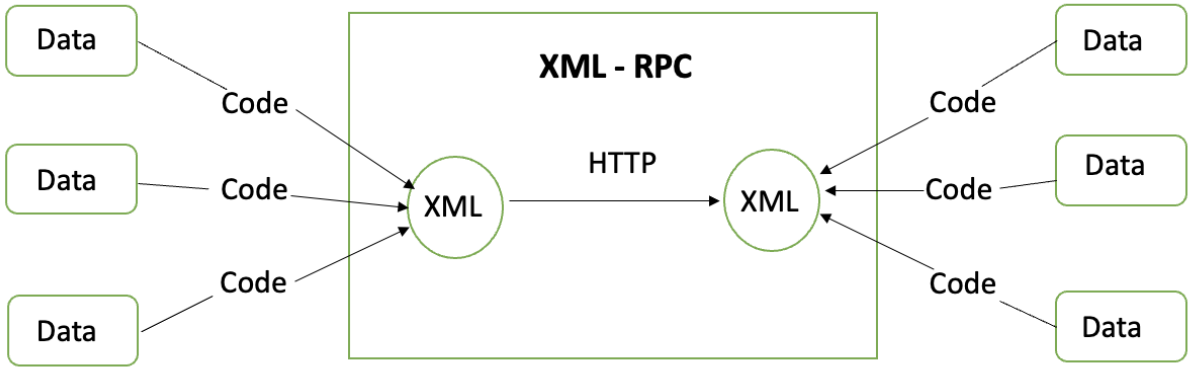
Lab - 3 Asterix and Multi-Trader Trouble: Replication, Fault Tolerance and Cache Consistency

1. Program Design

1.1. Programming Language and Communication -

We have chosen our computing environment to be python for implementing the bazaar environment. We have used the python library named XMLRPC in order to enable RPC communication. XMLRPC is a RPC method that makes use of the XML that is passed via a HTTP in the form of a transport. With the help of this the client can call the methods with the various parameters on the remote server and get the structured data in return. There are 2 main modules for the client and the server namely xmlrpc.client and xmlrpc.server.

This library supports only a single threaded operation and in order to make it multithreaded we have used the ThreadingMixIn library which helps in defining an attribute daemon-threads which help indicate whether the server should wait for the thread termination or not. We have used the threading module inside the peer and used the Multiprocessing module of python to ensure that each of the processes runs as an isolated process.



1.2. Peer Structure -

Bazaar is a p2p network with 10 peers in the network which is fully connected where any peer cannot have more than 3 direct neighbors. We have established an unstructured network where the peer chooses a random peer to be its neighbor and checks if there are no more than 3 neighbors. We have considered the network to have 5 buyers and 5 sellers and each peer consists of a peer-id, host address, role in the bazaar i.e buyer or seller. If the role is a seller then it will have an inventory of what item is being sold and how much quantity can be sold. If the role is a buyer then it will have a shopping list of what item he is willing to buy and how much quantity to buy. All of this information is specified during the configuration of the network. We have chosen the network to have 2

trading posts and therefore each trading post will have 1 trader which will be elected after the network is formed. All these values are passed using Json.

1.3. Overall Structure - The initial network has 10 peers out of which there are 5 buyers and 5 sellers. We have configured our system to have 2 trading posts having 1 trader each. Once the network is formed we perform the Bully election to choose a trader for the trading posts and the trader is not allowed to either buy/sell the items. We have a database which will act as a warehouse which consists of the items available for sale and also keeps updating as the transactions take place. This database is common for both the traders to help in the trading process. The peers of the network can choose either of the traders to complete the trading.

2. How it works

We have implemented a distributed system to ensure trading in the bazaar using the leader election algorithm, synchronized approach, cache-based approach along with fault tolerance.

