Understanding and Analysis of LTE Handover 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));
  enbpositionAlloc->Add (Vector(250, 0, 0));
  enbpositionAlloc->Add (Vector(500, 0, 0));
  enbpositionAlloc->Add (Vector(750, 0, 0));
  enbpositionAlloc->Add (Vector(750, 250, 0));
  enbpositionAlloc->Add (Vector(500, 250, 0));
  enbpositionAlloc->Add (Vector(250, 250, 0));
  enbpositionAlloc->Add (Vector(0,250, 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
("-2000|2000|-2000|2000"));
```

Random disk placement of UEs within 100m to 125 m radius of eNB

```
mobility.SetPositionAllocator ("ns3::RandomDiscPositionAllocator",
                               "X", StringValue ("0.0"),
                               "Y", StringValue ("0.0"),
                               "Rho", StringValue
("ns3::UniformRandomVariable[Min=100|Max=125]"));

mobility.Install(ueNodes[0]);


mobility.SetPositionAllocator ("ns3::RandomDiscPositionAllocator",
                               "X", StringValue ("250.0"),
                               "Y", StringValue ("0.0"),
                               "Rho", StringValue
("ns3::UniformRandomVariable[Min=100|Max=125]"));

mobility.Install(ueNodes[1]);

mobility.SetPositionAllocator ("ns3::RandomDiscPositionAllocator",
                               "X", StringValue ("500.0"),
                               "Y", StringValue ("0.0"),
                               "Rho", StringValue
("ns3::UniformRandomVariable[Min=100|Max=125]"));

mobility.Install(ueNodes[2]);

mobility.SetPositionAllocator ("ns3::RandomDiscPositionAllocator",
                               "X", StringValue ("750.0"),
                               "Y", StringValue ("0.0"),
                               "Rho", StringValue
("ns3::UniformRandomVariable[Min=100|Max=125]"));

mobility.Install(ueNodes[3]);

mobility.SetPositionAllocator ("ns3::RandomDiscPositionAllocator",
                               "X", StringValue ("750.0"),
                               "Y", StringValue ("250.0"),
                               "Rho", StringValue
("ns3::UniformRandomVariable[Min=100|Max=125]"));

mobility.Install(ueNodes[4]);
```

```
mobility.SetPositionAllocator ("ns3::RandomDiscPositionAllocator",
                               "X", StringValue ("500.0"),
                               "Y", StringValue ("250.0"),
                               "Rho", StringValue
("ns3::UniformRandomVariable[Min=100|Max=125]"));

mobility.Install(ueNodes[5]);

mobility.SetPositionAllocator ("ns3::RandomDiscPositionAllocator",
                               "X", StringValue ("250.0"),
                               "Y", StringValue ("250.0"),
                               "Rho", StringValue
("ns3::UniformRandomVariable[Min=100|Max=125]"));

mobility.Install(ueNodes[6]);

mobility.SetPositionAllocator ("ns3::RandomDiscPositionAllocator",
                               "X", StringValue ("0.0"),
                               "Y", StringValue ("250.0"),
                               "Rho", StringValue
("ns3::UniformRandomVariable[Min=100|Max=125]"));

mobility.Install(ueNodes[7]);
```

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[8];
for(uint16_t j=0;j<8;j++)
   ueLteDevs[j] = lteHelper->InstallUeDevice (ueNodes[j]);

 Ipv4InterfaceContainer ueIpIface[8];
```

