

Lập trình socket

Mục đích: làm thế nào để xây dựng các ứng dụng client/server truyền dữ liệu qua socket

Socket API

- ❑ Đưa ra trong BSD4.1 UNIX, 1981
- ❑ Cho phép ứng dụng tạo ra và sử dụng socket
- ❑ Nguyên lý client/server
- ❑ Có hai dạng socket
 - ❖ Truyền gói tin, không tin cậy
 - ❖ Truyền dòng bytes, tin cậy

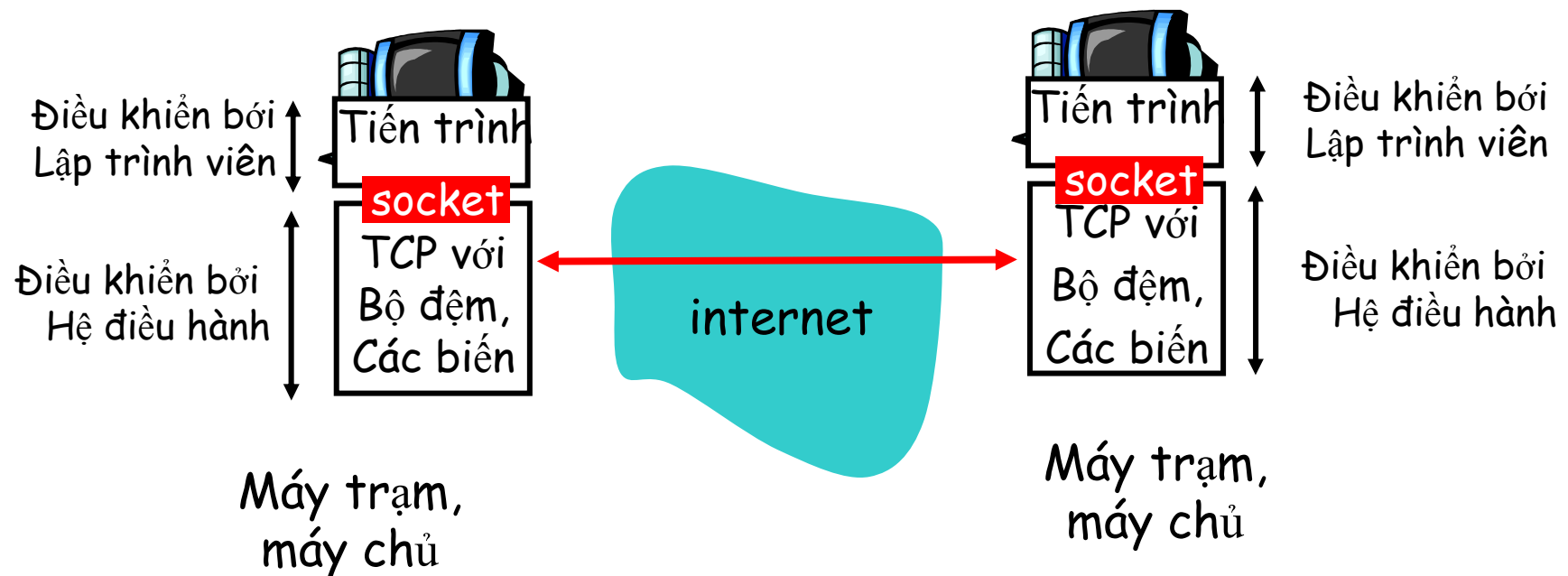
socket

Là giao diện (cửa) *do ứng dụng tạo ra trên máy trạm, quản lý bởi OS* qua đó các ứng dụng có thể **gửi và nhận** thông điệp đến/từ các ứng dụng khác

Lập trình socket với TCP

Socket: cửa giao tiếp giữa các tiến trình và giao thức giao vận (UCP hoặc TCP)

Dịch vụ TCP: truyền các **bytes** tin cậy từ một tiến trình đến các tiến trình khác



