A picture containing graphical user interface

Description automatically generated

COMP 6231 DISTRIBUTED SYSTEMS DESIGN

SUMMER 2021

DESIGN DOCUMENT FOR PROJECT

TEAM SMARTY PANTS

BINIT KUMAR - 40172005

ABDUL AZIZ RAZA SHAIK - 40182726

CONCORDIA UNIVERSITY

MONTREAL, QUEBEC, CANADA

AUGUST 2021

**Introduction:**

Software Failure Tolerant/Highly Available Distributed Class Management System (DCMS):

Distributed Class Management System is a distributed system used by center managers to manage information regarding the teachers and students across different centers.

The three centers locations are:

* Montreal (MTL)
* Laval (LVL)
* Dollard-des-Ormeaux (DDO)

The server for each center (called CenterServer) maintains a number of Records. The two types of Records are:

* TeacherRecord
* StudentRecord

**Objectives:**

Fault Tolerant: Making available multiple replicas of the primary server to maintain backups which would be useful in event of primary server failure.

High Availability: Failure Detection subsystem which runs periodically and checks for failed processes and elects a new leader if required using distribution election algorithm when leader has failed.

**Description of the technique used:**

Common Object Request Broker Architecture (CORBA):

DCMS has been designed using CORBA which is the object oriented equivalent of remote procedure calls (RPC). ORB (Object Request Broker) is responsible for management of the remote access to objects. Common ORB architecture is a software bus for distributed objects. CORBA provides a nearly transparent access to remote objects from a local program using the client-server paradigm. Unlike RMI, CORBA has an advantage of being totally platform and language independent.

CORBA implementation looks like:

Diagram

Description automatically generated

**Design and Architecture of the system:**

Diagram, engineering drawing

Description automatically generated

HIGHLY AVAILABLE & FAULT TOLERANT DCMS

The DCMS architecture has 3 locations (Montreal, Laval and DDO) which are the clients of the system.

The Managers at these locations have access to the database which consists of the records: Student Record and Teacher Record. The managers can perform functions such as createTRecord, createSRecord, getRecordCounts, editRecord and transferRecord as per requirement.

ManagerClient is the client program which handles these requests and attempts to create, edit or transfer the appropriate record with the corresponding server associated with the manager in the form of a hashmap and each server also maintains a log containing the history of all the operations that have been performed on that server in a separate text file. By invoking getRecordCounts, the manager of a center can also get the number of record (both teacher and student) present in their respective database.

This system has multiple centers and one CenterServer for each of them which make use of a centralized ManagerClient to invoke the required method from the repository, for each operation it finds the information about the requesting server and invokes the corresponding operation.

Diagram

Description automatically generated

**Workflow/Dataflow of the system:**

* CORBA remote reference registered with Naming Service to the Front End.
* Request sent to Front End (FE) using remote invocation by client.
* Request received by FE is forwarded to the request buffer and in turn to the Primary Server.
* Primary Server:
  + Prepare Replica Requests and multicasts to other servers using UDP FIFO broadcast.
  + Write data (hashmap) to the primary server backup.
  + Process requests and sends response to FE.
* Replica Server:
  + Send “Ack” to primary server upon receiving request.
  + Write data (hashmap) to replica backup.
  + Process requests and send response to FE.
* Upon receiving all responses, FE will:
  + Log all responses (primary and replicas) to the primary logs and replica logs.
  + Primary Server response is communicated to the client.
  + In case of crash (primary server), FE will update and send request to new primary elect.

Maintenance of multiple replicas for each server with access to all hashmaps of MTL, LVL and DDO.

FIFO Broadcast is implemented upon receiving requests from FE, primary server broadcasts them using reliable UDP and FIFO queue to replicas as well.

