Title :
Dynamic Control of RLC Buffer Size for Latency Minimization in Mobile RAN

Names and Roll Nos of Group members :
P V Asish - CS18BTECH11037
Sai Nishith J - CS18BTECH11018
G Vishal Siva Kumar - CS18BTECH11013

Problem statement:
Minimization of exogenous delay for an interactive flow that shares a mobile wireless link (radio bearer) with at least one greedy flow(video streaming service). Exogenous delay is the delay that occurs when packets of the application flow find the link buffer loaded with packets of other applications and it is particularly harmful to low-bandwidth interactive applications, as it inflates their latencies despite the negligible load that they impose on the network.

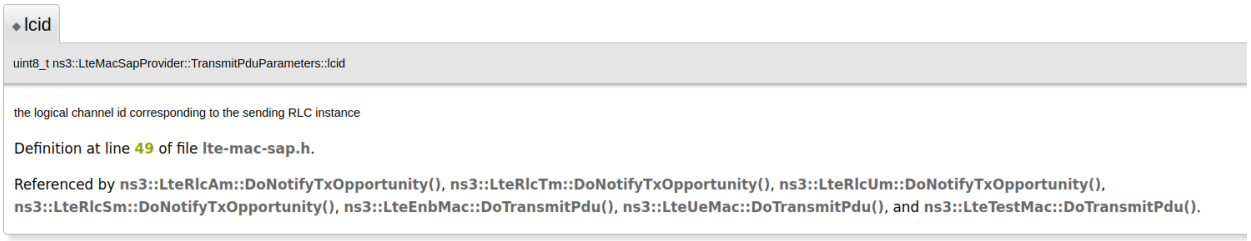
So the idea was to introduce per flow queuing at the PDCP layer. Basically, paper proposes to store the packets from different flows in different queues. Also there needs to be a scheduler which takes a certain amount of packets from different queues and sends them to the RLC layer rather than directly sending packets to the RLC layer. The paper proposes that doing this will significantly reduce the exogenous delay for URLLC traffic.

Project description (Summary of the work done)
i. Discussion about challenges faced and how they are addressed (if applicable) :
We started off by studying the base paper in detail and got a clear understanding of the problem which was being discussed in the paper. Later, we started working on the simulation of a problem which was mentioned in the paper in ns3 as per feedback from the first presentation. This took us some time to implement. We used the example code “nr-demo.cc” and modified it to our use case. We generated eMBB video traffic and a URLLC traffic using On-Off helper. Initially we weren’t getting results we were expecting i.e the delay of URLLC traffic was not increasing with the increase in traffic rate of eMBB. But later we realized that the link was already saturated and tried the simulation with reduced traffic. This gave expected results and delay of URLLC traffic was being affected by eMBB traffic. Everything until this point is one major part of the project.

Then we started to implement queuing at the Pdcp layer which is the most crucial part of the project. The idea was to use an extra scheduler thread which takes the receiving packets placed in different queues in the original pdcp method and sends them to the RLC layer. But the issue we were facing here is, the method which invoked the scheduler thread wasn’t accepting new packets unless the scheduler thread joins back. This was the major problem which held us back. We couldn’t find any solution for this problem.

Later we started looking at MAC layer codes as suggested by sir in the pre-final review. The suggestion given to us was to check how the RR scheduler was working. But on looking at

