ibm.com`
  - **giá trị** là tên chính thống
- Type=MX
  - **giá trị** là tên của máy chủ thư gắn với máy "**tên**"

Giao thức và thông điệp DNS

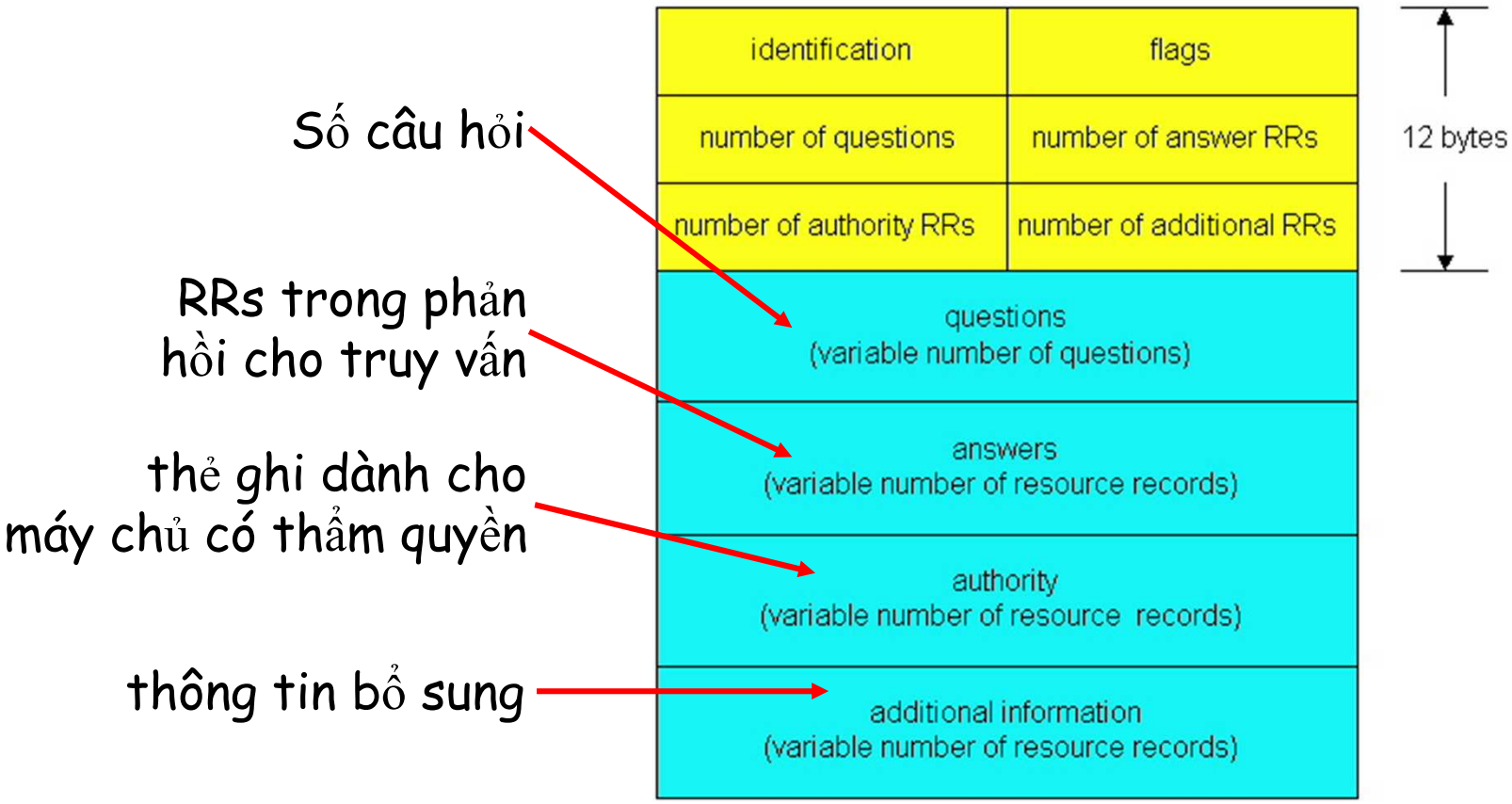
Giao thức DNS: *truy vấn (query)* và *phản hồi (reply)* có cùng *định dạng*

Phần mào đầu

- số hiệu định danh: là một số 16 bit trong thông điệp truy vấn, thông điệp phản hồi sử dụng chính số đó
- cờ hiệu:
  - truy vấn hay phản hồi
  - sử dụng đệ qui
  - đệ qui sẵn sàng
  - phản hồi có thẩm quyền



Giao thức và thông điệp DNS



Chèn thẻ ghi vào DNS

- ví dụ: một công ty mới thành lập "Network Utopia"
- đăng kí tên networkuptopia.com tại *DNS nhà quản lý tên miền* (vd: Network Solutions)
  - cung cấp tên, địa chỉ IP, địa chỉ IP của máy chủ dsn có thẩm quyền (sơ cấp và thứ cấp)
  - nhà quản lý chèn 2 thẻ RR vào máy chủ TLD "com" :

```
(networkutopia.com, dns1.networkutopia.com, NS)
(dns1.networkutopia.com, 212.212.212.1, A)
```

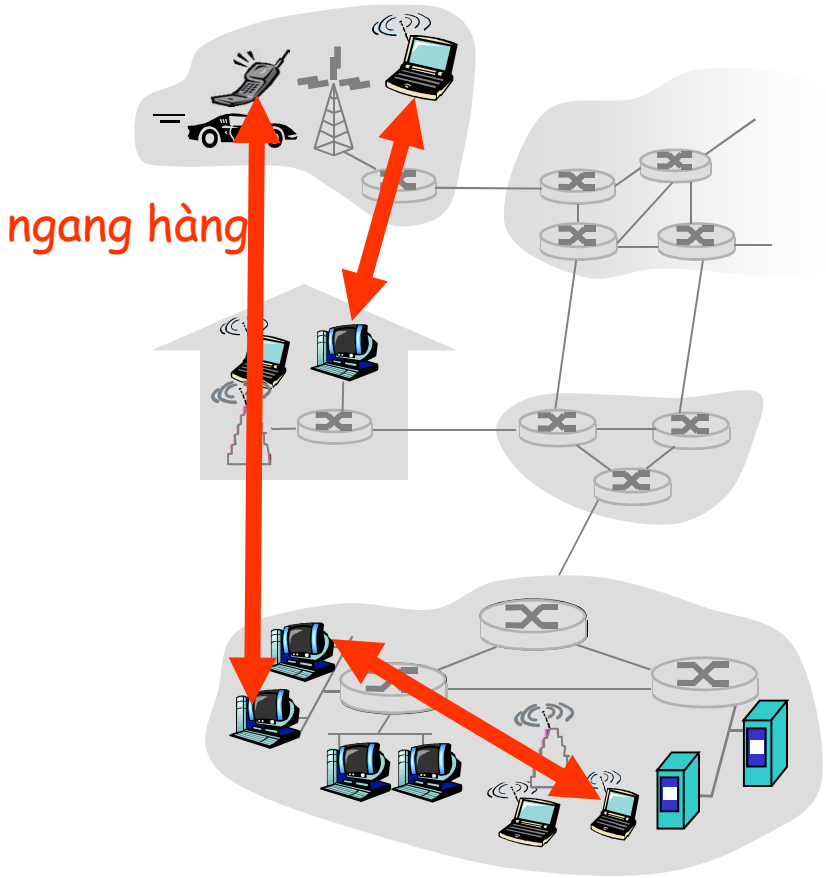
- trong máy chủ tên miền cục bộ tạo ra thẻ loại A cho www.networkuptopia.com; thẻ loại MX cho networkutopia.com
- Làm sao mọi người lấy được địa chỉ IP của trang web của bạn?

Chapter 2: Application layer

- 2.1 Principles of network applications
  - app architectures
  - app requirements
- 2.2 Web and HTTP
- 2.4 Electronic Mail
  - SMTP, POP3, IMAP
- 2.5 DNS
- 2.6 P2P applications
- 2.7 Socket programming with TCP
- 2.8 Socket programming with UDP

