Distributed System Design

COMP 6231 – Winter 2022

Instructor: Rajagopalan Jayakumar

Design Documentation

Software Failure Tolerant and/or Highly Available

Distributed Appointment Management System (DAMS)

Submitted By:

Harshal Modi (40195060)

Mahavir Patel (40198619)

Raviraj Savaliya (40200503)

Overview

This Distributed Appointment Management System (DAMS) is for a health care sector. This system is used by hospitals. It connects different hospitals from different cities like Montreal, Sherbrooke and Quebec. This system's aim is to connect hospitals from different cities and help user to book and manage their appointments in any of the hospital.

There are two types of user in this system: Patient & Admin.

Patient can book their appointments through his/her own hospital server and also manage them. The tasks that are performed by patient is given below.

- Book an Appointment
- Get Appointments Schedule
- Cancel an Appointment
- Swap an Appointment

In an addition to book an appointment for patient. Patient can also book their appointment in other cities. Ex. Montreal patient can book their appointments in Quebec and Sherbrooke too.

Also the patient can also swap his registered appointment to on new different slot in the of the same appointment type in other cities also.

Other User is hospital Admin. It's main task are to mange the appointment slot for the patient. The tasks that are performed by the admin are:

- Add an Appointment
- Remove an Appointment
- List Appointment Availability

Admin can also perform the patient's operations like admin can also book or cancel the appointments. When an admin invokes List

Appointment Availability function he/she will get all the list of appointments given by appointments type.

There are three types of appointment type that admin can add depending on the availability of doctors. Types are *Physician, Surgeon* and *Dental.* There are three times slots available for each appointment type in a day : Morning(M), Afternoon(A) and Evening(E). Every appointment type has a unique appointment ID which consists of hospital acronym, time slot and date.

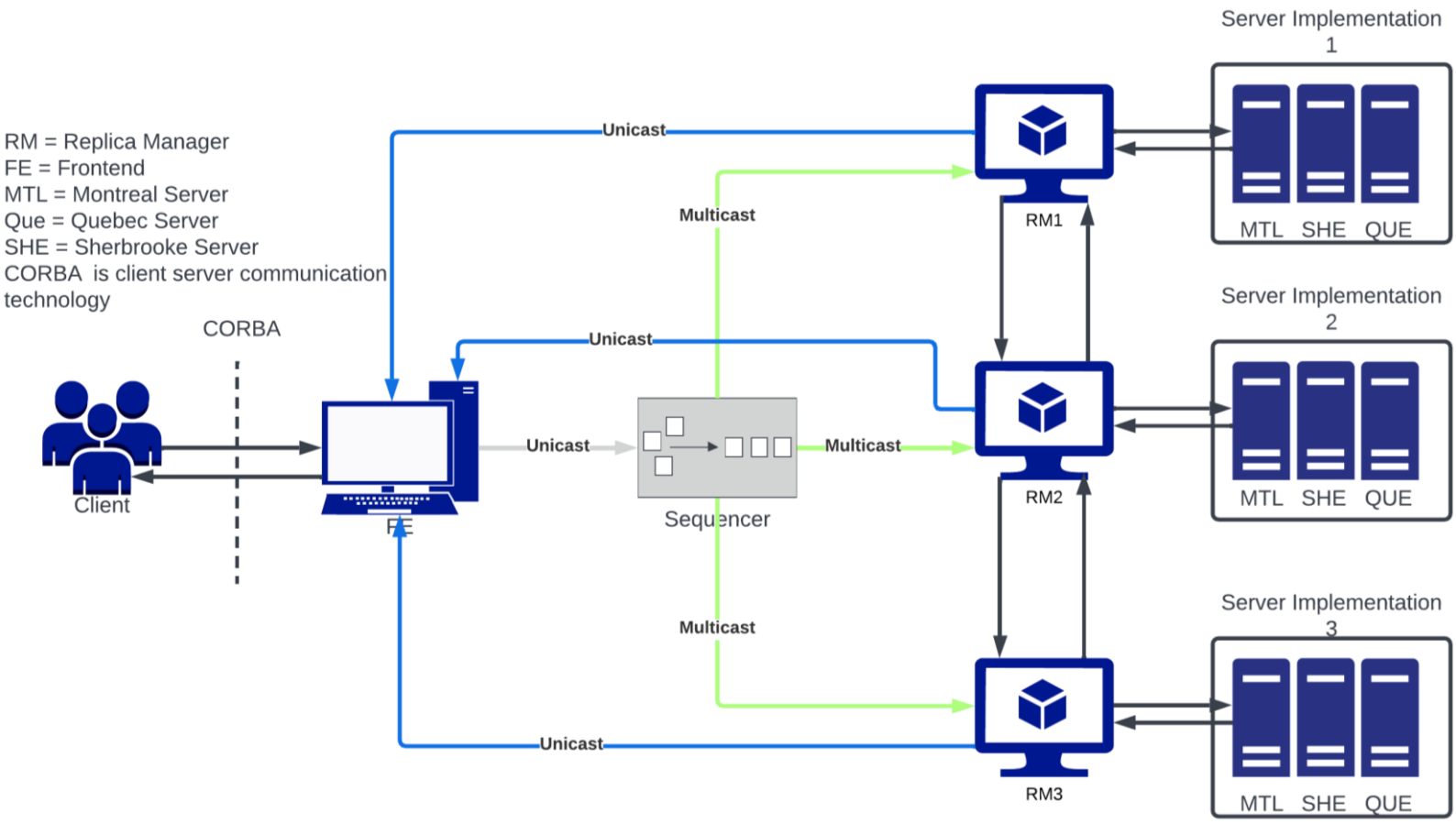
For example if Montreal hospital has a availability for physician on 9th Feb,2022 in the morning then appointment id for that would be : MTLM090222 where MTL denotes Montreal hospital, M for morning slots and 090222 represents date in 'ddmmyy' format.

