Distributed System Design
COMP 6231: Summer 2019

Instructor: Sukhjinder K. Narula

Project: Software Failure Tolerant or Highly Available

CORBA Distributed Event Management System

Submitted By:

1. Mehul Prajapati (40076930)

2. Darshan Dhananjay (40079241)

3. Jayaprakash Kumar (40083709)

1. Overview

The main objective of this project is to make use of our CORBA Distributed Event Management System implementation and make it to be software failure tolerant and/or highly available under process crash failure using the replica of our project.

The Distributed Event Management System consists of three servers (Montreal, Ottawa, Toronto) and one client.The distributed system will be used by manager and customer to perform various roles and functionalities in three different cities.

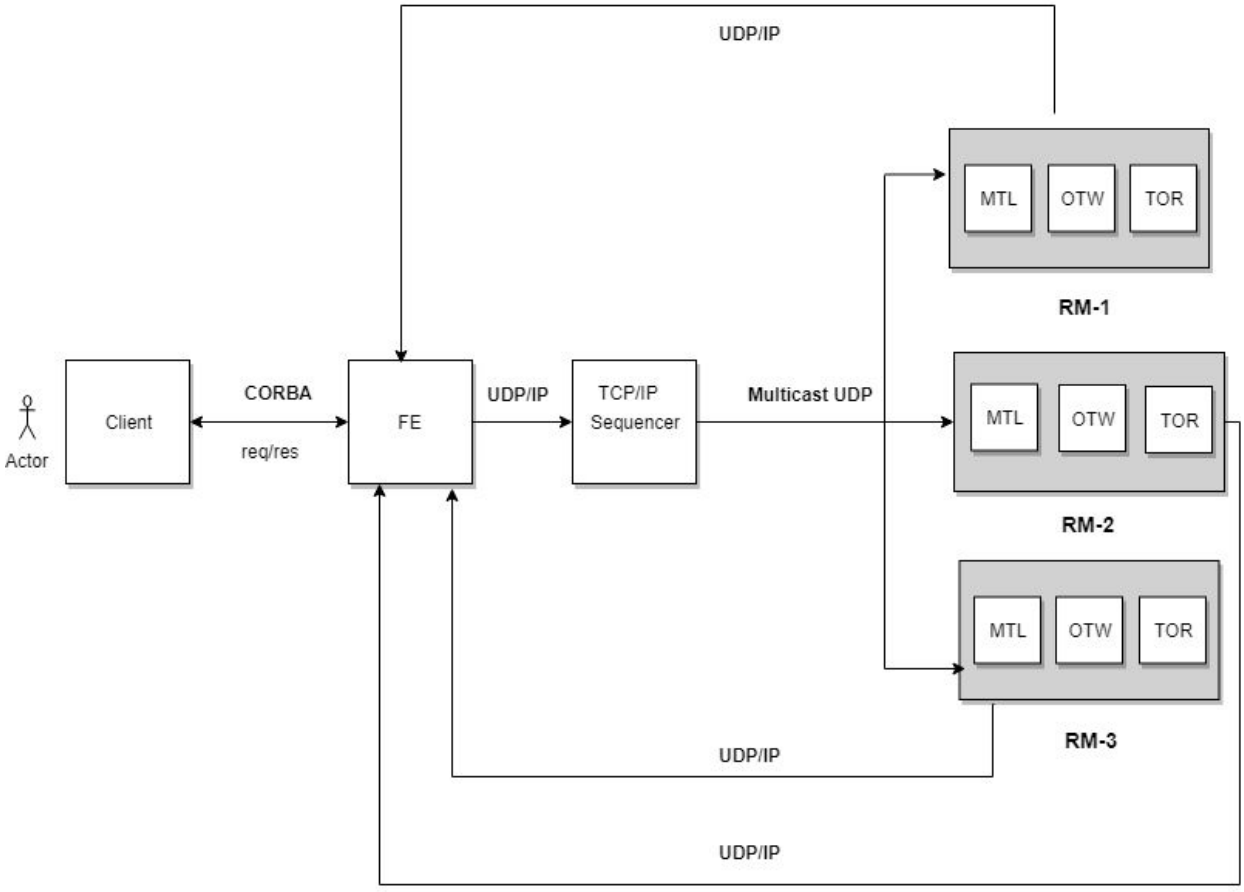
In the distributed system based on customer/Manger logged in, the authorised request send to the respective server via Front End(FE) through CORBA call. Then the FE will make the UDP call to the sequencer. Also FE responsible for checks the majority of responses from the RM which helps to maintain the DEMS highly available under process crash failure and software failure. Then the sequencer will make Multicast UDP to all the Replication Manager(RM). The RM processing the request and send back the response to the FE. Also, RM responsible for data synchronisation, server bug and crash implementation. Finally, FE send back the response to the client.