Initially we create an unstructured p2p network with at least 10 peers which are fully connected with no peer having no more than 3 direct neighbors. The peers can be either buyer, seller and we have configured the network such that there are 5 buyers and 5 sellers in the network with 2 different trading posts. Once the network is created we run the Bully leader election algorithm to elect the two traders for the trading posts. The seller is assigned an item at random to sell along with the maximum quantity he can sell and a buyer is assigned with a random item to buy along with the quantity. Then we elect the 2 traders of the network using the Bully algorithm where the peers start the election and the peer with the highest ID wins the election and broadcasts the message to all the peers in the network in order to proceed with the trading. There are 2 elections which take place and after the first election is started and the first trader is elected then he will not participate in the second election.

Once the traders are elected the trading begins in the network where the seller registers the item he is selling along with the quantity with either of the traders. The traders will in turn register the data into the database. The buyers on the other hand will wait for a small amount of time in order for the seller to register/caches themselves with the trader and then proceed with the lookup of the items. This information is also broadcasted to the entire network. When a trader gets a lookup request and if there is a seller selling the item and has enough quantity to sell then it contacts the database in order to complete the transaction. On the database end the quantity being sold is deducted from the seller and the updated quantity is broadcasted to the trader. Also, the trader has a log of all the requests it serves. If the seller runs out of the item he is selling then he randomly chooses a new item with a particular quantity which is broadcasted both of the traders. Also, as per our design the sellers will acquire Ng goods(5 goods) every Tg seconds(2 Seconds). Once the transaction is completed the trader will pass on the information to both the buyer and

the seller. The buyers can reach out to either of the traders to complete the transaction. If one of the traders is busy then the peer can reach out to the other trader to perform the transaction. As per our design if there are multiple sellers that qualify for trading then we choose the first seller in the list. We have assumed that the buyer has infinite money and can buy any item in any quantity.

We have implemented both synchronized approach as well as the cache based approach. In the synchronized approach whenever there is a buy request to the trader the trader inturn checks with the database if the requested item and the desired quantity is available or not. If it is available then the inventory count is decremented and the buy transaction is successfully completed. If it is not available then the trader informs the buyer that the requested goods are not available. To ensure that there is proper synchronization and also to ensure that there is no over-selling or under-selling of the items we have used the locks.

In the cache based approach we have replicated the master inventory cache at each trader using locks. The trader uses its cache to accept/deny buy requests as per the inventory. To minimize the errors in the cache consistency, we have used the stateless model. The database server pushes the information to traders when there is registration of a new product or in the case of a successful transaction. In any of these events, the database server broadcasts this information to the traders. This makes the cache consistent in the two traders and the database server.

To ensure that the fault tolerance and the heartbeat protocol is applied we have specified a particular time-out of 2 seconds and a time interval of 5 seconds. Trader A sends a message to Trader B to check if it is alive and if it doesn't receive a reply within the timeout 2 seconds then Trader A will confirm that the Trader B has died and will broadcast this message to the entire network and also ensures that all of the future lookup requests to the Trader A. In our design we have killed the trader a random amount of time. Once Trader B is dead then Trader A will check the log of Trader B to check if there are any ongoing/incomplete transaction requests and if there are any then Trader A will complete these transactions.

3. Design tradeoffs

3.1. Design Decisions

We have made a few design decisions in a few situations which are described below:

- Multiple sellers - When there is a buyer requesting for an item and there are multiple sellers who are willing to sell the same item and have the required quantity we are considering the first seller in order to proceed with the trading.

- Trader Dies - In order to ensure that the election is restarted and the new trader is chosen when the current trader is no longer available we are killing the trader after a random amount of time so as to ensure the bully election algorithm is in place.

- Electing the traders - We have chosen the number of trading posts to be 2 thereby, having 2 traders from the network. We are electing these 2 traders with two elections. This has caused additional messages to be exchanged between the peers thereby increasing the total time taken to elect the leaders.

- Cache consistency - The database pushes all of the changes to both the traders at both the trading posts.

3.2. High-level Design

3.2.1. Bully Leader Algorithm

1. Start_election() - Initially, as the network is initiated, the lower-id peers start the election process. There are three types of messages in the bully leader algorithm - election, ok , I won. The peer upon receiving the “election” message returns an OK to its predecessor if present and starts an election of its own by sending election to higher ID processes.If the peer did not receive OK messages after some time, it knows it is the highest ID process in the system and sends I won messages to all other processes.

