Abhishek Gupta 2017005
Anuj Verma 2017026

Contribution Of Group Members

1) We both have thought about the implementation of the scheduler and then discuss with each other on implementing the scheduler part.
2) We both have equal contribution in the coding as well as in the report part.
3) In every part of the project we both have contributed equally.

Description Of Simulation Scenario And SetUp

1) We have done a 30 min simulation of a network on a bandwidth of 20MHz.
2) There are three types of clients onoff, bulk send and http.
3) Packet size of onoff client is 400 bytes

Description Of Trace Analysis

We are describing this point using the graphs in the below sections.

Design Of the Scheduler

1) In designing the scheduler we have kept two things in my priority on the type of station and greedy approach on the number of available RUs.
2) From the given bandwidth we know how many RUs are available of each type.
3) We are assigning 26 tone RUs to OnOff 52 tone to Bulk Send and again 26 tone to HTTP in general case but the assignment can vary on the availability of stations and RUs for those we have written extra lines of codes.
4) For Bulk Send clients we are trying to assign as much high number RU as we can.
5) Next from on off , bulk send and http our scheduler give priority first to on off then to bulk send and then to http.
6) If I only talk about the on off they are scheduled on the basis of data packet size and all the on off will get the same 26 tone RU.
7) Then for the bulk send we are scheduling on the basis of remaining data so that we can overcome the effect of starvation of the bulk send client.
8) Initially we are assigning 52 tone RU to the bulk send client but according to the situation we are trying upto 242 RU to bulk send the client.
9) At last we are scheduling for HTTP clients and each http client will get the 26 tone RU and we are scheduling the http client on the basis of their data packet size.

This is the algo that we are following for our scheduler.

Implementation Of The Scheduler

How we implemented our scheduler:

a)We have formed a new data structure m_dataInfo which is vector a tuple of MacAddress, DataSize,DlStationInfo

b)First we identified which clients are there to which Ap want to send data and separate them on the basis of mac address. It is hard coded as we already know which type of client has what mac address. For this we have formed onoff1, bulksend1, http1 in rr-ofdma-manager.h

c)Then we divide the scheduler function into two parts

- i) When it does not have any bulk sender client to schedule
- ii) When it has bulk sender client

d) In the first case we used previous approach of scheduling ie. assign equal best available and equal RU's on the basis of no of clients

e) In the second case, we try to assign 26 tone RU's first to all the onoff and if RU's are left then assigning to bulk sender clients on the greedy basis and then after that to the rest of HTTP clients.

f) As RU's can be mixed so we returned a vector of assigned ru's from the getNoAndTypeOfRus

g)We passed that mixed RU vector in the InitTxVectorAndParams so when the OFDMA reaches the client they can easily identify which RU is assigned for them.

Pseudocode:

```
GetNumberAndTypeOfRU's()
    Vector onoff;
    Vector bulksend;
    Vector HTTP;
    Vector ruAssigned;
    For i from 0->len(m_dataInfo):
        If m_dataInfo[i].mac_address in onoff1:
            onoff.push(m_datainfo[i])
        If m_dataInfo[i].mac_address in bulksend1:
            bulksend.push(m_datainfo[i])
        If m_dataInfo[i].mac_address in http1:
            http.push(m_datainfo[i])
    // sort in descending order of datasize
    sort(onoff);
    sort(bulksend);
    sort(http);
    If bulksend.length==0 ||  only station to send data:
        For ru in  subCarrierGroups:
            If ru.stations_supported<=no_of_clients:
                ruType=ru.ruType
        For i in no_of_clients:
            ruAssigned.push(ruType)
        m_dataInfo.push(onoff);
        m_dataInfo.push(http);
        m_dataInfo.push(bulksend)
```

```
Else;
    If onoff.length>7:
        Assign all the available 26 Ru’s to the onoff
        ruType=RU_26_TONE
        For i in no_of_clients:
            ruAssigned.push(ruType)
        m_dataInfo.push(onoff);
        m_dataInfo.push(http);
        m_dataInfo.push(bulksend)
    Else:

        allocated_26_Tone=onoff.length // how many 26 RU are  allocated
        allocated_52_Tone=0 // how many 52 RU are allocated
        allocated_106_Tone=0          // how many 106 RU are alloacated
        //tones_left_after_assigning_26_to_available_onoff
        tones=242-26*onoff.length
        length<-bulksend.length
        If tones>=106 and length>0:
            tones=tones-106
            allocated_106+=1
            length-=1

        While tones>=52 and length>0:
            tones-=52
            allocated_52+=1
            length-=1

        length<-http.length
        While tones>=26 and length>0:
            tones-=26
            allocated_26+=1
            length-=1
        i<-0
        j<-0
        k<-0
        If allocated_106>0:
            ruAssigned.push(RU_106_TONE)
            m_dataInfo.push(bullsend[i++])


        Else:
            ruAssigned.push(RU_52_TONE)
            m_dataInfo.push(bullsend[i++])
```

```
                allocated_52-=1
                Assign 2 26 RU to the onoff and http
                Push onoff or http to m_datainfo on the basis of order in
                which they are assigned RU

                allocated_26-=2
            If allocated_26.length is odd:
                Assign RU_26_TONE 5th index to the onoff and http
            Assign rest of 52_TONE Ru to bulksend and push clients in the
            same order in m_dataInfo
            Assign rest of 26_Tone RU to onoff and http and push clients in
            the same order in m_dataInfo

            Push rest of clients left in the m_dataInfo
    return ruAssigned
```

