COMP 6231

Distributed System Design

Project Report

Distributed Class Management System (DCMS)

Submitted to: Professor Mohamed Taleb

By

Stallone Mecwan: 40161375

Jay Patel: 40163706

TABLE OF CONTENTS

TABLE OF FIGURES

Architecture - Fault Tolerant & High Available DCMS

Fault Tolerant

Implement multiple replicas for the primary server to maintain backups which could in turn serve as primary when the primary server fails.

Highly Availability

Implement a failure detection subsystem in which the servers in the group periodically check each other for failed process

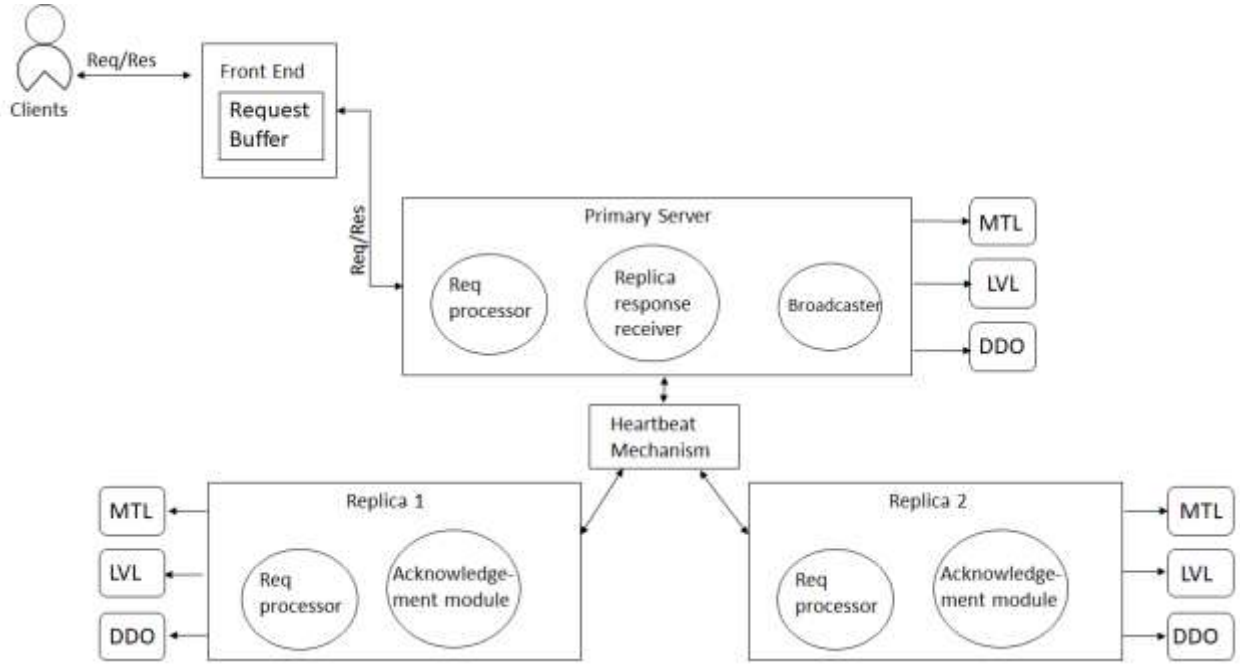


Figure 1: DCMS Architecture

Flow

- Front End will register CORBA remote reference with Naming Service
- Front end receives requests from clients, puts it into the request buffer
- Requests are forwarded to the Primary server
- Primary Server prepares the replica request, multicasts it to other two replicas using UDP communication, which is a FIFO Broadcast implementation
- Primary Server writes the hash map to the primary server backup source.
- Primary Server processes the received request in the FIFO order and sends the response back to the front end
- Replica servers receive the broadcasted request and sends the acknowledgement to the primary server

- Replica servers write the hash map to the replica server backup source.
- Replica servers process the request and sends back the response to the primary server
- Once front end receives the response from all the replicas, all the responses from primary, as well as from the replicas are logged in primary log and replicas log respectively.
- The primary server's response will be returned back to the client.
- In case of current primary server's crash, the front end sends the request again to the currently elected Primary server.

