A peer-to-peer market (bazaar) CS677 lab1 project

Group member: Shahrooz Pouryousef
University of Massachusetts Amherst
shahrooz@cs.umass.edu

ABSTRACT

This paper explains our implementation of a peer-to-peer market (bazaar). The bazaar contains two types of nodes: buyers and sellers. Each seller sells one of the following goods: fish, salt, or boars. Each buyer in the bazaar is looking to buy one of these three items. We have used python and socket programming as a mechanism for communication between nodes for our system implementation. Each transaction of buying an item in our implementation includes three phases; the announcement phase that a buyer announces a message about the item it wants to purchase to each of its neighbors and each node to its other neighbors until a seller that has that item receives the announcement or all sellers receive the announcement. In the back propagation phase, the negotiation message sent by the seller that has that item is back propagated to the buyer. In the final phase, the final message is sent by the buyer to the seller through other neighbors if they are not connected directly and the transaction is finished. We have evaluate the transaction time in a network with 10 nodes and also the affect of message propagation and network size on the transaction time. While our implementation design (the three exploration phases) was well-defined, implementing the system using socket programming as a mechanism for exchanging messages between nodes was not a good implementation decision. Dealing with low level things made the development process so slow and we had to simplify the system design such as assuming only one seller in the network.

1 OVERVIEW

In this section, we explain the high level of our implementation design. Figure 1 shows the DAG (directed acyclic graph) for three phases of an transaction. We explain this figure later in this section.

1.0.1 client and server nodes: In our implementation, we run a thread for each node that runs a server that continuously listens to a socket. For each received connection from any client, this thread will run a separate thread to handle that connection. At the same time, each node builds a connection with each of its neighbors and becomes its client.

1.0.2 Three phases of a transaction. There are three phases for buying an item in our system. These phases are shown in figure 1. Note that this figure does not necessarily shows the network topology but the message DAG. In another word, in our routing topology, the final nodes are not necessarily the seller nodes. However, in this figure and in the message DAG, the final nodes are the seller nodes that either have the item to sell or not. It is clear from the messaeg DAG the announcement phase will fizzle (finish) at the seller nodes.

Each transaction process starts by the buyer node that announces the target product to its neighbors. Each of its neighbors checks if they have that item in their warehouse or not (if they are a seller). If not, they will announce the received message to all of its neighbors. Each node saves the cause of each received announcement message in a data structure called *cause*. The cause of an update message is the neighbor that has sent that announcement message. The key for this data structure is the buyer_ID concatenated by the product name. In addition, in this phase, each node saves the neighbors that it has announced the announcement message to them in a data structure called *sent*. This announcement phase process continues until a seller that has the target product receives the announcement or all of the nodes receive the announcement message. At this moment, the announcement message is basically fizzled and the backpropagation phase starts.

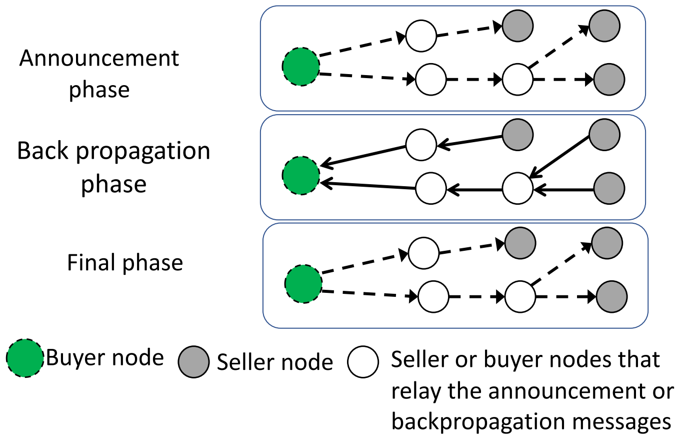

Announcement phase
Back propagation phase
Final phase
Buyer node
Seller node
Seller or buyer nodes that relay the announcement or backpropagation messages

Figure 1: The three phases of an transaction in our implementation

Nodes use the *cause* data structure to send back the backpropagation messages. When the buyer receives a backpropagation message that identifies a seller that has that item to sell, the final phase starts. At this moment, the buyer will send a final message to all of its neighbors and all other neighbors will use the *sent* data structure to send the final message. When the seller receives the final message, it will remove the product from its warehouse.

1.1 Message format

Nodes exchange the messages in JSON format [1] in our system. In the announcement phase, the valid fields are the product name and the buyer ID. In the backpropagation phase, we have the seller ID field that indicates which seller has set that field with its ID. In the final phase, we have all these fields together.