The system is implemented in such a way that client wouldn't know the location of the server where they interacted and produce accurate response even in software failure and process crash.Overall it maintains the system highly available in both scenarios.

To experiment software failure, manually creating bug in the server of replica 1 that will detected by FE and the process crash is implemented in replica 2.

The methods and functionalities for Customer/Manager are mentioned below.

Architecture Diagram:



- UDP - User Datagram Protocol
- TCP - Transmission Control Protocol
- RM - Replica Manager
- FE - Front End
- IP - Internet Protocol

1.1 Manager Roles

- **addEvent:**
    - Event manager can add event in his city using this method. When the event is created, a success message is displayed and logs are being recorded in server and userlog. If event manager tries to add the same event, then booking capacity gets updated.

- **removeEvent:**
    - Event manager can remove event in his city using this method. When event does not exist and if event manager tries to remove the event, no action is performed. Success and failure response will be recorded in logs.

- **listEventAvailability:**
    - Event manager can use this method to get all events which are created across all 3 cities. This method used UDP server communication to get response from other servers. All response messages will be recorded in logs.

1.2 Customer Roles

- **Book Event:**
    - Customers can book event in with respect to the event-Id that is available in his city or other cities. He can only book only 3 events outside his city. Once the event is booked a success message is displayed and logs are being recorded in server and userlog.

- **Get Booking Schedule:**
    - In this functionality customer will be able to see all his booking from all the cities.If not appropriate messages is shown .The logs are being recorded accordingly.

- **Cancel Event:**
    - In this functionality customers will be able to cancel the event .If the event is not booked by the user and wants to cancel an error message is displayed .

- **Swap Event:**
    - In this functionality customers will be able to swap the old event with the new one. If the event was not booked by the user previously and wants to swap the event, an error message is displayed .

2. Description

This System fundamentally is intended to clarify the whole usefulness of the Java Java IDL (CORBA). The following are a few points that ought to be known.

