Compsci 711 / Softeng 710 Assignment 2 Report

Wasiq Kashkari

October 16, 2014

1 Introduction

This assignment required students to implement two algorithms in distributed systems that deal with multicast messages. These were the "causal order" and "total order" algorithms respectively.This repport describes some of the aspects of the implenetation of the solution.

2 Run Instructions

1. Open a command prompt and ensure the environment variables are set up such that a csc.exe file is located on the class files. Another option is to simply open a Visual Studio Developer Command Prompt.
2. In the command prompt navigate to the assignment folder with the command "cd /path/to/asg2/"
3. **To run Part 1:** run the commands "cd Part1/" and then "run.bat"
4. **To run Part 2:** run the commands "cd Part2/" and then "run.bat"

a

3 Testing

This application has been tested in the Computer science laboratory machines and functions as expected.

4 Identification Scheme

Each middleware is denoted by an integer id which uniquely identifies it. This number is assigned based on the TCP port at which the middleware listens for messages. The scheme used to assign the middleware numbers is as follows:

A Middleware is assigned the id x, where x = port number - 1081

Each message can be uniquely identified within the middleware that created it simply using a sequence number i.e. "Message #1". Furthermore, each message can be identified uniquely globally, from any middleware which has recieved it, through a combination of the sequence and middleware numbers contained within the message data i.e. "Message #1 from Middleware 1".

5 Scenarios

5.1 Part 1

The following scenario was presented in Part 1 to show causal order of the multicast messages. Middleware 4 and 5 are excluded in this example for the sake of simplicity. Max sleep time was set at 30000.

Messages Sent

```
Middleware 1
"Message #1 from Middleware 1:1 0 0 0 0"

Middleware 2
"Message #1 from Middleware 2:0 1 0 0 0"
```

Messages Received

```
Middleware 1
"Message #1 from Middleware 1:1 0 0 0 0" is Ready

Middleware 2
"Message #1 from Middleware 2:0 1 0 0 0" is Ready

Middleware 3
"Message #1 from Middleware 1:1 0 0 0 0" is Ready
```

Messages Sent

```
Middleware 3
"Message #1 from Middleware 3:1 0 1 0 0"
```

Messages Received

```
Middleware 1
"Message #1 from Middleware 3:1 0 1 0 0" is Ready

Middleware 2
"Message #1 from Middleware 3:1 0 1 0 0" is not Ready, waiting
```

for Messages
"Message #1 from Middleware 1:1 0 0 0 0" is Ready
"Message #1 from Middleware 3 :1 0 1 0 0" is now Ready After Receiving appropriate Messages

Middleware 3
"Message #1 from Middleware 3:1 0 1 0 0" is Ready
"Message #1 from Middleware 2:0 1 0 0 0" is Ready

Ready List

Middleware 1
"Message #1 from Middleware 1:1 0 0 0 0"
"Message #1 from Middleware 2:0 1 0 0 0"
"Message #1 from Middleware 3:1 0 1 0 0"

Middleware 2
"Message #1 from Middleware 2:0 1 0 0 0"
"Message #1 from Middleware 1:1 0 0 0 0"
"Message #1 from Middleware 3:1 0 1 0 0"

Middleware 3
"Message #1 from Middleware 1:1 0 0 0 0"
"Message #1 from Middleware 3:1 0 1 0 0"
"Message #1 from Middleware 2:0 1 0 0 0"

As is shown above, Middleware 2 waits for Message #1 from Middleware 1 before processing the message from Middleware 3. At the end, each of the ready lists are different, showing this is causal order rather than total ordering.

5.2 Part 2

The following scenario was presented in Part 2 to show causal order of the multicast messages. Middleware 4 and 5 are excluded in this example for the sake of simplicity. Max sleep time was set at 20000.

Messages Sent

Middleware 1
"Message #1 from Middleware 1"

”Message #2 from Middleware 1”

Middleware 1
”Message #1 from Middleware 2”
”Message #2 from Middleware 2”

Middleware 1
”Message #1 from Middleware 3”
”Message #2 from Middleware 3”

Ready List

Middleware 1
”Message #2 from Middleware 1”
”Message #1 from Middleware 1”
”Message #1 from Middleware 2”
”Message #2 from Middleware 3”
”Message #2 from Middleware 2”
”Message #1 from Middleware 3”

Middleware 2
”Message #2 from Middleware 1”
”Message #1 from Middleware 1”
”Message #1 from Middleware 2”
”Message #2 from Middleware 3”
”Message #2 from Middleware 2”
”Message #1 from Middleware 3”

Middleware 3
”Message #2 from Middleware 1”
”Message #1 from Middleware 1”
”Message #1 from Middleware 2”
”Message #2 from Middleware 3”
”Message #2 from Middleware 2”
”Message #1 from Middleware 3”

Here we can see the total ordering algorithm ensured that each message could be uniquely ordered across the whole system. Hence the messages were placed in the ready list in the same order for all middleware.