UNIVERSITÉ Concordia UNIVERSITY

COMP6231 Project

Software Failure Tolerant or Highly Available Distributed Appointment Management System

Instructor: R. Jayakumar

TA: Brijesh Lakkad

Darshak Kachchhi (40206619)

Mansi Lakhani (40197791)

Vikyath Srinivasulu (40218245)

Table of Contents

1. Overview

The Distributed Appointment Management System (DAMS) is a distributed system for health care; It is used by an admin of the hospitals who manages the information about the medical appointments and patients to book or cancel a medical appointment across three different hospitals Montreal (MTL), Sherbrooke (SHE), and Quebec (QUE) within a system.

Hospital admins and patients are uniquely identified by the admin id (e.g., MTLA0000) and patient id (QUEP2981) respectively. There are 3 types of appointment types for which slots can be created by the admin: Physician, Surgeon, Dental. There are three-time slots are available for each appointment type in a day. Each appointment type is a combination of city, appointment slot, and date.

Each server maintains its database using the HashMap. Client and Server are communicating using the CORBA. While inter-server communication is done by the UDP communication. Each server maintains a log file for all the operations performed by the server. Also, for each patient and admin client log file is maintained.

This is a distributed system which tolerates a single software (non-malicious Byzantine) failure and is highly available under a single process crash failure using active replication. It actively replicates DAMS server system with four replicas each running a different implementation in different hosts on the network. It also has a front end (FE) which receives a client request and forwards it to a failure-free sequencer. The entire server system (replicas, sequencer, FE, and RM) runs on a local area network, the server replicas, sequencer, FE, and RM communicate among themselves using the unreliable UDP protocol.

G. Tools

- Java IDE Eclipse
- Java JDK version 1.8

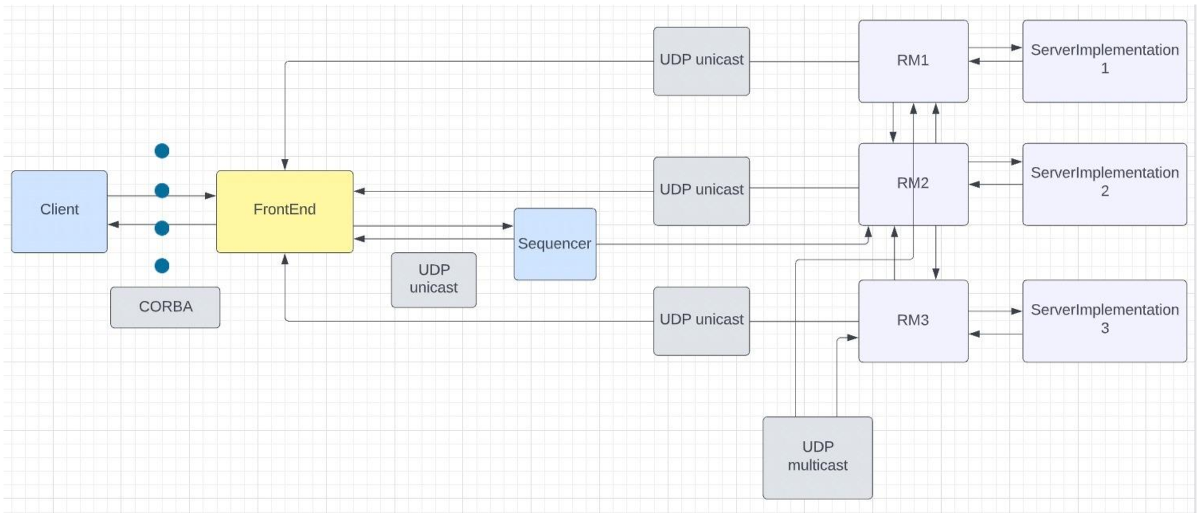
2. Architecture

There will be three replica managers (RM) each containing different server implementations of our four servers and database of that replica.

Highly available or fault tolerant Distributed Appointment Management System will be implemented using CORBA as an extension of assignment 2.