This System Uses CORBA concepts of JAVA. CORBA stands for Common Object Request Broker Architecture. It is a standard defined by the Object Management Group (OMG) that enables software components written in multiple computer languages and running on multiple computer to work together.

CORBA is a standard for distributing objects across networks so that operations on these objects can be called remotely. CORBA is not associated with a particular programming language, and any language with a CORBA binding can be used to call and implement CORBA objects. Objects are described in a syntax called Interface Definition Language (IDL).

System Architecture

We will implement the CORBA technology for this project from the Assignment 2. We will implement client that send the request to Front-End through CORBA. After that Front-End will send the request to the Sequencer using UDP IP unicast.



In this project we are going to use Active Replication strategy for the fault tolerance. This system consists of Frontend, Sequencer and Replica Manager (RM). Each RM has its own replicas of server which consists of Montreal, Quebec and Sherbrooke Server.

When Client want to book an appointment he will communicate with the frontend using the CORBA technology. The frontend will now send this request to the sequencer. The sequencer will assign the

unique sequence number to each request that it will receive. After assigning the sequence id it will forward that request and id to each replica manager using totally ordered reliable multicast protocol. At the RM, RM will process the request and send to the corresponding server in the same order that client has initiated, using the sequence id.

Server will execute the request and it will send results back to the frontend.

Here all the communication between Frontend and Client will be done using CORBA technology. Sequencer and RM will communicate using the totally ordered reliable Multicast. And other all communication will be done using the UDP IP protocol.

Active Replication:-

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

- **Request** : Front End attaches a unique id and uses totally ordered reliable multicast to send request to RMs.
- **Coordination** : The multicast delivers requests to all the RMs in the same order.
- **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.
- **Agreement** : No agreement is required because all RMs execute the same operations in the same order, due to properties of the totally ordered multicast.
- **Response** : FEs collect responses from RMs ,FE may just use one or more responses. It is the 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 single operation instead of a send to each member.

