Assignment - 2

Student ID : 40230634
Name : Mayurkumar Kanbhai Jodhani
Subject : COMP 6231 Distributed Systems Design

Overview

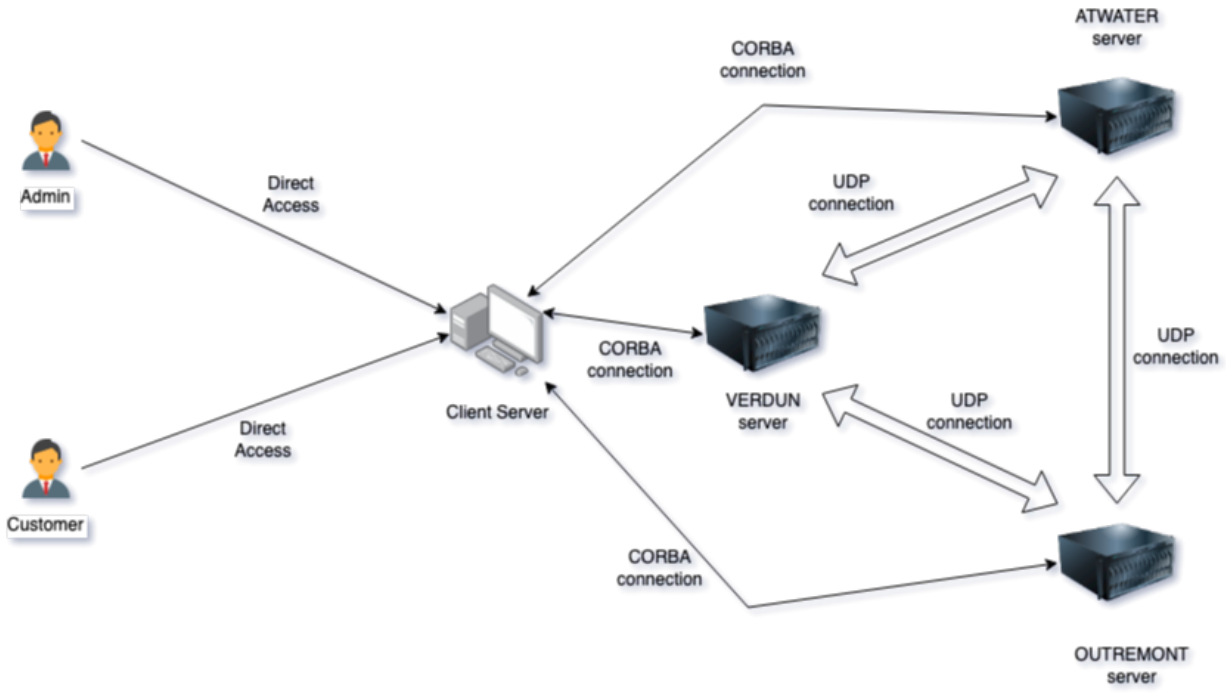
The given task involved implementing a CORBA (Common Object Request Broker Architecture) service to replace the existing RMI (Remote Method Invocation) service that was being used for communication between the client and server. By using CORBA, we were able to take advantage of its inherent multithreading capabilities, which allowed for increased scalability and efficiency in the system.

However, setting up the CORBA service and establishing connections to remote objects proved to be the most challenging part of the implementation process. To overcome this hurdle, we used the idlj tool to generate classes such as AdminPOA and CustomerPOA. These classes were then used to extend the functionality of the CORBA service and make it more robust.

CORBA is a middleware technology that provides a platform-independent, language-neutral approach to distributed computing. It enables components written in different languages and running on different platforms to interact seamlessly with each other. The CORBA architecture consists of an object request broker (ORB), which acts as a middleware layer between the client and server, and a set of object adapters, which provide a bridge between the ORB and the underlying object implementation.

One of the key advantages of using CORBA over RMI is its support for multithreading. In a multithreaded system, multiple threads can execute concurrently, which can greatly improve performance and efficiency. With CORBA, multithreading is built into the architecture, which means that we didn't have to worry about implementing it ourselves.

Architecture



The current system allows for various users, including administrators and customers, to directly access the client server in order to perform a range of operations. In addition to the previously available actions, a new feature has been introduced that enables users to exchange their tickets. To facilitate these actions, the system has shifted from using the RMI service to the CORBA service.

With the addition of the ticket exchange feature, customers are now able to modify their booked tickets for the same movie slot or select different movies altogether. This functionality is designed to provide greater flexibility and convenience for users.

The client server is able to connect to three different servers using the CORBA service that is provided by the associated server. If a user wishes to perform any actions on the associated server, only a CORBA connection is required to fulfill their needs.

In cases where a user needs to perform an action on a server that is not associated with them, such as a customer from Atwater wanting to book a movie ticket for a showing in Verdun, a UDP connection is required. This is necessary to ensure that the user is able to access the required information from the relevant server.

Overall, this updated system provides users with more options and greater ease of access. By utilizing the CORBA service and implementing different types of connections, the system is able to efficiently handle a range of user requests and enable seamless interaction between various servers.

Implementation

DMTBS.idl

This file is to represent the interfaces provided by CORBA service.By defining all the interfaces and the definition of different methods for operations perform by remote server.By using **idlj** tool to compile the IDL(Interface Definition Language) file, it will generates all the necessary class and interfaces which can be used to implement CORBA service.

AdminPOA.java

This class has been generated by **idlj** by compiling DMTBS.idl.By extending this class to AdminImpl, all the services are implemented for admin such as adding slot, removing slot and showing all the available seats for a given movie.

