Understanding and Analysis of LTE Schedulers algorithms in NS-3

Arjun V Anand, CS13B1041 and Akilesh B, CS13B1042

Changes made in the code for creating the scenario described in the assignment.

Allocating the position of eNBs

```
 Ptr<ListPositionAllocator> enbpositionAlloc =
CreateObject<ListPositionAllocator> ();
  enbpositionAlloc->Add (Vector(0, 0, 0));

  //Placing eNB2
  enbpositionAlloc->Add (Vector(1000, 0, 0));

  //Placing eNB3
  enbpositionAlloc->Add (Vector(1000, 1000, 0));

  //Placing eNB4
  enbpositionAlloc->Add (Vector(0,1000, 0));
```

Install Mobility model for eNB

```
  MobilityHelper mobility;
  mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
  mobility.SetPositionAllocator(enbpositionAlloc);
  mobility.Install(enbNodes);
```

Set Random Walk Mobility model for UEs

```
mobility.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
                           "Mode", StringValue ("Time"),
                           "Time", StringValue ("2s"),
                           "Speed", StringValue (spd),
                           "Bounds", StringValue
("-500|1500|-500|1500"));
```

Random disk placement of UEs within 500 m radius of eNB

```
mobility.SetPositionAllocator ("ns3::RandomDiscPositionAllocator",
                               "X", StringValue ("0.0"),
```

```
                                    "Y", StringValue ("0.0"),
                                    "Rho", StringValue
("ns3::UniformRandomVariable[Min=0|Max=500]"));

mobility.Install(ueNodes[0]);


mobility.SetPositionAllocator ("ns3::RandomDiscPositionAllocator",
                                    "X", StringValue ("1000.0"),
                                    "Y", StringValue ("0.0"),
                                    "Rho", StringValue
("ns3::UniformRandomVariable[Min=0|Max=500]"));

mobility.Install(ueNodes[1]);

mobility.SetPositionAllocator ("ns3::RandomDiscPositionAllocator",
                                    "X", StringValue ("1000.0"),
                                    "Y", StringValue ("1000.0"),
                                    "Rho", StringValue
("ns3::UniformRandomVariable[Min=0|Max=500]"));

mobility.Install(ueNodes[2]);

mobility.SetPositionAllocator ("ns3::RandomDiscPositionAllocator",
                                    "X", StringValue ("0.0"),
                                    "Y", StringValue ("1000.0"),
                                    "Rho", StringValue
("ns3::UniformRandomVariable[Min=0|Max=500]"));

mobility.Install(ueNodes[3]);
```

Set number of RBs as 50 in UL and DL

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

Install LTE Devices to the nodes

```
NetDeviceContainer ueLteDevs[4];
for(uint16_t j=0;j<4;j++)
   ueLteDevs[j] = lteHelper->InstallUeDevice (ueNodes[j]);

 Ipv4InterfaceContainer ueIpIface[4];
```

Install the IP stack on the UEs

```
  for(uint16_t j=0;j<4;j++){
    internet.Install (ueNodes[j]);

ueIpIface[j] = epcHelper->AssignUeIpv4Address (NetDeviceContainer
(ueLteDevs[j]));
```

Assign IP address to UEs, and install applications

```
    for (uint32_t u = 0; u < ueNodes[j].GetN (); ++u)
      {
        Ptr<Node> ueNode = ueNodes[j].Get (u);
        //Set the default gateway for the UE
        Ptr<Ipv4StaticRouting> ueStaticRouting =
ipv4RoutingHelper.GetStaticRouting (ueNode->GetObject<Ipv4> ());
        ueStaticRouting->SetDefaultRoute
(epcHelper->GetUeDefaultGatewayAddress (), 1);
      }
```

Attach all UEs to eNodeB

```
    for (uint16_t i = 0; i < numberOfUEs; i++)
        {
          lteHelper->Attach (ueLteDevs[j].Get(i), enbLteDevs.Get(j));
          // side effect: the default EPS bearer will be activated
        }
    }
```

Graph 1: SINR Radio Environment Map (REM)

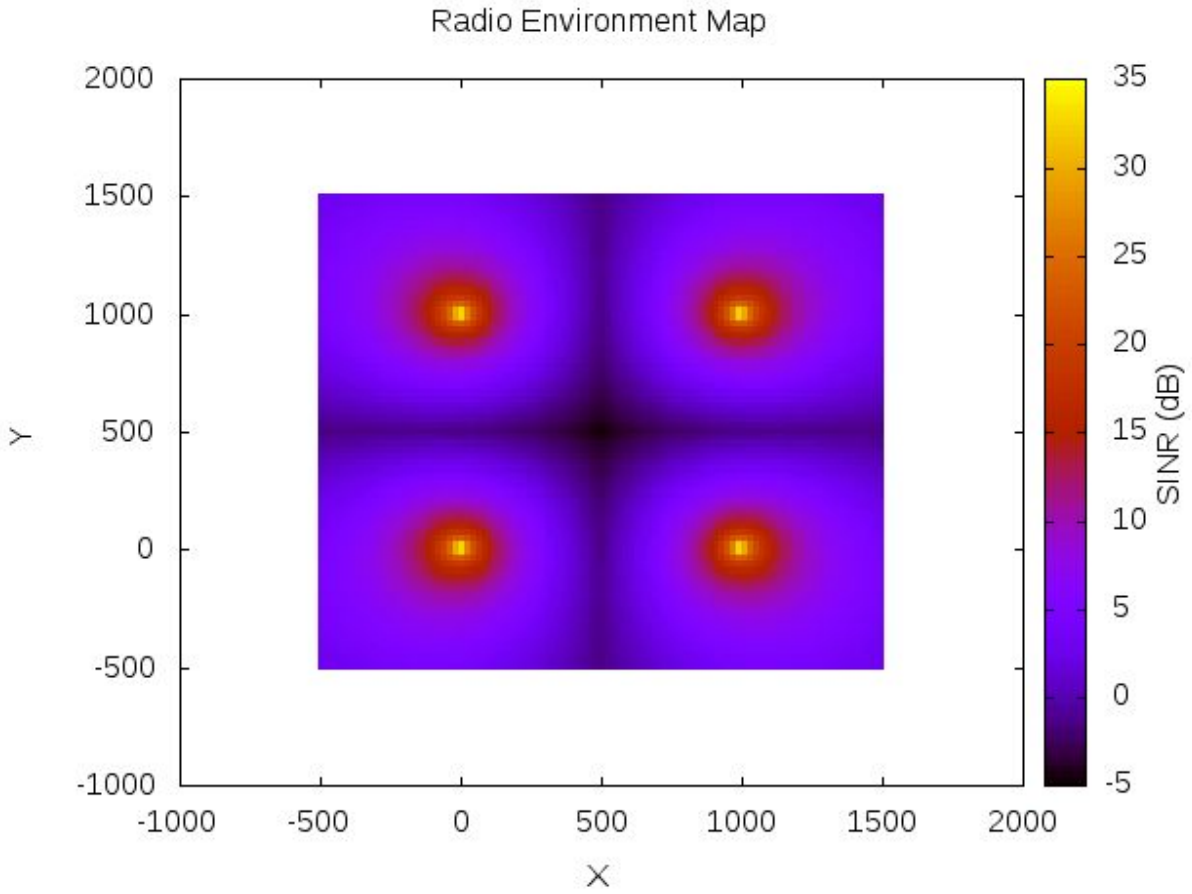
A REM is a uniform grid of 2D values that represent the signal to noise ratio in the downlink with respect to the eNB that has the strongest signal at each point. REM can be generated for either data channel or control channel.