FIFO Broadcasting

After the Primary server receives requests from the Front end, it broadcasts the requests to the replicas using Reliable UDP communication and FIFO queue implementation.
In this case, the sender is the Primary server

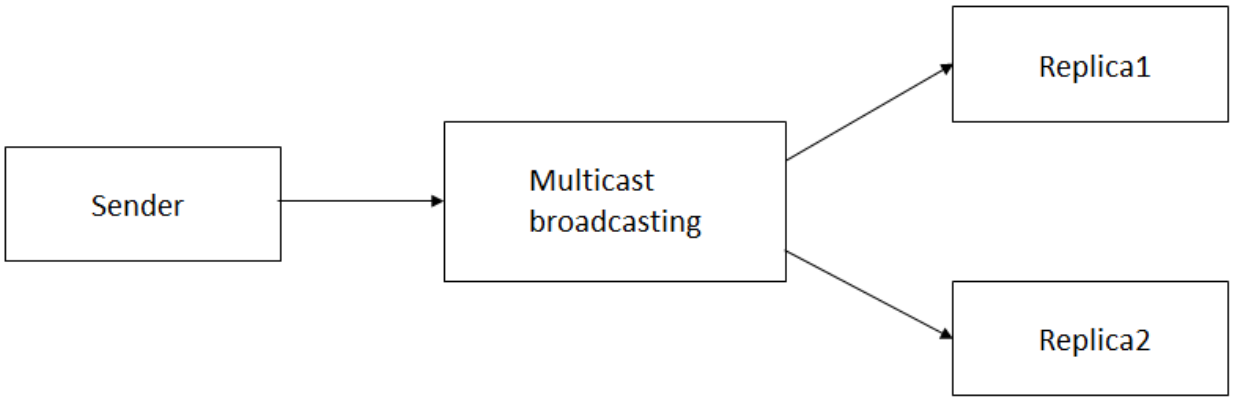


Figure 2: FIFO Broadcasting

Fault Detection

To implement this we used heartbeat mechanism. Constant communication between processes takes places to know the status of the servers.
When a server isn't active, it gets detected and Election algorithms comes to play.

Election Algorithm (Bully Algorithm)

When the primary process fails, the process that senses the failure first sends the 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.
With various forms of Bully algorithm available, we implemented a type of bully algorithm that requires each process to have a queue and a flag to maintain a record history of all the leaders and failed servers. This way, the redundancy involved with the messages can be reduced.

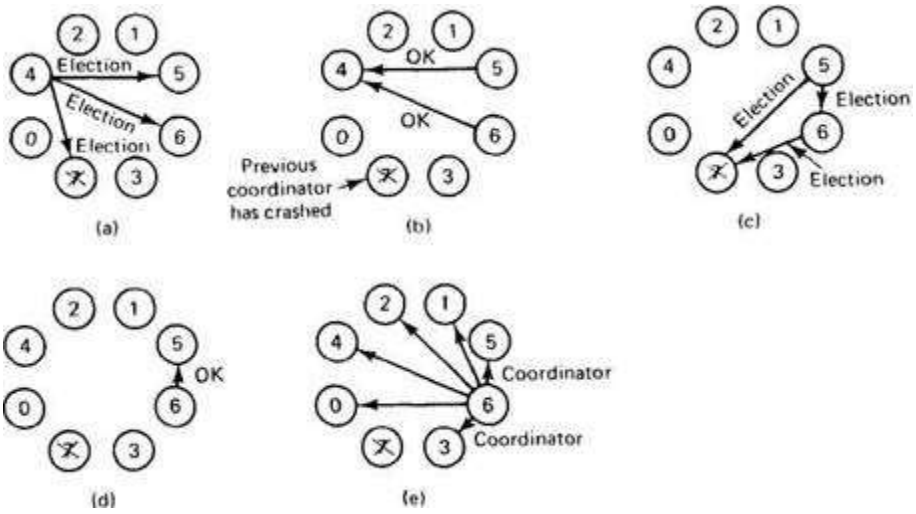


Figure 3: Bully Algorithm

Synchronization & Concurrency Implementation

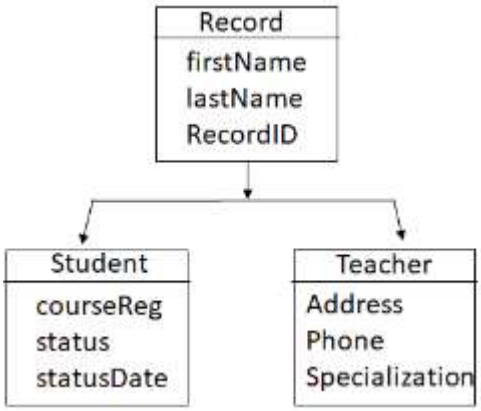
The data structures & statements are synchronized using the **Java's synchronization technique.**

