CS 6378: Project I

Instructor: Ravi Prakash

Assigned on: September 20, 2015
Due date and time: October 8, 2015, 11:59 pm

每个进程的id：广播的
存在同个
space的file
里

This project has two parts:

1. Implement causally ordered broadcasting.
2. Implement causally ordered multicasting.

This is an individual project and you are expected to demonstrate its operation to the instructor and/or the TA. Your program should be written in C, C++ or JAVA. No other programming language will be accepted.

1 Requirements

initiation phase

1. There are ten nodes in the system, numbered from 0 to 9. Each node executes on a different machine. You can choose the machines from the machines set aside for network programming ($dc01$, $dc02$, $\ldots$, $dc45$). Each node must execute on a different machine.
2. There are reliable socket connections (TCP) between each pair of nodes. All messages are sent over these connections.
3. Each node repeatedly goes through the following sequence of operations until each node has sent one hundred messages. In the case of broadcast communication, sending one copy of the message to all the nodes counts as sending one message. In the case of multicast communication, sending one copy of the message to all the destination nodes counts as sending one message.

   100

   (a) Waits for a period of time that is uniformly distributed in the range [20, 100] milliseconds before sending a message.

   (b) In the case of causally ordered broadcast, identical messages are sent to all the nodes. In the case of causally ordered multicast perform the following operations:

      i. Randomly select an integer, x, that is in the range [1,9].
      ii. Randomly select x nodes (other than itself) as destinations for the multicast.
      iii. Send identical messages to the x selected destinations.

   The message contains physical clock value of the sending process, along with the causal dependency information you may need to send to ensure causally ordered delivery. The physical clock value can be obtained using an appropriate system call.

   (c) When a node receives a message, the node lets a period of time, t, to elapse before inspecting its vector/matrix to make delivery decision. The value of t is uniformly distributed in the range [50, 200] milliseconds and is used to mimic channel delays through a network of geographically distant nodes. However, you may encounter a situation where a later message sent along a channel is inspected before an earlier message sent along the same channel (TCP connection). If you have implemented the logic correctly then it will not be a problem because the later message can only be delivered after its causal predecessor, the earlier message sent along the same channel, is delivered.

4. Once a node has sent one hundred messages, it does not make any more attempt to send messages, and sends a *completion notification* to node 0.

5. Node 0 brings the entire distributed computation to an end once its has received *completion notification* from all the nodes, including itself.

2 Data Collection

For your implementation report the following:

1. The total number of messages exchanged.
2. The average latency of communication, and the standard deviation. For each message delivered at a node, latency is equal to the difference between the time (physical clock value) at the destination node when the message was delivered, and the clock value carried by the message.
3. The number of messages that had to be buffered prior to delivery, the maximum and mean time for which a message had to be buffered at its destination prior to delivery. Note that the time, t, mentioned above should not be included in the time for which messages were buffered. A message is buffered when its matrix/vector is inspected to make a delivery decision and it is determined that the message cannot be delivered immediately because at least one of its causal predecessors meant for the same destination has yet to be delivered.
4. The maximum number of messages buffered at each of the nodes during your experiment.
5. The time-averaged number of messages buffered by each node during your experiment.
6. The number of messages that did not have to be buffered at the destination prior to delivery.

Also, describe how you verified the correctness of your implementations.

3 Point Distribution

Implementation (50%): Source code of your well structured and well documented program. You may write your code in C, C++ or Java.

Correctness (50%): Output that your program produces and the statistical analysis of the results.

4 Submission Information

The submission should be through eLearning in the form of an archive consisting of:

1. File(s) containing the source code.
2. The `makefile`.
3. A sample output file.
4. A short report (text file) containing the analysis of the results in the output file, and information about the platform/machines on which you executed your program.

Please do not submit the executable file. Also, any code that is not written by you, but is obtained from some other source should be properly acknowledged in the source file, as well as in the report.

The algorithm for causally ordered multicasting and broadcasting must be implemented by you, and not copied from some other source.