Necessary changes in code are:

```
Ptr<RadioEnvironmentMapHelper> remHelper =
CreateObject<RadioEnvironmentMapHelper> ();
remHelper->SetAttribute ("ChannelPath", StringValue
("/ChannelList/1"));
remHelper->SetAttribute ("OutputFile", StringValue ("rem.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 ();
```

These lines are added before calling Simulator::Run()



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

X min and Y min are -500, because the UE distribution is a random walk of radius 500 around the eNB. X max and Y max are 1500.

Stats collection for graph 2,3,6:

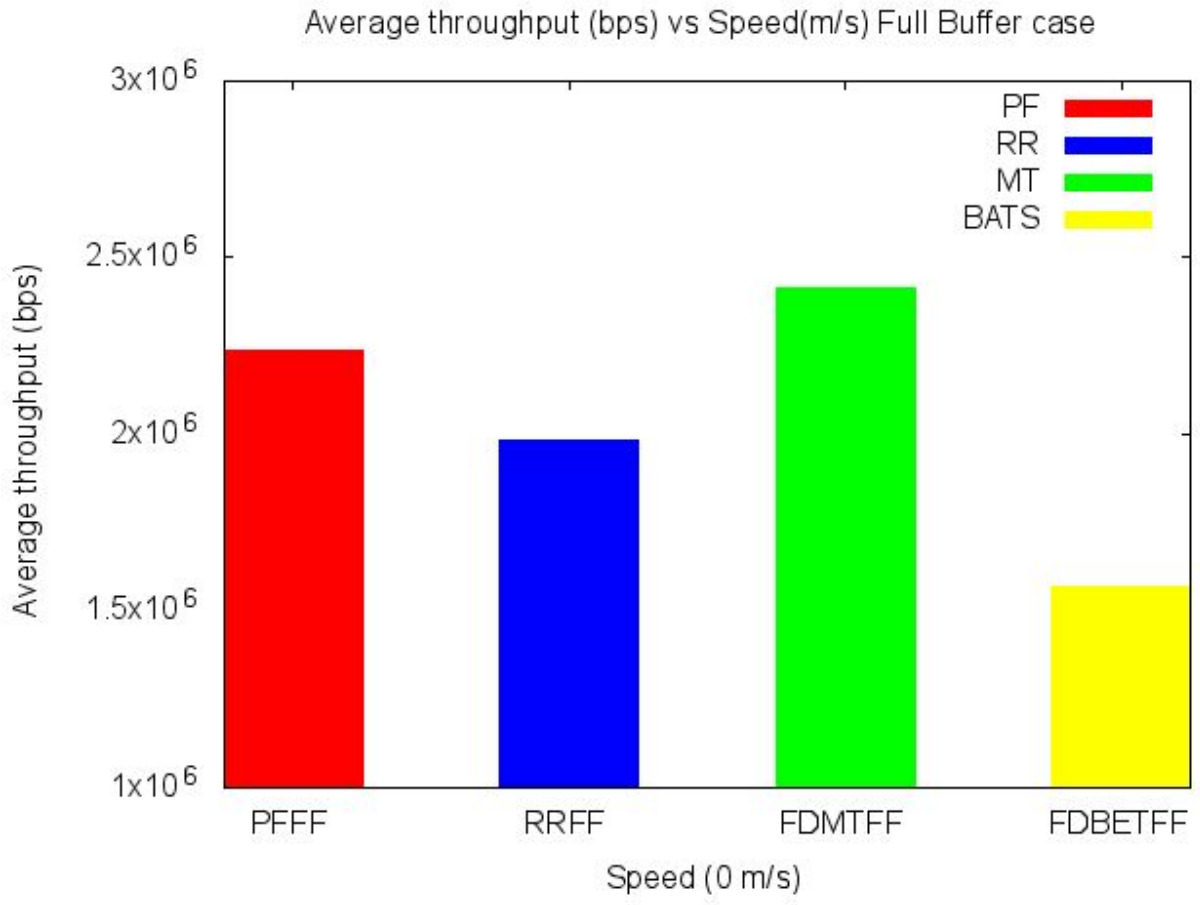
We wrote the following script which automatically performs the simulation and redirects the results in appropriate folders with proper convention.
It considers four schedulers (PF, RR, MT, BATS) as given in the question, two different speeds (0,5) ms, full buffer/ half buffer case and it runs all the simulations five times for seed values(RngRun) between 42 and 46.

```
import os
import sys

schedulers =
["ns3::PfFfMacScheduler","ns3::RrFfMacScheduler","ns3::FdMtFfMacSched
uler","ns3::FdBetFfMacScheduler"];
mapping = {'ns3::PfFfMacScheduler': 'pfff', 'ns3::RrFfMacScheduler':
'rrff', 'ns3::FdMtFfMacScheduler':
'fdmtff','ns3::FdBetFfMacScheduler':'fdbetff'};
speed =[0,5];
ipi = [1,10];
map2={'1':'f','10':'h'};
rngrun=[42,43,44,45,46];

for s in schedulers:
     for vel in speed:
          for buf in ipi:
               for rng in rngrun:

outfile='testoutput/'+mapping[s]+'_'+map2[str(buf)]+'_'+str(vel)+'_'+
str(rng)
os.system('./waf --run \"ass1_cmd --speed='+str(vel)+ '
--interPacketInterval='+str(buf)+' --schedType='+s+'
--RngRun='+str(rng)+'\"'+' >'+ outfile+'.txt')
```

Note: The above code can be found in the file run_tests.py

Graph 2: Average Aggregate System throughput vs Speed for all four schedulers for full buffer scenario

