Abhishek Gupta 001084250

I have implemented Q6 and Q7 in this homework

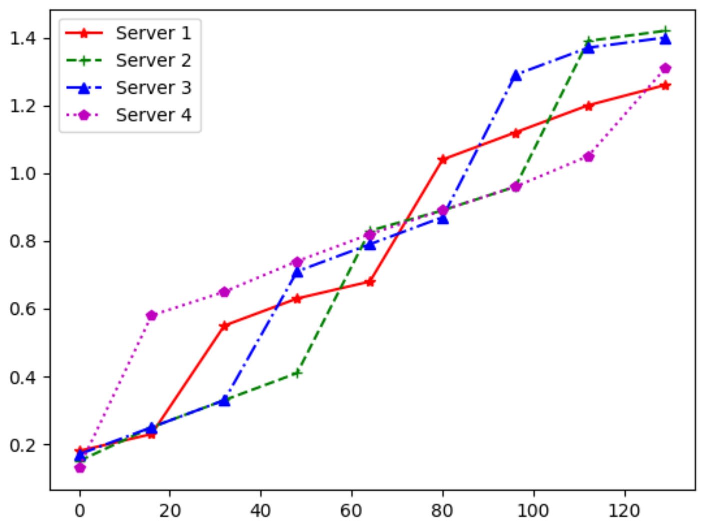
Ans 6

1

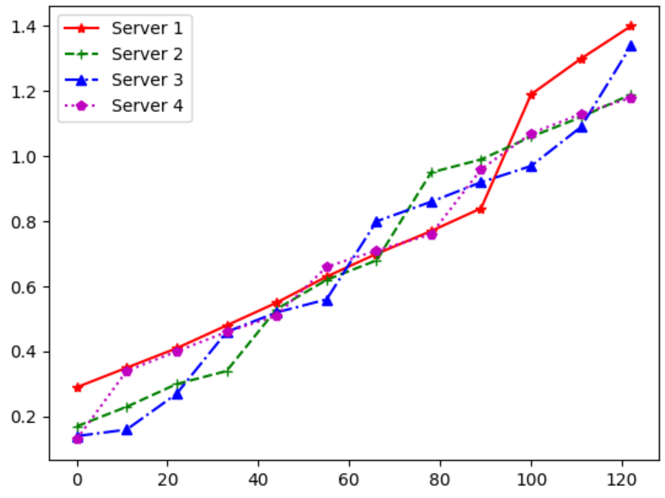
Please refer dnsloadbalacercpu.py to see how cpu time is considered as load. We have plotted the graphs obtained at different feedback frequency. You can also refer respective csv files submitted along with the code.

Tolerance = 1 , T = 60 sec which is constant in all the cases

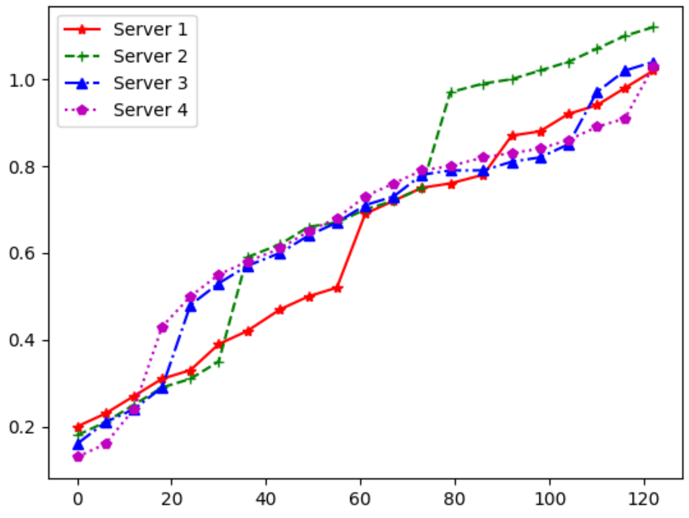
Plot when frequency of feedback is 15



Plot when frequency of feedback is 10



Plot when frequency of feedback is 5



From the theoretical analysis we can see that frequency of feedback is high i.e. we are taking feedback more often, chances of meeting the load balancing goal is better. Practically in this experiment we confirm this. When the frequency of feedback is 5, we see all the servers lines are closely spaced indicating better load balancing.

2. We have used sufficiently random load

```
#!/bin/bash
```

```
while :
do
        for i in U1 U2
        do
                rand=$(( ( RANDOM % 10 )  + 1 ))
                for j in {1..$rand};do sudo docker exec $i curl abc.org:8000/use-cpu; sleep 1;done;
        done
done
```

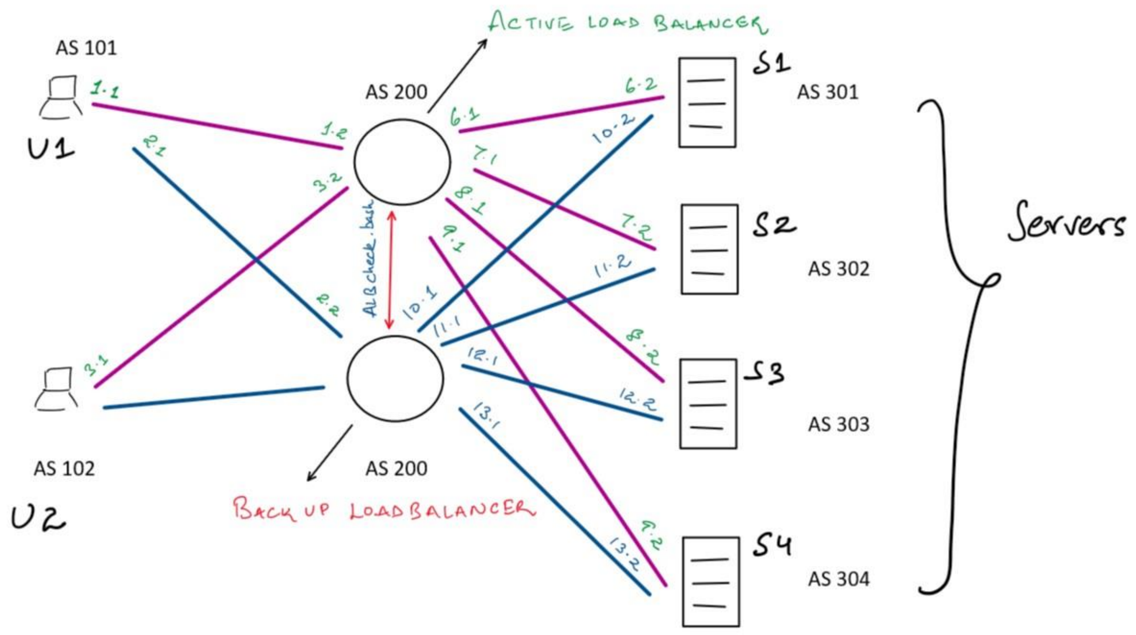
curl.bash could be find along with code.

I could not understand the "**vary the randomness**" fully. But I have got the above results by random loads so the assertion stands the same.

Ans 7

Please refer folder Q7 for all the required code.

1. **System Diagram**
   We are using anycast Load Balancer approach as demonstrated in Last Load balancing Lab. We are using BGP path manipulation to direct traffic to least loaded Server.



Active and Backup Load Balancer forms eBGP membership with all other nodes in the system diagram

Back Load Balancer monitors the health of Active Load Balancer (by pinging on its connected interface) using ALBcheck.bash. Once it detects pack loss on that link it runs anycastloadbalancer process on it and now all the traffic is passed through it and it balances the load on the logic as Active Load balancer was doing that is by periodically checking the load on servers and selecting the server which is least loaded.

2

Trade offs

1. During switch between Active and Backup Load Balancer, the connectivity will be disrupted. In our case it is 5 sec but could be reduced to 3 sec and even more if more time is invested for future work.
2. In this implementation logs stored in Active Balancer is lost but refined implantation with some overhead can get the logs as well.
3. Since in our implementation both Active and Backup load balancer is connected through BGP peering with Servers and Users, there might be lot BGP message overhead. This might cause problem in scaling up the solution.

3

Implementation and Evaluation

Please video file attached for video demonstration

4

We have done this implementation only.