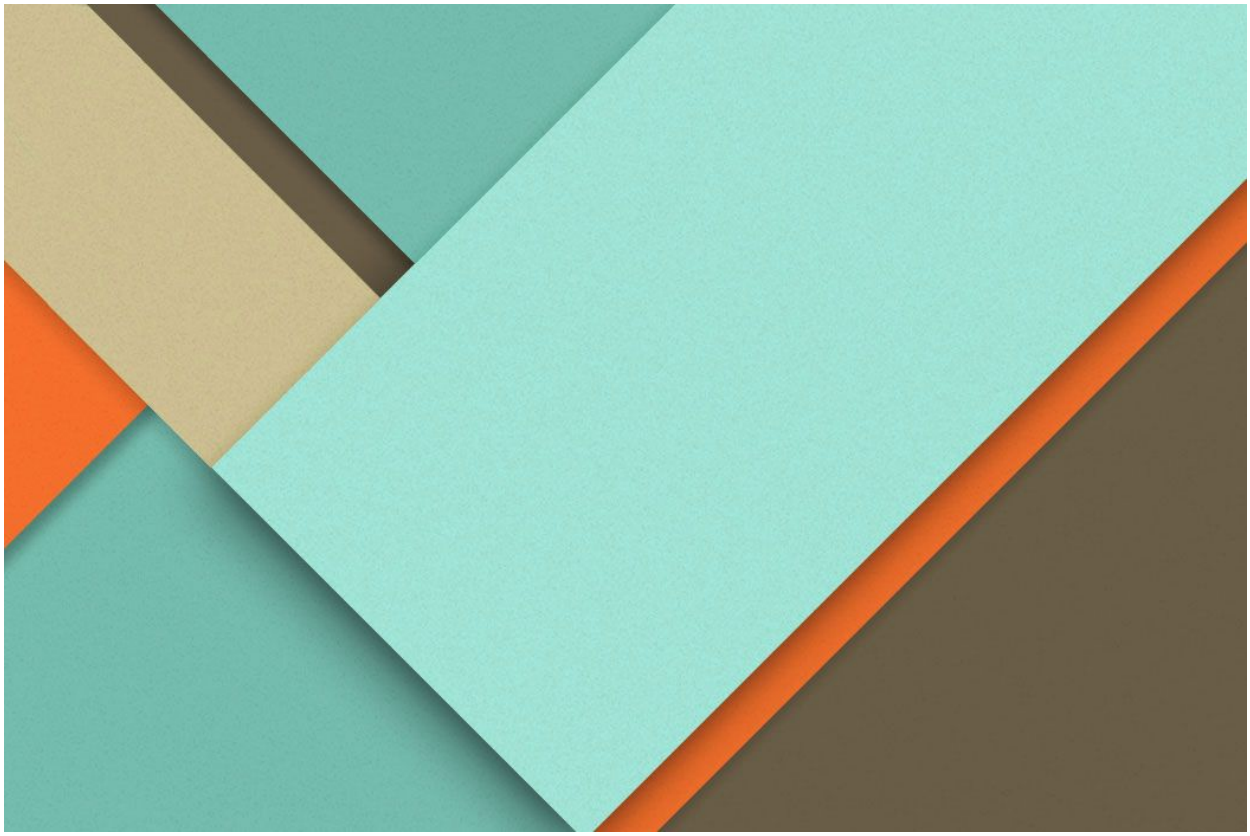


Term Project

CS 432

Ahmed Abdella	12
Khaled Ali	24
Mohamed Ayman	62
Mahmoud Hussein	70
Moustafa Mahmoud	76

Code Organization

The code is organized into 3 main components:

- Client

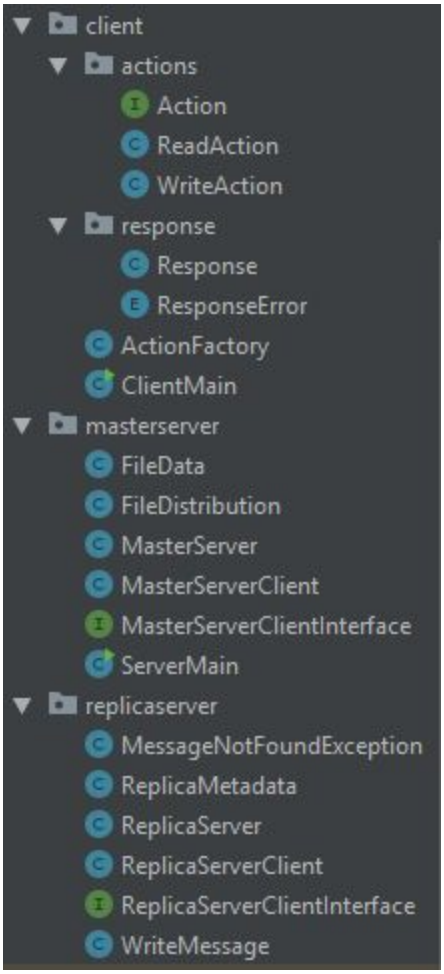
  Contains the code for our client used for testing purposes. This includes the available actions (read and write) and the response object.