2. Election_message(message, neighbor) - The three types of messages are handled in this function. The first type of message is the ‘election’ message. Upon receiving this message ‘election’, the peer replies to the sender with an "OK" message. If it has any neighbors with a peer id greater than itself, it forwards the election message and waits some time to receive the OK message. If the peer does not get back any OK message or there are not any higher peers , then it declares itself as the leader and forwards the ‘I won’ message to the peers. The second type of message is the ‘OK’ message. Upon receiving the ‘OK’ message, the current peer drops out from the election process and does not forward any more election messages. The third type of message is the ‘I won’ message. With this message, the current peer saves the details of the current trader and starts to trade.

3.2.2. Database Server Process

1. Lookup(item_name) - The lookup function on the database server end takes the name of the item as a parameter and gives out the quantity of the

particular item. This helps in understanding if the seller has enough quantity required by the buyer in order to complete the transaction.

2. Register_products(item_name, seller_data, trader_data) - This function takes the item name and the seller data and also the trader data like the ID's in order to register the item along with their quantity into the database. Once this information is registered then the trader is informed about the seller with the item details and the quantity. This helps the trader the new addition of a seller for an item and sell the item accordingly.

3. Transaction(item_name, seller_data) - This function takes the item name and the seller data as parameters in order to perform the buy transaction. When there is a buyer willing to buy an item and finds a seller selling the item and has enough quantity then this function helps in deducting the quantity that has been sold to the buyer and thereby informing the trader about the updated quantity after the transaction.

4. Register_trader(trader_data) - This function takes the trader data as a parameter in order to register the trader to perform the transaction.

3.2.3. Synchronized approach

When a buy request is made to a trader, the trader sends a buy command to the database server. In response to the buy command, the database server checks the inventory for the product that is requested, if the product is available, it decrements the inventory count by the requested count and responds back to the trader that the required goods are given. Else if the product is unavailable, the database server responds by saying that the goods are not available. After the trader receives this message, it informs the buyer whether the transaction it requests is successful or not. Similarly, when a sell request is made to a trader, the trader sends a sell command to the database server with the count of the products it wants to sell. The trader in response to this command, asks the database server to increase the inventory count of the product and send a message to the trader saying that the goods were increased.

We are using locks for each product to achieve this synchronization. For the prevention of under and overselling, the buy and sell requests run synchronously for each product which was accomplished by a lock.

3.2.4. Cache

Implemented cache in the function sync_cache() in peer.py. We have implemented a cache at each trader which replicates the master inventory

state of the warehouse using locks.To avoid the need to synchronously communicate with the warehouse before responding to a buyer, the trader can use their cache for this problem. The trader uses its cache to deny buy requests if the cache shows no inventory is available for a particular product. Due to cache in cache inconsistency, there can be the possibility of under and over selling. To minimize the errors in cache consistency, we have a stateless model. The database server pushes the information to traders when there is registration of a new product or in the case of a successful transaction. In case of any of the two events above, the database server communicates this information to the traders .This makes the cache consistent in the two traders and the database server.

3.2.5. Fault tolerance

This is implemented in the periodic_ping_timer function in the peer.py program. The traders continuously send messages to each other to check if the other trader is alive or not. If one of the traders (trader-1) does not get a message back from another trader (trader-2) within some period of time (timeout of two seconds), the trader-1 broadcasts to all the peers that the other trader is down. When the peers receive this message, they don't send any buy/sell requests to the inactive trader. The trader-1 which is active reads the log files to check if there are any incomplete transactions which are pending on the side of trader-2 and if found any, the trader-1 takes responsibility for them and resolves them.

4. Improvements

- Election - We are conducting 2 different elections to select the 2 traders for the 2 trading posts which can be actually done in one single election which will decrease the total amount of time taken to elect the traders.

- Cache consistency - Our database pushes all of the changes to both the traders which can cause a major bottleneck when there are a lot of transactions which are done at the same time.

- Election Algorithm - In our current design we are implementing the bully algorithm for leader election which can be replaced with a much more effective algorithm like the ring algorithm.

