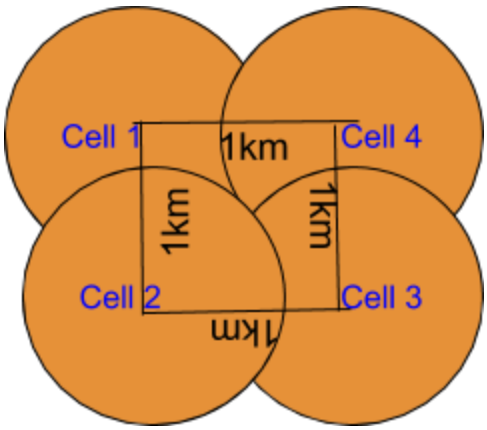
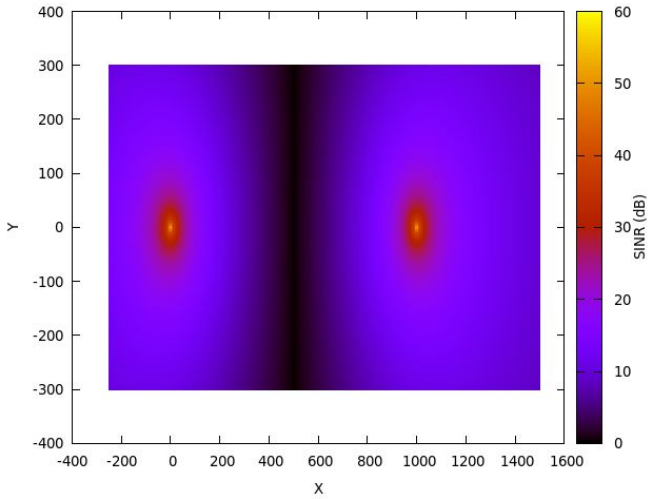
Group Assignment

Group size: 2

The objective of this assignment is to understand and change code of LTE Schedulers algorithms in NS-3 for necessary stats collection. Further, you need to evaluate and compare performance of different Scheduler algorithms.

Create a topology as shown in below figure. Add P-GW and Remote Host to this topology and connect them with point-to-point link of 1 Gbps





A Sample REM plot with two eNB

Configure eNBs and their UEs with the parameters as given in below Table:

Simulation Parameter	**Value**
Number of UEs	5 per eNB; 1 Downlink UDP Flow per UE from the Remote Host
Number of eNBs	4
Inter distance between eNBs	1 KM
eNB Tx Power	30 dBm (1W)
Application Type	UDP
Full buffer case (UDP Traffic)	1500 bytes per every **1ms** by UDP; Each UE is configured with 1 DL UDP flow of 12 Mbps
Non Full buffer case (UDP Traffic)	1500 bytes per every **10ms** by UDP; Each UE is configured with 1 DL UDP flow of 1.2 Mbps
UE mobility speeds	0, 5 m/s; where in a given expt all UEs are configured with one of these two speeds
UE mobility model	RandomWalk2d Mobility
UEs placement in a Cell	Random disc placement within 500m radius of eNB
# of RBs	50 in DL and 50 in UL (LTE FDD)
UE attachment to eNB	Automatic to one of eNBs based on received signal strength, so handovers may take place during mobility
Total simulation time	10 seconds
Number of seeds per experiment	5; RngRun1 = "Last TWO DIGITS of one of your ROLL NUMBERS" RngRun5 = RngRun1+4

Compare Proportional Fair (PF), Round Robin (RR), Max Throughput (MT) and Blind Average Throughput Schedulers (BATS) available in NS-3 LENA LTE module by creating a 4-cell LTE

network as shown above with the simulation parameters given in the table. **Your main script (e.g., asg1.cc) should take scheduler type, speed, RngRun, and fullBufferFlag as inputs.** Refer here to know more about these scheduling algorithms. References section contains all necessary reading material to complete the assignments. If you still have any doubts, post your queries in Group Asg 1 Discussion thread on GC to get help from TAs/me/other groups.

You need to turn in the following graphs:

Graph 1: SINR Radio Environment Map (REM) of 4-cell topology given above.