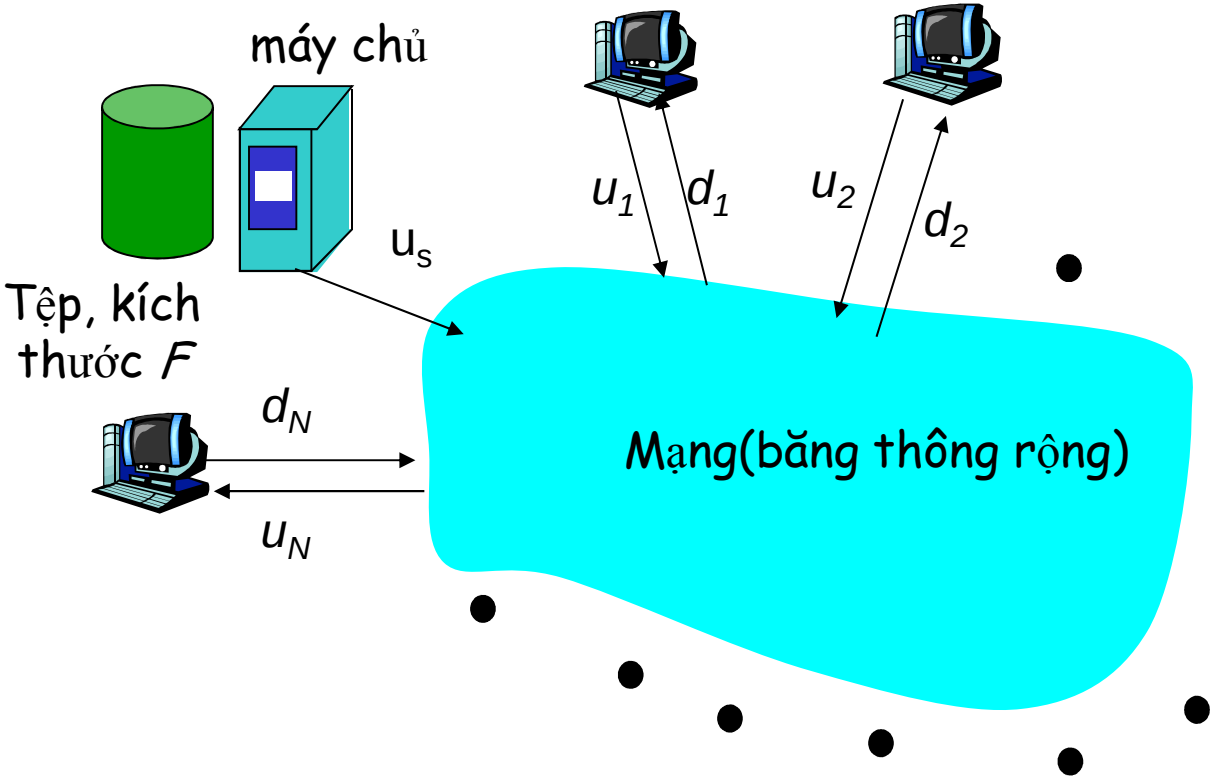
Cấu trúc P2P thuần túy

- máy chủ *không* luôn luôn mở
- nhiều máy đầu cuối giao tiếp trực tiếp với nhau
- các bên kết nối không liên tục và có thể có địa chỉ IP động
- 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 t/g để phân phối tệp từ 1 máy chủ tới N người dùng?



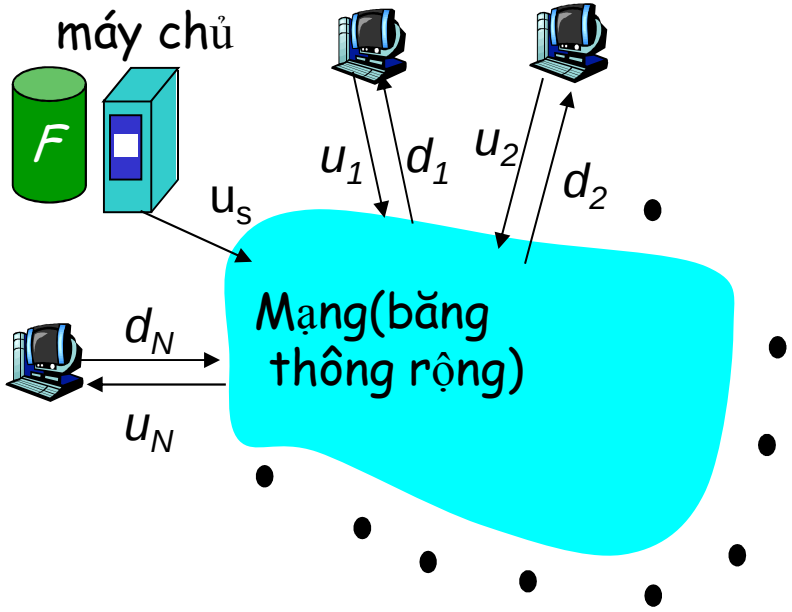
u_s: băng thông tải lên của máy chủ

u_i: băng thông tải lên của mỗi khách

d_i: băng thông tải xuống của mỗi khách

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

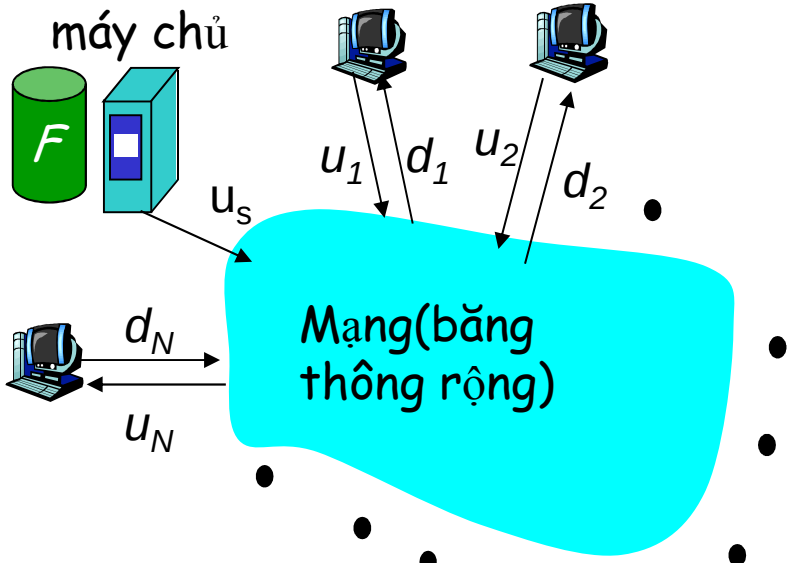


Thời gian để phân phối F tới N khách sử dụng = $d_{cs} = \max\{ NF/u_s, F/\min_i(d_i) \}$ mô hình khách/chủ

tăng tuyến tính theo N (với N lớn)

Phân phối tệp tin: P2P

- máy chủ tải lên một bản sao trong t/g: F/u_s
- khách *i* cần F/d_i t/g để tải về
- NF bit phải được tải lên

