Assignment 2

Due date: 23:59 2021-10-15

The purpose of this assignment is to gain good understanding of the ordering property of multicast primitives through implementing the total order multicast primitive.
Many applications need to use multicast mechanisms that satisfy various ordering properties to send information. To simplify the task of application development, many systems provide multicast mechanisms using middleware or OS services. In this assignment, you are required to implement a middleware that ensures total order of multicast messages.

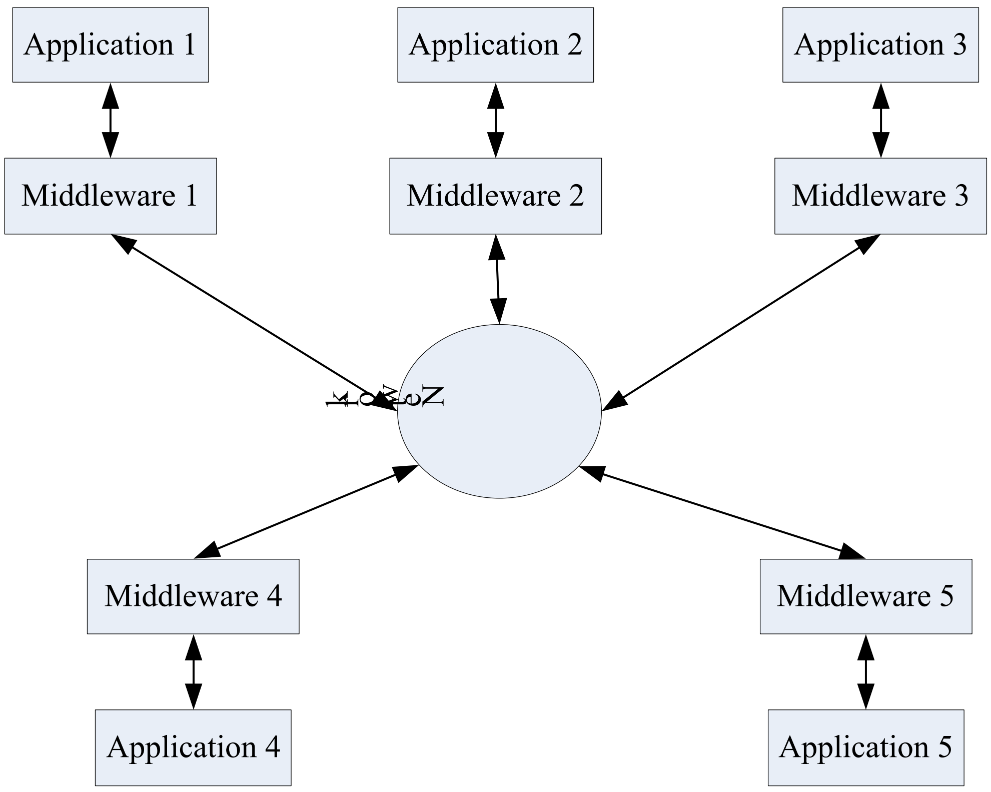


Figure 1 Multicasting through Middleware

Figure 1 shows how the middleware is used.

- Each application program has a corresponding middleware. An application and its corresponding middleware reside on the same machine.
- The middleware uses a protocol to guarantee the ordering property of the multicast messages.
- When an application needs to multicast a message to other applications, the application sends the message to its corresponding middleware. The middleware is responsible for multicasting the message to the corresponding middleware of the other applications through the network.
- When a middleware receives a multicast message from the network, it determines whether the message can be delivered to its corresponding application according to

the protocol for guaranteeing the message ordering. The middleware buffers a message until the message can be delivered to the application.

In this assignment, a C# program for simulating the network is provided[1]. The multicast messages sent by the middleware on behalf of the application programs MUST be sent to the network program first. When you use the provided program, apart from changing the value of constant *maxSleepTime* (explained later), you should not change any other part of the network program. The marker will use this network program when marking your assignment[2]. The network program and some simple programs for testing the network program can be downloaded from Canvas. The details of the network program are as below:

- The middleware communicates with the network program through TCP sockets.
- The network program receives multicast messages sent by the middleware and broadcasts each received message to ALL the middleware in the system[3]. The broadcast is carried out by point-to-point communication using TCP socket.
- To simulate network delay, while broadcasting a message, the network program waits for a random period before it sends the message to the middleware. The waiting time for different middleware might be different. For example, when broadcasting a message *m*, the network program might wait for 1000ms and 2000ms before sending *m* to Middleware 1 and 2 respectively.
- In the network program, constant *maxSleepTime* is used to specify the maximum waiting time before a message is sent to the middleware.
- In order to use the network program, each multicast message MUST end with a five-character sequence "<EOM>".
- Port 8081 is used by the network program for receiving multicast messages from the middleware.

