



Computer Networks 1

(Mạng Máy Tính 1)

Lectured by: Dr. Phạm Trần Vũ

[cuu duong than cong . com](http://cuuduongthancong.com)



Lecture 9:

Socket Programming with Java

cuu duong than cong . com



Using InetAddress (1)

□ Get local address

```
import java.net.*;

public class HostInfo {
    public static void main(String args[]) {
        HostInfo host = new HostInfo();
        host.init();
    }

    public void init() {
        try {
            InetAddress myHost = InetAddress.getLocalHost();
            System.out.println(myHost.getHostAddress());
            System.out.println(myHost.getHostName());
        } catch (UnknownHostException ex) {
            System.err.println("Cannot find local host");
        }
    }
}
```



Using InetAddress (2)

□ In địa chỉ IP của proxy.hcmut.edu.vn

```
import java.net.*;
class kku{
    public static void main (String args[]) {
        try {
            InetAddress[] addresses =
                InetAddress.getAllByName("proxy.hcmut.edu.vn");
            for (int i = 0; i < addresses.length; i++) {
                System.out.println(addresses[i]);
            }
        }
        catch (UnknownHostException e) {
            System.out.println("Could not find
                                proxy.hcmut.edu.vn");
        }
    }
}
```



Using Socket (1)

□ Kết nối đến 1 số webserver

```
import java.net.*;
import java.io.*;
public class getSocketInfo {
    public static void main(String[] args) {
        for (int i = 0; i < args.length; i++) {
            try {
                Socket theSocket = new Socket(args[i], 80);
                System.out.println("Connected to " +
                    theSocket.getInetAddress() +
                    " on port " + theSocket.getPort() +
                    " from port " +
                    theSocket.getLocalPort() + " of " +
                    theSocket.getLocalAddress());
            }
        }
    }
}
```



Using Socket (2)

```
    } catch (UnknownHostException e) {  
        System.err.println("I can't find " + args[i]);  
    } catch (SocketException e) {  
        System.err.println("Could not connect to " +  
                               args[i]);  
    } catch (IOException e) {  
        System.err.println(e);  
    }  
    } // end for  
} // end main  
} // end getSocketInfo
```

cuu duong than cong . com



Using ServerSocket (1)

□ DateTime Server

```
import java.net.*;
import java.io.*;
import java.util.Date;
public class DayTimeServer {
    public final static int daytimePort = 5000;
    public static void main(String[] args) {
        ServerSocket theServer;
        Socket theConnection;
        PrintStream p;
        try {
            theServer = new
                ServerSocket(daytimePort);
```



Using ServerSocket (2)

