Anna Rift
Logan Hurd
CS 455
2022-04-08

# CS455 Project 3 Design Whitepaper

**General:**
Each server will assign itself a randomly-generated UUID on startup. There is always exactly one coordinator (between elections). Clients perform all actions, both read and write, on the coordinator. Coordinator updates are replicated out to the other servers, maintaining the order the coordinator sees everywhere (coordinator assigns sequence numbers). All servers have a background timed thread sending a ping to the coordinator every SYNC_PERIOD seconds containing the timestamp of the latest action it has seen.

**Client server model:**
One-shot clients with multiple separate, linked servers. The primary reason for this is simplicity, because I don't think there would be any additional benefit on either client or server side to using servants. Additionally, one-shot use is more accurate to the idea of an identity server, which would just be getting periodically consulted by other programs.

Clients take as an argument a list of all servers they should be able to use, and ask servers from this list at random who is the coordinator until a response is received.

Each server takes as an argument a list of other servers in the group, excluding itself. It's acceptable for the server to be started with a list of servers that are not yet up, for example if it is the first server started. This will simply lead to those other servers being left out of its election on startup.

**Communications:**
Avoid using IP multicast, so we can easily use this on AWS or any network. Use RMI for all connections including between servers. Just one RMI method, read and write requests will be encapsulated as an object passed to this message. These "action objects" contain complete information about a performed action, and can be logged/replayed/etc.

**Elections:**
Bully algorithm, winner is the highest server UUID. Each server initiates an election upon startup (with the assumption that it may be recovering from failure). Servers detect the coordinator is down if their ping is not replied to within ELECTION_WAIT. ELECTION_WAIT is also used at every point in the bully algorithm that calls for waiting for a response for a certain amount of time.

**Consistency:**

Use the remote-write protocol, with the coordinator being the primary for all data items. Writes block until replication has completed and acknowledgement received from all replicas. To ensure sequential consistency between multiple requests from the same client, requests are timestamped.

The coordinator maintains a circular buffer of the last ACTION_LOG_SIZE action objects. The coordinator will respond to ping messages (which contain the replica's last update timestamp) with synchronization info, which is one of the following:

1. empty, if the replica is up-to-date
2. if the replica is behind but we know that they are only behind by a set of actions available in the action log, the list of actions to perform
3. if the replica is too far behind to catch up from the action log, the entire data set

Upon receiving catch-up data, the replica immediately applies it. The longest a replica can remain unknowingly out-of-date ("inconsistency window") is SYNC_PERIOD, plus the time it takes to request, receive, and apply the catch-up data. Transferring the entire data-set is the worst-case, and the motivation for having a small log of actions is that it can be avoided most of the time. In theory the ACTION_LOG_SIZE should be large enough that we are unlikely to incur the penalty of transferring the whole database, except in the case that a new server is being brought online after others.

**Synchronization:**
We will implement Lamport timestamps. Total ordering accomplished by attaching Lamport timestamp from coordinator to all actions before replication. Vector clocks are unnecessary since all actions go through the coordinator which sets the canonical order of all actions.

**Initial Parameter Values:**
These are estimations of a good value and could be tuned to change performance.
SYNC_PERIOD: 5 seconds
ACTION_LOG_SIZE: 3 items
ELECTION_WAIT: 2 seconds