COMPSCI 711 – Parallel and Distributed Computing

Assignment 1

Luis Rodriguez

UPI: lrdo555

Tested your programs on a lab machine.

I used the computer lab on the second floor of the computer science building to do my testing.

Instructions on how to run your programs.

I. Download the compressed folder.
II. Un compress and extract the folder using the right button option.
III. You can replace the delay document provided by the instructor to the same directory.
IV. Open the Developer Command Prompt for VS 2022 and use 'cd' to navigate to the directory containing the middleware folders and the network folder.
V. Execute the command 'run.bat' in the Developer Command Prompt for VS 2022
VI. The command should compile all the Middleware.cs files and the Network.cs creating executable files and automatically it should run those files.

Information is used to identify a multicast message.

The messages use a two part identifier the first part is 'Msg' number and the second is the middleware number it was sent from, the port number is just used as prompt for the user. Example: "Msg #1 from Middleware 5 8086". The message is processed by the middleware extracting the number after the '#' and the number after the word 'Middleware'

Total order multicast scenario

As a computer scientist a common scenario that uses total-order multicast are the ones using distributed database systems, specifically the ones that involve financial institutions. They need to handle transactions that come from a number of locations and ensure that the balances of the accounts can be calculated accurately constructing a proper transaction history. The system must ensure that the transactions are processed in an identical order across every node to avoid inconsistencies that could corrupt the data. The total-order multicast is used to ensure the delivery of every transaction to all of the nodes and that the order it arrives matches the same sequence, helping preserve data integrity in the distributed network.

Difference between total order and causal order multicast scenario

Let's pretend we are on a online multiplayer game and the following actions occur:

i. Player 1 picks up a key.
ii. Player 2 opens a door using the key.
iii. Player 3 enters the room through the door.

The actions need to be preserved as events in the game, and these events must synchronize through the computers from other players. In the case of causal order multicast it ensures that events are delivered in the order that uses a cause-effect relationship between actions.

Causal order multicast ensures that events are delivered in a way that respects the cause-effect relationships between actions. This is crucial in a game setting to maintain logical game progress. Picking up the key by Player 1 is event A. Then opening the door with the key by Player 2 is event B, making it causally related to event A because the door cannot be opened without the key being picked up first. After that entering the room by Player 3 is event C, which exclusively happens after event B. The system ensures that all players see event A before event B, and event B before event C, preserving the game logic. If player X performs an unrelated action (event D) it can be delivered at any point as long as the causal sequence of events A, B, and C are respected.

In total order multicast a global order are imposed on all events across the system, regardless of their causal relationships. Using the previous action events the system might establish a specific sequence for events, such as [A,D,B,C]. Every node in the system processes the events in this exact order: first, Player 1 picks up the key (A), then Player X an action (D) is processed, followed by Player 2 opening the door (B), and finally, Player 3 entering the room (C).

Causal order allows the game to process events in a way that makes logical sense based on the game's rules and the cause-effect relationships between player actions. This ensures that the game world remains consistent and understandable for players, but might allow for some flexibility in the ordering of unrelated events.

On the other hand, total order ensures that all events are seen in the same sequence by all players and the game's backend systems, which can be critical for certain types of game logic or for resolving conflicts in a highly interactive environment. This strict ordering guarantees consistency but might require more coordination and thus potentially introduce latency depending on network speeds.