Team Members: Ashvin Shivram Gaonkar (agaonka2@ncsu.edu)
Daksh Mehta (dmehta4@ncsu.edu)

Date: 22nd November 2022
Course: CSC_573 (Section 002)

Programming Assignment 2

How to run

First run the server i.e. `auc_server_rdt.py`
`python auc_server_rdt.py <IP Address> <Port Number>`

```
D:\Academics\Project 2>python auc_server_rdt.py 127.0.0.1 1234
Auctioneer is ready for hosting auctions!
```

Once the server is running, run the client i.e. `auc_client_rdt.py`
`python auc_client_rdt.py <Server IP Address> <ServerPort> <RDT Port> <rate>`

```
D:\Academics\Project 2>python auc_client_rdt.py 127.0.0.1 1234 4321 0.5
Connected to the Auctioneer server
Your role is: [Seller]
Please submit auction request:
Enter Bid Information:
```

`<rate>` parameter can be changed as per the requirements. If we want to run without packet loss then we can set `<rate>` to 0

Note: On VCL machines the command may need to be '`python3`' instead of '`python`'

Additional Library Used: `numpy`.
Installing using pip : `pip install numpy`

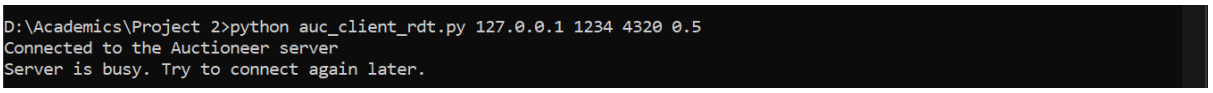
Data validation has been implemented in Seller as well as the buyer. The server will only accept valid requests.

```
D:\Academics\Project 2>python auc_client_rdt.py 127.0.0.1 1234 4321 0.5
Connected to the Auctioneer server
Your role is: [Seller]
Please submit auction request:
Enter Bid Information: 5 100 2 Sword
Invalid Request Information
Your role is: [Seller]
Please submit auction request:
Enter Bid Information:
```

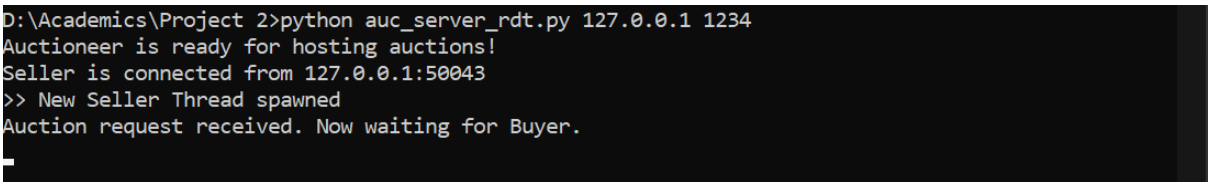
Below image shows a valid request

```
Please submit auction request:
Enter Bid Information: 1 100 3 Sword
Server: Auction start
```

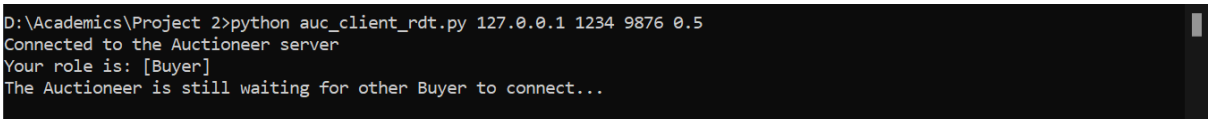
If the buyer tries to connect to the server prior to seller auction is received then the buyer will get Server Busy prompt on the console.



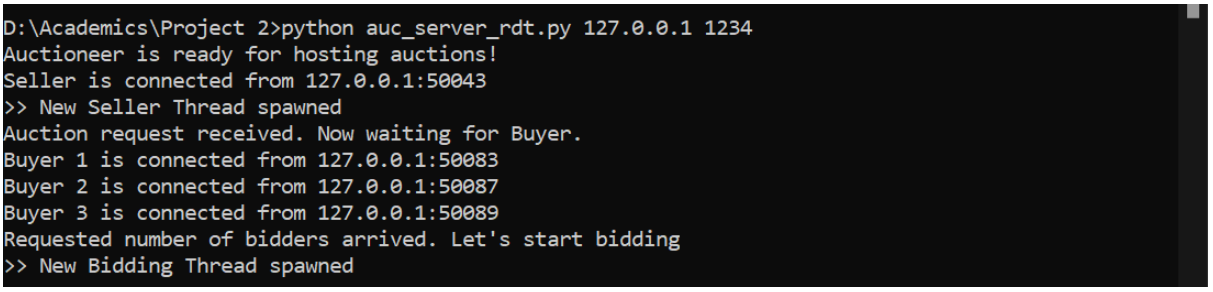
Once the valid request is received by the server it will spawn a new thread and wait for buyers to connect. It will only accept a specified number of buyers mentioned by the seller.