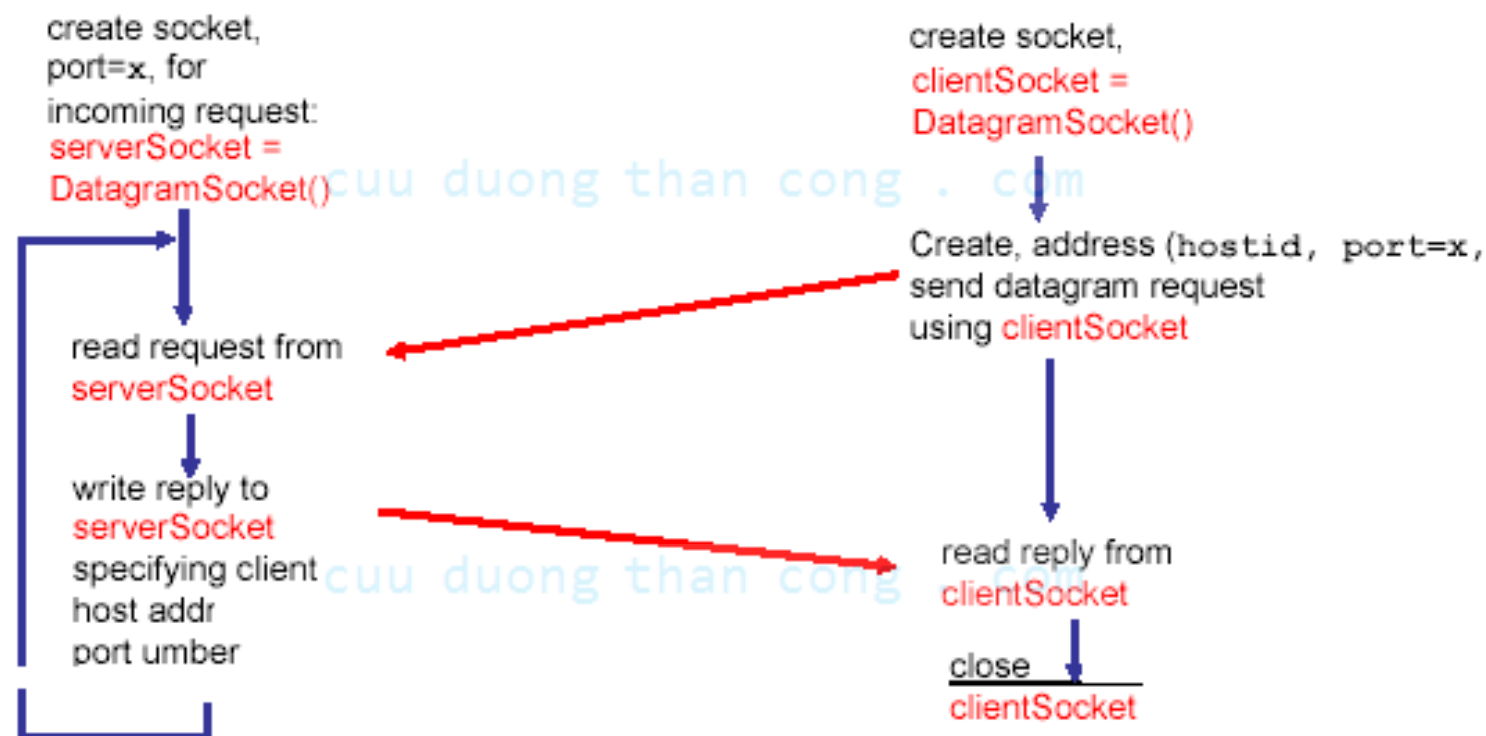
```
        while (true) {
            theConnection =
theServer.accept();
            p = new
PrintStream(theConnection.getOutputStream());
            p.println(new Date());
            theConnection.close();
        }
        theServer.close();
    } catch (IOException e) {
        System.err.println(e);
    }
}
```




Client-Server Application with UDP

Server (running on `hostid`)

Client





UDP Client (1)

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

class UDPClient {
    public static void main(String args[]) throws Exception
    {
        Create input stream → BufferedReader inFromUser =
                                new BufferedReader(new InputStreamReader(System.in));
        Create client socket → DatagramSocket clientSocket = new DatagramSocket();
        Translate hostname to IP address using DNS → InetAddress IPAddress = InetAddress.getByName("hostname");

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

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



UDP Client (2)

```
Create datagram with  
data-to-send, length, IP addr, port } DatagramPacket sendPacket =  
                                     } new DatagramPacket(sendData, sendData.length, IPAddress, 9876);  
  
Send datagram to server } clientSocket.send(sendPacket);  
  
                           DatagramPacket receivePacket =  
                           new DatagramPacket(receiveData, receiveData.length);  
  
Read datagram from server } clientSocket.receive(receivePacket);  
  
                           String modifiedSentence =  
                           new String(receivePacket.getData());  
  
                           System.out.println("FROM SERVER:" + modifiedSentence);  
                           clientSocket.close();  
                           }  
                           }
```



UDP Server (1)

