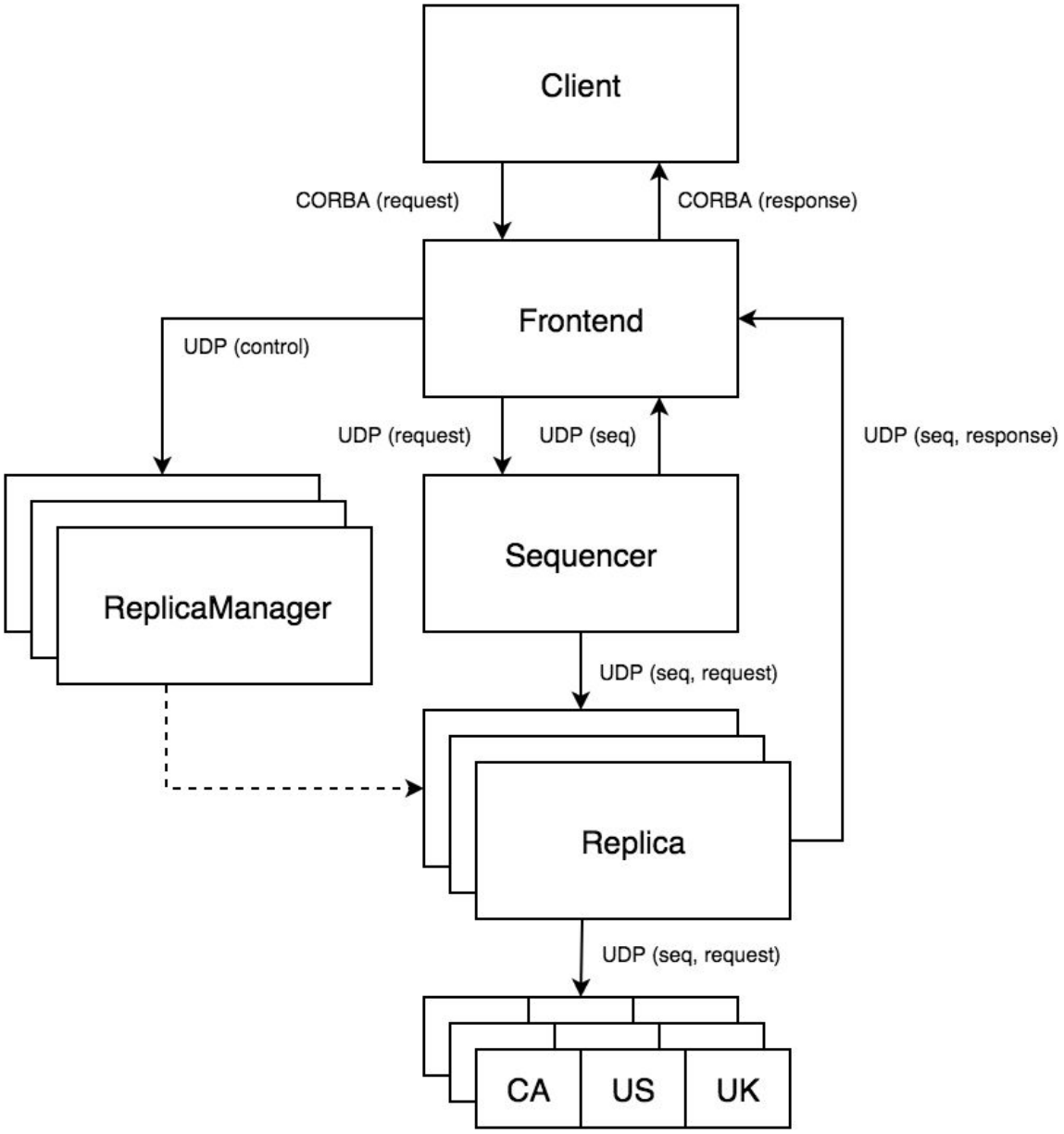
Project Design

https://github.com/g-harel/SOEN423P

Name and Student id	GitHub id	Student Id
Charles-Antoine Hardy	Winterhart	27417888
Andrés Vazquez	andavazgar	40007182
Gabriel Harel	g-harel	40006459
Christopher McArthur	prince-chrismc	40004257

1. The Overall architecture



2. Different Components in the system

Client

The client will be one of the team member's existing clients which will speak with the frontend using CORBA. On startup, it will validate the manager ID and lookup the frontend remote object implementing the regional server interface.

