BÀI GIẢNG MÔN HỌC

LẬP TRÌNH MẠNG

Chương 6 :
LẬP TRÌNH SOCKET VỚI UDP

1. Tổng quan về giao thức UDP
    - 1.1. Hoạt động của giao thức UDP
    - 1.2. Các ưu và nhược điểm của UDP
    - 1.3. Khi nào thì sử dụng UDP
2. Lớp DatagramPacket
    - 2.1.Các constructor để nhận datagram
    - 2.2.Constructor để gửi các datagram
3. Lớp DatagramSocket
4. Nhận các gói tin
5. Gửi các gói tin
6. Demo giao thức UDP

1. TỔNG QUAN VỀ UDP

+ TCP/IP không phải là một giao thức mà thực sự là một họ các giao thức, bao gồm các giao thức như : IP, TCP và UDP

+ UDP nằm ở tầng giao vận, phía trên giao thức IP. Tầng giao vận cung cấp khả năng truyền tin giữa các mạng thông qua các gateway.

+ Nó sử dụng các địa chỉ IP để gửi các gói tin trên Internet hoặc trên mạng thông qua các trình điều khiển thiết bị khác nhau.

+ Giao thức UDP là giao thức đơn giản, không liên kết và cung cấp dịch vụ trên tầng giao vận với tốc độ nhanh.

