Distributed System Design

COMP 6231: Winter 2019

Instructor: Rajagopalan Jayakumar
Department of Computer Science & Software Engineering
Concordia University, Montreal, QC

Distributed Library Management System (DLMS)

Documentation Submission Date: 15/03/2019
Project Submission Date: 05/04/2019

Submitted By:

Md. Hasibul Huq – 40087646

Jaignesh Varadharaju – 40081862

Hao Lei – 40056875

Introduction:

The distributed library management system is aims to connect a group of libraries. This system is used by two type of users. Those who manage library are managers and those who uses the library facilities like borrowing books are user. In this system managers can perform few actions like:

1. Add Item
2. Remove Item / Decrease the number of items
3. List of the items

On the other hand, Users have some functionality like:

1. Borrow Item
2. Return Item
3. Find Item
4. Exchange Item

In this project 3 servers are used, and the name of the servers are

1. Concordia Server
2. McGill Server
3. Montreal Server

And there will be three replicas of the same servers in this project as a replica and each replica contains a Replica managers

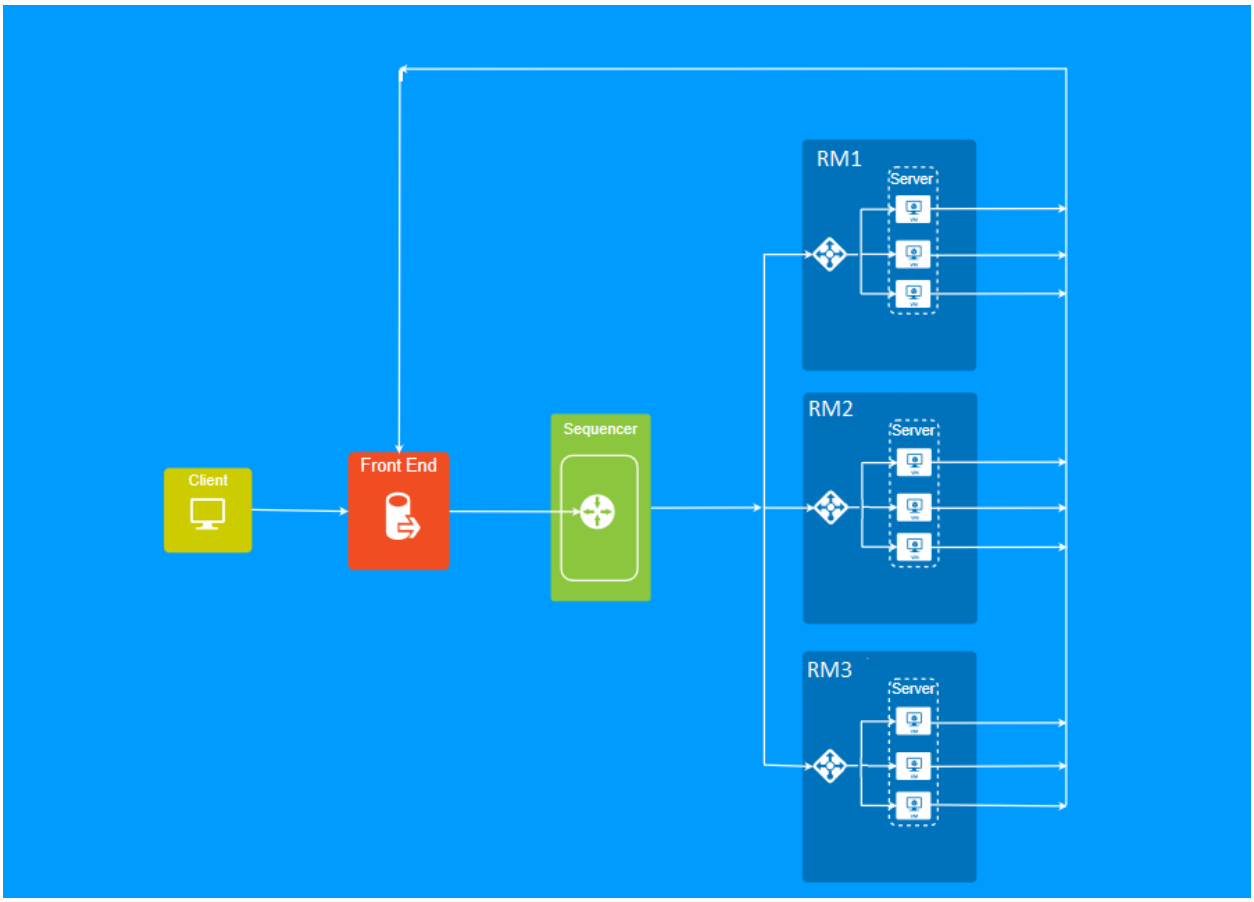
This document has the details of high availability of distributed system for library management system. This document is describing the how high availability is achieved in the implementation of this DLMS using CORBA. To achieve a highly available system total order implemented through FIFO. It also has the detail explanation of all the client, Front End, Sequencer and Replica manager. It also discusses about the data structure used for implementation and what type of problem we are going to have to adopt the whole replicas.

