Reliable Queueing System

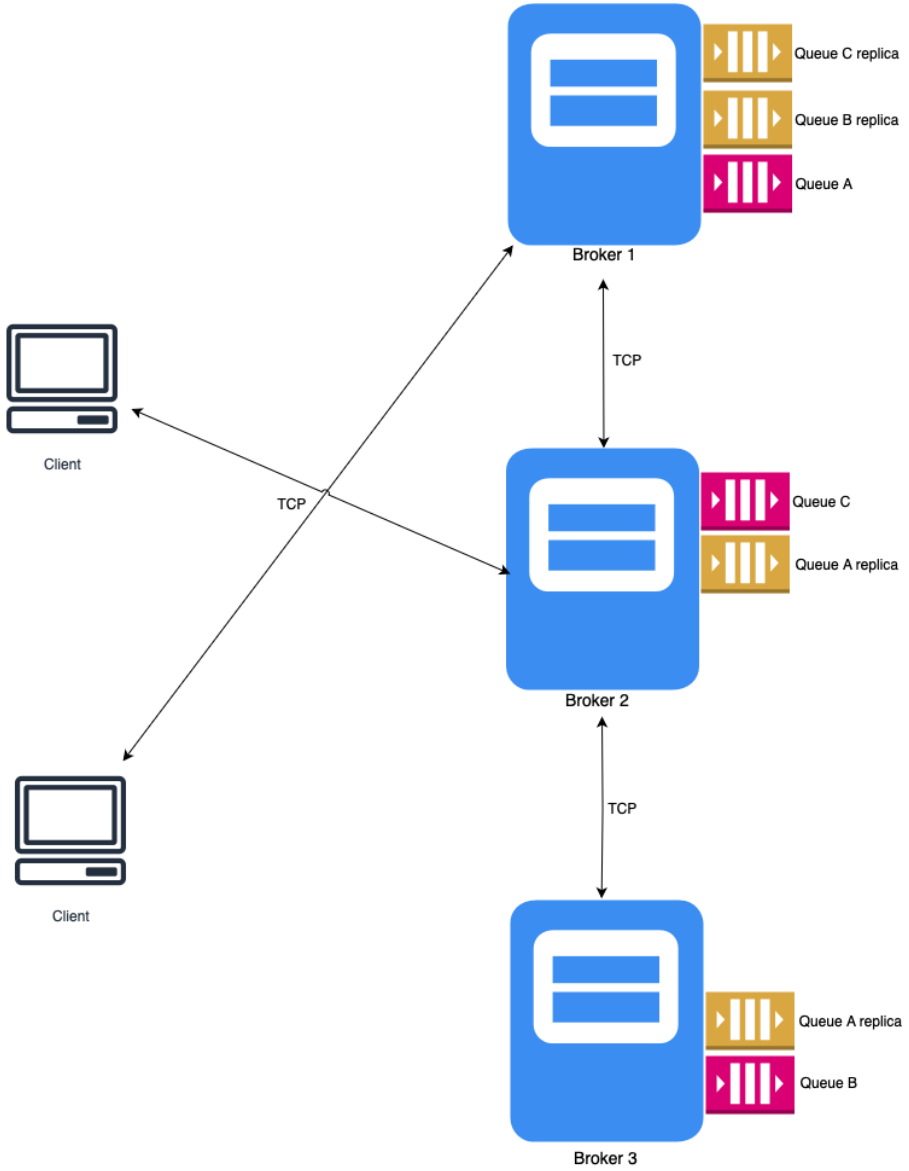
Java Implementation

Bahri Alabey-11047308 & Furkan Bulbul-11053014

GitHub Repository

What is Our Queuing System?

- Distributed message queue with replication and fail tolerance.
- Each queue has a **leader broker** and a set of **follower brokers**.
- **Read/write actions** on a queue are **replicated** from leader to followers.
- The system handles **broker failures** with **leader election**.
- Clients can **resume operations** from the same point after a failover.



