What type of client server model will you use? One-shot client with separate servers. Or servant model where clients are servers as well and are always on.

The clients will be thin, one-shot clients and the servers will be separate from the clients. We'll run each server in its own Docker container.

Coordinator: How to elect? How to detect if one is down or missing?

Our servers will use the Bully Algorithm implemented such that each server generates a large random number when starting, then they all compare numbers, the largest number becomes the coordinator

The secondary servers will use heartbeats and timeouts for communicating with the primary server. If a secondary server does not receive a response from the primary within the timeout period, it will initiate a vote for a new primary.

The clients all have the list of all servers at startup, then they just pass the command to each server until either the command executes or the list of servers is empty.

Consistency model: Why did you choose a specific model? How will you implement it?

Consistency will be implemented by the MongoDB replica set. There are three servers in the cluster, with one primary and two secondary servers. The primary server writes data and sends the write commands to the secondaries via its operations log or oplog. The secondary servers execute the commands in the oplog until they are in a consistent state with the primary. The replica set elects a new primary if it cannot communicate with the primary for 10 seconds (default).

What is your inconsistency window?

The coordinator processes requests serially. A single request is not finished until mongoDB returns a result. Therefore, the inconsistency window is 0.

Server synchronization:

Lamport timestamps or Vector clocks:
This is implemented by the MongoDB cluster

Granularity: do we copy the entire database when syncing or do we log messages and use checkpoints to reduce communication bandwidth required?

The MongoDb Replica Set maintains consistency by executing the commands in the operations log of the primary server across all secondary servers.

https://docs.mongodb.com/manual/replication/

https://docs.mongodb.com/manual/core/replica-set-secondary/

https://docs.mongodb.com/manual/core/replica-set-elections/

https://docs.mongodb.com/manual/core/replica-set-oplog/