Install the IP stack on the UEs

```
  for(uint16_t j=0;j<8;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)

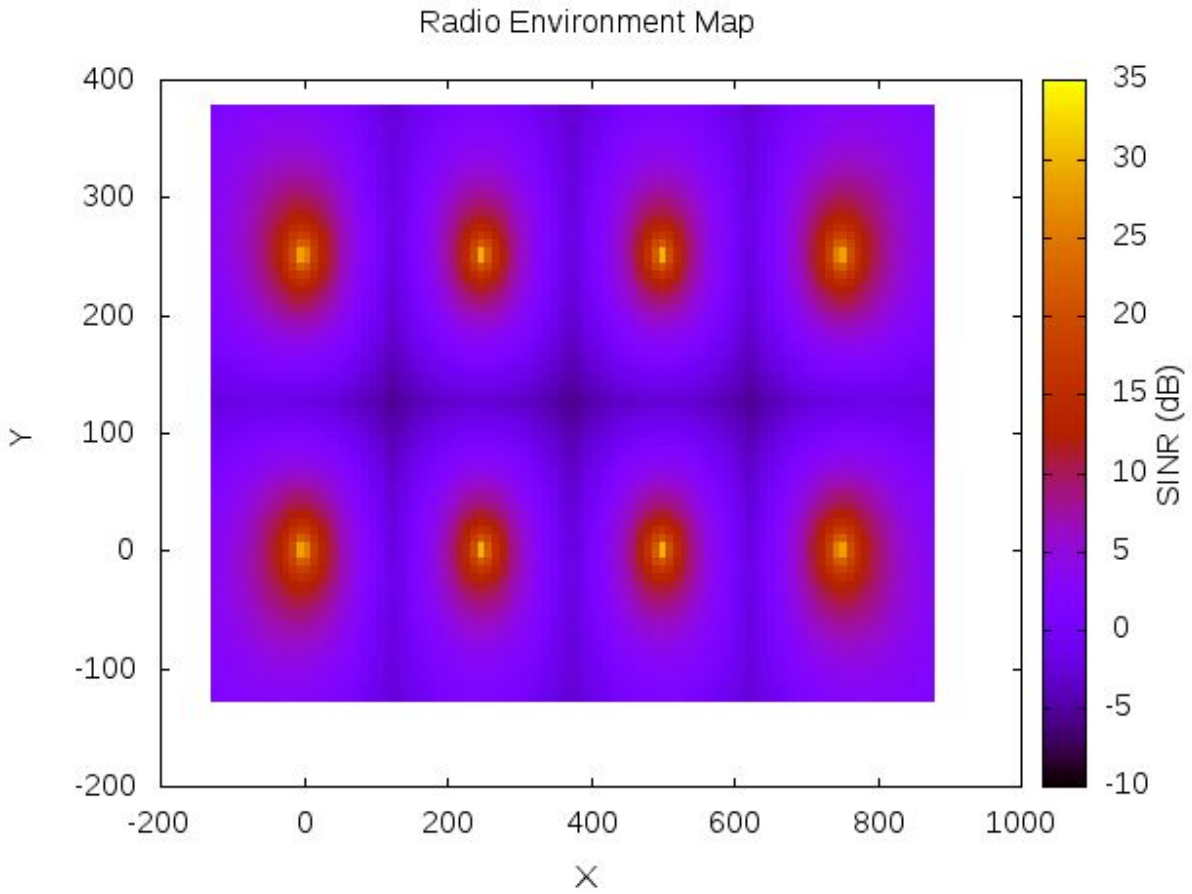
A REM is a uniform grid of 2D values that represent the signal to noise ratio in the downlink with respect to the eNB that has the strongest signal at each point. REM can be generated for either data channel or control channel.

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 (-125.0));
remHelper->SetAttribute ("XMax", DoubleValue (875.0));
remHelper->SetAttribute ("XRes", UintegerValue (100));
remHelper->SetAttribute ("YMin", DoubleValue (-125.0));
remHelper->SetAttribute ("YMax", DoubleValue (875.0));
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()



Stats collection for graph 2,3:

We fixed the seeds as 42 for graphs 2 and 3. We wrote the following script which automatically performs the simulation and redirects the results in appropriate folders with proper convention.

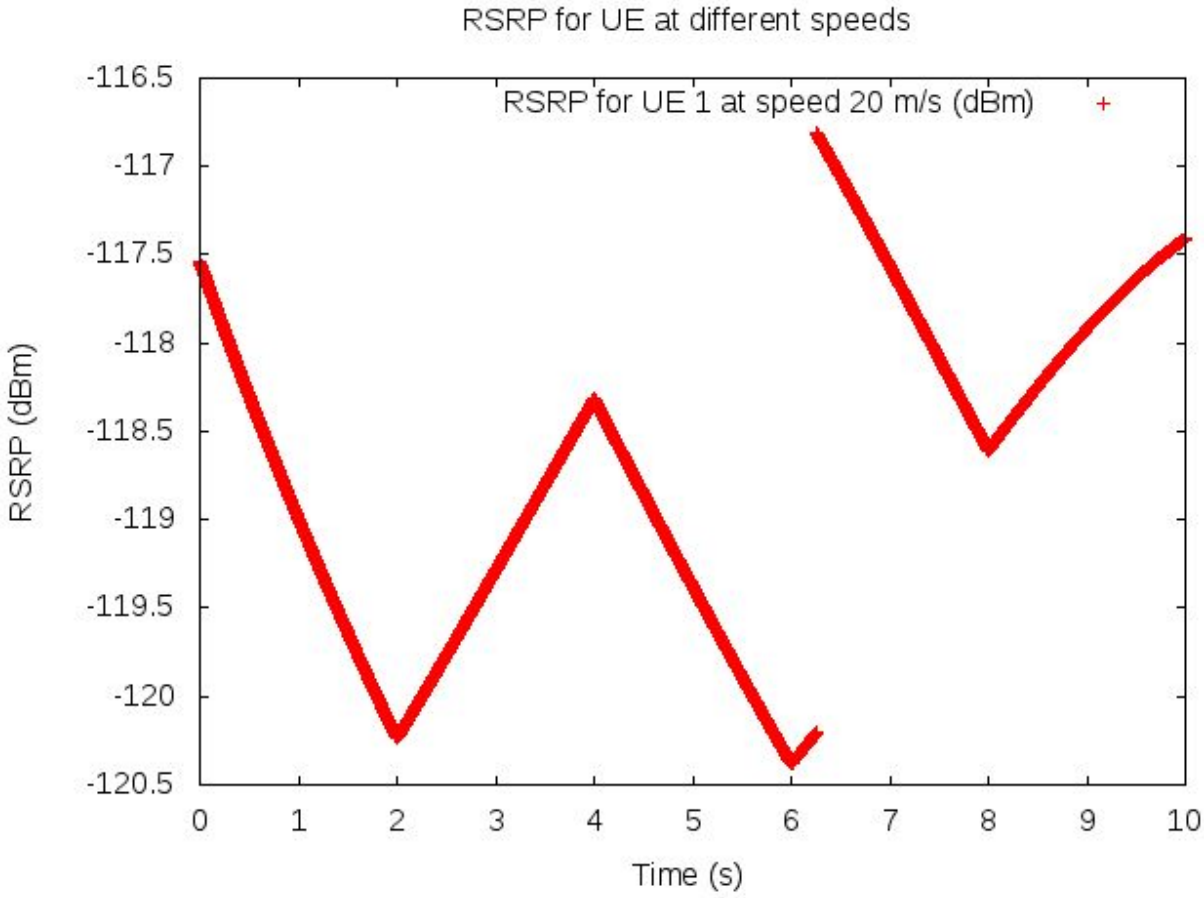
```
import os
import sys
algorithms = ['ns3::A3RsrpHandoverAlgorithm'];
mapping = {'ns3::A3RsrpHandoverAlgorithm': 'sc'};
speed =[20,50,100];
rngrun=[42];

for a in algorithms:
     for vel in speed:
                for rng in rngrun:

outfile='graph2retry/'+mapping[a]+'_'+str(vel)+'_'+str(rng)
                    print './waf --run \"scratch/lena-x2-handover
--speed='+str(vel)+' --handAlgo='+a+' --RngRun='+str(rng)+'\"'+' >'+
outfile+'.txt'
                    os.system('./waf --run
\"scratch/lena-x2-handover --speed='+str(vel)+' --handAlgo='+a+'
--RngRun='+str(rng)+'\"'+' >'+ outfile+'.txt')
                    os.system('cp DlRsrpSinrStats.txt
graph2retry/DlRsrpSinrStats_'+mapping[a]+'_'+str(vel)+'_'+str(rng)+'.
txt')
                    os.system('cp DlPdcpStats.txt
graph2retry/DlPdcpStats_'+mapping[a]+'_'+str(vel)+'_'+str(rng)+'.txt'
)
```

Note: The above code can be found in the file graph2_stats.py

Graph 2: RSRP in dBm for a UE vs speed

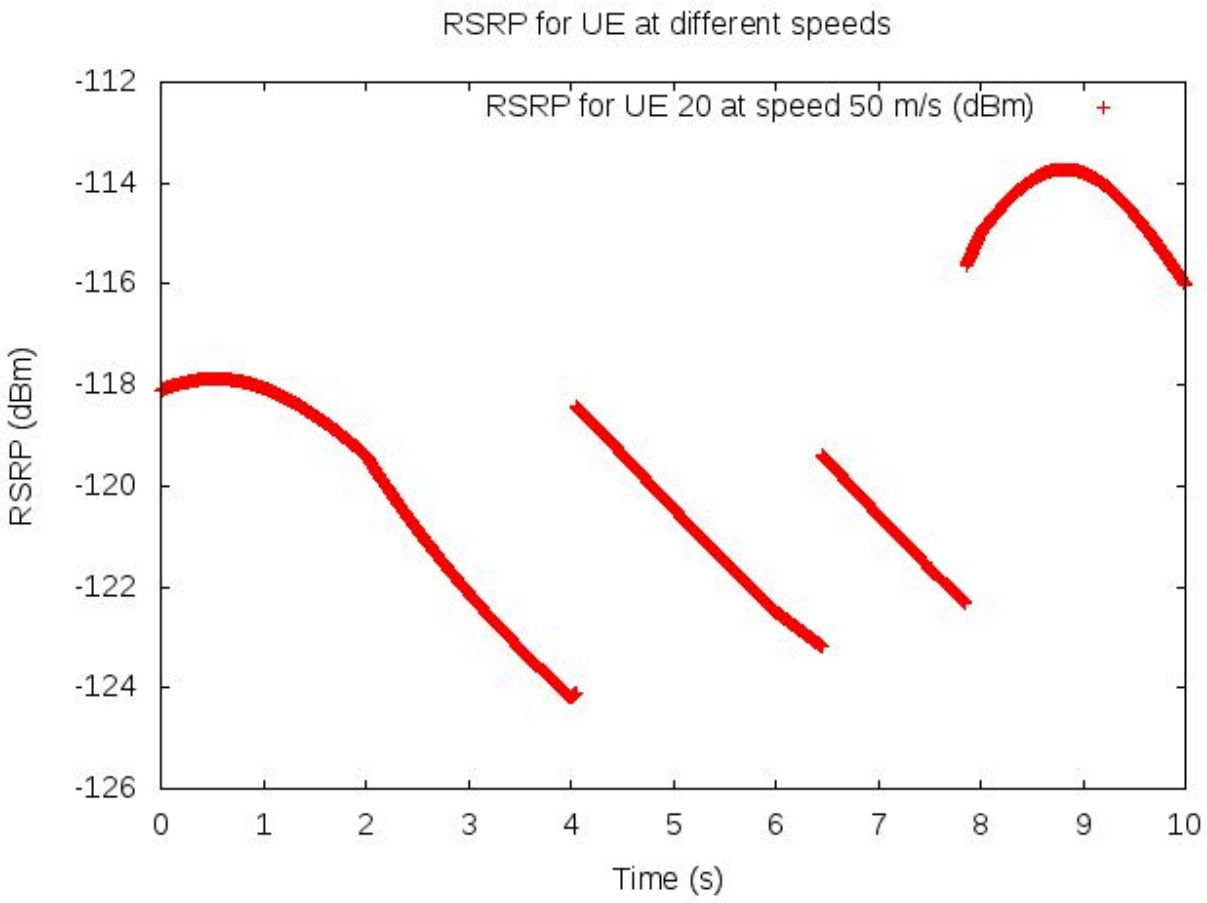
We fixed the seed to be 42 for this purpose (although we collected the stats for 5 seeds (42 to 46)) for speeds 20 m/s, 50 m/s and 100 m/s.