Lập trình socket với **TCP**

Client phải gửi yêu cầu tới server

- Tiến trình máy chủ phải đang được thực hiện
- máy chủ phải mở socket (cổng) để nhận yêu cầu từ client

Client yêu cầu server bằng cách:

- Tạo một socket TCP trên máy
- Chỉ rõ IP address & port number của tiến trình máy chủ
- Khi **client tạo socket**: client TCP tạo liên kết tới server TCP

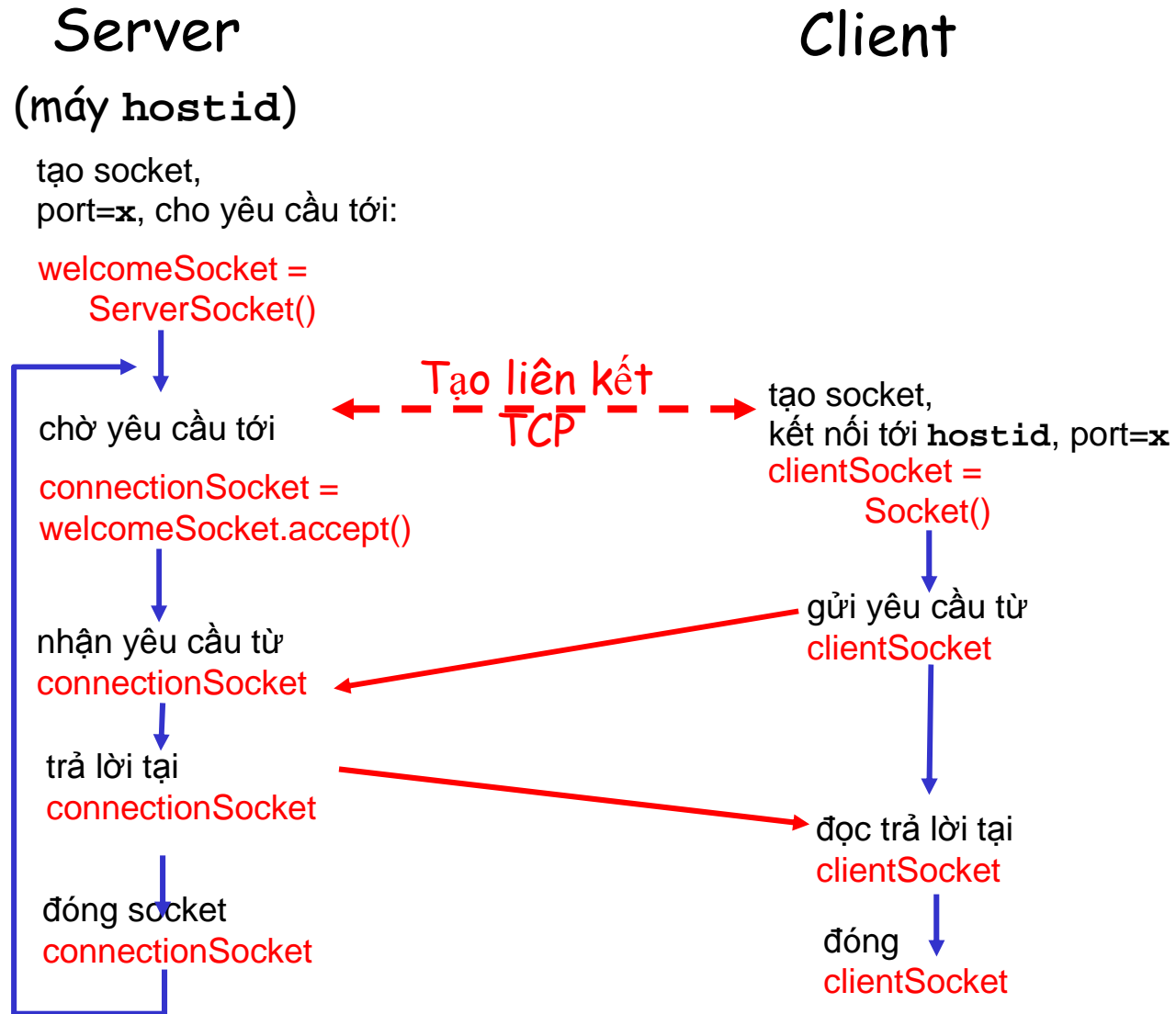
- khi được client liên lạc, **server TCP tạo socket mới** để tiến trình máy chủ giao tiếp với client
 - ❖ cho phép nói chuyện với nhiều clients
 - ❖ phân biệt client bằng số hiệu cổng (chương 3...)

