Elementary Sockets

Lecture 3 (A)

Topic Outline

- **Introduction to TCP and UDP sockets**
- **TCP and UDP client-server programs**
- SCTP association
- I/O multiplexing and non-blocking I/O
- Socket options
- Remote Method Invocation
- Name and address conversions

Review of TCP vs UDP

TCP	UDP
Reliable	Unreliable
Connection-oriented	Connectionless
Segment retransmission and flow control through windowing	No windowing or retransmission
Segment sequencing	No sequencing
Acknowledge segments	No acknowledgement

Sockets

- A socket is an object that encapsulates a TCP/IP connection
- There is a *socket* on both ends of a connection

TCP Sockets

- Stream communication
  - Uses the **InputStream** and **OutputStream** classes in Java, as well as their subclasses, e.g. **DataInputStream, DataOutputStream, ObjectInputStream, ObjectOutputStream**, etc.

- Server program
  - Instantiates a **ServerSocket** object, bind the TCP port number
  - Invokes the **accept** method of the **ServerSocket** object to initiate listening. It returns a **Socket** object which will be used to obtain the required **InputStream** and **OutputStream** objects using the corresponding accessor methods

- Client program
  - Instantiates a **Socket** object to connect to the server host name/address and TCP port number.
  - **Socket** object which will be used to obtain the required **InputStream** and **OutputStream** objects using the corresponding accessor methods