Frontend

The Frontend will register itself as a regional server as the remote object with CORBA and wait for requests from the client. Upon receiving a request, they are forwarded to the sequencer. The frontend then waits for the replicas to send their responses. The frontend can then make a decision about which response is correct and send it back to the waiting client. In the event that a replica is detected to be faulty, the frontend will send a message to the replica managers informing them about a fault.

Sequencer

The sequencer will receive requests from the frontend and give them a sequential ID. This id should be auto-incrementing and will act as a way to ensure all replicas execute the operations in the correct order. Once each request has an ID, it is multicast to all the replicas. The locations inside the replicas that should receive the request is determined by the request's manager ID. The sent message is also stored in a history which can be played back in the even of a replica failure.

Replica Manager

The replica manager manages a single replica (representing a single student's implementation). This means it is responsible for restarting its replica if the frontend thinks it is misbehaving. It will also need to know how to seed new data into a restarted replica so that it is in the same state as the others.

Replica

The replicas should make sure to execute requests in the correct order, using the ID from the sequencer to ensure proper operation. The replicas will also act as adapters for each student's implementation, normalizing the response sent to the frontend to the same type using a common interface that each team member has adapted to their design

Center Servers

Center servers represent concrete location implementations of the application, they are each unique implementations of the same set of actions and store the data they need to complete these actions. To facilitate the agreement on the record IDs, requests which create a new record should use the sequencer-generated ID as the numeric part of the record ID.

Address Book

The address book is a shared data structure (same class in all jars) which handles the system's address discovery needs. For example, a reliable UDP message can be sent to any entity in the system using their address book entry. These values cannot be changed at runtime to reduce complexity.

Reliable UDP

Being able to reliably send messages with UDP is a core requirement for communication between multiple parts of the system. This custom protocol will need to be shared by the multiple components and their implementations and be able to multicast to groups of addresses in order to enable sequencer-to-replica and frontend-to-replica-manager messages.

UDP Payloads

Because only communication between the client and the frontend is going to be using CORBA, we will use specially encoded UDP payloads to represent different types of requests. These requests need to be generated by the frontend and understood by the center servers. Responses will also need to be represented as they are sent from the central servers and need to be transformed into CORBA responses by the frontend. These messages will include the target address, operation type, destined location and, of course, data.

3. Communication between components

Client to Frontend

CORBA. The client sends the commands to be performed to the frontend. These commands are meant to be performed on the Center Servers. As a result, the commands are forwarded to the Sequencer for further processing.

Frontend to Sequencer

UDP Unicast. The front-end sends the messages received from the Client to the Sequencer. All these messages will receive a Sequence ID and forwarded to the Replica Managers.

Sequencer to Replicas

UDP Multicast. The sequencer will generate a Sequence ID for every message received from the Front-End. It will later create UDP packets with the Sequence ID and the message and send them to all the Replicas.

Replicas to Center Server

UDP Unicast. The Replicas forward the command to be performed to the appropriate Center Server depending on the Manager that sent the request. If the manager is Canadian (e.g.: CA1234), then the command will be forwarded to the CA Center Server.

Replicas to Frontend

UDP Unicast. All the replicas will send the result they computed to the Frontend individually. After a decent amount of time, the Frontend will have received most, if not all of the responses, and will be able to detect if an error has taken place.

Frontend to Client

CORBA. The Frontend will reply to the Client by using CORBA.

Frontend to Replica Managers

UDP Multicast. If the Frontend detects an error (software failure or process crash), it will inform all the Replica Managers about it.

Replica Manager to Replica

UDP Unicast. The Replica Managers check if the replica has crashed or not. If it has, the Replica Manager responsible for that Replica will replace it with a new one.

