Concordia University

Distributed Event Management System using CORBA

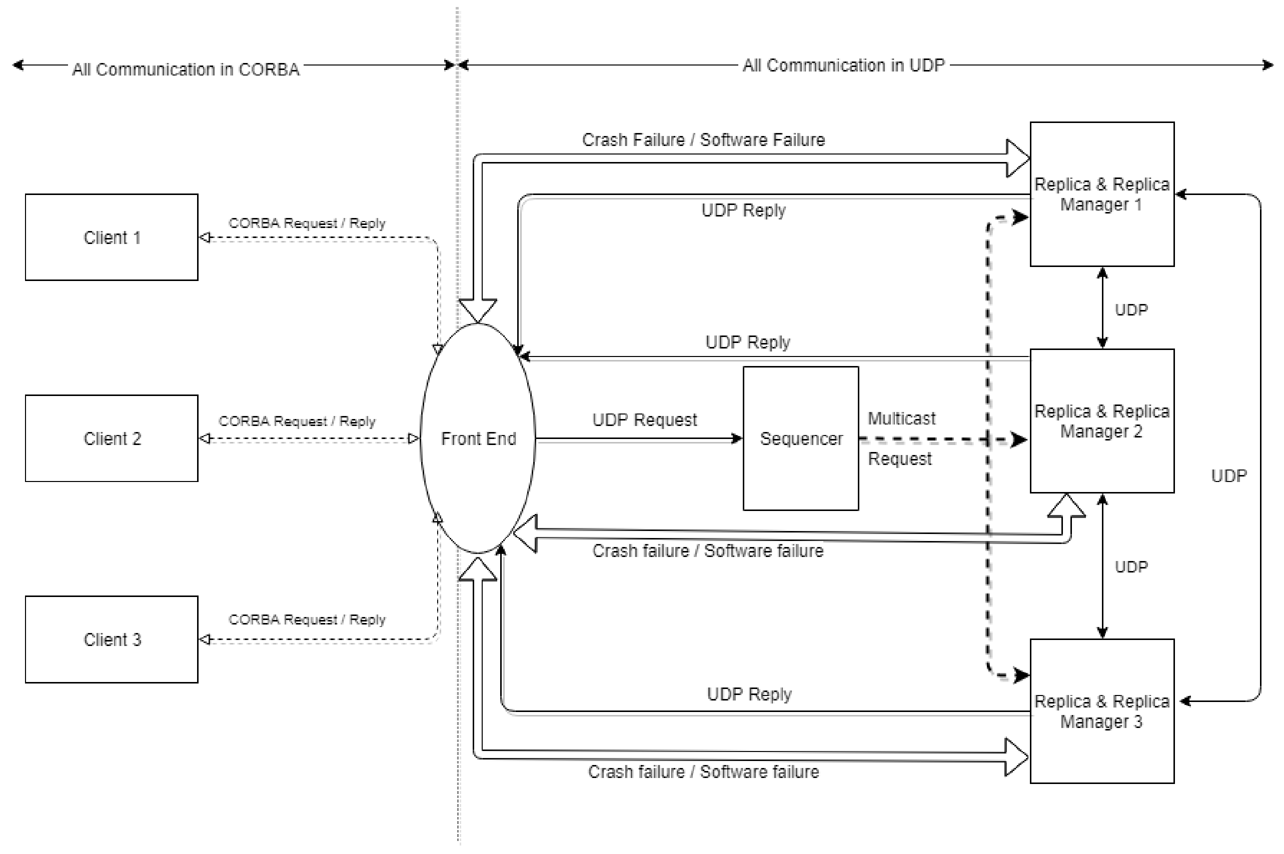
Project Design Document

Yogesh Nimbhorkar: 40093384
Concordia University – Montreal, Quebec, yogesh007@live.in

