Smart Backlog Management to Fight Bufferbloat in 3GPP Protocol Stacks

Pasquale Imputato*, Natale Patriciello†, Stefano Avallone*, Josep Mangues-Bafalluy†
*Università degli studi di Napoli Federico II
Via Claudio 21, 80125 Napoli, Italy
†Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA)
Av. Carl Friedrich Gauss 7, 08860 Castelldefels (Barcelona), Spain

***Abstract*—In this work, we propose an innovative approach to contrast the bufferbloat, one of the primary sources of queueing latency, in 3GPP mobile networks. Our purpose is to validate an all-software, in-node traffic-control infrastructure, similar to what the Linux operating system already employs for Ethernet and WiFi networks. We introduce the IP traffic-control infrastructure in the RAN part of an LTE network, for both UE and eNB, that can perform packet scheduling on flows directed to the same Data Radio Bearer. The effectiveness of such strategy leverages on a reduced usable size of the in-node RLC buffer, thanks to the BQL algorithm, which we evaluate for the first time in the context of LTE. By using BQL, we can create a backlog of packets that stays in the Traffic Control layer, on which it is possible to perform packet scheduling without bloating the RLC buffer. We compare the performance of BQL to another algorithm, called DynRLC, proposed to dynamically size the RLC buffer eNB side. The preliminary evaluation of the proposed approach is done on the ns-3 network simulator and it shows that the proposed approach with BQL significantly improves the flow delay, by controlling the queueing latency, as well as enhancing the flow isolation.**

I. Introduction

The word bufferbloat entered the dictionary of scientists and researchers since 2011 when Jim Gettys first discovered it in residential settings. It is a term to define the existence of unreasonably large and full buffers inside any network. Over the years, technology standards have not kept up with research on this matter. Buffering policies are considered implementation-specific, and the designers' efforts went instead towards defining different cooperating layers in the 3GPP standard to provide a guaranteed Quality of Service (QoS) for paying customer. Different flow requirements are reflected in various bearer (a virtual transport pipe) properties, that so are served differently from the network.

Every User Equipment (UE) device has an "always on" default bearer activated at the time of device initialization. Other dedicated bearers, to serve flows with different priorities, are left for particular customers (such as public safety operators, or companies with significant contracts). Moreover, an Evolved Node B (eNB) encapsulates an end-to-end data bearer over a radio bearer for the over-the-air transmission. Unfortunately, opening and closing a bearer is not straightforward, and requires signaling between the network user and the network operator, increasing the network management effort. Then, the operator is prone to keep a fixed number of opened bearers to relieve the network management overhead. From our investigation in Android devices, and according to [1], the operator uses a single bearer for a regular user where all the user applications data (such as HTTPS video transfer, PUT/GET HTTPS request, a Skype call, a WhatsApp message) pass through. Therefore, in the rest of this paper, we assume this typical model in which all the user data goes into the default bearer.

In this paper, we present a novel way to solve bufferbloat problem in 3GPP mobile networks, hopefully shedding some light on this problem for the current standardization process of 5G New Radio (NR). In this proposal, we connect the Linux Traffic Control infrastructure on top of the 3GPP stack, employing a cross-layer approach to fight bufferbloat. In practice, the 3GPP and the IP level apply a mild form of *flow-control*, which will allow packets to be stored for a while inside the IP layer itself. The flow control is managed by the Byte Queue Limits (BQL) algorithm [2]. We present a preliminary evaluation of BQL in LTE as a means for keeping the size of the Radio Link Control (RLC) buffers at the minimum value that ensures that throughput is not impacted due to starvation. In this way, IP can do packet scheduling between different flows directed to the same bearer, prioritizing interactive traffic over bulk traffic by reusing existing packet schedulers and Active Queue Management (AQM) algorithms, such as FQ-Codel [3]. The result is the possibility of performing QoS-based decisions even with a single bearer. We analyze our architectural proposal with ns-3 simulations, comparing in different scenarios the performance obtained with and without flow control, testing both BQL and DynRLC (one of the most promising proposals already present in the literature to dynamically sizing the RLC buffer size on the eNB). The ns-3 network simulator has an advanced Long Term Evolution (LTE) model (LENA) [4] and a complete representation of the traffic control layer at IP level, designed by taking inspiration from the Linux kernel [5]. In this way, we can show that by having a limited amount of buffering inside the LTE devices, the latency and the throughput performance improves dramatically, as well as providing fair scheduling between different flows. Being tied with the reality allows, in a line of principle, a device vendor to port our findings to existing

Android devices with a minimum effort.

The paper is organized as follows: in Section II we present the related work, and why our proposal differs. In Section III we briefly explain our view for flow-control, traffic control, and how these are implemented in the Linux kernel, as well as a quick recap on the part of the 3GPP stack affected by our proposal. In Section IV we detail the core of our proposal and explain how we have implemented it in the ns-3 network simulator. In Section V we present the simulation results, and then in Section VI we present our conclusions.

II. Related Work

In [1], authors characterize the traffic of the LTE networks to estimate the bufferbloat-inducted latency on the UEs. Trying to mitigate the problem, the authors introduced a flow control to limit the amount of data injected into the device firmware and propose a differentiation scheme in the Android Traffic Control (TC). The work does not consider the use of fairness scheme or the cross effect of their flow control on the AQM. Indeed, they study the poor performance of CoDel on the Android stack.