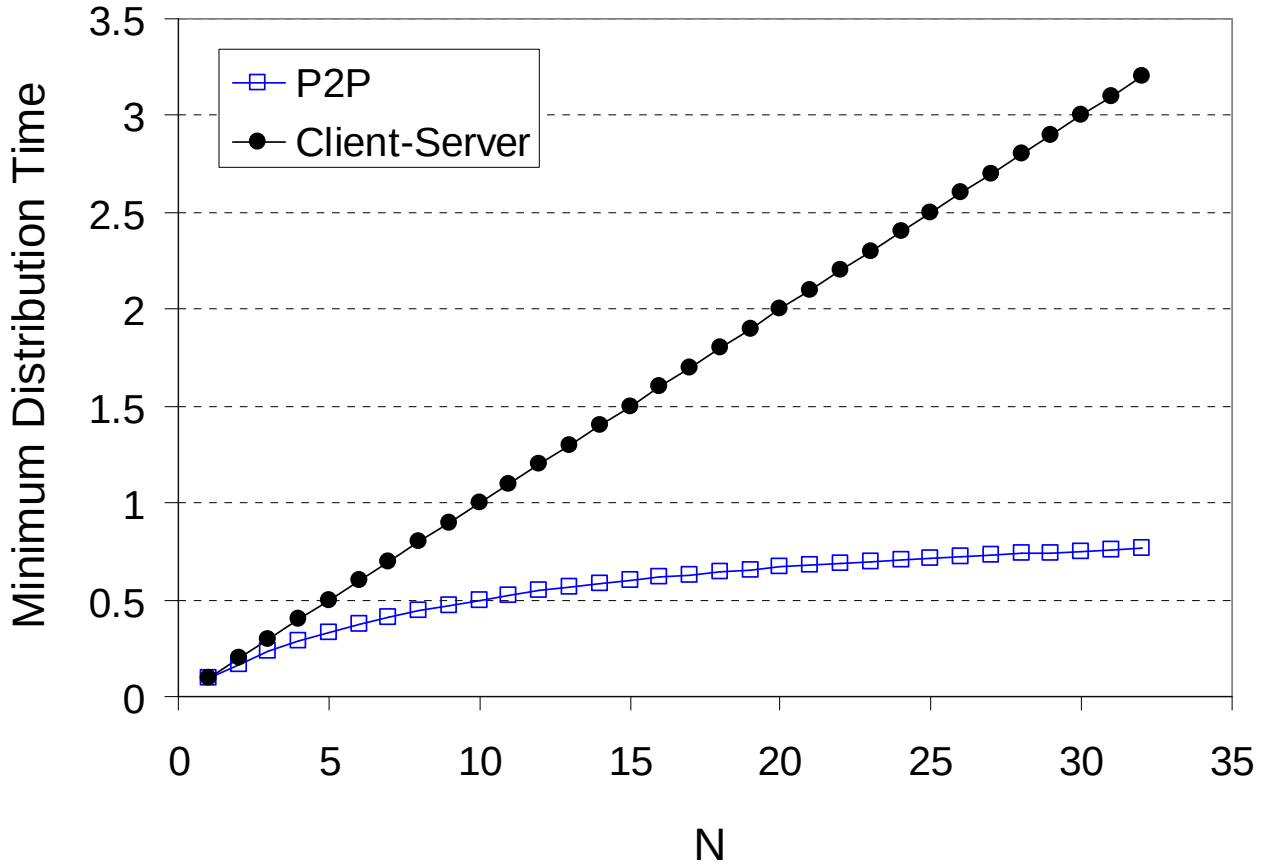


- vận tốc tải lên nhanh nhất có thể: $u_s + \Sigma u_i$

$$d_{P2P} = \max\{F/u_s, F/\min(d_i), NF/(u_s + \sum_{i=1}^{N} u_i)\}$$

Khách-chủ so với P2P: ví dụ

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

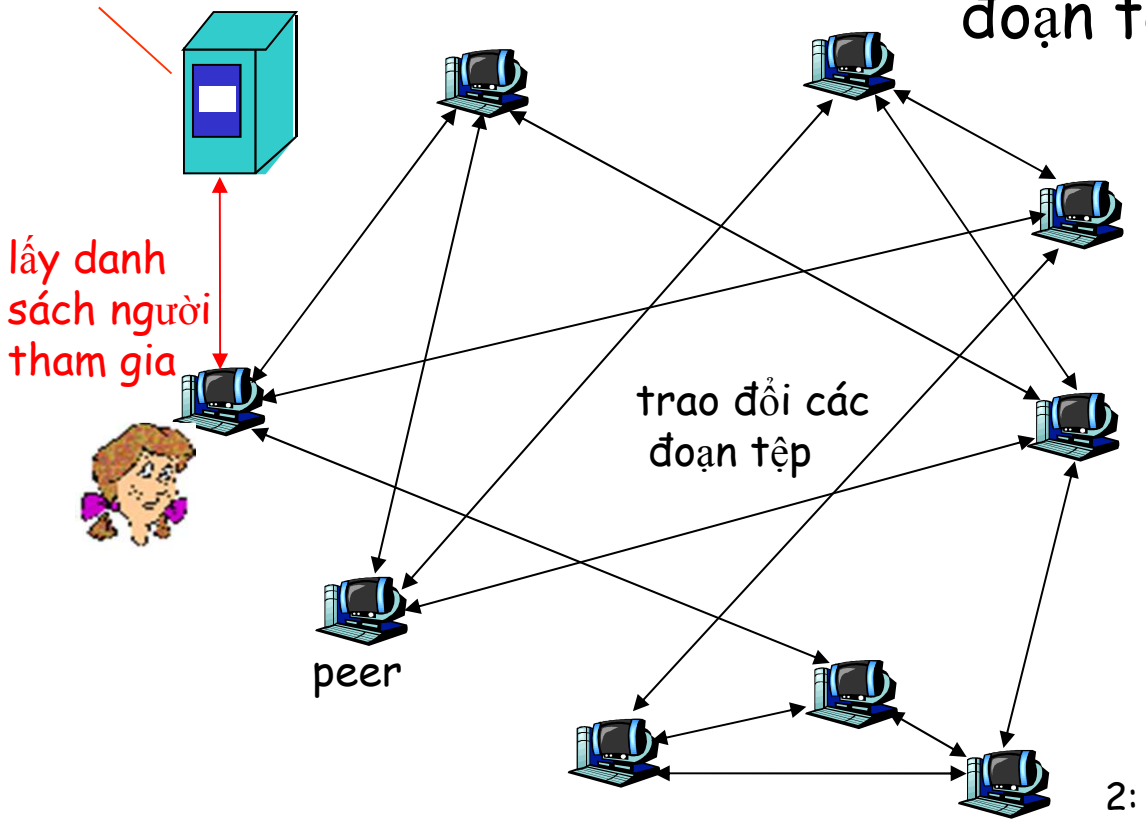


Phân phối tệp tin: BitTorrent

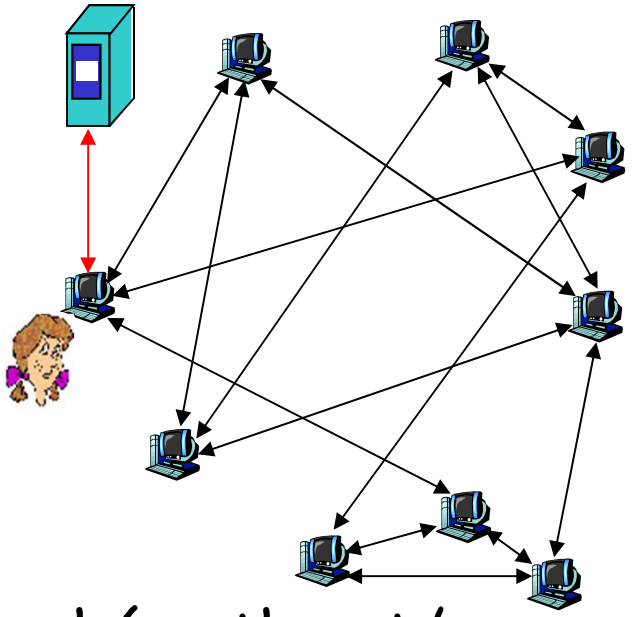
- phân phối tệp kiểu P2P

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

torrent: nhóm các cá nhân chia sẻ những đoạn tệp



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)

Tải các đoạn tệp

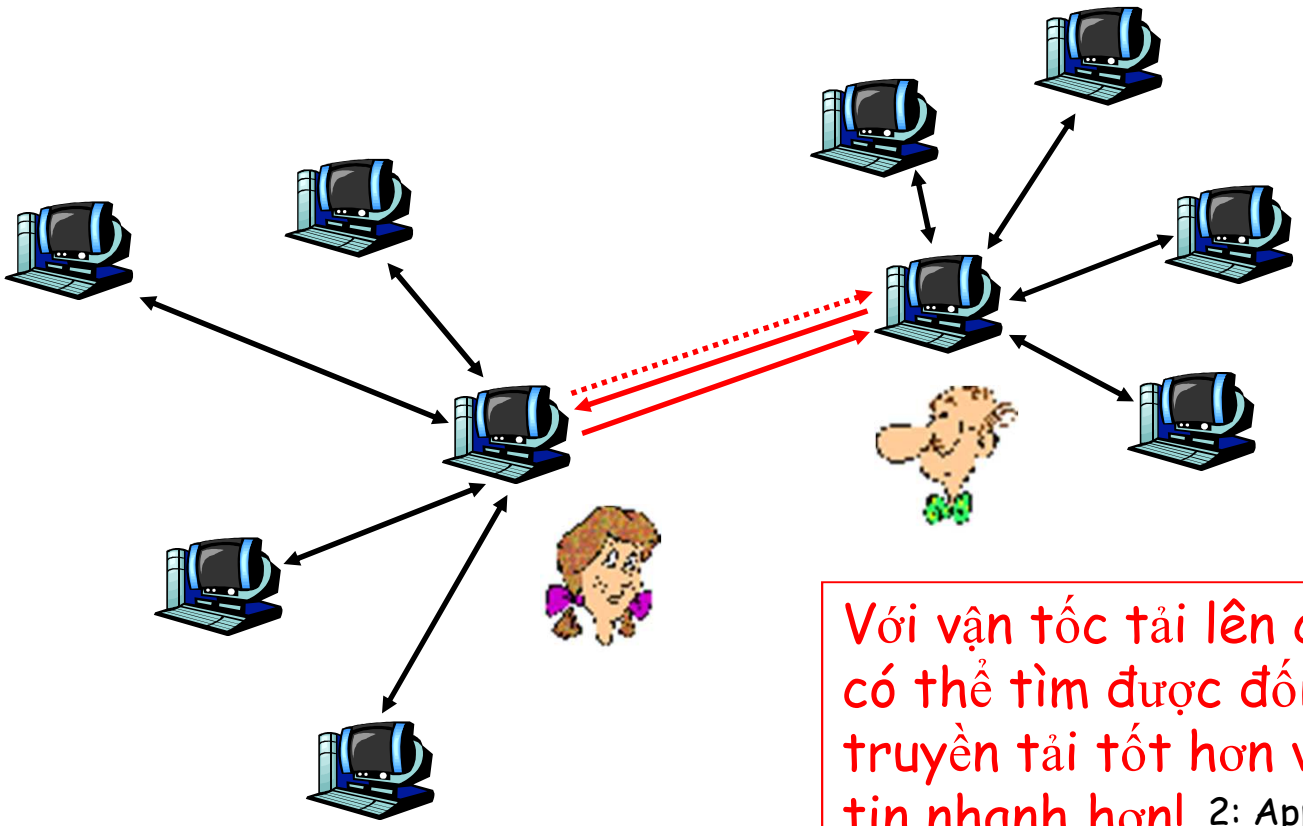
- tại bất kì thời điểm nào, các thành viên khác nhau sẽ có những đoạn khác nhau của một tệp
- một cách định kì, một thành viên (Alice) sẽ yêu cầu từ các hàng xóm danh sách các đoạn mà họ có.
- Alice gửi yêu cầu tới các đoạn mà cô ta thiếu
  - đoạn hiếm nhất trước

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 10 s
- sau mỗi 30 s: chọn ngẫu nhiên một thành viên khác, và gửi đoạn cho nó
  - thành viên mới này có thể vào tóp 4

BitTorrent: Tit-for-tat

(1) Alice "gửi thử nghiệm" cho Bob
(2) Alice trở thành 1 trong tốp 4 của Bob; Bob gửi phản hồi
(3) Bob trở thành 1 trong tốp 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

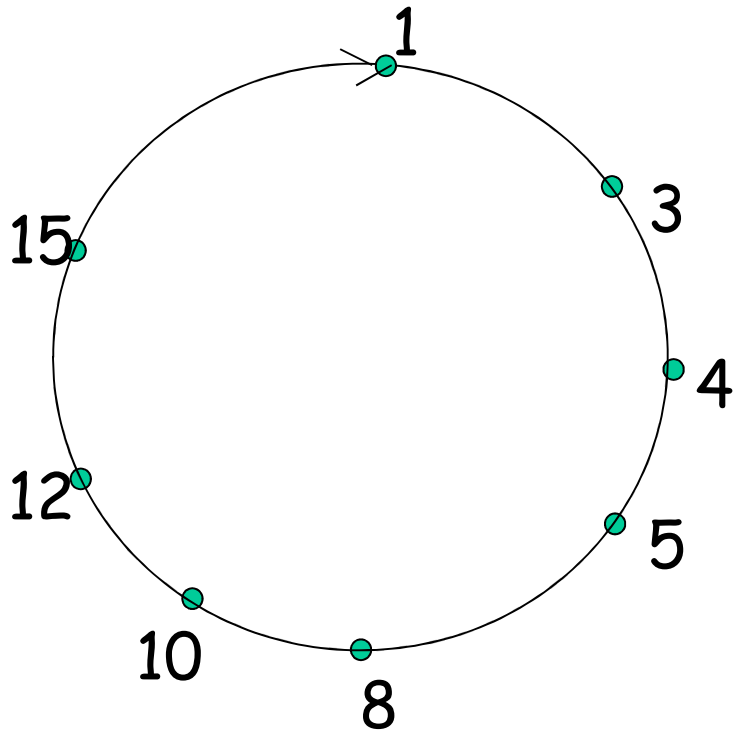
Định danh DHT

- 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 bít.
- 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: key = h("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.
- Qui luật: gán khóa cho thành viên mà có ID gần nhất.
- Qui 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
  - khóa = 15, thành viên gần nhất = 1