Đối với ứng dụng

TCP cung cấp dịch vụ truyền dòng bytes tin cậy và có thứ tự giữa client và server

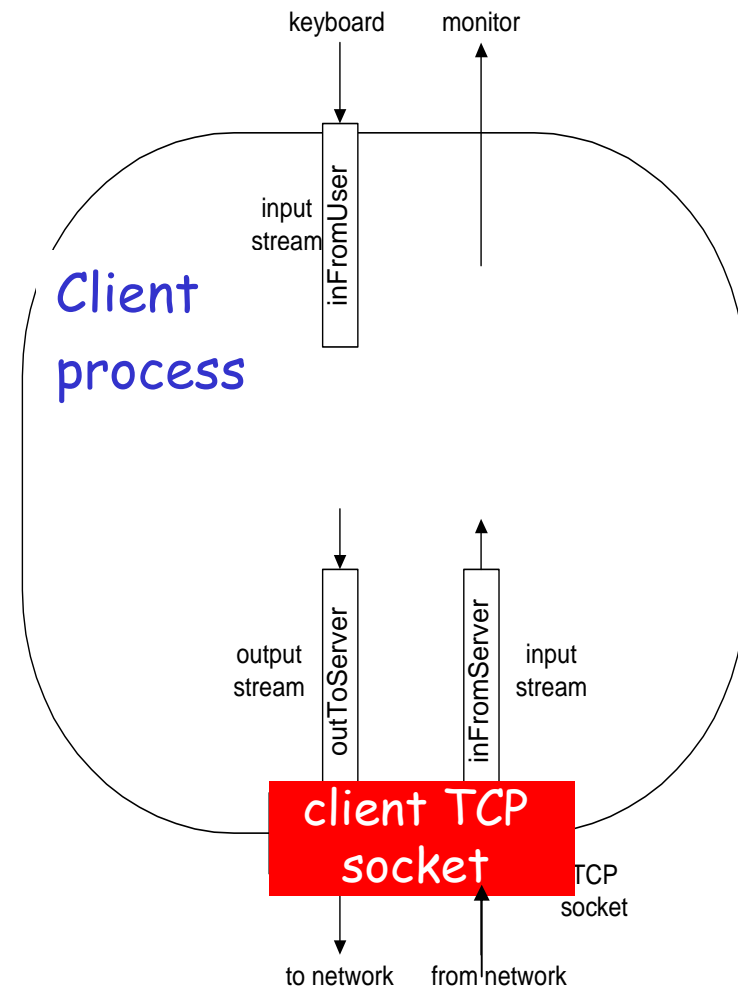
Tương tác giữa client/server qua socket

TCP



Stream

- **stream** một chuỗi ký tự vào/ra một tiến trình.
- **input stream** được gắn với một nguồn vào, e.g. bàn phím, socket
- **output stream** được gắn với một nguồn ra, e.g., màn hình socket.



