CS765- Assignment-2

Simulation of P2P network

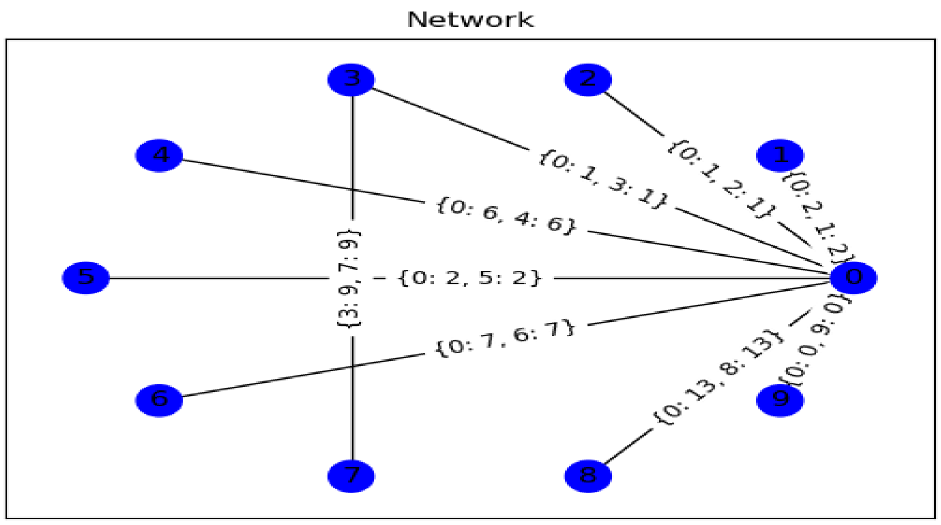
200050012- Anushka Dubey
200050056- Kajal
200050083- Naman Singh Rana

Introduction

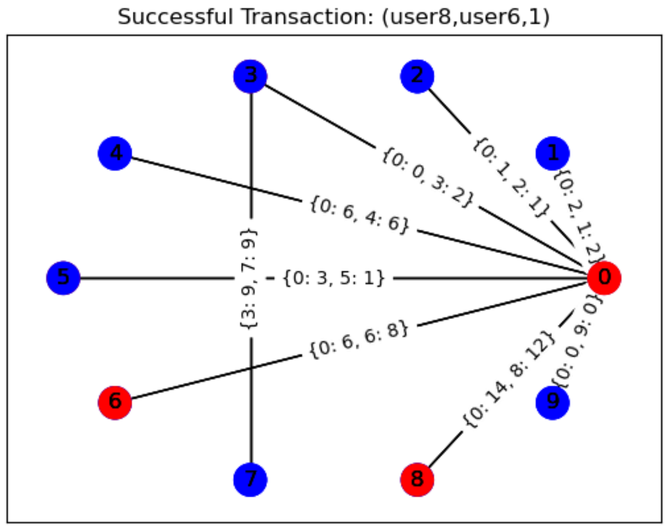
In this assignment, we created a decentralized application where each user forms a joint account with other users of interest and specifies their individual contribution to the joint account. To simulate this we created a connected graph between nodes and an edge represents a joint account between users. A user can send an amount to another user through a path between them. For a transaction, say A -> C where A wants to transfer 3 to C, to be successful first of all A should have enough balance and all the nodes in the chosen path of transaction should have enough balance (in this case 3). We have implemented our sendAmount function (using **BFS**) such that it finds a path to transact such that all the nodes in that path have enough balance instead of finding the shortest path, i.e. **the only way a transaction can fail is when the sender node doesn't have enough balance or there isn't path between the two nodes such that all nodes between the nodes have enough balance**. This implementation causes the success rate to be high and not fluctuate much, as more and more transactions take place (will be shown in the result).