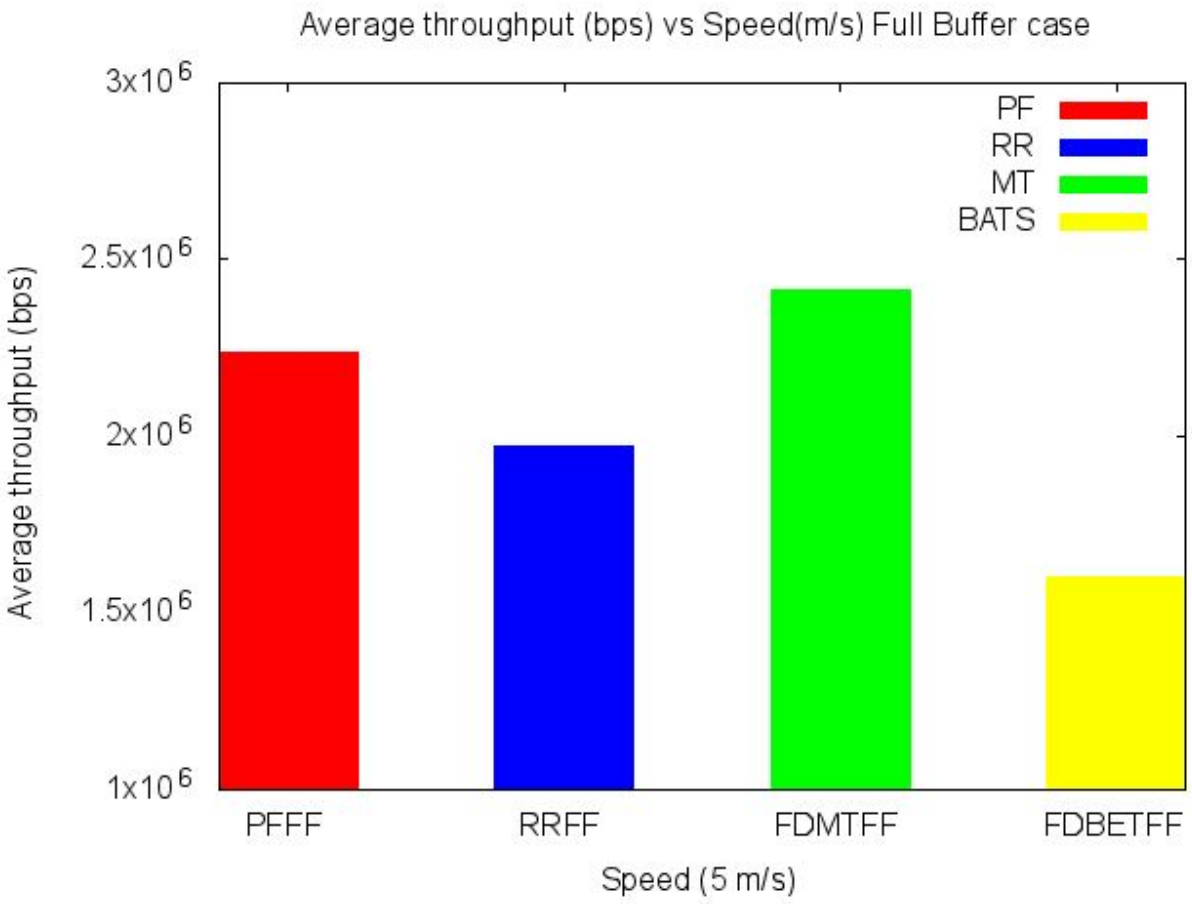
Case 1: Speed = 0m/s



Observations:

- Max throughput scheduler performs the best.
- BATS gives the worst performance.
- Throughput order: MT > PF > RR > BATS

Case 2: Speed = 5m/s

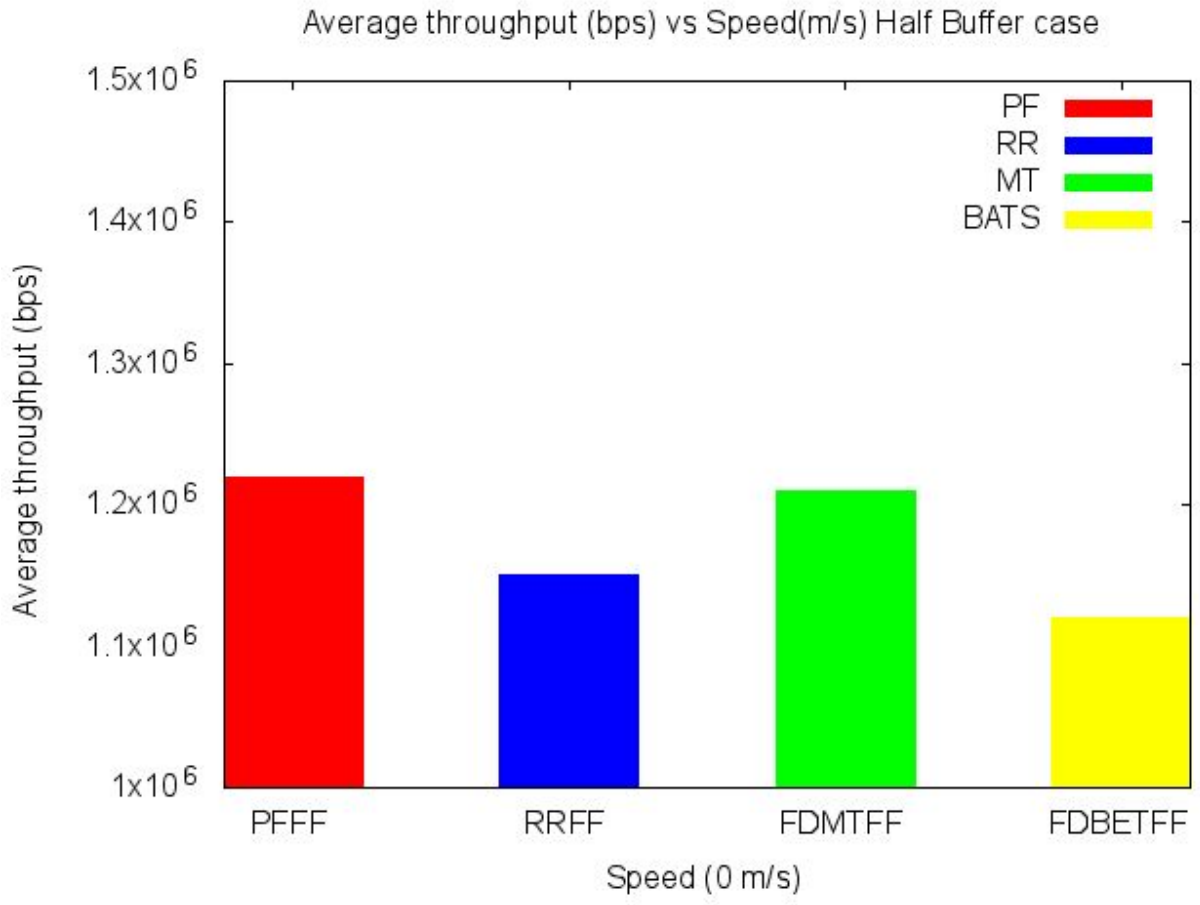


Observations:

- Max throughput scheduler performs the best.
- BATS gives the worst performance.
- Throughput order: MT > PF > RR > BATS
- Speed of UE doesn't affect the throughput.