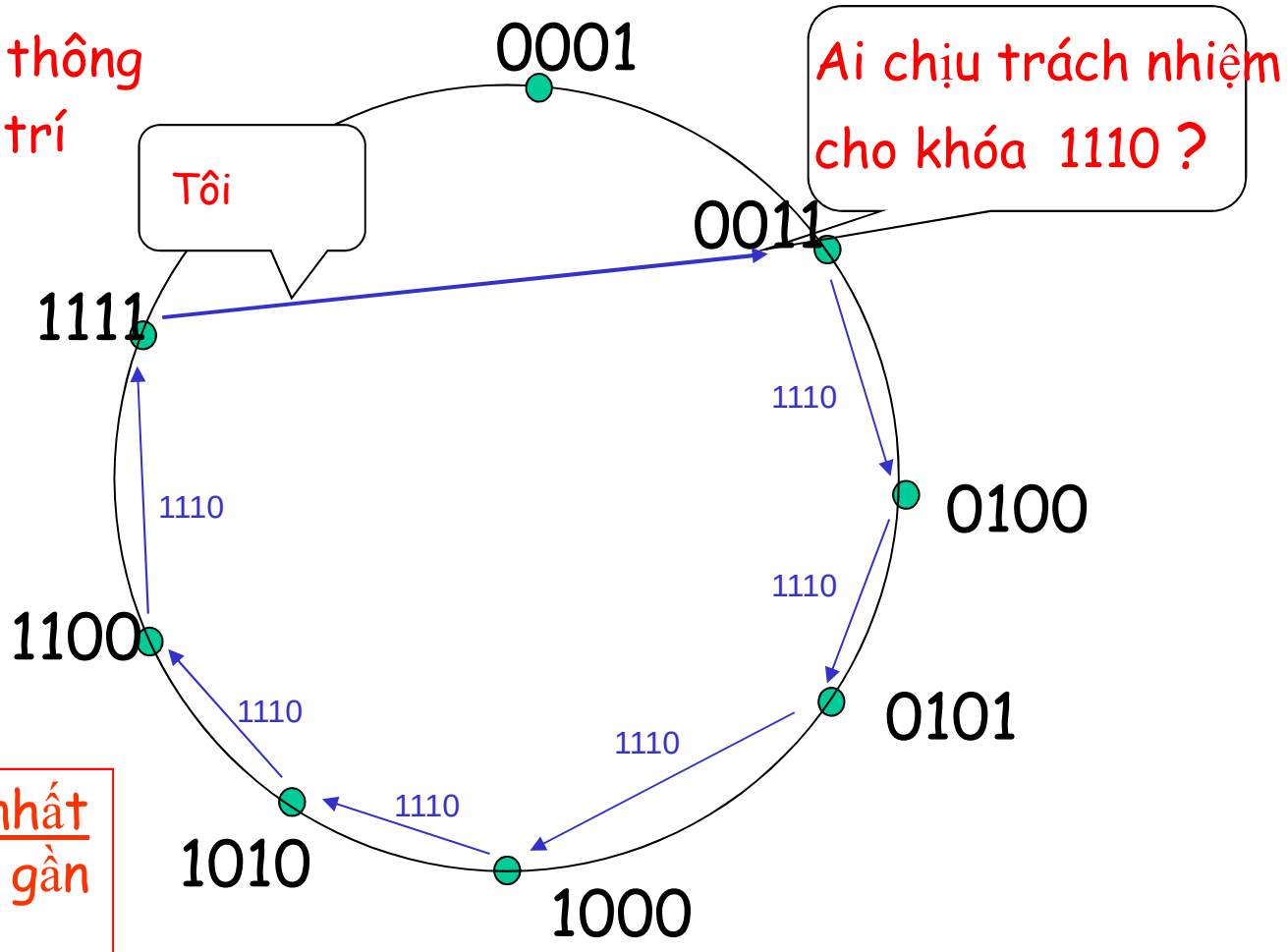
DHT xoay vòng (1)



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

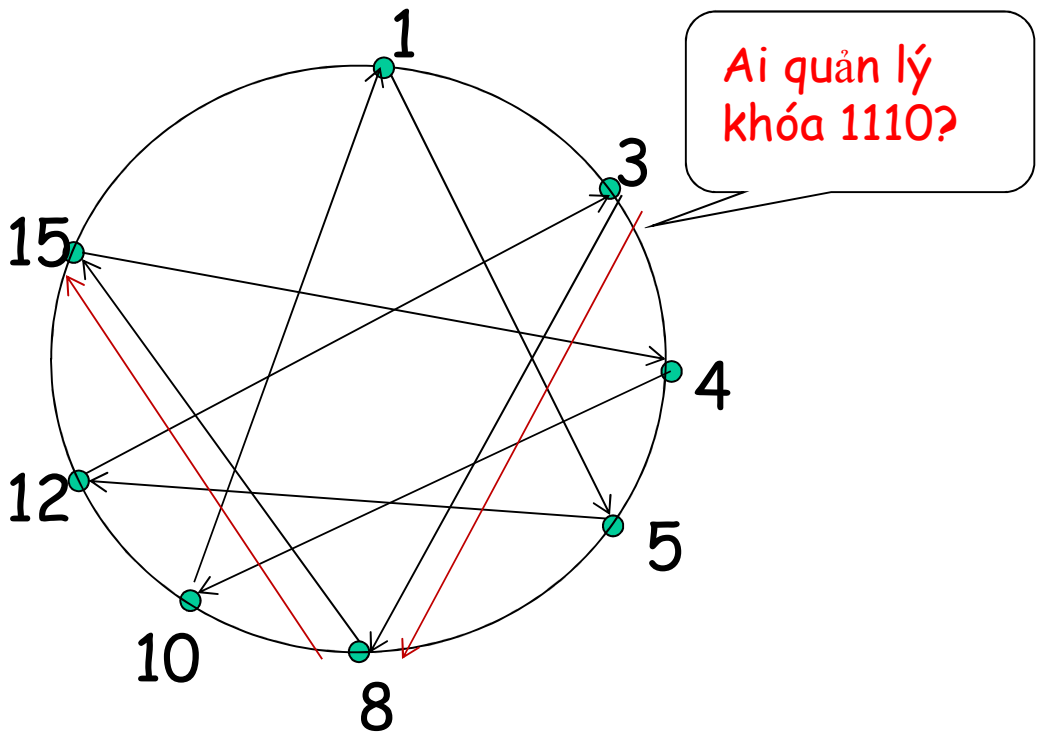
Circle DHT (2)

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



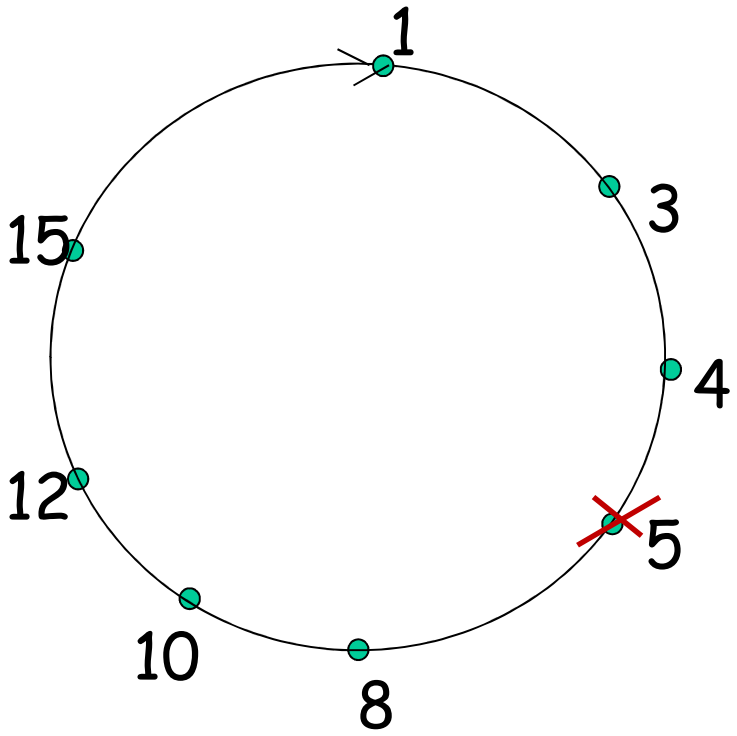
Định nghĩa gần nhất là người liền sau gần nhất