Plot 1



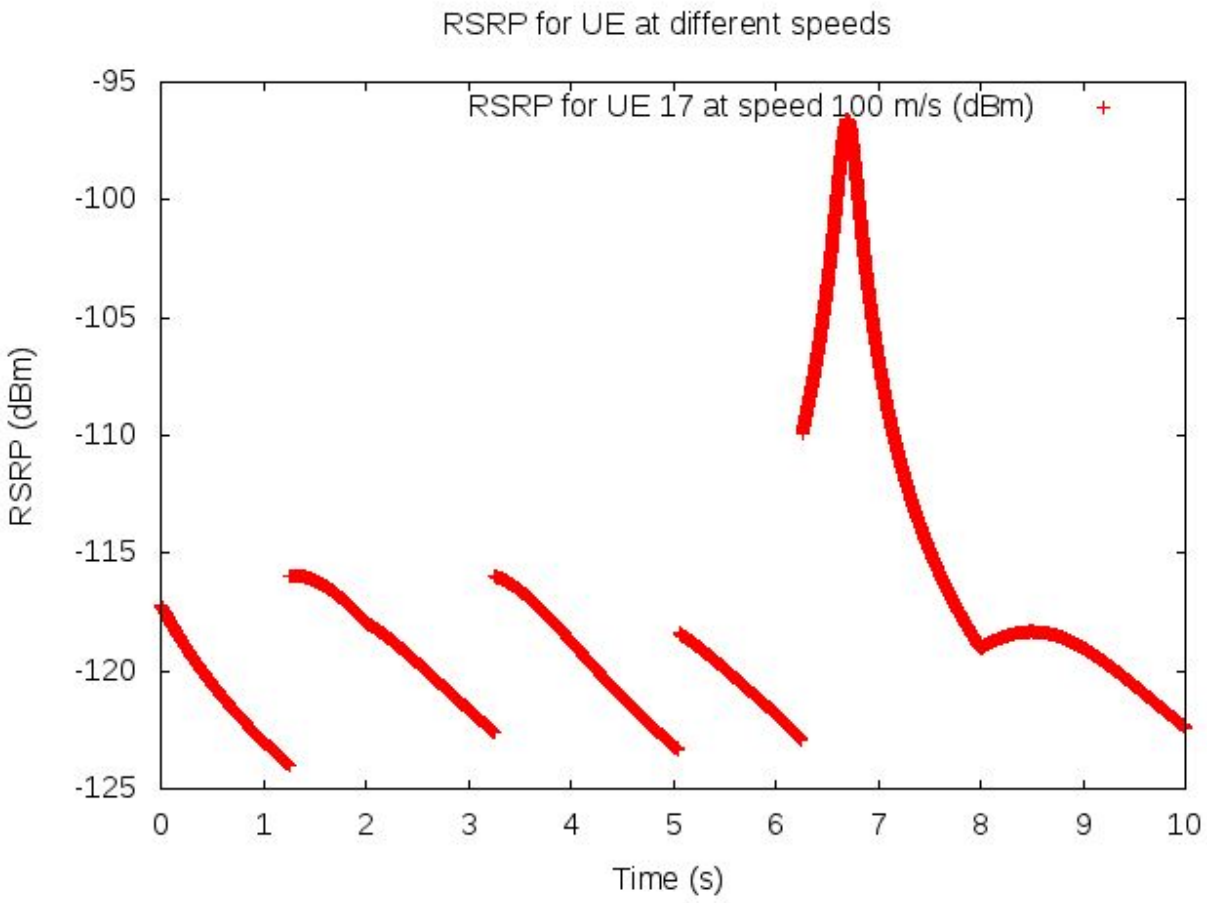
At 20 m/s, we found that UE 1 had maximum number of handovers, the number being 1. This is also evident from the graph. There is one point of discontinuity in graph which indicates that at this instant an handover occurred. Also, we notice that there is a big leap in RSRP value after the handover.

Plot 2:



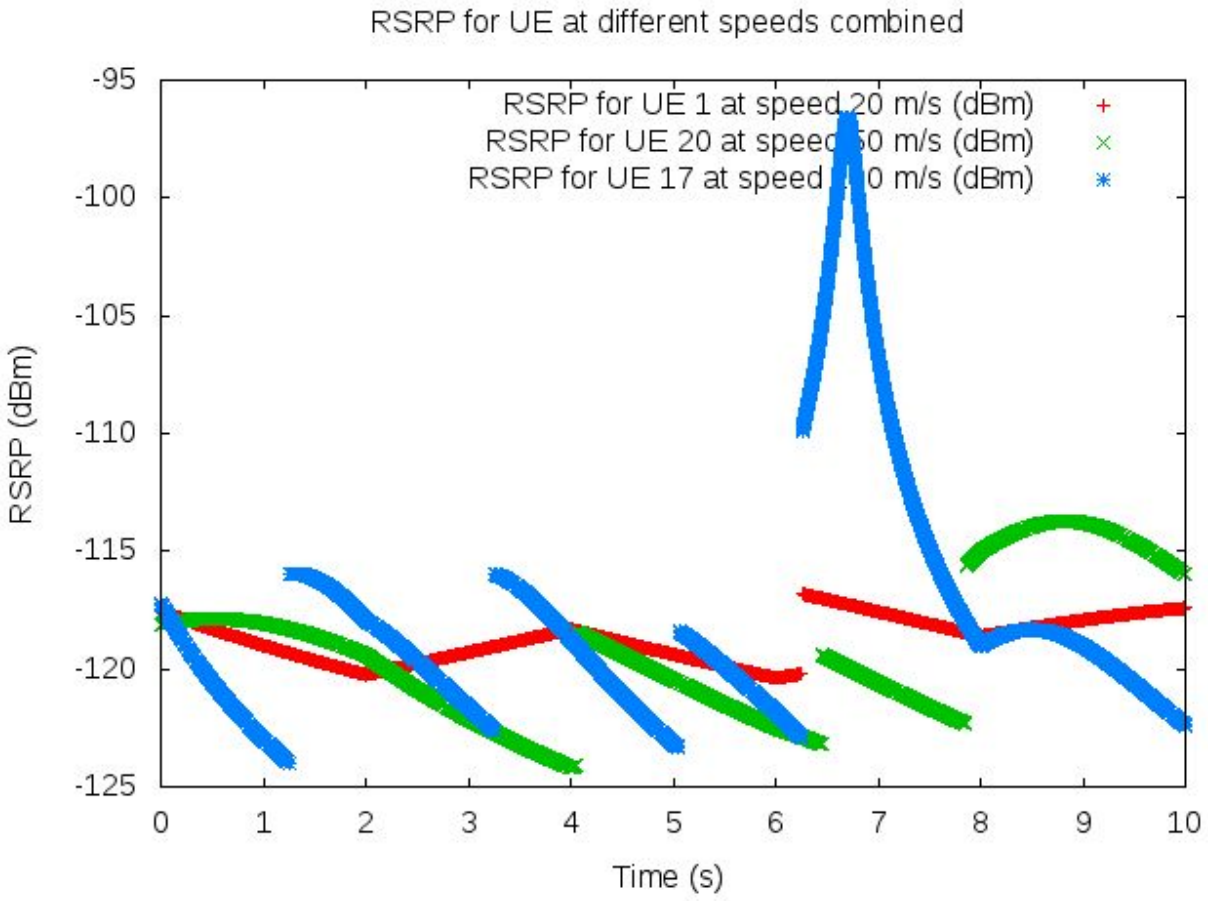
At 50 m/s, we found that UE 20 had maximum number of handovers, the number being 3. This is also evident from the graph. There are three points of discontinuity in graph which indicate that at these instances an handover occurred. Also, we notice that there is a big leap in RSRP value after every handover.

Plot 3



At 100 m/s, we found that UE 17 had maximum number of handovers, the number being 4. This is also evident from the graph. There are four points