Understanding and Analysis of LTE Schedulers algorithms in NS3

Akshay Kumar - CS23MTECH11022
Sanket Deone - CS23MTECH11034

Changes made to the code to create the events described in the question.

Allocating the position of eNBs :

```
Ptr<ListPositionAllocator> positionAlloc = CreateObject<ListPositionAllocator>();
positionAlloc->Add(Vector(0, 0, 0)); // First eNodeB at origin
positionAlloc->Add(Vector(Enbdistance, 0, 0));         // Second eNodeB
positionAlloc->Add(Vector(0, Enbdistance, 0));          // Third eNodeB
positionAlloc->Add(Vector(Enbdistance, Enbdistance, 0)); // Fourth eNodeB
```

Install Mobility model for eNB :

```
MobilityHelper enbMobility;
enbMobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
enbMobility.SetPositionAllocator(positionAlloc);
enbMobility.Install(enbNodes);
```

Set Random Walk Mobility model for UEs :

```
uemobility.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
                             "Mode", StringValue ("Time"),
                             "Time", StringValue ("2s"),
                             "Speed", StringValue ("ns3::ConstantRandomVariable[Constant=0.0]"),
                             "Bounds", StringValue ("-500|1500|-500|1500"));
```

Random disk placement of UEs within 500 m radius of eNB :

```
// Setting position of UEs attached to eNB 1
   uemobility.SetPositionAllocator("ns3::RandomDiscPositionAllocator",
                        "X", DoubleValue(0.0),
                        "Y", DoubleValue(0.0));
   Ptr<RandomVariableStream> theta1 = CreateObject<ConstantRandomVariable>();
   theta1->SetAttribute("Constant", DoubleValue(0.0));
   Ptr<RandomVariableStream> rho1 = CreateObject<ConstantRandomVariable>();
   rho1->SetAttribute("Constant", DoubleValue(cellRadius));
   uemobility.Install(ueNodes1);

   // Setting position of UEs attached to eNB 2
   uemobility.SetPositionAllocator("ns3::RandomDiscPositionAllocator",
                           "X",
                           DoubleValue(Enbdistance),
                           "Y",
                           DoubleValue(0.0));
   Ptr<RandomVariableStream> theta2 = CreateObject<ConstantRandomVariable>();
   theta2->SetAttribute("Constant", DoubleValue(0.0));
   Ptr<RandomVariableStream> rho2 = CreateObject<ConstantRandomVariable>();
   rho2->SetAttribute("Constant", DoubleValue(cellRadius));
   uemobility.Install(ueNodes2);

   // Setting position of UEs attached to eNB 3
   uemobility.SetPositionAllocator("ns3::RandomDiscPositionAllocator",
                           "X",
                           DoubleValue(0.0),
                           "Y",
                           DoubleValue(Enbdistance));
   Ptr<RandomVariableStream> theta3 = CreateObject<ConstantRandomVariable>();
   theta3->SetAttribute("Constant", DoubleValue(0.0));
   Ptr<RandomVariableStream> rho3 = CreateObject<ConstantRandomVariable>();
   rho3->SetAttribute("Constant", DoubleValue(cellRadius));
   uemobility.Install(ueNodes3);
```

```
// Setting position of UEs attached to eNB 4
uemobility.SetPositionAllocator("ns3::RandomDiscPositionAllocator",
                                "X",
                                DoubleValue(Enbdistance),
                                "Y",
                                DoubleValue(Enbdistance));
Ptr<RandomVariableStream> theta4 = CreateObject<ConstantRandomVariable>();
theta4->SetAttribute("Constant", DoubleValue(0.0));
Ptr<RandomVariableStream> rho4 = CreateObject<ConstantRandomVariable>();
rho4->SetAttribute("Constant", DoubleValue(cellRadius));
uemobility.Install(ueNodes4);
```

Set number of RBs as 50 in UL and DL :

```
lteHelper->SetEnbDeviceAttribute ("DlBandwidth", UintegerValue (50));
lteHelper->SetEnbDeviceAttribute ("UlBandwidth", UintegerValue (50));
```

Setting the seed value for different run :

```
RngSeedManager::SetSeed(RngRun1);
```

Initializing full buffer and half buffer case :

```
// Initializing variable for full buffer and half buffer case base on cmd value.
   if(udpFullBuffer){
    interPacketInterval = MilliSeconds(1.0);
   }
   else{
   interPacketInterval = MilliSeconds(10.0);
   }
```

Attach all UEs to eNodeB :

```
lteHelper->AttachToClosestEnb(ueLteDevs1, enbLteDevs);
lteHelper->AttachToClosestEnb(ueLteDevs2, enbLteDevs);
lteHelper->AttachToClosestEnb(ueLteDevs3, enbLteDevs);
lteHelper->AttachToClosestEnb(ueLteDevs4, enbLteDevs);
```

Graph 1 :

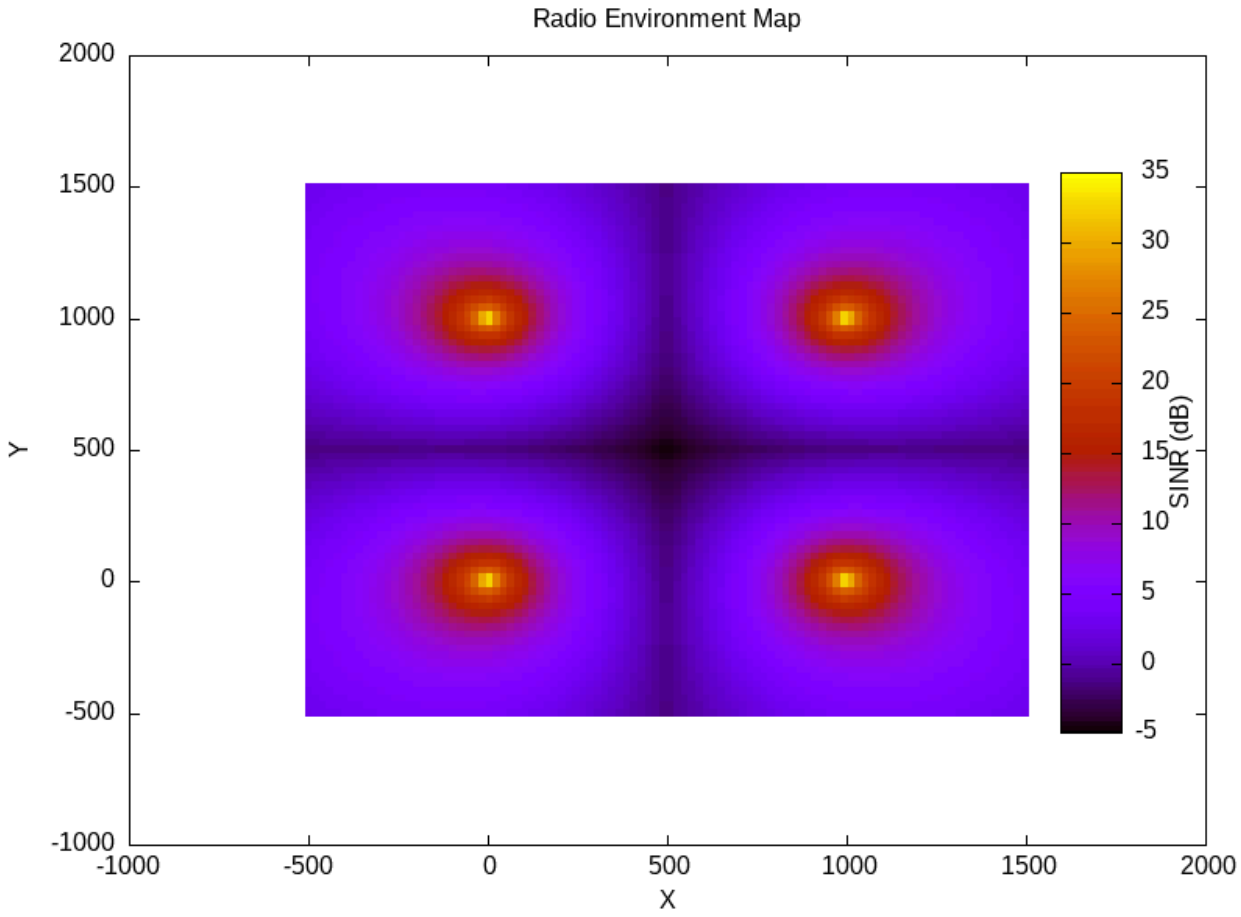
SINR Radio Environment Map (REM) of 4-cell topology given above.

