

Homework 3

EXPERIMENTATION WITH NETWORKED COMMUNICATIONS

VERSION 1.0

The objective of this assignment is to get you to be comfortable coding in a networked setting where you need to manage the underlying communications between nodes. All communications will be based on TCP and you will be using the sockets to implement this assignment. The programming assignment should be implemented in Java. You are required to work alone on this assignment. The assignment accounts for 7.5% towards your cumulative course grade.

This assignment may be modified to clarify any questions (and the version number incremented), but the crux of the assignment and the distribution of points will not change. If there are any changes to the assignment, all changes will be documented in the "Change History" section of this assignment.

Released: March 6th, 2024

DUE DATES:

Submission 1: Wednesday, March 27th, 2024 @ 8:00 pm

Submission 2: Wednesday, April 3rd, 2024 @ 8:00 pm

Final Submission: Wednesday, April 10th, 2024 @ 8:00 pm

1 Description of Task

As part of this assignment, you will be responsible for designing programs that facilitate communications in a networked setting. Three machines will be involved during program execution. The programming assignment should be developed in Java, and your classes must reside in the package **cs250.hw3**.

You will be designing two separate programs that implement functionality relating to key roles in the system: a server and a client.

Command line execution:

```
java cs250.hw3.TCPServer <port-number> <seed> <number-Of-Messages>
```

Your server program will be provided with 3 arguments at the command line. The first argument specifies the port number that your server "listens" to. The second argument specifies the seed (an integer number) that is used to initialize the Server's random number generator. The final argument, number-Of-Messages, specifies the number of messages that will be sent by each client that is connected to the server.

Your client program will be provided with 2 arguments at the command line. The two arguments specify the hostname and port-number where the Server is listening to for communications.

Command line execution:

```
java cs250.hw3.TCPClient <server-host> <server-port>
```

1.1 Task 1: Server start-up

The server should first check to see if it is able to successfully create a server socket on the specified port. The value should be > 1024 $\leq 65,535$. You may encounter an exception if a server socket currently is bound to that port number, print that exceptions message using `exception.getMessage()`.

Upon start-up the server should initialize its random number generator using the specified seed.

1.2 Task 2: Registering two clients to the Server

Upon start-up the clients should first connect to the Server. Provided you have specified the correct server host/port information and the server is currently running this will result in a successful connection to the server.

In one message the server side sends two integers to the client. The first integer specifies the number of messages that the server expects the client to send. This number is the argument that was specified to the server at its command line: you should **not** hard-code this number in your client-side code.

The second integer is a random number from the Server that is used to initialize the client-side random number generator.

1.3 Task 3: Sending messages from the client to the server

Once the registration process is complete. The client must do two things.

First, it must wait for 10 seconds. This wait is to allow you to start-up another client and connect to the server.

Second, the client is required to use its client-side random number generator to produce the specified number of messages. Each message is an integer that was produced using the random number generator.

The client should also maintain a variable (long senderSum) which is used to maintain a running sum of all integer numbers that are being sent. Setting this variable as a long allows you to circumvent any overflow situations that would arise if you used an int instead.

The client should maintain a second variable (int numOfSentMessages) that tracks the number of messages that were sent by the client.

1.4 Task 4: Processing a message that is relayed from the server to the client

Any message received by a server from a client (say A) will be relayed (or forwarded) to the other client (say B) that is connected to the server.

A client should maintain a running count of the messages (each of which is just a number) forwarded by the server. This information is maintained in another variable (long receiverSum).

The client should maintain a second variable (int numReceivedMessages) that tracks the number of messages that were received by the client.

1.5 Task 5: Printing metadata information

This task entails printing metadata information at the server and the two clients.

In particular, the information that should be printed includes the number of messages and the running counts of the messages that are sent and received by each client.

This information is printed at the server and each of the clients. Printing this metadata information is what allows you to confirm the correctness of your implementation. The running counts should match up. For example:

1. Let senderSum at client A be X
2. Let the receiverSum at client B be Y. This receiverSum corresponds to messages from client A that were relayed by the server.
3. If your implementation is correct then $X=Y$. That is, what you sent is what was received!

2 What to Submit

Use the CS250 Canvas to submit a single .zip file that contains:

- Your java source codes, matching this directory structure:
 - cs250
 - hw3
- A README.txt file containing a description of each file and any information you feel the TAs need to grade your program.

Filename Convention: The archive file should be named as <FirstName>-<LastName>-HW2.zip . E.g., if you are Cameron Doe then the zip file should be named Cameron-Doe-HW3.zip.

3 Grading

The assignments must compile and function correctly on machines in the CSB-120 Lab. Assignments that work on your laptop on your particular flavor of Linux, but not on the Lab machines are considered unacceptable.

