CS677 Lab 1

Bharath Narasimhan, Ronak Zala

Asterix and the Bazaar

March 3, 2019

1 Readme

1.1 Environment Setup

There are two config files *ips.csv* and *connections.csv*. The former describes the peers in the format *Node ID, IP Address*. The latter describes the network topology in the form of edges *Node1, Node2*. Modify the config files as required to setup the environment.
There are two Python files *node.py* and *main.py* - *node.py* represents the peer and its functionalities. *main.py* is used to initialize the network and start program execution.

1.2 Program Execution

Start the Pyro4 name server from any of the machines using *pyro4-ns -n hostname -p port* Start the individual peers on their respective machines using the command *python3 node.py NodeID*. Note that the NodeID here must match the ones used in the config files *ips.csv* and *connections.csv*. This is imperative for the Pyro4 name server to figure out the network topology. Run *main.py* from the machine where the configs are located.

2 Program Design

Framework used - Pyro4[1] (Python Remote Objects). The Pyro4 library enables building of applications where objects can communicate with each other over the network. The Pyro4 server is started at each node. The *Pyro4.config.NS_HOST* and *Pyro4.config.NS_PORT* variables tell the peer how to locate the name server, which then handles all the requests.

2.1 Class Outline

The class Node represents the peer in the question. It has the following methods:

1. init - This constructor is called with every object creation call in *main.py*. It initializes the id, ip and type of the peer (buyer/seller at random). It also initializes the product

[1] https://pythonhosted.org/Pyro4/

name and product count if the peer is a seller. If the peer is a buyer, a list of possible sellers and hop count of the buyer are set.

2. node_start and node_start_t - This function is called on every buyer in *main*. It calls the helper function *node_start_t*. This then waits a random amount of time after calling *lookup*, aggregating the sellers who have replied and then randomly choosing one to transact with. This marks the start of program execution.

3. add_neighbour - This function populates a list *neighbourlist* with the current node's adjacent nodes. It is called in *main*.

4. lookup and lookup_t - *lookup_t* is a helper function for lookup. It contains the necessary logic to perform a lookup at a buyer node. It involves reducing the hopcount by 1 and seeking out sellers in a flood-fill manner by calling *lookup* recursively on the current node's neighbours.
   The *lookup* function involves spawning a new thread for each call to it.

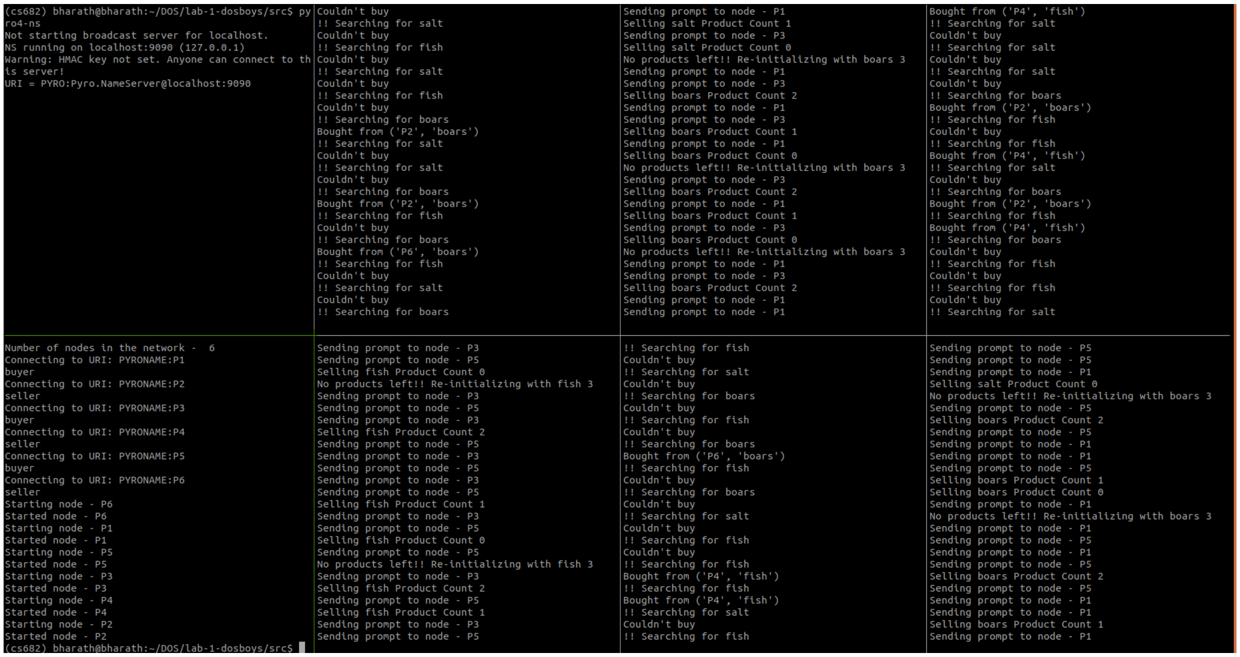
5. reply and reply_t - When a seller is found, *reply* is called in the reverse of the direction that the lookup was made in. This is ensured by passing the path of the lookup *peer_path* as an argument to *reply* and *reply_t*. If the *peer_path* is empty, it means that the current node at which *reply* is being called is the original buyer, hence a transaction can begin at this point.
   The relation between *reply* and *reply_t* is the same as that of *lookup* and *lookup_t* (Just a helper function). *reply_t* has another check that ensures that the buyer is getting what he actually wants. This check is subtly different from the earlier one. The former handles changes at the seller node while this one, the latter handles changes at the buyer node.

