Distributed Systems Project

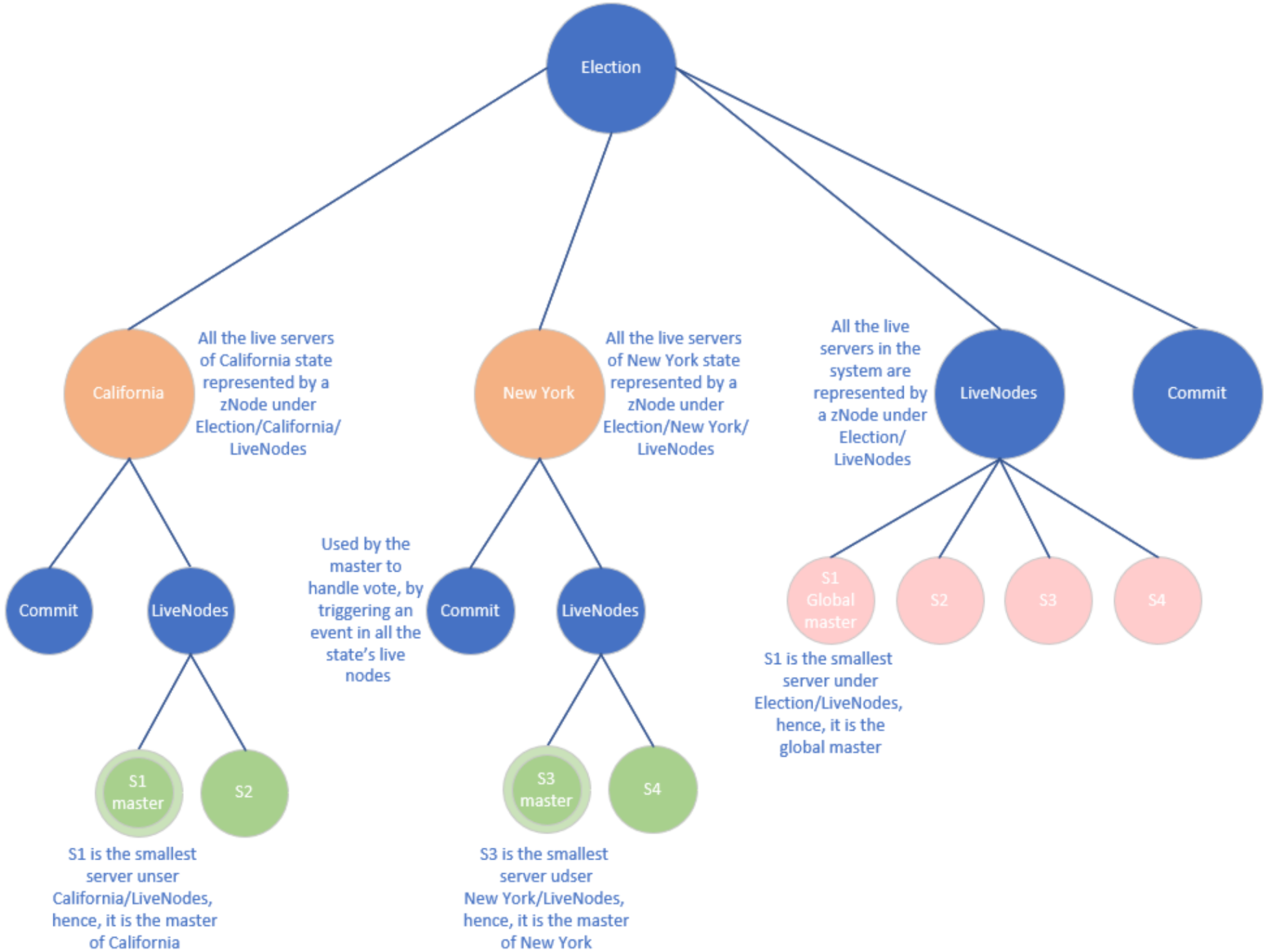
GAL SHALOM & RON YITZHAK

Introduction

- Implementing a distributed Elections system for the USA
- Each state represented by a shard of nodes
- Each node is a valid elections server
- Using Zookeeper to implement distributed services
- Using Docker for running and testing
  - Containers for the replicated zookeeper
  - Containers for the election's servers
  - Containers for clients
  - A Container for Committee Client