Lập trình socket với TCP

Ví dụ về ứng dụng client-server:

- 1) client đọc các dòng văn bản do người dùng gõ từ bàn phím (**inFromUser** stream) , gửi tới server qua socket (**outToServer** stream)
- 2) server đọc các dòng gửi từ socket
- 3) server chuyển sang chữ hoa và gửi trả lại cho client
- 4) client đọc và in lại dòng văn bản nhận được từ socket (**inFromServer** stream)

Ví dụ: Java client (TCP)

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

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

```
        String sentence;
        String modifiedSentence;
```

Tạo
input stream



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

Tạo
client socket,
nối tới server



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

Tạo
output stream
nối tới socket



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

Ví dụ: Java client (TCP), cont.

tạo
input stream
nối tới socket

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

```
sentence = inFromUser.readLine();
```

Gửi tới
server

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

đọc tin từ
server

```
modifiedSentence = inFromServer.readLine();
```

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

```
clientSocket.close();
```

```
}
```

```
}
```


Ví dụ: Java server (TCP)

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

```
class TCPServer {
```

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

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

Tạo socket
chờ tại port 6789

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

chờ yêu cầu
từ client

```
        while(true) {
```

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

tạo input stream,
nối tới socket

```
            BufferedReader inFromClient =
```

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

Ví dụ: Java server (TCP), cont

tạo output
stream
nối tới socket

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

đọc thông tin
từ socket

```
clientSentence = inFromClient.readLine();
```

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

ghi ra socket

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

```
}  
}  
}
```

kết thúc while loop
và chờ yêu cầu từ client khác

Chương 2: Tầng ứng dụng

- ❑ 2.1 Nguyên lý của các ứng dụng mạng
- ❑ 2.2 Web và HTTP
- ❑ 2.3 FTP – File Transfer Protocol
- ❑ 2.4 Electronic Mail
 - ❖ SMTP, POP3, IMAP
- ❑ 2.5 DNS
- ❑ 2.6. Lập trình socket với TCP
- ❑ 2.7. Lập trình socket với UDP

Lập trình socket với *UDP*

UDP: không có liên kết giữa client và

- ❑ không có giai đoạn bắt tay
- ❑ bên gửi chỉ rõ IP address và port number của phía nhận trên mỗi gói tin
- ❑ server sẽ tìm địa chỉ IP và số hiệu cổng tương ứng bên trong gói tin

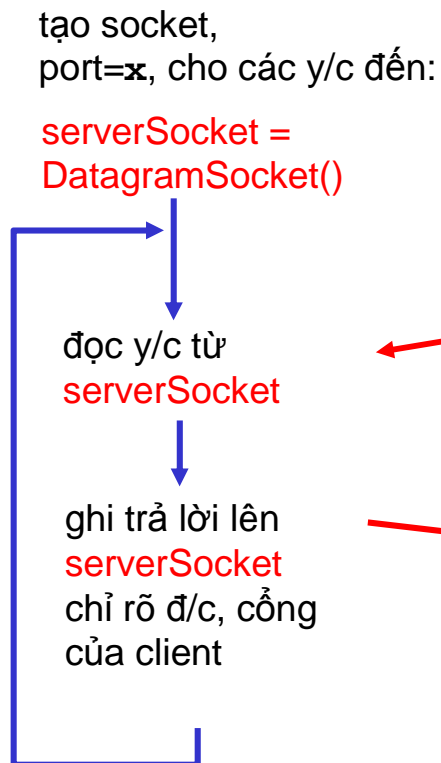
UDP: các gói tin có thể bị mất hoặc đến không theo thứ tự