Radio Environment Map (REM), is indeed a structured grid representing the signal-to-noise ratio (SNR) in the downlink of a wireless communication network.

Each grid point corresponds to a specific geographic location, and the value at each point indicates the SNR relative to the base station (eNB) that has the strongest signal at that location.

Add the given below lines before calling Simulator::Run()

```
Ptr<RadioEnvironmentMapHelper> remHelper =
CreateObject<RadioEnvironmentMapHelper> ();
   remHelper->SetAttribute("Channel",
PointerValue(lteHelper->GetDownlinkSpectrumChannel()));
   remHelper->SetAttribute ("OutputFile", StringValue ("rems2.out"));
   remHelper->SetAttribute ("XMin", DoubleValue (-500.0));
   remHelper->SetAttribute ("XMax", DoubleValue (1500.0));
   remHelper->SetAttribute ("XRes", UintegerValue (100));
   remHelper->SetAttribute ("YMin", DoubleValue (-500.0));
   remHelper->SetAttribute ("YMax", DoubleValue (1500.0));
   remHelper->SetAttribute ("YRes", UintegerValue (75));
   remHelper->SetAttribute ("Z", DoubleValue (0.0));
   //remHelper->SetAttribute ("UseDataChannel", BooleanValue (true));
   remHelper->SetAttribute ("RbId", IntegerValue (10));
   remHelper->Install ();
```



The location of eNBs are (0, 0), (1000, 0), (1000,1000) and (0, 1000).

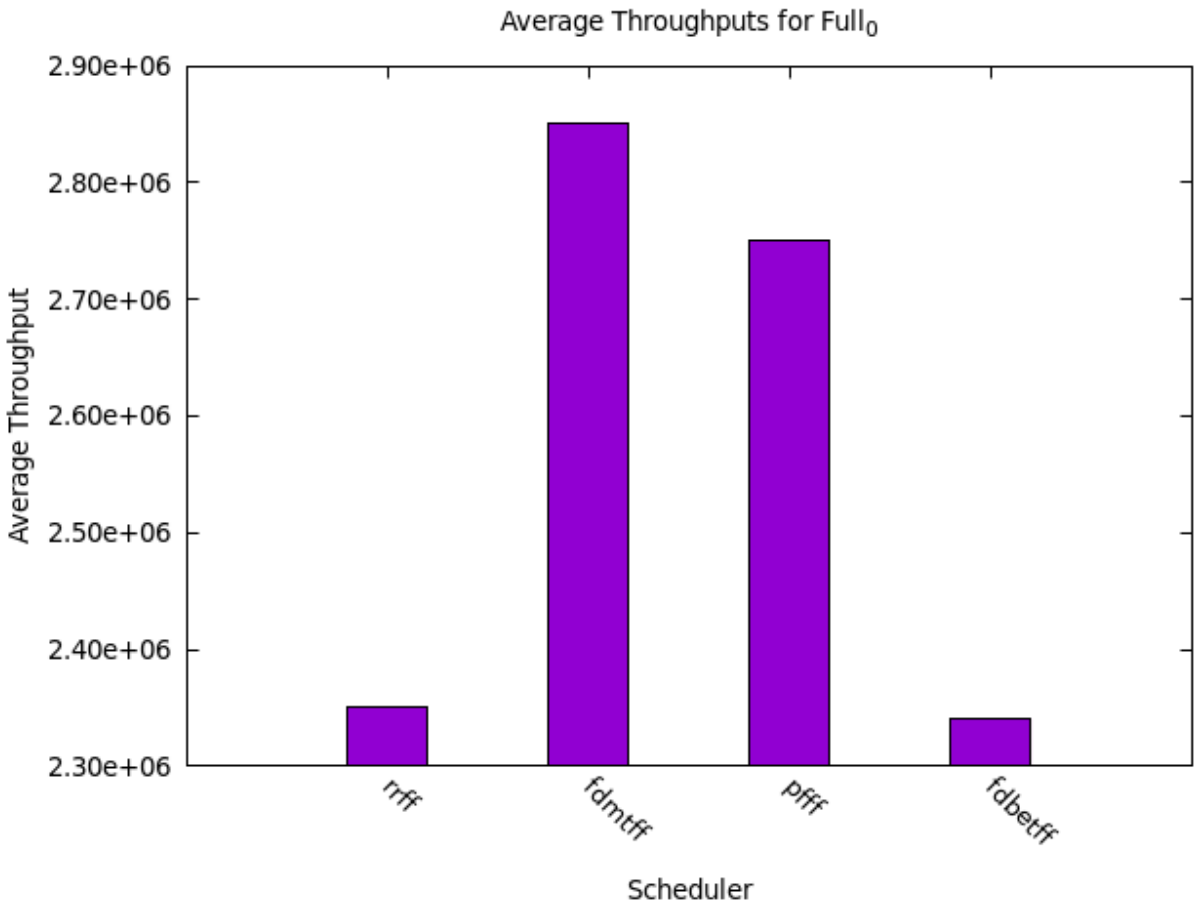
From the above plot we can observe that the SINR value is maximum where the tower is located and it's varying while going away from it.