Graph 6: Average Aggregate System throughput vs Speed for all four schedulers for half buffer scenario

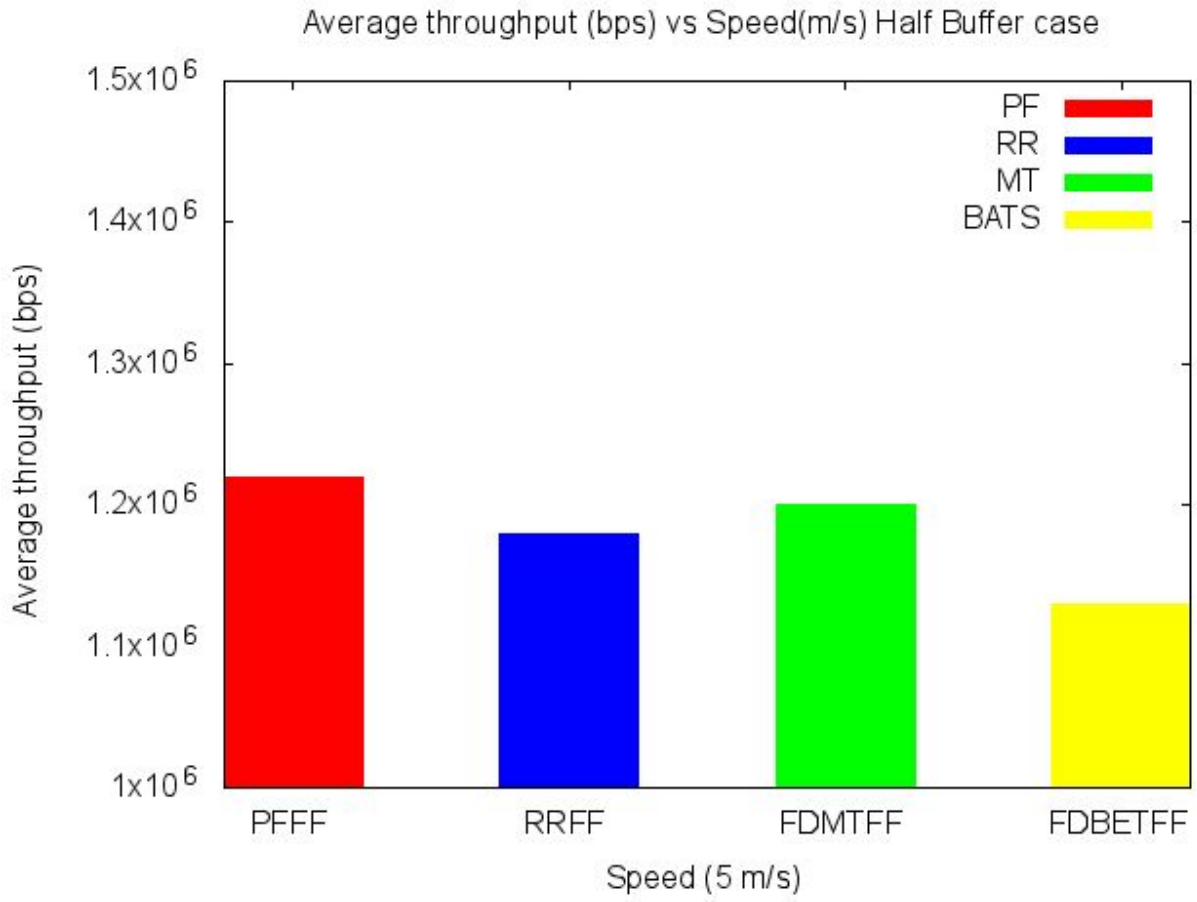
Case 1: Speed = 0m/s



Observations:

- PF scheduler performs the best.
- BATS gives the worst performance.
- Throughput order: PF > MT > RR > BATS

Case 2: Speed = 5m/s



Observations:

- PF scheduler performs the best.
- BATS gives the worst performance.
- Throughput order: PF > MT > RR > BATS
- Speed of UE doesn't affect the throughput.

Analysis:

1. Round Robin (RR):

The scheduler provides resources cyclically to the users without considering channel conditions into account. It's a simple procedure giving the best fairness. But it would propose poor performance in terms of cell throughput. RR meets the fairness by providing an equal share of packet transmission time to each user. In Round Robin (RR) scheduling the terminals are assigned the resource blocks in turn (one after another) without considering CQI. Thus the terminals are equally scheduled. However, throughput performance degrades significantly as the algorithm does not rely on the reported instantaneous downlink SNR values when determining the number of bits to be transmitted. That's why throughput is not the best in this case which can be observed from the graphs too.

2. Proportional fair (PF):

Main purpose of Proportional Fair algorithm is to balance between throughput and fairness among all the UEs. It tries to maximize total [wired/wireless network] throughput while at the same time it provides all users at least a minimal level of service. The scheduler can affect Proportional Fair (PF) scheduling by allocating more resources to a user, comparatively with better channel quality. This is done by giving each data flow a scheduling priority that is inversely proportional to its anticipated resource consumption. This gives high cell throughput as well as fairness satisfactorily. Thus, Proportional Fair (PF) scheduling gives the best results most of the time and that can also be seen from the graphs.

3. Max Throughput(Best CQI):

This scheduling algorithm is used for strategy to assign resource blocks to the user with the best radio link conditions. The resource blocks assigned by the Best CQI to the user will have the highest CQI on that RB. The MS must feedback the Channel Quality Indication (CQI) to the BS to perform the Best CQI. In order to perform scheduling, terminals send Channel Quality Indicator (CQI) to the base station (BS). Basically in the downlink, the BS transmits reference signal (downlink pilot) to terminals. These reference signals are used by UEs for the calculation of the CQI. A higher CQI value means better channel condition. We can see from the graphs that this performs as good as PF algorithm and even betters it in the full buffer case. But generally proportional fair beats this in most of the cases.

