**COSC-5301\_01 Adv. Operating System**

**Project Report**

**on**

**Data Replication Management System**

**for**

**Meeting Room Scheduler**

**Instructor - Dr. Lawrence Osborne**

**Project Members:**

**Li Lin((L20320514)**

**Balram Agrawal(L20365062)**

**Shubhashish Mishra(L20365063)**

**Date of Submission: 05/04/2015**

# 1. Introduction:

## 1.1 Data Replication

Data replication is an age-old technique for tolerating faults, increasing system availability, and reducing latency in data access with improved response time. Most of the properties required from a distributed system depend on the replication of data. Without replication, the loss of a server hosting a unique copy of some data item results in unrecoverable damages. Similarly, replication also brings other advantages, including the ability to distribute read/write operations for improved scalability. However, such improvement is achieved at the expense of having to manage replication by implementing replica control protocols which raises the following intricate issues in the face of system failures:

* **Performance:**

Writing several copies of an item takes more time, which may affect the throughput of the system.

* **Consistency:**

Consistency is the ability of a system to behave as if the transaction of each user always runs in isolation from other transactions, and never fails. Consider for instance a transaction on an e-commerce site. There is a “basket” which is progressively filled with bought items. At the end the user is directed to a secure payment interface. Such a transaction involves many HTTP accesses, and may last an extended period of time (typically, a few minutes). Consistency in this context means that if the user added an item to her basket at some point, it should remain there until the end of the transaction. Furthermore, the item should still be available when time comes to pay and deliver the product.

* **Conflict Resolution:**

If multiple access requests is received for access to a specified resource, there has to be any mechanism to maintain conflict resolution so that only one process can have access to it.

## 1.2 Reliability vs. Availability

When concerned about implementing Replica management in distributed system we come by two primary motivation called Reliability and Availability.

Reliability represents the probability of components, parts and systems to perform their required functions for a desired period of time without failure in specified environments with a desired confidence. Reliability, in itself, does not account for any repair actions that may take place. Reliability accounts for the time that it will take the component, part or system to fail while it is operating. It does not reflect how long it will take to get the unit under repair back into working condition.

Whereas, availability represents the probability that the system is capable of conducting its required function when it is called upon given that it is not failed or undergoing a repair action. Therefore, not only is availability a function of reliability, but it is also a function of maintainability.

Reliability and availability address two orthogonal issues. A server that is reliable but rarely available serves no purpose. Similarly, a server that is available but frequently malfunctions or supplies incorrect or stale data causes headache for all.

## 1.3Client Centric Consistency Model

A contract between a (distributed) data store and processes, in which the data store specifies precisely what the results of read and write operations are in the presence of concurrency. Consistency models define rules for the apparent order and visibility of updates, and it is a continuum with tradeoffs. The Client Centric Consistency Model the emphasize more on maintaining a consistent view of things for the individual client process that is currently operating on the data-store. It concentrates on what specific clients want, instead of what should be maintained by servers.

## 1.4 Total order multicast:

This is a form of atomic multicast in which every member of the group is required to accept all messages in identical order. It implies that if every process i maintains a queue Q.i (initially empty) to which a message is appended as soon as it is accepted, then eventually, for any two distinct processes i and j, Q.i = Q.j. The order in which the messages are accepted has no connection with the real time at which these messages were sent out.

# Problem Statement

In the given program, we are asked to design a Data Replication Management System for reserving eight meeting room shared by 10 Departments on an as-needed basis. The reservation protocol used by the Data Replication Management System should be efficient enough to ensure the reservation of the meeting room efficiently and without conflict. The process of reservation should also be optimized for fair allocation and without chaos. However, maintaining a chaos and conflict-less reservation of the meeting room is complicated due to the multiple requests for the reservation and concurrent access. Besides, maintaining a global clock to synchronize the reservation process to determine the order of request as well as maintaining use of global variables is the major challenge, while designing the system. The system is intended to solve the problem of efficient and conflict-less reservation process and must accompany its intended goal or else it will result in incongruous and discrepant use of available meeting rooms.