- Master Server

  Contains the code for the master server, along with the file holder class and the RMI interface.

- Replica Server

  Contains the code for the replica server (primary and non-primary). Also, have the RMI interface and the replica metadata.



Main Functions

Action::executeAction

This function is defined in the **Action** interface which is implemented by **ReadAction** and **WriteAction** classes. It contains the logic implementation of the action using a command design pattern and it returns a **Response** object containing the errors if there exists any and the content of the file as with the case of **ReadAction**.

1. **ReadAction:** It sends a request to the master server to fetch all the available replicas for a file, then sends a request with the file name to the primary replica and receive the content of the file from the replica.
2. **WriteAction:** It sends the request information to the master server and fetches the replicas available for the file, then it sends another request to the primary replica with the transaction id, new content to be appended to the file, and the file

distribution object specifying all the other replicas to allow the primary replica to maintain and synchronize this file across replicas.

MasterServer::runServer

It's responsible for running the master server using the following steps:

1. Creates an object from **MasterServerClient** which implements **MasterServerClientInterface**.
2. Rebinds the **MasterServerClient** object it created to the master server domain in the RMI registry.

ReplicaServer::runServer

It's responsible for running the master server using the following steps:

3. Creates an object from **ReplicaServerClient** which implements **ReplicaServerClientInterface**.
4. Rebinds the **ReplicaServerClient** object it created to the replica server domain in the RMI registry using the replica server identifier to create a unique domain.

ServerMain::main

This is the main function for the master server and it does the following:

1. Parses the replica servers configuration file and creates for each replica server a metadata object.
2. Launches the RM Registry.
3. Launches the replica servers.
4. Launches the Master server.

MasterServerClient::read

The read function for the master server which is available to the client interface. It checks if the file exists, otherwise, it throws a FileNotFoundException and then it returns the distribution of replicas for the file to allow the client to fetch it from the primary replica.

MasterServerClient::write

The write function for the master server which is available to the client interface. If the file doesn't exist, it creates it and returns the information necessary to the client such as:

1. Transaction ID
2. File Distribution

ReplicaServer::read

The read function for the replica server which is available to the client interface. It locks the file and then reads all of its content which is sent back to the client as a result.

In case the file doesn't exist, it will throw a FileNotFound exception.

ReplicaServer::write

The write function for the replica server which is available to the client interface. In case of the primary replica server, it will propagate the action to the remaining replica servers, where it is only stored in memory and not flushed until the commit function is called.

ReplicaServer::comit

The replica server commit function which is responsible for committing a transaction and flushing the changes to the disk.

ReplicaServer::abort

The replica server abort function which is responsible for aborting the transaction from the remaining replicas in case it was called on the primary replica. It also removes the appended content from the memory.

How To Compile And Run

./start_server -ip 127.0.0.1 -port 1900

Components Of The File System

Client

It's responsible for parsing the input file and mapping it to a set of transactions which are then executed against the master server.

Master Server

It acts as a mediator between the client and replicas where it provides the client with the information needed regarding the file distribution and runs periodic heartbeats to check for any silent failures.

Replica Server

It is responsible for receiving requests and executing it whether as a primary replica or a generic replica. It executes read/write actions as a part of transactions.

Assumptions

- There exists more than or equal to 3 replica servers up and running upon launching the master otherwise, an exception is thrown.
- Transaction executes on a single file, otherwise, an exception is thrown.
- Transactions in the client input file are separated by an empty new line.
- Read actions in the client input file have two parameters, the keyword ‘read’ and file name, which is separated by semicolons. (i.e ‘read;file_1.txt’)
- Write actions in the client input file have three parameters, the keyword ‘write’, file name, and the content to be appended to the file, which is separated by semicolons. (i.e ‘write;file_1.txt; Hello world’)
- Write actions append to the file and don’t modify previous content.

Team Members Roles

Team was split into two sub-groups:

First group (2 People) to launch the master node, handle client requests and decide how clients will communicate with the master node.

Second group (3 People) focused mainly on replicas and implementing the main replica and its backups and the different scenarios of failures.

After that comes the integration part as we implemented the needed parts for integration part as heartbeats and launching via SSH.

Finally everyone collaborated on writing the report.