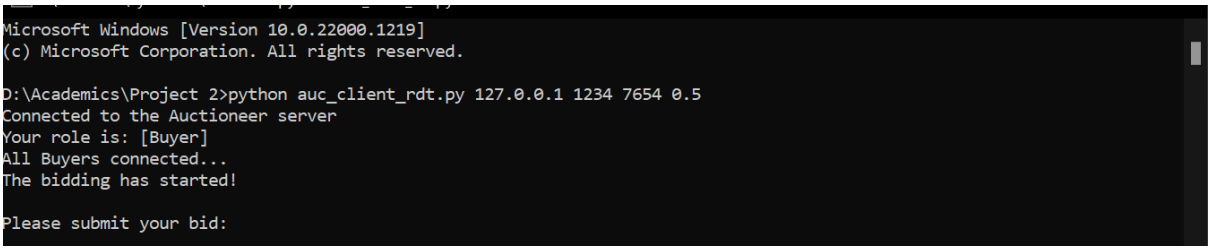
The auction won't start and buyers won't be able to submit their bids until all the buyers are connected.



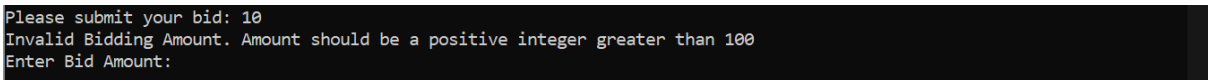
Server will display all the connected buyers IP and the Port number



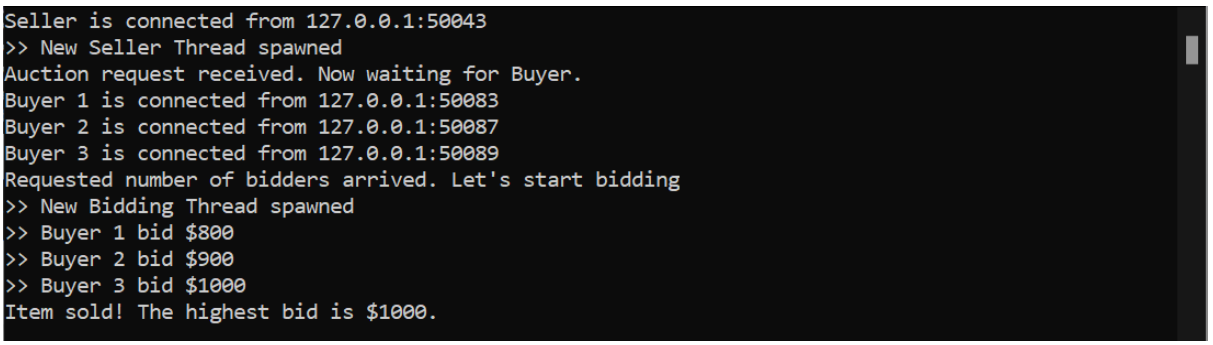
Once all the buyers are connected, the submit bid will be enabled.



Below image demonstrates data validation on Buyers side.



Once the bids are received from all the buyers, the winner is computed and the payment due is sent to the winner and all the rest connection are closed



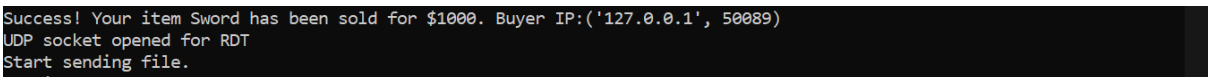
Note: All parties in the previous project are still present, and will complete the auction (Auction Type 1 and Auction Type 2) process as before. This report emphasises more on programming assignment 2.