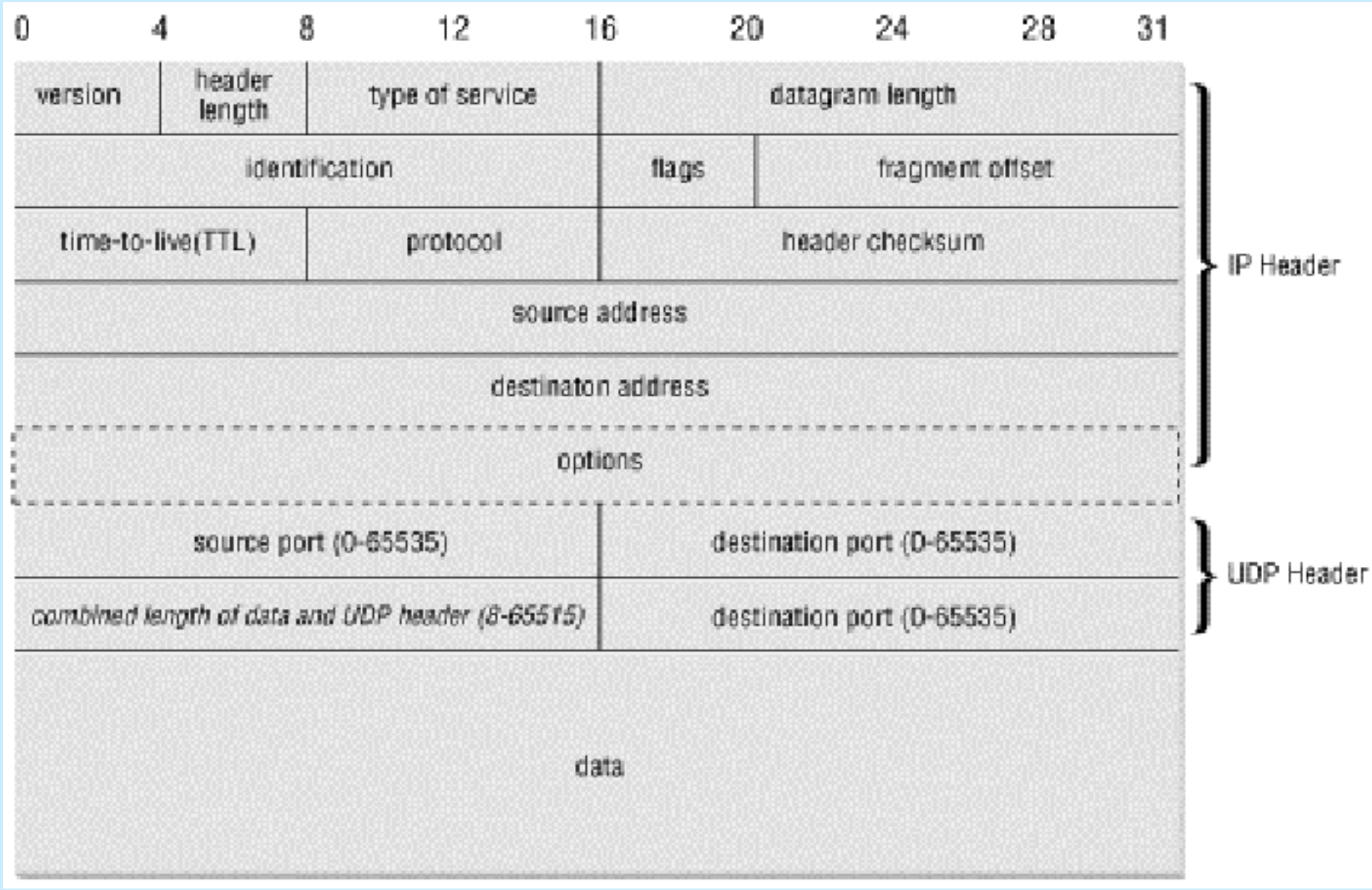
11. Hoạt động của giao thức UDP

+ Khi một ứng dụng dựa trên giao thức UDP gửi dữ liệu tới một host khác trên mạng, UDP thêm vào một header có độ dài 8 byte chứa :

- *số hiệu cổng nguồn và đích*
- *tổng chiều dài dữ liệu*
- *checksum*

+ IP thêm vào header của riêng nó vào đầu mỗi datagram UDP để tạo nên một datagram IP

11. Hoạt động của giao thức UDP



1.2. Các ưu và nhược điểm của UDP

* Ưu điểm:

+ Không cần thiết lập liên kết

+ Vì UDP không sử dụng các tín hiệu handshaking, nên có thể tránh được thời gian trễ.

- Đó chính là lý do tại sao DNS thường sử dụng giao thức UDP hơn là TCP

- DNS sẽ chậm hơn rất nhiều khi dùng TCP.

+ Tốc độ UDP nhanh hơn so với TCP.

+ UDP hỗ trợ các liên kết 1-1, 1-n, ngược lại TCP chỉ hỗ trợ liên kết 1-1.

+ Kích thước header : UDP chỉ có 8 byte header cho mỗi đoạn, ngược lại TCP cần các header 20 byte, vì vậy sử dụng băng thông ít hơn.

1.2. Các ưu và nhược điểm của UDP

* Nhược điểm:

+ UDP không đảm bảo việc dữ liệu đã đến đích hay chưa.

- Do UDP không gửi các tín hiệu bắt tay giữa bên gửi và bên nhận.

- Vì thế phía gửi không có cách nào để biết datagram đã đến đích hay chưa.

+ UDP không hỗ trợ bất kỳ phiên nào. Trong khi đó TCP sử dụng các chỉ số phiên (session ID) để duy trì các liên kết giữa hai host.

1.2. Các ưu và nhược điểm của UDP

* Nhược điểm:

+ UDP không đảm bảo rằng chỉ có một bản sao dữ liệu tới đích.

- Để gửi dữ liệu tới các hệ thống cuối, UDP phân chia dữ liệu thành các đoạn nhỏ.

- UDP không đảm bảo rằng các đoạn này sẽ đến đích đúng thứ tự như chúng đã được tạo ra ở nguồn.

+ TCP sử dụng các số thứ tự cùng với số hiệu cổng và các gói tin xác thực thường xuyên

- Điều này đảm bảo rằng các gói tin đến đích đúng thứ tự mà nó đã được tạo ra.

1.2. Các ưu và nhược điểm của UDP

* Nhược điểm:

+ TCP có tính bảo mật cao hơn UDP.

- Trong nhiều tổ chức firewall và router cần các gói tin UDP

- Vì các hacker thường sử dụng các cổng UDP

+ UDP không có kiểm soát luồng

- Một ứng dụng UDP được thiết kế tồi có thể làm giảm băng thông của mạng.

1.2. Các ưu và nhược điểm của UDP

Các đặc trưng	TCP	UDP
Hướng liên kết	*Có*	*Không*
Thiết lập phiên	*Có*	*Không*
Độ tin cậy	*Có*	*Không*
Xác thực	*Có*	*Không*
Đánh thứ tự	*Có*	*Không*
Điều khiển luồng	*Có*	*Không*
Bảo mật	*Nhiều hơn*	*Ít*

1.3 Khi nào sử dụng UDP

Rất nhiều ứng dụng trên Internet sử dụng UDP. Dựa trên các ưu và nhược điểm của UDP chúng ta có thể kết luận UDP có ích khi:

+ Sử dụng cho các phương thức truyền broadcasting và multicasting khi chúng ta muốn truyền tin với nhiều host.