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

class UDPServer {
    public static void main(String args[]) throws Exception
    {
        Create datagram socket at port 9876 → DatagramSocket serverSocket = new DatagramSocket(9876);

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

        while(true)
        {
            Create space for received datagram → DatagramPacket receivePacket =
                new DatagramPacket(receiveData, receiveData.length);

            Receive datagram → serverSocket.receive(receivePacket);
```



UDP Server (2)

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

Get IP addr } InetAddress IPAddress = receivePacket.getAddress();
port #, of  }
sender      } int port = receivePacket.getPort();

String capitalizedSentence = sentence.toUpperCase();

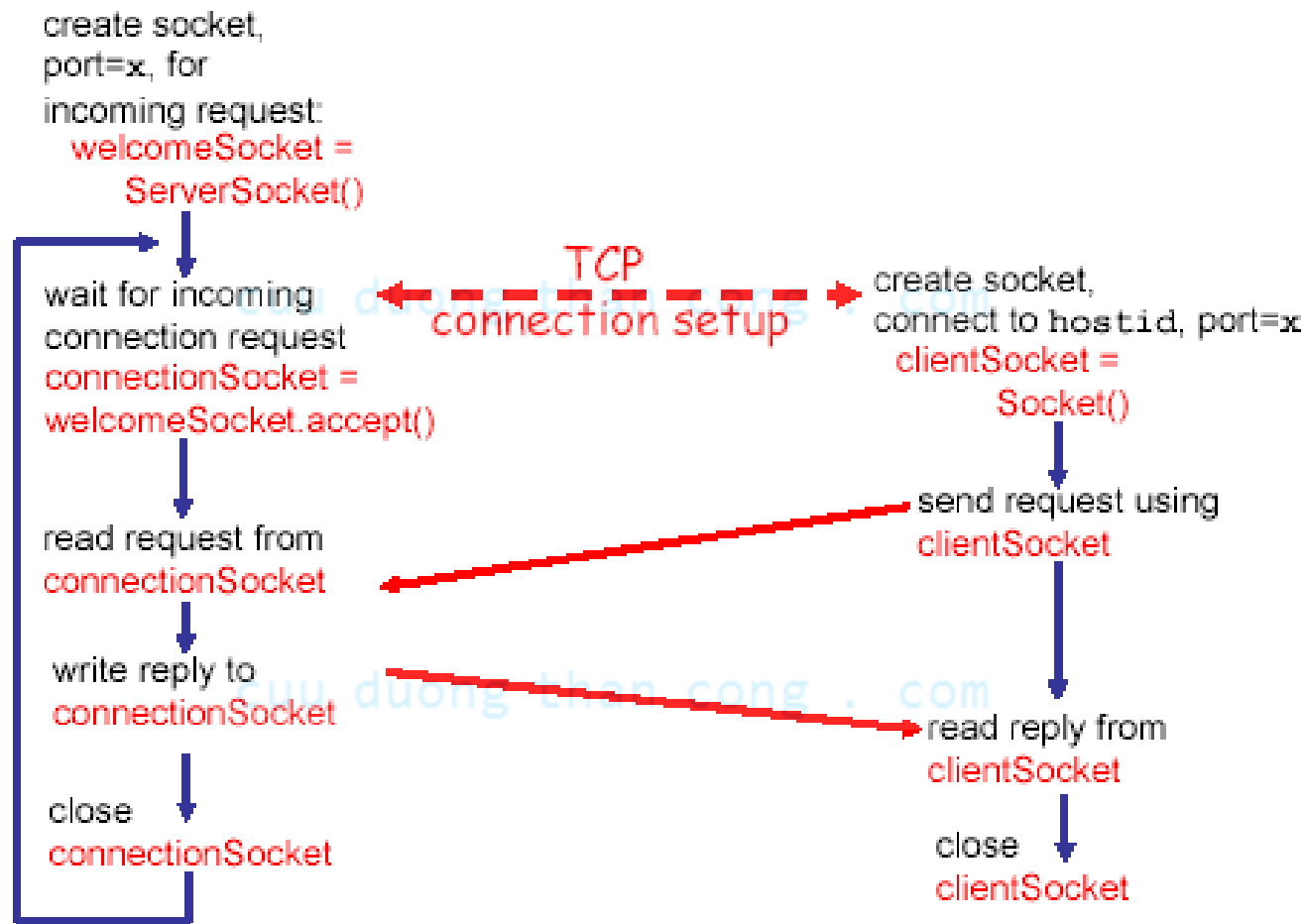
Create datagram } sendData = capitalizedSentence.getBytes();
to send to client } DatagramPacket sendPacket =
                  } new DatagramPacket(sendData, sendData.length, IPAddress,
                  } port);

Write out } serverSocket.send(sendPacket);
datagram  }
to socket }
          }
          }
          }
```