Files that we have modified

a) rr-ofdma-manager.cc
b) Rr-ofdma-manager.h

Name of The functions that are modified

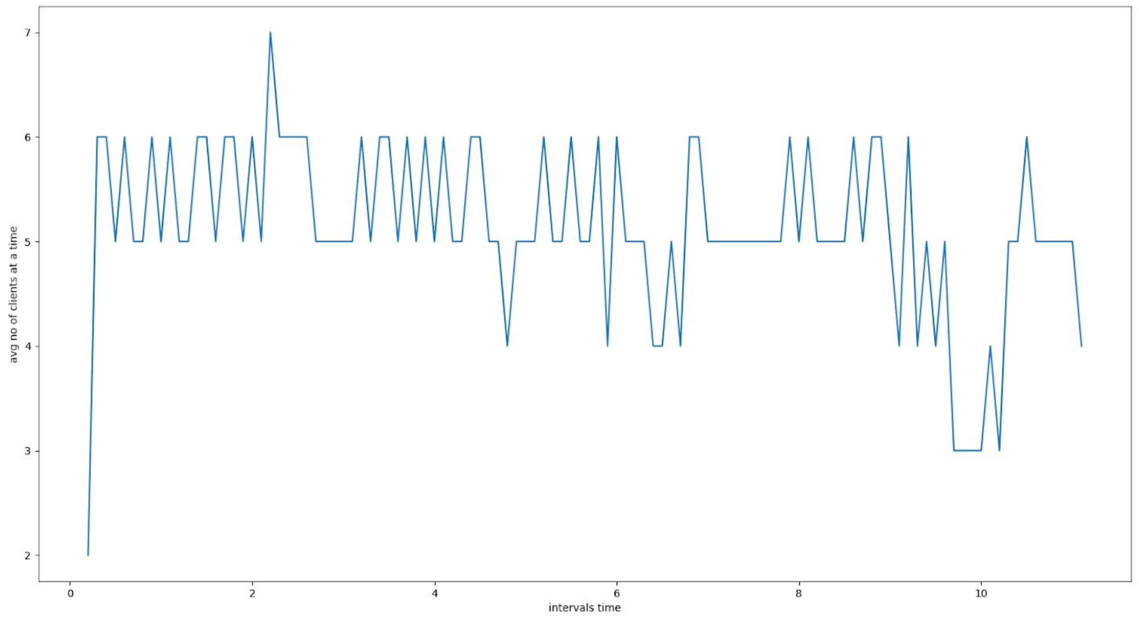
a) SelectTxFormat
b) GetNumberAndTypeOfRus
c) ComputeDlOfdmaInfo
d) InitTxVectorAndParams

Name of functions that are added:

a) merge
b) mergesort
c) decToHex

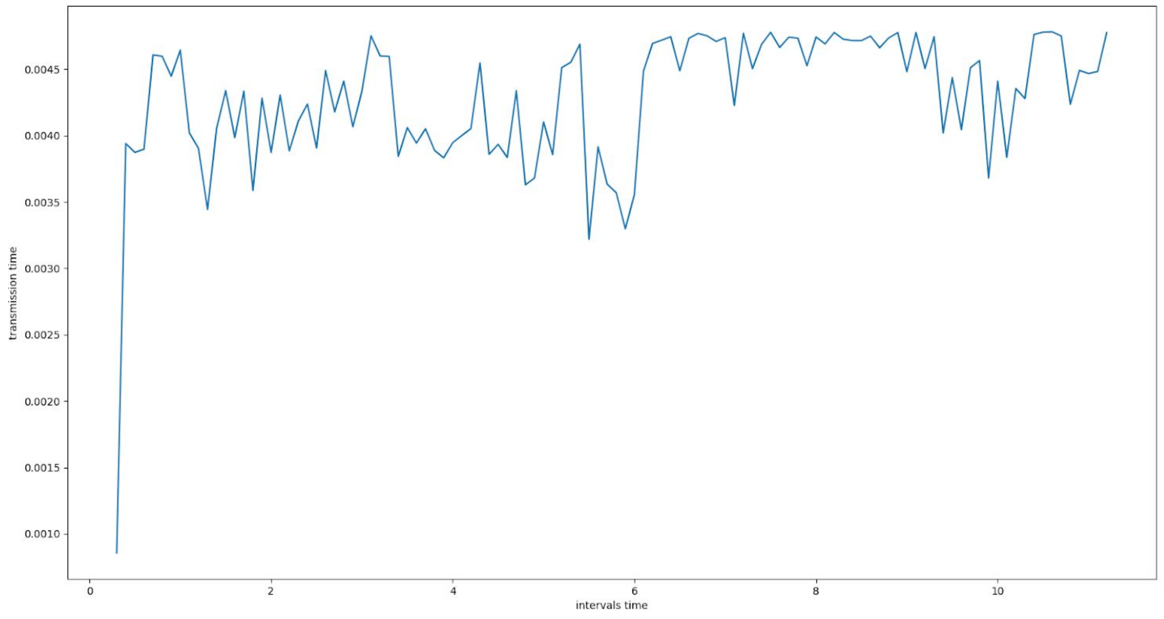
Results For Our Custom Scheduler

1) Average number of clients puts in OFDMA transmission



Points for understanding the above graph- interval time(x-axis) is in seconds
Avg Number of Clients(y-axis)