Transaction Graphs :

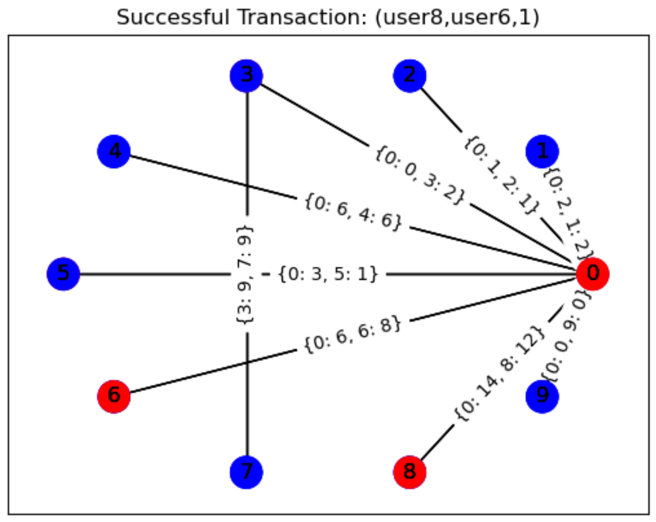
Below is a visual representation of the graph created and how transactions are taking place in our network. For this example we took number of nodes = 10 and number of transactions = 5