End of while loop,
loop back and wait for
another datagram



Client-Server Application with TCP (4)





TCP Client (1)

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

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

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

```
        Create input stream → BufferedReader inFromUser =
                                new BufferedReader(new InputStreamReader(System.in));

        Create client socket, connect to server → Socket clientSocket = new Socket("hostname", 6789);

        Create output stream attached to socket → DataOutputStream outToServer =
                                                    new DataOutputStream(clientSocket.getOutputStream());
```



TCP Client (2)

```
        Create  
        input stream  
        attached to socket }  
        BufferedReader inFromServer =  
            new BufferedReader(new  
                InputStreamReader(clientSocket.getInputStream()));  
        sentence = inFromUser.readLine();  
  
        Send line  
        to server }  
        outToServer.writeBytes(sentence + '\n');  
  
        Read line  
        from server }  
        modifiedSentence = inFromServer.readLine();  
        System.out.println("FROM SERVER: " + modifiedSentence);  
  
        clientSocket.close();  
    }  
}
```




TCP Server (1)

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

class TCPServer {

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

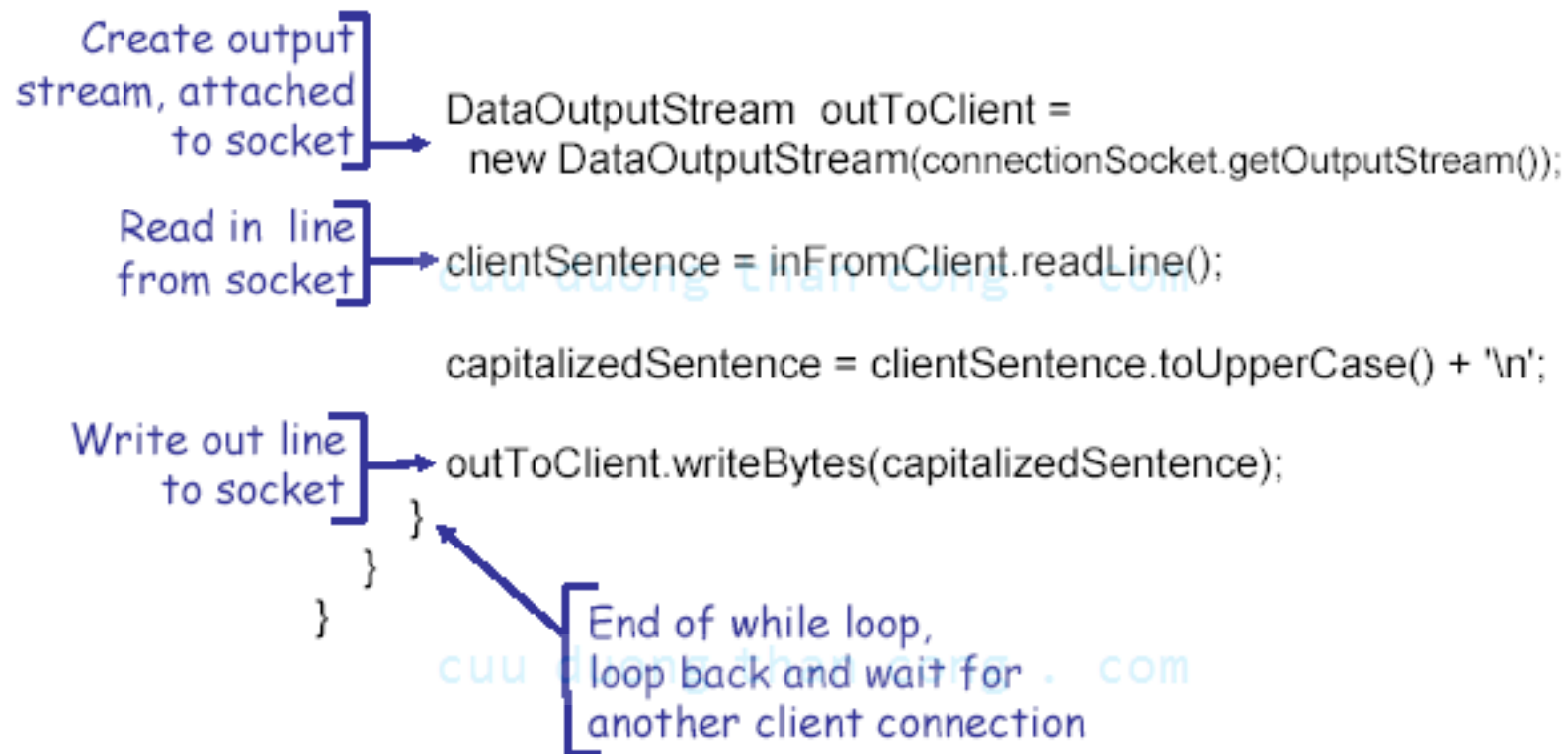
        Create welcoming socket at port 6789 → ServerSocket welcomeSocket = new ServerSocket(6789);

        while(true) {
            Wait, on welcoming socket for contact by client → Socket connectionSocket = welcomeSocket.accept();