+ Kích thước datagram nhỏ

+ Không cần thiết lập liên kết

+ Không cần truyền lại các gói tin

+ Ứng dụng không gửi các dữ liệu quan trọng

+ Băng thông của mạng đóng vai trò quan trọng

2. DatagramPacket

2.1. Constructor để nhận datagram
2.2. Constructor để gửi datagram

2. DatagramPacket

+ Việc cài đặt ứng dụng UDP trong Java cần có hai lớp là DatagramPacket và DatagramSocket.

+ DatagramPacket đóng gói các byte dữ liệu vào các gói tin UDP được gọi là datagram và cho phép ta mở các datagram khi nhận được.

+ Một DatagramSocket đồng thời thực hiện cả hai nhiệm vụ nhận và gửi gói tin.

- Để gửi dữ liệu, ta đặt dữ liệu trong một DatagramPacket và gửi gói tin bằng cách sử dụng DatagramSocket.

- Để nhận dữ liệu, ta nhận một đối tượng DatagramPacket từ DatagramSocket và sau đó đọc nội dung của gói tin.

2. DatagramPacket

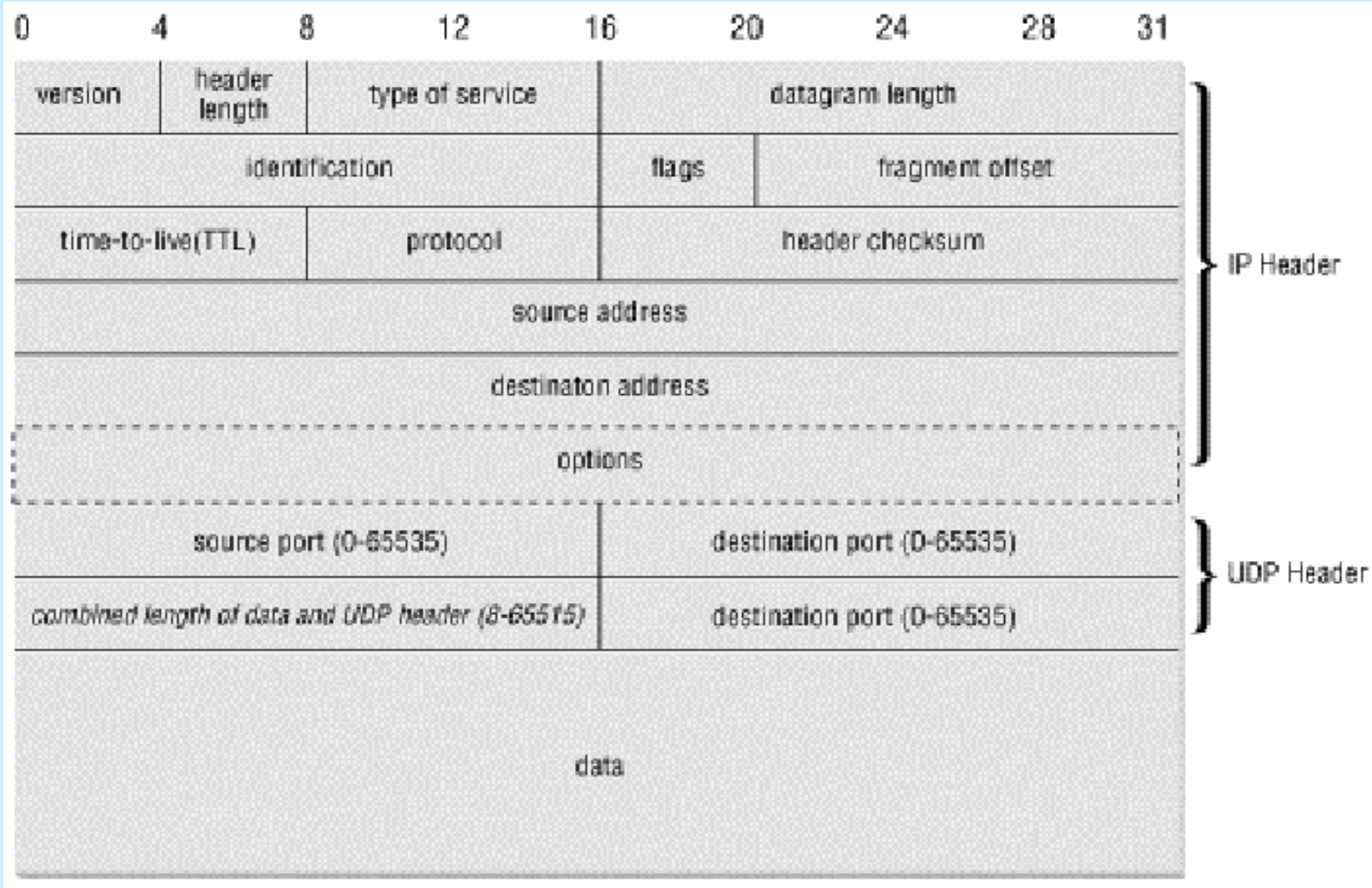
+ Các datagram UDP đưa rất ít thông tin vào datagram IP.

