CS202: IT Workshop

Java

Networking

Ref:

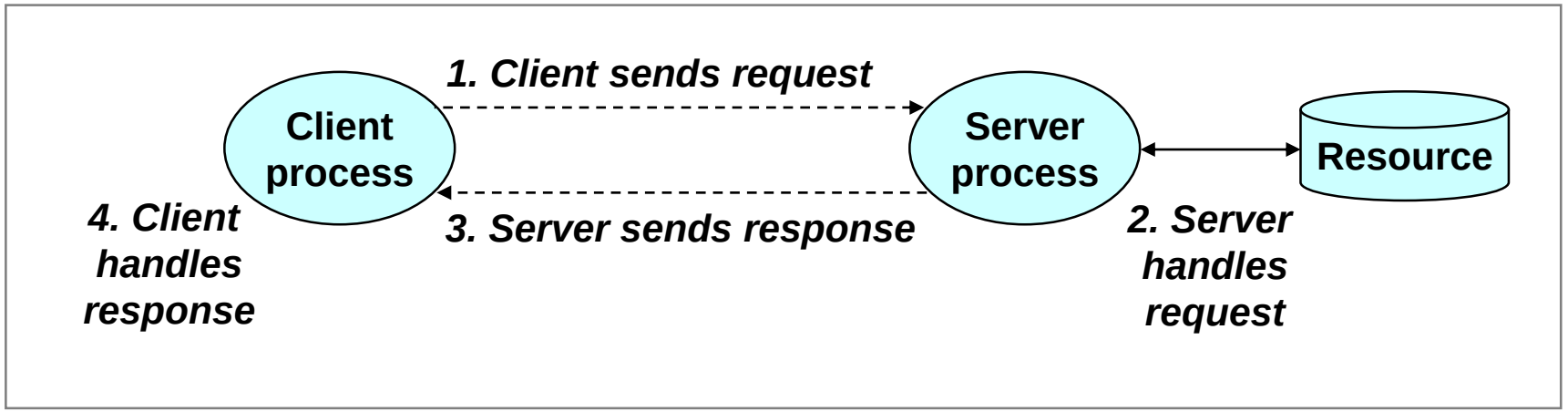
1. Harvey Deitel, Paul Deitel, **Java: How to Program**, 9/e, Prentice Hall India.
2. Herb Schildt: Java The Complete Reference, 8/e Tata Mcgraw Hill Education.

Networking basics

- Network: Connecting multiple computing elements to perform some work
- Computing elements (or hosts) have their addresses
- A 32 bit decimal dotted number is used as an IP address (IPv4) - - e.g. 127.0.0.1
- Applications on one host is identified by port number (16bit - - a few are reserved e.g. HTTP: 80, FTP: 21)
- Communication happens as per some pre-defined rules, called **protocols** (e.g. TCP, UDP, HTTP, FTP etc.)
- TCP→ Connection oriented → Connection is established → Data flows as stream (analogy: Telephone system)
- UDP → Connectionless → Doesn't guarantee delivery of packets (analogy: Postal system)

Client-Server architecture

- There is one server process and one or more client processes
- Client requests for some service from Server
- Server manages some resources and provides service by manipulating resources for clients.



Network programming in Java

- Writing programs that run in multiple systems which are connected using a network
- Support is available from **java.net** package
- Offers *stream-based* and *packet-based* communications (Handling data that follows from one device to another)
- Communication is established using **sockets**
- **Socket** is a software construct that enables one end-point of a communication
- Java supports both TCP and UDP
- A connection includes source **address**, destination **address**, source **port number**, destination **port number**

Network programming in Java: Address

- **InetAddress** class in Java supports a number of methods to deal with addresses (name and decimal dotted)

```
InetAddress address = InetAddress.getByName("google.com");

address.getHostName();

address.getHostAddress();
```

Returns the IP address associated with the INetAddress object address **216.58.196.78**

Returns the host name associated with the INetAddress object address **google.com**

Code (INetAddressTest.java): Screen share

Socket programming: Establishing connection using TCP (Server side)

1. **Creating a server socket**

   ServerSocket server = new ServerSocket (*portNumber*, *queueLength*);

   queueLength: maximum number of clients that can wait to connect to the server

2. **Server needs to wait for a *connection***

   Socket connection = server.accept();

   server listens indefinitely for an attempt by a client to connect.

3. **Get Socket's IO stream to read and write data**

   DataInputStream input = new DataInputStream (connection.getInputStream());
   DataOutputStream output = new DataOutputStream (connection.getOutputStream());

   Server sends information to client via Outputstream
   Server receives information from client via Inputstream

Socket programming: Establishing connection using TCP (Server side)

4. **Communication and processing of data**

5. **Closing the *stream* and *connection***

```
input.close();
connection.close();
server.close();
```

Code (MyServer.java): Screen share

Socket programming: Establishing connection using TCP (Client side)

1. Create a Socket to connect to the server

   Socket connection = new Socket(*serverAddress*, *portNumber*);

   Returns a socket if the connection is successful

2. Get Socket's IO stream to read and write data

   DataInputStream input = new DataInputStream (
   connection.getInputStream());
   DataOutputStream output = new DataOutputStream (
   connection.getOutputStream());

   Client sends information to server via Outputstream

   Client receives information from server via Inputstream

Socket programming: Establishing connection using TCP (Client side)

3. Communication and processing of data

4. Closing the *stream* and *connection*

```
output.close();
connection.close();
```

Code (MyClient.java): Screen share

Client-Server program using UDP

- Communication using TCP is reliable but it is costly!
- Java supports UDP using Datagrams
  - **Datagram packets**: Container for data
  - **Datagram socket**: used to send or receive datagram packets
- Datagram socket provides many methods

```
void send(DatagramPacket packet);

void receive(DatagramPacket packet);
```

receive() method waits for a packet to be received from the port specified by packet and returns the result

send( ) method sends packet to the port specified by packet

Code (WriteServer.java): Screen share

Questions?