Graph 2 :

X-axis: Speed (0, 5) m/s;

Y-axis: (Average Aggregate System throughput) with bars for four scheduler algorithms for full buffer scenario.

Case 1: Speed = 0 m/s

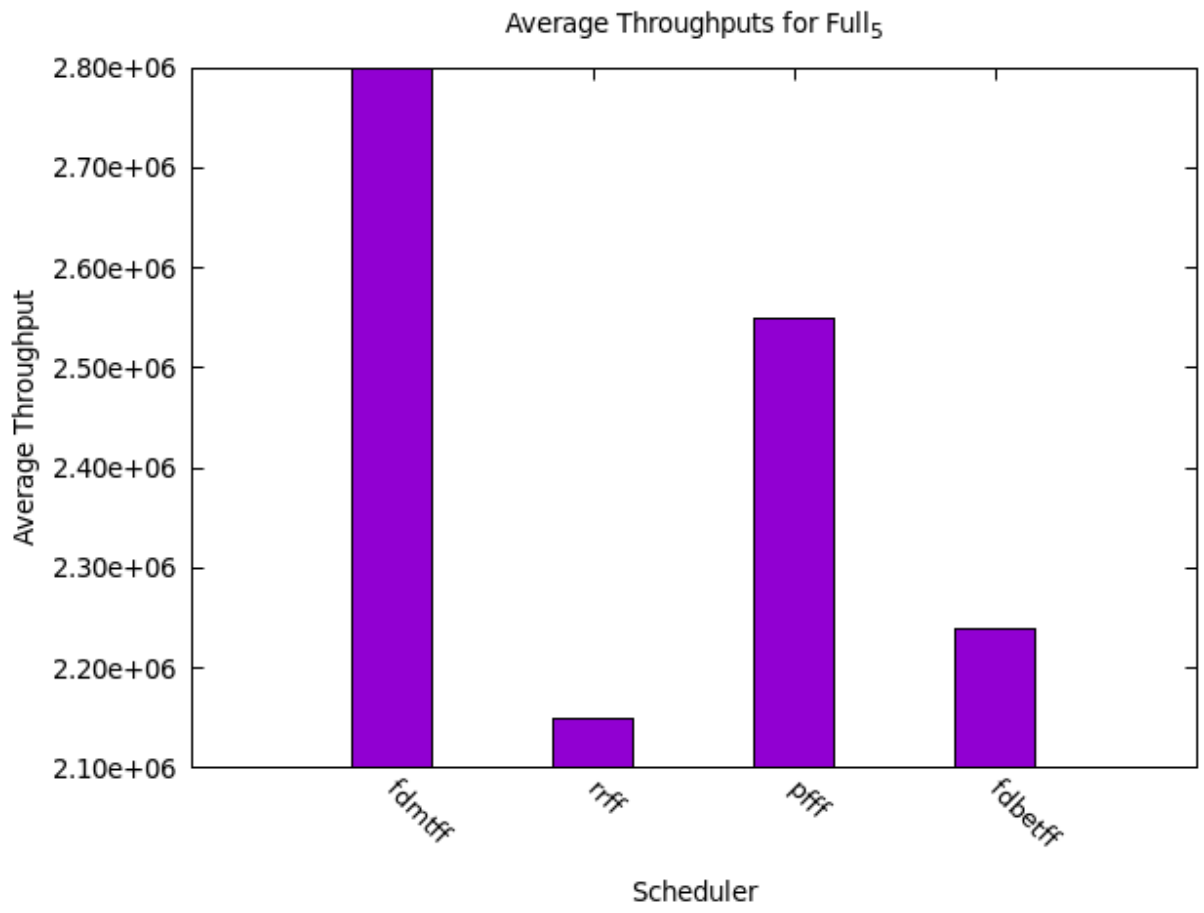


The Max Throughput (MT) scheduler demonstrates superior performance compared to other schedulers.

Round Robin (RR) exhibits the least favorable performance among the tested schedulers.

The throughput ranking, from highest to lowest, is as follows: MT > PF > BATS > RR.

Case 2: Speed = 5m/s



The Max Throughput (MT) scheduler outperforms all others in terms of performance.

Round Robin (RR) scheduler demonstrates the poorest performance among the tested schedulers.

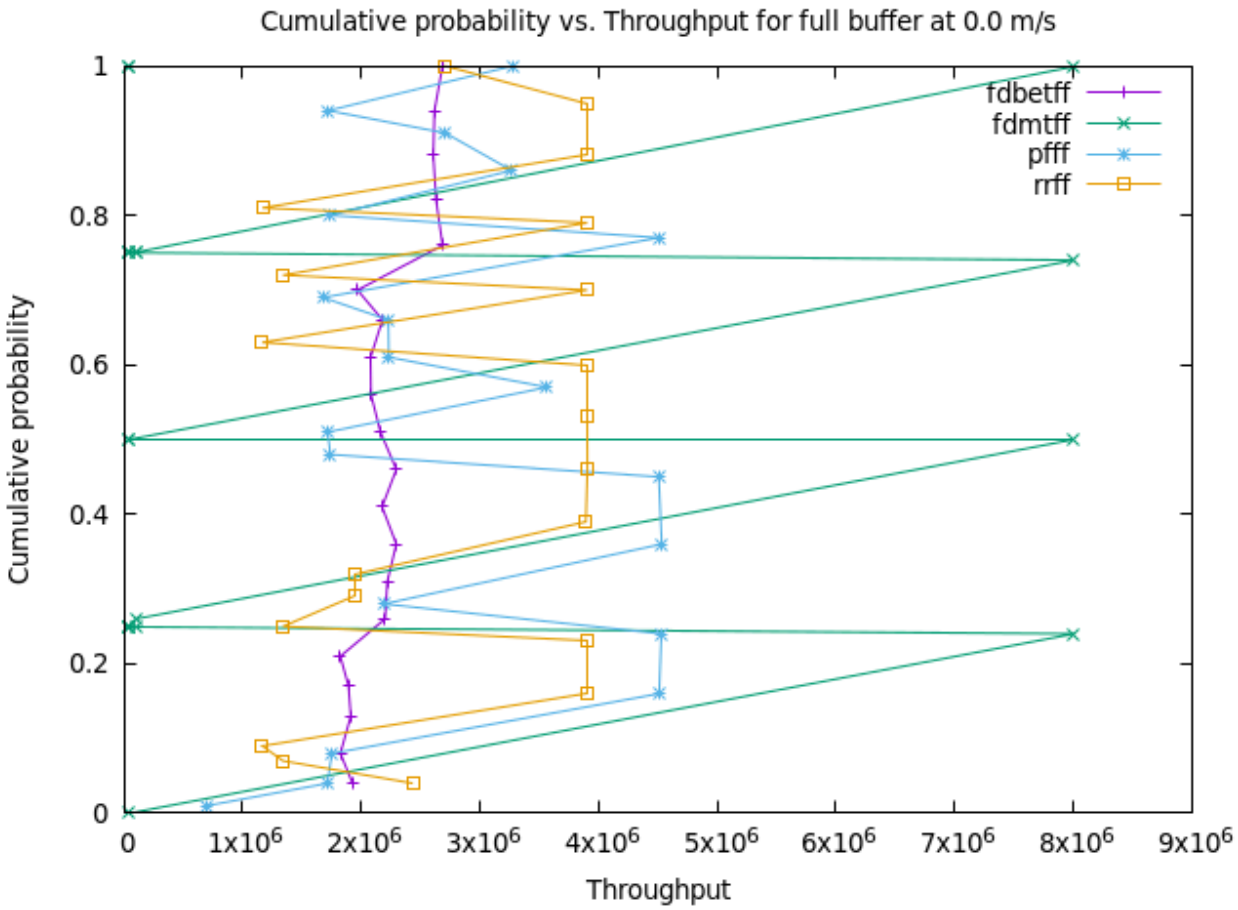
The throughput ranking, from highest to lowest, is as follows : MT > PF > BATS > RR.