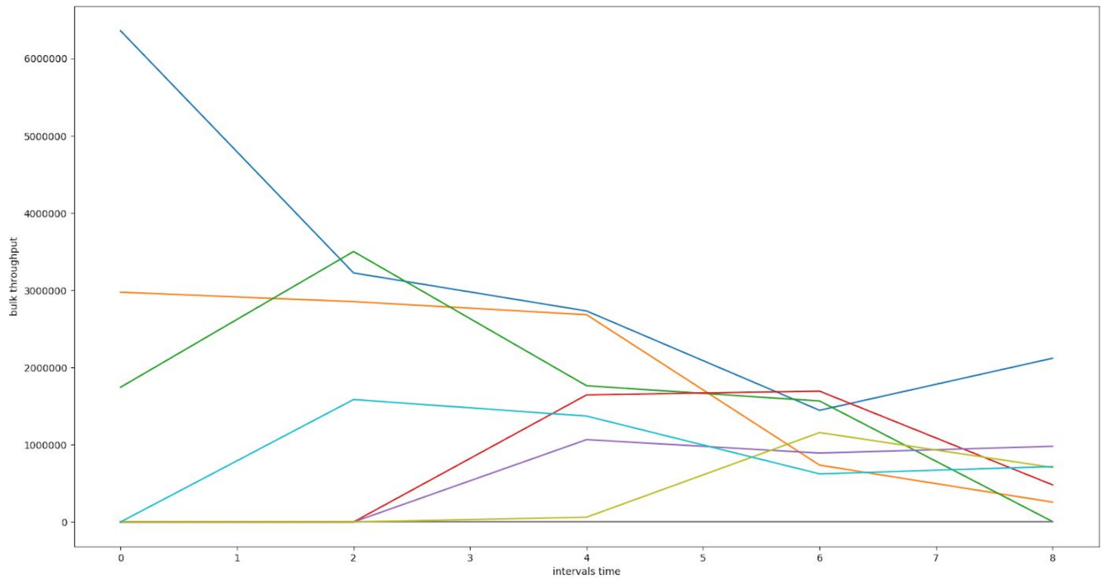
2) OFDMA frame transmission duration



Points for understanding the above graph- interval time(x-axis) is in seconds
Transmission time(y-axis) is also in seconds

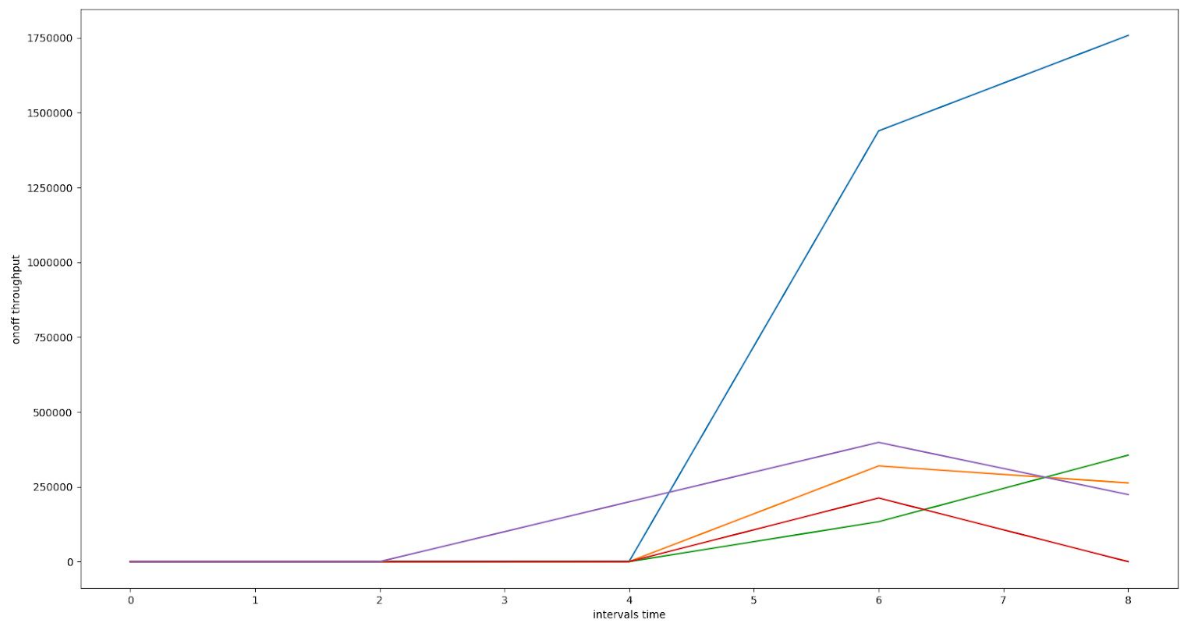
There are single and multi user transmissions both
OFDMA is always providing performance benefits as it reduces the waiting time of the stations to get scheduled and also increases the throughput. So, OFDMA is always beneficial.
But one drawback is that its transmission is more as compared to normal transmissions.

3) Throughput For the BulkSend



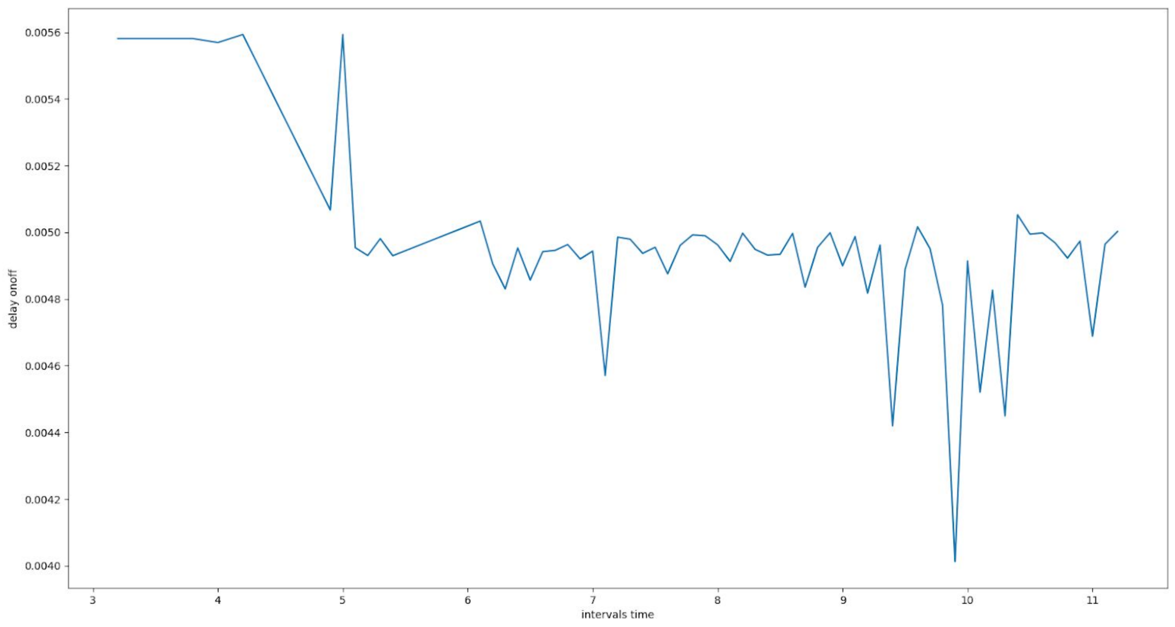
Points for understanding the above graph- interval time(x-axis) is in seconds throughput(y-axis)