- Trader died - We have observed a time interval when the trader dies and this information is passed onto the peer and this can be reduced and to ensure a smooth trading process.

5. Extensions

In our current design the dead peers are not being removed from the system dynamically and are removed once the program is re-run. Dynamical removal of the dead peers can be implemented. We can have a proper GUI setup for clear implementation and better user experience.

6. Something thing you did not accomplish

We wanted to elect both the traders in one single election but couldn't do it because it was highly complex. We did face a few errors and roadblocks along the way but we could successfully finish the lab.

7. How to run your program

The procedure to run the program is as follows:

1. We have created a bash file which consists of the entire network topology with the number of peers, peer ID, role, inventory, shopping list. We can modify the bash file as required to increase or decrease the number of peers in the network.

2. To run cd into the project and run the bash file as follows:

   $bash run_me.sh

8. Test Cases - Descriptions and Performance Results

1. Fault Tolerance - Trader Died

In this experiment we have tested the scenario where one of the trader doesn't reply within the timeout(2 seconds) then he is considered dead and the trader alive will broadcast this message to all of the peers also indicating that he is the only trader alive in the network and all of the subsequent transactions have to be sent only to him.

```
08.12.2022 16:11:24 Election 1 has started
08.12.2022 16:11:24 Dear buyers and sellers, My ID is 10 and I am Trader 1
08.12.2022 16:11:24 The Peer ID  7  has received the Election Won Msg
Let us begin trading
08.12.2022 16:11:24 The Peer ID  2  has received the Election Won Msg
08.12.2022 16:11:25 The Peer ID  1  has received the Election Won Msg
08.12.2022 16:11:25 The Peer ID  3  has received the Election Won Msg
08.12.2022 16:11:25 The Peer ID  4  has received the Election Won Msg
08.12.2022 16:11:25 The Peer ID  5  has received the Election Won Msg
08.12.2022 16:11:26 The Peer ID  6  has received the Election Won Msg
08.12.2022 16:11:26 The Peer ID  8  has received the Election Won Msg
08.12.2022 16:11:27 The Peer ID  9  has received the Election Won Msg
08.12.2022 16:11:27 Election 2 has started
08.12.2022 16:11:27 Dear buyers and sellers, My ID is 9 and I am Trader 2
Let us begin trading
08.12.2022 16:11:27 The Peer ID  2  has received the Election Won Msg
08.12.2022 16:11:27 The Peer ID  7  has received the Election Won Msg
08.12.2022 16:11:27 The Peer ID  1  has received the Election Won Msg
08.12.2022 16:11:27 The Peer ID  3  has received the Election Won Msg
08.12.2022 16:11:27 The Peer ID  4  has received the Election Won Msg
08.12.2022 16:11:28 The Peer ID  5  has received the Election Won Msg
08.12.2022 16:11:28 The Peer ID  6  has received the Election Won Msg
08.12.2022 16:11:29 The Peer ID  8  has received the Election Won Msg
08.12.2022 16:11:30 Trader 1 to Trader 2 : Are you alive?
08.12.2022 16:11:30 The Peer ID  1 has issued a lookup request to the trader for the item  Fish to Trader 2
08.12.2022 16:11:30 Trader 2 to Trader 1 : Yes, I am alive