            Create input stream, attached to socket → BufferedReader inFromClient =
                new BufferedReader(new
                InputStreamReader(connectionSocket.getInputStream()));
        }
    }
}
```



TCP Server (2)





Java Multi-Threading

```
1.  class PrimeRun implements Runnable {
2.      long minPrime;
3.      PrimeRun ( long minPrime ) {
4.          this.minPrime = minPrime;
5.      }
6.      public void run() {
7.          // compute primes larger than minPrime
8.          . . .
9.      }
10. }
11. PrimeRun p = new PrimeRun(143);
12. new Thread(p).start();
```



Stop a Thread (1)

- Using Thread.interrupt(), after changing loop condition
- **This method does not work with ServerSocket.accept()!**

```
public void stop() {  
    running = false;  
    this.interrupt();  
}  
  
public void run() {  
    running = true;  
    while (running){  
        Socket s = ssocket.accept();  
        ...  
    }  
}
```



Stop a Thread (2)

❑ Close ServerSocket

```
public void stop() {  
    running = false;  
    ssocket.close();  
}  
public void run() {  
    running = true;  
    while (running){  
        Socket s = ssocket.accept();  
        ...  
    }  
}
```



Server Side Thread

Main window thread

- Main Server Form
- Create a thread to listen and accept incoming connection

Child thread

- Listen and accept incoming connection
- Create a thread to handle connection

Grand-child thread

- Handle a connection
- Create a thread to handle socket InputStream

Grand-grand-child

- Handle socket InputStream



Client Side Thread

Main window thread

- Main Client Form
- Create a window for new connection

Chat window thread

- Chat window
- Connect to a server socket
- Create a thread to handle socket InputStream

Grand-child thread

- Handle socket InputStream