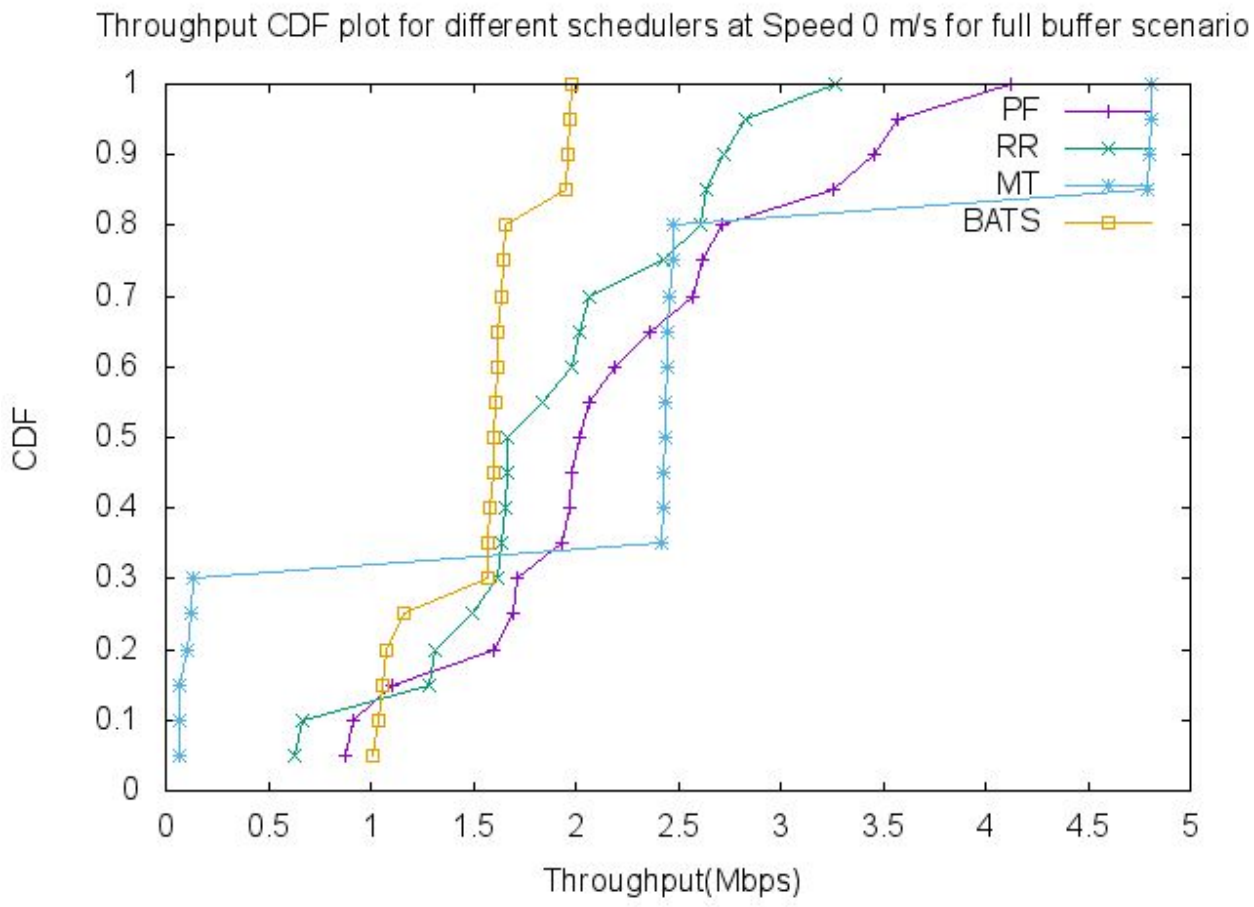
4. BATS:

The Blind Average Throughput scheduler aims to provide equal throughput to all UEs under eNB. In the time domain blind average throughput (TD-BET), the scheduler selects the UE with largest priority metric and allocates all RBGs to this UE. On the other hand, in the frequency domain blind average throughput (FD-BET), every TTI, the scheduler first selects one UE with

lowest pastAverageThroughput (largest priority metric). Then scheduler assigns one RBG to this UE, it calculates expected throughput of this UE and uses it to compare with past average throughput T_{j}(t) of other UEs. The scheduler continues to allocate RBG to this UE until its expected throughput is not the smallest one among past average throughput T_{j}(t) of all UE. Then the scheduler will use the same way to allocate RBG for a new UE which has the lowest past average throughput T_{j}(t) until all RBGs are allocated to UEs. The principle behind this is that, in every TTI, the scheduler tries the best to achieve the equal throughput among all UEs. Because of the fact that it tries to give equal throughput to all UE's under an eNB, we can see that this algorithm gives the least performance among all the given algorithms in all the cases.

Graph 3: Throughput CDF plot for different schedulers at Speed (0,5) m/s for full buffer scenario