+ Header UDP chỉ đưa 8 byte vào header IP.

- Bao gồm số hiệu cổng nguồn và đích
- Chiều dài của dữ liệu và header UDP
- Tiếp đến là một checksum tùy chọn.

+ Vì mỗi cổng được biểu diễn bằng hai byte nên tổng số cổng UDP trên một host sẽ là 65535.

2. DatagramPacket



2.1. Constructor để nhận datagram

+ Hai constructor tạo ra các đối tượng DatagramPacket mới để nhận dữ liệu từ mạng:

public DatagramPacket(byte[] b, int length)

public DatagramPacket(byte[] b, int offset, int length)

- Khi một socket nhận một datagram, nó lưu trữ phần dữ liệu của datagram ở trong vùng đệm b cho tới khi gói tin được lưu trữ hoàn toàn hoặc cho tới khi lưu trữ hết length byte.

- Nếu sử dụng constructor thứ hai, thì dữ liệu được lưu trữ bắt đầu từ vị trí b[offset].

2.1. Constructor để nhận datagram

Ví dụ : Xây dựng một DatagramPacket để nhận dữ liệu có kích thước lên tới 2579 byte

```
byte b[]=new byte[2579];
DatagramPacket dp=new DatagramPacket(b,b.length);
```

2.2. Constructor để gửi datagram

+ Bốn constructor tạo các đối tượng DatagramPacket mới để gửi dữ liệu trên mạng:

- public DatagramPacket(byte[] b, int length, InetAddress dc, int port)
- public DatagramPacket(byte[] b, int offset, int length, InetAddress dc, int port)
- public DatagramPacket(byte[] b, int length, SocketAddress dc, int port)
- public DatagramPacket(byte[] b, int offset, int length, SocketAddress dc, int port)

+ Mỗi constructor tạo ra một DatagramPacket mới để được gửi đi tới một host khác.

+ Gói tin được điền đầy dữ liệu với chiều dài là length byte bắt đầu từ vị trí offset hoặc vị trí 0 nếu offset không được sử dụng

2.2. Constructor để gửi datagram

Ví dụ : Gửi đi một xâu ký tự đến một host như sau:

```
String st="Minh họa sử dụng UDP";
byte[] b= st.getBytes();
try{
  InetAddress
dc=InetAddress.getByName("www.ud.edu.vn");
   int port =17;
  DatagramPacket dp=new
DatagramPacket(b,b.length,dc,port);
// Gửi gói tin đi
}catch(IOException e){System.err.println(e);}
```

2.3. Các phương thức nhận các thông tin từ DatagramPacket

+ DatagramPacket có 6 phương thức để tìm các phần khác nhau của một datagram

public InetAddress getAddress()

- Phương thức getAddress() trả về một đối tượng InetAddress chứa địa chỉ IP của host ở xa.

- Nếu datagram được nhận từ Internet, địa chỉ trả về chính là địa chỉ của máy đã gửi datagram (địa chỉ nguồn).

- Mặt khác nếu datagram được tạo cục bộ để được gửi tới máy ở xa, phương thức này trả về địa chỉ của host mà datagram được đánh địa chỉ.

2.3. Các phương thức nhận các thông tin từ DatagramPacket

public int getPort()

- Phương thức getPort() trả về một số nguyên xác định cổng trên host ở xa.

- Nếu datagram được nhận từ Internet thì cổng này là cổng trên host đã gửi gói tin đi.

public SocketAddress()

- Phương thức này trả về một đối tượng SocketAddress chứa địa chỉ IP và số hiệu cổng của host ở xa.

public byte[] getData()

- Phương thức getData() trả về một mảng byte chứa dữ liệu từ datagram. *Thông thường cần phải chuyển các byte này thành một dạng dữ liệu khác trước khi chương trình xử lý dữ liệu.*

2.3. Các phương thức nhận các thông tin từ DatagramPacket

Một cách để thực hiện điều này là chuyển đổi mảng byte thành một đối tượng String sử dụng constructor sau đây:

public String(byte[] buffer,String encoding)

- Tham số đầu tiên buffer là mảng các byte chứa dữ liệu từ datagram.