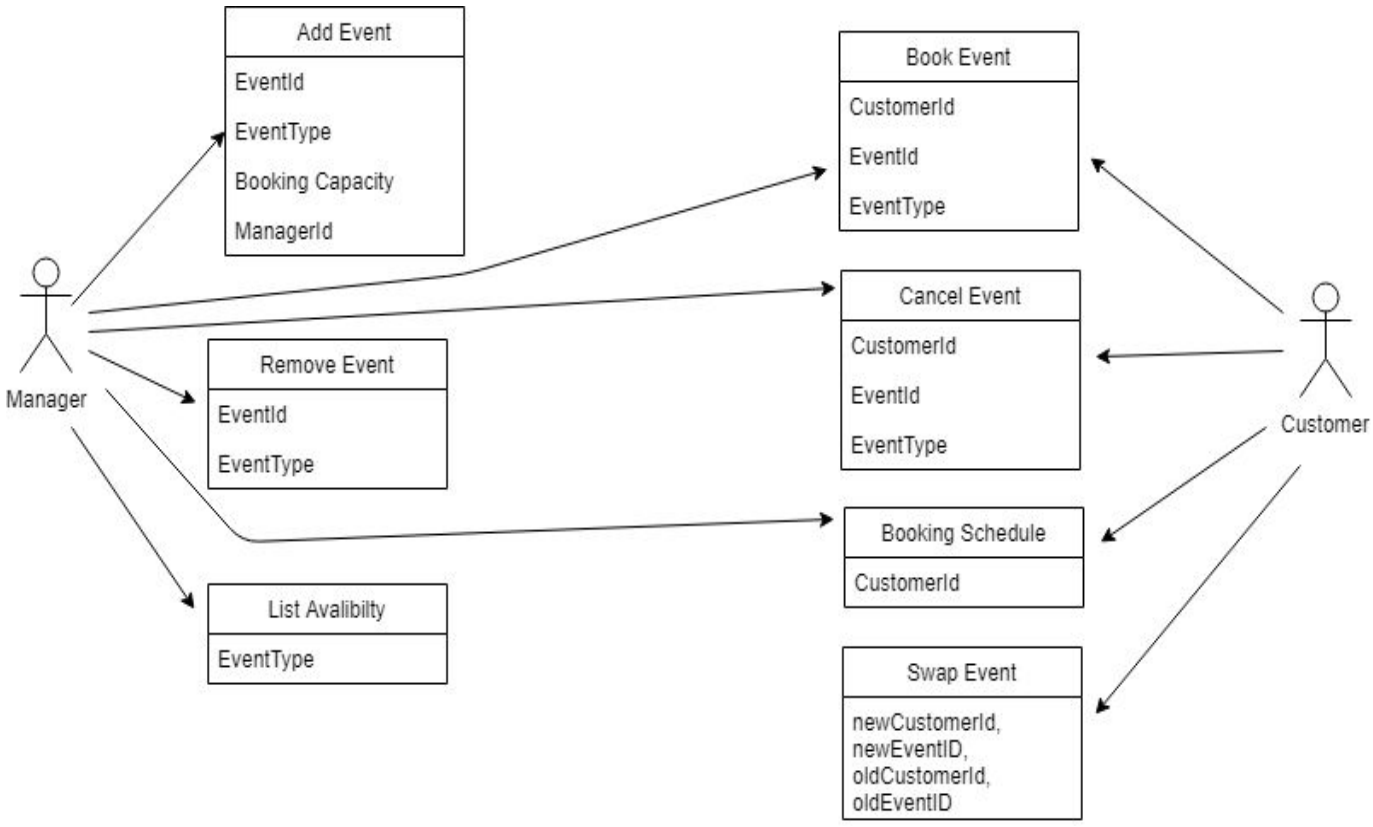
Definition - What does the Common Object Request Broker Architecture (CORBA) mean?

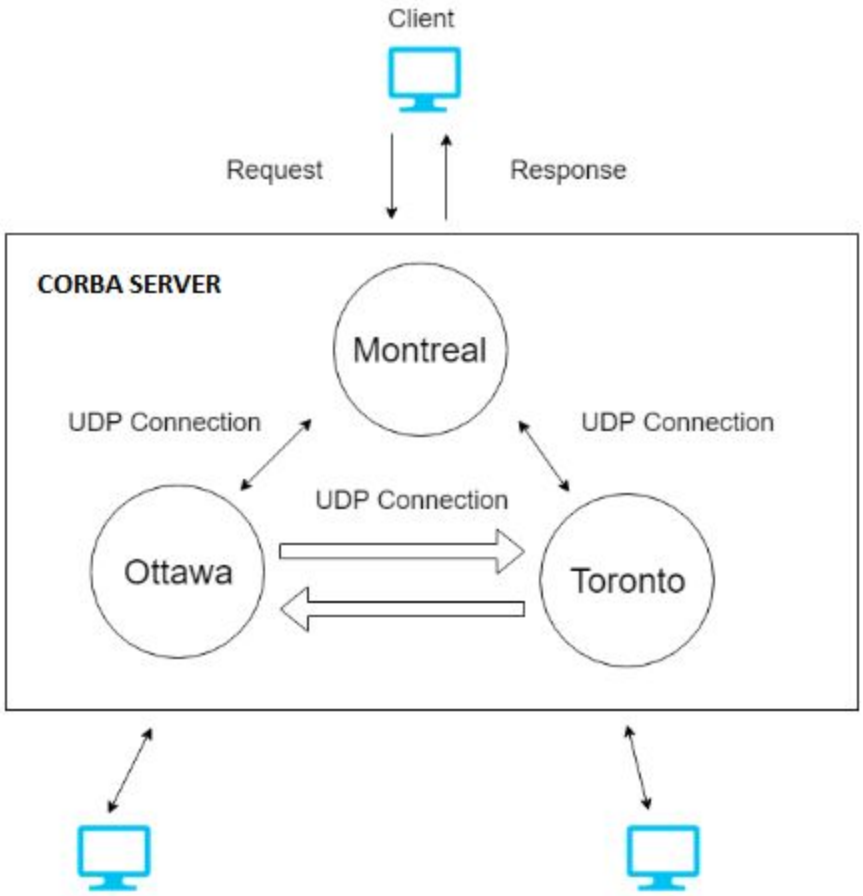
CORBA enables collaboration between systems on different operating systems, programming languages, and computing hardware. CORBA uses an object-oriented model although the systems that use the CORBA do not have to be object-oriented. CORBA is an example of the distributed object paradigm.