Software Solutions

- Java 23
- Networking
  - UDP Multicasts
    - Broker multicast group
    - Client-Broker multicast group
  - TCP Sockets
    - Persistent connection between brokers
    - Client-broker communication
- Concurrency Model
  - Thread pools
  - Synchronized data structures
  - Locks

Inter-Broker Multicast

- Discovery
- New queue registration
- Leader announcement

Client-Broker Multicast

- Broker discovery

Broker to Broker TCP Messages

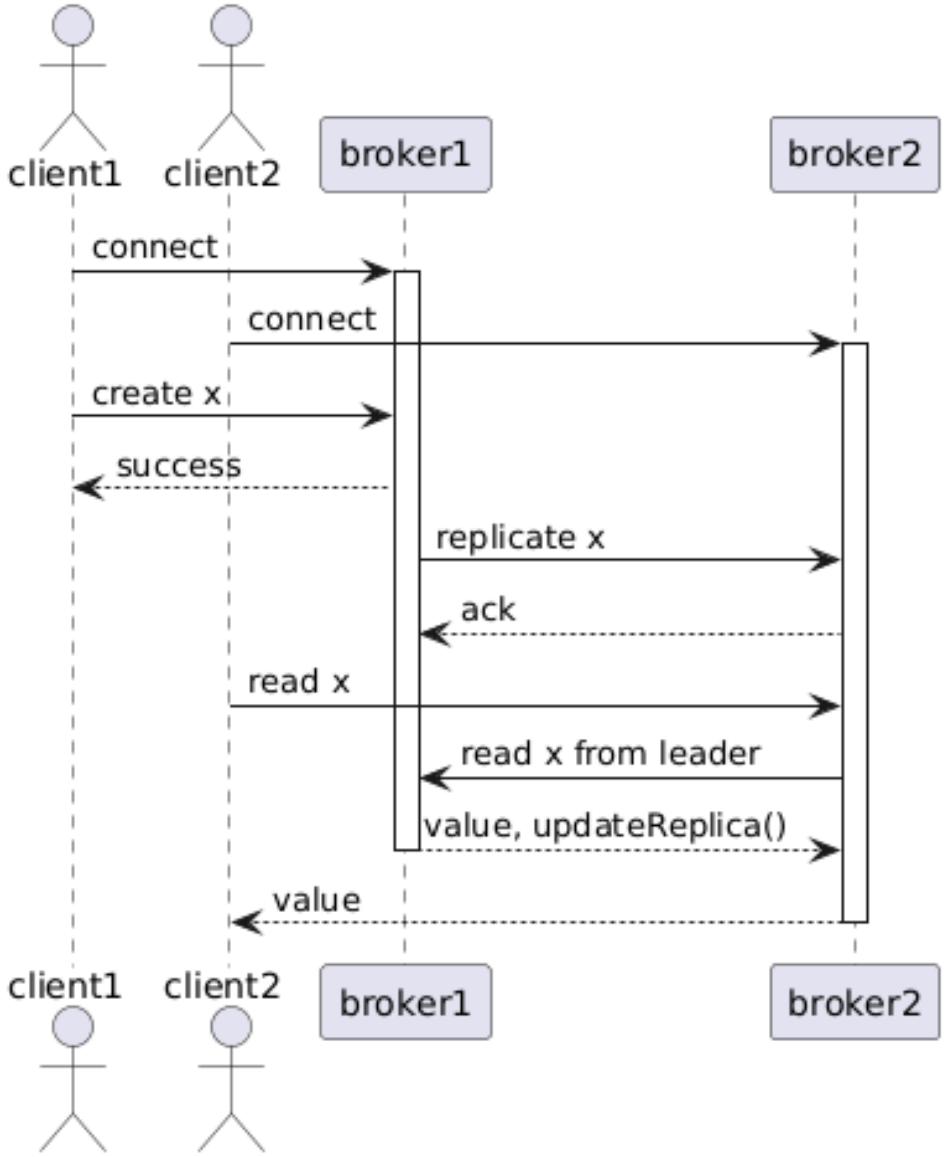
- Replication: Creating a replication of a queue
- Write Replication: Updating replication on write
- Read Replication: Updating replication on read
- Broker Read: Forwarding read request to leader of that queue
- Broker Write: Forwarding write request to leader of that queue
- Healt Check: PING sent to followers for replications
- Election: Starting election on failure of the leader
- Vote: Sending vote