- Tham số thứ hai cho biết cách thức mã hóa xâu ký tự.

Cho trước một DatagramPacket dp được nhận từ mạng, ta có thể chuyển đổi nó thành xâu ký tự như sau:

String s=new String(dp.getData(),"ASCII");

3. DatagramSocket

+ Để gửi hoặc nhận một DatagramPacket, bạn phải mở một DatagramSocket.

+ Trong Java, một DatagramSocket được tạo ra và được truy xuất thông qua đối tượng DatagramSocket

public class DatagramSocket extends Object

+ Tất cả các datagram được gắn với một cổng cục bộ, cổng này được sử dụng để lắng nghe các datagram đến hoặc được đặt trên các header của các datagram sẽ gửi đi.

3. DatagramSocket

+ DatagramSocket được sử dụng để gửi và nhận các gói tin UDP

- Nó cung cấp các phương thức để gửi và nhận các gói tin

- Cũng như xác định một giá trị timeout khi sử dụng phương pháp vào ra không phong tỏa (non blocking I/O)

- Kiểm tra và sửa đổi kích thước tối đa của gói tin UDP, đóng socket.

3. DatagramSocket

Các phương thức :

void close() : Đóng một liên kết và giải phóng khỏi cổng cục bộ.

void connect(InetAddress remote_address, int remote_port) : phương thức này trả về địa chỉ remote mà socket kết nối tới, hoặc giá trị null nếu không tồn tại liên kết.

InetAddress getLocalAddress() : Trả về địa chỉ cục bộ

Int getSoTimeOut() : Trả về giá trị tùy chọn timeout của socket. Giá trị này xác định thời gian mà thao tác đọc sẽ phong tỏa trước khi nó đưa ra ngoại lệ InterruptedException. *Ở chế độ mặc định, giá trị này bằng 0, chỉ ra rằng vào ra không phong tỏa được sử dụng.*

3. DatagramSocket

Các phương thức :

void receive(DatagramPacket dp) throws IOException

Phương thức đọc một gói tin UDP và lưu nội dung trong packet xác định

void send(DatagramPacket dp) throws IOException

Phương thức gửi một gói tin

void setSoTimeOut(int timeout)

Thiết lập giá trị tùy chọn của socket

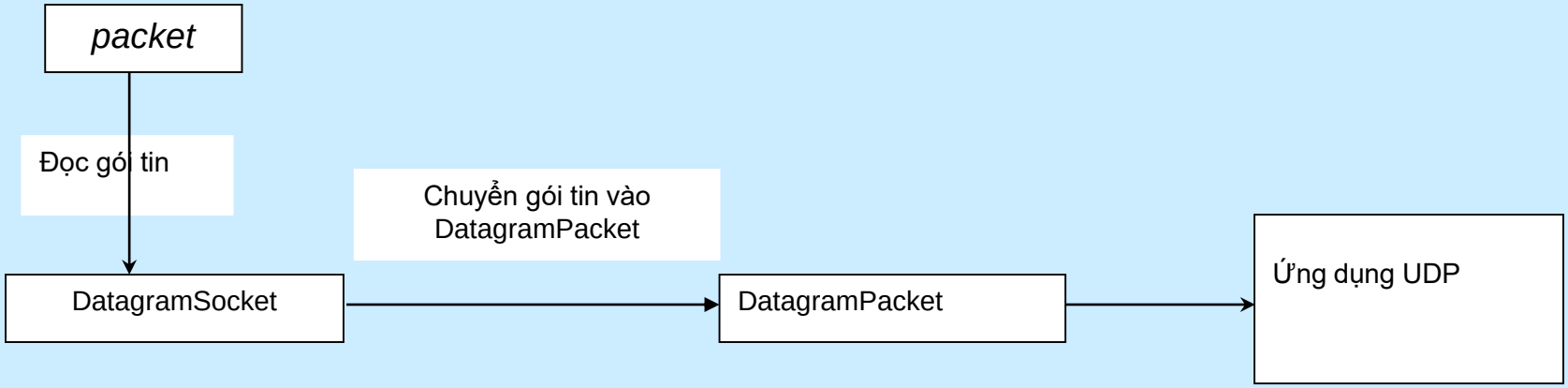
4. NHẬN CÁC GÓI TIN

+ Trước khi một ứng dụng có thể đọc các gói tin UDP được gửi bởi các máy ở xa

+ Nó phải gán một socket với một cổng UDP bằng cách sử dụng DatagramSocket

+ Ngoài ra khi tạo ra một DatagramPacket sẽ đóng vai trò như là một bộ chứa dữ liệu của gói tin UDP

4. NHẬN CÁC GÓI TIN



+ Khi một ứng dụng muốn đọc các gói tin UDP, nó gọi phương thức DatagramSocket.receive()

+ Phương thức này sao chép gói tin UDP vào một DatagramPacket xác định.