Algorithm for reliable multicast:

On initialization

Received:={};

For process p to R-multicast message m to group g

B-multicast(g, m); // p ∈ g is included as a destination

On B-deliver(m) at process q with g = group(m)

if(m ∉ Received)

then

Received := Received ∪ {m};

if(q ≠ p) then B-multicasting(g , m);end if

R-deliver m;

end if

Total Order :

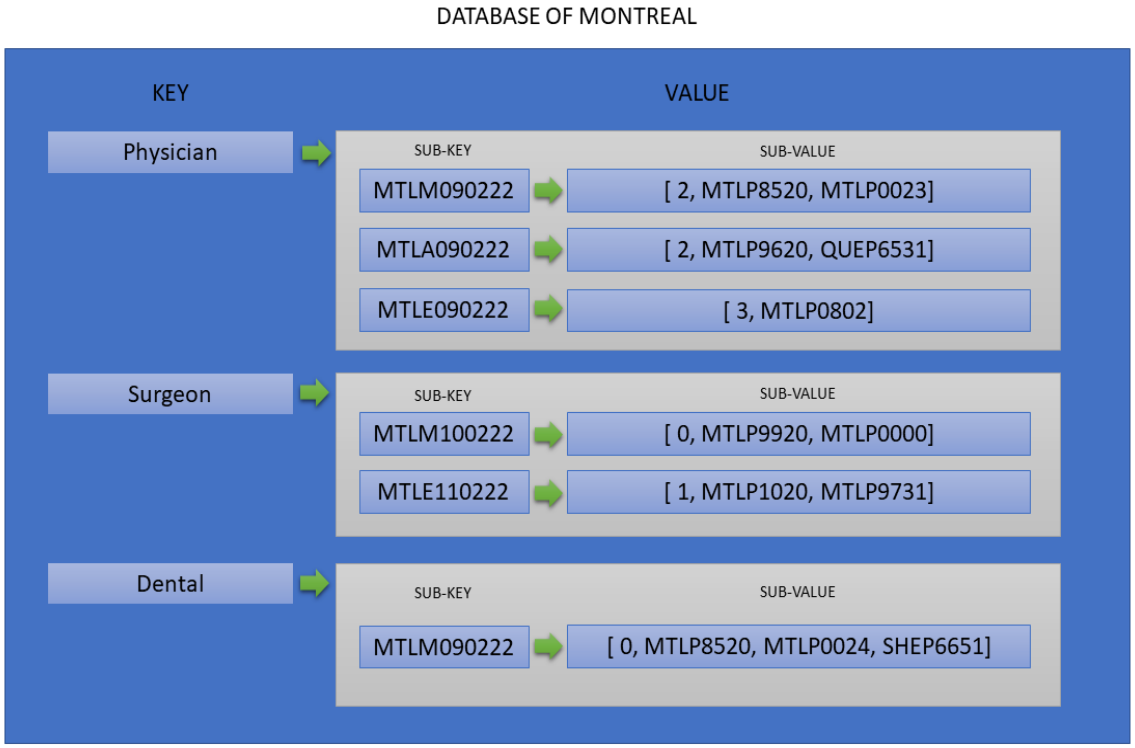
The basic multicast algorithm delivers messages to processes in an arbitrary order. To execute clients request as he/she has initiated through front end we are going to use the Total ordering multicast.

If a correct process delivers message m before it delivers m', then any other correct process that delivers m' will deliver m before m'.

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

We will implement the total ordered multicast using a sequencer.

Data Structure :



Each server contains its own database which consists of HashMap. This database includes appointment type and associated appointment id. Every appointment id contains data like capacity and patient id which had booked that appointment.

Form user id system will assign the server by getting prefix of user id. Each server has specified one admin only he can perform the operations on that server and each admin requires password to perform the task.

This table represents the database on Montreal Hospital. It consists of HashMap where appointment type as a key and its value

contains sub HashMap. In the sub HashMap the key are appointment ids and values contains array details like capacity and patient is who has booked that appointment. The first element of array is always denotes the capacity available of the appointment. The remaining element represents the patient ids who has booked the appointment.