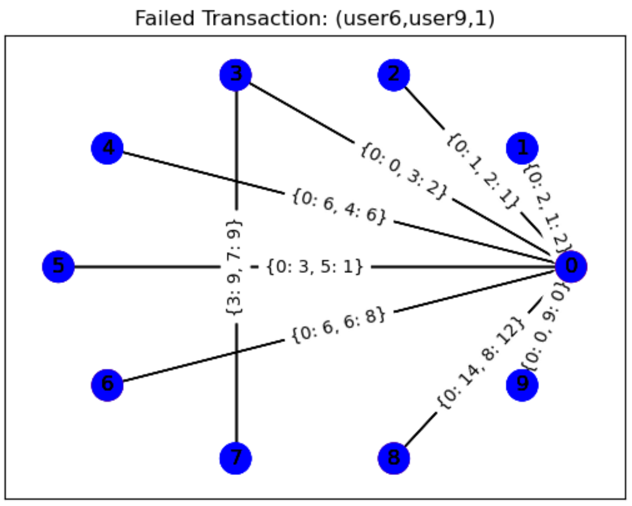
Initial state



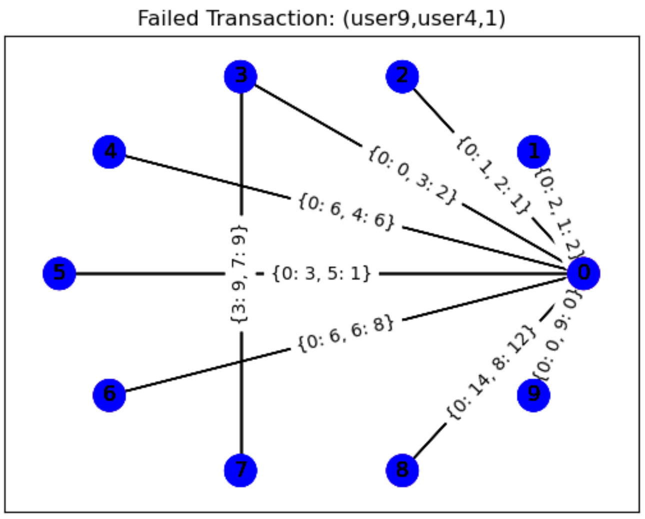
Transaction 1



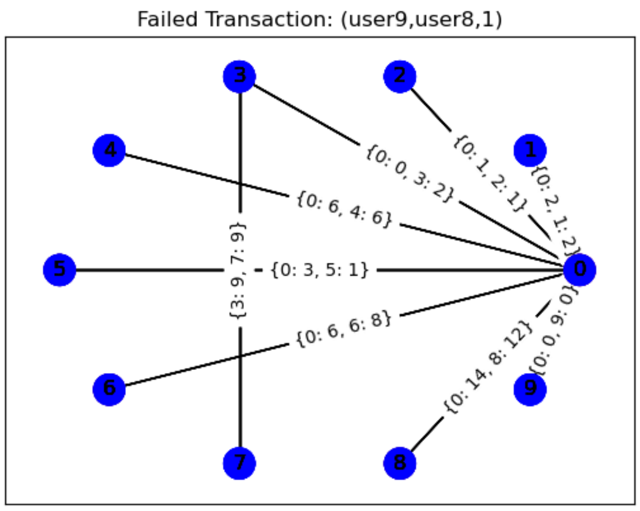
Transaction 2



Transaction 3



Transaction 4



Transaction 5

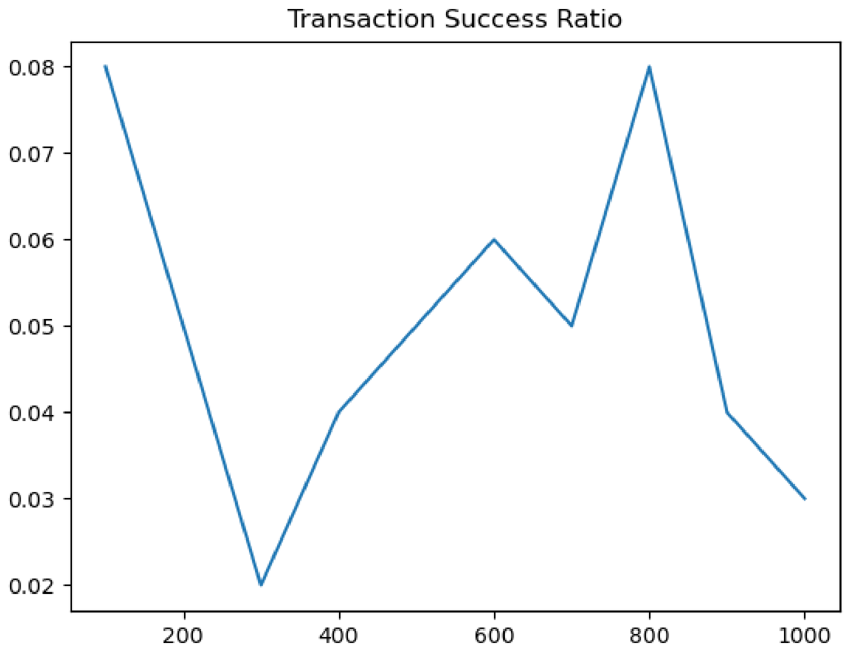
Results

We ran the code on three different values of parameter mean_starting_balance which is the mean starting balance of all nodes. The success rate, after every 100 transactions, is calculated as the ratio of successful transactions to the total transactions in that batch(i.e. Number of successful transactions in that 100 transactions/100). The transactions here are unit transactions(transfer of 1 coin taking place) as specified in the problem statement. We took the number of nodes as 100 and number of transactions as 1000 for each result

Below are the results and observations we obtained after running the simulations:

1. mean_starting_balance = 1

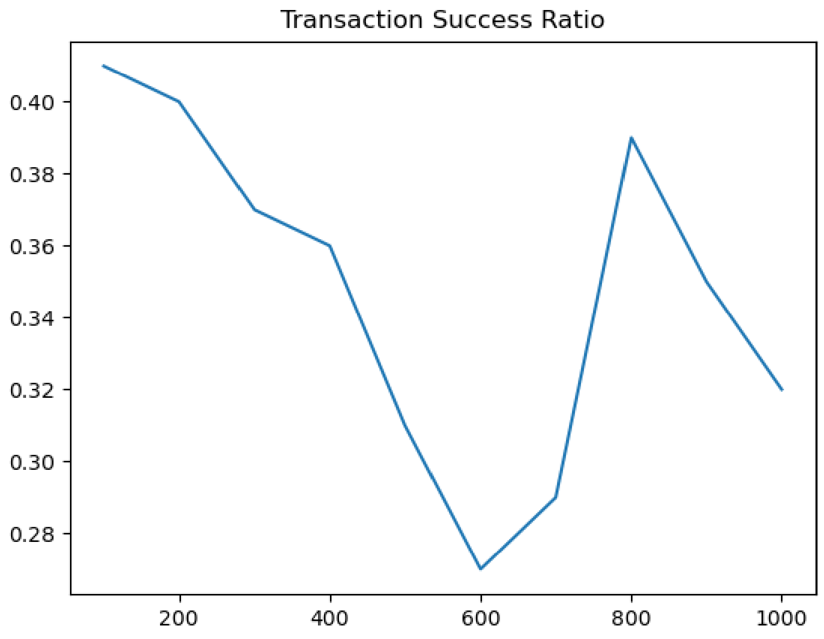
Total transactions	Success Rate
100	0.08
200	0.05
300	0.02
400	0.04
500	0.05
600	0.06
700	0.05
800	0.08
900	0.04
1000	0.03



The plot

2. mean_starting_balance = 10

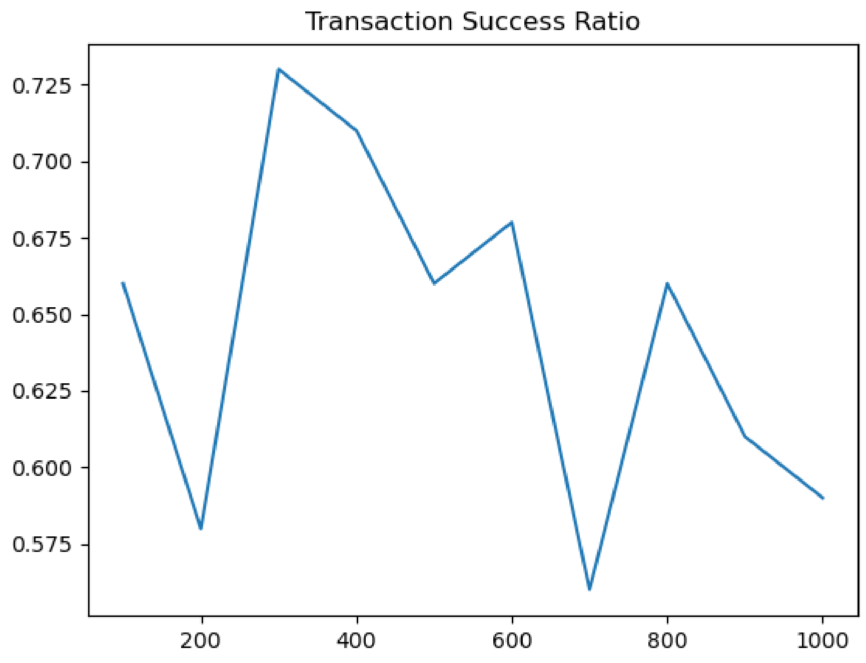
Total transactions	Success Rate
100	0.41
200	0.4
300	0.37
400	0.36
500	0.31
600	0.27
700	0.29
800	0.39
900	0.35
1000	0.32



The Plot

3. mean_starting_balance = 100

Total transactions	Success Rate
100	0.66
200	0.58
300	0.73
400	0.71
500	0.66
600	0.68
700	0.56
800	0.66
900	0.61
1000	0.59



The plot

Observations :

We can clearly see when the mean_starting_balance is more the success ratio is more. This is because when the initial balance of nodes is high we have more money in the network and having more money in the network allows for more transactions of unit amount to take place. Also because of the way we have implemented the algorithm(discussed in intro), the success rate is high and also because of this the fluctuation of the success ratio is also not high.

Also here we can’t comment much on the trend of the success ratio for each case since it depends highly on the network and between what nodes the transaction is taking place which is both random here.

Thus we can conclude that when the **starting balance** of nodes is **high** the **success rate** for the network for unit amount transactions is also **high**.