# LAB 8. LẬP TRÌNH CLIENT-SERVER SỬ DỤNG TCP [4,6,9]

## A. MỤC TIÊU

Trang bị cho sinh viên các kỹ thuật lập trình hướng kết nối:

**-** Sử dụng lớp Socket xây dựng ứng dụng kết nối gửi nhận dữ liệu phía Client theo cơ chế TCP.

**-** Sử dụng Server Socket, xây dựng ứng dụng Client-Server theo cơ chế TCP.

**B. NỘI DUNG**

**-** Viết các ứng dụng phía Client sử dụng **class** Socket.

**-** Viết các ứng dụng phía Server sử dụng **class** Server Socket.

**C. KẾT QUẢ SAU KHI HOÀN THÀNH**

* Hiểu được nguyên lý lập trình mô phỏng các giao thức mạng.
* Viết được chương trình kết nối TCP thực hiện trao đổi dữ liệu giữa Client và Server.
* Xây dựng ứng dụng mạng theo mô hình 3 lớp.

**D. YÊU CẦU PHẦN CỨNG, PHẦN MỀM**

**-** Máy tính cài HĐH windows, RAM tối thiểu 1GB, có kết nối Internet.

**-** Phần mềm NETBEAN 8.0, JDK 1.8.

**E. HƯỚNG DẪN**

1. **Giao thức hướng kết nối- TCP**: sử dụng kết nối (ảo) để truyền thông.

**Đặc điểm:** Truyền thông theo kiểu điểm-điểm

Quá trình truyền thông được thực hiện qua 3 giai đoạn:

* Thiết lập kết nối
* Truyền dữ liệu kèm theo cơ chế kiểm soát chặt chẽ
* Huỷ bỏ kết nối

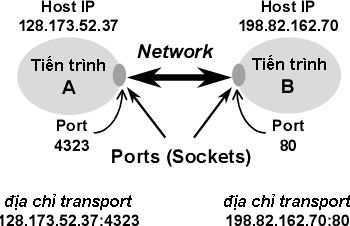
2. Giao diện socket, địa chỉ socket

Socket là giao diện và đóng vai trò như một điểm cuối (end point) để truyền thông. Mỗi tiến trình khi muốn truyền thông bằng socket, đầu tiên phải tạo một socket và được gán một định danh duy nhất được gọi là địa chỉ socket.

**Giao diện socket chia làm các loại sau:**

* Stream socket: Cho phép truyền thông với giao thức TCP. TCP sử dụng một cặp stream socket để kết nối ứng dụng này với ứng dụng khác qua Internet.
* Datagram socket: Cho phép truyền thông với giao thức UDP. UDP sử dụng một cặp datagram socket để kết nối từ một ứng dụng này tới một ứng dụng khác qua Internet

Một địa chỉ socket là một tổ hợp gồm 2 địa chỉ: IP và cổng (port)- xác định một đầu mút cuối truyền thông. Nó chỉ ra tiến trình nào (port) chạy trên trên máy nào (IP).

**

*Hình 8. Địa chỉ socket*

**3. Một số lớp lập trình với TCP**

a) Socket

b) ServerSocket

**3.1.** **Viết các ứng dụng kết nối phía Client sử dụng lớp Socket**

Tạo đối tượng Socket phía Client.

**Các phương thức tạo:**

**public** Socket (**String** host, **int** port) **throws** UnknownHostException, IOException

**public** Socket (InetAddress address, **int** port) **throws** IOException

**public** Socket (**String** host, **int** port, InetAddress localaddr, **int** localPort)

**Các phương thức:**

**public** InputStream getInputStream(): Trả về luồng nhập của socket.

**public** OutputStream getOutputStream(): Trả về luồng xuất của socket.

**public** **void** close() **throws** IOException: Đóng socket

**Bài 1.** Viết ứng dụng phía Client truy xuất đến một trang web sử dụng **class** Socket

**Hướng dẫn :**

|  |
| --- |
| **import** java.io.\*;  **import** java.net.Socket;  **public class** ConnectHttpUsingSocket {  **public** **static** **void** main(**String**[] args) **throws** Exception{  //Kết nối  Socket socket=**new** Socket("localhost", 80);  //Gửi yêu cầu  OutputStream os = socket.getOutputStream();  PrintWriter out=**new** PrintWriter(os,**true**);  out.println("GET / HTTP/1.0");  out.println();  out.flush();  //Nhận dữ liệu  InputStream is = socket.getInputStream();  **int** len=0;  byte []buffer=**new** byte[4086];  **while**((len=is.read(buffer))!=-1)  {  **String** data=**new** **String**(buffer,0,len);  System*.out.*println(data);  }  socket.close();  }  } |

**Bài 2.**

Viết ứng dụng phía Client trả về chuỗi thời gian của một máy chủ đã cho, sử dụng dịch vụ ngày giờ chuẩn có sẵn trong cổng TCP 13. Các đối số là tên máy chủ và số cổng (số cổng mặc định là 13). Nếu chương trình chạy không có đối số sẽ in ngày giờ của localhost.