of discontinuity in graph which indicate that at these instances, an handover occurred. Also, we notice that there is a big leap in RSRP value after every handover.

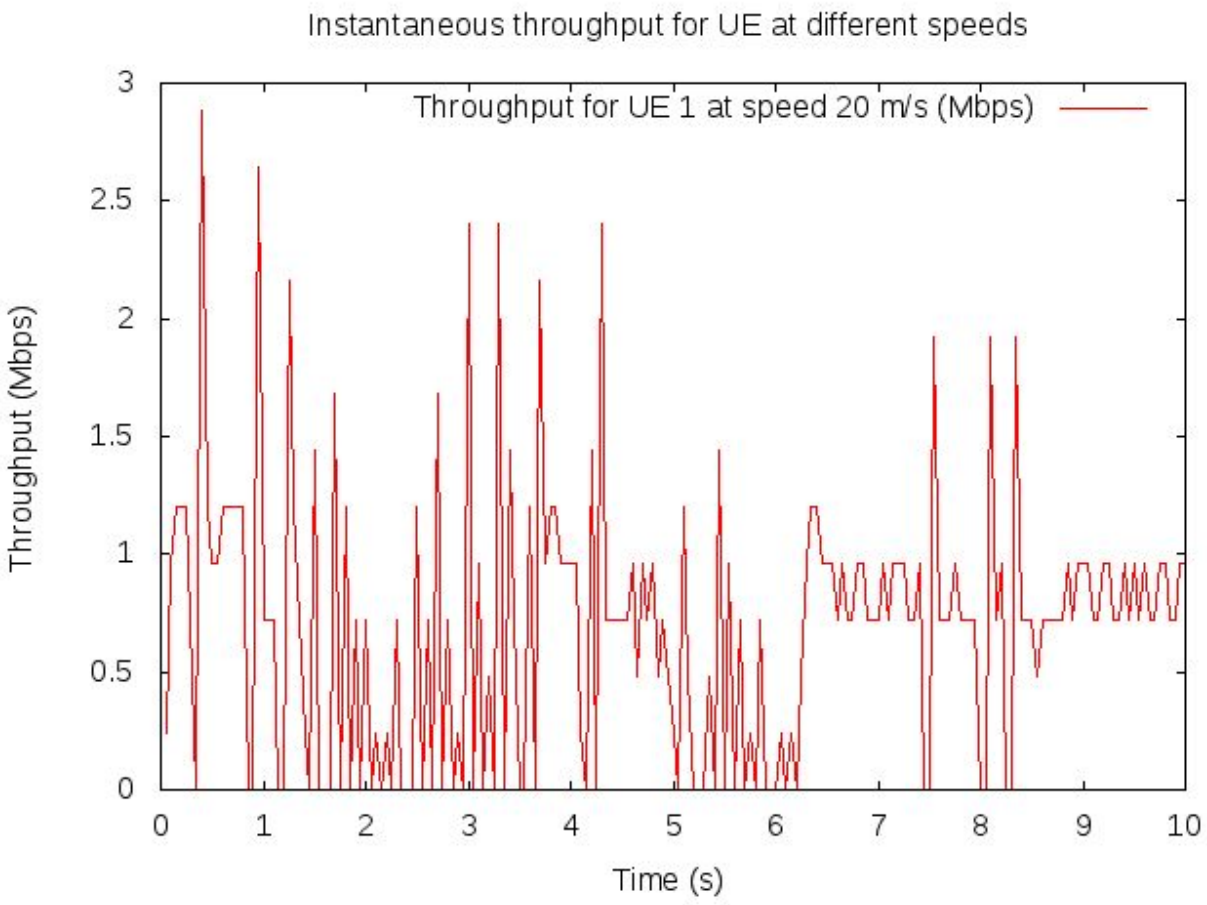
Plot 4



This is the plot for three different cases above combined. RSRP values range between -100 to -125 dBm. General trend observed is RSRP value keeps decreasing steadily when the UE is connected to a fixed eNB, after a successful handover there is a big leap in RSRP values.

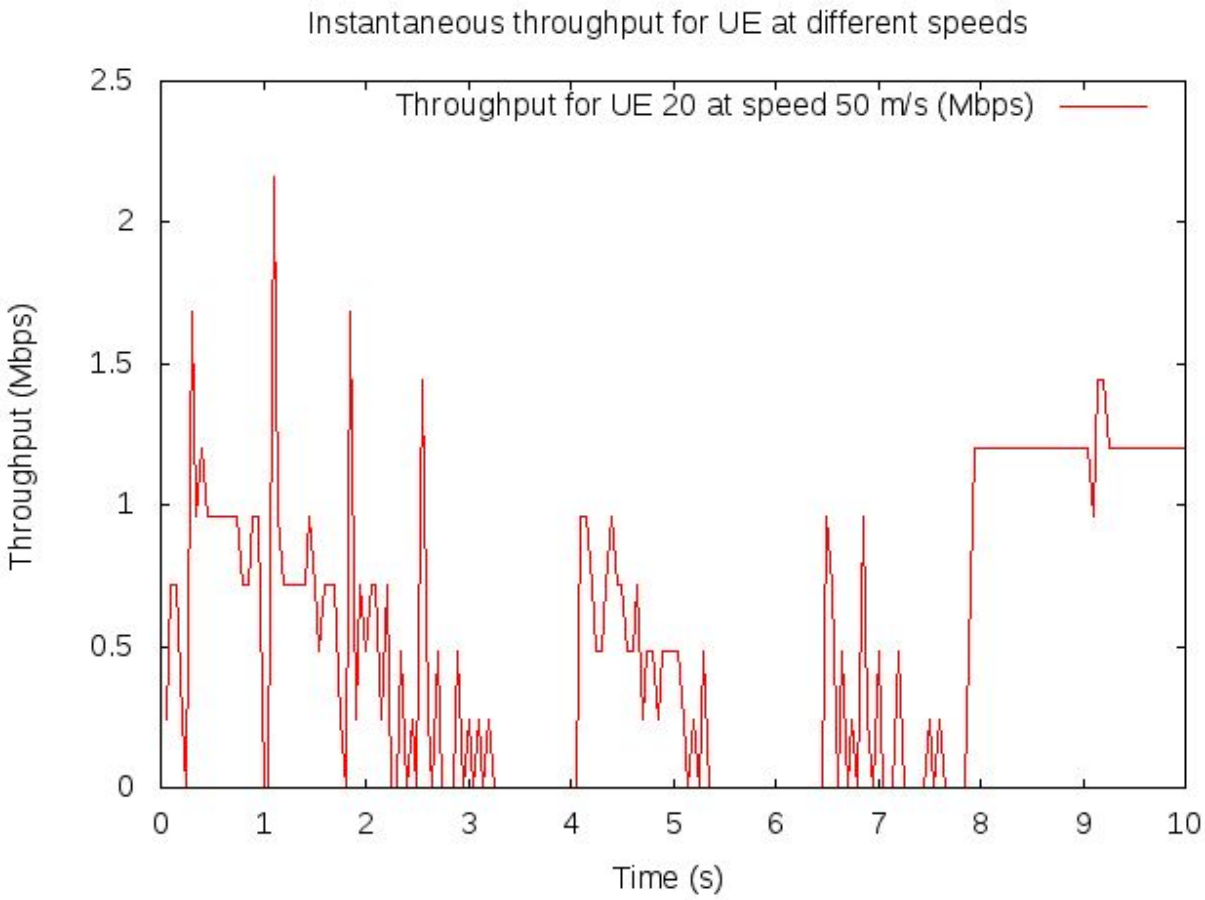
Graph 2: Instantaneous throughput in Mbps for the same UE vs speed

Plot 1:



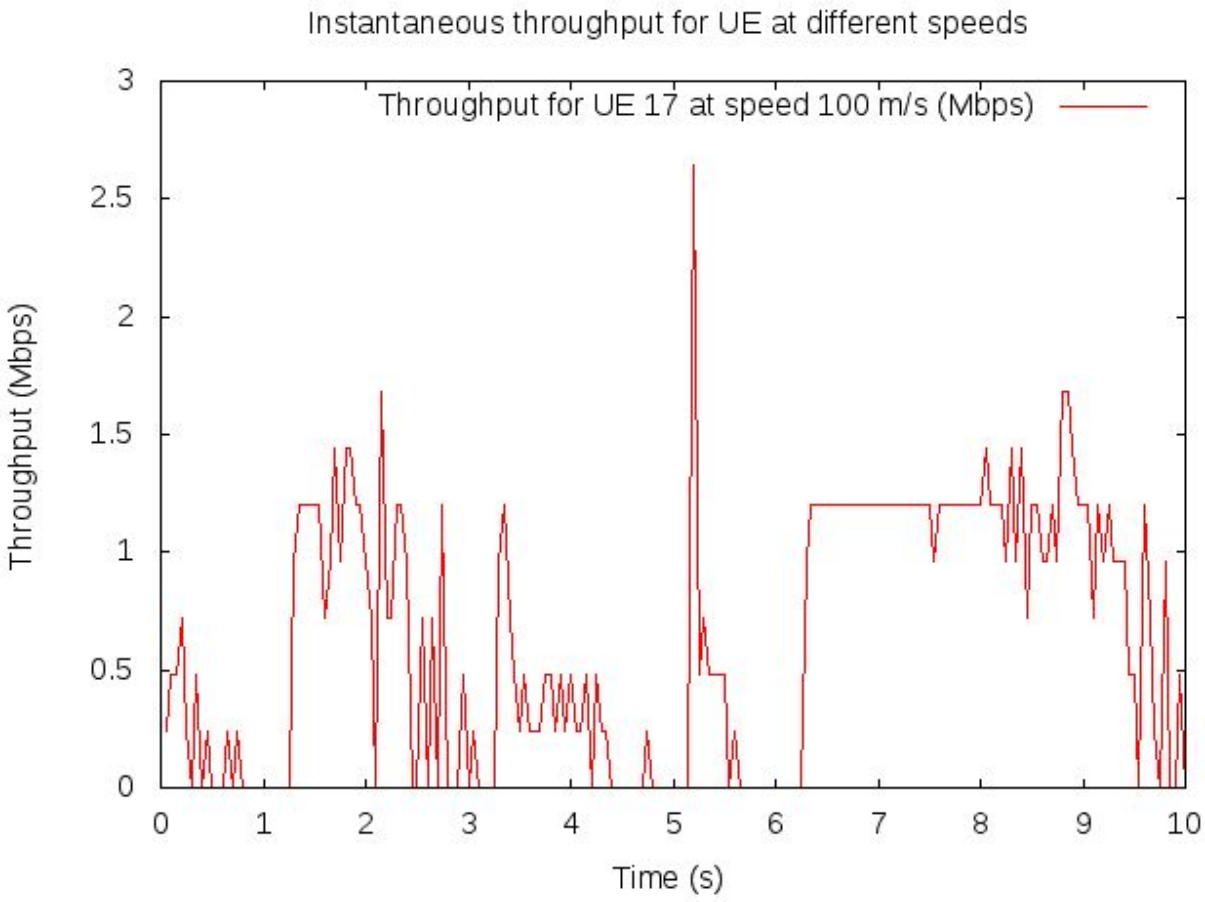
Here we consider the same UE as in graph 2 plot 1. It is clear that around 6s, there is a handover, as we observe considerable increase in throughput.

Plot 2



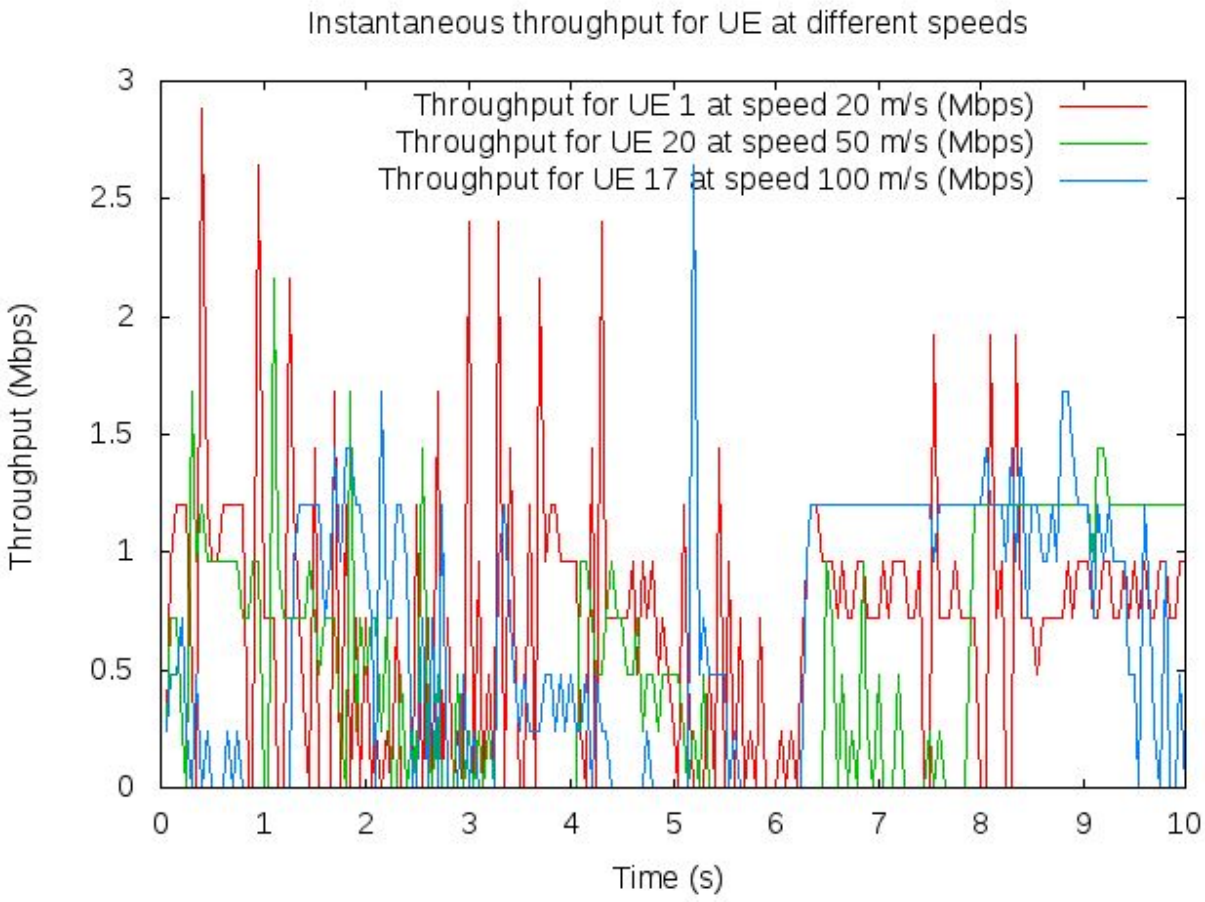