Test Cases :

1. Login as a Admin (id: MTLA0000) and provide password “password” : passed (Admin logged in montreal hospital)
2. Add two appointment for Physician and Dental and the appointments ids are MTLM120322 and MTLA120322 : passed (Appointments added successfully)
3. Book an appointment for the patient (id: MTLP4562) for dental type on the montreal hospital on the day 12th March,2022 in the afternoon : passed (Appointment booked Successfully)
4. Logout the system and again login as a admin (id: SHEA0000) with password “password” : passed (Admin logged in Sherbrooke hospital)
5. Add the appointment for dental in the morning on the day 14th March,2022 (Appointment id: SHEM140322) : passed (Appointment added successfully)
6. Now Logout as admin and login in the system as a patient (id: MTLP4562) : passed (Shows the welcome message)
7. Get the schedule of the appointments it will return the appointment for dental and id: MTLA120322 : passed
8. Patient swap the appointment of dental type from old appoinrment id: MTLA120322 to Sherbrooke hospital appointment id: SHEM140322 to dental type appointment : passed (Appointment has successully swapped from old appointment id to new one)
9. Get the schedule of the appointments it will return the appointment for dental and id: SHEM140322 : passed

Test Scenario

No.	Method	Scenario	Test Cases	Status
1.	User Login (User)	Admin/Patient Client	○ verify the id of user, is it valid (Like length of 8) ○ verify the user if admin or patient ○ If user is admin then check is it valid admin. If yes then get the password and verify it. ○ Next determine the server port using the prefix of user id.	PASS
2.	Menu Task (User)	Admin/Patient Client	○ Verify the user. Based on user provide the proper task menu.	PASS
3.	Add Appointment (Admin)	Admin Client	○ Get the details like appointment id, type and capacity. ○ If the appointment already exist then stop overriding the details.	PASS
4.	Remove Appointment (Admin)	Admin Client	○ Get the appointment Id and type which has to be removed. ○ Verify the type and id exist or not ○ If the they do not exist then do nothing ○ If they exist then check the appointments. If appointments are already booked by patient then reschedule that patient appointment on the next day ○ Remove the appointment	PASS

5.	List Appointment Availability (Admin)	Admin Client	○ Get the appointment from the user then list the appointment id and number of spaces available in it. ○ Perform the inter server communication through UDP/IP to get the list of id and spaces available in other server	PASS
6.	Book Appointment (User)	Patient/Admin Client	○ Get the patient id and appointment id and type ○ Check the appointment type and is exist or not, if not user can't book the appointment ○ If appointment id exist check the spaces are available if not patient can't book the appointment ○ If appointment has space then book the appointment for the patient and decrease the space.	PASS
7.	Get Appointment Schedule (User)	Patient/Admin Client	○ Get the patient id ○ Provide the appointments list that patient has booked in his/her city. ○ Communicate with other server to Using UDP/IP to get appointment list that patient has booked in other hospitals.	PASS
8.	Cancel Appointment (User)	Patient/Admin Client	○ Get the patient and appointment id ○ Check appointment exist or not ○ If appointment does not exist then do nothing ○ If appointment exists then check the patient had booked it or not.	PASS

			○ If the patient has booked that appointment then remove the patient from that appointment ○ IF the patient has not booked that appointment then patient can't cancel that appointment	
9.	Swap Appointment	Patient/Admin Client	○ Get the patient id, get the information of the appointment that user want to swap like, old appointment id and type and new appointment id and type (on which user to swap the appointment). ○ Check the new appointment id and type exists if exists then book the appointment on the given id else cancel the operation. ○ Check the old appointment id and type are booked by the patient. If it is booked by the patient then cancel the appointment else cancel the operation. ○ If any of the above operation failed then cancel the whole swap operation. Else perform the swap operation for the patient.	PASS

Tasks:

- **Mahavir Patel (40198619) : Front End (FE)**

  Design and implement the front end (FE) which receives a client, forwards the request to the sequencer, receives the results from the replicas and sends a single correct result back to the client as soon as possible. The FE also informs all the RMs of a possibly failed replica that produced incorrect result.

- **Harshal Modi (40195060) : Sequencer**

  Design and implement a failure-free sequencer which receives a client request from the FE, assigns a unique sequence number to the request and reliably multicast the request with the sequence number and FE information to all the three server replicas.

- **Raviraj Savaliya (40200503) : Replication Manager (RM)**

  Design and implement the replica manager (RM) which creates and initializes the actively replicated server subsystem. The RM also implements the failure detection and recovery for both types of failures