**Failure Detection:**

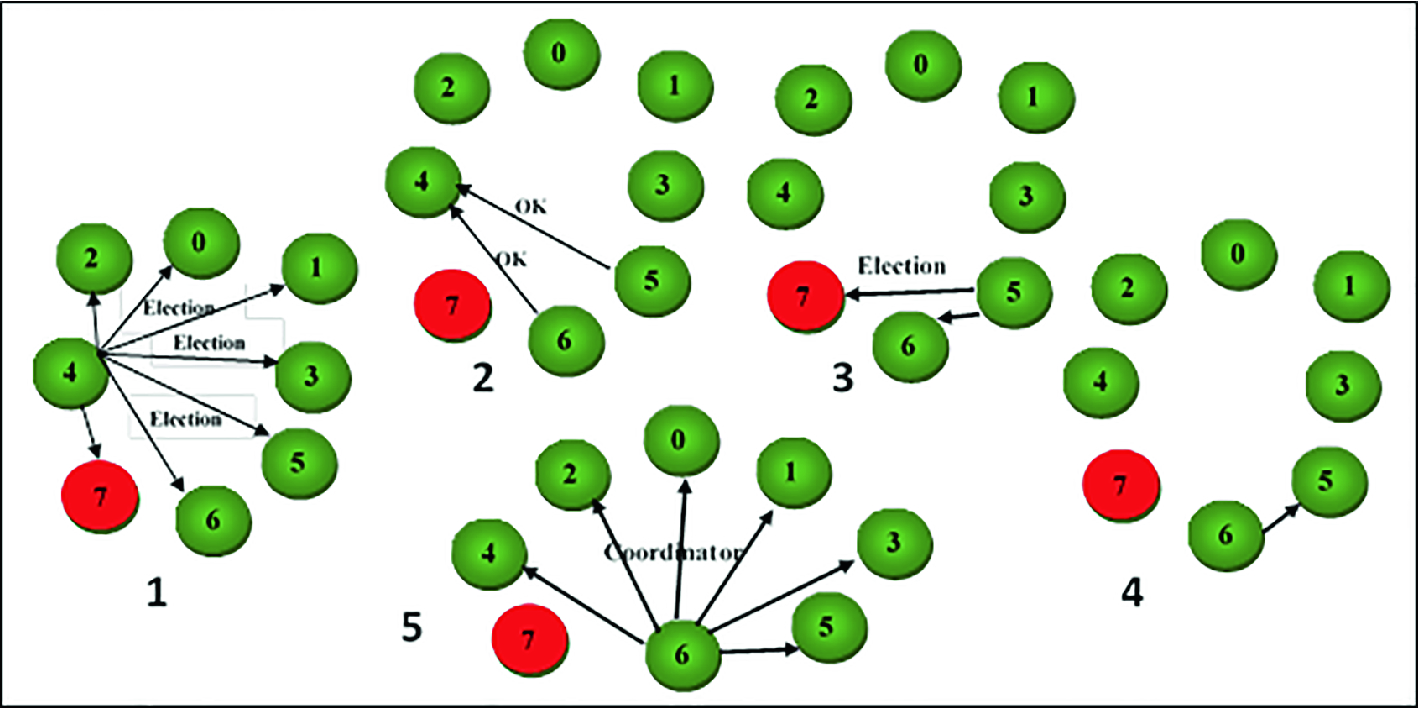
Heartbeat Implementation in Server Front End enables failure detection mechanism by constantly sending heartbeat messages within the system from each server process to another and also communicates status of the server processes in its group.

If a server stops responding or crashes, the absence of heartbeat message will trigger the fault detection and in order to resolve it initiates the Election Algorithm to elect a new primary server.

**Election Algorithm for Failure Correction:**

Upon the primary process’ failure, the process that recognizes the failure first sends an election message to all the remaining processes with higher weight/priority.

If a process with a higher weight/priority is available, that particular process begins the election again. This goes on until the process with the highest priority elects itself as the leader and sends the coordinator message.



Bully Algorithm for electing new leader

**Working of Heartbeat Implementation in case of Failure of a server:**

Diagram

Description automatically generated

**Synchronization and Concurrency Control:**

Concurrency is an important aspect for a good distributed system. In our project, we may have instances of multiple managers accessing the data at the same time and may lead to faults or failures if not handled properly. Synchronization of commonly used methods can be used by multiple managers at the same time.