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

Peer Churn

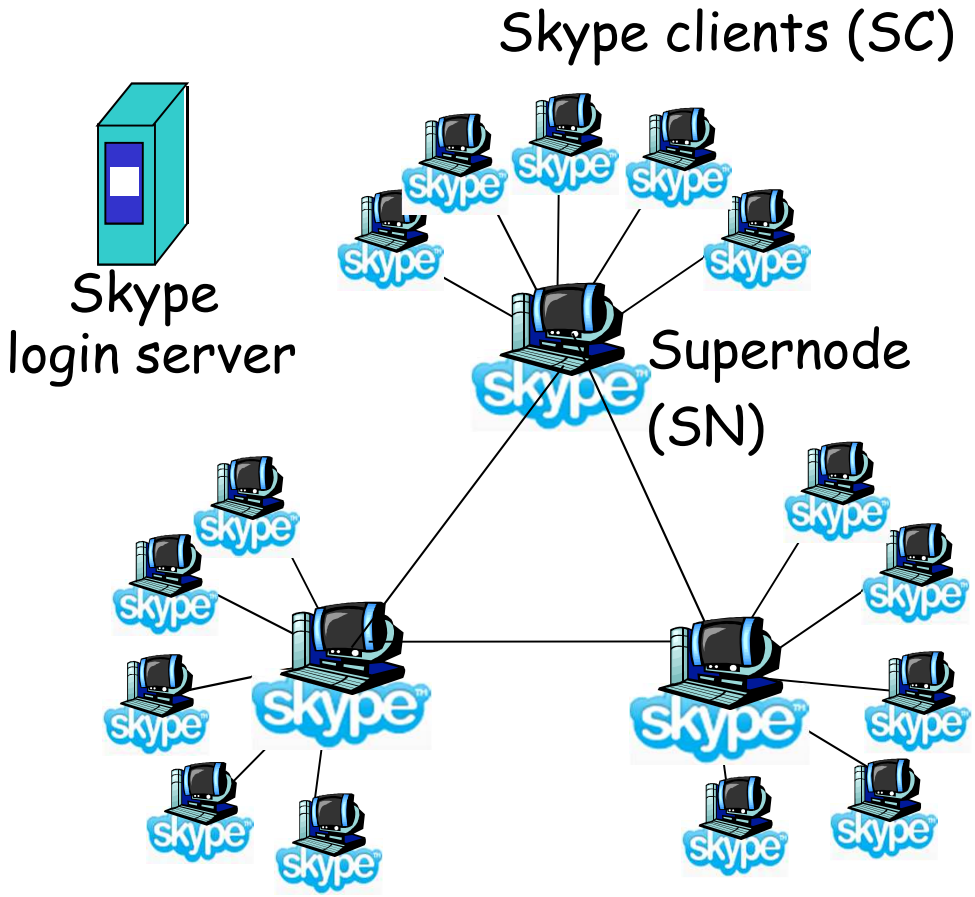


- To handle peer churn, require each peer to know the IP address of its two successors.
- Each peer periodically pings its two successors to see if they are still alive.

- Peer 5 abruptly leaves
- Peer 4 detects; makes 8 its immediate successor; asks 8 who its immediate successor is; makes 8's immediate successor its second successor.
- What if peer 13 wants to join?

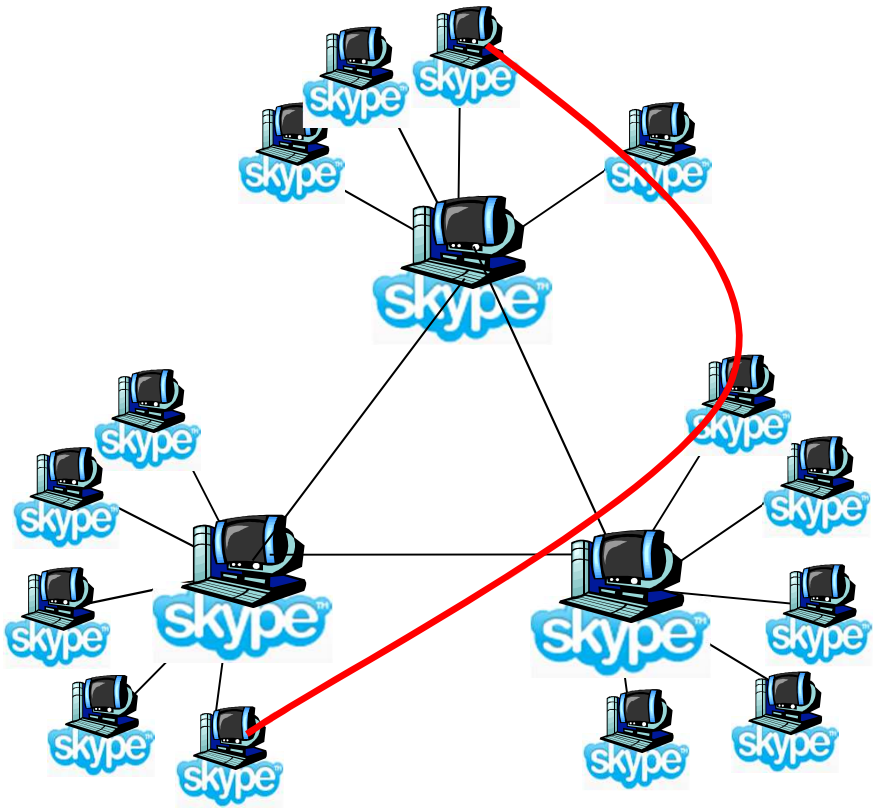
P2P Case study: Skype

- inherently P2P: pairs of users communicate.
- proprietary application-layer protocol (inferred via reverse engineering)
- hierarchical overlay with SNs
- Index maps usernames to IP addresses; distributed over SNs

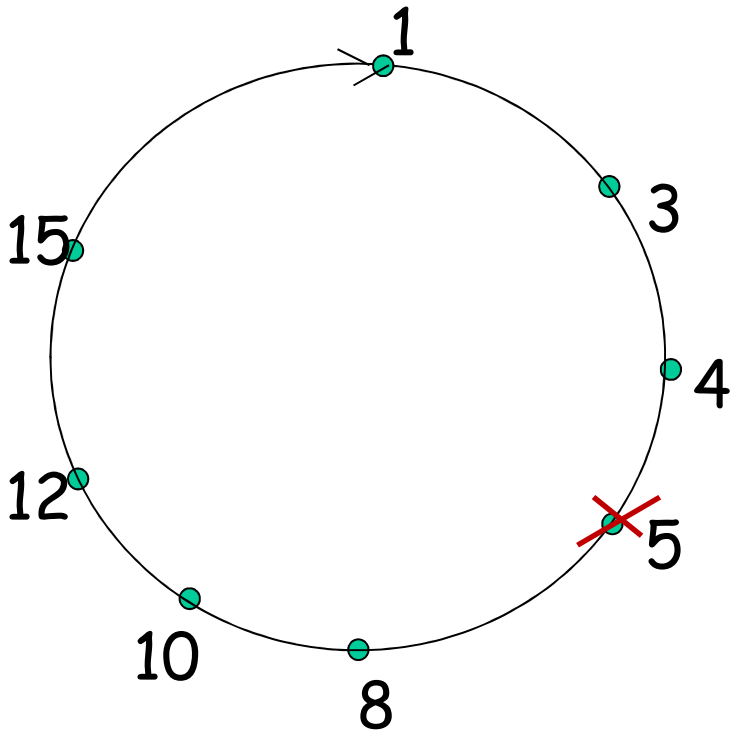


Peers as relays

- Problem when both Alice and Bob are behind "NATs".
  - NAT prevents an outside peer from initiating a call to insider peer
- Solution:
  - Using Alice's and Bob's SNs, Relay is chosen
  - Each peer initiates session with relay.
  - Peers can now communicate through NATs via relay



Peer Churn



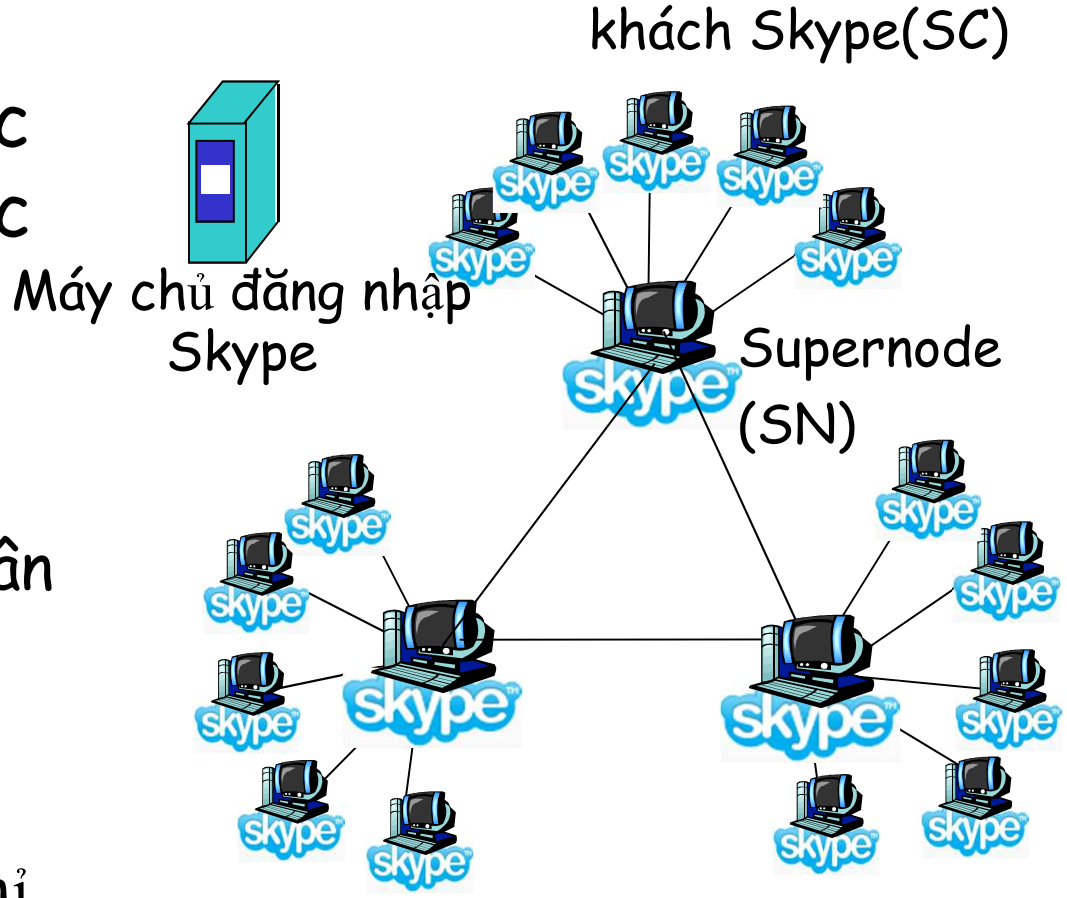
- Để xử lý peer churn, yêu cầu mỗi t/viên phải biết địa chỉ của 2 người liền sau nó.
- Mỗi t/viên theo định kì ping 2 người liền kề nó để xem họ còn trên mạng ko.