08.12.2022 16:11:30 The Peer ID  3 has issued a lookup request to the trader for the item  Salt to Trader 1
08.12.2022 16:11:31 The Peer ID  5 has issued a lookup request to the trader for the item  Boar to Trader 2
08.12.2022 16:11:31 The Peer ID  8 has issued a lookup request to the trader for the item  Boar to Trader 1
08.12.2022 16:11:31 The Peer ID  1  has bought the item  Boar  from the seller with the ID  2  via Trader 2
08.12.2022 16:11:31 The Peer ID  3 has issued a lookup request to the trader for the item  Boar to Trader 2
08.12.2022 16:11:32 Trader 2 to Trader 1 : Are you alive?
08.12.2022 16:11:32 Trader 2 has informed peer ID 3 that no Boar was available
08.12.2022 16:11:32 Warehouse rejected buy request of peer ID 3 of boar because inventory was oversold.
08.12.2022 16:11:33 Trader 1 to Trader 2 : Yes, I am alive
08.12.2022 16:11:33 Trader 2 has acquired 5 items of Salt after 3 seconds.
08.12.2022 16:11:33 Trader 6 has acquired 5 items of Fish after 3 seconds.
08.12.2022 16:11:33 Trader 4 has acquired 5 items of Boar after 3 seconds.
08.12.2022 16:11:33 Trader 8 has acquired 5 items of Fish after 3 seconds.
08.12.2022 16:11:33 The Peer ID  5 has issued a lookup request to the trader for the item  Salt
08.12.2022 16:11:33 Trader 1 to Trader 2 : Are you alive?
08.12.2022 16:11:33 The Peer ID  1 has issued a lookup request to the trader for the item  Salt to Trader 2
08.12.2022 16:11:33 The Peer ID  3 has issued a lookup request to the trader for the item  Fish to Trader 1
08.12.2022 16:11:33 The Peer ID  3  has bought the item  Fish  from the seller with the ID  2  via Trader 1
08.12.2022 16:11:33 The Peer ID  5 has issued a lookup request to the trader for the item  Boar to Trader 2
08.12.2022 16:11:33 The Peer ID  8 has issued a lookup request to the trader for the item  Boar to Trader 1
08.12.2022 16:11:33 The Peer ID  5  has bought the item  Boar  from the seller with the ID  2  via Trader 2
08.12.2022 16:11:33 Since there is no reply from Trader 2 on the alive status within the timeout 2 second we consider
Trader 2 dead.
08.12.2022 16:11:34 Dear buyers and sellers, I am Trader 1(peer ID 10) and I am the only trader in the network.
08.12.2022 16:11:34 The Peer ID  6  has received the Msg
Let us begin trading
08.12.2022 16:11:34 The Peer ID  2  has received the Msg
08.12.2022 16:11:35 The Peer ID  1  has received the Msg
08.12.2022 16:11:35 The Peer ID  4  has received the Msg
08.12.2022 16:11:35 The Peer ID  7  has received the Msg
08.12.2022 16:11:35 The Peer ID  5  has received the Msg
08.12.2022 16:11:35 The Peer ID  3  has received the Msg
08.12.2022 16:11:36 The Peer ID  8  has received the Msg
08.12.2022 16:11:36 The Peer ID  7 has issued a lookup request to the trader for the item  Fish to Trader 1
08.12.2022 16:11:36 The Peer ID  3 has issued a lookup request to the trader for the item  Salt to Trader 1
08.12.2022 16:11:36 The Peer ID  7  has bought the item  Fish  from the seller with the ID  4  via Trader 1
08.12.2022 16:11:36 The Peer ID  5 has issued a lookup request to the trader for the item  Salt to Trader 1
08.12.2022 16:11:36 The Peer ID  1 has issued a lookup request to the trader for the item  Boar to Trader 1
08.12.2022 16:11:36 The Peer ID  5  has bought the item  Salt  from the seller with the ID  6  via Trader 1
```

