Matt Kennedy
Brent Thompson

CIS 457
Section 101
Lab 03

**Question 1**

**a) Describe briefly, what this application is doing.**

The client sends the string “Hi world” to the server localhost on port 8888. The server, which we made, is waiting on port 8888. When it gets the string, it processes this request from the client.

**b) What happens if you use different port number at the client side to connect to the server from what the server uses?**

There is an error in the client side java application. The exception is “connection refused” because there is no localhost server on port 8889.

**c) Suppose you run TCPClient before you run TCPServer. What happens?**

The same thing as part b. There is no server on port 8888 to send a message to, so an exception that states “connection refused” happens.

**d) Test your app 3 times by starting 3 clients connections from the same terminal and record the port numbers that the client used. Are they the same in each connection and why?**

35760, 35762, 35764. No, the connections are not the same. In each connection, the port number is different because the server makes a new socket for each connection. Each socket is on a different port.

**e) Use the ps command with the appropriate switch to determine the Process ID for your application.**

29676

**f) Use the netstat command with the appropriate switch to determine the TCP connection state for this process**

The TCP connection state is ESTABLISHED.

**g) Provide a screen capture for the testing process that shows the application’s output in the 2 terminals.**

See attached screenshot #1.

**Question 2**

**a. Describe briefly, what this application is doing.**

The client is contacting the server, and the server is sending the client the current time.

**b. What is the return type of the read () method in this program?**

int num = in.read(b); This means the read() method returns an int.

**c. Why is there a single stream attached to either the client or the server side in this application?**
This is the pipe for the data to transfer between the client and the server.
**d. What is the type of I/O streams that is attached to the I/O sockets?**
InputStream in = server.getInputStream(); The type of stream is an InputStream.
The server has an OutputStream object.

**e. What are the limitations of using this type of I/O stream?**
The InputStream means the client can't send data to the server. It can only receive and request data.

**f. Provide a screen capture for the testing process that shows the application's output in the 2 terminals.**
See attached screenshots #2 & 3

**Question 3**
**a. How different is this application's implementation from the first one?**
In this example, the client is using a DataInputStream with a BufferedInputStream to help get messages. The buffer can help data transfer process.

**b. What is the type of I/O stream that is attached to the sockets?**
The I/O stream in use is a BufferedStream.

**c. In TimeClient2 and TimeServer2, what do these lines do?**

```
DataInputStream in = new DataInputStream(new
BufferedInputStream(server.getInputStream()));

DataOutputStream out = new DataOutputStream(new
BufferedOutputStream(client.getOutputStream()));
```

In the first example, this is in the TimeClient2.java file. This creates a new data input stream to receive data from the server through a BufferedInputStream object. In the second example, this is from the TimeServer2.java file, and this creates an output stream from the server through a BufferedOutputStream object. This helps in handling data sent from the server to the client.

**d. Which application implementation (TimeClient and TimeServer) or (TimeClient2 and TimeServer2) is recommended and why?**
The TimeClient2 and TimeServer2. The BufferedStream object on each I/O stream helps in handling data in each. The use of a buffer is significantly faster than a regular I/O stream with no buffer on it.

**Question 4**

**a) Provide a screen capture for the testing process that shows the application's output in the 2 terminals.**

See attached screenshots.

**b) Suppose you run UDPClient before you run UDPServer. What happens?**

The client will continue to take input from the user and sending packets to a supposed server. However, no server exists. The client will continue sending packets and waiting for a response, but because no confirmation is made, nothing happens.

**Suppose that in the UDPClient.java we replace the line**
**DatagramSocket clientSocket = new DatagramSocket();**
**with**
**DatagramSocket clientSocket = new DatagramSocket(5432);**

**c) Will it become necessary to change UDPServer.java? and why?**

No. From some research on StackOverflow: "The port which you're using is logically of no significance, the reason being server uses request.getPort() to determine the port while seeding the response; (request is a DatagramPacket sent by the client). "

**Question 5**

**a) Compile the given IPFinder.java program and describe its functionality.**

The program creates an InetAddress object, then uses the InetAddress.getByName(host) to determine the IP address of a given hostname.

**b) Modify the given IPFinder.java program so that it takes a list of host names from the command line and prints the host name and the IP address (s) for each host specified in the user input. The program should be able to retrieve the IP address of the local machine. (Hint: you need to use the getAllByName(), getHostName() and getHostAddress() methods to complete this part). Feel free to use the supplied code (listed below) and fill in the missing info or create your own one from scratch. The program output should look as it is shown in the figure below.**

**c) Provide a screenshot and the program code for the testing process. You can use any two domains that you may wish for the testing process.**

See attached screenshot #4.