- 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?

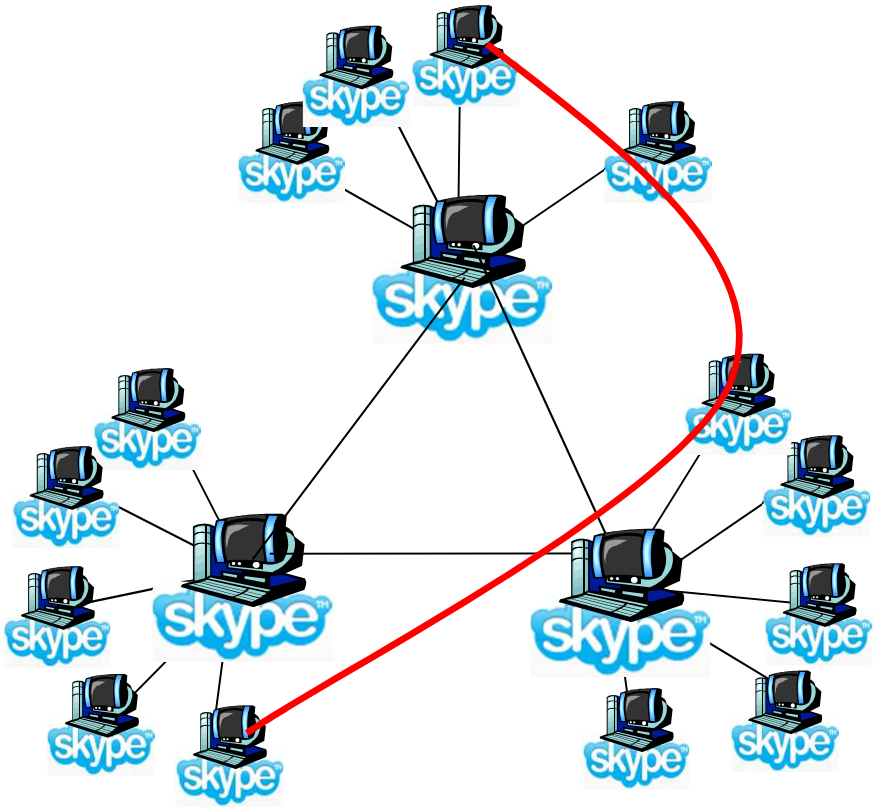
Trường hợp nghiên cứu P2P: Skype

- có tính chất P2P: các cặp n/dùng liên lạc với nhau.
- giao thức tầng ứng dụng sở hữu
- 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, với Chế độ chuyển tiếp
  - 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"



Chapter 2: Application layer

- 2.1 Principles of network applications
- 2.2 Web and HTTP
- 2.3 FTP
- 2.4 Electronic Mail
  - SMTP, POP3, IMAP
- 2.5 DNS
- 2.6 P2P applications
- 2.7 Socket programming with TCP
- 2.8 Socket programming with UDP

Lập trình hốc kết nối

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

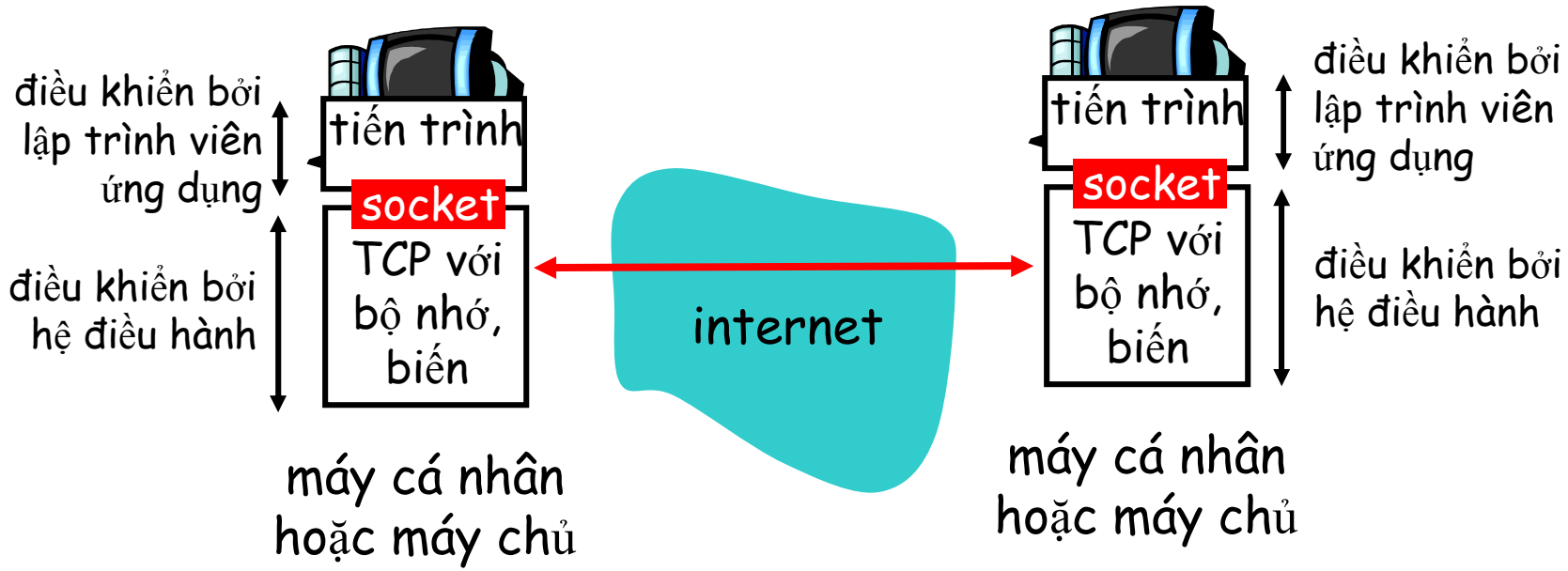
hkn

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 (cửa) và sẵn sàng nhận sự liên hệ từ khách

Khách liên hệ chủ bằng cách:

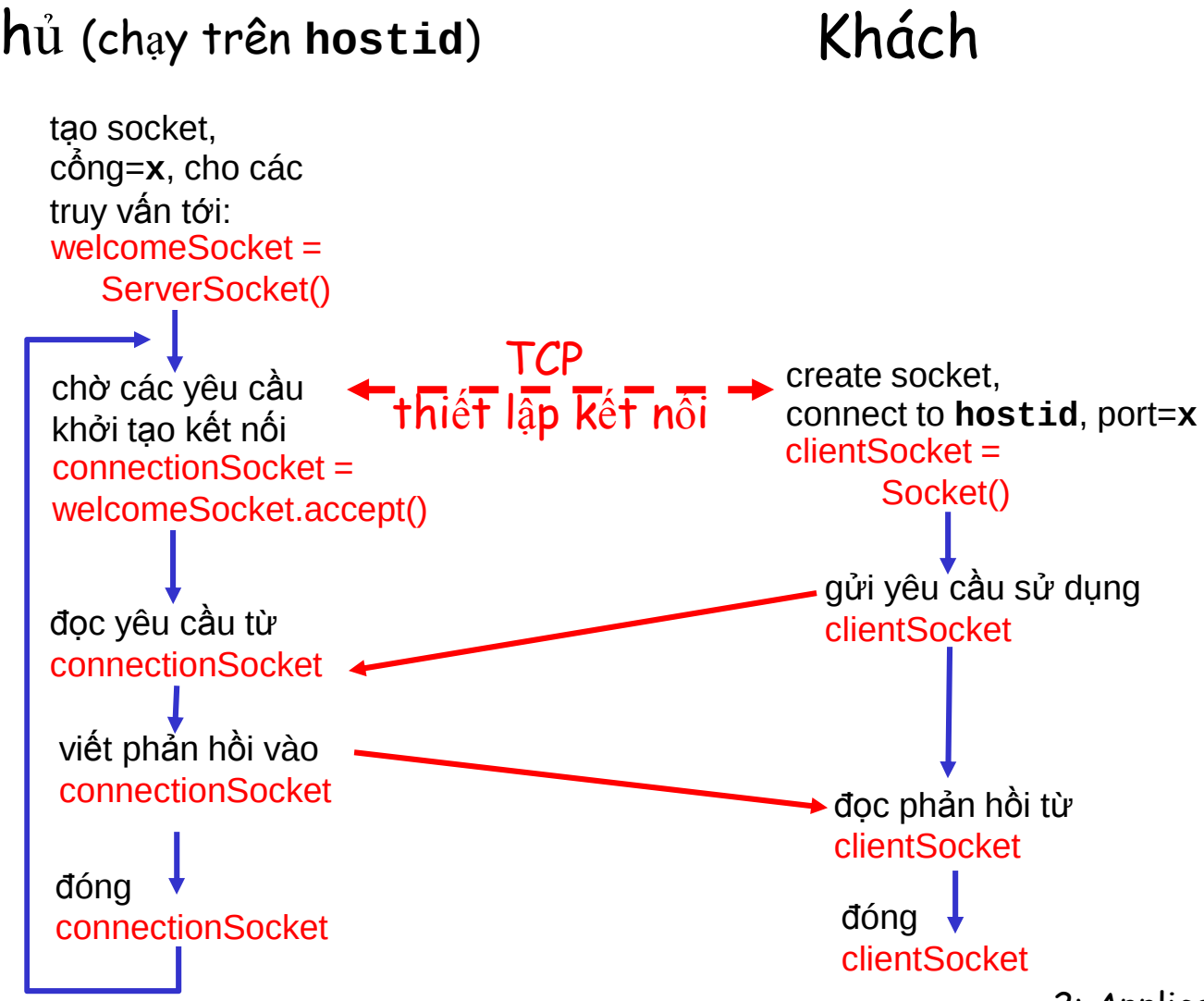
- tạo ra một socket TCP cục bộ trên máy khách
- chỉ rõ địa chỉ IP, số cổng của tiến trình chủ
- Khi khách tạo socket: khách TCP thiết lập kết nối tới máy chủ TCP