In this assignment, you are required to implement programs corresponding to Middleware 1 to 5 in Figure 1. You do not need to provide any of the application programs shown in Figure 1. You can use either C# or Java to do this assignment.

Programming (16 marks)

In this part, you are required to implement the middleware that ensures the total order of the multicast messages. In order to test the middleware, your implementation must include a GUI interface for the middleware. The layout of the interface for each middleware should be similar to Figure 2.

[1] To run the program on Mac, you need to install .NET https://dotnet.microsoft.com/download.
[2] The marker might set a different value to constant *maxSleepTime* in the network program.
[3] The original sender of a message will also receive a copy of its message in a broadcast.

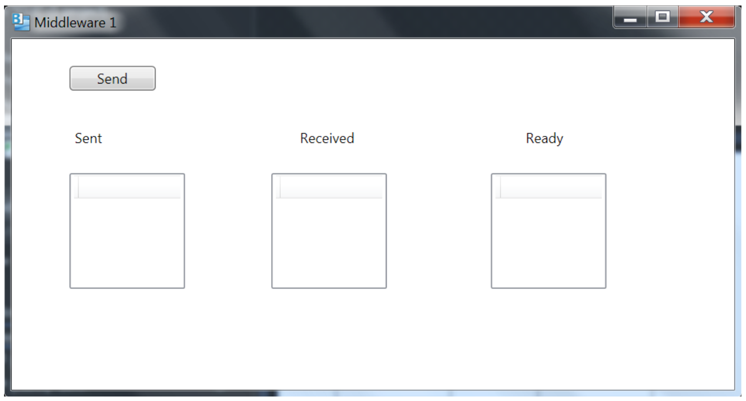


Figure 2 The GUI for a Middleware

The middleware and its GUI should satisfy the requirements below:

- Each middleware should have its own GUI.
- A GUI should have a title indicating the ID of the middleware, e.g., "Middleware 1".
- Each GUI should have a button "Send". When the button is clicked, the middleware should send a multicast message to the network. The contents of the message should be generated automatically by your program. Each multicast message sent by a middleware must include information that uniquely identifies the message. For example, "Msg #1 from Middleware 1 xxx <EOM>" indicates that this is the first message sent by Middleware 1[4].
- Each GUI should have three lists for displaying messages.
    - The "Sent" list shows the multicast messages that have been sent by the middleware to the network. The messages in the list should appear in the order in which they were sent to the network.
    - The "Received" list contains the multicast messages that have been received by the middleware from the network. The received messages should be displayed in the order in which they are received by the middleware.
    - The "Ready" list presents the multicast messages that are ready to be delivered to the application program. This means that the messages in the "Ready" list should be a subset of the messages in the "Received" list. The order in which the messages appear in the "Ready" list should conform to the total-order property of the multicast messages.
- Your implementation should have five middleware programs. Each middleware program should be stored in its own folder. There is NO need to run your programs concurrently on different machines. You only need to test them on one machine.
- Each middleware program should create a TCP socket for receiving messages from the network program. The sockets used by the five middleware programs MUST use ports 8082, 8083, 8084, 8085, and 8086 respectively.
- The middleware programs might need to exchange some "control" information amongst them when determining the order in which the multicast messages should be

[4] #1 means the first message. In this case, "Msg #1 from Middleware 1" uniquely identifies the message. "xxx" denotes other information in the message, e.g. timestamps. <EOM> must be used to indicate the end of a message.

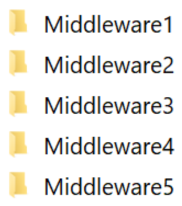
delivered. The middleware programs are allowed to exchange the control information between them directly using TCP sockets.

Report (1 mark)

Your report should include instructions on how to run your programs. Save your report as a PDF file. Name it as *Report.pdf*.

Submission

- You only need to submit the implementations of the middleware. Your implementations should have the following directory hierarchy.



Middleware1
Middleware2
Middleware3
Middleware4
Middleware5

- You should only submit the source code of your programs.
- You should create a batch file for compiling the submitted programs and for executing the programs. Name the batch files as run.bat.
- Zip your implementations, the batch file and the report into file upi.zip where upi is your university login ID, e.g., jbon007.zip. Submit the file through https://adb.auckland.ac.nz/. The directory hierarchy of your implementation should be preserved when you zip your files.
- **NO** email submission will be accepted.

Marking Schedule

Programming

1. The batch file for compiling and executing programs works correctly. [1 mark]
2. There are five middleware programs, and the programs are placed in the right folders. [1 mark]
3. Each GUI shows the ID of the middleware. [1 mark]
4. Each multicast message contains the information that uniquely identifies the message. [1 mark]
5. Multicast messages are added to the “Sent” list correctly when the “Send” button is clicked. [2 marks]
6. Multicast messages are added to the “Received” list correctly. [3 marks]
7. Multicast messages are added to the “Ready” list correctly. [7 marks]

Report

Clear instructions on running the programs. [1 mark]