+ Xử lý nội dung gói tin và tiến trình lặp lại khi cần

4. NHẬN CÁC GÓI TIN

```
DatagramPacket dp=new DatagramPacket(new byte[256],256);
DatagramSocket ds=new DatagramSocket(2012);
boolean finished=false;
while(!finished)
{
ds.receive(dp);
//Xử lý gói tin
}
ds.close();
```

4. NHẬN CÁC GÓI TIN

+ Khi xử lý gói tin ứng dụng phải làm việc trực tiếp với một mảng byte.

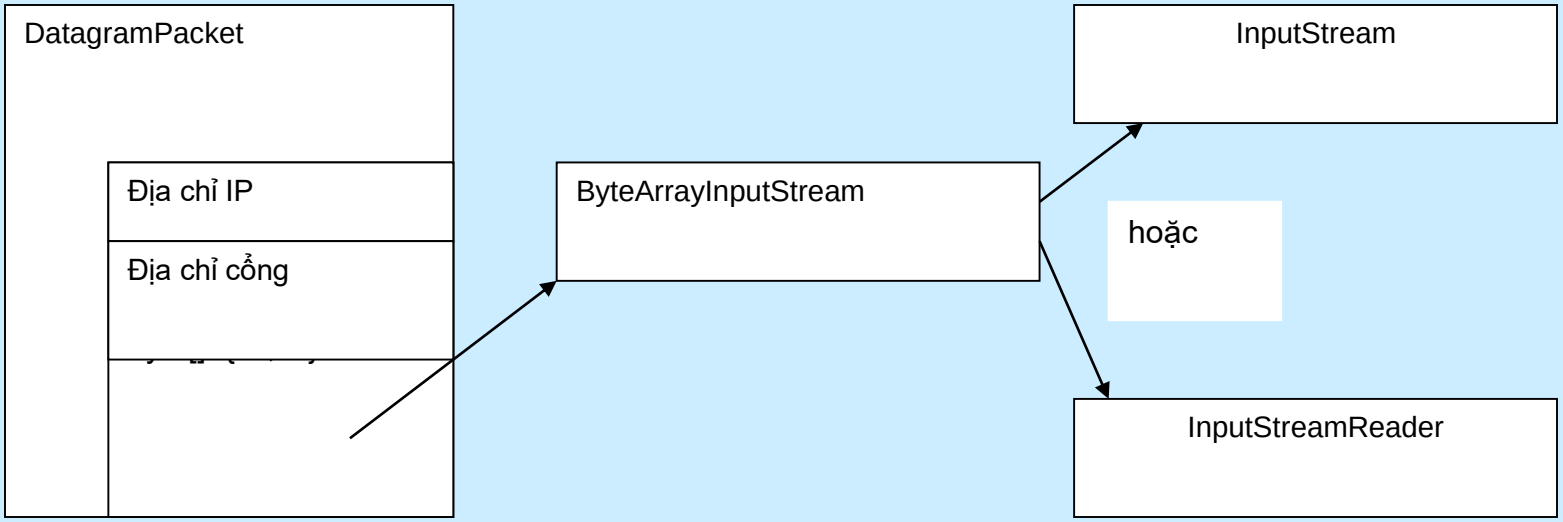
+ Nếu ứng dụng là đọc văn bản thì ta có thể sử dụng các lớp từ gói vào ra để chuyển đổi giữa mảng byte cùng với luồng stream và luồng reader.

-Bằng cách gắn kết luồng nhập ByteArrayInputStream với nội dung của một datagram và sau đó kết nối với một kiểu luồng khác, khi đó bạn có thể truy xuất tới nội dung của gói UDP một cách dễ dàng.

+ Ngoài ra ta có thể dùng các luồng vào ra I/O để xử lý dữ liệu

- Bằng cách sử dụng luồng DataInputStream hoặc BufferedReader để truy xuất tới nội dung của các mảng byte.