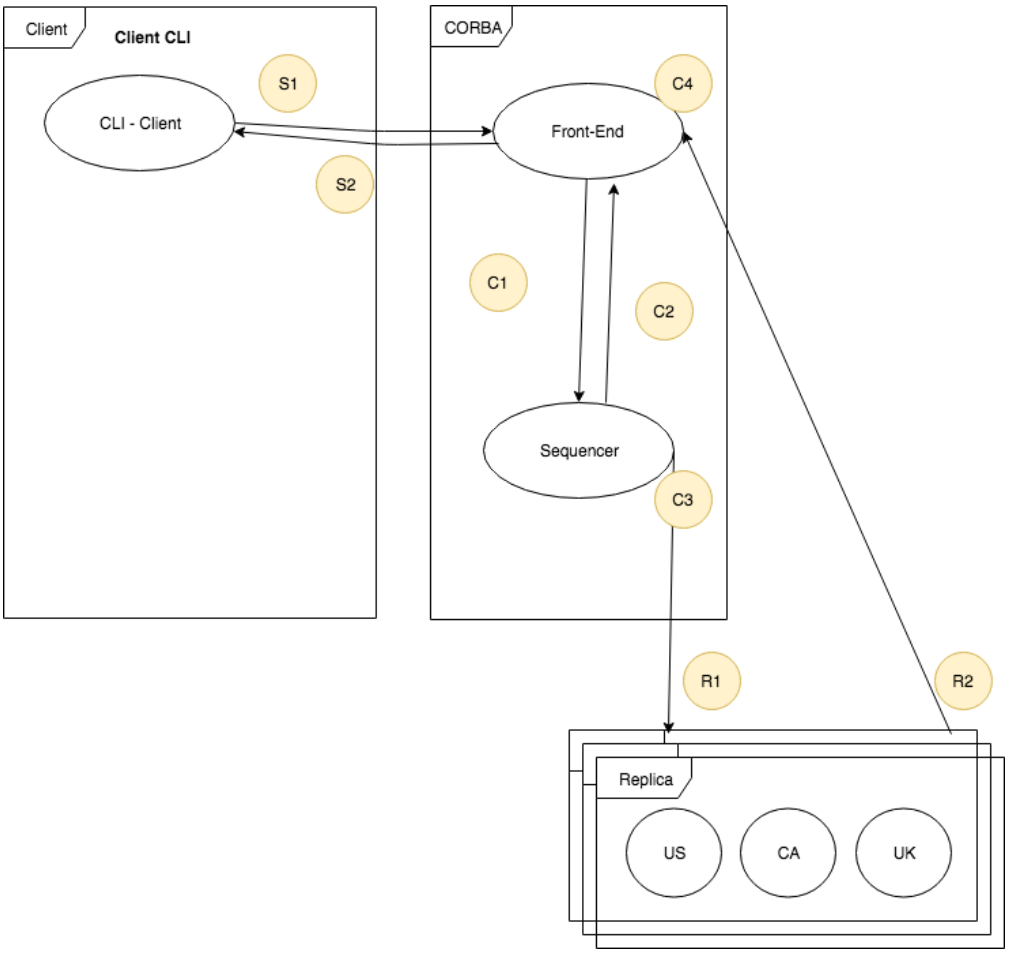
4. Overall workflow (Complete dataflow)

Our overall system is described by the data flow graph below. This graph is divided into three layers a client (CLI) layer, a CORBA layer, and a replica layer. Communication between client and the CORBA layer is made with CORBA. The communication between replicas and CORBA is made with UDP communications.

Data Flow Operations

S1: Sends client request to Front-End (with CORBA)
S2: Receives response from Front-End (with CORBA)
C1: Redirect the client's request to the Sequencer (internal object communication)
C2: Informs back the Front-End with the id given by the Sequencer (internal object communication)
C3: Sends request to the replica (with UDP)
C4: Waits to receive the answers from the replicas (with UDP)
R1: Receives the request from Front-End (with UDP)
R2: Sends back a response to the Front-End (with UDP)

Dataflow Communication Graph



5. Recovery from failure

Software (non-malicious Byzantine) failure

- If any one of the Replicas produces an incorrect result, the Frontend informs the Replica Manager about it so that Replica can be restarted.
- If the same Replica produces incorrect results for 3 consecutive client requests, the Replica Manager replaces that Replica with a new one.

Single process crash failure

- Single process failure will lead to the replica giving incorrect data or timing out with the frontend. This will lead to the frontend asking the associated replica manager to restart the failing replica.

Error recovery

- The sequencer will keep an in-memory log of all the sent messages. After a replica is restarted, its replica manager will ask the sequencer to replay the log to its replica for it to catch up with the state of the other replicas.

6. Distribution of work between students

Task	**Student**
Front-End (Student 1)	Andrés Vazquez
Replica Manager (Student 2)	Charles-Antoine Hardy
Failure-free Sequencer (Student 3)	Gabriel Harel
Test Cases + Reliable UDP (Student 4)	Christopher McArthur