4) Throughput For the On Off Client



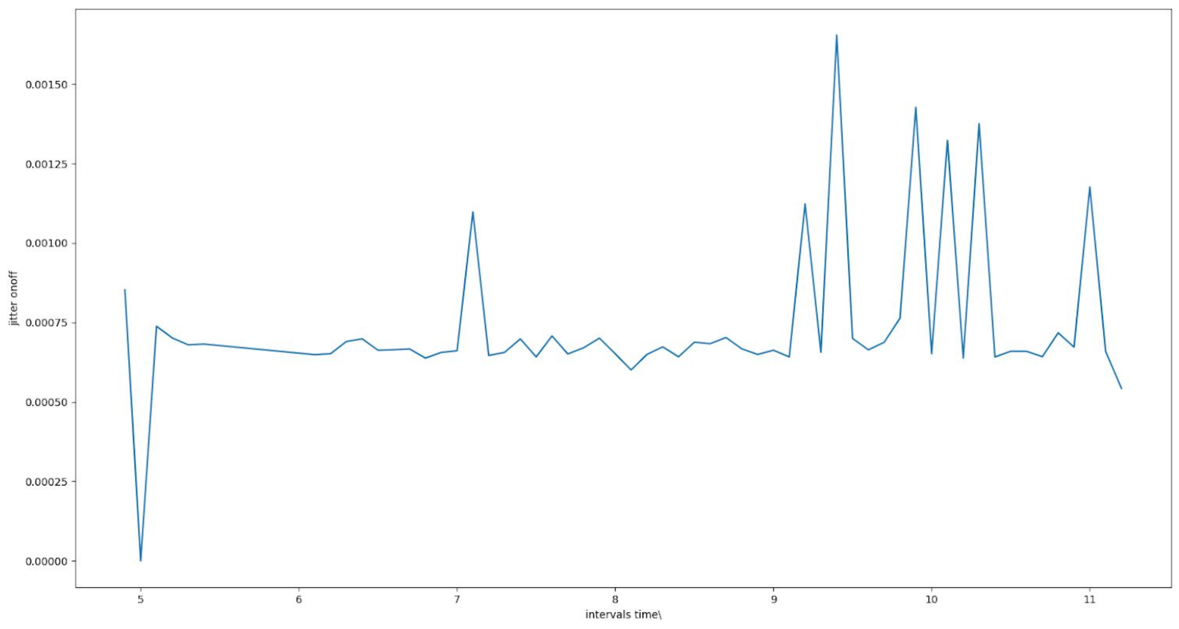
Points for understanding the above graph- interval time(x-axis) is in seconds
throughput(y-axis)

5) Delay for the ON OFF client



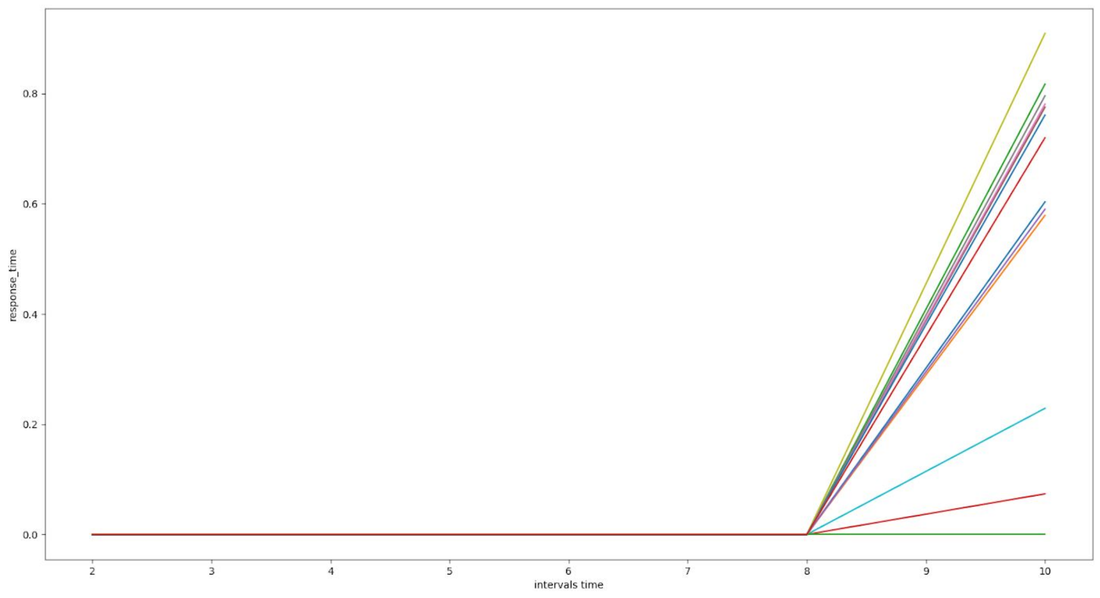
Points for understanding the above graph- interval time(x-axis) is in seconds
Delay on off(y-axis) in seconds

6) Jitter For the ON OFF Client



Points for understanding the above graph- interval time(x-axis) is in seconds
Jitter on off(y-axis) in seconds

