Engineering Notebook - Logical Clocks

Hanying Feng and Vera Yang

**Design Choices and Implementations:**

In this design assignment, we used threading to run multiple machines concurrently and sockets for communication between machines. The code is written in Python. We first created a class **VirtualMachine** with the following important attributes:

* *vm\_id*: A unique identifier for the virtual machine.
* *peers*: A list of peer machine ports.
* *port*: The port number for the current virtual machine to listen on.
* *clock\_speed*: The number of clock ticks per real-world second for the virtual machine (1-6 in the first half of the experiments, and 2-4 in the later experiments to test with smaller variations in clock speed).
* *logical\_clock*: A number that records the logical clock values for the virtual machine.
* *message\_queue*: A queue that stores all messages sent from other machines.
* *running*: A boolean value indicating whether the virtual machine should continue running.

We first created three virtual machines by passing parameters for the ID, peer list, and port number, then started a thread for each virtual machine so that they run as separate processes. Each thread starts by calling the in-class function **run()**.

**run(self)**

This function handles the main execution loop. It first starts a thread for listening to messages. Then, we implemented a while loop to ensure the code runs for 60 seconds. In each iteration, the code calls the function **process\_cycle()** and waits for 1 / *clock\_speed* seconds so that the virtual machine executes *clock\_speed* number of events per real-world second.

**listen(self)**

This function starts a socket that listens on the assigned port number. If a message is received, it is added to the *message\_queue*.

**process\_cycle(self)**

This function first checks whether the *message\_queue* is empty:

* If the message queue is not empty, it compares the received message’s logical clock value with the current virtual machine’s logical clock. The logical clock is updated to the larger value plus 1. The event is logged, starting with “Received message.” The log also records the number of unread messages (the size of the queue after retrieving the current message).  
  *Example: Received message: Logical Clock = 2, System Time = 2025-02-28 11:32:29.742007, Queue Length = 0*
* If the message queue is empty, the logical clock is incremented by 1. A random number between 1 and 10 is generated to determine the next action (1-5 in later experiments to test for a smaller probability of internal events).
  + If action = 1 or 2, the logical clock value is sent to the first or second virtual machine in the peers list by calling **send\_message(target)**. The event is logged as “Sent message” with the recipient's virtual machine ID.  
    *Example: Sent message to 6002: Logical Clock = 4, System Time = 2025-02-28 11:32:30.416880*
  + If action = 3, the logical clock value is broadcast to both peer virtual machines using **send\_message(target)** iteratively. The event is logged as “Broadcast message,” listing the recipient IDs.  
    *Example: Broadcast message: Logical Clock = 47, System Time = 2025-02-28 11:32:39.211176*
  + For other values of action, it is considered an internal event and logged as “Internal event.”  
    *Example: Internal event: Logical Clock = 1, System Time = 2025-02-28 11:32:29.405071*

**send\_message(target)**

This function sends a message containing the logical clock value to a *peer[target]* through a socket.

**stop()**

This function stops execution and closes the log file.

After running the program for 60 seconds, we call the **stop()** function to terminate each virtual machine’s execution.

Once we finished writing the code, we tested the program by checking the log files to ensure it worked as expected. We found a bug where the logical clock was mistakenly updated in **send\_message(target)**, causing the value to be incremented twice when a virtual machine broadcasted a message (action = 3). We fixed this by updating the logical clock only once when the virtual machine did not have a message in its queue. Additionally, we encountered an “Address already in use” error at the beginning. We discovered that this was due to using port 5000, which conflicted with a system port on macOS. We resolved this by changing the port number to 6000. After debugging other minor issues, the code ran successfully.

Next, we wrote unit tests to verify that the virtual machine could listen, send messages, process internal events, receive messages, and stop correctly. Finally, we added comments and docstrings throughout the code. We also created a README.md file to document and explain all the functions of the **VirtualMachine** class.

After ensuring the code worked correctly, we ran the experiment. To better analyze the results, we wrote additional code to calculate the average size of jumps in the values of the logical clock for each virtual machine. This helped us better quantify the results.