The throughput remains unaffected by the speed of the User Equipment (UE).

Graph 3:

Throughput CDF plot for different schedulers at Speed (0,5) m/s for full buffer scenario; One curve each for 0 m/s and 5 m/s.

Case 1: Speed = 0m/s



For the case of **full buffer and speed=0 m/s**, we calculated per UE average over 5 runs for all the UEs.

Then we sorted all the 20 values in the increasing order and plotted those values on the x axis and the corresponding CDF value on the y-axis.

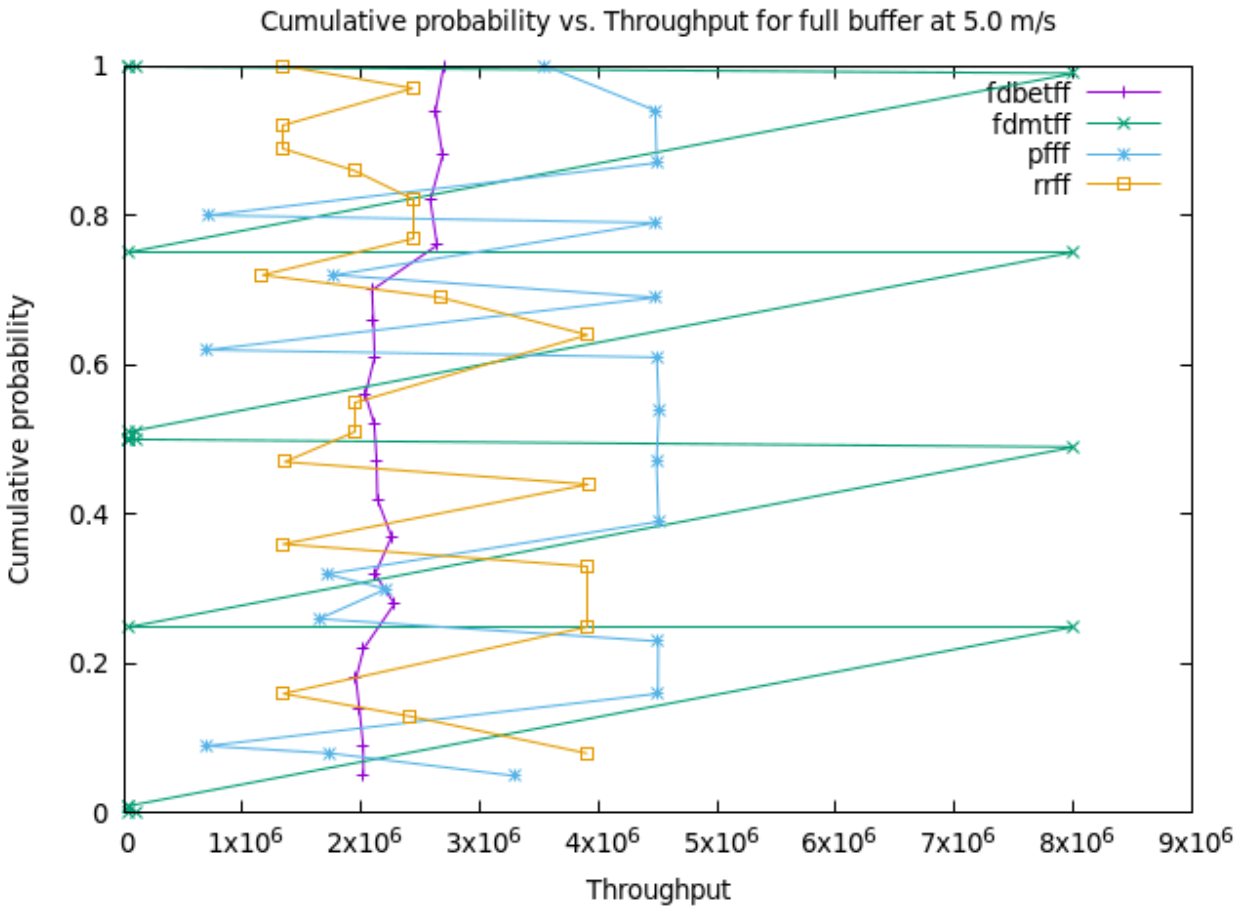
The Max Throughput (MT) algorithm exhibits significant fluctuations in its curves within the throughput range of 0 to 8 Mbps, which represents the maximum throughput in this scenario.

For the Proportional Fairness (PF) algorithm, the curve reaches a Cumulative Distribution Function (CDF) value of 1 for very low throughputs, suggesting that the maximum throughput ranks second in this instance.

The order of maximum throughput is as follows: MT > PF > BATS > RR.

In scenarios where the buffer is full, the trends for both maximum throughput and average throughput remain consistent.

Case 2: Speed = 5m/s



The MT algorithm's curve exhibits significant fluctuations within the throughput range of 0 to 8 Mbps, representing the maximum throughput in this scenario.

For the PF algorithm, the curve achieves a CDF value of 1 for very low throughputs, indicating that the maximum throughput ranks second.

In terms of maximum throughput, the order is MT > PF > BATS > RR.

In cases where the buffer is full, the trends for both maximum throughput and average throughput remain consistent.

Throughput remains unaffected by speed, as evidenced by the similarity of graphs for both speeds

Graph 4:

SINR/Instantaneous throughput values for UE 0 in the simulation for one seed (RngRun1).

X-axis: Time in msec,

Y-axis: SINR and Instantaneous throughputs of UE0 for Speed of 0 m/s for all four schedulers for full buffer scenario.