This assignment will contribute a maximum of 7.5 points towards your final grade. The grading will also be done on a 7.5 point scale. There will be multiple submissions each building upon the previous for this assignment to ensure steady progress is made throughout the assignment duration. The points breakdown is as follows:

Points Breakdown

Submission One (2 points):

Tasks 1 and 2 should be completed. We will supply a compiled Client that can be used to connect to your server and will expect to receive one message with two integers. The first integer is the number of messages that the client should send back to the server. The second integer is the seed that should be used to initialize the client's random number generator.

Example Outputs:

Success

Server (Neptune):

Command: java cs250.hw3.TCPServer 1345 42 4

IP Address: neptune/129.82.44.122

Port Number 1345

waiting for client...

Clients Connected!

Sending config to clients...

earth.cs.colostate.edu -1170105035

jupiter.cs.colostate.edu 234785527

Finished sending config to clients.

Client A (Jupiter):

Command: java SubmissionOneClient neptune 1345

Received config

number of messages = 4

seed = 234785527

Client B (Earth):

Command: java SubmissionOneClient neptune 1345

Received config

number of messages = 4

seed = -1170105035

Port already in use

Server (Neptune):

Command: java cs250.hw3.TCPServer 1345 42 4

IP Address: neptune/129.82.44.122

Port Number 1345

Address already in use (Bind failed)

Submission Two (2 points):

Task 3 should be completed. We will supply a Server that your client will connect to. The server will send the client one message with two integers. The first integer is the number of messages that the client should send back to the server. The second integer is the seed that should be used to initialize the client's random number generator. The client should then send that number of messages back to the server each containing one random integer. The client should maintain a count of messages sent along with the sum of the random numbers sent.

Example Output:

Server (Neptune):

Command: java cs250.hw3.SubmissionTwoServer 1345 42 30

IP Address: neptune/129.82.44.122

Port Number 1345

waiting for client...

Clients Connected!

Sending config to clients...

earth.cs.colostate.edu -1170105035

jupiter.cs.colostate.edu 234785527

Finished sending config to clients.

Starting to listen for client messages...

Finished listening for client messages.

earth.cs.colostate.edu

Messages received: 30

Sum received: -6382135587

jupiter.cs.colostate.edu

Messages received: 30

Sum received: -2496366311

Client A (Jupiter):

Command: java cs250.hw3.TCPClient neptune 1345

Received config

number of messages = 30

seed = 234785527

Starting to send messages to server...

Finished sending messages to server.

Total messages sent: 30

Sum of messages sent: -2496366311

Client B (Earth):

Command: java cs250.hw3.TCPClient neptune 1345

Received config

number of messages = 30

seed = -1170105035

Starting to send messages to server...

Finished sending messages to server.

Total messages sent: 30

Sum of messages sent: -6382135587

Submission Three (3.5 points):

Task 4 and 5 should be completed. Your client and server should be compatible with our implementations meaning that given any combination of Client, Server task 4 and 5 should result in the same output given the same input arguments.

Example Output:

Server (Neptune):

Command: java cs250.hw3.TCPServer 1345 42 30

IP Address: neptune/129.82.44.122

Port Number 1345

waiting for client...

Clients Connected!

Sending config to clients...

earth.cs.colostate.edu -1170105035

jupiter.cs.colostate.edu 234785527

Finished sending config to clients.

Starting to listen for client messages...

Finished listening for client messages.

earth.cs.colostate.edu

Messages received: 30

Sum received: -6382135587

jupiter.cs.colostate.edu

Messages received: 30

Sum received: -2496366311

Client A (Jupiter):

Command: java cs250.hw3.TCPClient neptune 1345

Received config

number of messages = 30

seed = 234785527

Starting to send messages to server...

Finished sending messages to server.

Total messages sent: 30

Sum of messages sent: -2496366311

Starting to listen for messages from server...

Finished listening for messages from server.

Total messages received: 30

Sum of messages received: -6382135587

Client B (Earth):

Command: java cs250.hw3.TCPClient neptune 1345

Received config

number of messages = 30

seed = -1170105035

Starting to send messages to server...

```
Finished sending messages to server.  
Total messages sent: 30  
Sum of messages sent: -6382135587  
Starting to listen for messages from server...  
Finished listening for messages from server.  
Total messages received: 30  
Sum of messages received: -2496366311
```

Deductions:

1 point each for not following the prescribed argument specification.

3 points for hardcoding any values in the system; everything should be argument driven including the random number generator.

You are required to **work alone** on this assignment.

4 Late Policy

Please check the class policy on submitting late assignments. You are allowed to submit assignments up to 2 days with a per-day deduction of 7.5%.

5 Version Change History

This section will reflect the change history for the assignment. It will list the version number, the date it was released, and the changes that were made to the preceding version. Changes to the first public release are made to clarify the assignment; the spirit or the crux of the assignment will not change.

Version	Date	Comments
1.0	3/6/2024	First public release of the assignment.