- 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

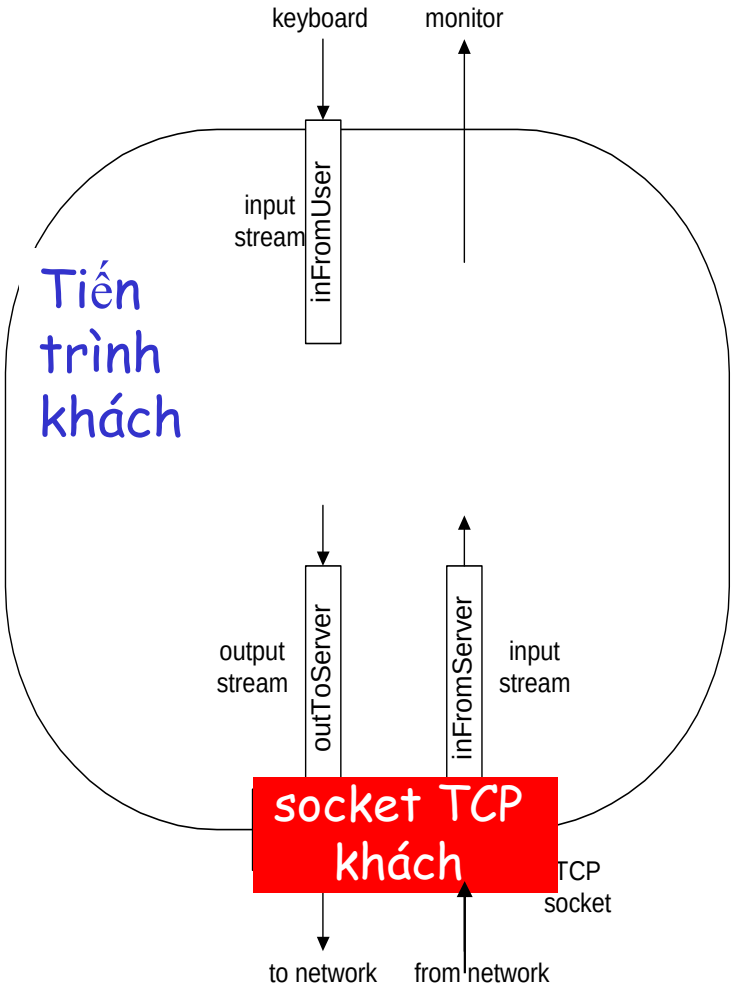
TCP cung cấp dịch vụ truyền tải tin cậy, theo thứ tự của các byte giữa khách và chủ

Tương tác socket Khách/Chủ: TCP



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 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;

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

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

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

Tạo luồng đầu vào

Tạo socket khách, kết nối tới chủ

Tạo luồng đầu ra gắn vào Socket

Ví dụ: Khách Java (TCP), tt.

Tạo luồng đầu vào gắn với socket

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

sentence = inFromUser.readLine();

outToServer.writeBytes(sentence + '\n');

modifiedSentence = inFromServer.readLine();

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

clientSocket.close();

    }
}
```

Gửi dòng tới chủ

Đọc dòng từ chủ

Ví dụ: chủ Java (TCP)

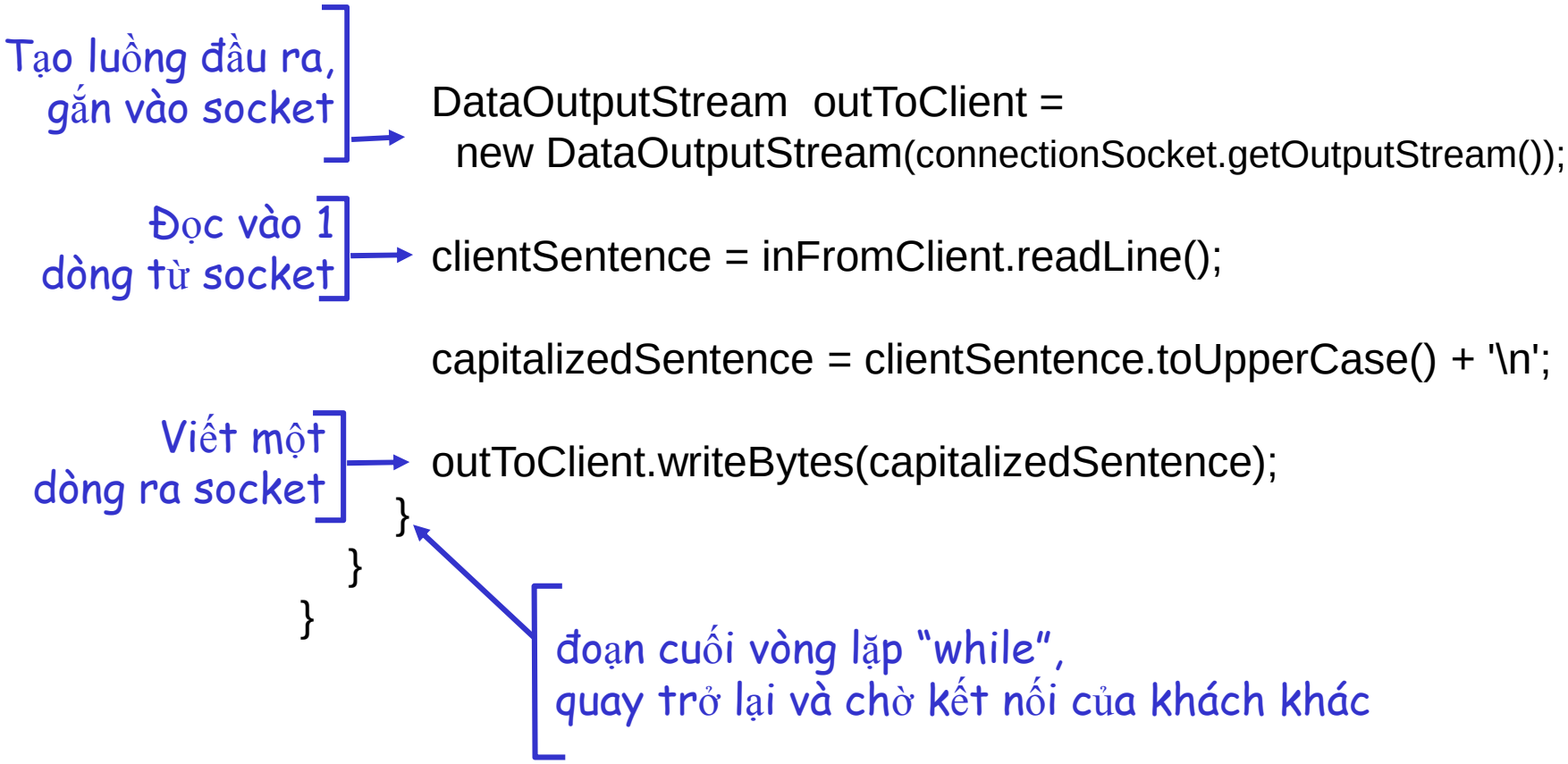
```
import java.io.*;
import java.net.*;

class TCPServer {

  public static void main(String argv[]) throws Exception
   {
     String clientSentence;
     String capitalizedSentence;

     ServerSocket welcomeSocket = new ServerSocket(6789);

     while(true) {

        Socket connectionSocket = welcomeSocket.accept();

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

Tạo Socket đón khách tại cổng 6789

Chờ khách tới liên hệ tại Socket

Tạo luồng đầu vào, gắn với socket

Ví dụ: máy chủ Java (TCP), tt



Chapter 2: Application layer

- 2.1 Principles of network applications
- 2.2 Web and HTTP
- 2.3 FTP
- 2.4 Electronic Mail
  - SMTP, POP3, IMAP
- 2.5 DNS
- 2.6 P2P applications
- 2.7 Socket programming with TCP
- 2.8 Socket programming with UDP