**Hướng dẫn:**

1. Xử lý đối số
2. Tạo kết nối mạng
3. Tạo luồng dữ liệu
4. Đọc chuỗi thời gian và in ra dạng chuẩn
5. Đóng kết nối

**Tạo class DaytimeClient**

|  |
| --- |
| **import** java.net.\*;  **import** java.io.\*;  **public class** DaytimeClient {  **static** **final** **int** defaultPort = 13;  **public** **static** **void** main(**String**[] args) {  **int** portNumber;  Socket ClientSocket;  BufferedReader timeStream;  **String** hostName;  **switch**(args.length) {  **case** 1: hostName = args[0];  portNumber = defaultPort;//daytimePort;  **break**;  **case** 2: hostName = args[0];  portNumber = **new** **Integer**(args[1]).intValue());  **break**;  **default**:  hostName = "localhost";  portNumber = defaultPort;  }  **try** {  // Thực hiện kết nối tới Server  ClientSocket = **new** Socket(hostName,portNumber);  // Tạo luồng dữ liệu từ kết nối  timeStream = **new** BufferedReader(**new** InputStreamReader(ClientSocket.getInputStream()));  // Lấy dữ liệu từ Server  **String** timeString = timeStream.readLine();  System*.out.*println("It is " + timeString + " at " + hostName);  // Đóng kết nối  timeStream.close();  ClientSocket.close();  }  **catch** (UnknownHostException e) {  InterruptedException(" Unknown host error");  }  **catch** (ConnectException e) {  System*.out.*println(" Service unavailable on port "+portNumber+"of host "+hostName);  }  **catch** (IOException e) {  InterruptedException(" Communication error occured\r\n "+e);  }  }  } |

**Bài 3.**

Viết ứng dụng đọc vào một chuỗi từ máy khách kết nối với máy chủ qua cổng TCP xác định. Nếu gặp dấu ‘.’: đóng kết nối và dừng lại, ngược lại sẽ gửi chuỗi đến máy chủ. Sau khi gửi đi, máy khách nhận trả lời từ máy chủ trả về là một chuỗi định dạng chuẩn. Các đối số đầu vào là tên máy chủ và số cổng (7).

**Hướng dẫn:**

1. Tạo đối số
2. Tạo kết nối mạng
3. Tạo luồng dữ liệu
4. Nhập vào mỗi chuỗi khác dấu “.” và gửi tới Server. Ngược lại đóng kết nối
5. Đọc xâu từ Server và in ra dạng chuẩn

**Tạo class echoClient**

|  |
| --- |
| **import** java.net.\*;  **import** java.io.\*;  **public class** echoClient {  **final** **static** **int** defaultPort=7;  **public** **static** **void** main(**String**[] args) {  Socket ClientSocket;  **String** hostName;  **int** portNumber;  BufferedReader theInput;  PrintWriter theOutput;  BufferedReader userInput;  **String** inputString;  **switch**(args.length) {  **case** 1: hostName = args[0];  portNumber = defaultPort;  **break**;  **case** 2: hostName = args[0];  portNumber = **new** **Integer**(args[1]).intValue());  **break**;  **default**:  hostName = "localhost";  portNumber = defaultPort;  }  **try** {  System*.out.*println("Client side Echo utility");  ClientSocket = **new** Socket(hostName,portNumber);  theInput = **new** BufferedReader(**new** InputStreamReader(ClientSocket.getInputStream()));  theOutput = **new** PrintWriter(ClientSocket.getOutputStream());  userInput = **new** BufferedReader(**new**  InputStreamReader(System.in));  System*.out.*println("Enter your text or put only \'.\' to quit.");  **while** (**true**) {  inputString = userInput.readLine();  **if** (inputString.equals("."))  **break**;  theOutput.println(inputString);  System*.out.*println(theInput.readLine());  }  }  **catch** (UnknownHostException e) {  InterruptedException(" Unknown host error");  }  **catch** (ConnectException e) {  System*.out.*println(" Service unavailable on port " + portNumber + " of host " + hostName);  }  **catch** (IOException e) {  InterruptedException(" Communication error occured\r\n " + e);  }  }  } |

**3.2. Viết các ứng dụng kết nối phía Server sử dụng lớp ServerSocket**

Tạo đối tượng socket phía Server và truyền thông với giao thức TCP. Sau khi tạo ra, Server đặt ở trạng thái lắng nghe chờ tín hiệu kết nối gửi tới từ Client.