- Front-End (FE) implementation includes receiving client request using CORBA invocation and then forwards the request to the Sequencer. This request forwarding is done using UDP Unicast. It receives response from all RM's and sends a single response back to client. The FE also informs all the RMs of a possibly failed replica that produced incorrect result.
- Replica Manager creates and initializes the actively replicated server subsystem. The RM also implements the failure detection and recovery for both types of failures.
- Failure-free Sequencer 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 four server replicas.
- A test case suite which will test all the possible failures and a client program which will run these test cases.

The flow of the project is described below:



The Front-End(FE) receives request from the client request(CORBA invocation). FE will send the request to the sequencer using UDP unicast and the sequencer with the agreed sequence number multicasts the sequence id to all the replica managers using UDP multicast. Replica managers process each request identically and sends to server implementation to execute it using UDP unicast. On processing it replies to Front-End with the result.

To achieve High availability or fault tolerance in the system we will implement using below criteria:

- To achieve high availability, we will make sure that whenever a server crash is detected by Front-End we can recover the it by initiating the recovery by replica manager
- To achieve fault tolerance Front-End will send a request to all the three Replica Managers and it will collect the response from it. Using the difference in the response from all the replica manager Front-End will detect the fault tolerance.
- As UDP is not a reliable protocol, we use timeout resend technique to detect server crashes. The FE sends 3 requests before it receives any reply from RM, if it still does not get a reply, it detects it as server crash.
- Total ordering is achieved using sequence number attached to every request. FIFO will be used to manage the order of processing. If the request gets lost the RM will ask other Replica managers for the lost request. This check is based on the criteria if the received sequence number is subsequent to the last executed request.

Task Assignment:

- Mansi Lakhani: Front-End and Replica 1
- Darshak Kachchhi: Replica Managers and Replica 2
- Vickyath Srinivasulu: Failure-Free Sequencer and Replica 3

3. Test Case

Already Added Test Data

Server Name	Appointment Type	Appointment ID	Patient List
Montreal	Physician	MTLA030222	**MTLP2345, QUEP5465**
		MTLE030222	**MTLP1245, MTLP2463, MTLP9875**
		MTLM010222	**MTLP2345, MTLP9875, MTLP3246**
	Dental	**MTLM030222**	**MTLP2345, MTLP3246**
		MTLE030222	**MTLP2345**
		MTLA010222	**MTLP3246, MTLP1245, MTLP2463, MTLP5465**
		MTLA020222	**MTLP2345, MTLP5465**
	Surgeon	**MTLA030222**	**MTLP1245, MTLP2463**
		MTLM030222	**MTLP3246, MTLP9875**
Quebec	**Physician**	**QUEA040222**	**MTLP2345, QUEP5465**
	Dental	**QUEA010222**	**QUEP5465**
		QUEA020222	**QUEP5465**
Sherbrooke	**Physician**	**SHEE080222**	**MTLP2345, SHEP5565, SHEP2475**
	Dental	**SHEA050222**	**SHEP5565, SHEP2475**
	Surgeon	**SHEE070222**	**MTLP2345, SHEP5565, SHEP2475**

Test Data

Test Method	Expected Output
Add appointment	**Success: Appointment Added**
	Failed: Cannot book appointment Id of another server
Book Appointment	**Failed: Patient has already book appointment in the same day with same Appointment Type**

	Failed: No appointment available for selected slot
	Failed: Patient has already booked 3 appointments other than its server
	Success: appointment successfully booked
Remove appointment	**Success: Appointment is removed**
	Success: Appointment is removed with not available next appointment slot
	Success: Appointment is removed with patient is transferred
	Failed: No slot available
List Appointment	**Success: All Appointment list**
Get Appointment Schedule	**Success: Empty appointment schedule**
	Success: Appointment Schedule
Cancel Appointment	**Success: cancelled appointment**
	Failed: No record of appointment found
Swap Appointment	**Success: Appointment swapped**
Swap Appointment	**Failed: No appointment booked of old Appointment ID**
	Failed: No appointment available for new appointment Id