6. buy - The *buy* function calls *transact* on the seller matched to the current buyer. It has locks to ensure that shared resources (*product_name* and *product_count*) are not manipulated by other threads.

7. transact - This method carries out the actual transaction. Product counts are reduced by 1 and re-initialized if necessary as required by the problem statement. The *transact* has checks to ensure that the buyer is actually buying the product he wants.

3 Github

The source code can be found at https://github.com/umass-cs677-spring19/lab-1-dosboys. It is divided into 3 folders *docs*, *src*, and *test* for documentation, code and tests respectively.



```
(cs682) bharath@bharath:~/DOS/lab-1-dosboys/src$ py
ro4-ns
Not starting broadcast server for localhost.
NS running on localhost:9090 (127.0.0.1)
Warning: HMAC key not set. Anyone can connect to th
is server!
URI = PYRO:Pyro.NameServer@localhost:9090
```

```
Couldn't buy
!! Searching for salt
Couldn't buy
!! Searching for fish
Couldn't buy
!! Searching for salt
Couldn't buy
!! Searching for fish
Couldn't buy
!! Searching for boars
Bought from ('P2', 'boars')
!! Searching for salt
Couldn't buy
!! Searching for salt
Couldn't buy
!! Searching for boars
Bought from ('P2', 'boars')
!! Searching for fish
Couldn't buy
!! Searching for boars
Bought from ('P6', 'boars')
!! Searching for fish
Couldn't buy
!! Searching for salt
Couldn't buy
!! Searching for boars
```

```
Sending prompt to node - P1
Selling salt Product Count 1
Sending prompt to node - P3
Selling salt Product Count 0
No products left!! Re-initializing with boars 3
Sending prompt to node - P1
Sending prompt to node - P3
Selling boars Product Count 2
Sending prompt to node - P1
Sending prompt to node - P3
Selling boars Product Count 1
Sending prompt to node - P1
Selling boars Product Count 0
No products left!! Re-initializing with boars 3
Sending prompt to node - P3
Selling boars Product Count 2
Sending prompt to node - P1
Selling boars Product Count 1
Sending prompt to node - P3
Selling boars Product Count 0
No products left!! Re-initializing with boars 3
Sending prompt to node - P1
Sending prompt to node - P3
Selling boars Product Count 2
Sending prompt to node - P1
Sending prompt to node - P1
```

```
Bought from ('P4', 'fish')
!! Searching for salt
Couldn't buy
!! Searching for salt
Couldn't buy
!! Searching for salt
Couldn't buy
!! Searching for boars
Bought from ('P2', 'boars')
!! Searching for fish
Couldn't buy
!! Searching for fish
Bought from ('P4', 'fish')
!! Searching for salt
Couldn't buy
!! Searching for boars
Bought from ('P2', 'boars')
!! Searching for fish
Bought from ('P4', 'fish')
!! Searching for boars
Couldn't buy
!! Searching for fish
Couldn't buy
!! Searching for fish
Couldn't buy
!! Searching for salt
```

```
Number of nodes in the network -  6
Connecting to URI: PYRONAME:P1
buyer
Connecting to URI: PYRONAME:P2
seller
Connecting to URI: PYRONAME:P3
buyer
Connecting to URI: PYRONAME:P4
seller
Connecting to URI: PYRONAME:P5
buyer
Connecting to URI: PYRONAME:P6
seller
Starting node - P6
Started node - P6
Starting node - P1
Started node - P1
Starting node - P5
Started node - P5
Starting node - P3
Started node - P3
Starting node - P4
Started node - P4
Starting node - P2
Started node - P2
