CS 677 Lab1 Design Document

1. Architecture:

1.1 Peer:

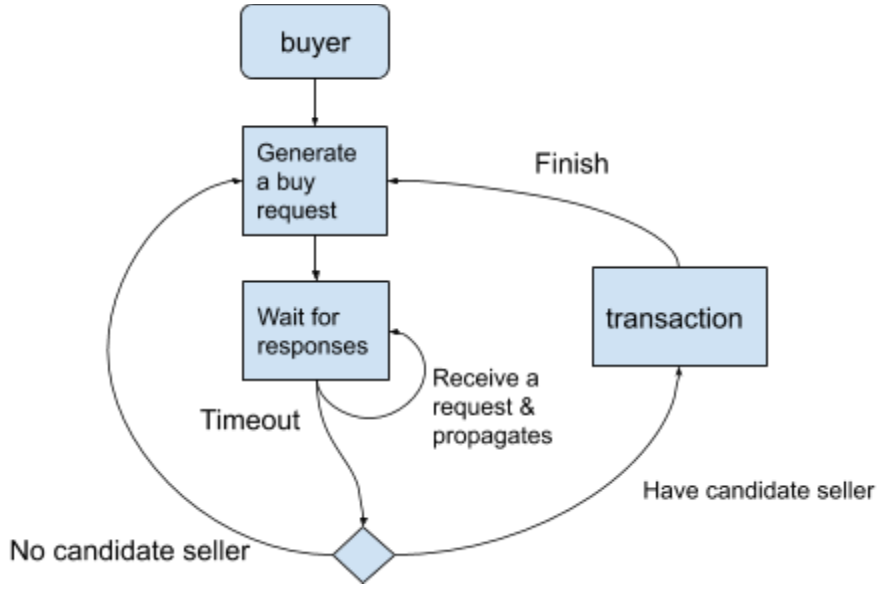
Each peer has a unique peer_id, a role it plays, and a list of neighbors. If a peer plays as a buyer, it has additional attributes such as a target product to buy and candidate sellers who have the product. If a peer plays as a seller, it has additional attributes such as commodity to sell and its corresponding quantity, and a lock for synchronization. If a peer plays both roles, it will have those attributes.

It is worth mentioning that we design a peer always being a RPC server. No matter what role a peer plays, it always accepts incoming requests and handles them accordingly. For instance, a buyer could propagate a “lookup” or a “reply” request to its neighbors when it receives them from its neighbors.

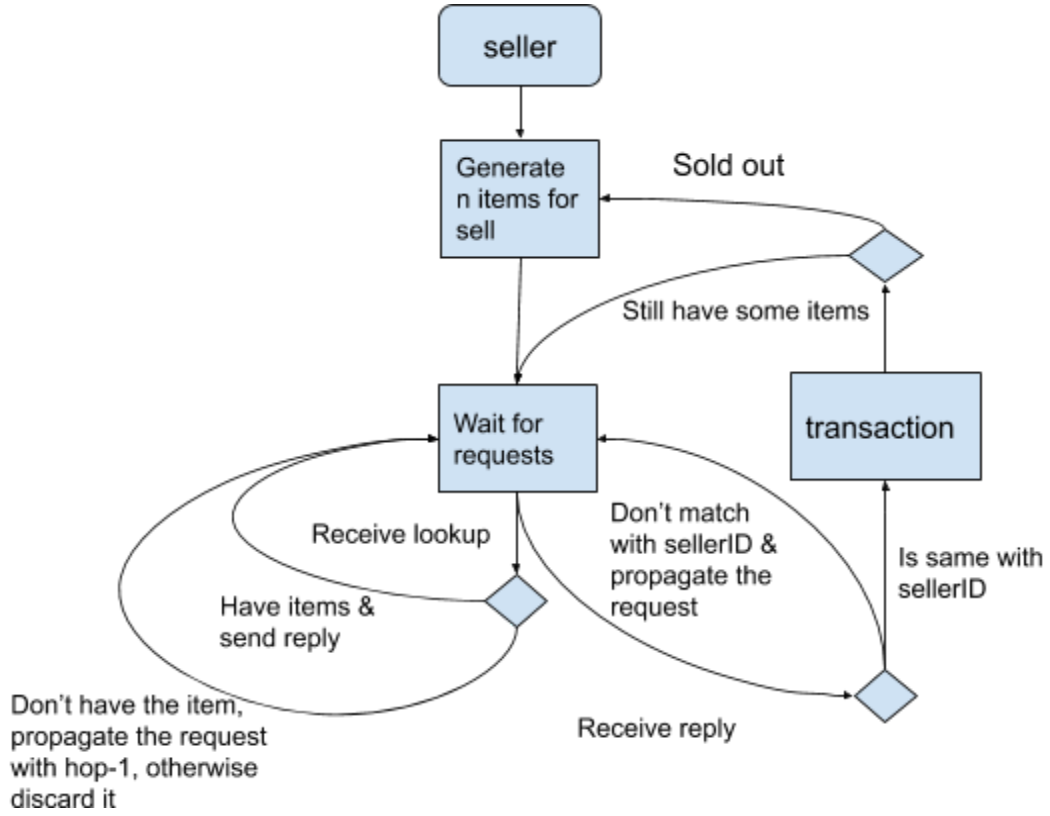
1.2 Role:

Each peer could be a buyer, a seller or both. The transition of states for each role is as below:

State transition of buyer:



State transition of seller:



2. How it works:

When our system starts, the main thread creates other threads that will run as peers. That is, if there are 6 peers to run on a machine, then 6 threads will be created, each stands for a peer. If there are 6 peers to run on 2 machines, then each machine is responsible for creating 3 peers. For each peer, it will create a new thread which is used for running its RPC server. Then, it will initiate several variables based on its role. A buyer randomly generates a target product to buy and a seller also randomly generates commodity to sell and its quantity.

A buyer sends a lookup request to its neighbors, then it starts waiting. If the neighbor of the buyer is also a buyer, it simply propagates the request to its neighbors. If the neighbor of the buyer is a seller, it replies to the buyer if it owns the product, otherwise the request would be propagated to its neighbor. A lookup request is discarded when the hop count is 0. When the buyer stops waiting, it checks its candidate seller list. The buyer will send a buy request to the first seller in the list. If the seller has sold out the product, the buyer would send a buy request to the next seller until it successfully buys the product. If all candidate sellers have sold out their product, the buyer just generates a new target to buy.

The jobs for a seller to do are simple: it generates a commodity to sell or propagates lookup/reply requests. When a buyer sends a buy request, it makes sure that the quantity of its product is correct.

3. Trade-offs:

1) Enabling each peer to be a RPC server reduces the complexity of topology and the blocking condition when clients send requests. However, it requires more system resources to run.

4. Improvements:

1) A RPC server might shut down unexpectedly or refuse to connect especially when all peers are both sellers and buyers in a fully connected network. We need to implement a mechanism that can automatically restart the server as well as setting the RPC server in a better configuration.

2) A peer might reply to duplicate lookup requests from the same peer due to the fact that lookup requests sent from a buyer might take multiple different routes to a seller. This might cause waste of network bandwidth and computing resources. For improvement, a seller can record duplicated requests to avoid the situation.