Here we consider the same UE as in graph 2 plot 2. It is clear from graph that there are three handovers (around 4s, around 5s, around 8s) as we find that throughput increases considerably.

Plot 3



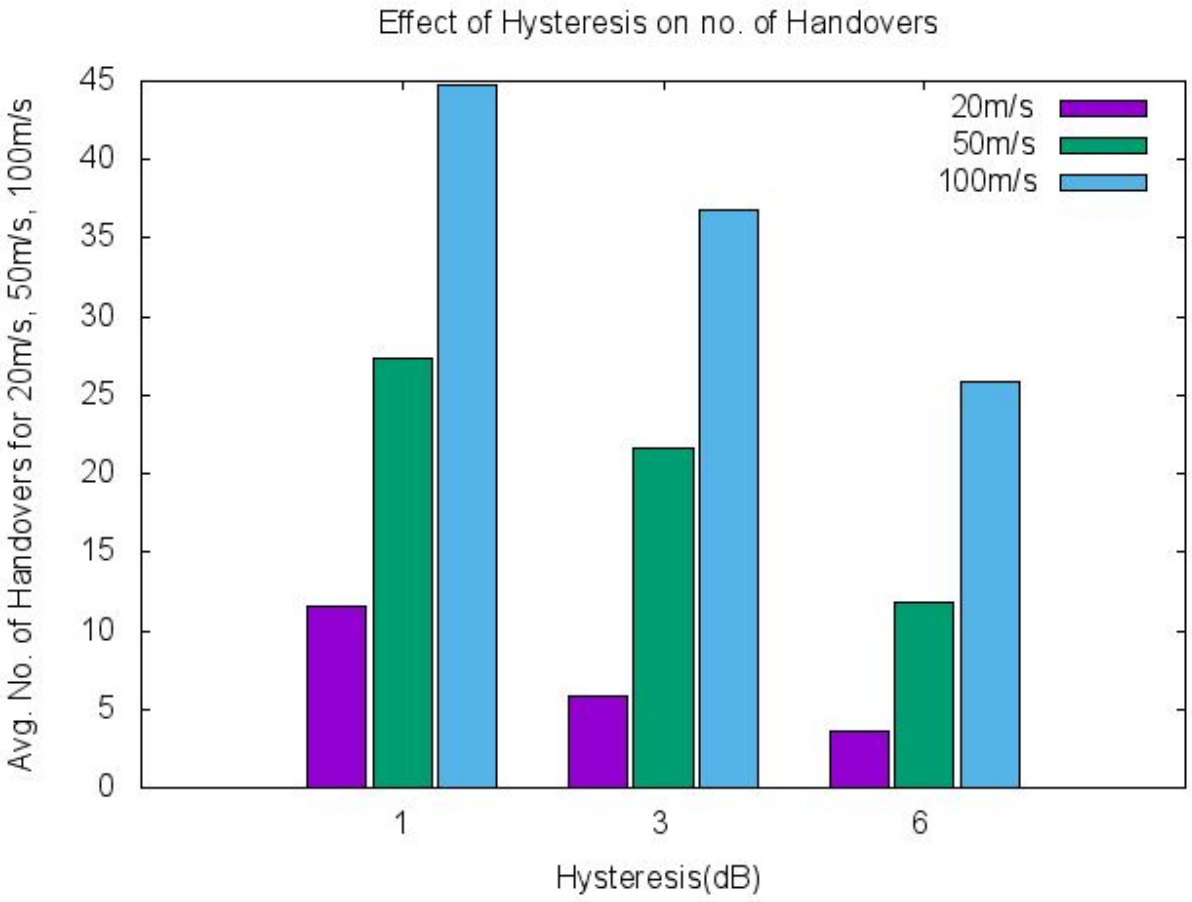
Here we consider the same UE as in graph 2 plot 3. It is clear from graph that there are three handovers (between 1 and 2s, between 3 and 4s, between 5 and 6s and between 6 and 7s) as we find that throughput increases considerably.

Plot 4



This is the plot for three different cases above combined. Throughput values range between 0 to 3 Mbps. General trend observed is that throughput increases by a considerable amount after an handover. By comparing the graphs in part 2 and part 3, we can verify the consistency of the results (handovers occur around the same time).

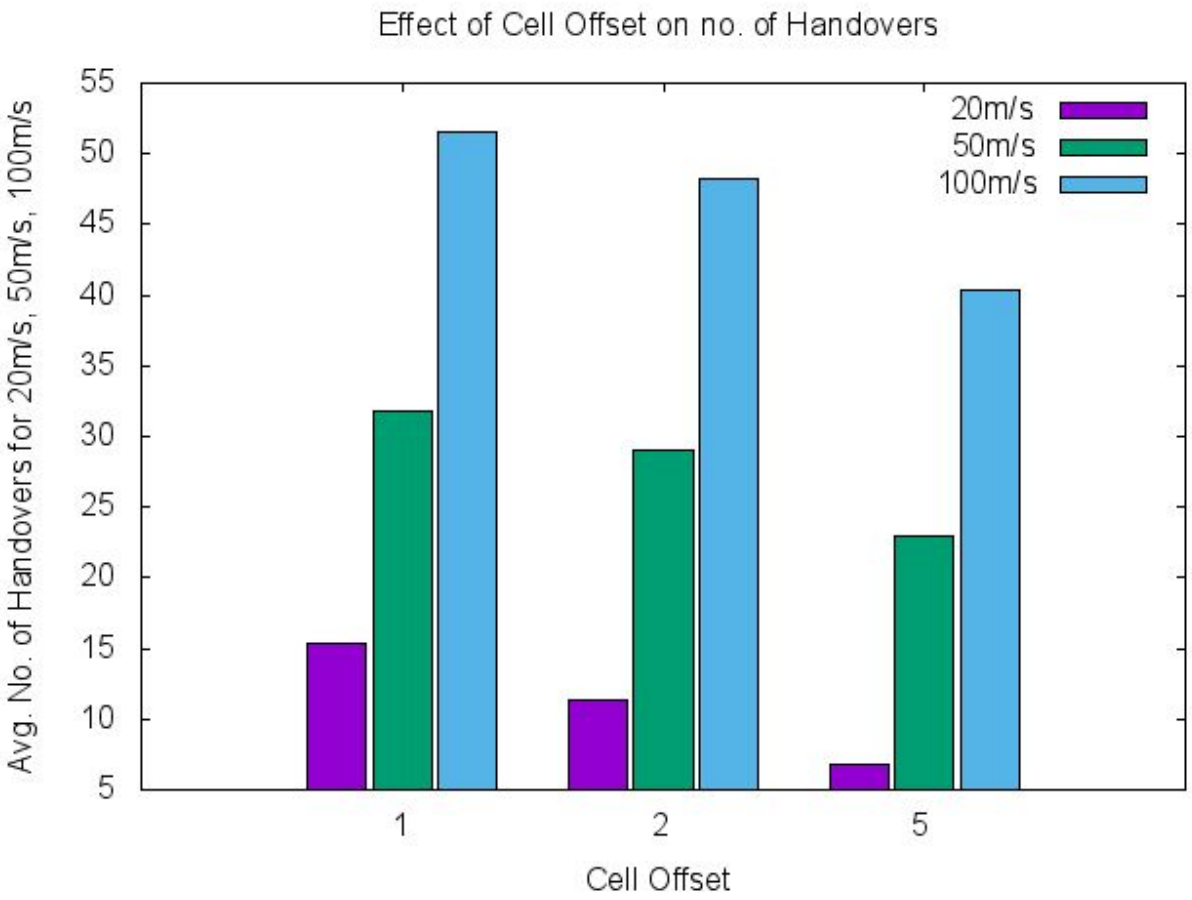
Graph 4: Study of the effect of Hysteresis on no. of handovers (A3-RSRP handover algorithm)