System Design:

Highly Available CORBA Distributed Library Management System will be implemented over the CORBA project implemented as part of assignment2. We will implement a Front-End module which would accept the Client Request through CORBA. And Front End will send the request to the sequencer and the

sequencer send the messages with a sequence id replica manager. This is multicasting of each request from sequencer to the group of RMs. It requires totally ordered reliable multicast so that all RMs perform the same operations in the same order. And then RMs will process each request identically and send it to the corresponding servers. Specific servers are going to execute that request and going to reply to the frontend. The communication between Front End and the sequencer will be established through UDP IP connection. From the Sequencer to RMs multicast UDP IP connection. Communication between RMs and replica will be established by the UDP IP connection. Replica and frontend is going to communicate with each other by reliable UDP connection.

The design model is given bellow.



Picture: Model of a DLMS achieved by Active Replication

Here, the CORBA implementation is between client and front end instead of client and server. CORBA object and the implementation will be in the front end. But instead of logic it will call the UDP connection of sequencer and add one sequence id and send it to each RMs through Multicast UDP/IP from the sequencer. Then all the RMs are going to call the procedure from the server and will execute the function and will reply. And front end will receive the replies and correct result will be sent to the client.

Active Replication:

How the active replication works is explained bellow and the information is taken from the slides.

1. **Request**: Front End attaches a unique id and uses totally ordered reliable multicast to send request to RMs.
2. **Coordination**: The multicast delivers requests to all the RMs in the same order.
3. **Execution**: Every RM executes the request. They are state machines and receive requests in the same order, so the effects are identical. The id is put in the response
4. **Agreement**: No agreement is required because all RMs execute the same operations in the same order, due to the properties of the totally ordered multicast.
5. **Response**: FEs collect responses from RMs. FE may just use one or more responses. If it is only trying to tolerate crash failures, it gives the client the first response.

Reliable Multicast:

Multicast communication requires coordination and agreement. The aim is for members of a group to receive copies of messages sent to the group. Many different delivery guarantees are possible. A process can multicast by the use of a single operation instead of a send to each member.

On initialization
Received := {};

For process p to R-multicast message m to group g
B-multicast(g, m); // $p \in g$ is included as a destination

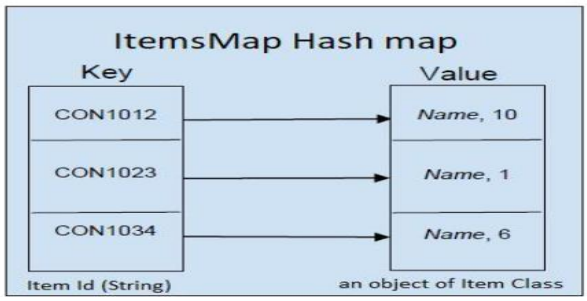
On B-deliver(m) *at process q with g = group*(m)
if ($m \notin$ *Received*)
then
Received := *Received* $\cup$ {m};
if ($q \neq p$) *then B-multicast*(g, m); *end if*
R-deliver m;
end if

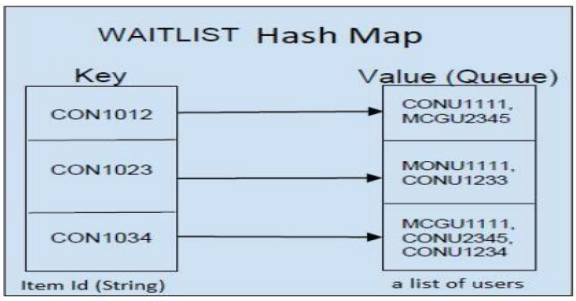
Total Order:

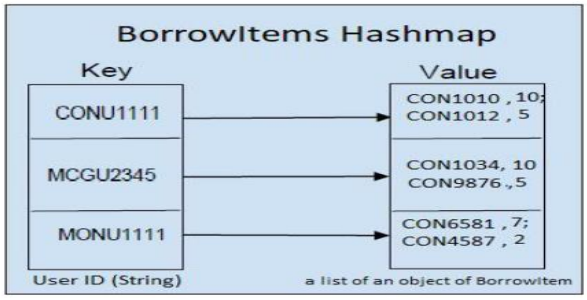
The general approach is to attach totally ordered identifiers to multicast messages. Each receiving process makes ordering decisions based on the identifiers like the FIFO algorithm, but processes keep group specific sequence numbers. Operations TO-multicast and TO-deliver.

Data Structure:

To store data three different hash maps are used in each server. One hash map is containing item name and quantity with a unique item key which is named itemId here.







To store borrowed item list, another hash map is used where user id is unique and the list of items with number of days are stored. If the item is not available during borrowing user can wait. To keep this record here waitingList hash map is used. As the replicas are developed by different people, the implementation and variables with data type may defers from each other. But the return value is same.

Test Scenario:

Sce. Id	Requirement Name	Test Scenario	Test Cases
1	Login	User name	1. Validate with valid user name 2. Validate with invalid username 3. Validate access with username 4. Validate server access with Username
2	Manager Facility	Menu Selection	1. Validate menu selection
3	Manager Facility	Add Item	1. Validate Item name 2. Validate Item id with valid id 3. Validate Item id with an invalid id 4. Validate item Quantity 5. Either it will add a new item or will increase the existing id
4	Manager Facility	Remove Item	1. Validate Item Id with valid id 2. Validate item id with invalid id 3. Validate Item Quantity 4. It will decrease item quantity or remove it completely
5	Manager Facility	List Item	1. It will show the items available in the server
6	Manager Felicity	Logout	1. It will logout the user and will ask for username to login
7	User Facility	Borrow Item	1. Validate Item Id with valid item id 2. Validate Item Id with invalid item id 3. If valid the item will be borrowed 4. Validate with an Item id which is not available 5. Validate with item id which is available in different server 6. Validate one user can not borrow multiple item from Different server
8	User Facility	Return Item	1. Validate item id with valid item id. 2. Validate with item id which is available in different server
9	User Facility	Find Item	1. Validate with Item name which is available in one of the servers or in any of them
10	User Facility	Exchange Item	Enter a new Item Id and old item id 1. to exchange the book.
11	User Facility	Logout	1. It will logout the user and will ask for username to login

Test Cases:

1. Login as CONM1111 and add Item Item Id CON9876 Name: DS and quantity 2 (Item Added)
2. Login as MONM1111 and add item Id MON9876 Name DS and quantity 1 (Item added)
3. Login as MONM1111 and add item Id MON9877 Name APP and quantity 1 (Item added)
4. Login as CONU1234 and Borrow item CON9876. (item Borrowed) and exchange CON9876 with MON9876. (Item Exchanged).
5. Login as MCGU1234 and Try to Borrow MON9876 (asked for waiting list. Yes. Added in waiting List)
6. Login as CONU1234 return item MON9876. (Item returned and allocated to MCGU1234)
7. Login as MCGU1234 and try to exchange MON9876 with MON9877 (Exchanged)
8. Login as CONM1234 and show the list item. It will return 2 correct result and one wrong result. Try 3 times. In the 4th time all the servers will give the correct result
9. Login as CONM0000 and show the list item. It will crash the Concordia server in RM one. and the server will be restarted.

The Front End is taken care by Hao Lei – 40056875, sequencer is developed by Jaignesh Varadharaju – 40081862 and the replica managers are created by Md. Hasibul Huq – 40087646 Though the Specific parts of the projects is done by as above mentioned. But all of us worked on every portion of the project.

Most of the details are taken from the project description and slides from the lectures. Some information's and concept are taken from the StackOverflow and different places of internet.