The HashMap is synchronized, and so is the Array List within the HashMap.

(Semaphore usage in the previous builds was removed.)

The **methods in the class** that are accessed by multiple threads are synchronized and synchronized blocks are used to obtain locks on different objects/variables/data structures wherever concurrent access is possible.

HashMap



Starting the application
To start using the project got o Server folder and run “RunServer”.
Run “ManagerClient” from Client folder to start using the client.

(orbd and other parameters are saved as configuration within the files itself).

Test Cases

a)

1. Add a record in any server
2. Get the record count
3. Kill that particular server
4. Get the record count

```
Enter First name:
Jau
Enter Last name:
Patel
Enter Address:
H3S 1T2
Enter Phone number in the format (514-888-9999):
514-514-1413
Specialization courses:
DSD, APP
Enter the Location (MTL/LVL/DDO) according to manager ID used
MTL
Teacher record is created and assigned with TR1
4
Total Record Count from all 3 servers is :: MTL 1 , LVL 0 , DDO 0
6
Enter the Location (MTL/LVL/DDO)
MTL
MTL1 Server is killed and elected new leader MTL3 in the location MTL
4
Total Record Count from all 3 servers is :: MTL 1 , LVL 0 , DDO 0
```

Figure 4: Test Case a

b)

1. Kill the server you're logged in
2. Add a record
3. Get the record count

```
----- Select which operation you want to perform -----
1. Creating a teacher record (TR)
2. Creating a student record (SR)
3. Edit record
4. Display total Record count of all servers.
5. Transfer records.
6. Kill the Primary Server
7. Logout manager
6
Enter the Location (MTL/LVL/DDO)
MTL
MTL1 Server is killed and elected new leader MTL3 in the location MTL
1
Enter First name:
Stallone
Enter Last name:
Mecwan
Enter Address:
Avenue Linton, Montreal
Enter Phone number in the format (514-888-9999):
514-704-3265
Specialization courses:
DSD, Algorithms, Problem Solving
Enter the Location (MTL/LVL/DDO) according to manager ID used
MTL
Teacher record is created and assigned with TR1
4
Total Record Count from all 3 servers is :: MTL 1 , LVL 0 , DDO 0
```

Figure 5: Test Case b

c)

1. Add a record in any server
2. Get the record count
3. kill that particular server,
4. Edit the record

5. Get the record count

```
Enter the managerID
MTL1111
----- Select which operation you want to perform -----
1
Enter First name:
Mohammad
Enter Last name:
Taleb
Enter Address:
Montreal, Canada
Enter Phone number in the format (514-888-9999):
514-141-1234
Specialization courses:
Distributed Systems
Enter the Location (MTL/LVL/DDO) according to manager ID used
LVL
Invalid location entered, Please insert appropriate location
MTL
Teacher record is created and assigned with TR1
```

As we can see here, we logged in the Montreal Server, when any record is created, one can not enter any other location other than MTL while creating a record

```
4
Total Record Count from all 3 servers is :: MTL 1 , LVL 0 , DDO 0
6
Enter the Location (MTL/LVL/DDO)
MTL
MTL1 Server is killed and elected new leader MTL3 in the location MTL
3
Enter the Record ID
TR1
What do you want ot change? (1.Address 2.Phone or 3.Location)
2
Enter the value of the field to be updated
Enter the new Phone number in (514-888-9999) format
514-111-1234
Updated record with Phone :: 514-111-1234
```

```
4
Total Record Count from all 3 servers is :: MTL 1 , LVL 0 , DDO 0
```

Figure 6: Test Case c

d)