Lập trình Socket với UDP

UDP: không "kết nối" giữa khách và chủ

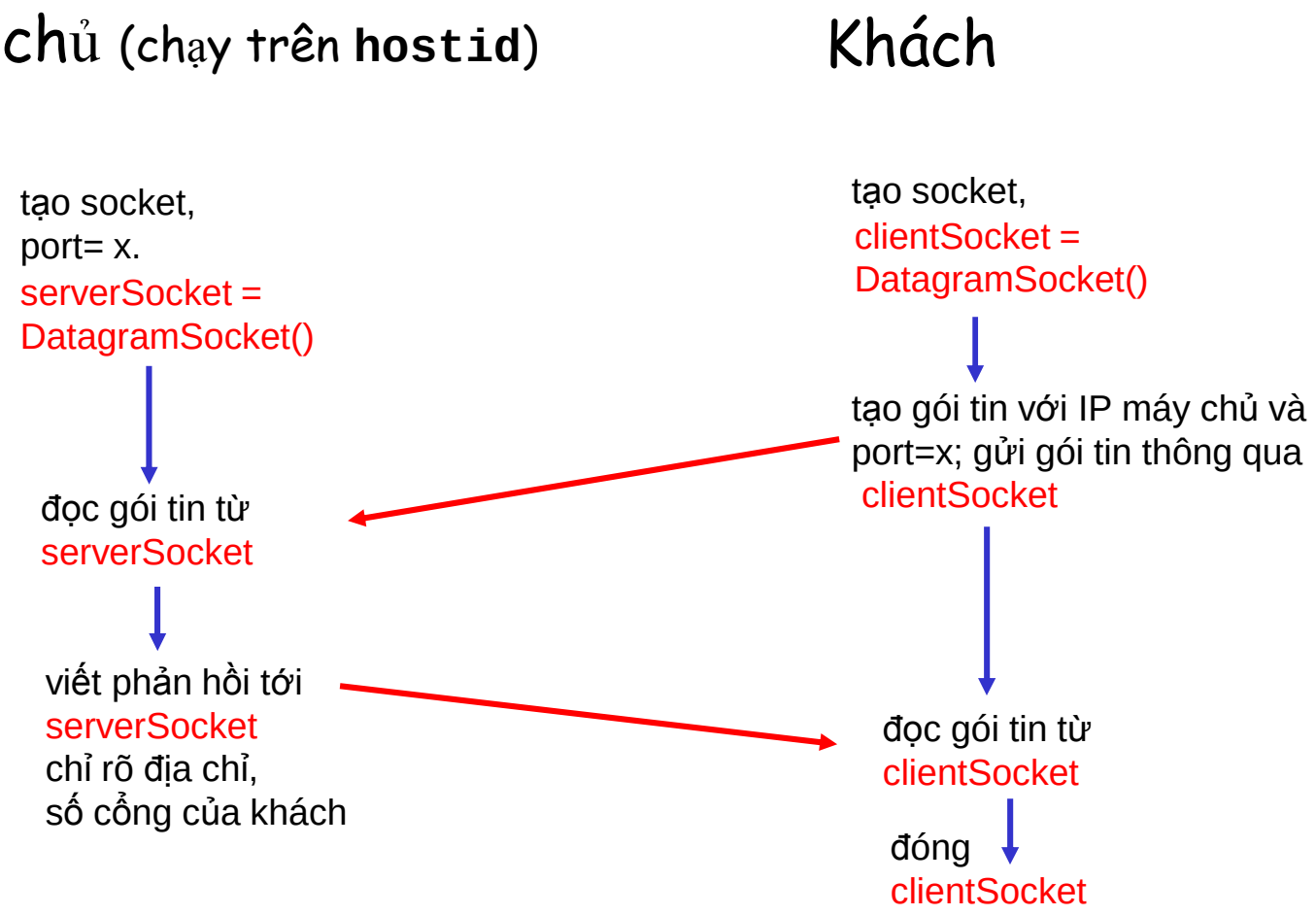
- không bắt tay
- người gửi gán địa chỉ IP và cổng của người nhận vào mỗi gói tin
- máy chủ phải tách địa chỉ IP và cổng của người gửi từ gói tin nhận được

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

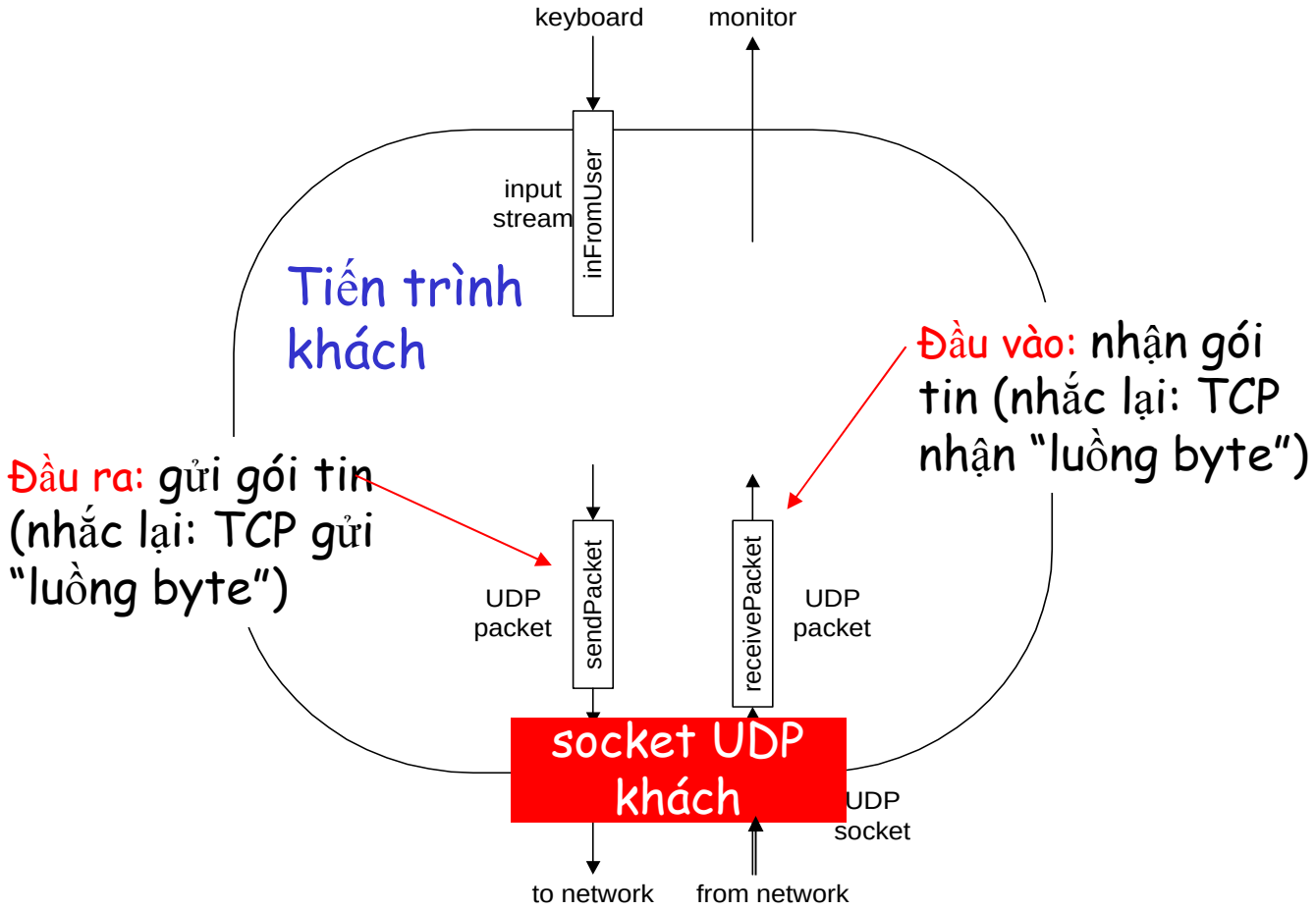
từ góc nhìn của ứ/d

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

Tương tác socket khách/chủ: UDP



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
    {

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

      DatagramSocket clientSocket = new DatagramSocket();

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

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

      String sentence = inFromUser.readLine();

      sendData = sentence.getBytes();
```

Tạo luồng đầu vào

Tạo socket khách

Dịch tên máy sang địa chỉ IP sử dụng DNS

Ví dụ: khách Java client (UDP), tt.

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

Gửi gói tin tới máy chủ

Đọc gói tin từ máy chủ

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

                clientSocket.send(sendPacket);

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

                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
   {
      DatagramSocket serverSocket = new DatagramSocket(9876);

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

      while(true)
       {
         DatagramPacket receivePacket =
           new DatagramPacket(receiveData, receiveData.length);
         serverSocket.receive(receivePacket);
```

Tạo socket UDP tại cổng 9876

Tạo bộ nhớ cho gói tin đến

Nhận gói tin

Ví dụ: Máy chủ Java (UDP), tt

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

InetAddress IPAddress = receivePacket.getAddress();

int port = receivePacket.getPort();

String capitalizedSentence = sentence.toUpperCase();

sendData = capitalizedSentence.getBytes();

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

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

Lấy địa chỉ IP số cổng, của người gửi

Tạo ra gói tin để gửi tới khách

Viết gói tin ra socket

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

Chương 2: Tổng kết

our study of network apps now complete!

- application architectures
  - client-server
  - P2P
  - hybrid
- application service requirements:
  - reliability, bandwidth, delay
- Internet transport service model
  - connection-oriented, reliable: TCP
  - unreliable, datagrams: UDP
- specific protocols:
  - HTTP
  - FTP
  - SMTP, POP, IMAP
  - DNS
  - P2P: BitTorrent, Skype
- socket programming