

COMP 4704 – Spring 2009

HW#6

Instructor:

Prof. Matt Rutherford – mjr@cs.du.edu

Due by:

5pm on April May 21, 2009

Please choose **ONE** of following two options. The deliverables should be emailed to me at mjr@cs.du.edu before the due date.

Option #1: Logical Clock Middleware

The goal of this exercise is to implement a thin middleware layer that provides totally ordered message delivery, as discussed in class, by using Lamport's Logical Clocks. This middleware layer should be implemented in a class / module that runs in its own thread(s). It will be instantiated and started by the "application layer". The application layer can call the middleware through a "send" interface (method, function, etc) that accepts a **string** of data to send out to the other processes running the middleware. When the middleware needs to deliver a message to the application layer, it invokes a callback method, function, etc. that is specified by the application layer when it instantiates the middleware.

The focus of this assignment is on the middleware layer, but a simple "application-layer" must be implemented to demonstrate the functionality. The application layer, in this case, will be a simple program that instantiates and starts the middleware module and then reads line-delimited strings from standard input. These strings are passed off to the middleware to be multicast to the group of processes.

When the middleware delivers a string back to the application, it should just be printed to standard output.

Middleware Details

The middleware should accept parameters (passed in via constructors or equivalent) similar to the following:

- The name of a file containing a list of all of the middleware endpoints (in line-delimited `host:port` records) involved in the group.
- The `host:port` that it should listen on (must be one of the ones listed in the file).

The middleware will start with its internal logical clock initialized to zero. Communication with the other group members should be accomplished via UDP (datagrams). Initially, assume that localhost networking is reliable (but keep an eye out in case you are sending more datagrams than can be buffered and they are being dropped). Assume all group members are up and running. When sending, the middleware will tag each application message with the appropriate logical clock value and strip it off on the receiving end before delivering to the application layer.

NOTE: if you would like to play around with multicast (depends on your OS settings, but if you are just doing localhost, it should work), you can use this instead of UDP unicast to all of the group members. In this case, you just need to know the multicast address to use and the number of processes in the system, not the full list of endpoints.

Application Details

As described above, the application should simply instantiate the middleware module (passing command line arguments to it as necessary) and then read line-delimited messages from standard input that are to be delivered to the group.

Deliverables

- A ZIP file containing only source code, no binary or compiled files. This should include:
 - A `README.txt` file that describes the command-line interface to your program and how to compile / run if it is not obvious.
 - The source code for your program.
 - A simple test script / program that will start 5 instances of the program and run to completion.

Please try not to send elaborate directory structures with your code, if everything exists in the same directory (i.e., the top-level one) it makes it much easier to grade.

Option #2

Please answer the following questions.

Question 1

To achieve totally-ordered multicasting with Lamport logical clocks, is it strictly necessary that each message be acknowledged? Explain why or why not.

Question 2

The textbook makes the statement that “causally-ordered multicasting (i.e., from vector clocks) is weaker than totally-ordered multicasting (i.e., from logical clocks)”. Explain what they mean by this statement. In distributed systems why might it be desirable to use a synchronization means (i.e., vector clocks) that is “weaker”?

Deliverables

- A document containing detailed answers for each of the questions.