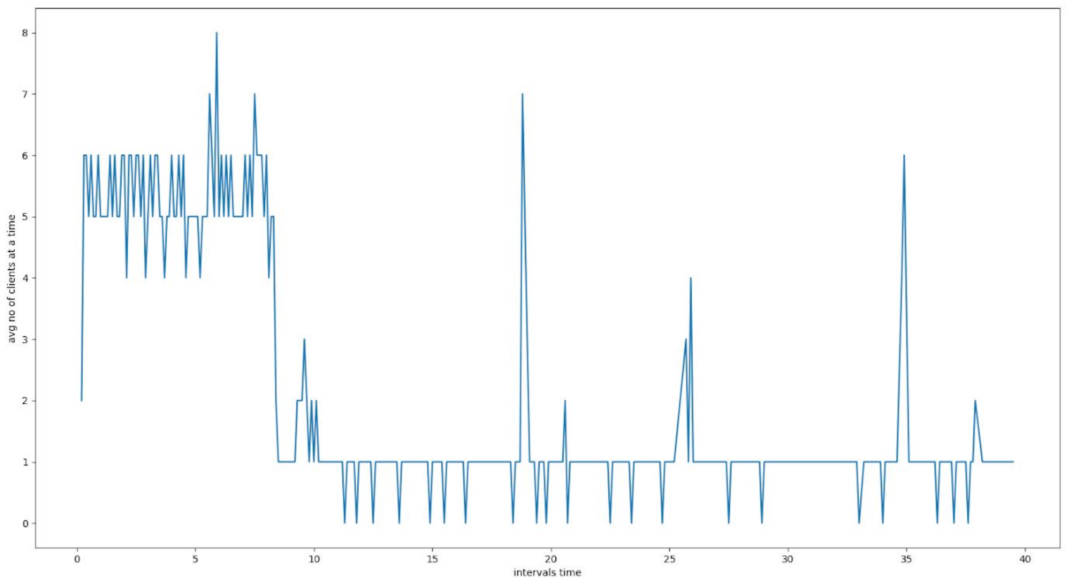
7) HTTP response time



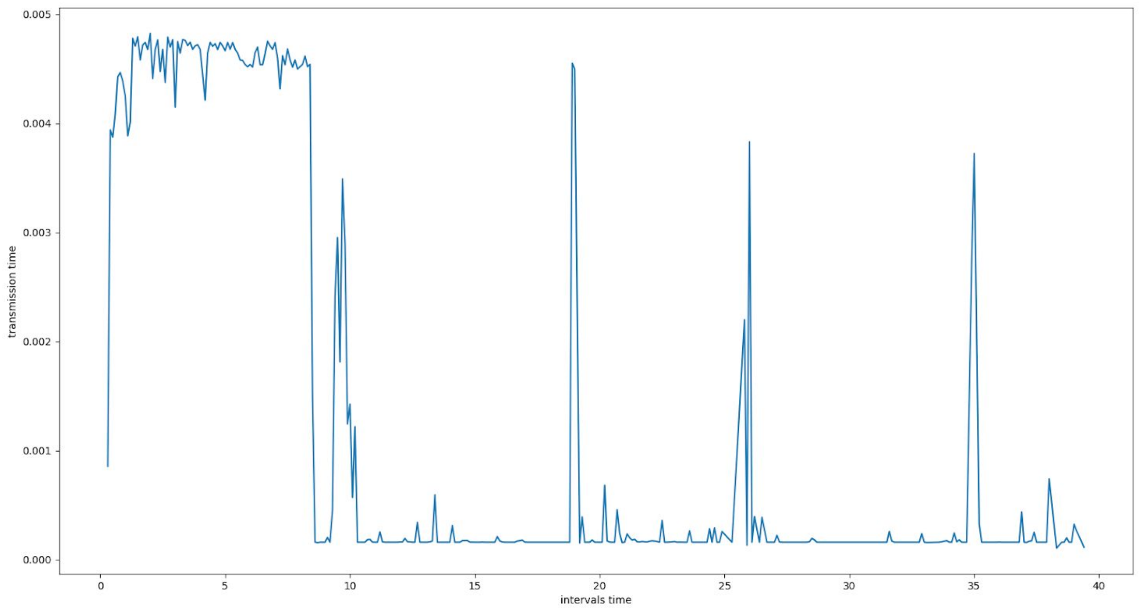
Points for understanding the above graph- interval time(x-axis) is in seconds
response_time(y-axis) in seconds

My earlier plots
Please consider the labelling of the below graphs as the above graphs.

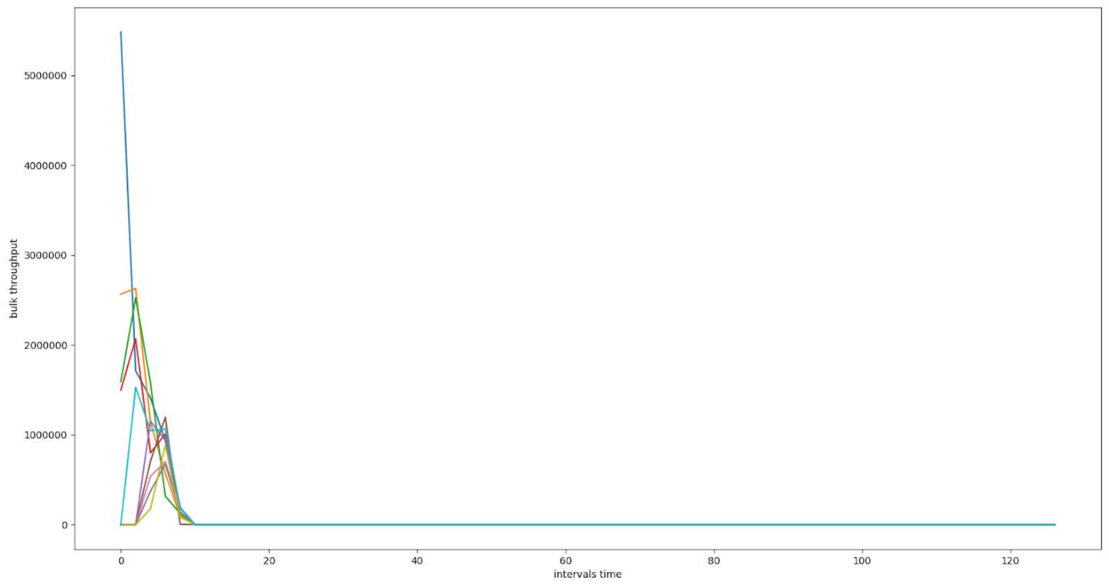
1) Number of clients in OFDMA transmission