4. NHẬN CÁC GÓI TIN



Ví dụ : Để gắn kết một luồng DataInputStream với nội dung của một DatagramPacket, ta sử dụng đoạn mã sau:

ByteArrayInputStream bis=new ByteArrayInputStream(dp.getData());
DataInputStream dis=new DataInputStream(bis);
//đọc nội dung của gói tin UDP

5. GỬI CÁC GÓI TIN

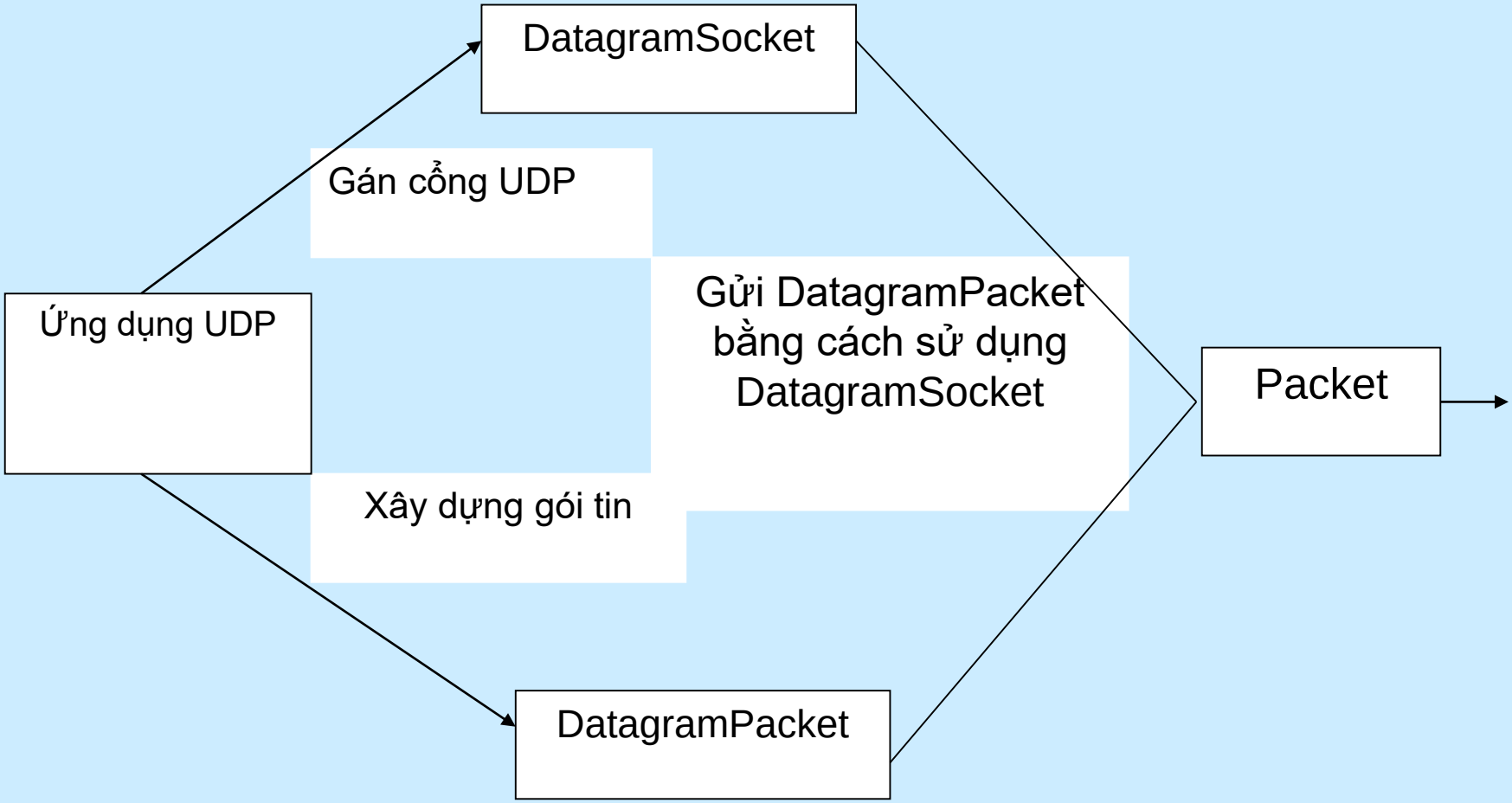
+ Lớp DatagramSocket cũng được sử dụng để gửi các gói tin.

- Khi gửi gói tin, ứng dụng phải tạo ra một DatagramPacket, thiết lập địa chỉ và thông tin cổng, và ghi dữ liệu cần truyền vào mảng byte.