1. Add a record
2. Get the record count
3. Kill the server in which you added the record
4. Transfer the record
5. Get the record count

```
Enter first name of the student
Stallone
Enter  last name of the student
Mecwan
Enter the number of courses registered by the student
3
Enter the 3 courses(one by one) registered by the student
DSD
SOEN
Algorithms
Enter the status of student: (Active/Inactive)
Inactive
Enter the date(Format dd-mm-yyyy)
12-12-2021
student record is created and assigned with SR1
4
Total Record Count from all 3 servers is :: DDO 1 , MTL 0 , LVL 0
6
Enter the Location (MTL/LVL/DDO)
DDO
DDO1 Server is killed and elected new leader DDO3 in the location DDO
5
Enter the record ID
SR1
Enter the location(MTL/LVL/DDO)
LVL
Record transferred from DDO to LVL
```

```
4
Total Record Count from all 3 servers is :: DDO 0 , MTL 0 , LVL 1
```

Figure 7: Test Case d

e)

1. Kill the server in which you're logged in
2. Add a record
3. Get the record count
4. Edit the record
5. Get the record count

```
6
Enter the Location (MTL/LVL/DDO)
DDO
DDO1 Server is killed and elected new leader DDO3 in the location DDO
2
Enter first name of the student
Stallone
Enter  last name of the student
Mecwan
Enter the number of courses registered by the student
2
Enter the 2 courses(one by one) registered by the student
DSD
SOEN
Enter the status of student: (Active/Inactive)
Active
Enter the date (Format dd-mm-yyyy)
11-12-2021
student record is created and assigned with SR1
4
Total Record Count from all 3 servers is :: DDO 1 , MTL 0 , LVL 0
```

```
3
Enter the Record ID
SR1
What do you want ot change? (1.CoursesRegistered 2.Status or 3.statusDate)
3
Enter the value of the field to be updated
Enter the new Date (Format dd-mm-yyyy)
03-03-2002
Updated record with status date :: 03-03-2002
4
Total Record Count from all 3 servers is :: DDO 1 , MTL 0 , LVL 0
```

Figure 8: Test Case e

f)

1. Add a record in the server
2. Get the record count
3. Kill the server in which you added the record
4. Transfer the record from the server which you killed to another server
5. Logout
6. Log in the server to which you transferred the record
7. Edit the record in the logged in server
8. Get the record count

```
1
Enter First name:
Stallone
Enter Last name:
Mecwan
Enter Address:
Avenue Linton
Enter Phone number in the format (514-888-9999):
514-413-1212
Specialization courses:
Maths
Enter the Location (MTL/LVL/DDO) according to manager ID used
DDO
Teacher record is created and assigned with TR1
4
Total Record Count from all 3 servers is :: DDO 1 , MTL 0 , LVL 0
```

```
6
Enter the Location (MTL/LVL/DDO)
DDO
DDO1 Server is killed and elected new leader DDO3 in the location DDO
5
Enter the record ID
TR1
Enter the location(MTL/LVL/DDO)
LVL
Record transferred from DDO to LVL
7
Manager with DDO1212is logging out
Enter the managerID
LVL1212
3
Enter the Record ID
TR1
What do you want ot change? (1.Address 2.Phone or 3.Location)
1
Enter the value of the field to be updated
WestMount
Updated record with address :: WestMount
4
Total Record Count from all 3 servers is :: LVL 1 , MTL 0 , DDO 0
```

Figure 9: Test Case f

g)

1. Add a record
2. Get the record count
3. Kill the server in which you created the record
4. Transfer the record to any other server
5. Kill the server to which you transferred the record
6. Get the record count

```
2
Enter first name of the student
Stallone
Enter  last name of the student
Mecwan
Enter the number of courses registered by the student
1
Enter the 1 courses(one by one) registered by the student
DSD
Enter the status of student: (Active/Inactive)
Inactive
Enter the date(Format dd-mm-yyyy)
12-12-2001
student record is created and assigned with SR1
4
Total Record Count from all 3 servers is :: MTL 1 , LVL 1 , DDO 0
6
Enter the Location (MTL/LVL/DDO)
MTL
MTL1 Server is killed and elected new leader MTL3 in the location MTL
5
Enter the record ID
SR1
Enter the location(MTL/LVL/DDO)
LVL
Record transferred from MTL to LVL
6
Enter the Location (MTL/LVL/DDO)
LVL
LVL1 Server is killed and elected new leader LVL3 in the location LVL
4
Total Record Count from all 3 servers is :: MTL 0 , LVL 2 , DDO 0
```