Đối với ứng dụng

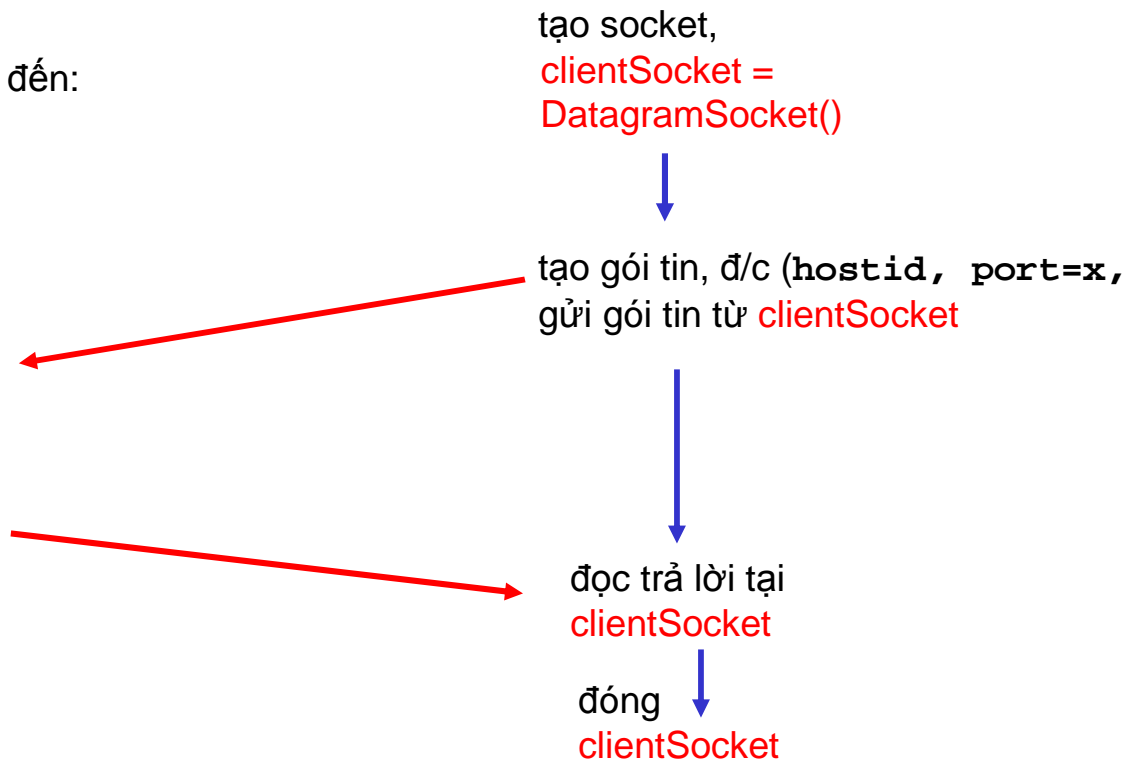
UDP cung cấp dịch vụ truyền dữ liệu không tin cậy giữa client và server