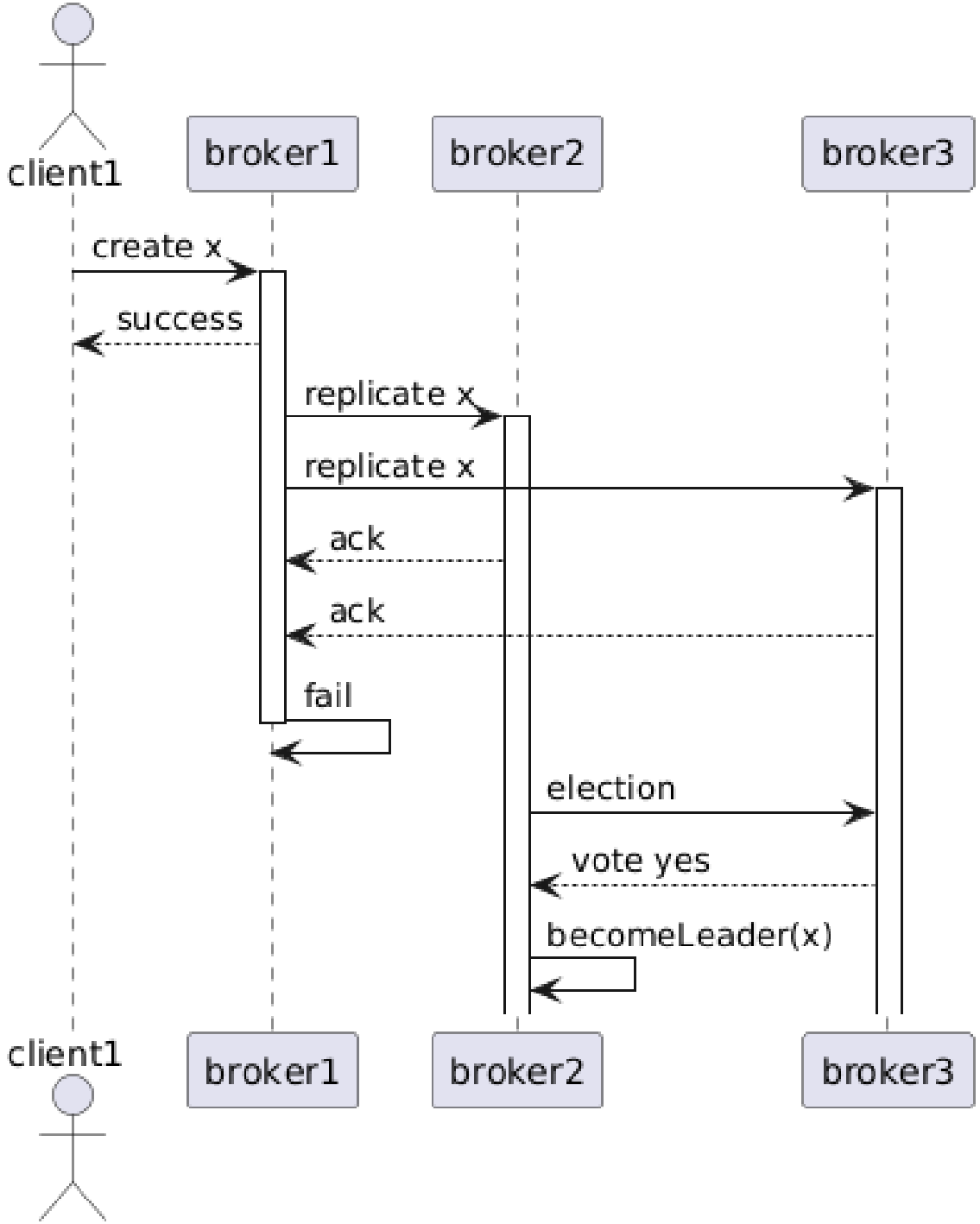
Client to Broker TCP Messages

- Create: Creating a queue in the connected broker
- Write: Writing value to the queue
- Read: Reading value from the queue
- Connect: Connecting to a broker
- Disconnect: Disconnecting from a broker

Runtime Architecture Replication - Read From Non Leader

- Clients can operate on the queues without connecting to their corresponding leader.
- This is done by forwarding client request to leader of the queue.





Runtime Architecture – Leader Election

- Client creates a queue, after replicating it the leader fails. Two followers starts an election to choose a new leader.
- After election finished, the result will be multicasted to all brokers. Client can resume operating on the queue.

Reliability of Our System

- (N+1)/2 broker quorum guarantees message durability – queues remain available even if up to N/2 brokers fail.
- Raft-inspired election:
  - Term-based voting (ELECTION/VOTE messages) for conflict-free leader transitions.
- Healthcheck-Driven Failure Detection.
- Failed replication attempts are retried with exponential backoff.

Possible Improvements

- We assumed that all brokers are started at the beginning, but this might not be the case. After discovering a new broker, we might send the current state (queue owners, clients offsets) to newly discovered broker. This also solves the issue when a failed broker is re-initialized.
- After leader realizes one of the followers failed, it can grab one non-follower broker from the known brokers set and make it a follower. It'll give the system more reliability.
- Client connects to brokers manually by using index, this is done for better debugging purposes, from the viewpoint of client, it shouldn't matter to which broker its connected to. Auto-connection might be added.
- Algorithm that distributes queue replications to other brokers might be implemented to improve reliability.

Thank you!