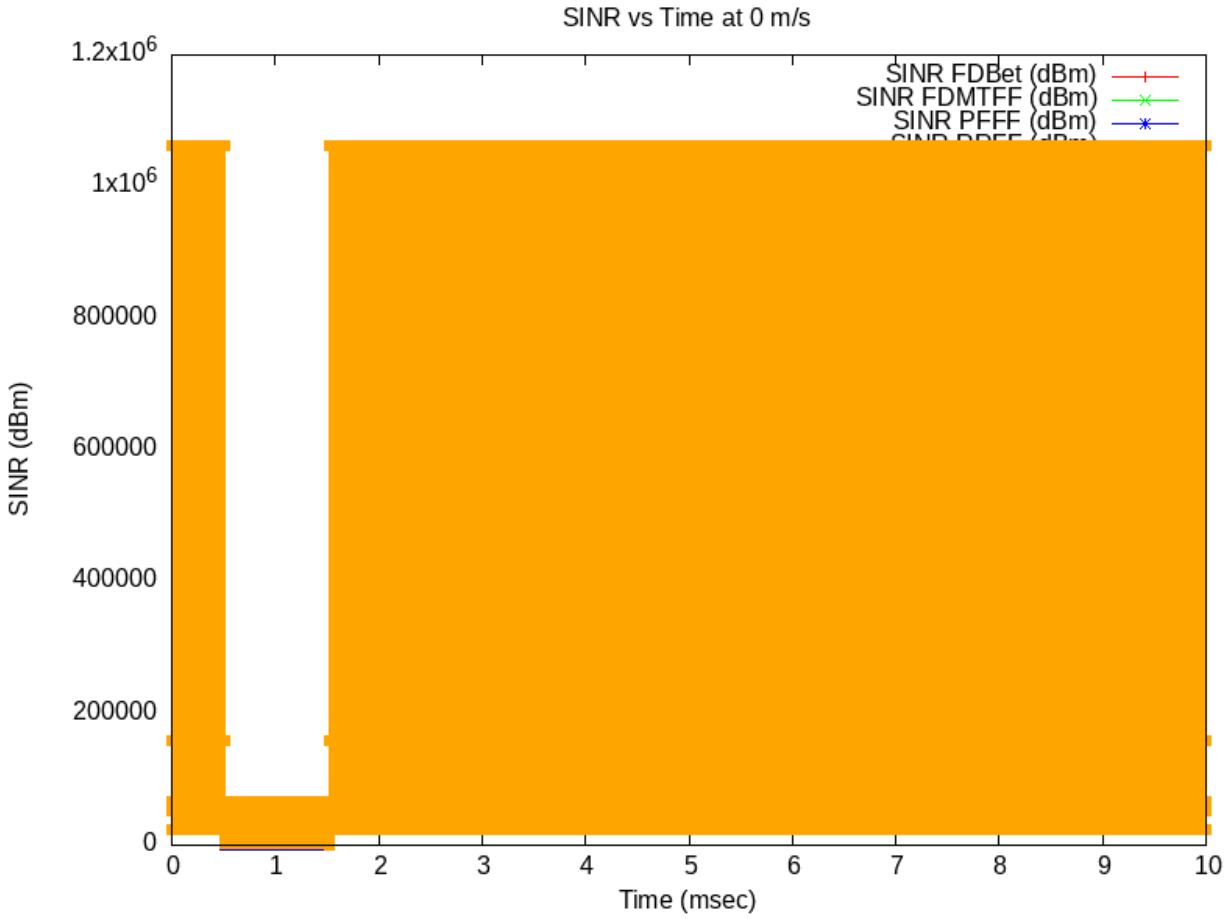
For each scheduler, we get DlRlcStats.txt and DlRsrpSinrStats.txt. Since, we need to plot the SINR and Throughput values only for UE 0, we have to parse these stat files.

```
awk '$2==1' DlRsrpSinrStats_fdbet.txt > sinr_fdbet.txt
awk '$2==1' DlRsrpSinrStats_fdmtff.txt > sinr_fdmtff.txt
awk '$2==1' DlRsrpSinrStats_pfff.txt > sinr_pfff.txt
awk '$2==1' DlRsrpSinrStats_rrff.txt > sinr_rrff.txt

awk '$4==1' DlRlcStats_fdbet.txt > filter_throughput_fdbet.txt
awk '$4==1' DlRlcStats_fdmtff.txt > filter_throughput_fdmtff.txt
awk '$4==1' DlRlcStats_pfff.txt > filter_throughput_pfff.txt
awk '$4==1' DlRlcStats_rrff.txt > filter_throughput_rrff.txt

awk '{ print $2, $10 * 0.000032}' filter_throughput_fdbet.txt > throughput_fdbet.txt
awk '{ print $2, $10 * 0.000032}' filter_throughput_fdmtff.txt >
throughput_fdmtff.txt
awk '{ print $2, $10 * 0.000032}' filter_throughput_pfff.txt > throughput_pfff.txt
awk '{ print $2, $10 * 0.000032}' filter_throughput_rrff.txt > throughput_rrff.txt
```

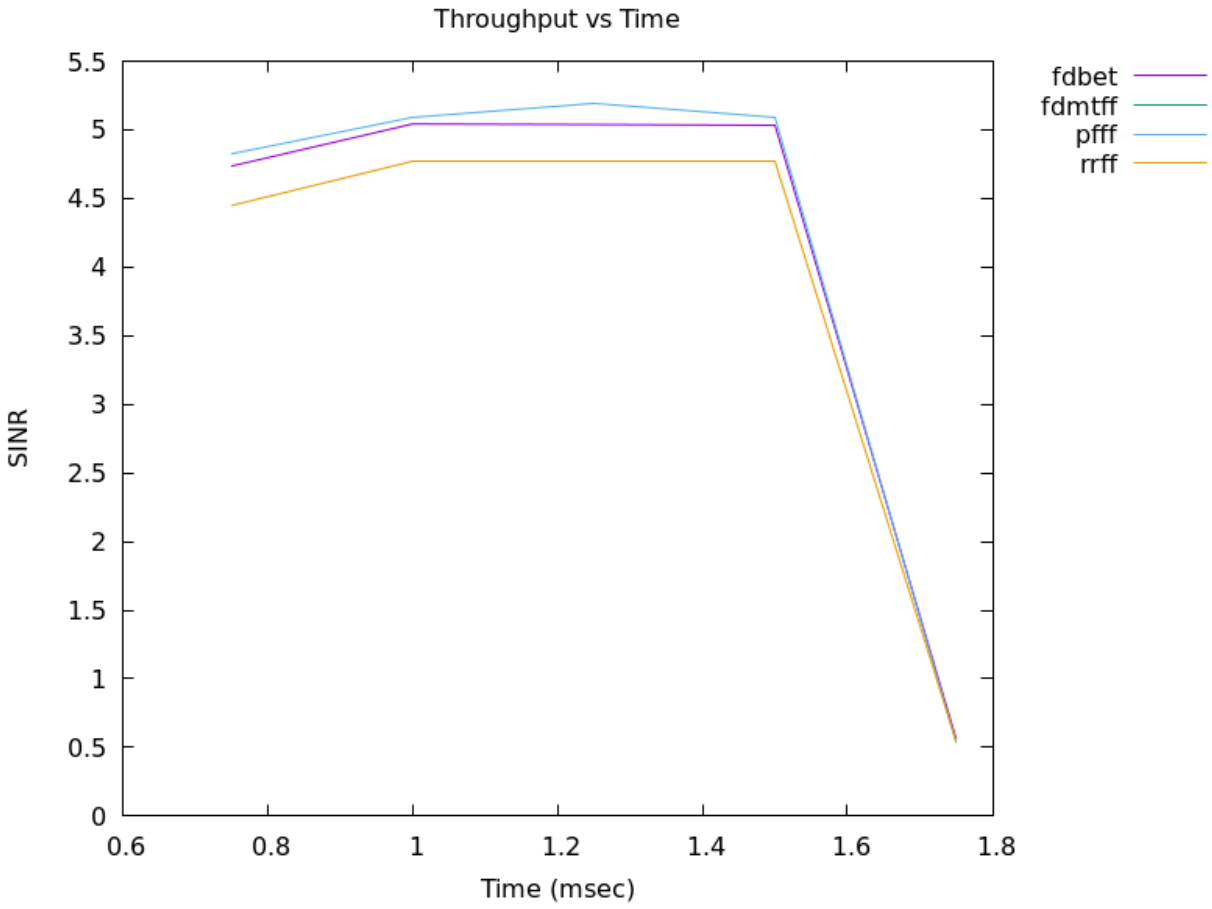
SINR plot:



Proportional Fair (PF) scheduler has the highest SINR value of 1 x 10^6 dbm.

There isn't much difference in SINR for Max Throughput (MT) scheduler, PF, Blind Average Throughput (BAT) scheduler and Round Robin (RR) scheduler for time t > 1s.

Throughput plot :



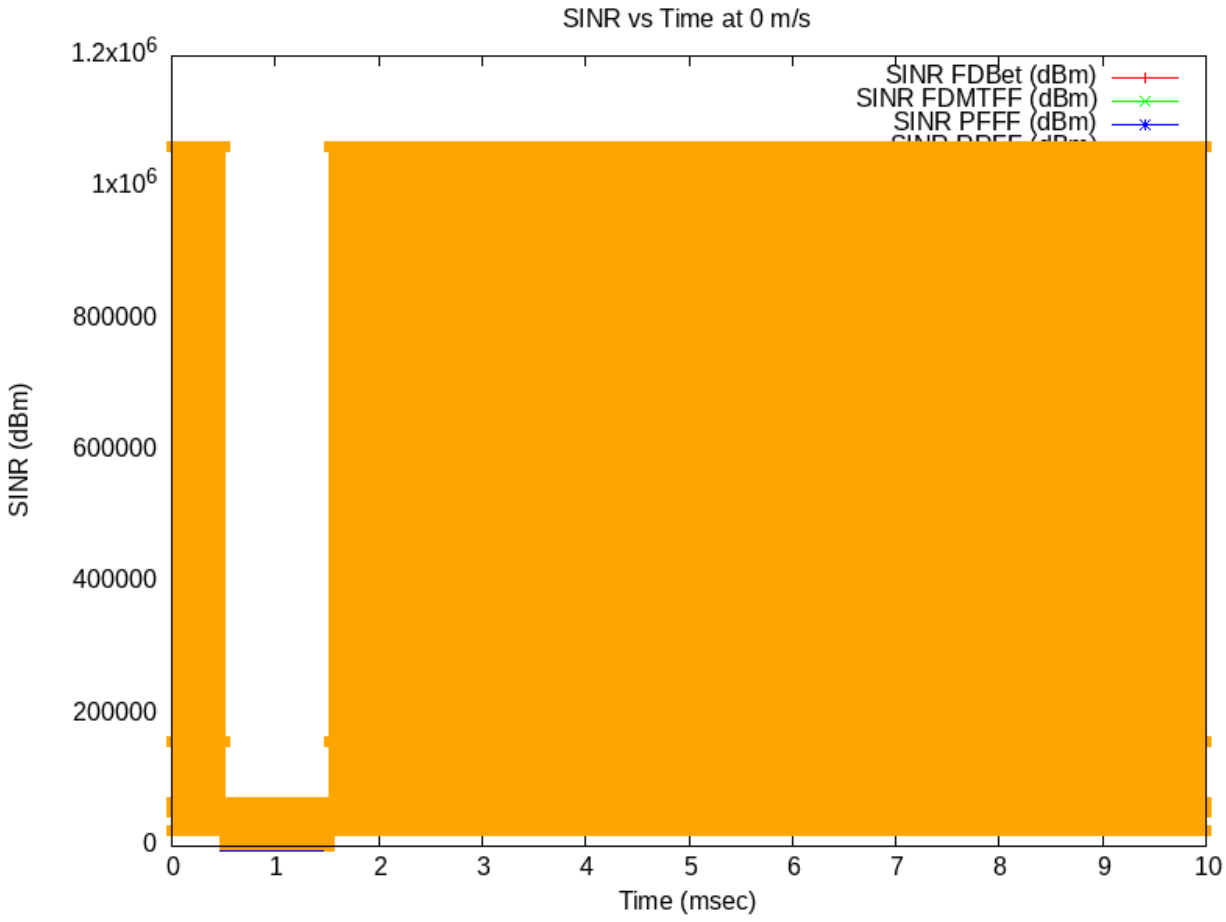
Throughput for Proportional Fair (PF) scheduler was highest followed by Blind Average throughput (BAT) scheduler and then Round Robin (RR) scheduler. All the schedulers behave in a similar manner, the throughput first increases, remains constant for a while and then decreases. At t = 0.75 s, the value of throughput is almost the same for all the schedulers.

Graph 5:

SINR/Instantaneous throughput values for UE 0 in the simulation for one seed (RngRun1).

