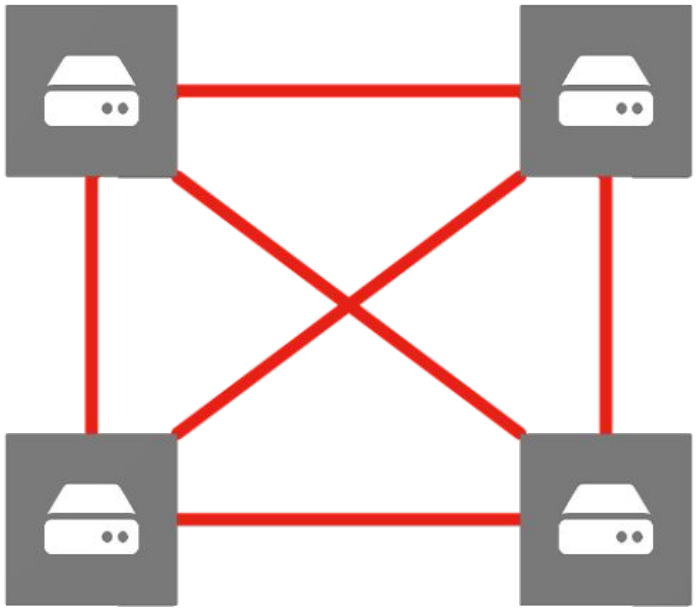
Logical Grid Mesh

Designs Considered - Fully Connected



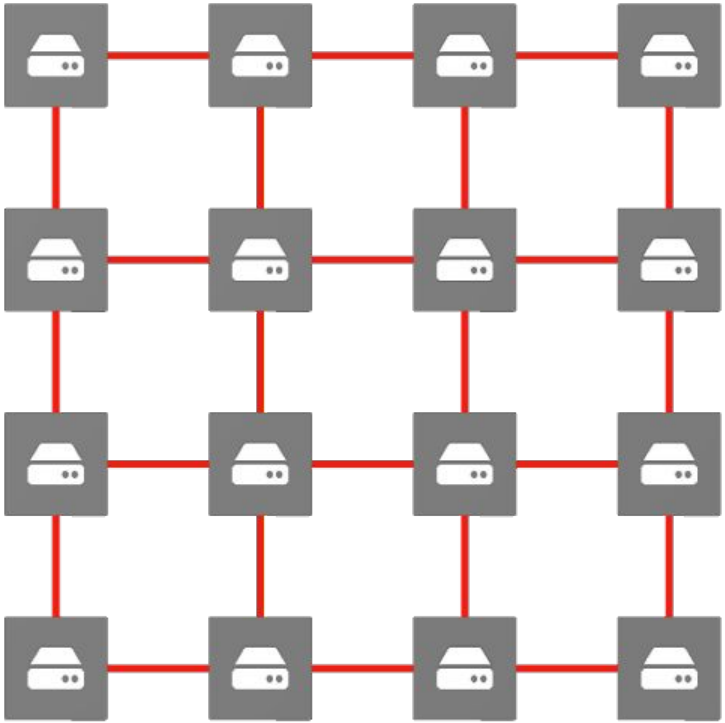
Pros:

- Direct connections with each node
- Low communication latency
- Easy to implement

Cons:

- Too much noise and communication overhead for just maintaining the network
- High resource utilization in establishing connections to all nodes in the network for every new node added
- Heartbeats have to be sent to all other nodes in the network

Grid Mesh Design



Grid Mesh Creation:

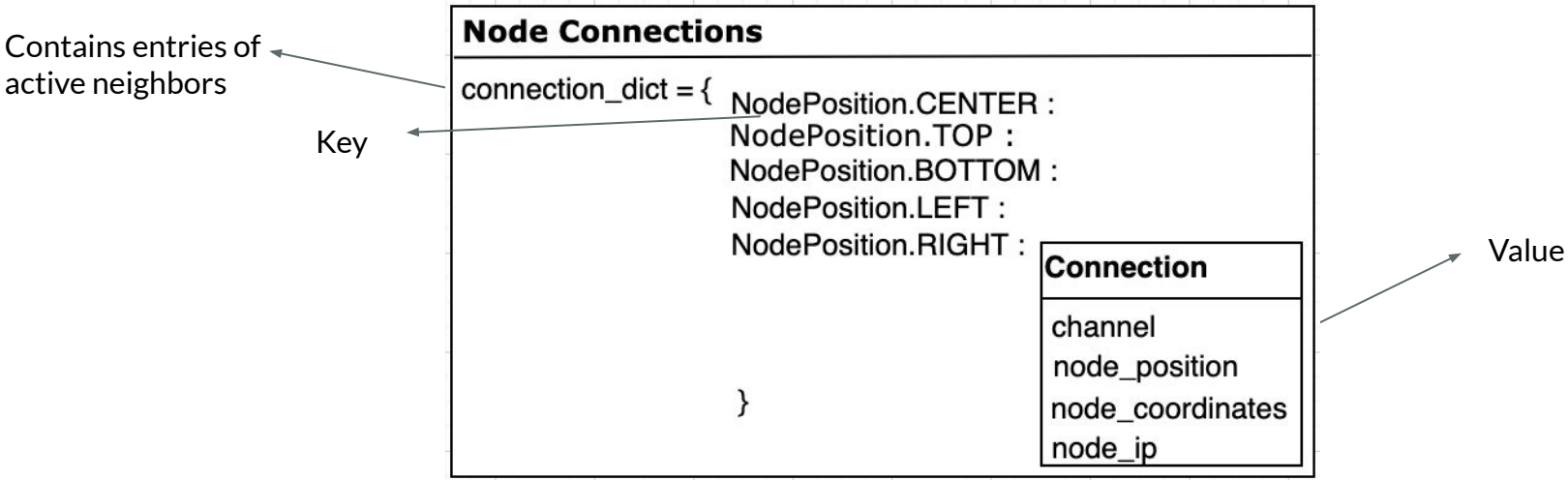
- Nodes are arranged in a grid structure
- Nodes are uniquely identified by their x and y coordinates
- Direct connections with only their neighbor node
  - Keeps the network loosely coupled
- Network will always remain in grid-like structure
  - Network will adapt to addition or deletion of nodes
- Nodes maintain gRPC channel for re-use
- All nodes capable of handling requests for addition of new nodes and deletion of failed nodes
- At most 4 connections per node but can be extended to 6 to create a 3D mesh with minimal changes

Grid Mesh Creation: Algorithm

1. When a new node wants to get added to the network, it will try to greet one of the nodes in the network by invoking `SayHello` gRPC method.
2. The node that received the greeting message now has to assign a position (out of TOP, BOTTOM, LEFT and RIGHT) and coordinates to the new node.
3. The node will first find all its available and unavailable neighbor positions/coordinates by looking at the `connection_dict` in the `NodeConnections` object.
4. After that, it will try to assign a position and coordinates from the available positions in such a way that the mesh structure remains compact after the new node addition.
5. It may also try to get information about its neighbor's neighbor (if required) for assigning a new node a position and coordinates to create a compact grid structure.
6. After it has assigned a new node a position and coordinates, it will send a list of IPs to the new node if the new node has other neighbors where it has been put in the network to make additional connections with them.
7. If the new node gets a list of IPs in the `additional_connections` list, it will inform the other neighbor nodes about its existence so that they know about its new neighbor.

Maintaining channels with Neighbor Nodes

- Maintaining persistent channels with neighbor nodes for reusability
- Each node uses **NodeConnections** object to manage neighbor connection information
    - NodeConnections object uses connection_dict to store connection information of a node with other nodes



Managing Shared Resources

Problem we faced:

- NodeConnections object is modified by both server and client threads and also by several other threads who want to transfer data between nodes.
- Ensuring consistency and stability of NodeConnections object is important

Solution we designed:

- Single thread should modify the NodeConnections object at a time.
- **Applied locks to the critical section** of the code where the NodeConnections are modified.
- Locks are used in NodeConnections.add_connection and NodeConnections.remove_connection

Pulse Module

Problem we faced:

- **Nodes go down frequently** due to irrecoverable exception in the code, network failure, disk failure, memory failure, power outages
- Network should be resilient to these failures and always remain in a stable and consistent state.

Solution we designed:

- Node down if physically disconnected from the network or node process is down
- Used **ping to test for the physical connection** and **grpc.channel_ready_future for ensuring channel connection** with the node and the node process is active.
- Once sure that the node is down, we remove it from active connections.

Logging

- Levels of logging implemented
  - Information logging messages logged using logger.info
  - Error logging messages logged using logger.error
  - Debugging messages logged using logger.debug.
- Classifying logging messages helped in proper analysis of system status
- Significant reduction in logs can be done by turning off certain types of log messages
- Levels of logging implemented
  - Example: After complete system testing, we can set the logging level to logging.INFO and it will silence all logging below the level. As logging.DEBUG is below that level, we won't see debug logs and with that, we can significantly reduce down the log I/O and increase system performance.

Pros and Cons of Grid Mesh Design

Pros of Grid Mesh Design:

- The network is loosely coupled.
- The network will adapt itself accordingly to the addition or deletion of nodes to maintain the compact grid structure.
- Nodes maintain gRPC channels with neighbors for channel reusability.
- All nodes are capable of handling requests for the addition of new nodes.
- Maintaining direct connections only with their neighbor nodes.
- Unlike fully connected mesh topology, the 2D Mesh network highly reduces the overhead on each node as it has to maintain connections with at most 4 nodes.
- As all the nodes are arranged in a grid and have cartesian coordinates, it is easy to locate and find the shortest path based on simple math calculation.
- The structure can be extended to create a 3D mesh with minimum code changes.

Cons of Grid Mesh Design:

- Since each node is not connected to every other node in the network, the time to forward data to a particular node increases.