File Transmission Overview

Seller opens a UDP socket. It reads the 'toSend.file' located in the same directory as '`auc_client_rdt.py`' and stores it in a local buffer. The file is divided into chunks of 2000 bytes each. The seller transmits these chunks using RDP. Some packets are dropped on both seller/buyer side depending on the packet loss rate specified by the user. To tackle out of order/duplicate messages, sequence numbers have been incorporated. If the buyer receives an out of order packet, it replies with the same sequence number indicating to the seller to resend the packet with the correct sequence number. If the sequence number matches with the sequence number the buyer is expecting to receive, it sends an acknowledgment to the seller and stores the received data in a local buffer. When the seller receives the acknowledgement, it then proceeds to send the next packet and this process continues.

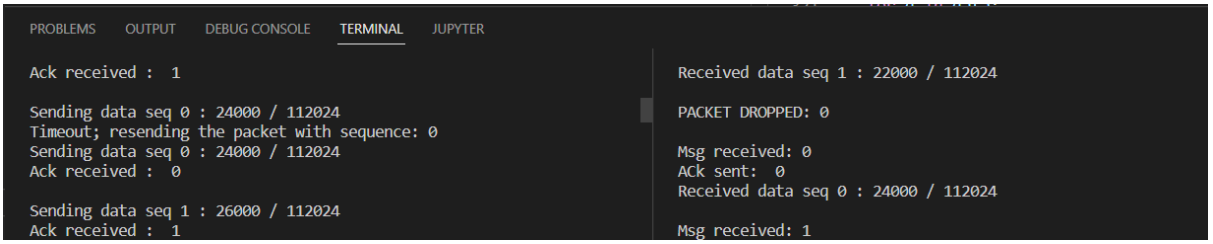
Once the buyer encounters the control message 'fin', the transfer is complete and then it writes the data stored in the local buffer to a binary file named as '`recved.file`'

The seller will receive the Buyer IP and Port Number and start UDP communication with Winner through RDT Port



Similarly, the Winner will receive Seller IP and Port Number and disconnect from the server. and start UDP communication with Seller through RDT Port

```
Disconnecting from the Auctioneer server. Auction is over
Seller IP and PORT: ('127.0.0.1', 50043)
UDP Socket opened for RDT
Start receiving file
```



Seller Buyer

In the above screenshot you can see that the data seq 0 sent by the seller is dropped by the buyer and hence no ack is sent. Thus, after a 2 second timeout at the seller side, the data seq 0 is sent again and this time it is received by the buyer and corresponding ACK is sent back to the seller.

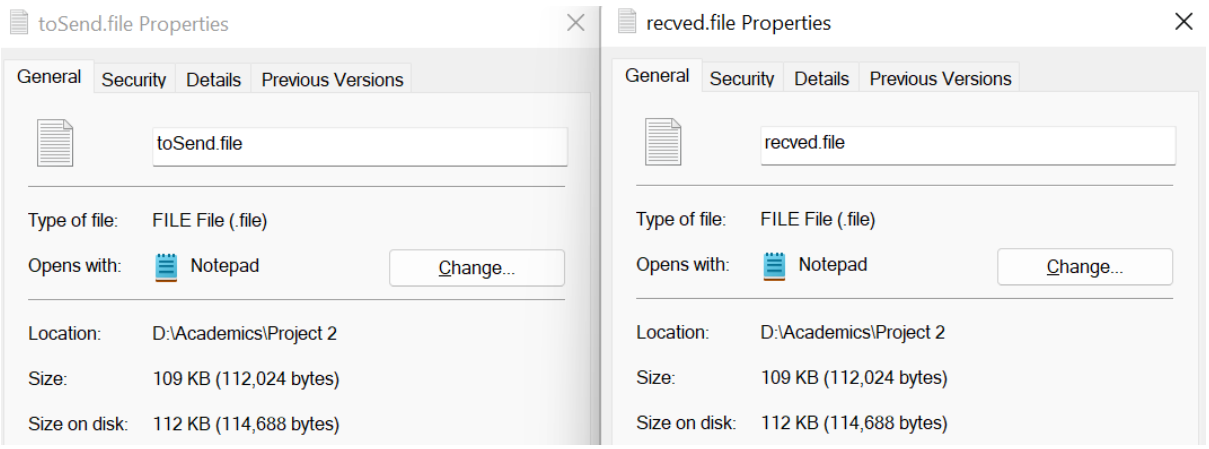
The seller will send `fin` message to tell buyer end of data. The buyer will stop communicating with the seller and write the data from the buffer to the file i.e. `received.file`. By the end of the transmission the Buyer will also compute the Total time taken for transmission and further compute throughput.

```
encountered fin
All Data Received! Exiting...
Transmission finished: 112024 bytes / 198.89181780815125 = 563.2408674953992 bps
```