X-axis: Time in msec,
Y-axis: SINR and Instantaneous throughputs of UE0 for Speed of 5 m/s for all four schedulers for full buffer scenario

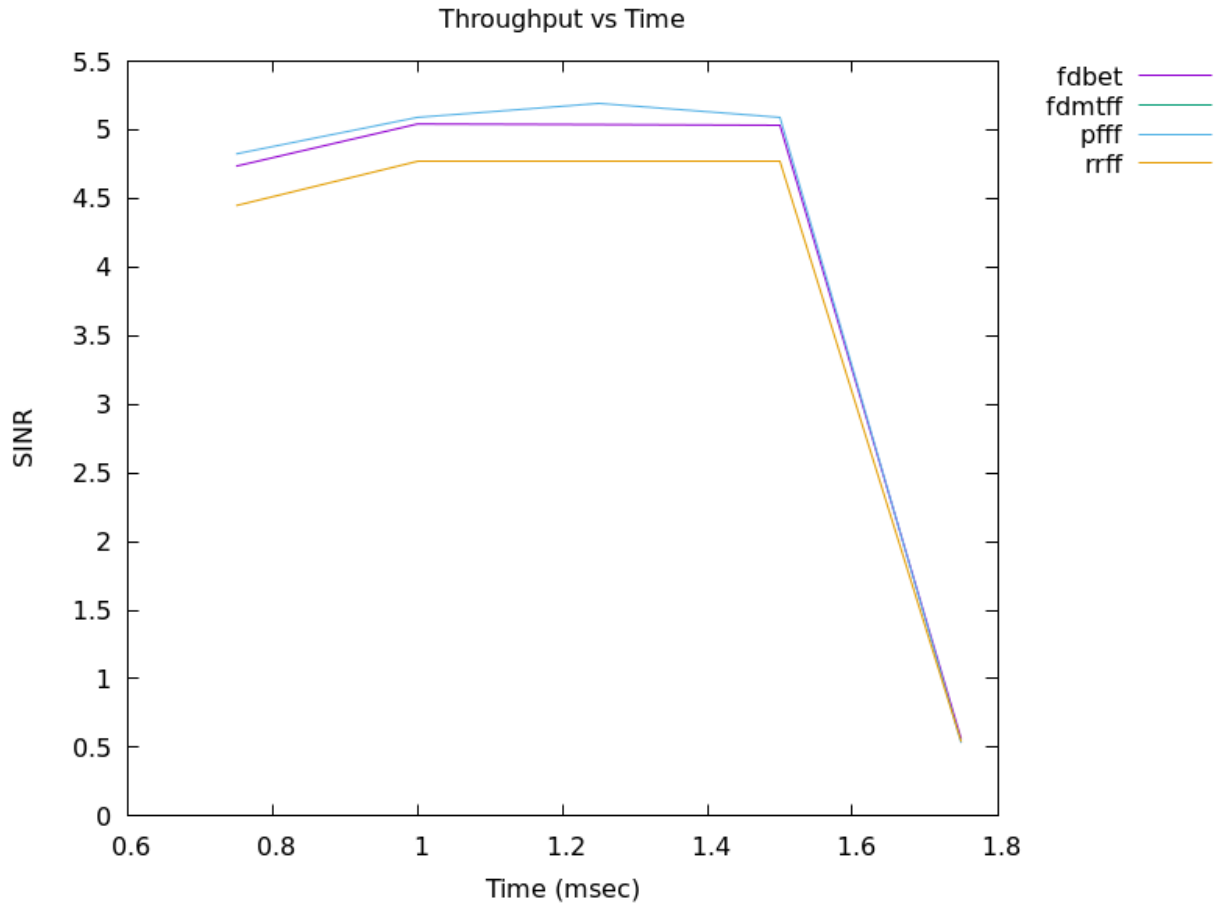
SINR plot :



This case is the same as case 4, except that the UE mobility speed is 5 m/s. Changing this, will not affect the nature of the graphs.

Proportional Fair (PF) scheduler has the highest SINR value of 1 x 10^6 dbm. There isn't much difference in SINR for Max Throughput (MT) scheduler, PF, Blind Average Throughput (BAT) scheduler and Round Robin (RR) scheduler for time t > 1s.

Throughput plot :



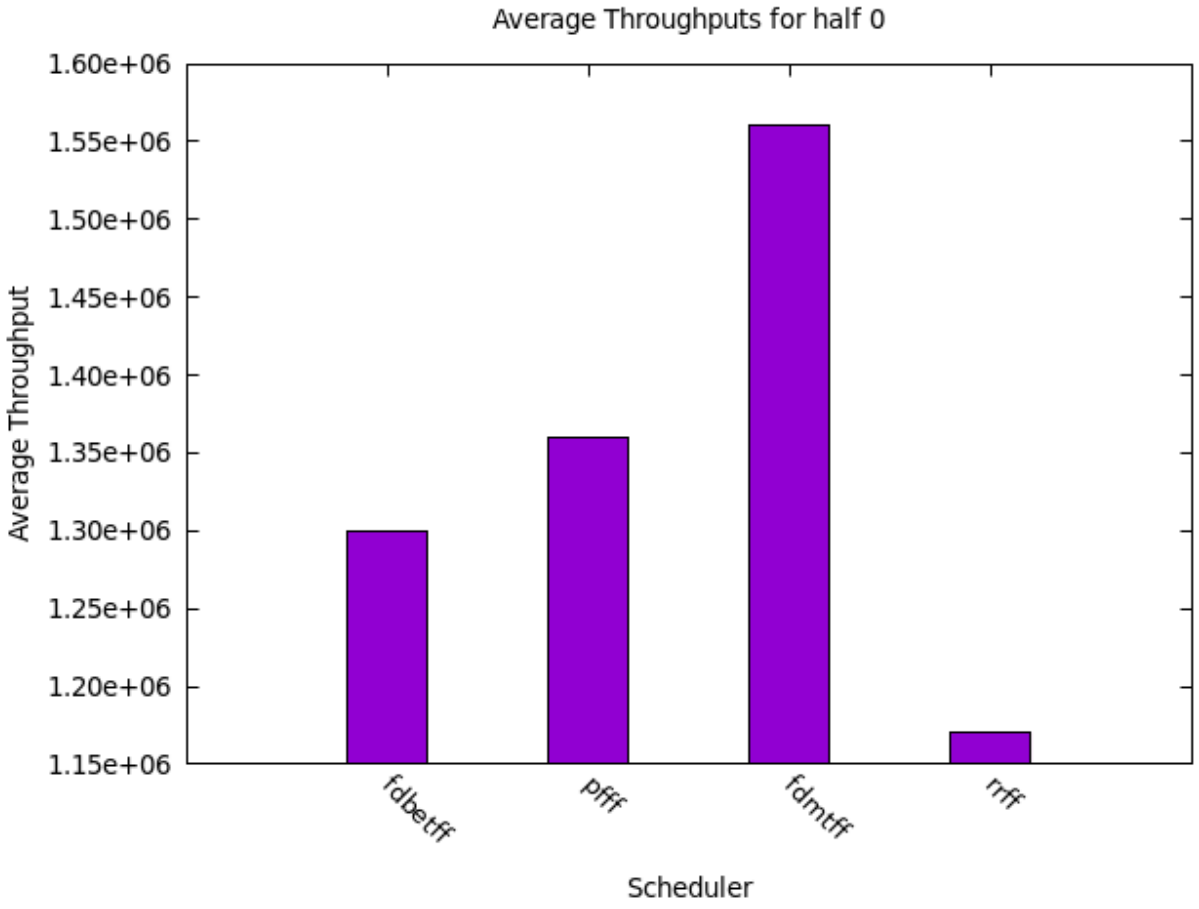
This case is the same as case 4, except that the UE mobility speed is 5 m/s. Changing this, will not affect the nature of the graphs.