TCP Sockets: Client Sockets & Server Sockets

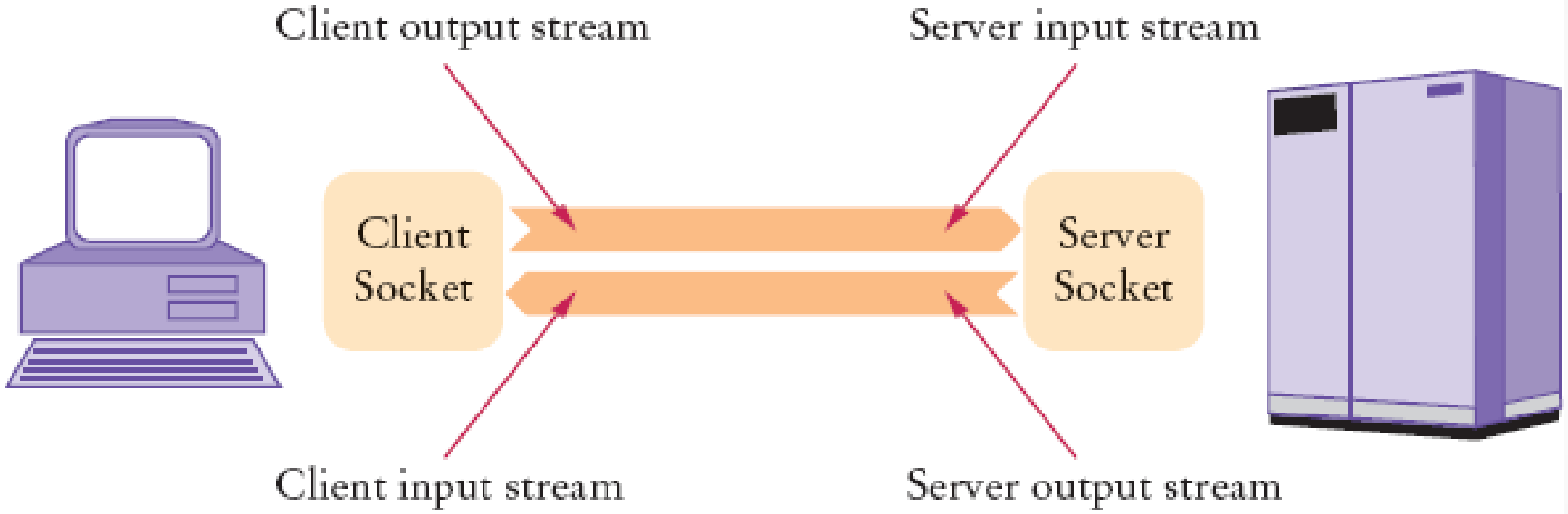
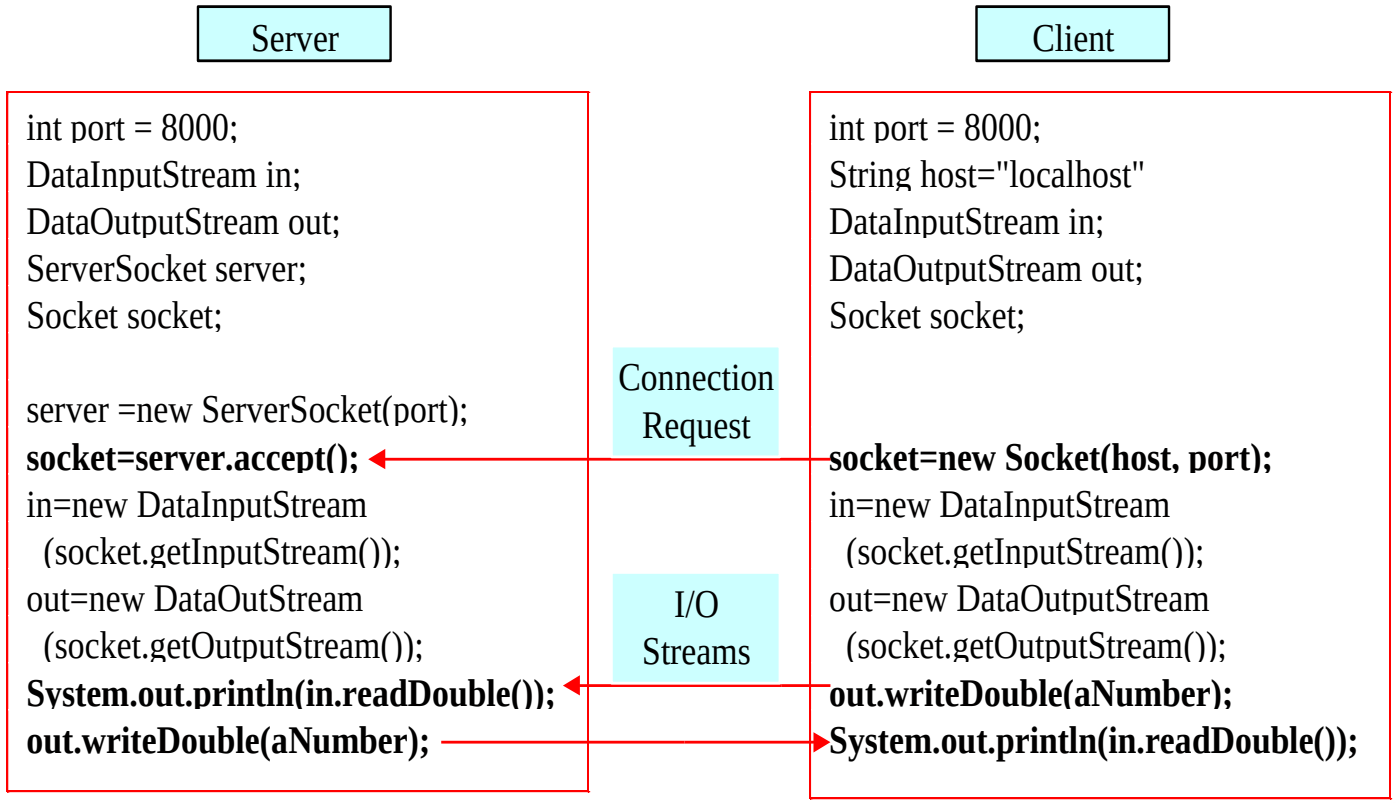


Figure 4 Client and Server Sockets

Data Transmission using TCP Sockets




```
InputStream input = socket.getInputStream();
OutputStream output = socket.getOutputStream();
```

