CS 677 Lab1 Design Document

1. Architecture:

1.1 Peer:

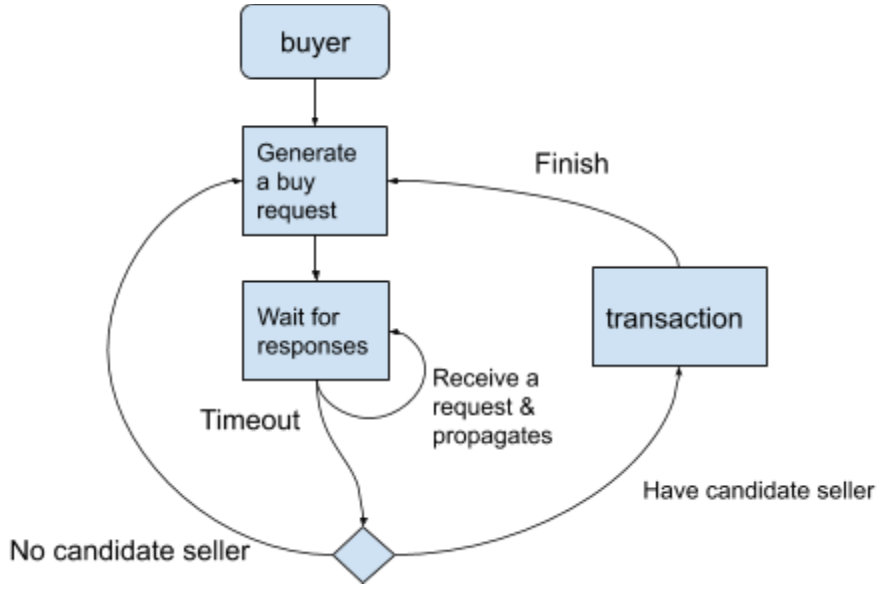
Each peer has a unique peer_id, a role it plays, and a list of neighbors. If a peer plays as a buyer, it has additional attributes such as a target product to buy and candidate sellers who have the product. If a peer plays as a seller, it has additional attributes such as commodity to sell and its corresponding 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 one of its neighbors.

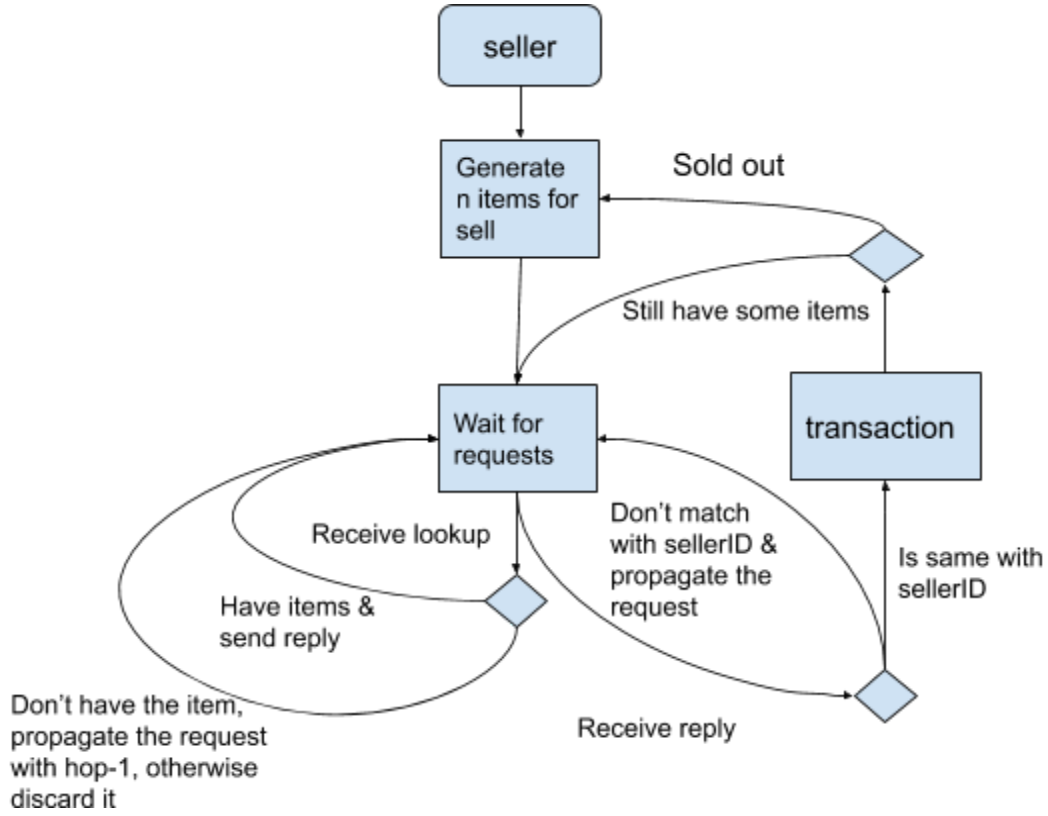
1.2 Role:

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

State transition of buyer:



State transition of seller:



2. How it works:

When our system starts, the main thread creates other threads that run as peers. That is, if there are 6 peers to run, then 6 threads will be created, each stands for a peer. For each peer, it will create a new thread which is used for running a 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 a peer is another buyer, it simply propagates the request to its neighbors. If a peer is a seller, it replies to the buyer if it owns the product, otherwise the request is propagated. A lookup request is discarded when the hopcount is 0. When the buyer stops waiting, it checks every peer id in its candidate seller list. The buyer sends a buy request to the first seller in the list. If the seller has sold the product, the buyer will 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: a seller generates a commodity to sell and 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) For now the system can only run on two different machines. Support for running on 3+ more machines should be supported.
2) Sometimes a thread might stop unexpectedly which might cause the system to pause. We need a further check to ensure the system runs as stable as possible.
3) A peer might reply to duplicate lookup requests from the same peer due to flooding algorithm. That is, lookup requests sent from a buyer might take multiple different routes to a seller. This might cause waste on network bandwidth and computing resources. For improvement, a seller can record duplicated requests to avoid the situation.