Basics of a CORBA Application

- Firstly we create the interfaces in IDL language and compile that IDL file using Java IDL compiler to generate all other necessary files like Stub, Skeleton, Helpers, POA etc. Then we implement the interfaces in other file. After that in server code, we need to initialize the ORB and using naming service, bind the object with the appropriate name.
- The client program will also initialize the ORB and get the object reference using the naming service. After that, the client can invoke methods on a remote object.

3.Manager and customer Role:





4. Test Cases

No.	Test Description	Expected output	Actual output
1	Customer Provides user Id if it is not correct Ex:TORC1234WW	Invalid user	Invalid user
2	Once Customer logs in successfully. Customer can perform book an event providing CustomerId, eventId,EventType and if the event is available it is booked Ex:MONC1234 TORM123456 c	You successfully booked the event	You successfully booked the event

3	When Customer performs wrong eventId Ex:MONM123444444	It shows Invalid Message	It shows Invalid Message
4	Customer can cancel event that he has booked by providing eventType and eventId Ex:TORM123456 c	Event has been cancelled	Event has been cancelled
5	If customer Provide wrong event Id to cancel.	No such event	No such event.
6	Customer can see his bookings by eventType Ex: c for conferences	Events List	Events List
7	If No events booked by the customer and he wants to see his booking	No events Booked by Customer	No events Booked by Customer
8	Customer can book events from other cities as well .But with a restriction of maximum 3.	Event Booked Successfully	Event Booked Successfully
9	If booked More than 3	Maximum amount reached	Maximum amount Reached
10	If customer is booking an event with 0 capacity	No Event Found	No Event Found
11	Customer can cancel only those events which are booked by him	Event Cancelled	Event Cancelled
12	Customer cannot cancel other events which he has not booked	Logged in server logs Invalid Event	Invalid Event
13	Add a new event from any server with all correct parameters. Username: TORM1234 TORA200519 c 3	Event should be created and log messages should get updated	Same as Expected

14	Add a new event of different city. Username: TORM1234 MONA200519 c 3	Event should not be created, and log messages should show the reason	Same as Expected
15	Add same event with a different booking capacity Username: TORM1234 TORA200519 c 4	Event should be created with new booking capacity.	Same as Expected
16	Add a new event with invalid event type. Username: TORM1234 TORA200519 x 4	Event should not be created, and log messages should show the reason	Same as Expected
17	Add a new event with invalid event id. Username: TORM1234 TORA20899 c 4	Event should not be created, and log messages should show the reason	Same as Expected
18	Remove event which was added previously Username: TORM1234 TORA200519 c 4	Event should be removed from database	Same as Expected
19	Remove event which was not added previously Username: TORM1234 TORA210519 c 4	Logs should record the reason why event is not removed.	Same as Expected
20	Remove event with invalid event type Username: TORM1234 TORA210519 x	Event should not be created, and log messages should show the reason	Same as expected
22	Remove event with invalid event id Username: TORM1234 TORA2519 c	Event should not be created, and log messages should show the reason	Same as expected
23	List events after creating more then events Username: TORM1234 c	It should show all events from 3 servers	Same as expected
24	List events when there are no	It should display no	Same as expected

	events created for an event type Username: TORM1234 x	events.	
25	List events with invalid event id Username: TORM1234 x	Event should not be displayed, and log messages should show the reason	Same as expected
26	List events when booking is exhausted	It should show event with 0 as booking capacity	Same as expected
27	Customer Wants to Swap Events with the new Event that is present in the Server	The event has been Swapped Successfully	Same as expected
28	Customer wants to swap events with the events that are not present or the events which is not booked by him	The event swapping failed	Same as expected
28	Crash event		

5. References

- https://docs.oracle.com/javase/8/docs/technotes/guides/idl/corba.html
- https://community.particle.io/t/multicast-udp-tutorial/19900