**Experiments and Observations:**

We conducted multiple runs of the system, observing varying magnitudes of differences in the speeds of the three machines. We also experimented with smaller variations in machine speeds by shrinking the range of random speeds from 1-6 to 2-4. Additionally, we reduced the probability of an internal event by selecting a random integer from 1-5 instead of 1-10, which resulted in runs where more messages were sent. Our observations and findings from these experiments are summarized below.

1. **Size of jumps in the values for the logical clocks:**

We observe that machines with higher speeds generally have smaller jumps in logical clock values, while machines with lower speeds tend to have larger jumps. For example, in an experiment where the machines have speeds of 1, 3, and 6, the average jump sizes are approximately 3.5, 1.9, and 1, respectively. When the speed differences between machines are smaller, the differences in jump sizes also become smaller. For instance, machines with speeds of 3, 2, and 2 have average jump sizes of 1, 1.5, and 1.5, respectively. When all machines have the same speed, their jump sizes are also nearly identical. For example, machines with speeds of 4, 4, and 4 all have average jump sizes of around 1.3.

This is because faster machines have logical clocks that update more quickly through internal events and receive messages less frequently than slower machines within the same amount of real-world time. As a result, their logical clock values are more likely to be larger than those sent from other machines, which allows them to update their logical clocks at their own pace most of the time. This leads to smaller sizes of jumps. In contrast, slower machines receive messages more frequently relative to their own ticking rate, and the logical clock values they receive are more likely to be larger than their current values. Consequently, they update their logical clocks based on the values received from faster machines, which results in larger jumps in their logical clocks.

1. **Drift in the values of the logical clocks:**

We find that the drift in the values of the logical clocks is usually larger when there is a greater difference in machine speeds. For example, in an experiment with machines running at speeds 1, 3, and 6, the logical clock values after 1 minute are 209, 347, and 151, respectively. We also notice that the message queue length for the machine with speed 1 was 36 at the end of the experiment. We believe this large drift is primarily due to the slow machine updating its logical clock too infrequently, preventing it from processing incoming messages from faster machines in a timely manner.

|  | Speed 1 | Speed 3 | Speed 6 |
| --- | --- | --- | --- |
| 1 second | 1 | 1 | 1 |
| 2 second | 2 | 7 | 7 |
| 3 second | 7 | 10 | 13 |
| 4 second | 9 | 17 | 19 |
| 5 second | 11 | 23 | 25 |

The table above shows the logical clock values recorded during the first 5 seconds of real-world time for the same experiment. We observe that the machine with speed 1 already exhibits a significant drift in its logical clock values after just a few seconds. Additionally, its message queue length reaches 4 by the 5-second mark. In contrast, the drift between the machines with speeds 3 and 6 is relatively small, as the slower machine (speed 3) is able to update its logical clock frequently enough to stay synchronized with the faster machine. This is further supported by the fact that the machine with speed 3 had a message queue length of 0, which indicates that it can process messages in time.

In all other cases where the speed differences between machines were smaller (≤3) and the probability of internal events was 7/10, the observed drift at the end of the experiment is typically less than 5.

1. **Length of the message queue:**

We observe that, when the speeds of the three machines differ by a larger magnitude, the slower machines will have longer queues than the faster machines. For instance, in one run, we had speed 1, 3, and 6 for the three virtual machines. At the end of the 1-minute run, the machine with speed 1 had 36 messages queued, while the speed-3 machine had only 1 message queued occasionally, and the speed-6 machine never had any message queued. The faster machine with speed 6 can process more messages every second, and thus can always get to all messages it receives. On the contrary, the speed-1 machine not only processes messages slower, but also receives more messages from its faster peers, and thus it has a longer queue of messages that accumulates as the system runs.

1. **Experiment with a smaller variation in machine speeds:**

We changed the range of random machine speeds from 1-6 to 2-4 to obtain multiple runs with smaller differences in the speeds of the three virtual machines. Below are our observations on how a smaller variation affects the size of jumps in logical clock time, the size of the drift, and the length of message queues respectively.