Figure 10: Test Case g

h)

1. Add a record
2. Transfer the record from the server you created to any other server
3. Kill the initial server in which you created it
4. Kill the server to which you transferred it
5. Get the record count

```
2
Enter first name of the student
Stallone
Enter  last name of the student
Mecwan
Enter the number of courses registered by the student
3
Enter the 3 courses(one by one) registered by the student
DSD
SOEN
Algorithms
Enter the status of student: (Active/Inactive)
Active
Enter the date (Format dd-mm-yyyy)
12-12-2013
student record is created and assigned with SR1
 5
 Enter the record ID
 SR1
 Enter the location(MTL/LVL/DDO)
 MTL
 Record transferred from DDO to MTL
6
Enter the Location (MTL/LVL/DDO)
DDO
DDO1 Server is killed and elected new leader DDO3 in the location DDO
6
Enter the Location (MTL/LVL/DDO)
MTL
MTL1 Server is killed and elected new leader MTL3 in the location MTL
 4
 Total Record Count from all 3 servers is :: DDO 0 , MTL 1 , LVL 0
```

Figure 11: Test Case h

i)

1. Add a record
2. Get the record count
3. Kill the server in which you created the record
4. Transfer the record from the server you created it to any other server

5. Kill the server to which you transferred the record
6. Get the record count
7. Log out from the initial server
8. Log in to the server to which you transferred the record
9. Transfer the record from the server to the remaining server
10. Kill the server to which you transferred
11. Get the record count

```
1
Enter First name:
Mohammad
Enter Last name:
Taleb
Enter Address:
WestMount, QC
Enter Phone number in the format (514-888-9999):
514-514-5141
Specialization courses:
DSD
Enter the Location (MTL/LVL/DDO) according to manager ID used
DDO
Teacher record is created and assigned with TR1
4
Total Record Count from all 3 servers is :: DDO 1 , MTL 0 , LVL 0
6
Enter the Location (MTL/LVL/DDO)
DDO
DDO1 Server is killed and elected new leader DDO3 in the location DDO
5
Enter the record ID
TR1
Enter the location(MTL/LVL/DDO)
MTL
Record transferred from DDO to MTL
6
Enter the Location (MTL/LVL/DDO)
MTL
MTL1 Server is killed and elected new leader MTL3 in the location MTL
4
Total Record Count from all 3 servers is :: DDO 0 , MTL 1 , LVL 0
```

```
7
Manager with DDO1212is logging out
Enter the managerID
MTL1211
5
Enter the record ID
TR1
Enter the location(MTL/LVL/DDO)
LVL
Record transferred from MTL to LVL
6
Enter the Location (MTL/LVL/DDO)
LVL
LVL1 Server is killed and elected new leader LVL3 in the location LVL
4
Total Record Count from all 3 servers is :: MTL 0 , LVL 1 , DDO 0
```

Figure 12: Test Case i

j)

1. Add a record
2. Get the record count
3. Kill the server to which you are about to transfer the record
4. Transfer the record from the server in which you created to the server which you just killed
5. Get the record count
6. Kill the remaining server
7. Get the record count
8. Log out
9. Log in the server to which you transferred
10. Transfer the record from logged in server to the remaining server

```
2
Enter first name of the student
Stallone
Enter  last name of the student
Mecwan
Enter the number of courses registered by the student
2
Enter the 2 courses(one by one) registered by the student
DSD
Comparative Studies
Enter the status of student: (Active/Inactive)
Status assigned Invalid!
Enter the status of student: (Active/Inactive)
Active
Enter the date (Format dd-mm-yyyy)
12-12-2020
student record is created and assigned with SR1
4
Total Record Count from all 3 servers is :: MTL 1 , LVL 0 , DDO 0
6
Enter the Location (MTL/LVL/DDO)
LVL
LVL1 Server is killed and elected new leader LVL3 in the location LVL
5
Enter the record ID
SR1
Enter the location(MTL/LVL/DDO)
LVL
Record transferred from MTL to LVL
4
Total Record Count from all 3 servers is :: MTL 0 , LVL 1 , DDO 0
6
Enter the Location (MTL/LVL/DDO)
DDO
DDO1 Server is killed and elected new leader DDO3 in the location DDO
4
Total Record Count from all 3 servers is :: MTL 0 , LVL 1 , DDO 0
```

```
7
Manager with MTL1111is logging out
Enter the managerID
LVL2222
 5
 Enter the record ID
 SR1
 Enter the location(MTL/LVL/DDO)
 DDO
 Record transferred from LVL to DDO
```