(cs682) bharath@bharath:~/DOS/lab-1-dosboys/src$ 
```

```
Sending prompt to node - P3
Sending prompt to node - P5
Selling fish Product Count 0
No products left!! Re-initializing with fish 3
Sending prompt to node - P3
Sending prompt to node - P5
Sending prompt to node - P3
Selling fish Product Count 2
Sending prompt to node - P5
Sending prompt to node - P3
Sending prompt to node - P5
Sending prompt to node - P3
Sending prompt to node - P5
Selling fish Product Count 1
Sending prompt to node - P3
Sending prompt to node - P5
Selling fish Product Count 0
Sending prompt to node - P5
No products left!! Re-initializing with fish 3
Sending prompt to node - P3
Selling fish Product Count 2
Sending prompt to node - P5
Selling fish Product Count 1
Sending prompt to node - P3
Sending prompt to node - P5
```

```
!! Searching for fish
Couldn't buy
!! Searching for salt
Couldn't buy
!! Searching for boars
Couldn't buy
!! Searching for fish
Couldn't buy
!! Searching for boars
Bought from ('P6', 'boars')
!! Searching for fish
Couldn't buy
!! Searching for boars
Couldn't buy
!! Searching for salt
Couldn't buy
!! Searching for fish
Couldn't buy
!! Searching for fish
Bought from ('P4', 'fish')
!! Searching for fish
Bought from ('P4', 'fish')
!! Searching for salt
Couldn't buy
!! Searching for fish
```

```
Sending prompt to node - P5
Sending prompt to node - P5
Sending prompt to node - P1
Selling salt Product Count 0
No products left!! Re-initializing with boars 3
Sending prompt to node - P5
Selling boars Product Count 2
Sending prompt to node - P5
Sending prompt to node - P1
Sending prompt to node - P1
Sending prompt to node - P5
Selling boars Product Count 1
Selling boars Product Count 0
Sending prompt to node - P1
No products left!! Re-initializing with boars 3
Sending prompt to node - P1
Sending prompt to node - P5
Sending prompt to node - P1
Sending prompt to node - P5
Selling boars Product Count 2
Sending prompt to node - P5
Sending prompt to node - P1
Sending prompt to node - P1
Selling boars Product Count 1
Sending prompt to node - P1
```

Figure 1: Output of system with 6 peers deployed locally

4 Evaluations

4.1 Local evaluations

Deployed 6 peers in different directories locally. The topology was a ring with alternate buyers and sellerss. The output of the program is shown in Figure 1. The 6 peers are on the right hand side 1(B),2(S),3(B) on top and 4(S),5(B),6(S) on bottom.

5 Tests

The following are possible scenarios:

1. A buy transaction
   Expected result - Product count of seller decreased by 1, and reset if hits 0.

2. 2 buy requests at the same time
   Expected result - First buyer should make the transaction

3. A buy request for an item that does not exist
   Expected result - Should not throw exception or cause untoward behaviour

4. A buy request for an item that already got sold
   Expected result - Seller informs the buyer that the product is sold out. No transaction occurs.

5. A sell reply for an item that already got bought
   Expected result - The buyer informs the seller that it will not buy the item

6. Positive product counts at all times for sellers

6 Possible Extensions

It could be possible to extend the problem statement to the case where a peer can be a buyer and seller at the same time.