Tương tác client/server qua UDP socket

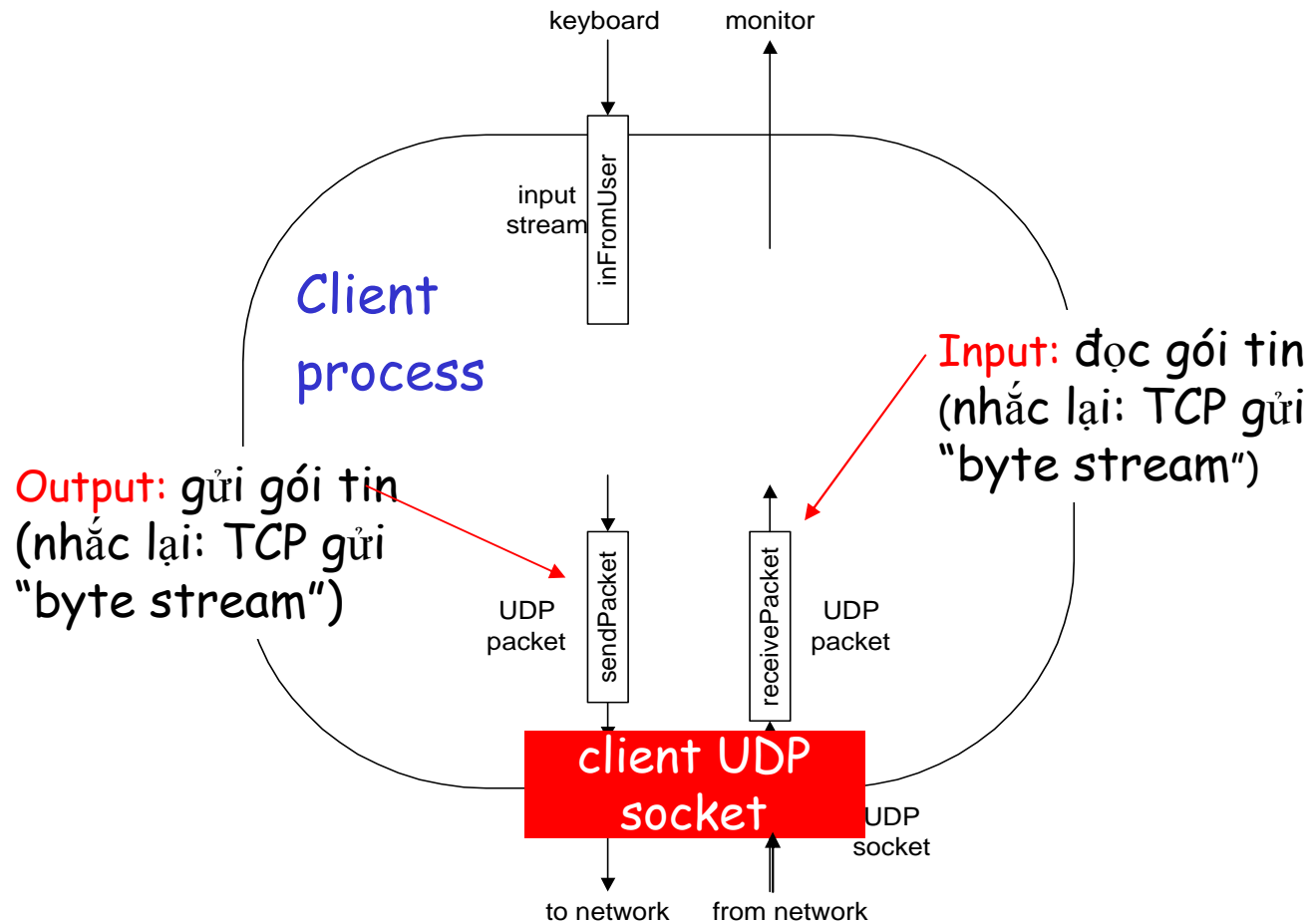
Server (máy `hostid`)



Client



Ví dụ: Java client (UDP)



Ví dụ: Java client (UDP)

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

class UDPClient {
    public static void main(String args[]) throws Exception
    {
        Tạo input stream → BufferedReader inFromUser =
            new BufferedReader(new InputStreamReader(System.in));
        Tạo client socket → DatagramSocket clientSocket = new DatagramSocket();
        chuyển đổi hostname -> 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: Java client (UDP), cont.

```
Tạo datagram với  
data-to-send,  
length, IP addr, port] → DatagramPacket sendPacket =  
                           new DatagramPacket(sendData, sendData.length, IPAddress, 9876);  
  
Gửi datagram  
tới server] → clientSocket.send(sendPacket);  
  
DatagramPacket receivePacket =  
               new DatagramPacket(receiveData, receiveData.length);  
  
đọc datagram  
từ server] → clientSocket.receive(receivePacket);  
  
String modifiedSentence =  
           new String(receivePacket.getData());  
  
System.out.println("FROM SERVER:" + modifiedSentence);  
clientSocket.close();  
}  
}
```


Ví dụ: Java server (UDP)

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

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

Tạo
datagram socket
tại cổng 9876

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

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

```
        while(true)  
        {
```

Tạo vùng đệm
nhận datagram

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

Nhận
datagram

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

Ví dụ: Java server (UDP), cont

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

Lấy IP addr
port # của người
gửi

```
InetAddress IPAddress = receivePacket.getAddress();  
int port = receivePacket.getPort();
```

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

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

Tạo datagram
để gửi tới client

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

Ghi
datagram
ra socket

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

Kết thúc vòng while,
chờ datagram khác