CS 677 Lab 1: Design Document

Group 14: Lurdh Pradeep Reddy Ambati(lambati@umass.edu), Ravi Choudhary(ravichoudhar@umass.edu)

Design Considerations:

Peer Structure:

Each peer will have peer-id, host address, role which states whether it is buyer or seller, an inventory variable(dictionary, it is None for buyer) and a shopping list(a list of items to buy, which is None for seller). And each peer will maintain a list of the lookup requests that pass through them, this list is useful to eliminate the duplicate lookup requests.(Same request coming from different peers.)

A buyer-id or seller-id which is used for lookup or reply, consists of pee-id and host address.

Choice of Communication:

We have used XMLRPC library in python for RPC communication. It supports only single threaded operation only, so we had to use ThreadingMixIn library from SocketServer to make it multithreaded[1].

Time-out & Hop count:

Time-out and hop-count depends on the number of the peers in the network, for the network of 6 to 12 peers that we tested, a time-out of around 10 sec was a good trade-off between latency and throughput. A hop-count of (3/4)th of the number of peers is used.

Program Design:

- Lookup Process: *lookup(self,buyer_id,product_name,hop_count,hop_path)*

   1. If a buyer has a item to buy, then it calls lookup method of its neighbor with its buyer-id, item to buy, also appending a hop count and hop path(which is a stack of the dictionaries with peer-id and host-address for back traversal).
   2. When the neighbors sees this call,it first checks it for duplication, if it is a duplicate or if the hop-count is -1, then the request is discarded(pass). Otherwise, it logs the request and if it is a buyer, it forwards the request(except the one who initiated the lookup request and to the one who sent the request) by appending its details to the hop path and decrements the hop count. If it is a seller and it has that item, it replies back to buyer by traversing back the path, else if it doesn't have the item, it forwards the request to neighbors.

- Reply Process: *reply(self,buyer_id,seller_id,product_name,hop_path)*

   1. Reply method is the response that seller initiates. Also, if seller has received multiple lookup requests i.e. multiple buyers are interested in buying same product, even if seller doesn't have enough number it sends them the reply back.
   2. Reply traverses back, until reply is received by the buyer who initiated the lookup method in the first place. Once it receives the reply message, it logs the seller information.

- Buy Process : *buy(self,buyer_id,product_name)*

   1. Buyer initiates this process, after the respective request's time-out is reached, then the buyer checks how many sellers responded. If there any, then he selects one of them and connects directly to him for transaction.
   2. If multiple buyers wanted to buy same product from a respective seller,at a time he can only sell that product to the few buyers only before running out of stock. So, if a buyer tries to buy from that seller, and now that product is out of stock, seller sends a negative acknowledge to buyer and vice versa.

3. After receiving acknowledgement, if it is negative, that respective buyer will check if there are any other seller selling that product, if there are, then the buyer starts the buying process again from him. This repeats until he gets the product or number of sellers zeroes out.

- Auxiliary Method: Duplicate Requests disposal

1. When a lookup request arrives at a peer, first it checks whether it received a copy of that request before by checking its log for the requests with same product_name and same buyer-id(one who initiated). If there are any, based on time-stamp, the lookup request is deemed duplicate.(i.e. difference between the present request's time-stamp and past-request's time stamp is greater than or less than 10 sec.)

How it Works:

All the peers start by starting server and client section of them. Server section starts the its local XMLRPCServer and registers the functions that can be called on them. In client section, if you are a buyer then it starts the lookup process for each product that is in its shopping list, else for seller, it doesn't to anything with client section.

Each peer will maintain a list of peers, if it wants to contact its neighbor, first it checks whether that neighbor is online or not, only then it contacts the neighbors(online). In this way, passing message to the neighbor who are offline is avoided.

Once a buyer initiates the lookup process it waits for some time(time-out : 10 sec) and then checks if there are responses from any sellers and starts the buy process as explained in the buy process.

Design Tradeoffs Considered:

1. Duplicate Requests Discarding: This implementation in each peer can reduce substantially number of requests passing through each peer and also while the responses traverse back, in that case each response may avoid going through same multiple times. On the other hand, some requests may not reach seller if the requests with lower hop_count are being logged, and if request with higher hop_count arrives it is discarded.
2. Usage of multiple shared resources in a peer: In the implementation, request-log and reply-log are shared resources, in case of multiple lookup request at a peer, first they have access the request log to check for duplication as well as to log the request in case. This can slow potentially slow down the process. While, this implementation avoids the logging of same multiple requests logging.
3. Seller replying to the multiple buyers, in case of stock size < number of interested buyers, is an trade-off. In case buyers wants to hold until a buyer buys the product one at a time and replies then, can result in high latency and some times time-out can happen at buyer side, and thus its reply can be discarded. On the other hand, a potential hazard of replying to multiple buyers is that if one of buyer contacts seller and seller is out of stock, that buyer has to contact another seller, cost of this transaction can be costly sometimes, but it is better than the seller holding off the reply.

Possible Improvements:

1. After few buying iterations, if a buyer can store information of the seller, then in the next iterations, it can directly contact seller for the product,in this way response time per lookup request can be decreased.
2. Duplicate request detection is based on time-stamp now, if it can be done through a generic lookup request id, then few comparisons can be avoided. Also, if the hop-count can be considered when discarding a duplicate request, this can improve the through-put.

References:

[1]http://stackoverflow.com/questions/1589150/python-xmlrpc-with-concurrent-requests