Figure 13: Test Case j

File Descriptions

Client.Implementation

Implementation gets the inputs from the Manager Client and sends them to the server to perform the corresponding operations
Returns the Success/Failure message to the client for the corresponding operations.

Client.ManagerClient

perform operations for MTL, DDO and LVL locations by sending the
request calls to the server.
Receives the Success/Failure message for the corresponding operations and
forwards them to the client.

Server.HeartBeat.HeartBeatSender

Heart beat sender class which keeps sending to the other servers
in the servers list.

Server.HeartBeat.HeartBeatReceiver

Heart beat receiver to keep checking the other servers' status

Server.RunServer

RunServer class creates the CORBA server instance for the current project
and establishes the initial set of communication between the client module
and the server module for performing various operations

FrontEnd.UDPRespReceiver

Here thread is initialized by the FrontEnd server in order to receive the responses from the servers and replicas and the data is added to the responses HashMap

FrontEnd.TransferRespToFrontEnd

Transfers the received response from primary server to the front end

FrontEnd.TransferReqToCurrServer

Performs the operation based on the type of requestID by routing the request to the current server.
After processing the data, the respective requestID and the response is sent to the Front End

FrontEnd.PrimaryServerFIFO

This Thread receives the input from the Front End and adds the data to the FIFO Queue and transfer the data to the TransferReqToCurrentServer class.

Server.CreateReplicaRequest

includes all the server operations' implementations, implements all the methods in the IDL interface. Performs the necessary operations and returns the result/acknowledgement back to the Client.

Server.ReplicaAckSender

This thread is called to send the acknowledgement from the respective replicas to the primary server

Server.UDPReqServer

forwards the request received to the respective server port

Server. UDPReqProvider

handles the incoming messages call (Get Rec,Trans Rec) and based on the incoming call
Routes the packet to the necessary location to retreive the data

Server.UDPReceiver

class that serves other servers' requests in form of UDP communication

Server.ServerImpl

includes all the server operations' implementations, implements all the methods in the IDL interface Performs the necessary operations and returns the result/acknowledgement back to the Client.

Server.ReplicaReqProcessor

Thread here that processes the request, once the replica receives the request, and calls the appropriate server's method and returns the response to the primary server, once the processing is done.

Server.ReplicaRespReceiver

Thread class that receives the acknowledgement or response from the replicas
and passes it on to the primary server

Important part/ Difficulty and changes

Implementing election algorithm and establishing communication between multiple servers after the new leader was elected was challenging.

Continuous and consistent communication among the servers and sending periodic messages to other servers in the group using UDP communication was challenging

Removed extra functionalities in the .idl file (displayAll, display) as it had to be incorporated in this huge code base.

Records had to be passed numerous amounts of time due to which objects of record class were used instead of passing all the arguments (name, phone, address, etc).

Communication with the right socket/port was very challenging. Primary server with the replicas have a lot of communication. This includes status messages, getting record count, transferring record etc. Especially with multiple threads involved.

For synchronization purposes we had used semaphore that was conflicting with other code, due to which some function implementations had to be changed.

Implementing request buffer for serving requests between Front end and primary server was exhausting.

Notes:
CPU Usage: Due to the usage of multiple threads, many client-server interactions suffer a delay. This might be due to higher CPU usage.
Restarting Server: After crashing any of the server (Primary or Replica), the server does not come back online automatically.

REFERENCES

1. https://medium.com/@AniketShukla/of-leader-and-nodes-3d66f0a3f5d8