Girish Kumar Kadapa: 40083533
Concordia University – Montreal, Quebec,

Mayank: 40101857
Concordia University – Montreal, Quebec,

ARCHITECTURE DIAGRAM


All Communication in CORBA
All Communication in UDP
Client 1
Client 2
Client 3
CORBA Request / Reply
CORBA Request / Reply
CORBA Request / Reply
Front End
UDP Request
Sequencer
Multicast Request
Crash Failure / Software Failure
UDP Reply
UDP Reply
UDP Reply
Crash failure / Software failure
Crash failure / Software failure
Replica & Replica Manager 1
Replica & Replica Manager 2
Replica & Replica Manager 3
UDP
UDP
UDP

DISTRIBUTED EVENT MANAGEMENT SYSTEM

Modules in the project

1.**Client**:

It receives the request from the end-user and once the end user selects the required function, client resolves the associated FE (Front-End) by the given name in the registry, makes a call to the remote method in the CORBA IDL interface which in turn calls the interface implementation to do the requisite function.

2.**FE (Front-End):**

It has the method definition of all the methods that the client invokes upon user selection. It frames the UDP message with respect to each request it receives and forwards it to the sequencer. It receives the reply from all the replicas, then finds the best match among the result and forwards it to requesting client using CORBA reply. If front end does not receive a reply from any replica, it then waits for a particular time and suspects that a replica has crashed. It informs a replica manager about a suspected crash and then replica manager restarts the replica. Similarly, If front end has not received a correct response from some replica, it suspects a software failure and thus ask that replica again to restart and validate data.

- All the methods like in assignment 2 to just forward the request to common method.
- Common method does as below:
    1. Create the record in request 3 hashmap for each request (String FE request identifier, RM#, status, result, fail count, req time, res time)
    2. Forwards the request to sequencer.
    3. Software failure Wait logic:
        - Sleep for 2 default seconds
        - Query request hashmap and validate result and update counter accordingly.
        - If any counter has reached > 3 then send udp request to respective RM to restart.
        - Return the correct answer to client.
    4. High availability Wait logic:
        - Sleep for 2 default seconds – this is dynamic
        - Query request hashmap
        - If any record status is blank then send udp request to respective RM to restart.
        - Return the correct answer to client.
        - Update the sleep logic time resp time – req time for all servers (slowest * 2)
    5. Separate thread to receive UDP responses from RM's.
        - Upon response , decode the response as mentioned above will update status , result and resp time for respective hashmap.

3.**Sequencer:**

The sequencer receives the UDP request from the Front-End. It appends a unique sequence number on the UDP request message and multicasts the UDP request to all the Replicas.

4.**Replica Manager:**

There will be three different hosts and each host consists of a Replica along with a Replica Manager. Each Replica holds the three Servers (Montreal, Toronto and Ottawa) implementations. Each Replica Manager will manage a Queue to hold the unique sequence numbers so that all the replicas will follow the same order to execute the client requests, thus fulfilling the total order of execution of requests.

In case of software failure where-in one of the replica sends an incorrect result, then the Front End will send UDP messages to all the Replica Manager informing about the faulty replica. If the Front End receives incorrect result from the same replica for 3 times, then the Replica Managers communicate with replica via UDP messages to replace the faulty replica.

In case of crash failure, the Front End will inform the Replica Managers through UDP messages about a suspected replica crash, if the Front End does not receive any response from that replica for a considerable amount of time. The Replica Manager will communicate with replica via UDP messages to replace the crashed replica with another working replica.

- Receives request from sequencer and forwards it to replica.
- Unique identifier is retained in the response and along with it append RM's unique identifier to identify which RM worked on this request.
- Logic to stop and start replica.
- Forward the replica response to FE receiver thread. (UDP)
- Have a queue to store every request as it is, so in case of system crash or software failure, good RM can send all requests occurred so far in the same order to bad RM to correct its state.

5. **Replica:**

- Execute the requests and send response to RM.

Contributions:
FE- developed by Mayank: 40101857
Replica Manager- developed by Yogesh Nimbhorkar: 40093384
Sequencer - developed by Girish Kumar Kadapa: 40083533

Integration and testing performed by all team members.