CS 677 Lab: Design Document

1. Design Considerations:

1.1 Peer Structure:

Each peer will have peer-id, host address, role which states whether it is buyer or seller or Trader, selling product list(seller), a shopping list(buyer) and a list of traders(trader-info) and in case of the traders, a list of the sellers and the products they sell .

1.2 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].

2. Implementation Details:

2.1 Database Server Process:

We have implemented the database server process using RPC's and can be accessed through port number 8057. This process maintains the list of the products being sold and respective sellers. It supports following functions:

1. Lookup(product_name): This function returns the count of the product available.

2. Register_Products(product_name,seller_info,trader_info): This functions registers the products in the DB and informs(Pushes the Info) the traders that the product is available from the respective seller.

3. Transaction(product_name,seller_info): Deduct the product count of the product_name and update the file maintained and then inform the traders about this transaction.

4. Register_Trader(trader_info): Register the peer as a trader.

This process maintains a csv list, in which each row correspond to a seller and product one sells.

2.2 Traders Electing Procedure:

We have used the implementation of bully algorithm that we did for lab-2. To elect two traders, we trigger two elections and at the end of each election, a trader is elected, thus two traders are elected using two elections. After the first election, second election is started, during the second election, the leader elected in first election doesn't participate, So a new leader is elected during the second election.

2.3 Trading Process:

After both the leaders/traders are elected, trading process starts.

A Seller registers the products that one sells to trader. Once traders receives the request to register the products, trader will registers/caches this information and contacts the Data base process to register this product at the data base, which in turn will broadcast/pushes the information to the other trader.

Buyers wait for a small time to allow the seller to register the goods and start the lookup process for the goods.

A Trader receives the lookup request and if a seller is found who sells that product, contacts the database server for the transaction. Once it receives a positive acknowledgement, then trader contacts both respective buyer and seller to inform that transaction was successful.Also, Trader maintains a log of the requests that it serves.

On the seller side, when ever a "Sold" message is received from trader after a product is sold to a buyer, he deducts the count of that product and picks a new product if seller does not have any products to sell and registers this product to one of the trader and same procedure as above is followed.

2.4 Cache Consistency:

We have used push model and stateless model. Data base pushes the information to traders in case a new product is registered or when a transaction is successful. For every event(register or transaction), data base pushes this information to the traders, so that cache is consistent across database and two traders.

2.5 Heartbeat Protocol:

Time-out is 2 seconds and time interval between messages is 5 sec. If a trader-X sends a message to other trader-Y and doesn't receive any reply within 2 sec, then trader-X broadcasts to every peer that trader-Y is down. Peers upon receiving this message, will update the trader-Y information as inactive and doesn't send lookup/product_register messages to trader-Y.

2.6 Fault Tolerance:

When trader-Y is down and active trader-X discovers this using heartbeat protocol, first trader-X sends the information to all peers that trader-Y is down and then trader-X reads the transaction log of trader-Y and checks whether there are any unresolved/incomplete lookup requests, if found any they will be attended by trader-X. In this way we are resolving incomplete requests.

3. Design Trade-offs:

1) Opting for two elections to elect two leaders:We could have elected two leaders using a single election, but complexity in that case would be higher. Instead, we are conducting two elections to elect two leaders, implementing this was simpler, but more messages needs to be exchanged, thus total time for electing two leaders is high in our case.

2) For cache consistency, database pushes each change to all the traders, this can be bottle neck when database there are a lot of transactions are being done at the same time.

4. Possible Improvements:

1) Implementing a Ring algorithm or other leader election algorithm can be effective than bully algorithm to elect two leaders.

2) Time interval between the trader is down and broadcast of this information is noticeable, as a result lookup requests from buyers are missed even after injecting fault tolerance. This interval needs to be reduced for consistent buying process.

5. References:

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