Observations and Explanation:

- No. of handovers increases with increase in speed.
- No. of handovers decreases with increase in hysteresis.
- Increase in hysteresis has no impact on handover failures.
- We didn't observe any handover failure during our experimentation.
- In this case, best value of hysteresis will be 1 because the average throughput is highest when hysteresis is set to one.
- No. of handovers is lesser in this case compared to the next algorithm(A2-A4RSRQ).

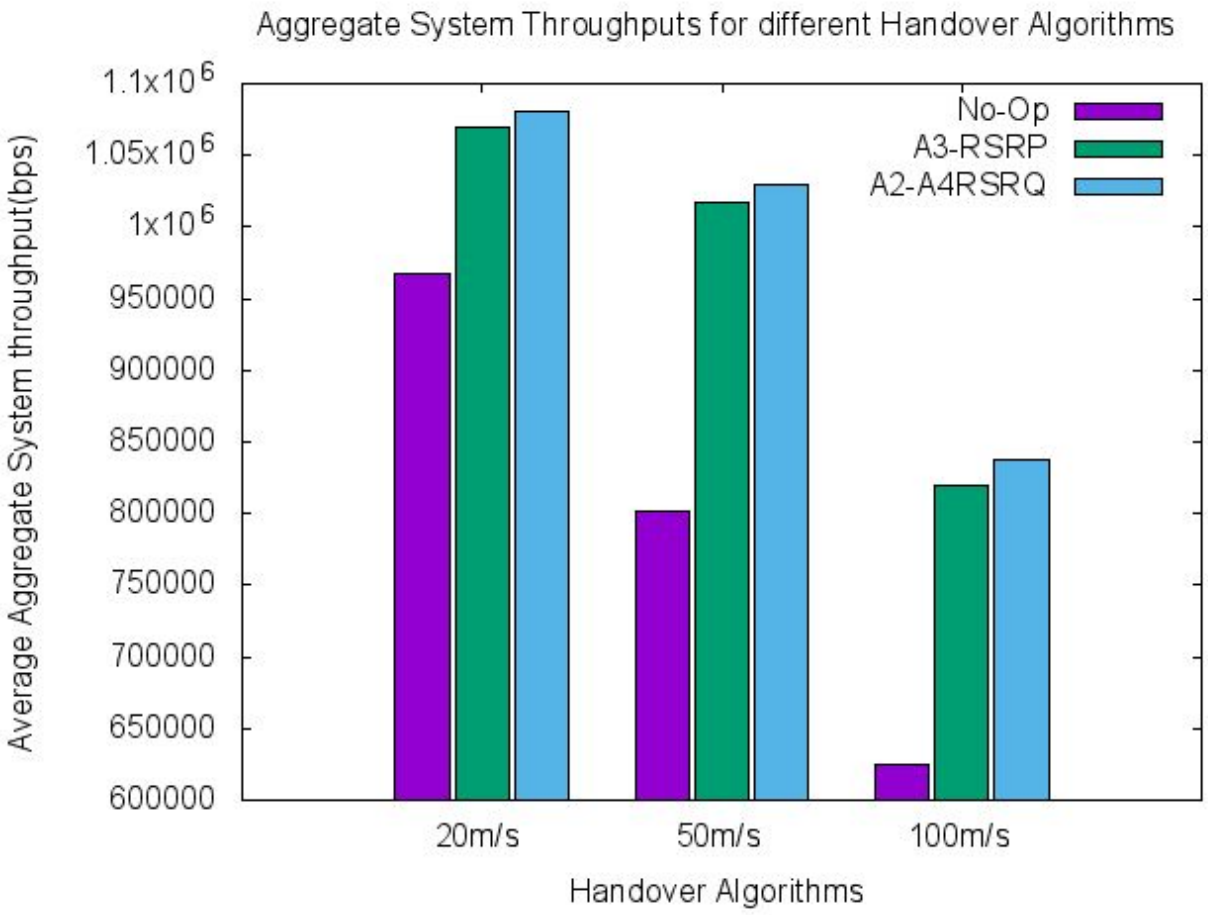
Graph 5: Study of the effect of Neighbour Cell Offset on no. of handovers(A2-A4RSRQ handover algorithm)



Observations and Explanation:

- No. of handovers increases with increase in speed.
- No. of handovers decreases with increase in neighbour cell offset.
- Increase in neighbour cell offset has no impact on handover failures.
- We didn't observe any handover failure during our experimentation.
- In this case, best value of neighbour cell offset will be 1 because the average throughput is highest when cell offset is set to one.
- No. of handovers is more in this case compared to the previous algorithm.
- Throughput in this case in higher than in the previous case.

Graph 6: Aggregate System Throughputs for different Handover Algorithms



Observations and Explanation:

- Throughput decreases with increase in hysteresis.
- Throughput decreases with increase in neighbour cell offset.
- Best value of hysteresis taken = 1 (High throughput)
- Best value of cell offset taken = 1 (High throughput)
- Throughput decreases with increase in speed.
- **Order of throughput:** A2-A4 RSRQ > A3-RSRP > No-Op
- So, RSRQ algorithm gives the best performance followed by RSRP and No-Op gives the worst performance.