We are analyzing the case where one trader dies in the system. Here, we will be killing the trader but in the real world, it happens due to many kinds of mishaps. From the screenshot, we can deduce the following things:

1. Initially Peers with ID 10 and 9 are selected as traders as priority for selected is the ID here.
2. There is a periodic check between both the traders checking each other if they are active or not. If any of the traders is not active, the other trader will broadcast this

message to all its neighbors, broadcasting it to all the alive peers eventually. The traders communicate with each other with messages “Are you alive” and “Yes, I am alive”.

2. **Fault Tolerance - In-progress transactions**

   In this experiment we have tested the scenario where one of the traders has died and the trader alive checks the transaction log of the trader who died to check if there are any unresolved/incomplete requests. If he finds any of them then he will finish the transactions.

```
08.12.2022 16:15:49 Election 1 has started
08.12.2022 16:15:49 Dear buyers and sellers, My ID is 10 and I am Trader 1
08.12.2022 16:15:49 The Peer ID  2  has received the Election Won Msg
Let us begin trading
08.12.2022 16:15:49 The Peer ID  7  has received the Election Won Msg
08.12.2022 16:15:49 The Peer ID  1  has received the Election Won Msg
08.12.2022 16:15:49 The Peer ID  5 has received the Election Won Msg
08.12.2022 16:15:49 The Peer ID  4  has received the Election Won Msg
08.12.2022 16:15:50 The Peer ID  3  has received the Election Won Msg
08.12.2022 16:15:50 The Peer ID  6  has received the Election Won Msg
08.12.2022 16:15:51 The Peer ID  9  has received the Election Won Msg
08.12.2022 16:15:51 The Peer ID  8  has received the Election Won Msg
08.12.2022 16:15:51 Election 2 has started
08.12.2022 16:15:51 Dear buyers and sellers, My ID is 9 and I am Trader 2
Let us begin trading
08.12.2022 16:15:51 The Peer ID  2  has received the Election Won Msg
08.12.2022 16:15:51 The Peer ID  4  has received the Election Won Msg
08.12.2022 16:15:52 The Peer ID  3  has received the Election Won Msg
08.12.2022 16:15:52 The Peer ID  1  has received the Election Won Msg
08.12.2022 16:15:52 The Peer ID  7  has received the Election Won Msg
08.12.2022 16:15:52 The Peer ID  5  has received the Election Won Msg
08.12.2022 16:15:53 The Peer ID  6  has received the Election Won Msg
08.12.2022 16:15:53 The Peer ID  8  has received the Election Won Msg
08.12.2022 16:15:54 Trader 1 to Trader 2 : Are you alive?
08.12.2022 16:15:54 The Peer ID  3 has issued a lookup request to the trader for the item  Fish to Trader 2
08.12.2022 16:15:54 The Peer ID  5 has issued a lookup request to the trader for the item  Salt to Trader 1
08.12.2022 16:15:54 Trader 2 to Trader 1 : Yes, I am alive
08.12.2022 16:15:55 The Peer ID  1 has issued a lookup request to the trader for the item  Boar to Trader 2
08.12.2022 16:15:56 The Peer ID  7 has issued a lookup request to the trader for the item  Boar to Trader 1
08.12.2022 16:15:56 Trader 2 to Trader 1 : Are you alive?
08.12.2022 16:15:56 The Peer ID  3  has bought the item  Fish  from the seller with the ID  2  via Trader 2
08.12.2022 16:15:58 Trader 1 to Trader 2 : Yes, I am alive
08.12.2022 16:15:58 The Peer ID  5 has issued a lookup request to the trader for the item  Fish to Trader 2
08.12.2022 16:16:00 Trader 1 to Trader 2 : Are you alive?
08.12.2022 16:16:00 Trader 2 has informed peer ID 5 that no Fish was available
08.12.2022 16:16:00 Trader 2 to Trader 1 : Yes, I am alive
08.12.2022 16:16:00 Warehouse rejected buy request of peer ID 5 of Fish because inventory was oversold.
08.12.2022 16:16:02 Trader 2 has acquired 5 items of Salt after 3 seconds.
08.12.2022 16:16:02 Trader 6 has acquired 5 items of Fish after 3 seconds.
08.12.2022 16:16:02 Trader 4 has acquired 5 items of Boar after 3 seconds.
08.12.2022 16:16:02 Trader 8 has acquired 5 items of Fish after 3 seconds.
08.12.2022 16:16:02 The Peer ID  5 has issued a lookup request to the trader for the item  Salt
08.12.2022 16:16:02 Trader 1 to Trader 2 : Are you alive?
08.12.2022 16:16:02 The Peer ID  5 has issued a lookup request to the trader for the item  Salt to Trader 2
08.12.2022 16:16:03 The Peer ID  3 has issued a lookup request to the trader for the item  Fish to Trader 1
08.12.2022 16:16:03 The Peer ID  3  has bought the item  Fish  from the seller with the ID  4  via Trader 1
08.12.2022 16:16:03 The Peer ID  7 has issued a lookup request to the trader for the item  Boar to Trader 2
08.12.2022 16:16:05 The Peer ID  1 has issued a lookup request to the trader for the item  Salt to Trader 1
08.12.2022 16:16:05 The Peer ID  5  has bought the item  Salt  from the seller with the ID  6  via Trader 2
08.12.2022 16:16:05 Since there is no reply from Trader 2 on the alive status within the timeout 2 second we consider
Trader 2 dead.
08.12.2022 16:16:07 Dear buyers and sellers, I am Trader 1(peer ID 10) and I am the only trader in the network.
08.12.2022 16:16:07 The Peer ID  6  has received the Msg
Let us begin trading
08.12.2022 16:16:07 The Peer ID  2  has received the Msg
```

```
08.12.2022 16:16:08 The Peer ID  1  has received the Msg
08.12.2022 16:16:08 The Peer ID  4  has received the Msg
08.12.2022 16:16:08 The Peer ID  7  has received the Msg
08.12.2022 16:16:10 The Peer ID  5  has received the Msg
08.12.2022 16:16:10 The Peer ID  3  has received the Msg
08.12.2022 16:16:10 The Peer ID  8  has received the Msg
08.12.2022 16:16:10 The Peer ID  5 has issued a lookup request to the trader for the item  Fish to Trader 1
08.12.2022 16:16:10 The Peer ID  3 has issued a lookup request to the trader for the item  Salt to Trader 1
08.12.2022 16:16:12 Trader 1 – Checking Trader 2 transaction log and serve the incomplete requests.
08.12.2022 16:16:12 The Peer ID  7  has bought the item  Boar  from the seller with the ID  8  via Trader 1
08.12.2022 16:16:13 The Peer ID  7 has issued a lookup request to the trader for the item  Salt to Trader 1
08.12.2022 16:16:13 The Peer ID  3 has issued a lookup request to the trader for the item  Boar to Trader 1
08.12.2022 16:16:15 The Peer ID  1  has bought the item  Salt  from the seller with the ID  6  via Trader 1
```