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. Get sum of throughputs of all 4 cells (i.e., all 20 UEs flows) in different runs by varying seed values and then get the average of that for plotting.

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. But here you need not to do any averaging. Have list of per UE throughputs across all cells in all different runs by varying seed value and use that for plotting CDF.

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. Refer Help section at the end of this document to know how to measure Instantaneous throughputs.

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

Graph 6: Repeat now for non full buffer scenario and report your observations. X-axis: Speed (0) m/s; Y-axis: (Average Aggregate System throughput) with bars for four scheduler algorithms

For each of the graphs, provide analysis of the trends observed and quantify improvement in system throughput for the scheduling algorithms studied.

Deliverables

The following items are supposed to be included in a tar.gz file and uploaded on GC.

1. Assignment Report having all graphs and their detailed analysis of trends
2. The ns-3 scripts involved in creating the testing scenarios described above with appropriate names.
    a. main script (e.g., GroupID-Asg1.cc)
    b. other supporting (modified) scripts of NS-3, if any
3. All stats collected and any other outside scripts (shell, python, perl, gnuplot etc) for collection of results and plotting of data.

4. README file should contain how to run your simulations and generate graphs.

NOTEs:

- *Running simulation, Stats collections and generating Gnuplots should be automated.*
- *Each of the above deliverables carries marks so you need to upload all valid docs, code, and scripts.*

Late Policy:
7 slip days overall
10% cut in marks for each day beyond slip dates.

References

1. **https://www.nsnam.org/docs/models/html/lte-design.html#mac**
2. https://www.nsnam.org/docs/models/html/lte-user.html#radio-environment-maps
3. https://www.nsnam.org/docs/models/html/lte-user.html
4. https://www.nsnam.org/docs/models/html/lte-design.html#round-robin-rr-scheduler
5. http://code.nsnam.org/ns-3-dev/file/028452e3b558/src/lte/examples/lena-rem.cc
6. http://code.nsnam.org/ns-3-dev/file/028452e3b558/src/lte/examples/lena-intercell-interference.cc

*While changing existing schedulers code, better make your own copies and rename them with your algorithm names. You can add your new programs (.cc and .h file names) in **wscript** file of lte module. So that other source files dependencies may not exist.*

- **How to measure instantaneous throughput at the MAC scheduler?**

Check the function DoSchedDlTriggerReq() in proportional fair scheduler code *PfFfMacScheduler.cc*

```
for (itStats = m_flowStatsDl.begin (); itStats != m_flowStatsDl.end (); itStats++)
{
        (*itStats).second.totalBytesTransmitted += (*itStats).second.lastTtiBytesTrasmitted;
 }
```

- **RR downlink scheduling ns3 implementation details.**

File Name: Rr-Ff-Mac-Scheduler.cc

Type0Allocation:

```
static const int Type0AllocationRbg[4] = {
  10,     // RGB size 1
  26,     // RGB size 2
  63,     // RGB size 3
  110     // RGB size 4
```

};

In Downlink scheduling trigger function: DoSchedDlTriggerReq ()

1. RBG size:

```
int rbgSize = GetRbgSize (m_cschedCellConfig.m_dlBandwidth);
int rbgNum = m_cschedCellConfig.m_dlBandwidth / rbgSize;
```

2. Finding Active flows and UEs: Traversing the below map active flows of UEs are identified.
RLC PDU to transport block mapping also done through below code

```
std::list<FfMacSchedSapProvider::SchedDlRlcBufferReqParameters>::iterator it;
 for (it = m_rlcBufferReq.begin (); it != m_rlcBufferReq.end (); it++)
        {
// find active flows
// find Total transport blocks
        }
```

3. Divide the resource equally among the active users according to Resource allocation type 0:

```
  int rbgPerTb = (nTbs > 0) ? ((rbgNum - rbgAllocatedNum) / nTbs) : INT_MAX;
```

4. Filling DCI for active UEs and send on PDCCH:

```
 // create the DlDciListElement_s
     DlDciListElement_s newDci;
          newDci.m_rnti = (*it).m_rnti;
          newDci.m_rbBitmap = rbgMask;
m_schedSapUser->SchedDlConfigInd (ret);
```