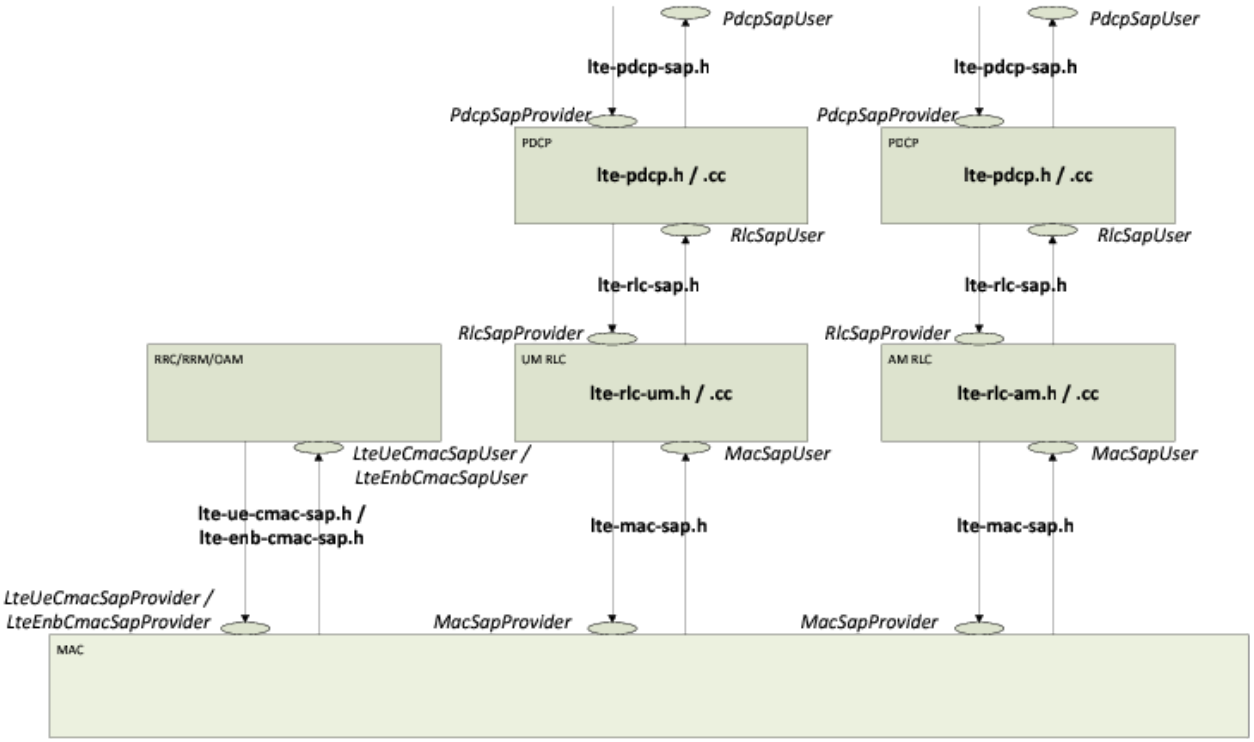
codes, lcid (logical channel id) wasn't really being used. But the rnti value was being used in a couple of codes. Below is the picture which shows various methods which references lcid.



Source :
https://www.nsnam.org/doxygen/structns3_1_1_lte_mac_sap_provider_1_1_transmit_pdu_parameters.html#a1aeffc9e47e7ecda7630ad4d0dd88506

This could be because Round Robin or any scheduler in general just focuses on traffic of various users i.e scheduling users one by one and not scheduling the different traffic types of a single user. One more approach we were trying to randomly shuffle the order of packets in the RLC buffer but no change was observed in delays of any flow. So this kind of put us at a dead end.

ii.Add illustrations to better convey your point. Acknowledge source(s) if you are using any figures from other sources



Source : https://www.nsnam.org/docs/release/3.14/models/html/lte-design.html#fig-lte-rlc-implementation-model

In the paper, we were trying to address the exogenous delay in downlink. As we can see in the ns3 structure illustration above, we started to work on the **PdcpSapProvider** which manages the inflow of packets to the pdcp layer. So we tried to create multiple flow queue buffers in the

pdcp layer to manage exogenous delay. But the **PdcpSapProvider** method takes a packet sequentially and transmits it to the Rlc layer using **DoTransmitPdcpPdu** method. We created a buffer and stored the packets instead of transmitting directly to the Rlc layer. Then we used a threaded method to check the buffer parallely to the **PdcpSapProvider (**where the packets are being added to the buffer**)** and transmit the packet based on Ip Flow. But the multithreaded method didn't work as intended as the **PdcpSapProvider** method call needs to return to let the next packets enter the layer. This is due to the single threaded nature of the ns3 application.

So we tried to replicate the functionality in the Rlc layer by trying to reorder the Rlc buffer.We were trying to randomly shuffle the packets in the RLC buffer but no change was observed in delays of any flow. So this kind of put us at a dead end.

Detailing how the project is implemented in/using NS-3

i. Provide details of modifications to NS-3 files and new files created:

Modifications :

1. pdcp-lte.cc and pdcp-lte.h : Made a change to DoTransmitPdcpSdumethod. Added a new method LtePdcp::pdcptorlc() (line 192), which aims to send packets from queues to the RLC layer. Declaration to this function has been added in pdcp-lte.h
2. lte-rlc-am.cc : In LteRlcAm::DoTransmitPdcpPdu function, changed the order of packets in the RLC buffer randomly to observe changes in delay. (line 177)

New Files :

1. exogenousdelay.cc : To simulate the problem statement and was originally supposed to compare the new implementation with the original ns3 implementation.

ii. Discussion about implementation challenges faced and how they are addressed (if applicable) :

The major implementation change was the threads issue as mentioned above which was left unsolved. Also, since most work revolved around Pdcp and RLC layer code, the code base was extremely large especially for RLC. (because of various modes, buffers).

