Distributed System Design

COMP 6231: Winter 2023

Instructor: R. Jayakumar

Distributed Movie Ticket Booking System (DMTBS) using web services

Submission Date: 13/03/2023

By: Omer Sayem

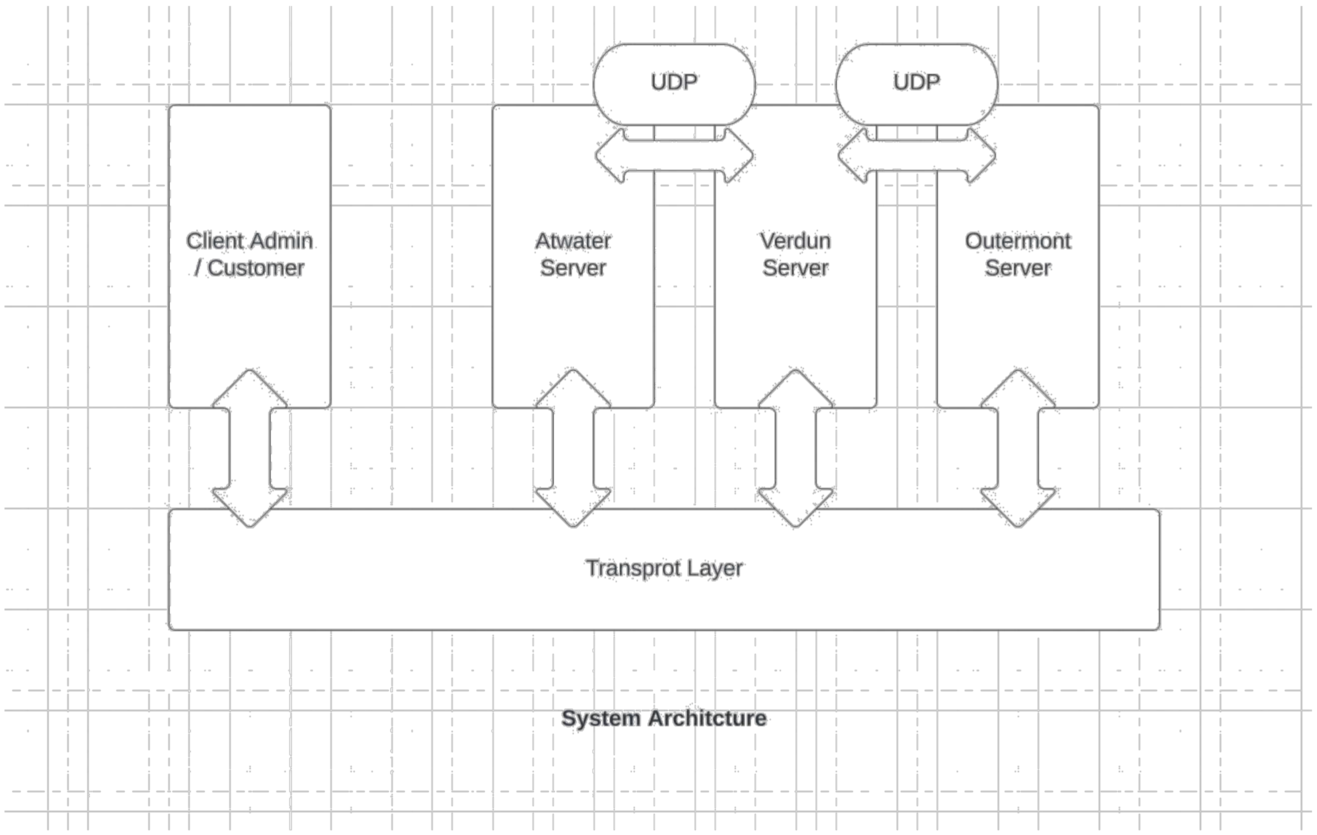
ID – 40226505

Overview: The Distributed Movie Ticket Booking System (DMTBS) is a robust and efficient platform for theatre managers and customers to manage information about ongoing and upcoming shows in different theatres. The system comprises three theatres in three different areas, Atwater (ATW), Verdun (VER), and Outremont (OUT), and is designed to cater to the needs of both admins and customers.