**Các phương thức tạo:**

* **public** ServerSocket(**int** port): Tạo đối tượng ServerSocket với số cổng xác định bởi tham số port, đáp ứng cực đại tới 50 kết nối đồng thời.
* **public** ServerSocket(**int** port, **int** queueLength): Chỉ ra số kết nối cực đại mà socket có thể đáp ứng đồng thời bởi tham số queueLenth.
* **public** ServerSocket(): Tạo đối tượng ServerSocket nhưng không gắn kết thực sự socket với một cổng cụ thể nào, như vậy không chấp nhận bất cứ kết nối nào gửi tới. Sử dụng phương thức bind() để gắn địa chỉ.

**Ví dụ:**

ServerSocket ss = **new** ServerSocket( );

SocketAddress http = **new** InetSocketAddress(80);

ss.bind (http) ;

**Các phương thức:**

**accept()**: Đặt đối tượng ServerSocket ở trạng thái “nghe” tại số cổng xác định, chờ tín hiệu kết nối gửi đến từ Client.

**close()**: Đóng socket, giải phóng tài nguyên

**Bài 4.**

Viết ứng dụng phía máy chủ lắng nghe trên cổng TCP cho trước và gửi ngày theo định dạng chuỗi tới Client. Đối số chương trình là số cổng mặc định giá trị cổng là 13.

**Hướng dẫn:**

1. Tạo đối số
2. Tạo Server socket
3. Chờ yêu cầu từ Client
4. Tạo kết nối tới Client
5. Tạo luồng dữ liệu
6. Gửi chuỗi thời gian tới Client
7. Đóng kết nối mạng tới Client

**Tạo class daytimeServer**

|  |
| --- |
| **import** java.net.\*;  **import** java.io.\*;  **import** java.util.Date;  **public class** daytimeServer {  **public** **final** **static** **int** daytimePort = 13;  **public** **static** **void** main(**String**[] args) {  ServerSocket theServerSocket;  Socket theConnectionSocket;  PrintWriter out;  **try** {  theServerSocket = **new** ServerSocket(daytimePort);  System*.out.*println("TimeServer ready at port "+daytimePort);  **try** {  **while** (**true**) {  theConnectionSocket = theServerSocket.accept();  System*.out.*println("Request arrived!");  out = **new** PrintWriter(theConnectionSocket.getOutputStream(), **true**);  out.println(**new** Date());  theConnectionSocket.close();  }  }  **catch** (IOException e) {  theServerSocket.close();  InterruptedException(e);  }  }  **catch** (IOException e) {  InterruptedException(e);  }  } |

**Bài 5.**

Viết ứng dụng phía máy chủ lắng nghe trên cổng TCP cho trước. Đọc chuỗi từ Client và gửi lại chuỗi cho Client như dịch vụ echo chuẩn. Đối số chương trình là cổng số 7.

**Hướng dẫn:**

1. Khởi tạo đối số
2. Tạo Server socket
3. Chờ yêu cầu từ Client
4. Tạo kết nối tới Client
5. Tạo luồng dữ liệu
6. Đọc dữ liệu từ Client sau đó gửi dữ liệu trả lại Client
7. Đóng kết nối mạng
8. Quay lại bước 3

