Assignment 4 Write-Up

Time Measurements

- **NOTE**: The professor said in class on March 18th that method B (using java.time.Instant) does not work and to only test our code with System.nanoTime
- **ASSUMPTION**: A lot of students misunderstood the sequence diagram for assignment 3 and I was not able to get a clear response from the prof or a ta, so for this assignment I assumed we want to keep track of the time it takes to process 1 RPC call in the Intermediate Host
    - 1 RPC Call
        - Start Time: When the intermediate host receives data from the client
        - End Time: When the intermediate host sends the client the response from the server
    - Because of the way I understood + built Assignment 3, first I need to add a new variable to the constructor to know which Intermediate Host thread was in charge of communicating to the client
    - Then I placed the marker to get the start time on **line 129** of IntermediateHost.java. This is the earliest place where I can know in my code that I got data from the client. It would be ideal to start the timer right when you receive a packet, but because of the way my code was optimized in Assignment 3, this is not possible for me.
        - I do not want to overwrite the start timer when I receive a packet from the client that is asking for data
    - I placed the marker to get the end time on **line 57** of IntermediateHost.java
        - Here, I check if I'm in the client thread. Then, I check if I'm sending a "Request acknowledged" packet, if not that means I'm sending data to the client.
        - Once I know that, I end the timer and add the elapsed time to a list
    - Once all 1000 RPC calls have been made, I print out the average of elapsed times and the variance of the 1000 time measurements

Calculations

- Output
    - Number of times collected: 1000
    - average: 3.0058228445E9
    - variance: 1.6017124182673544E16
- Average of 1000 RPC Calls
    - n = 1000
    - $\bar{X}$ = 3.0058228445 x 10^9 nanoseconds
- Variance of 1000 RPC Calls
    - 1.6017124182673544 x 10^{16} nanoseconds
- Std Dev of 1000 RPC Calls
    - sqrt(variance) = sqrt(1.6017124182673544 x 10^{16} nanoseconds)
    - Std dev (s1) = 126558777.581 nanoseconds

- What is the 95% Confidence Interval
  - $\bar{x} \pm Z * \frac{s}{\sqrt{n}}$
  - Z value for 95% is 1.960
  - $3.0058228445 \times 10^9 \pm 1.960 * \frac{126558777.581}{\sqrt{1000}}$
  - $3.0058228445 \times 10^9 \pm 7844194.303\ nanoseconds$
  - 95% of experiments will have a true mean within this range, $3.0058228445 \times 10^9 \pm 7844194.303\ nanoseconds$
    - Max = 3013667039 nanoseconds
    - Min = 2997978650 nanoseconds
  - After looking at the data points for 1000 RPC Calls, there is only one data point that can be considered an outlier
    - The last RPC call takes more than double the average time but that can be explained because the last call is made with invalid data and the server throws an IOException and closes its sockets. The intermediate host waits to receive something and there are lots of timeouts that occur throughout the end process

Interpretation of Data

- On average, it will take my program 3005822844.5 nanoseconds or 3.0058228445 seconds to process 1 RPC call/data packet
- You can expect the time for 1 RPC call to be processed in somewhere between 2997978650 nanoseconds and 3013667039 nanoseconds
- These numbers make sense because of my program's flow of events include:
  1. The client sends data to the Intermediate host 1, the Intermediate host 1 puts the data in a Box class, and sends an acknowledgement packet back.
  2. The Server sends a request for data to the Intermediate host 2, the Intermediate host 2 gets the data from the box and sends a packet to the server with the client data
  3. The Server processes the client data and sends the response to the intermediate host 2. Then, the intermediate host 2 receives it, puts it in the box, and sends an acknowledgement packet back.
  4. The Client sends a request for data to the intermediate host 1 and the intermediate host 1 gets data from the box and replies with the server data.
- In my multi threaded program, I am enforcing the steps above to be sequential so that the client receives the correct response from the server for the packets it sends. Since I am measuring the time between client sending data to the intermediate host 1 and Intermediate Host 1 sending the server data and since it is sequential, it makes sense that this whole process takes around 3 seconds. Within those three seconds, the intermediate host is receiving 4 packets, the intermediate host is sending out 4 packets, and is waiting for the server to process the client data.

Ran Program a Second Time

- Output
  - Number of times collected: 1000
  - average: 3.00567988E9
  - variance: 1.5993502451883284E16
  - Std dev: 126465419.985
  - → Very similar to the first run
- Combined Average of 2000 RPC Calls
  - n = 2000
  - $\bar{X}$ = (1000 * 3.0058228445 x 10^9 nanoseconds) + (1000 * 3.0056798800 x 10^9 nanoseconds)/2000
  - $\bar{X}$ = 3.005751362 x 10^9 nanoseconds
- Combined Variance of 2000 RPC Calls
  - $$s_p^2 = \frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}$$
  - s1= 126558777.581 nanoseconds
  - s2 = 126465419.985 nanoseconds
  - n1 = 1000
  - n2 = 1000
  - Combined variance = 1.600531332 x 10^{16} nanoseconds
- Combined Std Dev of 2000 RPC Calls
  - = sqrt(combined variance)
  - = 126512107.4 nanoseconds
- Confidence Interval
  - $\bar{x} \pm Z * \frac{s}{\sqrt{n}}$
  - Z value for 95% is 1.960
  - $3.005751362 \times 10^9 \pm 1.960 * \frac{126512107.4}{\sqrt{2000}}$
  - $3.005751362 \times 10^9 \pm 5544637.573\ nanoseconds$
  - 95% of experiments will have a true mean within this range, $3.005751362 \times 10^9 \pm 5544637.573\ nanoseconds$
    - Max = 3011296000 nanoseconds
    - Min = 3000206724 nanoseconds
- More precise values, very similar to first run