1.2 Non-blocking I/O

When a node calls the receive function call on a socket, it may block as there may be no data on the socket at that time. In order to have a non-blocking I/O in our system, we use *select* system call [2]. *select()* allows a program to monitor multiple file descriptors, waiting until one or more of the file descriptors become *ready* I/O operation. A file descriptor is considered ready if it is possible to perform the corresponding I/O operation (e.g., read(2)) without blocking. *select* system call has a timeout timer that indicates for how long the system will wait until return the results. We set the timeout value to 1 milliseconds in our implementation.

2 ASSUMPTIONS

We assume there is no failure (e.g., node or link failure) in the network. In our setup, we assume the IP addresses that are defined for each node in the configuration files are assignable. For our testing, we use one single machine. In addition, we assume each buyer wants to buy each item only one time. In another word, no buyer will try to buy a fish two times. It can buy different items, but one for each.

2.1 Network topology

We assume the network topology is given to the system in the "network_topology.txt" file. The IP address of each node also is defined in the "DNS_server_cache" file that is simulating the DNS server here. For clarity of testing the system, we assume there is only one seller in the network and other nodes are buyers.

3 EVALUATION

In this section, we evaluate our system implementation. Testing if the system is functioning as it should is easy by updating the configuration files and run the system. In this section, we evaluate the transaction time using our implementation when all nodes are running on the same machine.

3.1 Transaction time

The goal of this experiment is to measure the transaction time for each item that a buyer wants to purchase. We use a random connected network topology with 10 nodes for this experiment. We repeat the experiment 40 times. Each transaction identifies a run of the system where only one of the buyers wants to purchase an item and there is only one seller in the network that has that item.

Figure 2 shows a CDF on the transaction time cross all transactions in our experiment. The transaction time for around %60 of our transactions is around 40 milliseconds.

3.2 Propagation delay

The goal of this experiment is to check how the transaction time is changed when we increase the depth of the network. The depth of the network is the number of nodes that the seller is far away from the buyer. We assume the network topology is not a fully connected graph. We assume the network topology is a chain topology and there are only one of the buyers wants to buy an item. In addition,

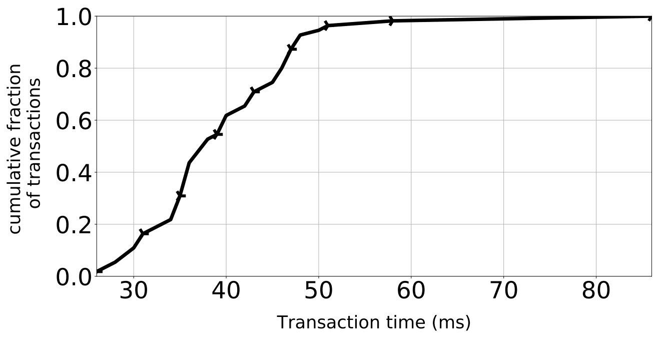
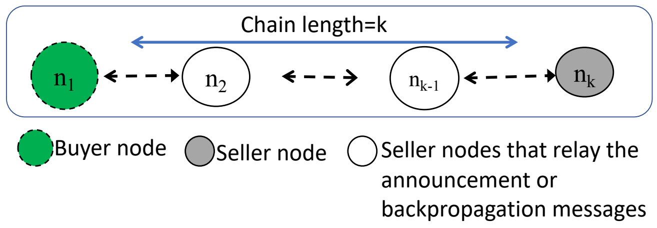


Figure 2: A CDF on the transaction time for buying an item in a network with 10 nodes



Chain length=k
n_1
n_2
n_{k-1}
n_k
Buyer node
Seller node
Seller nodes that relay the announcement or backpropagation messages

Figure 3: Experiment design for measuring the affect of propagation delay on transaction time

there is only one seller in the network that is not connected directly to the buyer. In this situation, other nodes need to relay the announcement messages in which the buyer be able to buy an item.

3.2.1 Process of running the experiment. The first node in the chain of nodes as it is shown in figure 3 is the buyer node and the last node is the seller node. Other nodes are buyer or seller nodes but they do not have anything to buy or sell. Basically, they are just relaying the messages in three phases of a transaction in this experiment. We increase the length of the chain each time and measure the transaction time when node n_1 buys from node n_k. For each chain length, we repeat this for 40 items.

Figure 4 shows the average of transaction time for each chain length setup. As we expect, the transaction time is increased by increasing the chain length.

4 AN ELECTRONIC COPY OF THE OUTPUT

Figure 5 shows an electronic copy of the output generated by running your program.

