DESIGN SPECIFICATION

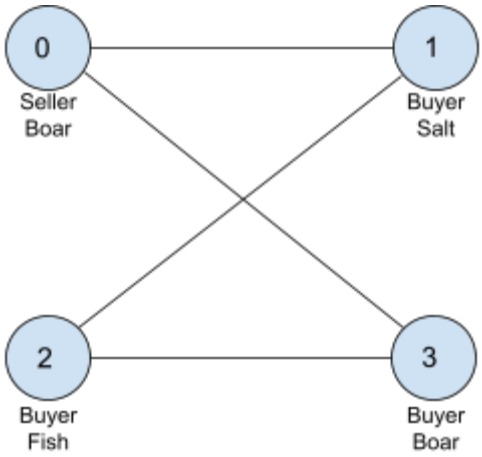
This project implements a peer to peer network with multiple peers depending upon the values of N and K which can be specified in the config file as mentioned in the project requirements. All the peers will be on the same machine based upon the requirements corresponding to 2nd Milestone.

Network Initialization

Based upon the value of N,K and the test-case number, a P2P network is initialized. All possible configurations of this P2P network is listed below -

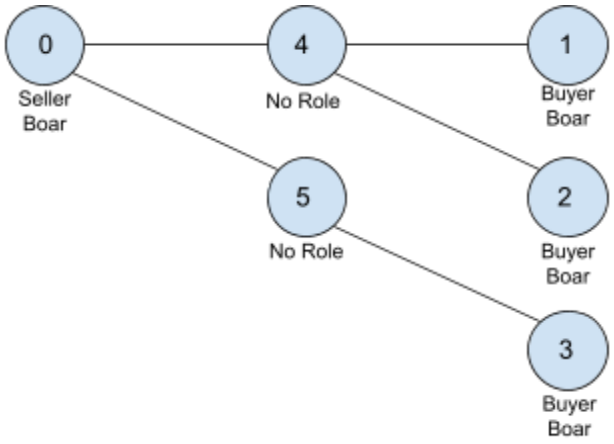
1. **Default Configuration**

   If no testcase parameter is specified, then based upon the value of N and K a random P2P network is initialized. For every peer in N, K peers are randomly selected and are initialized as the neighbors of that peer. Roles are randomly assigned to every peer (Seller or buyer). For the value of N = 4 and K = 2, the following network can be an example of the random initialization.



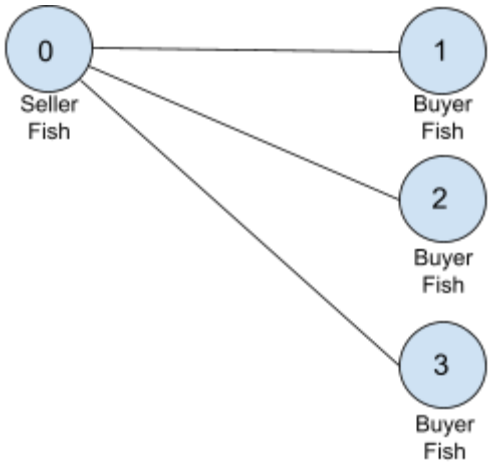
2. **Test Case 1**

In case of Test-Case 1, Based on the requirements (Refer to Test case Doc), the following network is initialized.



3. **Test Case 2**

   In case of Test-Case 2, Based on the requirements (Refer to Test case Doc), the following network is initialized.



Network Execution

The rest of the design remains same as before i.e.
A buyer sends a lookup request to its neighbors which is propagated further in the network through flooding procedure. Once an appropriate seller is found, the reply is sent back from the seler to the buyer through the same path. One the request reaches back to the buyer node, buyer adds the specific seller to its seller list. Similarly, all the sellers who respond to the request of a buyer gets added to the buyer's seller list. Once all the requests come back, the buyer randomly chooses a seller and proceeds for the connection. Once a connection is established between the buyer and the seller, the buyer sends a buy request to the seller and gets a response if it was able to successfully buy the item.

The seller generates m=5 quantities of an item, and once it runs out of items to sell, it restocks an item dependending on different scenarios.

Implementation Level logic

At the code level, we instantiate and deal with peers through the ‘Node’ class. Each peer is instantiated with a peer ID, role, port it listens on, it’s list of neighbours, item it sells or buys and the count of it.

Interfaces of a Peer:

Private:

- **lookup(productname, hopcount):**
    1. This interface is called by the buyer to lookup for the items in the neighbouring peers.
    2. This is achieved by calling the lookup_helper function of the adjacent peer. While calling, it passes productname, hopcount and the array of nodes through which the request has propagated.

- **reply(buyerID, sellerID):**
    1. This interface is called in case the back propagated request reaches back to the buyer.
    2. Upon being called, it adds the specific sellerID to the buyer’s list of all the sellers who responded affirmatively for the item lookup.

Public:

- **lookup_helper(productname, hopcount, <array of traversed nodes>):**
    1. This is a public helper function for private interface lookup.
    2. This interface basically checks if the current node is seller or buyer and acts accordingly.
    3. If the node is buyer, the interface will forward the incoming request to its other neighboring peers(flooding).
    4. If the node is a seller, the interface checks if the item requested matches with the item it is selling. If yes, it responds back to the node from where the request came with its id. Otherwise, it forwards the request to other nodes.

- **reply_helper(<array of traversed nodes>, sellerId):**
    1. This interface is called by lookup_helper of the same peer when the peer identifies itself as a seller of the required item.
    2. This inturn calls the previous node lookup_helper function until it reaches the buyer.

3. Once the back propagation request reaches the buyer, it calls the reply function of the same node to add the sellerID to the buyer's seller list.
4. When the peer(seller) realizes that the inventory is finished, it restocks the items according to different scenarios.

- **buy(sellerId):**
    1. If the peer is a buyer, it calls the buy interface of the specific seller to buy a particular item.
    2. If the peer is a seller, it checks if the item is still present in its stock. If yes, it confirms the transaction and returns. Otherwise, it sends a negative response to the buyer that the item is no longer available.

Taking example of 2 node networks: