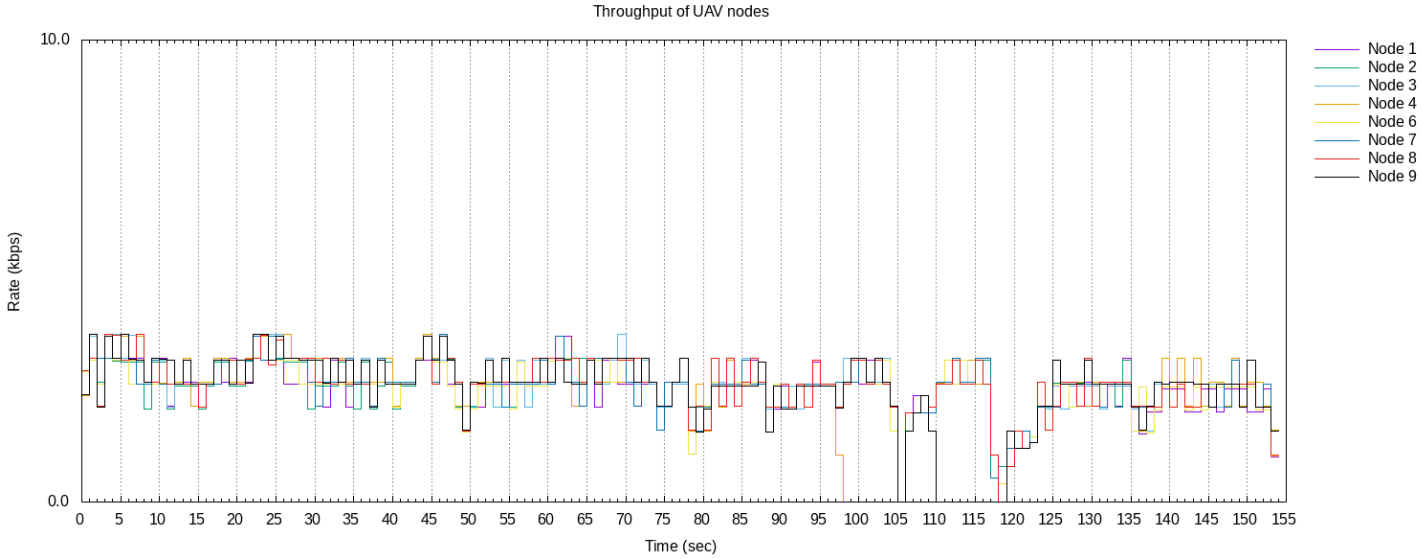
# EE597 Lab2

Qiushi Xu Yiyi Li

In Lab2, we used CORE to simulate the connection between UAV drones and targets, and tried to match each UAV drones to exactly one target and connected them with UDP. Given code named “test_uavs_grpc.py” and “track_target_grpc.py”, we created 2 new test cases, and by changing the testing shell, we can obtain the following 9 figures, with one plotting throughputs of 8 UAV nodes and separate figures for each node. The whole tracking and test process lasts about 155s, and the results are shown as below.

By analyzing the figures of throughput in total, we found that, although there are some fluctuations, the throughputs of each UAV node remain relatively stable between 2 and 3.5kbps, except when there are UAV drones crash.



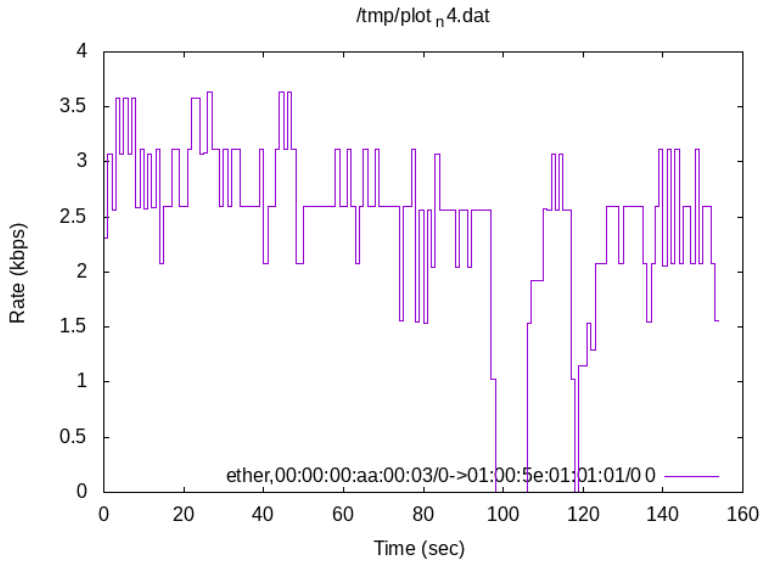

To find the relationship between the change in throughput and the corresponding test case, we changed the output in “latency.log”, set the start time of test case “1a” as 0s, and record the start time and end time of each test case based on test case 1a.

From the timestamp we can analyze the latency performance. Both test case 1 and 2 move each target into range separately with two seconds interval. Given that test case 1 moves all targets while test case 2 move 6 out of 8 targets, the results show reasonable latency which is about 16 sec and 12 sec. For test case 3,4 and 5, all targets are moved into range at the same time, we can tell that it takes about 2sec for track-target algorithm to form unique UAV-target pairs.

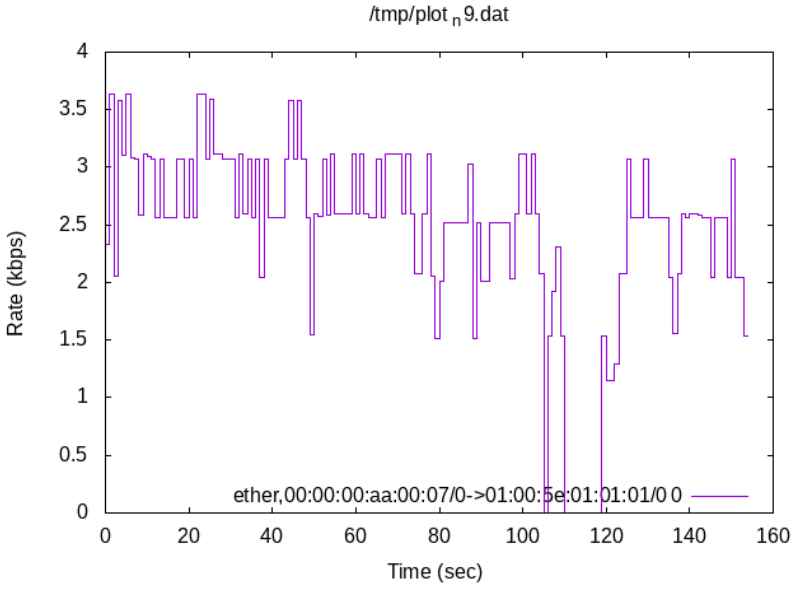
| Timestamp | Start Time(sec) | End Time(sec) | Latency(sec) |
|---|---|---|---|
| Test Case 1a | 0.0002 | 16.3503 | 16.3501 |
| Test Case 1b | 21.4927 | 37.8340 | 16.3413 |
| Test Case 2a | 42.9678 | 55.3377 | 12.3699 |
| Test Case 2b | 60.5127 | 72.8439 | 12.3313 |
| Test Case 3a | 77.9927 | 80.5373 | 2.5446 |
| Test Case 3b | 87.7279 | 89.8958 | 2.1679 |

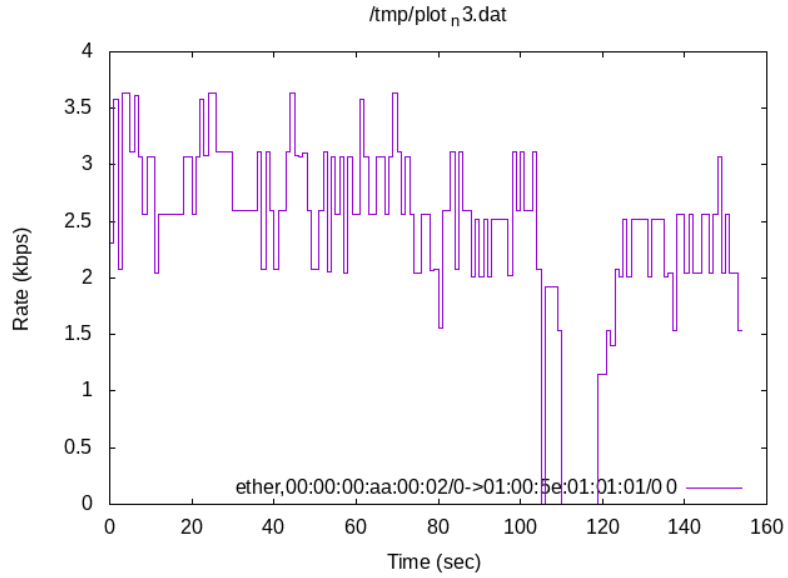
| | | | |
|---|---|---|---|
| Test Case 4 | 97.1603 | 99.3003 | 2.1400 |
| Test Case 5 | 109.5726 | 111.9387 | 2.3611 |
| Test Case 6a | 122.1320 | 125.3758 | 3.2439 |
| Test Case 6b | 135.5117 | 138.7286 | 2.2169 |

In addition, on the basis of the original test samples, we add test case 6a and 6b, which move 6 out of 8 targets in order and out of order respectively. Different with test case 2 which moves targets with time interval, test case 6 moves targets simultaneously. Although the number of targets is less than of previous tests (3, 4, 5), the results show about a 3-sec latency, which increases even a little bit higher compared with 2-sec latency. It may seem counter-intuitive, but there's an explanation that since the scenario becomes no longer one-to-one mapping, conflicts occur when UAVs try to lock the same target and it takes more time to rearrange the ownership of the targets.
What's more, it is easy to find that, at 97.1603sec, test case 4 started and UAV 4 crashed. According to the figures of node 4, the throughput went to 0 when it was crashed, and recovered quickly after the test case ended.

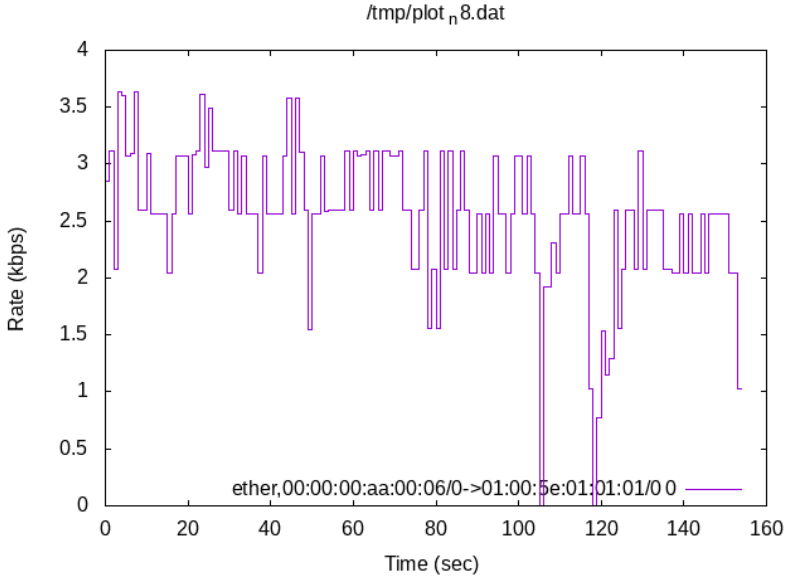


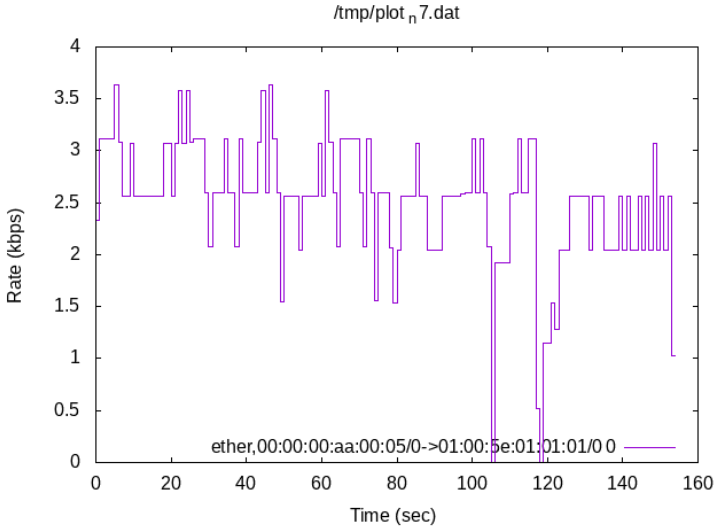

Similarly, in test case 5 there are 2 UAVs (3 and 9) crashed in the system, the throughputs of these two crashed UAVs decreased to 0, and the throughputs of the other 6 UAVs went to a little higher than the normal level, which indicates that they can share more medium than all 8 UAVs are active.

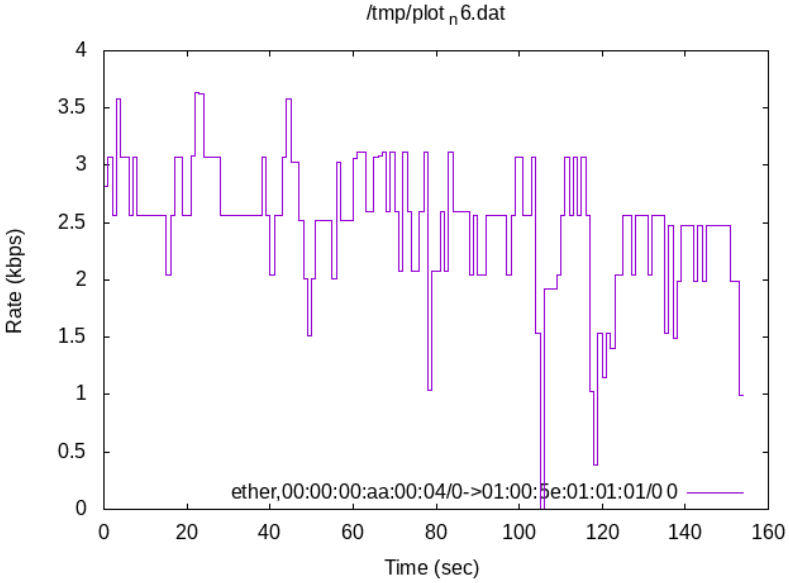


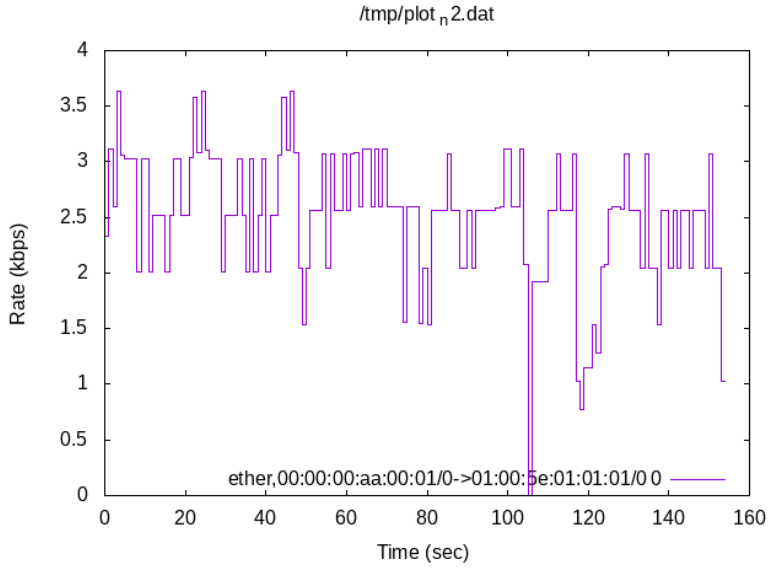



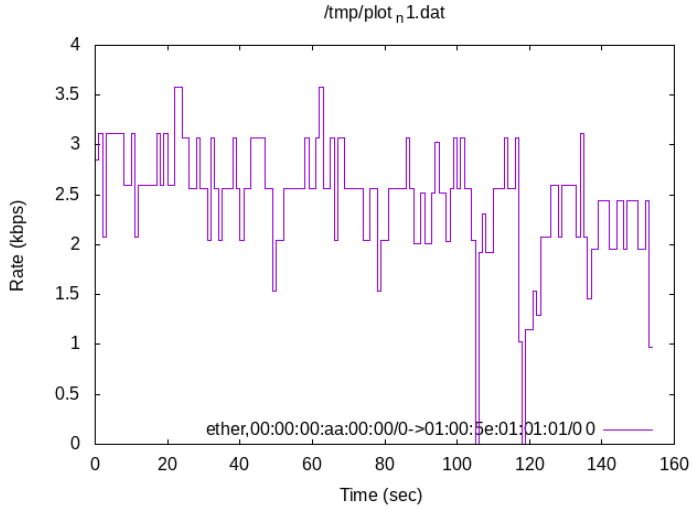

It is notable that every time crashed UAVs recover to active, it seems that all UAVs stop transmitting shortly (about 1 second). In the following 5 figures for the rest of UAVs which are not chosen to crash, we found that the throughput of each node goes to 0 at approximately 106s after test case 4 ended and 118s after test case 5 ended. Unfortunately, we are not able to figure out the reason why this phenomenon happened, but we've tried to come up with a plausible explanation that in order to let offline UAVs reconnect to the group after crash test, the system needs to re-initialize all the UAVs for synchronization.


/tmp/plot_n8.dat
Rate (kbps)
Time (sec)
ether,00:00:00:aa:00:06/0->01:00:5e:01:01:01/0 0


/tmp/plot_n7.dat
Rate (kbps)
Time (sec)
ether,00:00:00:aa:00:05/0->01:00:5e:01:01:01/0 0


/tmp/plot_n6.dat
Rate (kbps)
Time (sec)
ether,00:00:00:aa:00:04/0->01:00:5e:01:01:01/0 0


/tmp/plot_n2.dat
Rate (kbps)
Time (sec)
ether,00:00:00:aa:00:01/0->01:00:5e:01:01:01/0 0
0
0.5
1
1.5
2
2.5
3
3.5
4
0
20
40
60
80
100
120
140
160


/tmp/plot_n1.dat
Rate (kbps)
Time (sec)
ether,00:00:00:aa:00:00/0->01:00:5e:01:01:01/0 0
0
0.5
1
1.5
2
2.5
3
3.5
4
0
20
40
60
80
100
120
140
160