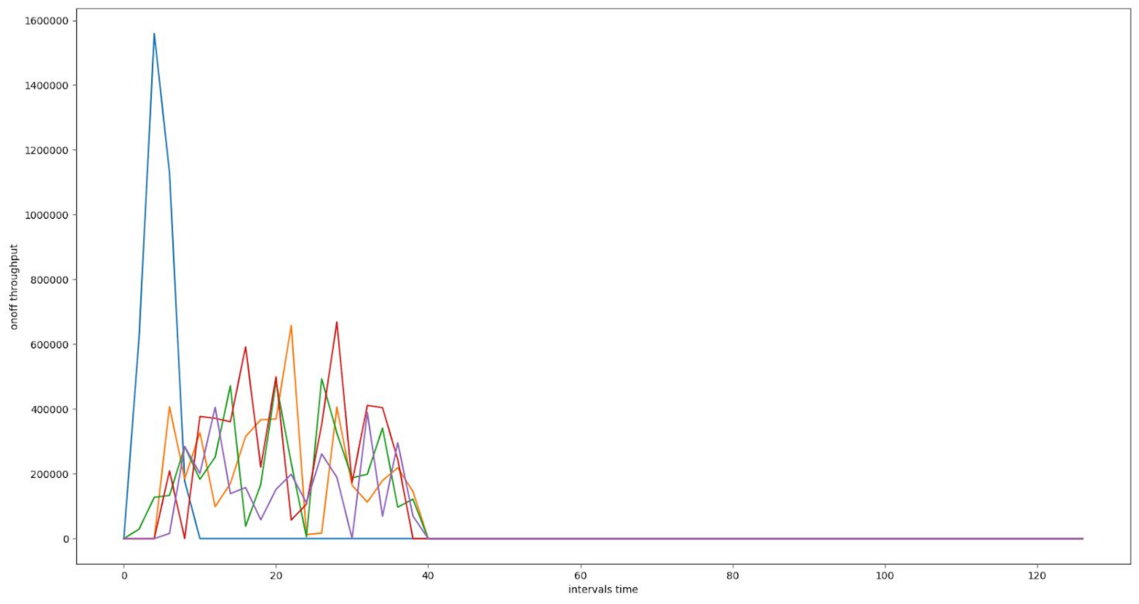
2) OFDMA transmission duration



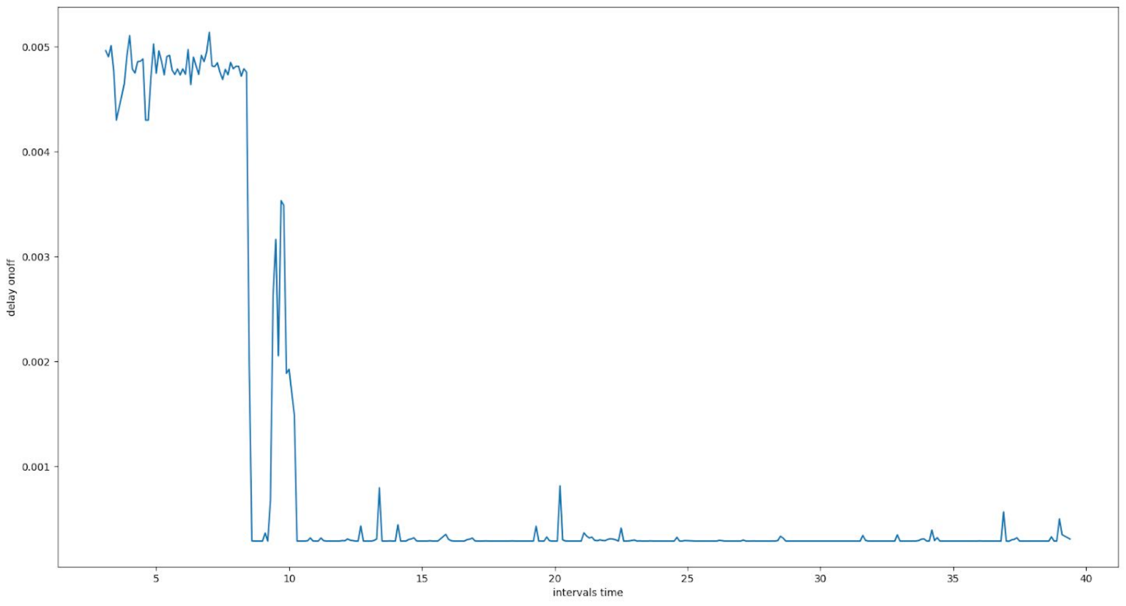
3) Throughput of BulkSend



4) Throughput of On Off

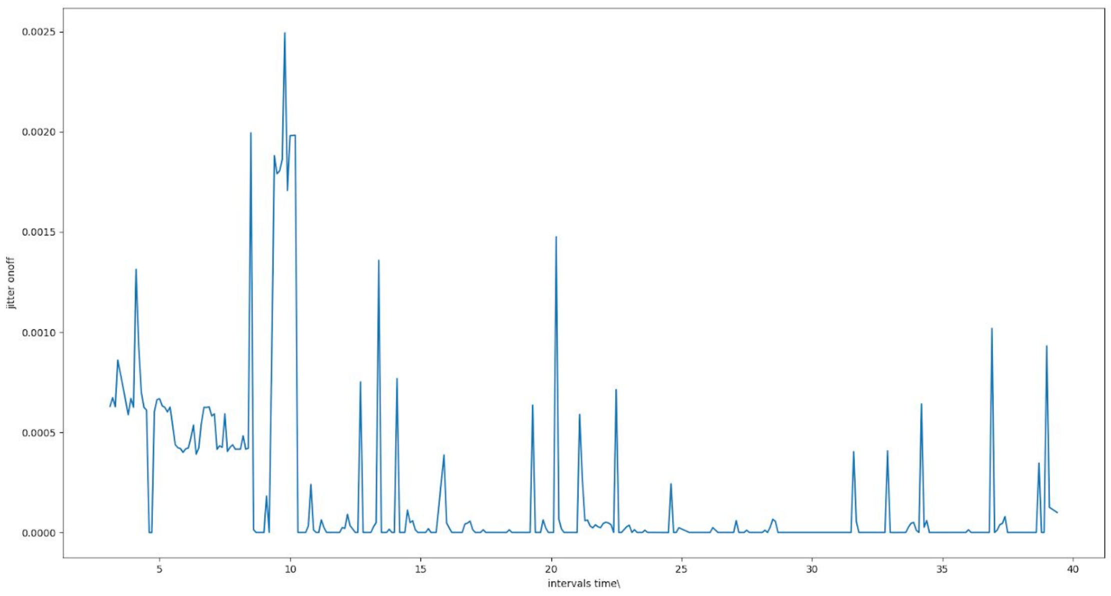


5) Delay On Off

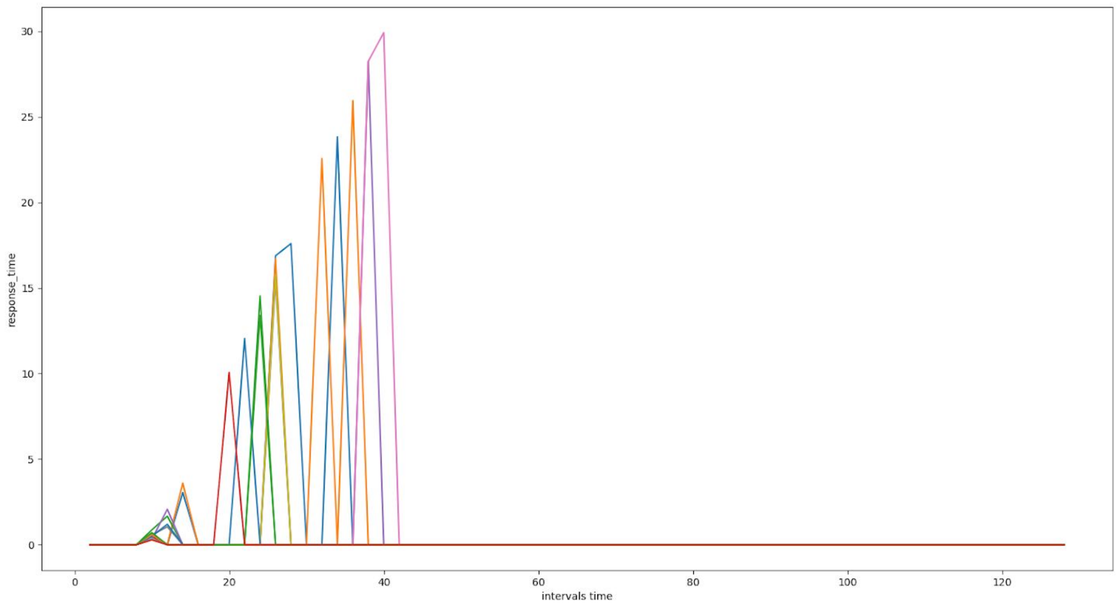


6) Jitter On Off

`



7) HTTP response time



Observations

Obs1) As we can see from the in case of round robin throughput of both on off and bulk send is high initially and then most of the time of simulation they are not transmitting any packet so they are idle most of the time and there is also wastage of bandwidth.