We are analyzing the case when one trader dies in the system with some transactions pending on its side. We have two traders- trader_1 and trader_2. If trader_2 dies in between leaving some incomplete transactions, the alive trader_1 takes its responsibility and completes them. We can see from the above screenshot that there are frequent checks going on between the traders checking if the other trader is alive or not. Trader_1 finds out that trader_2 died in between and broadcasts the message that trader_2 is dead and it is the only trader present so that the peers do not send messages to the dead trader_2. The trader_1 then checks if there are any incomplete transactions and completes them by checking the transaction log.

9. Test Cases - Did not work properly

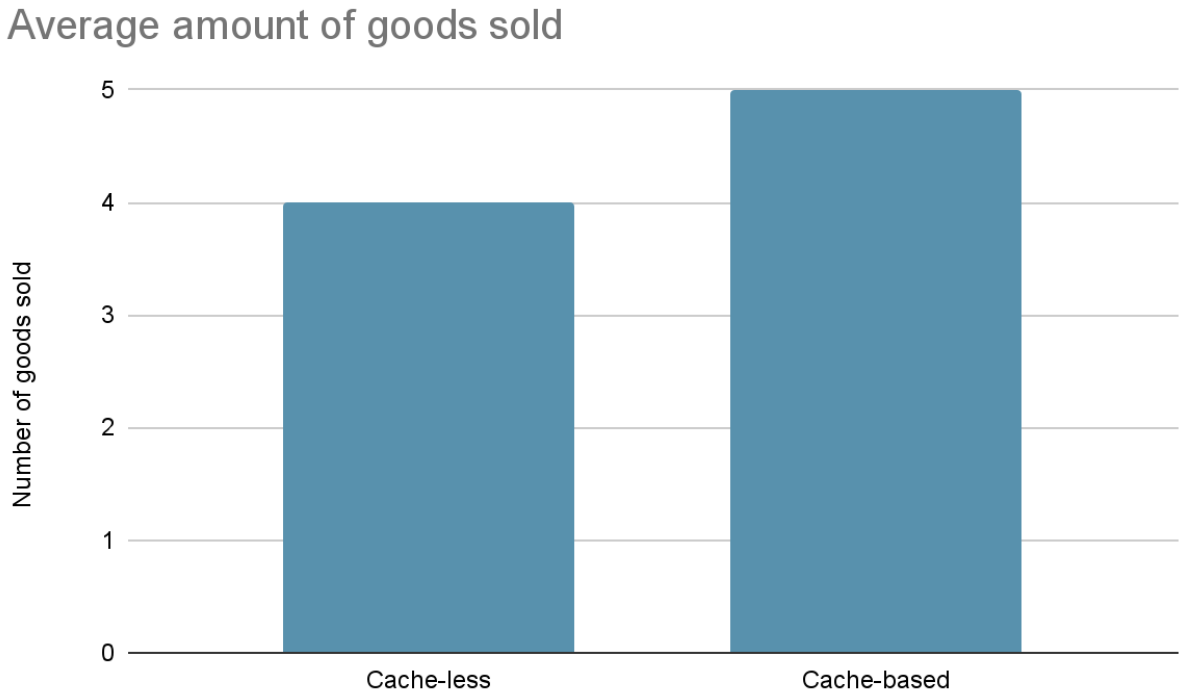
We wanted to start just one election and select both the traders to perform the transactions and we couldn't accomplish that. We tried to increase the number of sequential requests for each of the traders to 1000 and couldn't accomplish it because of the system requirements.