**Implementation of Synchronization:**

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated with medium confidence

**Measures taken to ensure reliability in the DCMS:**

* Consistent DCMS Backup by maintaining a copy of each change in the database in a separate file.
* Acknowledgements for each message received.
* Temporary request buffer in case of failure.

By taking these measures our DCMS is more fault tolerant, concurrent, reliable UDP and provides a backup of data.

One of the tradeoffs with this is the higher CPU usage for maintaining all the backup servers and replicas and also maintaining the buffers.

**Challenges Faced:**

* Achieving synchronization for the data.
* Implementation of bully algorithm in case of failure of primary server.
* Heartbeat for constant monitoring of all the servers to instantly know about failures.
* Acknowledgement for each message received.

TEST CASES EXECUTED:

# D. Test Scenarios:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Description | Expectation | Result |
| 1 | Manager creating Teacher records | Record should be saved in database | PASS |
| 2 | Manager creating Student records | Record should be saved in database | PASS |
| 3 | Managers requiring valid id to access the server Test Data: (MTLXXXX / LVLXXXX / DDOXXX) | Managers should be able to connect to the right server | PASS. |
| 4 | Managers not able to connect with invalid id | Invalid managers id will get an error message | PASS |
| 5 | Managers able to get all the record counts | The total count of all the records should be printed for each of the servers | PASS |
| 6 | Manager should be able to edit records | Manager should be able to modify records from the record id | PASS |
| 7 | Transfer Records to different data centers | The transferred record should only be deleted from the sender once it is atomically transferred to new datacenter | PASS |
| 8 | Status Field validation in student record | Only Active and InActive status | PASS |
| 9 | Managers should not be allowed to edit any fields other than Address, Phone and Location in the Teacher Record. | Any fields other than Address, Phone and Location in the Teacher Record are not modified | PASS. |
| 10 | Managers should not be allowed to edit any fields other than Course Registered, Status and Status Date in the Student Record. | Any fields other than Course Registered, Status and Status Date in the Student Record are not modified | PASS. |
| 11 | FIFO broadcast the request to all the server replicas using UDP datagrams, receives requests from the server replicas and sends a single response back | A linkedlist based queue is being created that forward request in FIFO based to server replicas and the accumulated response is atomically transferred back | PASS. |
| 12 | FIFO broadcast subsystem over the unreliable UDP layer | Multicast sender and receivers transfers and occupy the acknowledgement as a response | PASS. |
| 13 | Failure detection subsystem in which the processes in the group periodically check each other and remove a failed process from the group | Efficient mapping of reporting time of the 9 servers and a watcher thread to check which one is delayed by 1 second | PASS. |
| 14 | Distributed leader election subsystem (based on the bully algorithm), | Active election subsystem triggered once watcher thread identify if primary server has not reported to create a new primary server | PASS. |
| 15 | Test API to simulate the process killing and to check bully algorithm | An option 6 is added in manager client that kills whatever primary server is running based on the location (MTL,LVL,DDO) | PASS. |
| 16 | Deploy your application on a local area network, and test the correct operation of your application using properly designed test runs. | Successful deploy of distributed application on multiple machine to test the CORBA connection between client and server and UDP connection between server and within server location. | PASS |

UML Diagrams:

Use case diagram:

Diagram

Description automatically generated

Class diagram

A black and white map

Description automatically generated with low confidence



References:

* George Coulouris, Jean Dollimore, Tim Kindberg and Gordon Blair, "*Distributed Systems Concepts and Design"*, Fifth Edition, Addison-Wesley, 2012, ISBN: 0-13-214301-1.
* M. L. Liu, "*Distributed Computing: Principles and Applications*", Pearson Addison-Wesley 2004, ISBN 0-201-79644-9.
* https://www.javatpoint.com/q/4780/corba
* https://en.wikipedia.org/wiki/Common\_Object\_Request\_Broker\_Architecture
* https://docs.oracle.com/javase/8/docs/technotes/guides/idl/jidlExample.html https://www.codejava.net/java-se/networking/java-udp-client-server-program-example