Case 1: Speed = 0m/s

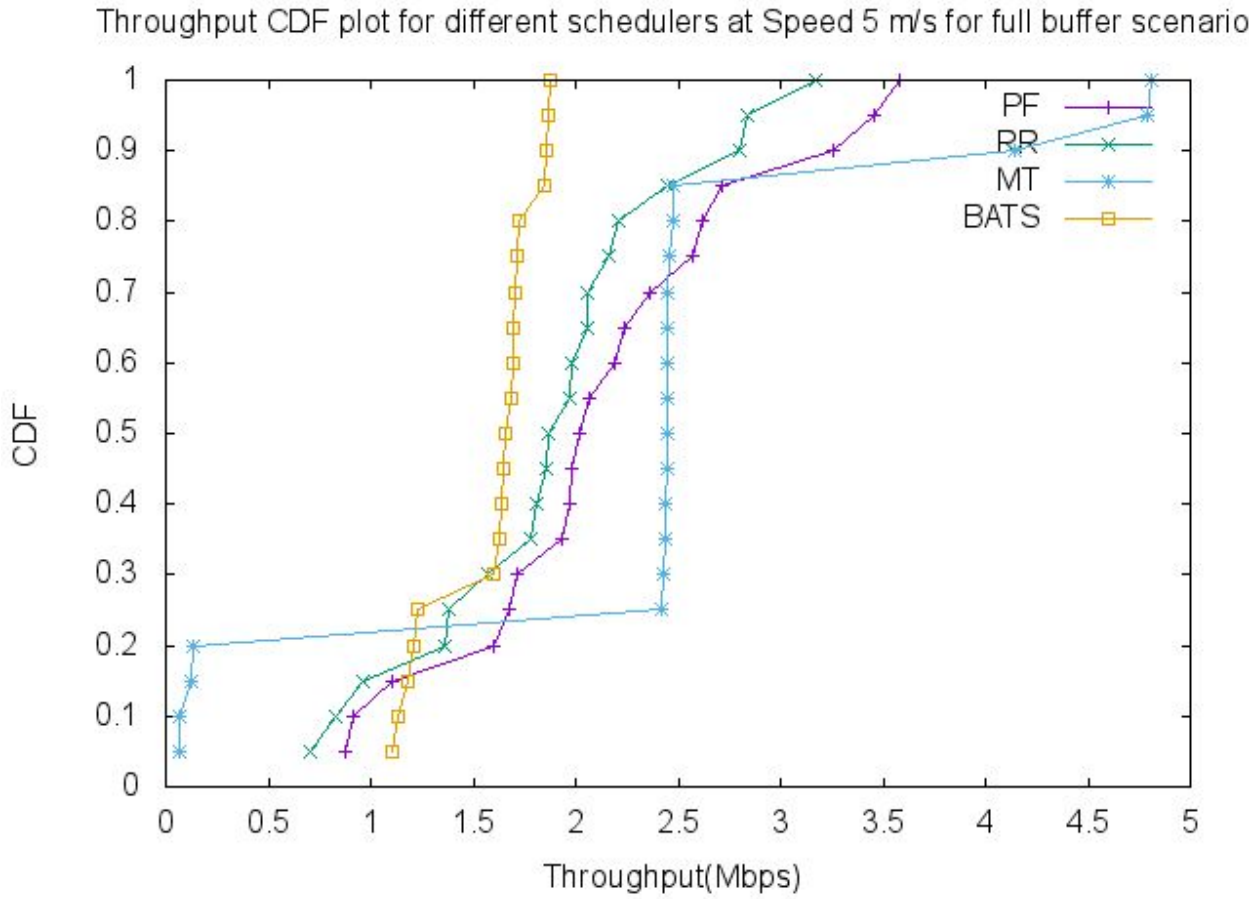


For the case of full buffer and speed=0 m/s, we calculated per UE average over 5 runs for all the UEs. Then we sorted the all the 20 values in the increasing order and plotted those values on the x axis and the corresponding CDF value on the y-axis.

Observations:

- Curve corresponding to BATS algorithm reaches CDF of 1 for very small throughput indicating that the max throughput is least in this case.
- Curve corresponding to MT algorithm reaches CDF of 1 for very large throughput indicating that the max throughput is highest in this case.
- Max. throughput order : MT > PF > RR > BATS
- Max. throughput and average throughput trends are same for the full buffer case.

Case 2: Speed = 5m/s



Observations:

- Curve corresponding to BATS algorithm reaches CDF of 1 for very small throughput indicating that the max throughput is least in this case.

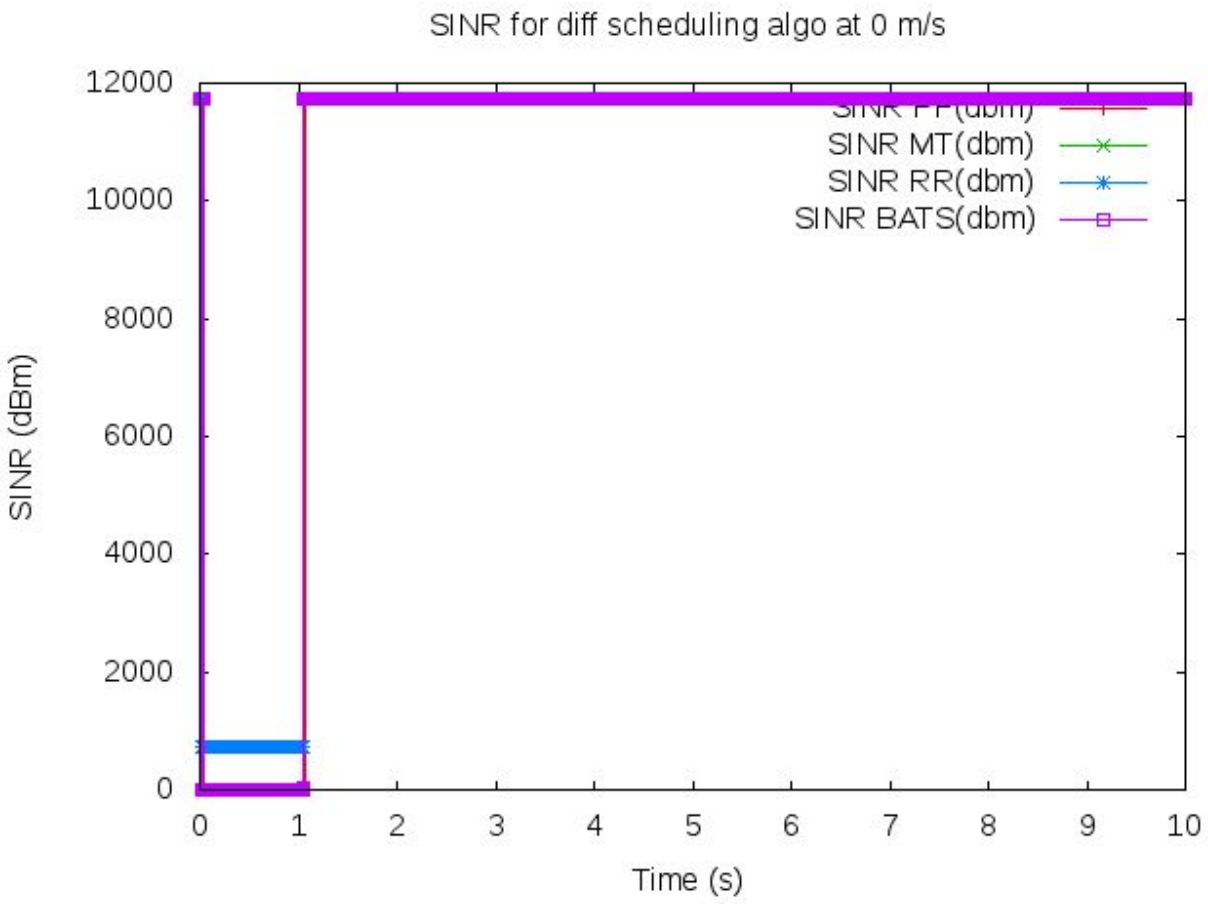
- Curve corresponding to MT algorithm reaches CDF of 1 for very large throughput indicating that the max throughput is highest in this case.
- Max. throughput order : MT > PF > RR > BATS
- Max. throughput and average throughput trends are same for the full buffer case.
- Speed doesn't affect throughput because the graphs are almost similar for both the speeds.

Graph 4: SINR / Instantaneous throughput for UE 0 in the simulation for seed with RngRun 42 at 0 m/s

The simulation was run for 10 s with different schedulers under full buffer mode with RngRun = 42 and speed = 0 m/s. 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.