File Integrity Check

Once the file has been transferred, we can compare if the two files are the same. The two files are exactly the same in the content, implying that file integrity is maintained.

```
PS D:\Academics\Project 2> comp
Name of first file to compare: toSend.file
Name of second file to compare: recved.file
Option:
Comparing toSend.file and recved.file...
Files compare OK
```



Performance Measurement

Below metrics are for packet-loss-rate = 0.1

```
encountered fin
All Data Received! Exiting...
Transmission finished: 112024 bytes / 17.152990102767944 = 6530.873003997298 bps
```

Below metrics are for packet-loss-rate = 0.2

```
encountered fin
All Data Received! Exiting...
Transmission finished: 112024 bytes / 33.36690831184387 = 3357.3383231384405 bps
```

Below metrics are for packet-loss-rate = 0.3

```
encountered fin
All Data Received! Exiting...
Transmission finished: 112024 bytes / 73.18631434440613 = 1530.6686913188212 bps
```

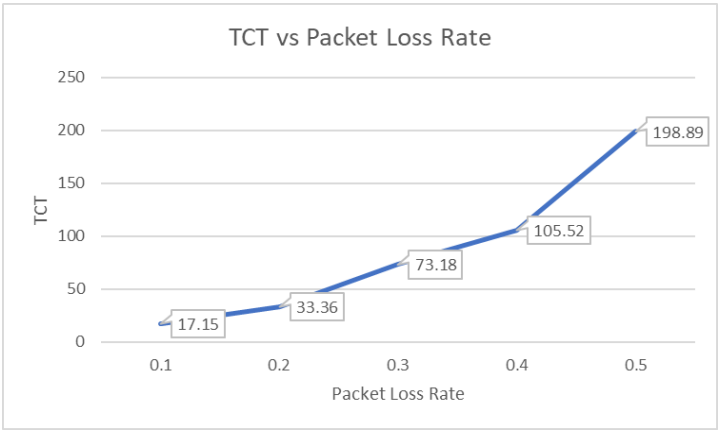
Below metrics are for packet-loss-rate = 0.4

```
encountered fin
All Data Received! Exiting...
Transmission finished: 112024 bytes / 105.52783417701721 = 1061.5587903764406 bps
```

Below metrics are for packet-loss-rate = 0.5

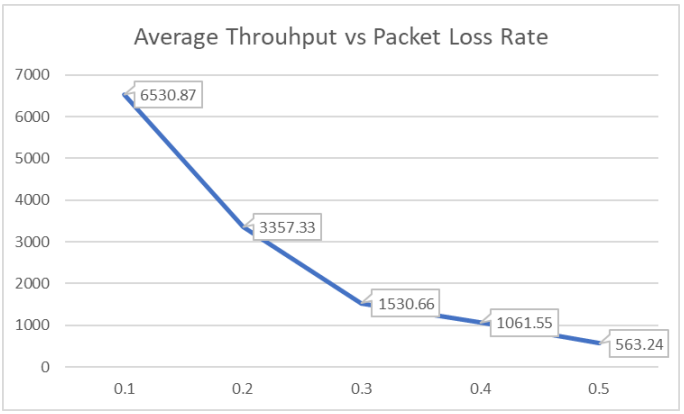
```
encountered fin
All Data Received! Exiting...
Transmission finished: 112024 bytes / 198.89181780815125 = 563.2408674953992 bps
```

TCT(Time taken to receive all bytes) v/s Packet Loss Rate



The total time taken increases as the packet-loss-rate is increased.

Average Throughput v/s Packet Loss



The throughput decreases as the packet loss rate is increased.

Conclusion

The total time taken for file transmission increases as the packet-loss-rate is increased because the frequency of packet drop increases as the packet-loss-rate increases which causes seller and buyer to resend packets. Similarly, Throughput decreases as the packet-loss rate is increased.