Design Overview

- We used Zookeeper to achieve leader election, group membership, failure detector and atomic broadcast
- The zNode tree structure:
  - Under the root (**Election**/) there is a 'LiveNodes' zNode, a 'Commit' zNode, and one zNode for each state
  - Under each state there is a 'LiveNodes' zNode and a 'Commit' zNode
  - '**Commit**' zNode – used to achieve atomic broadcast
  - '**LiveNodes**' – all the live servers have a zNode under the global 'LiveNodes' zNode, and under the state 'LiveNodes' zNode
    - Used to achieve leader election, group membership and failure detector


Election
California
All the live servers of California state represented by a zNode under Election/California/ LiveNodes
New York
All the live servers of New York state represented by a zNode under Election/New York/ LiveNodes
All the live servers in the system are represented by a zNode under Election/ LiveNodes
LiveNodes
Commit
Commit
LiveNodes
Used by the master to handle vote, by triggering an event in all the state's live nodes
Commit
LiveNodes
S1 master
S2
S1 is the smallest server unser California/LiveNodes, hence, it is the master of California
S3 master
S4
S3 is the smallest server udser New York/LiveNodes, hence, it is the master of New York
S1 Global master
S2
S3
S4
S1 is the smallest server under Election/LiveNodes, hence, it is the global master

Design Consideration

- In the process of implementing the exercise we considered three major models of implementation
  - Quorum-based
  - DHT-based
  - Leader-based
- Finally we chose the **Leader-based** model mainly because of the complexity and simplicity

Quorum-based

- **Pros:**
  - Fast vote's propagation to state's nodes
    - No master to pass through
    - Only need a quorum size of acknowledgements
  - Simple synchronize mechanism
    - No need to implement a failure detector or a group membership services
- **Cons:**
  - Read operation is slow – need to wait for a quorum of servers
  - Complicated election status calculation with high complexity
    - Need to get all the votes from all the servers and calculate for each vote the most updated value

DHT-based

- **Pros:**
  - If all state's voters hashed to the same range of hashes:
    - Each state is considered as a range with K back-up nodes
    - Routing the voters is simple – based on the hash
- **Cons:**
  - Mapping all voters of the same state to a specific range requires from us to design a special and specific hash function
  - Complicated elections status calculation with high complexity
    - Need to get all the votes from all the servers and calculate for each vote the most updated value

Leader-based

- **Pros:**
  - $O(1)$ complexity for a simple read operation
  - Simple elections status calculation – all the state's nodes have the same votes view
- **Cons:**
  - Write operation is slow
    - Need to wait for all the live servers
    - Need to pass through a leader (state's leader or a global leader) – bottle neck

Services Implementation

Failure Detector

- Our LiveNodes contains ephemeral sequential zNodes
- Each one has been created by its server, and it is destroyed when the connection from this server is no longer exists (ephemeral)
- This means for us that the server is not available anymore, because its zNode is no longer exists too

Leader Election

- Each state has a leader that represents by the smallest sequential zNode under Election/<state>/LiveNodes
- This zNode keeps the server's identity as data, hence the identity can be retrieved easily
- Once a leader is elected, all the live nodes in the shard start listening on the leader's zNode
- Once a leader fails, its zNode under /Election/<state>/LiveNodes is deleted
- The deletion triggers av event in all the state's nodes to make them choose a new leader

Group Membership

- The Election/<state>/LiveNodes namespace used as a view of group membership
- When a server from state A wants to forward a request to a server from state B, it simply chooses random zNode under B's LiveNodes zNode
- The chosen zNode contains the server's identity
- Server A then forward the request to B's server

Atomic Broadcast

- Master propagates a vote request to all the other servers:
  - It catches a lock until the vote request has been sent to all the live servers of the state
- Each server that receives the request, saves it locally without handling it
- Each server has an Atomic Boolean used to indicate whether it has a pending request to handle
- After the master sent the requests to all the state servers successfully, it changes the Election/<state>/**Commit** data to trigger an event on all the state servers
- This event is used to commit the previously sent request on the state servers **atomically**

Demo

Questions?