5 LEARNED LESSONS AND CONCLUSION

Implementing the system using socket programming as a mechanism for exchanging messages between nodes was not a good idea. Dealing with low level things made the development process so slow and we had to simplify the system design such as assuming only one seller in the network. It would be much easier if we did not care about the details of establishing a connection between each

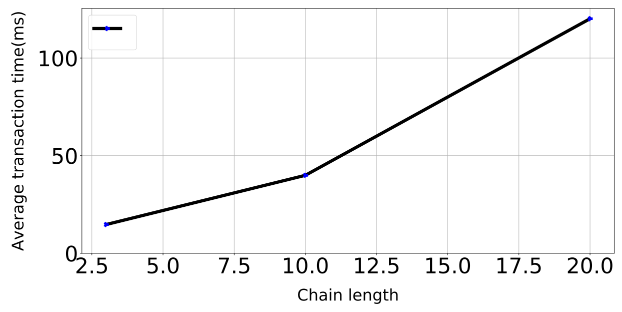
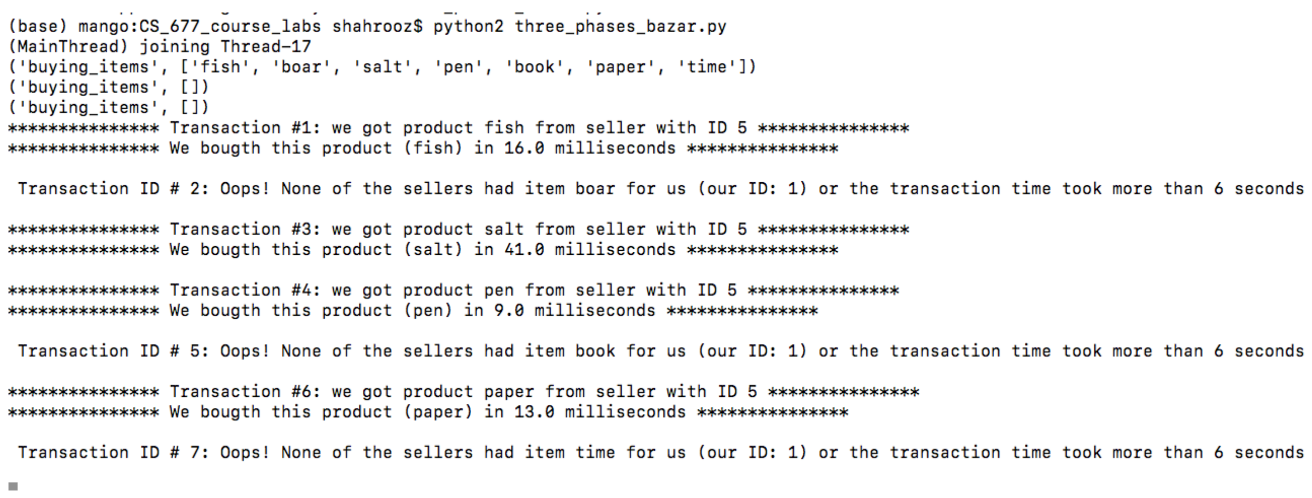


Figure 4: Average transaction time for each chain length

pair of nodes and the difficulties of caring about the blocking or non-blocking I/O calls. Using technologies such as RPCs and RMI would make the implementing of a real system with many nodes from scratch much much easier. Because RPC hides all of the network code into stub functions. Also, we do not have to worry about details (such as sockets, port numbers, byte ordering) etc. We would not appreciate how much RPCs and RMI can simplify the development process easier without trying to use socket programming first.

REFERENCES

[1] Json format. 2021. https://jsonformatter.curiousconcept.com/. [Online; accessed 20-March-2021].
[2] select system call. 2021. https://man7.org/linux/man-pages/man2/select.2.html. [Online; accessed 20-March-2021].



```
(base) mango:CS_677_course_labs shahrooz$ python2 three_phases_bazar.py
(MainThread) joining Thread-17
('buying_items', ['fish', 'boar', 'salt', 'pen', 'book', 'paper', 'time'])
('buying_items', [])
('buying_items', [])
*************** Transaction #1: we got product fish from seller with ID 5 ***************
*************** We bougth this product (fish) in 16.0 milliseconds ***************

 Transaction ID # 2: Oops! None of the sellers had item boar for us (our ID: 1) or the transaction time took more than 6 seconds

*************** Transaction #3: we got product salt from seller with ID 5 ***************
*************** We bougth this product (salt) in 41.0 milliseconds ***************

*************** Transaction #4: we got product pen from seller with ID 5 ***************
*************** We bougth this product (pen) in 9.0 milliseconds ***************

 Transaction ID # 5: Oops! None of the sellers had item book for us (our ID: 1) or the transaction time took more than 6 seconds

*************** Transaction #6: we got product paper from seller with ID 5 ***************
*************** We bougth this product (paper) in 13.0 milliseconds ***************

 Transaction ID # 7: Oops! None of the sellers had item time for us (our ID: 1) or the transaction time took more than 6 seconds
```

Figure 5: An electronic copy of the output