Throughput for Proportional Fair (PF) scheduler was highest followed by Blind Average throughput (BAT) scheduler and then Round Robin (RR) scheduler. All the schedulers behave in a similar manner, the throughput first increases, remains constant for a while and then decreases. At t = 1.25 s, the value of throughput is almost the same for all the schedulers.

Graph 6 :

X-axis: Speed (0) m/s;
Y-axis: (Average Aggregate System throughput) with bars for four scheduler algorithms

Case 1 : Speed = 0m/s

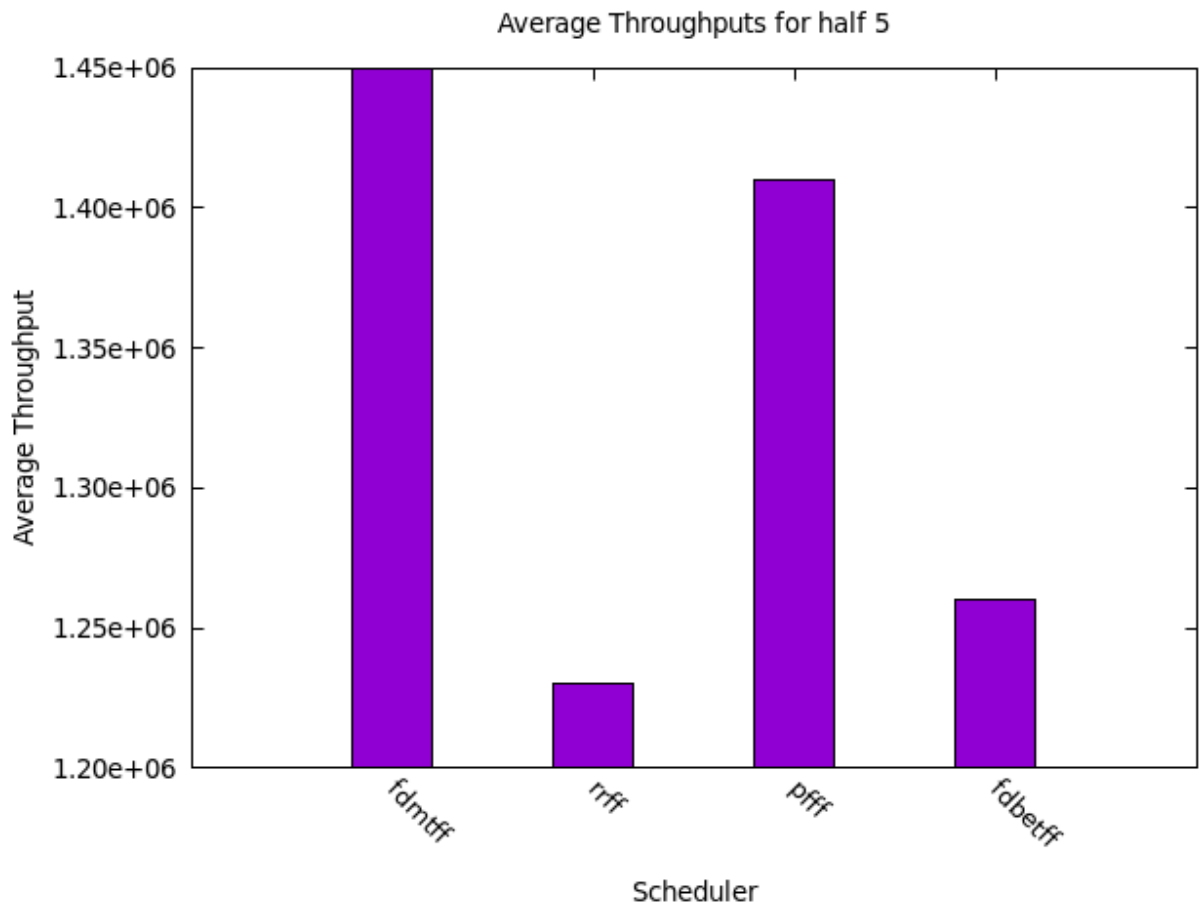


The Proportional Fairness (PF) scheduler demonstrates superior performance compared to other schedulers.

BATS scheduler exhibits the least favorable performance among the tested schedulers.

The throughput ranking, from highest to lowest, is as follows: MT > PF > BATS > RR.

Case 2: Speed = 5m/s



The Proportional Fairness (PF) scheduler shows the highest performance among all schedulers.

Round Robin (RR) scheduler displays the poorest performance among the tested schedulers.

In terms of throughput, the order is as follows: MT > PF > BATS > RR.

Throughput remains consistent regardless of the User Equipment (UE) speed.

Analysis :

1. Round Robin (RR) :

Round Robin (RR) scheduling is a method of resource allocation that cyclically assigns resources to users without taking into account their channel conditions. This straightforward approach aims to ensure fairness by giving each user an equal share of resources.

Although RR scheduling achieves fairness, it can lead to reduced throughput because it doesn't consider reported downlink Signal-to-Noise Ratio (SNR) values when determining transmission rates. This limitation becomes apparent when examining performance graphs.

2. Proportional fair (PF):

The primary objective of the Proportional Fair algorithm is to strike a balance between throughput and fairness among all users. It aims to maximize total network throughput while ensuring that all users receive at least a minimal level of service.

As a result, Proportional Fair scheduling typically yields high cell throughput while maintaining fairness among users. Performance graphs often demonstrate that Proportional Fair scheduling consistently delivers optimal results.

3. Max Throughput(Best CQI):

This scheduling algorithm prioritizes assigning resource blocks to users with the best radio link conditions. Known as Best Channel Quality Indicator (CQI), it allocates resource blocks based on the highest CQI value observed for each user.

Analyzing performance graphs reveals that this method performs comparably to the Proportional Fair (PF) algorithm and even surpasses it when the buffer is full. However, PF generally outperforms Best CQI scheduling in most scenarios.

4. BATS:

The Blind Average Throughput scheduler aims to distribute throughput equally among all User Equipments (UEs) under an eNB. In the time domain version (TD-BET), the scheduler selects the UE with the highest priority metric and allocates all Resource Block Groups (RBGs) to this UE.

The scheduler then repeats this process for a new UE with the lowest past average throughput until all RBGs are allocated. The goal is to achieve equal throughput among all UEs in every Transmission Time Interval (TTI). However, since the scheduler prioritizes equal throughput for all UEs, it tends to perform less effectively compared to other algorithms across various scenarios.