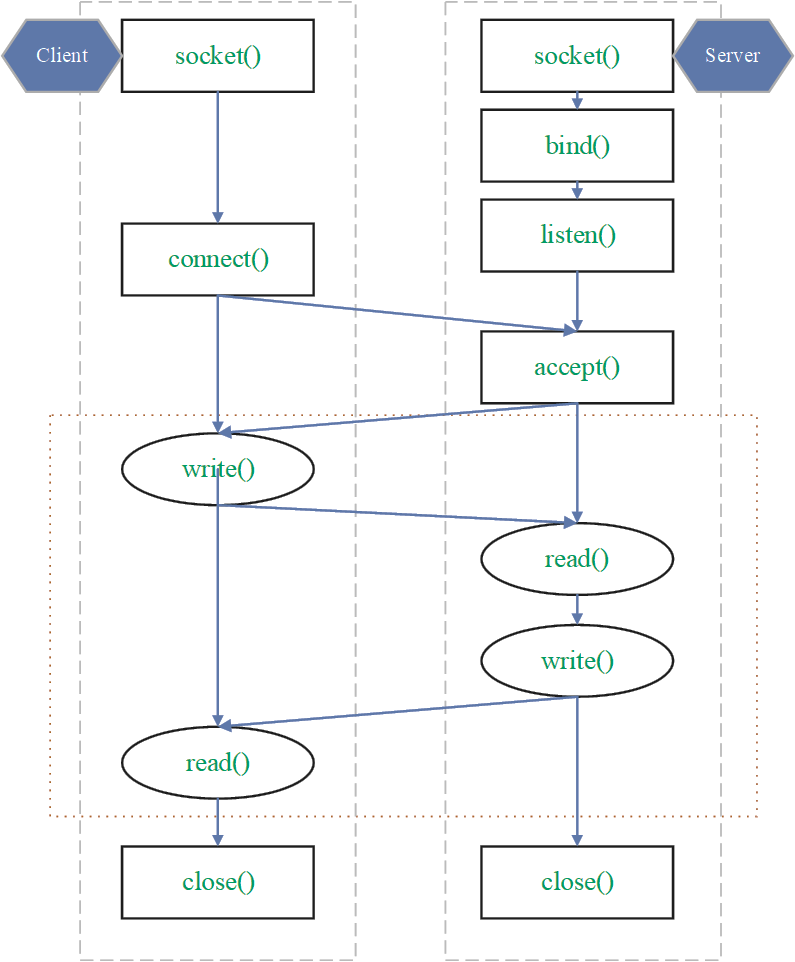
**Tạo class echoServer**

|  |
| --- |
| **import** java.net.\*;  **import** java.io.\*;  **public class** echoServer {  **public** **final** **static** **int** echoPort = 7;  **public** **static** **void** main(**String**[] args) {  ServerSocket theServerSocket;  Socket theConnectionSocket;  BufferedReader in;  PrintWriter out;  **try**{  theServerSocket = **new** ServerSocket(echoPort);  System*.out.*println("EchoServer ready at port "+ echoPort);  **while** (**true**) {  theConnectionSocket = theServerSocket.accept();  **try** {  System*.out.*println("Request arrived!");  in = **new** BufferedReader(**new**  InputStreamReader(theConnectionSocket.getInputStream()));  out = **new** PrintWriter(theConnectionSocket.getOutputStream(),**true**);  **while**(**true**) {  **String** readText = in.readLine();  out.println(readText);  }  }  **catch** (IOException e) {  theConnectionSocket.close();  }  }  }  **catch** (IOException e) {  InterruptedException(e);  }  }  } |

4. Lập trình Client- Server ở chế độ hướng kết nối

Để Client và Server có thể truyền thông được với nhau theo cơ chế hướng kết nối, mỗi phía phải thực hiện các thao tác sau:

1. **Phía Server TCP**
2. Tạo ServerSocket
3. Gọi thực thi phương thức accept() để chấp nhận thiết lập kết nối với Client => nhận được Socket giao tiếp với Client.
4. Lấy InputStream và OutputStream để nhận và gửi dữ liệu với Client.
5. Gửi và nhận dữ liệu với Client, sử dụng các phương thức read() và write() của các lớp InputStream và OutputStream.
6. Đóng Socket và ServerSocket
7. Kết thúc chương trình
8. **Phía Client TCP**
9. Tạo Socket kết nối đến Server
10. Lấy InputStream và OutputStream để nhận và gửi dữ liệu với Server.
11. Gửi và nhận dữ liệu với Server, sử dụng các phương thức read() và write() của các lớp đối tượng InputStream và OutputStream.
12. Đóng Socket
13. Kết thúc chương trình

**

*Hình 9. Mô hình Client-Server chế độ hướng kết nối*

**Lưu ý:**

* Chương trình Server luôn chạy trước chương trình Client.
* Một chương trình Server có thể phục vụ nhiều Client đồng thời

**Bài 6. Viết chương trình:**  
  - Client: Nhập vào từ bàn phím 2 số nguyên (a,b). Client chờ nhận kết quả từ Server để in ra màn hình  
 - Server: Nhận 2 số nguyên mà Client vừa gửi, tính tổng và gửi kết quả cho Client