Another issue which was mentioned earlier is the on-off helper, but luckily we could find online resources which helped us simulate the URLLC traffic.

Simulation/experimental Setup

i.Test scenario created/used to conduct the experiments :

A single user with two types of traffic.

1. URLLC traffic
2. Video Streaming traffic

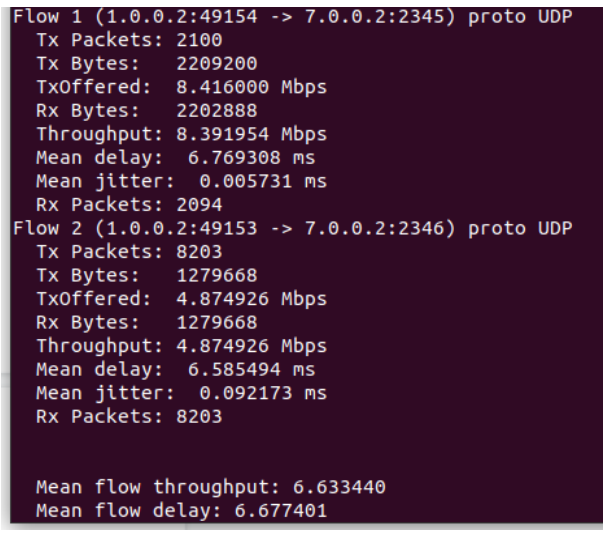
ii.Simulation parameters in tabular form:

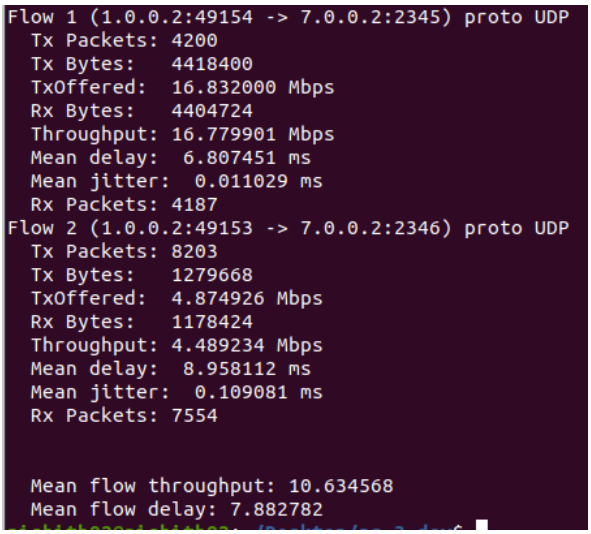
numerology	1
number of UE	1

number of Base stations	1
centralFrequencyBand	6 Ghz
bandwidth	25 Mhz
udpPacketSize(URLLC)	128
udpPacketSize(Video traffic)	1024
Simulation Time(seconds)	2.5

Performance metrics and Results along with their detailed analysis:
Problem simulation results:

....
Flow 1 : eMBB traffic, Flow 2 : URLLC (single user, two kinds of traffic - RLC AM mode)





In left picture, we can observe that delay of URLLC traffic is 6.58 ms but if we increase the eMBB rate of transmission from 8.41 Mbps to 16.83 Mbps, URLLC traffic delay is increased to 8.95 ms which is the exogenous delay observed in this scenario as buffer is being filled with eMBB packets.

Conclusions and possible future directions:
We couldn't resolve the thread issue, which is one possible implementation of the paper's idea. The sequential nature of ns3 is probably the stopping factor for this implementation. We couldn't figure out other ways to parallely receive and send the packets from queuing. Having spent a lot of time on the code base project, we wish we could have solved the issue. But, the claims made in the paper are pretty significant. If this can be implemented to a real 5G stack, then clearly there would be significant reduction in exogenous delay.

Describe how each member of the team contributed during the course of the project execution:

All of us have gone through the majority of the all codes required for the project due to occasional dead ends to specific people looking at certain codes. But to be more specific about work division :

Sai Nishith : Problem Simulation part and studying ns3-LTE documentation

PV Asish : Studying Pdcp layer codes and thread implementation at Pdcp

Vishal Siva Kumar: Studying RLC layer codes and thread implementation at Pdcp

References:

Dynamic Control of RLC Buffer Size for Latency Minimization in Mobile RAN**:**
https://ieeexplore.ieee.org/document/8377190

On-Off helper for URLLC traffic :
https://stackoverflow.com/questions/60791009/traffic-modelling-on-ns-3-network-simulator

NS3-LTE:
https://www.nsnam.org/docs/release/3.14/models/html/lte-design.html#fig-lte-rlc-implementation-model