* One observation is that when the machine speeds differ by a larger magnitude, the sizes of jumps in the logical clock time for each machine also differ by a notable amount. However, when the speeds of the three machines are very similar, the machines tend to have more similar sizes of jumps in their logical clock values. For example, in the run with machine speeds 1, 3, and 6, the average sizes of jumps in logical clock values for the three machines are 3.53, 1.95, and 1 respectively. However, in the run with machine speeds 3, 2, and 2, the average sizes of jumps in the logical clock values for the three machines are 1, 1.51, and 1.5 respectively. This exhibits the same pattern that faster machines have smaller jumps whereas slower machines have larger jumps, but the difference in sizes of jumps becomes smaller as the speeds of machines become closer to each other.
* Another observation is that when machine speeds differ by a large magnitude, the slowest machine tends to have a large drift, but when machine speeds are similar, the drifts in slower machines are smaller. For instance, in the run with machine speeds 1, 3, and 6, after running for 1 minute, the logical clock values are 209, 347, and 351. The speed-1 machine has a drift of 142 when compared to the speed-6 machine. However, in the run with machine speeds 3, 2, and 2, after running for 1 minute, the logical clock values are 179, 179, and 178, which indicates that the speed-2 machines were able to catch up with the speed-3 machine and there was almost no drift.
* We also observe that when machine speeds differ a lot, the slowest machine can have a long queue of messages, but when the machine speeds are closer, there are very short queues in all machines. For example, when the machine speeds are 1, 3, and 6, the speed-1 machine accumulates a queue of length 36 at the end of the 1-minute run, while the speed-3 machine has 1 message in the queue occasionally and the speed-6 machine has no queued messages. However, in the run with machine speeds 3, 2, and 2, all three machines only have 1 message in their queues occasionally, with no messages queued rest of the time.

We think that when machine speeds are similar, the slower machines are able to process the messages from their faster peers timely, and thus can catch up with the faster peers. This results in a smaller drift and a shorter length of queue for the slower machines.

1. **Experiment with a smaller probability of internal events:**

We changed the range of random integers from 1-10 to 1-5, and thus the probability of an internal event is reduced from 7/10 to 2/5. This causes the machines to send messages more often, and below are our observations on how a smaller probability of internal events affects the size of jumps in logical clock times, the size of the drift, and the message queue length respectively.

* We observe that when the probability of internal events is smaller, the size of jumps in the logical clock time for slower machines can become smaller.. For instance, in the run with machine speeds 3, 5, and 6 and probability of internal events 7/10, the average sizes of jumps for the three machines are 2, 1.21, and 1.02. In the run with machine speeds 2, 4, and 4 and probability of internal events 2/5, the average sizes of jumps for the three machines are 1.97, 1.1, and 1.1. When the probability of sending messages are higher, messages are generated more often and thus there are more close-up updates in the communication.
* We also observe that, when the probability of internal events is small, the drift in the slower machines can become larger. For example, in the run with machine speeds 3, 5, and 6 and probability of internal events 7/10, the logical clock values after 1 minute are 355, 356, and 357, with very small drift. However, in the run with machine speeds 2, 4, and 4, and probability of internal events 2/5, the logical clock values after 1 minute are 234, 260, and 259, with a much larger drift.
* In addition, the length of the queue can become longer for slower machines when the probability of internal events is small. For example, with machine speeds 3, 5, and 6 and probability of internal events 7/10, the maximum queue length for the three machines are 2, 1, and 0 respectively. With machine speeds, 2, 4, and 4 and probability of internal events 2/5, the maximum queue length for the three machines are 11, 0, and 0 respectively. This means that the speed-2 machine could not process the messages it receives in time.

We think that when the probability of internal events is small, the faster machines send larger numbers of messages quickly and provide closer updates about their logical clock times to the slower machine. The slower machine thus receives a larger number of messages with closer updates of logical time values. However, since the slower machine is limited in its processing speed, and since each time it processes a message the update in the logical clock value is smaller, it can be harder for the slower machine to catch up. Also, the large number of messages from the faster peers can accumulate and lead to a longer queue of messages in the slower machine. With accumulated unprocessed messages, the slower machine can have a large drift in its logical clock time when compared to the faster machines.