**Hướng dẫn:**

**Tạo class Socket\_tong2so\_Client**

|  |
| --- |
| **import** java.net.\*  **public class** Socket\_tong2so\_Client {  **public** **static** **void** main(**String** []args) **throws** IOException  {  System*.out.*println("Client dang ket noi voi Server... ");  **String** a,b,tong;  //tạo socket để kết nối tới Server, Locallhost: Server mặc định  Socket ClientSocket = **new** Socket("Localhost", 1234);  //thong bao da ket noi thanh cong  System*.out.*println("Da ket noi voi Server! ");  //tao luong nhap du lieu tu ban phim  DataInputStream inFromUser = **new** DataInputStream(System.in);  //tao luong nhan du lieu tu Server  DataInputStream inFromServer = DataInputStream(ClientSocket.getInputStream());  //tao luong gui du lieu len Server  DataOutputStream outToServer = **new** DataOutputStream(ClientSocket.getOutputStream());  // nhap du lieu tu ban phim  **try** {  System*.out.*println("\n Nhap a :");  a=inFromUser.readLine();  System*.out.*println("\n Nhap b :");  b=inFromUser.readLine();  // gui len Server  outToServer.writeBytes(a+'\n');  outToServer.writeBytes(b+'\n');  }**catch**(UnknownHostException e)  {  InterruptedException("Server Not Found");  **System**.exit(1);  }**catch**(IOException e)  {  InterruptedException("Cannot make a connection");  }  //nhan ve tu Server  tong=inFromServer.readLine();  //in ra man hinh  System*.out.*println("\n Ket qua :"+tong);  //dong luong gui du lieu len Server  outToServer.close();  //dong luong nhan du lieu tu Server  inFromServer.close();  ClientSocket.close();  }  } |

**Tạo class Socket\_tong2so\_Server**

|  |
| --- |
| **import** java.io.\*;  **import** java.net.\*;  **public class** Socket\_tong2so\_Server {  **public** **static** **void** main(**String** []args) **throws** IOException  {  System*.out.*println("Server dang khoi dong... ");  **String** so1, so2, so3;  **int** tong;  // tao Server socket  ServerSocket Server = **new** ServerSocket(1234);  System*.out.*println("Server da san sang! ");  //tao 1 socket do ket noi tu Client toi Server  Socket connectionSocket = Server.accept();  //tao luong nhan du lieu tu Client  DataInputStreaminFromClient=**new** DataInputStream(connectionSocket.getInputStream());  // tao luong gui du lieu toi Client  DataOutputStreamoutToClient=**new** DataOutputStream(connectionSocket.getOutputStream());  // truyen du lieu tu Client vao 2 bien so1 va so2  so1 = inFromClient.readLine();  so2 = inFromClient.readLine();  //ep so1 va so2 tu kieu **String** sang kieu **Integer**  **int** a = **Integer**.parseInt(so1);  **int** b = **Integer**.parseInt(so2);  //tinh tong a + b  tong = a + b;  //ep tong sang kieu **String**  so3 = **String**.valueOf(tong);  //gui so3 ve Client  outToClient.writeBytes(so3+'\n');  //dong luong nhan du lieu tu Client  inFromClient.close();  //dong luong gui du lieu ve Client  outToClient.close();  //dong Server socket  Server.close();  }  } |

**Bài 7.**

* Client: Nhập vào từ bàn phím một số nguyên a. Client chờ nhận kết quả từ Server để in ra màn hình
* Server: Nhận số nguyên Client vừa gửi, kiểm tra tính chẵn (lẻ, nguyên tố, hoàn hảo), gửi kết quả cho Client

**Hướng dẫn:**

**Tạo file Server**

1. Tạo Server socket
2. Tạo 1 socket kết nối từ Client tới Server
3. Tạo luồng nhận dữ liệu từ Client
4. Tạo luồng gửi dữ liệu tới Client
5. Truyền dữ liệu từ Client vào biến so
6. Ép so từ kiểu **String** sang kiểu **Integer**
7. Áp dụng phương thức kiểm tra nguyên tố, hoàn hảo, chẵn lẻ xử lý cho so.

**Trong file Client**

1. Tạo socket để kết nối tới Server, Locallhost: Server mặc định
2. Tạo luồng nhập dữ liệu từ bàn phím
3. Tạo luồng nhập dữ liệu từ Server
4. Tạo luồng gửi dữ liệu lên Server
5. Nhập dữ liệu từ bàn phím
6. Gửi lên Server
7. Nhận dữ liệu từ Server
8. Đóng luồng dữ liệu gửi lên Server
9. Đóng luồng nhận dữ liệu từ Server
10. Đóng socket Client