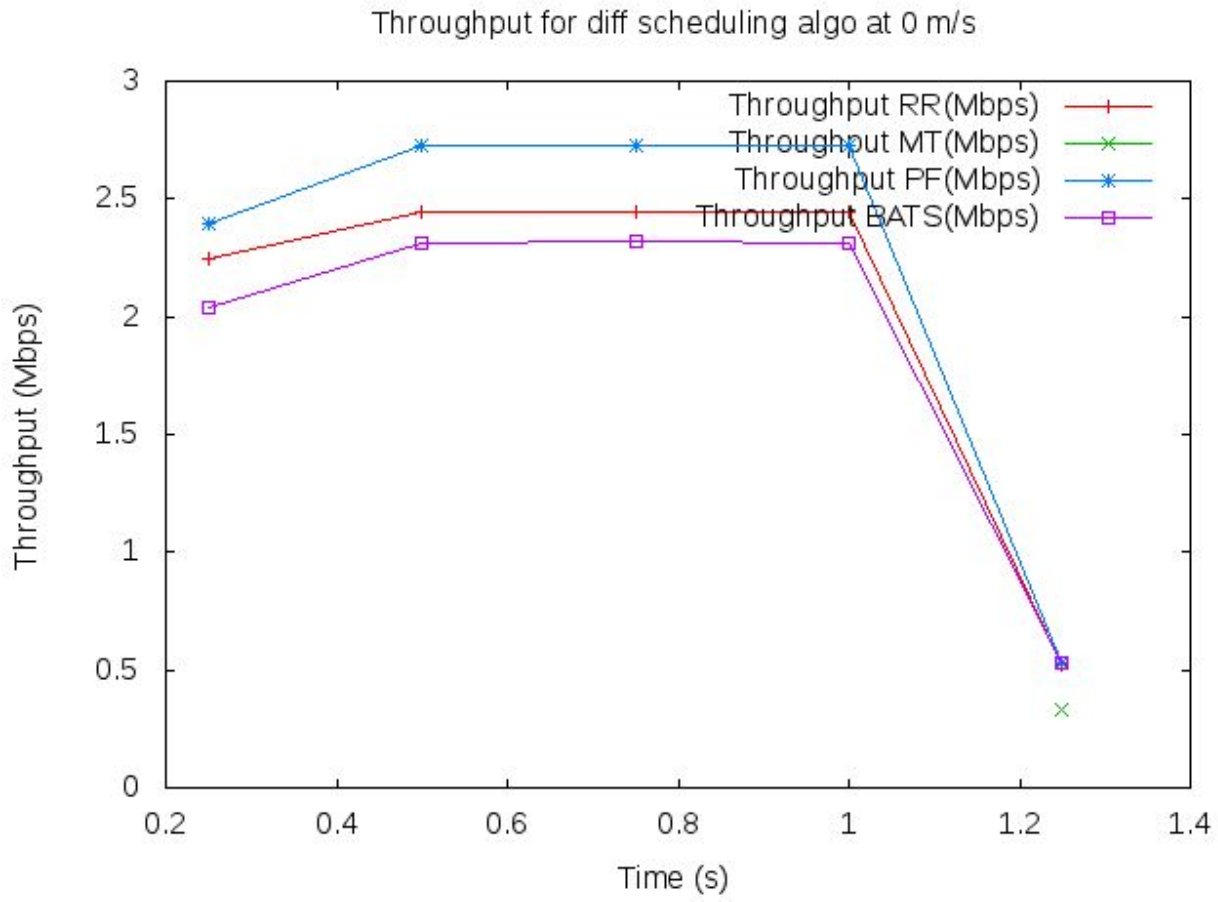
awk '$3==1' DlRsrpSinrStats.txt > sinr.txt (For getting SINR stats where IMEI = 1 ie for UE 0).
awk '$4==1' DlRlcStats.txt > filter_throughput.txt (For getting Throughput stats where IMEI = 1 ie for UE 0).
awk '{ print $2, $10 * 0.000032}' filter_throughput.txt > throughput.txt

SINR plot:



Proportional Fair (PF) scheduler has the highest SINR value of 11720 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. Round Robin has relatively less SINR around 700 dbm initially. BAT scheduler has a very low SINR around 8 dbm for most time between 0 and 1s.

Throughput plot:



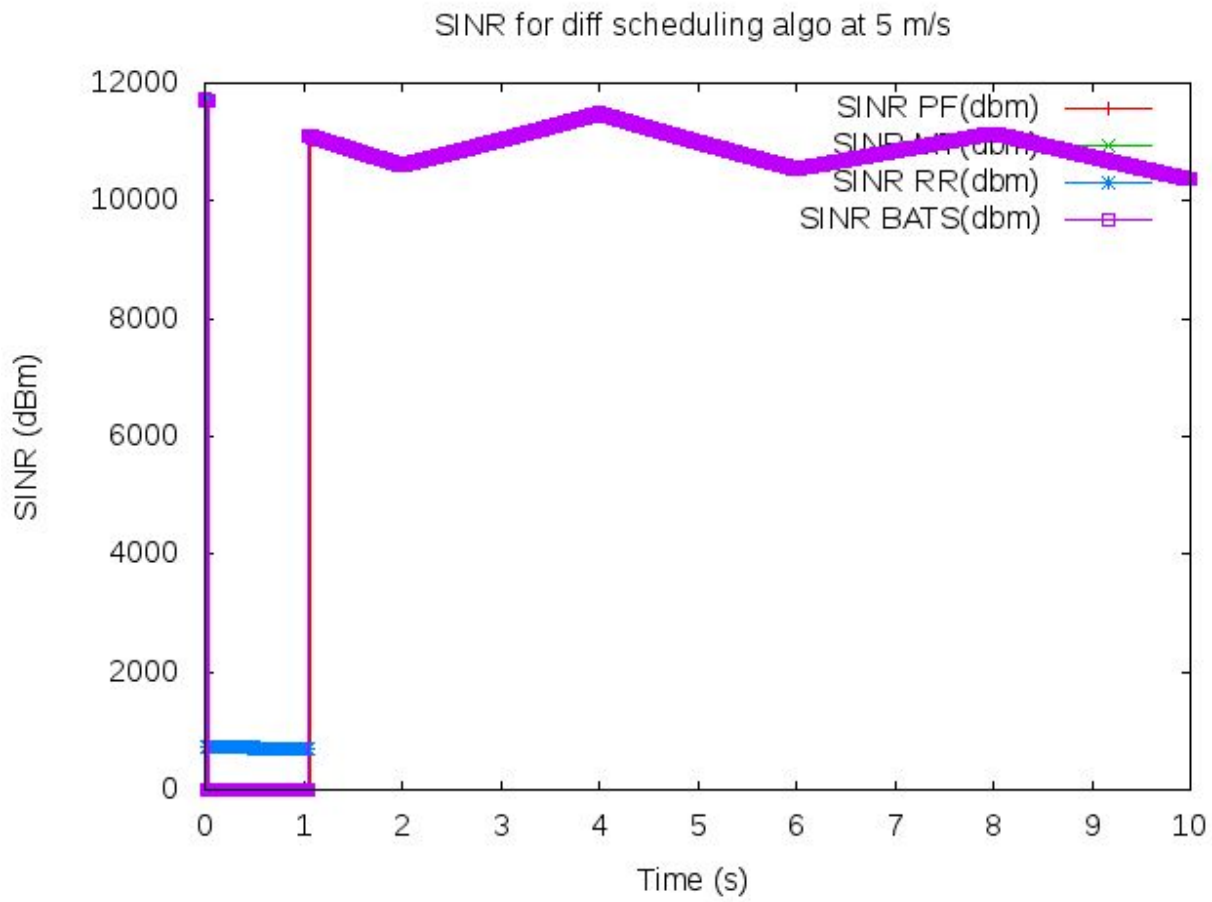
Throughput for Proportional Fair (PF) scheduler was highest followed by Round Robin (RR) scheduler and then Blind Average throughput (BAT) scheduler. All the schedulers behave in a similar manner, the throughput first increases, remains constant for a while and then decreases. At t = 1.25 s, the value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

