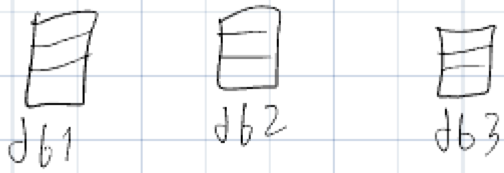
Engineering Notebook Entries

connect ...



add through connections

db1 add connection => db2, db3
waits no response
db1 is master

Later db2, db3 connect
through AddConnect

AddConnect tells other server to
to connect to it.
After connect all is done
Find master in all dbs

when db disconnects,
db relays info to other
dbs and Find_master restarts

Client calls leader server
to send message
leader server sends message
to other servers

---

cases

1/2 ...

3/22/2025

- Implemented the `isMaster()` functionality that allows servers to check if they should be the master
- Created the basic proto definitions for server-to-server communication:
    - `IsMasterRequest` and `IsMasterReply` for leader election
    - `AddConnectRequest` and `AddConnectReply` for establishing connections between servers
- Completed the server-side handler for isMaster requests which determines master status
- Started working on the client-side connection strategy:
    - Client will first connect to any available server
    - Then ask that server who should be the leader
    - This allows for dynamic master server selection based on availability
- Tested basic message passing between two server instances
- Researched Raft consensus algorithm for potential implementation in our distributed system
- TODO: Need to implement proper error handling for connection failures

3/23/2025

- Built the `AddConnect` functionality to establish inter-server connections
- When a server starts up, it attempts to connect to all known servers using the connect_all() function
- Added bidirectional connection logic to ensure all servers maintain connections with each other
- Implemented the leader election algorithm based on Raft principles:
    - By default, the server with the lowest port number becomes the leader
    - If that server is unavailable, the next lowest becomes leader
    - Added timeout mechanisms for election phases
- Designed the heartbeat system for servers to maintain awareness of cluster state
- Added tests for various server startup scenarios

3/24/2025

- Implemented the full master server selection protocol following Raft-inspired election process
- Added the commit request/reply handlers for database operations
- Created handling for server failure detection using heartbeats:
    - Each server periodically sends heartbeat signals to other servers
    - If a server doesn't respond, it's marked as potentially down
    - After multiple failed attempts, the server is considered down
    - If the master server goes down, a new election is triggered
- Added retry logic for client connections with fallback options
- Optimized heartbeat intervals to balance responsiveness and network traffic

3/25/2025

- Implemented the `DisconnectRequest` and `DisconnectReply` functionality for clean server disconnection
- Added proper shutdown sequence for graceful server termination
- Improved the leader election algorithm with timeout handling inspired by Raft:
    - Added exponential backoff for retrying failed connections
    - Implemented deadlock prevention in the election process
    - Created vote request/response mechanisms
- Tested multiple failure scenarios including network partitions
- Added comprehensive logging throughout the system
- Next steps:
    - Talk with alice about ip and port stuff