+ Nếu muốn gửi thông tin phúc đáp thì ta cũng đã biết địa chỉ và số hiệu cổng của gói tin nhận được.

-Mỗi khi gói tin sẵn sàng để gửi, ta sử dụng phương thức send() của lớp DatagramSocket để gửi gói tin đi.

5. GỬI CÁC GÓI TIN



5. GỬI CÁC GÓI TIN

```
//Socket lắng nghe các gói tin đến trên cổng 2013
DatagramSocket socket = new DatagramSocket(2013);
DatagramPacket packet = new DatagramPacket (new byte[256], 256);
packet.setAddress ( InetAddress.getByName ( somehost ) );
packet.setPort ( 2013 );
boolean finished = false;
while !finished )
{
// Ghi dữ liệu vào vùng đệm buffer
.....
socket.send (packet);
// Thực hiện hành động nào đó, chẳng hạn như đọc gói tin khác hoặc kiểm tra xem
// còn gói tin nào cần gửi đi hay không
.....
}
socket.close();
```

6. DEMO GIAO THỨC UDP

```
// Chương trình Client
import java.net.*;
import java.io.*;
public class UDPClient1
{
        public final static int DEF_PORT=1234;
        public static void main(String args[]) {
                String hostname;
                int port=DEF_PORT;
                if(args.length>0)
                {       hostname=args[0];
                        try{} catch(Exception e){
                                port =Integer.parseInt(args[1]);
                        }
                }
                else {
                        hostname="127.0.0.1";
                }
```

6. DEMO GIAO THỨC UDP

```
// Chương trình Client
                                try{
                                        InetAddress
dc=InetAddress.getByName(hostname);
                                        BufferedReader userInput=new
BufferedReader(new InputStreamReader(System.in));
                                        DatagramSocket ds =new
DatagramSocket(port);
                                        while(true){
                                                String line=userInput.readLine();
                                                if(line.equals("exit"))break;
                                                byte[] data=line.getBytes();
                                                DatagramPacket dp=new
DatagramPacket(data,data.length,dc,port);
                                                ds.send(dp);
                                                dp.setLength(50505);
```

6. DEMO GIAO THỨC UDP

```
// Chương trình Client
        ds.receive(dp);
        ByteArrayInputStream bis =new
ByteArrayInputStream(dp.getData());
        BufferedReader dis =new BufferedReader(new
InputStreamReader(bis));
        System.out.println(dis.readLine());
        }
        }catch(UnknownHostException
e){System.err.println(e);}
        catch(IOException e){System.err.println(e);}}
}
```

6. DEMO GIAO THỨC UDP

```
// Chương trình Server
import java.net.*;
import java.io.*;
public class UDPServer1
{
        public final static int DEF_PORT=1234;
    // phuong thuc doi sang chuoi vua hoa vua thuong
    private static String VuaHoaVuaThuong(String s3)    {
    int k =0, i;
    char c;
    String st3="";
    k = s3.length();
    for(i=0;i<k;i++)
    {
            c = s3.charAt(i);
            if(c>='A'&& c<='Z')  c = (char) (c + 32);
            else if (c>='a'&& c<='z') c=(char) (c-32);
            // Noi vao chuoi moi st3
                    st3=st3+c;
    }
    return st3;
  }
```

6. DEMO GIAO THỨC UDP

// Chương trình Server

```
public static void main(String args[])
{
        int port=DEF_PORT;
        try{
                }
                catch(Exception e){
                port =Integer.parseInt(args[1]);
                }
        try{
        DatagramSocket ds =new DatagramSocket(port);
        DatagramPacket dp=new DatagramPacket(new
byte[50505],50505);
        while(true){
                ds.receive(dp);
                ByteArrayInputStream bis =new
ByteArrayInputStream(dp.getData());
                BufferedReader dis =new BufferedReader(new
InputStreamReader(bis));
```

6. DEMO GIAO THỨC UDP

```
// Chương trình Server
                                String s0=dis.readLine();
                                System.out.println(s0);
                                // s.toUpperCase();
                                 String s=VuaHoaVuaThuong(s0);
                                  dp.setData(s.getBytes());
                                dp.setLength(s.length());
                                dp.setAddress(dp.getAddress());
                                dp.setPort(dp.getPort());
                                ds.send(dp);
                                } }catch(UnknownHostException e)
                                {
                                System.err.println(e);
                                }
                                catch(IOException e)
                                {
                                System.err.println(e);
                                }
                        }
}
```

Hết *!!!*