CustomerPOA.java

This class has been generated by **idlj** by compiling DMTBS.idl.By extending this class to CustomerImpl, all the services are implemented for customers such as adding slot, removing slot and showing all the available seats for a given movie.

AdminImpl.java

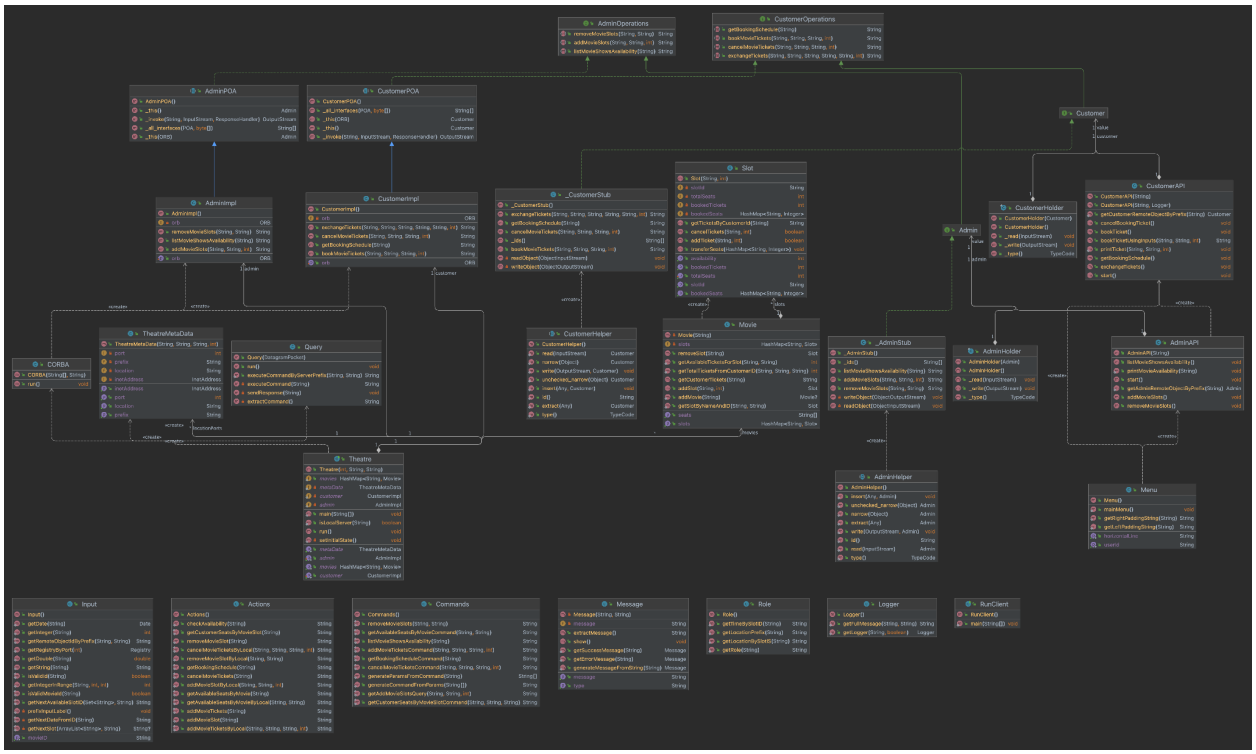
By extending AdminPOA class CORBA can make the object available for this class and remote server can invoke method of given objects using CORBA service.By accessing a object of this class, client can perform different operations such as add a slot, remove a slot and get available tickets for a movie.

CustomerImpl.java

This class is an extension of CustomerPOA which is used to make CORBA service up and running. By getting access to this object, clients can perform various operations such as book a ticket, cancel a ticket, exchange tickets, get a booking schedule.

Other classes such as Input, Message, Role, Logger and Commands are the same as the first assignment, only to provide basic functionality such as logging, taking input from users, etc.

Class Diagram



Test Cases

Along with the initial task, a new function has been introduced to the list of actions available for customers, which allows them to exchange their tickets. This added feature comes with two potential scenarios for ticket exchange, as described below. In other words, customers now have the option to exchange their tickets in addition to the previous actions that were available to them, and there are two ways in which this can be done, which are outlined in the following sections.

Customer Use Cases

Base Scenario	Specification	Result
Exchange movie ticket	if exchange ticket is possible, as the second is available to be exchange	It will update the booked ticket associated with the customer and also remote the entry at a given location.
	if something went wrong during exchanging tickets	It will show an error message with a proper message with the associated reason. It will also revert back the operation.

Summary

The process of setting up the CORBA service and making the necessary connections for remote object access can be quite complex. This is where the idlj tool comes in handy. This tool generates Java classes that provide a mapping between the IDL (Interface Definition Language) and Java programming language. By using these generated classes, we were able to extend the functionality of the CORBA service and make it easier to use.

The AdminPOA and CustomerPOA classes that were generated by idlj provide a way to implement the business logic of the system in a more structured and organized way. These classes define the methods that can be invoked on the remote objects and provide a way to handle any exceptions that may occur during method invocation.

In summary, the implementation of the CORBA service involved replacing the existing RMI service with a more robust and efficient middleware technology. Although the setup process was complex, the use of the idlj tool and the AdminPOA and CustomerPOA classes made it easier to extend the functionality of the service and implement the business logic of the system in a more structured and organized way. The use of CORBA's built-in multithreading capabilities also helped to improve the scalability and efficiency of the system.