10. Experimental Results

1. Throughput of cache-less and cache-based approaches

In this experiment we have compared the throughput of both of the approaches by averaging the average amount of goods that are being sold/shipped per second from the database in both of these approaches. We have also ensured that we run both of the approaches in the similar system conditions and with the same hyperparameters like the number of sellers/buyers/traders and the items being sold, etc. Our analysis of both the approaches is as follows.

Average amount of goods sold	
Cache-less	**Cache-based**
4	5



We observed that the cache-based approach was more responsive than the cache-less approach. The cache-based approach has an edge over the cache-less approach because the cache-based approach has a cache present at each trader acting as a high speed layer. The need for synchronous communication with the warehouse before responding to the buyer can be eliminated with traders using their cache for data. It has become much easier for the trader to deny requests than in the cacheless approach as there is no need for an additional request to the database server. By using cache, we have also eliminated the request sent to the checking of the product in the database, we can directly get this information from the cache. If there is a transaction or change in the data present in the database server, it pushes the new updated information to the cache. Therefore cacheless work is better in our situation.

2. **Over - selling rate of the cache - based approach**

In this experiment we tried to get the over - selling rate of the cache - based approach by tracking the number of times the trader has forwarded the buy request to the database which was denied because there is no stock. We have tried to change the values of Ng and Tg in order to find some interesting results. Our analysis and the plot of the experiment is as follows.

Number of times buy request forwarded to warehouse							
Ng - 5	**Tg - 3**	**Ng - 5**	**Tg - 6**	**Ng - 3**	**Tg - 5**	**Ng - 6**	**Tg - 5**

2	5	4	5

There can be the case of overselling when the transaction update push did not reach the trader and the trader forwards a buy request to the database server. After changing and studying various values of Ng and Tg, we observed that by keeping the value Ng constant and varying the values of Tg, the number of overselling requests increase with increase in the values of Ng and by keeping the value of Tg constant and varying the values of Ng, the number of overselling requests increase with increase in the values of Tg. We can also observe that when the values of Ng and Tg almost match, the number of overselling requests are also nearby.

3. **Throughput when the trader dies**

In this experiment we have compared the average number of goods sold/shipped from the database when there were 2 traders and the average number of goods sold/shipped from the database when there is only 1 trader along with completing the incomplete requests of the trader who died.

Average number of goods sold	
Both traders alive - 10 seconds	**Only one trader alive - 10 seconds**
7	4

When there are two traders and both of them are alive, we observed that the average number of goods sold were higher than the average number of goods sold when there is only one trader alive. This is because the number of buy requests are handled by both the traders and the load on a single trader is reduced.

.