In our solution to the Data Replication Management System for reservation, we employed logical timestamp with total ordering of request and client-centric consistency model of data store which specifies precisely what the results of read and write operations are in the presence of concurrency. Optimizing the reservation process with congruous use of the available meeting room is achieved by the specific implementation of our system

# Project Requirement

## User Requirement:

* Secretaries must be able to request for the reservation of meeting room at any time.
* The system must acknowledge the secretaries whether the reservation process is success or failure.
* The system must display information about the reserved meeting rooms.

## Reliability Requirement:

* A client, or a transaction, shall not be affected by a single server failure.
* The system shall be available 24x7 without any critical system failures
* No reservation process shall ever be in an inconsistent state.

## Performance Requirement:

* System should support up to 10 concurrent request for reservation of meeting rooms.
* System must allow 8 different rooms to be reserved for each 1 hour reservation slot.

## Software Requirement:

* Operating system: Linux
* IDE: Visual Studio 13; GCC Compiler

# Implementation:

The specific implementation of communication protocols and ordering of the request-reply- release message in the reservation system can be well explained using the following diagram:

**Replica**

**Secretary**

**S1**

**S2**

**S10**

**R1**

**R2**

**R10**

**Primary**

**Server**

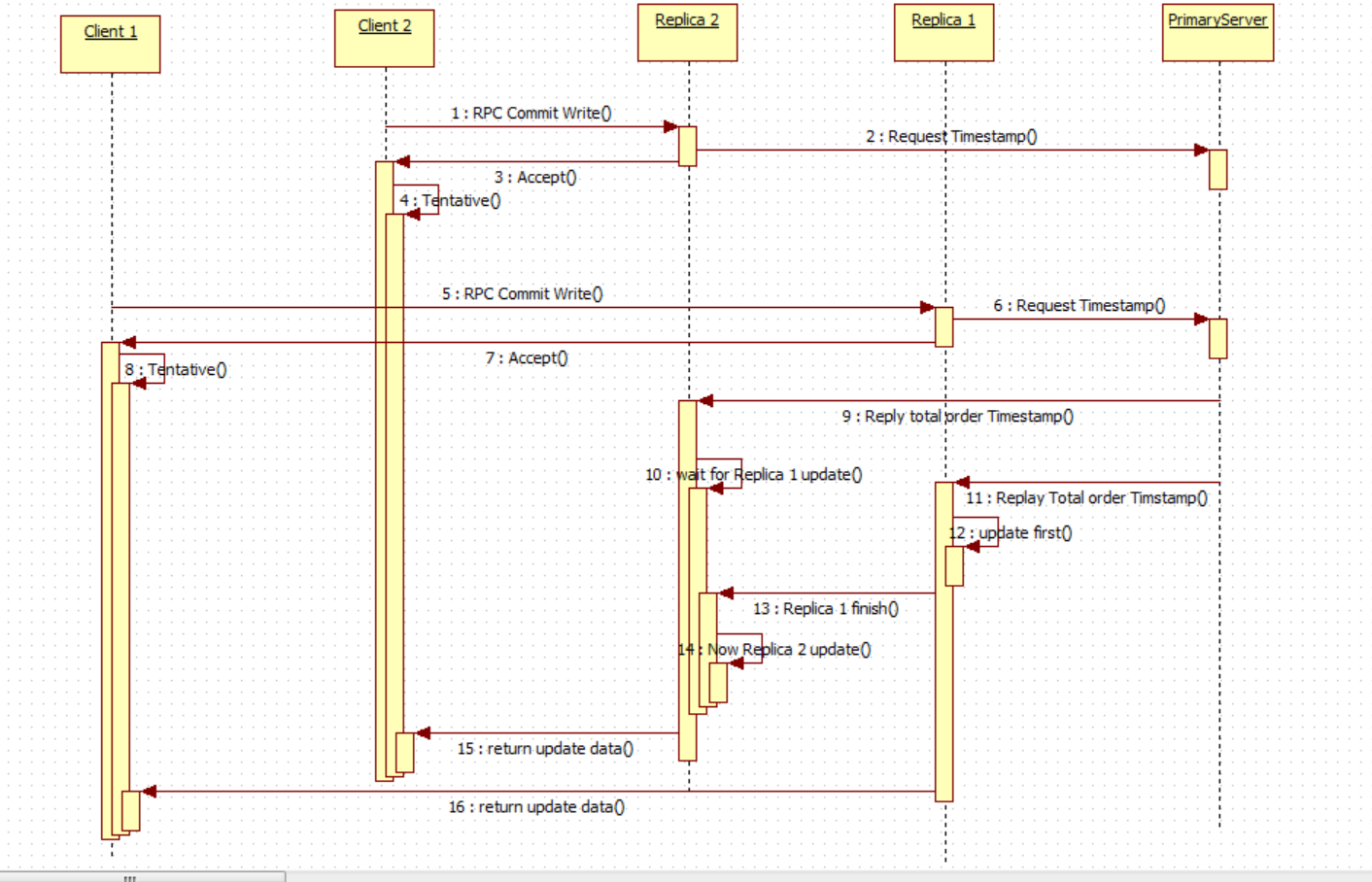
1. 1. Write
2. 2. Request
3. 3.Reply
4. 4. Update
5. 5. Release
6. 6. Release
7. 6. Release

