PROGRAMMING ASSIGNMENT 1

EXPERIMENTING WITH NETWORK COMMUNICATIONS

Version 1.0

DUE DATE: Wednesday, September 6th, 2017 @ 5:00 pm

OBJECTIVE
The objective of this assignment is to get you to be comfortable coding in a distributed setting where you need to manage the underlying communications between nodes. Upon completion of this assignment you will have a set of reusable classes that you will be able to draw upon for the next several years. The assignment requires you to verify correctness by ensuring that: (1) the number of messages that you send and receive within the system match, and (2) these messages have been not corrupted in transit to the intended recipient. Message exchanges in the system will be done in the presence of rapid setup/teardown of connections and while a connection is alive. This assignment will be modified to clarify any questions, but the crux of the assignment and the distribution of points will not change.

Grading: This assignment will account for 10 points towards your cumulative course grade. The components of this assignment, and the points breakdown is listed in the remainder of the text. This assignment is to be done individually. The lowest score that you can get for this assignment is 0. The deductions will not result in a negative score for this particular assignment.

Setting
In this assignment, you will be creating a set of processes P that are launched on different machines. At each process, you will read information about this set of processes from a text file: each line in the file will contain the host and port information for the process comprising this set. A process should never attempt to connect to itself; so it should not appear its list of available processes. It follows that the config file at each process will be different.

Each process participates in a set of rounds. Each round involves a process connecting to a randomly chosen process in the set of processes P. All communications in the system will be based on TCP. Once a connection is established to a random node, the initiating process sends 5 messages to the targeted process. The payload of each message is a random integer (positive or negative). At the end of each round the socket connection is closed and the process is repeated by choosing another node at random from the set P. Each process will initiate 5000 such rounds.

The number of processes will be fixed at the start of the experiment. We will likely use between 5-9 processes for the test environment during grading.

Each process will maintain two integer variables that are initialized to zero: the sendTracker and the receiveTracker. The sendTracker represents the number of messages that were sent by a process...
and the receiveTracker maintains information about the number of messages that were received by that process. Consider the case where there are 10 processes in the system as depicted in Figure 1. Since every process initiates 5000 rounds, which contain 5 messages each, the number of messages sent by every process is 25,000. With 10 processes in the system, the total number of messages would be 250,000. Since the sending process chooses the recipient process for each round at random, the number of messages received by different receivers would be different; however, because each round has 5 messages, the total number of messages received at a receiver would be a multiple of 5 and around 25,000 (i.e. it could 24000, 24595, 26905, ...).

Figure 1: Depiction of a possible distribution of the number of messages sent and received within a set of 10 processes.

Each process will maintain two additional integer variables that are initialized to zero: sendSummation and receiveSummation. The variable sendSummation, continuously sums the values of the random numbers that are sent as part of the payload, while the receiveSummation sums values of the payloads that are received. The values of sendSummation and receiveSummation at a process can be positive or negative.

Correctness Verification
We will verify correctness by: (1) checking the number of messages that were sent and received, and (2) if these packets were corrupted for some reason.

The total number of messages that were sent and received by the set of processes P must match i.e. the cumulative sum of the receiveTracker at each process must match the cumulative sum of the
sendTracker variable at each process. We will check that these packets were not corrupted, due to a bug in the code, by verifying that when we add up the values of sendSummation it will exactly match the added up values of receiveSummation.

**Outputs Collator:** To assist in the grading, you are also required to implement a Collator process. Each process will send metadata about the number of messages received and sent alongside the corresponding summation totals. The collator process should be initialized to track the number of processes. The final totals should be printed out at the Outputs Collator as depicted below.

**Example Output:**
The collated outputs from 10 nodes are depicted below. Note how the number of received messages may be slightly different than the number of sent messages. The summation of sent or received messages at a node may be positive or negative. In the particular example, depicted below, the final summation across all nodes is negative, it may well be positive in your case and that is fine!

<table>
<thead>
<tr>
<th>Node</th>
<th>Number of Messages Sent</th>
<th>Number of messagesReceived</th>
<th>Summation of sent messages</th>
<th>Summation of received messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node 1</td>
<td>25000</td>
<td>27640</td>
<td>-1623496115</td>
<td>37729102</td>
</tr>
<tr>
<td>Node 2</td>
<td>25000</td>
<td>28215</td>
<td>603118767</td>
<td>2047691194</td>
</tr>
<tr>
<td>Node 3</td>
<td>25000</td>
<td>28250</td>
<td>-297982786</td>
<td>958380326</td>
</tr>
<tr>
<td>Node 4</td>
<td>25000</td>
<td>28245</td>
<td>938740072</td>
<td>346940343</td>
</tr>
<tr>
<td>Node 5</td>
<td>25000</td>
<td>27495</td>
<td>-774595896</td>
<td>675396635</td>
</tr>
<tr>
<td>Node 6</td>
<td>25000</td>
<td>27125</td>
<td>-2033483570</td>
<td>-582001630</td>
</tr>
<tr>
<td>Node 7</td>
<td>25000</td>
<td>0</td>
<td>760422371</td>
<td>0</td>
</tr>
<tr>
<td>Node 8</td>
<td>25000</td>
<td>27270</td>
<td>-2110007534</td>
<td>1523906513</td>
</tr>
<tr>
<td>Node 9</td>
<td>25000</td>
<td>28050</td>
<td>1534124827</td>
<td>1443505176</td>
</tr>
<tr>
<td>Node 10</td>
<td>25000</td>
<td>27710</td>
<td>1960796424</td>
<td>1096023493</td>
</tr>
<tr>
<td>Sum</td>
<td>250000</td>
<td>25000</td>
<td>-1042363440</td>
<td>-1042363440</td>
</tr>
</tbody>
</table>

**Points distribution:**
3 points: Processes make connections and terminate connections without crashing.
2 points: Implementation of the Outputs Collator
2 point: Message totals for send and receive match
3 points: The cumulative totals for sendSummation and receiveSummation are identical.
Third-party libraries and restrictions:
The assignment must be implemented using the java.net.*, java.util.*, and java.io.* package, you can also use other utility packages in the core Java package so long as you are not using packages relating to RPC or distributed object frameworks. If you use distributed objects or RPC to develop this functionality there will be a 10 point deduction. You are also not allowed to use any external jar files. You can discuss the project with your peers at the architectural level, but the project implementation is an individual effort.

Testing Scenario
The grading process will involve a one-on-one interview session where you will be asked several questions about aspects of your code. You will be asked to launch between 5-10 processes on different machines. You may also be asked to make small changes to the logic of the program and redeploy the system during this grading process.

Submission deadline:
Please submit the source codes for your project by 5:00 pm on the due date. Please submit a zip file containing your source codes and a Readme.txt using CANVAS. We will rely on the honor system: please do not make any modifications to the codebase after the submission deadline has elapsed. There will be steep deductions for making modifications to the source code after you have submitted it.

Nota Bene: Please do not e-mail the source codes to the Professor or the GTA – there will be a 3 point deduction for doing this.