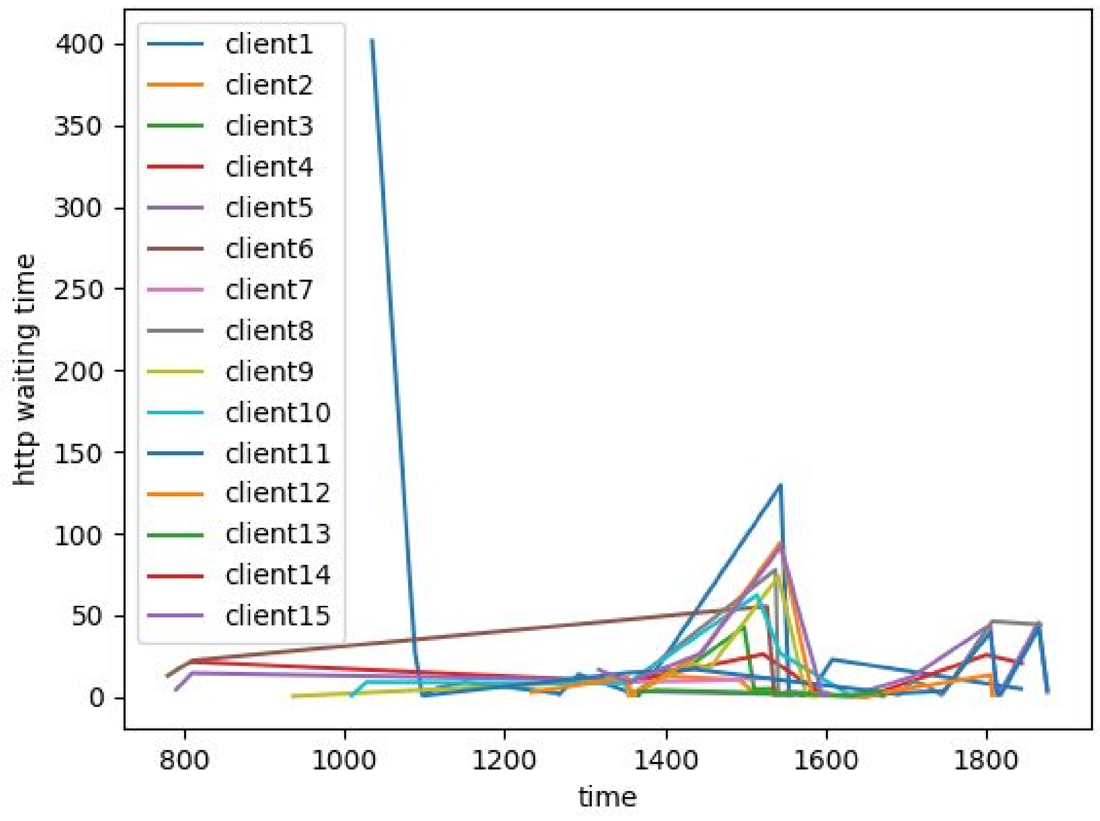
But in the case of our scheduler we have scheduled such that the throughput of the bulksend is high initially and then decreases but it is not the case that it is not sending any packet.
Then after some time the throughput of the on off client gets high as we are giving the priority to the on off client.
And from the graph it can also be seen that the scheduler is throughput driven.

Obs2) In case of round robin the number of clients in OFDMA transmission are very uneven but in case of the custom scheduler the number of clients in OFDMA transmission are very even.

Obs4) If we talk about the transmission time, it is higher in case of our custom scheduler as it is putting more number of clients in OFDMA rather than the round robin.
That is it is showing that the transmission time is high when there is OFDMA and it can be a major drawback of the OFDMA model.

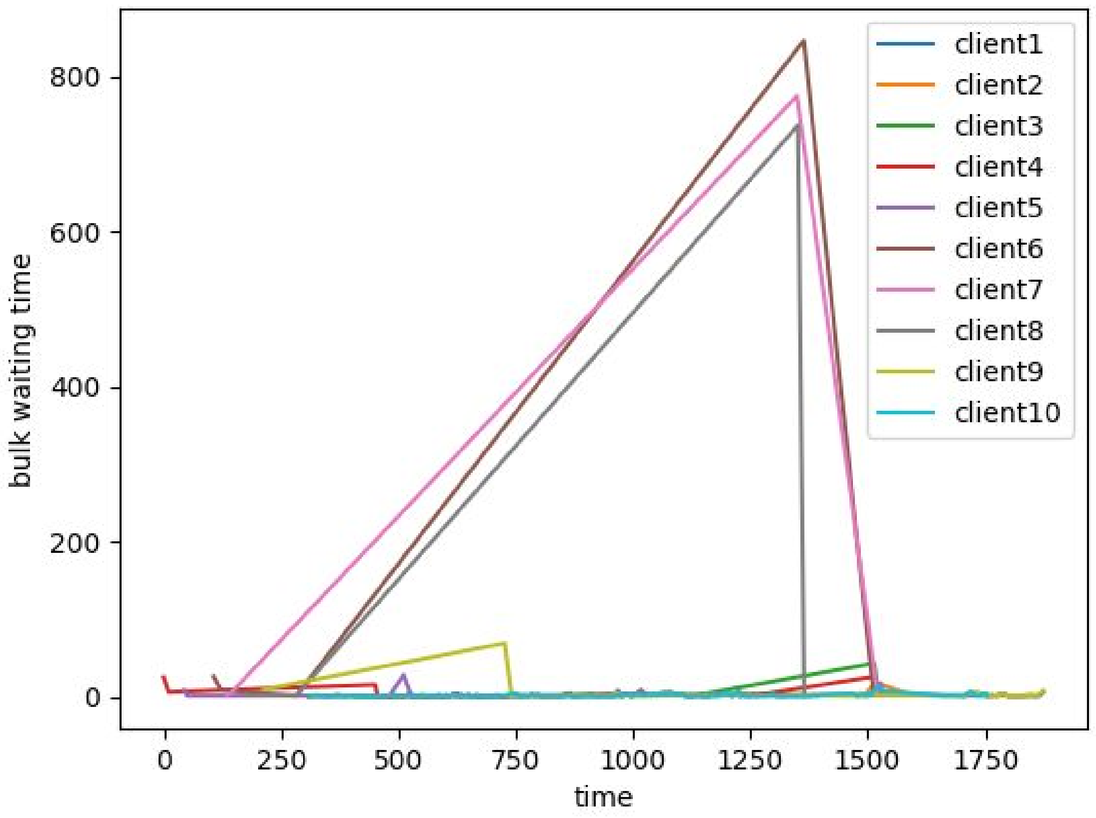
Obs4) Due to high transmission time, delay and jitter in case of custom scheduler are higher than the round robin.

Obs5) Waiting time of stations for in case of custom scheduler as we have prioritized the scheduling of onoff client the on off client waiting time is always zero but there is waiting time for both bulk send and http client.



Points for understanding the above graph- time(x-axis) is in seconds

Http waiting time(y-axis) in seconds



Points for understanding the above graph- time(x-axis) is in seconds
Bulk send waiting time(y-axis) in seconds

Obs6) It is also observed from the graph that http response time is quite more in round robbin as compared to the custom scheduler.
And the round robin is scheduling simultaneously the http client with bulk send which is not preferable as the http client can acquire channels for longer time even they have less amount of data.
But our scheduler is not doing this.