Since MT scheduler tries to maximize the overall throughput of eNB, it never allocates RB to UE 0. It always assigns RB in each TTI to the UE which can achieve maximum rate in the current TTI. Therefore, the throughput of the MT scheduler is 0. MT scheduler never assigns RBs to the UEs under a given eNB whose channel quality is poor in comparison to the other UEs under the same eNB.

The throughput of the BAT scheduler is least because it aims to provide equal throughput to all the UEs under the eNB irrespective of its channel conditions.
PF scheduler also assigns RBs to the UE based on channel conditions, ie it assigns more RBs to the UE with high channel quality and less RBs to the UE with poor channel quality. Therefore, its throughput is higher than BATS.

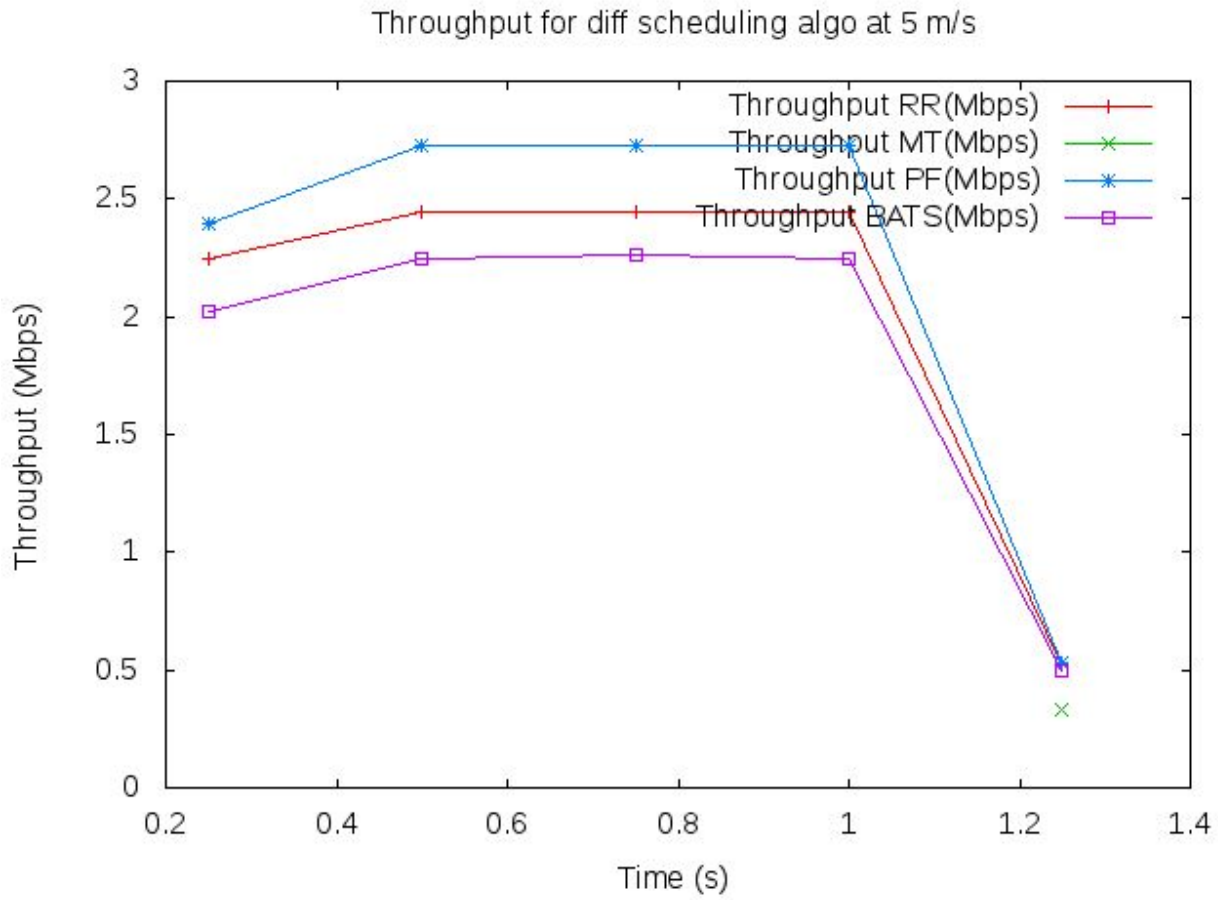
Graph 5: SINR / Instantaneous throughput for UE 0 in the simulation for seed with RngRun 42 at 5 m/s

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 11720 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. In this case, the value of SINR varies a bit with time as shown in graph. Round Robin has relatively less SINR around 700 dbm initially. BAT scheduler has a very low SINR around 8 dbm for most time between 0 and 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 Round Robin (RR) scheduler and then Blind Average throughput (BAT) 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.

Since MT scheduler tries to maximize the overall throughput of eNB, it never allocates RB to UE 0. It always assigns RB in each TTI to the UE which can achieve maximum rate in the current TTI. Therefore, the throughput of the MT scheduler is 0. MT scheduler never assigns RBs to the UEs under a given eNB whose channel quality is poor in comparison to the other UEs under the same eNB.

The throughput of the BATS scheduler is least because it aims to provide equal throughput to all the UEs under the eNB irrespective of its channel conditions.

PF scheduler also assigns RBs to the UE based on channel conditions, ie it assigns more RBs to the UE with high channel quality and less RBs to the UE with poor channel quality. Therefore, its throughput is higher than BATS.