Admins and customers are identified by unique adminID and customerID, respectively, which are constructed from the acronym of their area and a 4-digit number. Admins are responsible for creating slots for movies with movie type and booking capacity, while customers can book and cancel movie tickets across different theatres. There are three movies available for booking, Avatar, Avengers, and Titanic, and customers can book a movie ticket if it's available, i.e., if the movie shows are not yet full on a particular date. The server that receives the request maintains a booking count for every customer and a customer can book multiple tickets for the same show, but only if the capacity is not full.

Admins have the ability to add, remove, and list movie that shows availability, and their operations are logged into their log files. Adding a movie requires the admin to provide the movieID, movieName, and bookingCapacity while removing a movie requires the admin to provide the movieID and movieName. The listMovieShowsAvailability operation enables the admin to find out the number of tickets available for each movie shown in all servers, for a given movie name. This operation requires inter-server communication, which is done using UDP/IP messages.

In addition to these operations, customers can also book and cancel movie tickets. They can book as many movie tickets in their own area, but only at most 3 movies from other areas overall in a week. Each server maintains a log file containing the history of all operations performed on that server, providing valuable information about what operations were performed, at what time, and who performed them.



System Architcture

Overall, DMTBS is a highly scalable and secure system that provides a seamless movie booking experience to both theatre managers and customers. The system ensures that customers can only book movie tickets if they are available, while also allowing them to cancel their bookings as required. At the same time, it provides a robust platform for theatre managers to manage their movie shows, enabling them to add, remove, and list movie shows availability.

Web service technologies in this implementation:

1. **SOAP-based communication between server and client**:

SOAP is a messaging protocol used for exchanging structured information in web services across computer networks. It relies on application layer protocols such as HTTP or SMTP for message negotiation and transmission and uses XML for its message format. By using SOAP, services can call processes on different operating systems and communicate using XML, regardless of the language or platform used. This allows for greater interoperability between different systems since web protocols like HTTP are widely available on most operating systems.

2. UDP/IP network programming between servers:

UDP/IP socket programming is a popular method for implementing web services that require fast and efficient communication. Unlike other transport protocols, UDP provides low overhead and reduced latency, making it ideal for applications that require real-time data transfer. UDP socket programming communicates using user datagram packets (UDP), which are sent between clients and servers without the need for a dedicated connection. This allows for greater flexibility in the development of web services, as UDP can be used to build applications for a wide range of network configurations and architectures.

3. `ExecutorService` class to create a thread pool and submit the `Runnable` tasks to it for better concurrency:

This approach can be more efficient since it reuses threads from the thread pool instead of creating new threads every time. The `ExecutorService` is created with a fixed thread pool size of 3. The `ServerInstanceRunnable` class is used to encapsulate the server instance creation logic and takes the server name and command line arguments as constructor parameters. The `run` method of the `ServerInstanceRunnable` class is executed when the task is submitted to the executor. The executor is then shut down after all tasks are completed.

The given code implements a method for exchanging movie tickets in a distributed system. The method takes input such as customer ID, old and new movie IDs and names, and the number of tickets to be exchanged. The code initializes boolean variables and strings and extracts the necessary information from the input. Next, the code checks whether the old movie ID contains "OUT" or "VER". If it does, it sends a message to the respective server (OuterMontreal or Verdun) to handle the exchange. If the old movie ID doesn't contain either, the code checks if the old movie is available on the local server. If it is, it then checks if the new movie is available and if the customer hasn't already borrowed it. If the new movie is available and not already borrowed, the code books the new movie and cancels the old movie. If the new movie is not available or has already been borrowed, the code returns an appropriate message. The method logs the results of the exchange operation in a log file, but the code does not show the implementation details of the logging mechanism. Similarly, the code does not show how the method sends messages to other servers, so it is not possible to evaluate the correctness and efficiency of the message-passing mechanism.