**Fig: Interaction between secretary, replicas and primary server**

The sequence of message exchange for the reservation of meeting room can be explained as follow:

1. Write: The department Secretary wanting to reserve room send a write request to the replica
2. Request: The replica receiving write request sends a time stamped Request message to the primary server.
3. Reply: The primary server respond to the request message with a reply message
4. Update: After the replica receive the reply from primary server; it updates its data indicating that the room is reserved for the requested day and time.
5. Release: The replica after updating itself sends a release message to the primary server.
6. Release: The primary server then broadcast release message to all of the replica to update the status of the reservation to the local database.

We have underline three scenarios for our Data Replication Management System for reserving room that effectively circumscribe all the possible cases that might occur in process of reservation:



**Case 1: Single entry to reserve meeting room:**

The case deal with the scenario where request from any of the ten departments for the reservation of meeting room out of eight. It easier to deal with this scenario as any room can be allocated to the request.

|  |
| --- |
| Dept. 0 |
| Thursday  16:00-17:00  Book room 0 |

**Case 2: Multiple entries for different time:**

This case deals with scenario for the reservation of meeting room by different department at different timestamp. Here dealing with reservation is again not so complex as the request come at different time slot and only should be aware that the reservation process is limited to only other vacant meeting room.

There are 5 concurrency request sent to Replica. All of them book different room at different time.

Test data:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Dept. 0 | Dept. 2 | Dept. 4 | Dept. 6 | Dept. 8 |
| Monday  8am-9am  Book room 0 | Wednesday  10:00-11:00  Book room 2 | Friday  12:00- 13:00  Book room 4 | Tuesday  14:00-15:00  Book room 6 | Thursday  16:00-17:00  Book room 0 |

**Case 3: Multiple entries concurrently**

This is the most difficult case for the implementation of the system. With multiple request for the reservation concurrently, there has to be rule defined to overcome this conflict. Our conflict resolution rule is based on timestamps. The earliest entry for the slot is prevailed and other appointments are cancelled. In case if there is a tie for the earliest entry, the first name is used to break the tie.

Test data:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Dept. 1 | Dept. 3 | Dept. 5 | Dept. 7 | Dept. 9 |
| Tuesday  9am-10am  Book room 1 | Tuesday  9am-10am  Book room 1  Compete with Dept.1 Failed to book | Monday  13:00- 14:00  Book room 5 | Wednesday  15:00-16:00  Book room 7 | Friday  8:00-9:00  Book room 1 |

# Screenshot

# Conclusion

The system was successfully developed based on client centric consistency model for distributed system. The system had provision to take multiple request from different secretaries and then allocate the meeting room in efficient and conflict-less manner. The communication between the secretaries, replicas and primary server was based on UDP protocol. To summarize, the system accepted multiple reservation request and allocates the room on total ordering of the request being sent.