Multithreaded Server

- Multiple clients are quite often connected to a single server at the same time. Typically, a server runs constantly on a server computer, and clients from all over the Internet may want to connect to it. You can use threads to handle the server's multiple clients simultaneously. Simply create a thread for each connection. Here is how the server handles the establishment of a connection:

```
while (true) {
  Socket socket = serverSocket.accept();
  Thread thread = new ThreadClass(socket);
  thread.start();
}
```

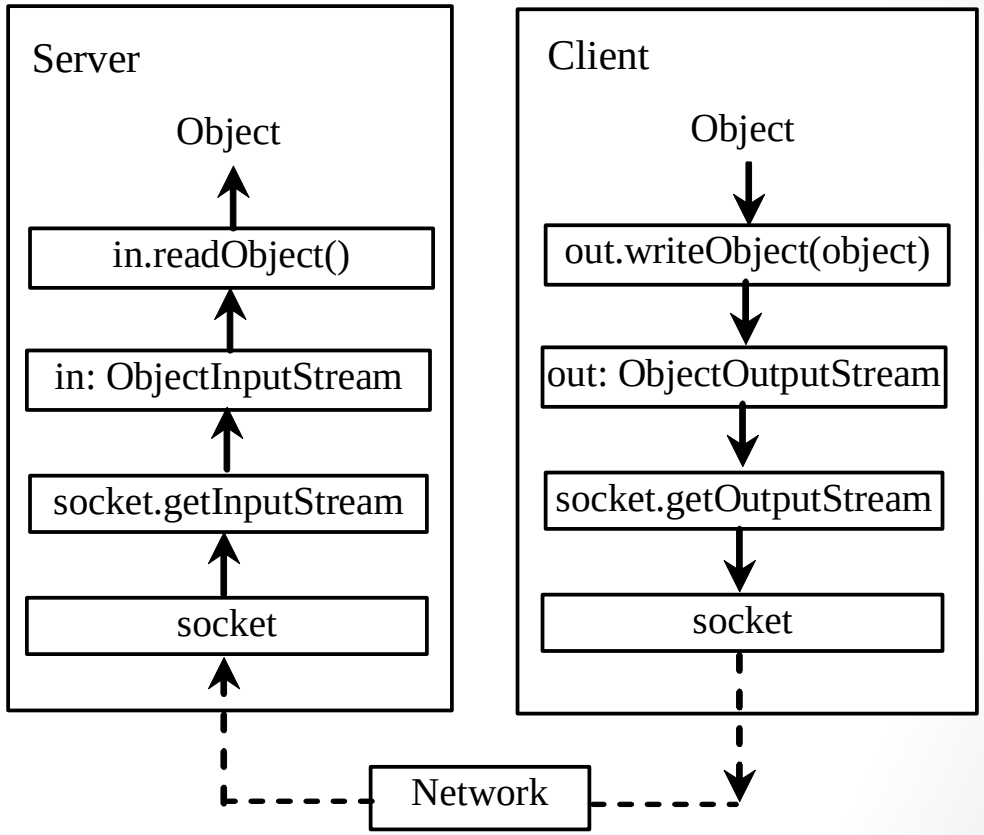
- The server socket can have many connections. Each iteration of the while loop creates a new connection. Whenever a connection is established, a new thread is created to handle communication between the server and the new client; and this allows multiple connections to run at the same time.

Sample Multithreaded Server Program & Client Program

- **Required Files:**
  - Circle.java
- **Server Program**
  - MultiThreadedServer.java
- **Client Program**
  - Client.java (console client program)
  - GUIClient.java (GUI client program)

Passing Objects in Network Programs

- Java objects can be passed between client & server programs
- The objects must be instantiated from a class that implements the java.io.**Serializable** interface



Note: Start the server first, then the client.

Sample Program: Object Passing

- **Server Program**
  - GeometryServer.java
  - **Required Files:**
    - Line.java
    - Point.java

- **Client Program**
  - GeometryClient.java
  - **Required Files:**
    - Point.java

Sample Program: ArrayList Passing

- **Server Program**
  - ArrayListDemoServer.java
  - **Required Files:**
    - Line.java
    - Point.java

- **Client Program**
  - ArrayListDemoClient.java
  - **Required Files:**
    - Line.java
    - Point.java