In [6], the authors propose an algorithm to dynamic sizing the RLC buffer for the eNBs, i.e., DynRLC, and introduce a per-flow fair queueing strategy at PDCP layer. A proposed flow control regulates the passing of packets between the two layers. They tune the proposed algorithm to reduce over queueing at RLC layer. The work does not consider either the use of AQM in their PDCP queueing scheme or the use of advanced fair queueing algorithms such as FQ-CoDel. Also, they do not consider and evaluate BQL.

Another attractive way to try to reduce the bufferbloat phenomena is to insert, at various levels in the stack, pure AQM algorithms to manage queues. However, employing an AQM (that can decide to drop packets if the algorithm has the perception that the amount of data stored in the buffer is not appropriate) can lead to issues when coupled with the MAC layer decisions. First, the AQM algorithms have to be modified to avoid the possibility of dropping a partial piece of a packet. In fact, at RLC level the packet can be segmented: what would happen with an algorithm such as CoDel, when the head is segmented and a piece transmitted, while the other portion (the new head) has to be dropped because its waiting time is higher than the threshold? This issue is investigated in details in [7], with a modification to the CoDel algorithm proposed (and implemented) to avoid such effect. However, another question arises: in the uplink, the amount of data stored in the buffer is passed through Buffer Status Report (BSR) to the eNodeB. A similar message is exchanged between MAC and RLC (inside the eNodeB) in the downlink case. Therefore, dropping entire packets inside the radio stack is problematic for the MAC scheduler. Given the reported buffer status, some space may be reserved for a particular data radio bearer. Due to the unexpected drops, at the moment of the transmission, the bearer could have less than the previously reported data, leading to an under-usage of the resources, and therefore to degraded performance. For this reasons, we perform any necessary drop at a higher level.

In [8], the authors propose a dynamic adjustment of the TCP receiver window to reduce the excess of bytes in flight, that in turn is helping the network to buffer less overall data. The problem of this widespread idea (limiting the receiver window to restrict the sender is used successfully in many other fields) is that until all the devices are updated, network-level bufferbloat created by non-patched terminals will continue to impact flows coming from updated devices. Nevertheless, limiting the receiver window still has the potential issue to limit the overall transmission rate in case something is not such as the receiver is guessing.

Our proposal is to introduce a flow control similar to the one in [6] for both UE and eNB, on top of the LTE/NR protocol stack, that uses the BQL algorithm (already available in the Linux kernel) to dynamically size the RLC buffers. On this flow control, for both UE and eNB, we exploit a consolidated IP TC infrastructure, conversely to what in [1], [6] and [7]. Our strategy allows to avoid changes in the AQM algorithms, conversely to what in [7], since we keep a FIFO RLC buffer size regulated with BQL. Indeed, our approach avoids interference on the control plane of LTE and its mechanism of BSR. Finally, our proposal works regardless of the TCP receiver and sender windows and its congestion avoidance algorithms, conversely to what in [8].

III. Background

A. Linux traffic-control

In the following, we will use the term TC to refer to an in-node infrastructure to police the incoming traffic, shape the outgoing traffic and schedule the packets to be transmitted. In the Linux kernel, once an outgoing interface has been determined, the packet is enqueued into the *queueing discipline* (commonly abbreviated as qdisc) virtually linked to the outgoing interface. A qdisc implements a scheduling policy: packets are stored in possibly distinct queues and the qdisc has to select the next packet to move to the device driver. Qdiscs implement various strategies, aiming to, e.g., prioritize specific flows (e.g., pfifo_fast, prio) or keep the experienced delay under control. Indeed, popular AQM algorithms to control the queuing delay, such as RED [9], CoDel [10] and FQ-CoDel [3] are available in the Linux kernel. Qdiscs can also drop packets either before enqueueing them or after dequeueing them. Such a possibility is exploited by AQMs to indirectly notify the TCP congestion control that the transmission rate should be decreased to avoid the occurrence of congestion.

Packets dequeued by the qdiscs are then passed to the network device drivers, which store them (in the so-called *transmission ring*) until they can be transferred to the hardware. As shown in [2] through experiments, the size of the transmission ring may have a significant impact on the delay experienced by packets and contribute to the bufferbloat phenomenon. Unfortunately, the right size is not constant, but it depends from the actual transmission rate of the device. As a countermeasure, the Linux kernel provides the Byte

Queue Limits (BQL) algorithm, whose goal is to dynamically identify the maximum amount of bytes that can be stored in the transmission ring to minimize the latency while preventing starvation.

A flow control mechanism is implemented to regulate the transfer of packets from a qdisc to the network device driver. Every time a device driver is ready to queue new data for the actual transmission, but it was in a stopped (or just initialized) state, it sends a notification to the qdisc (*wake*). If the device driver is overwhelmed by the qdisc with an excessive amount of data, which cannot be stored in its queues, there is also a *stop* notification to block the qdisc (this is a hard stop, defined as backpressure in the literature). In all other cases, the BQL algorithm decides how many bytes have to be passed to the device driver, soft-stopping the qdisc after the defined bytes quota has been moved to the device driver. This quota is calculated dynamically by taking as input the number of bytes that were enqueued in the transmission ring (*NotifyEnq*), as well as the number of bytes transmitted over the medium (*NotifyDeq*). To date, most of the Ethernet network device drivers in the Linux kernel support BQL.

B. LTE MAC/RLC

In the 3GPP model, the layer responsible for storing the data waiting for transmission over the air is RLC, with one buffer for each bearer. The RLC layer, to which Data Radio Bearer (DRB)s belong, provides that data fragmentation and reassembly feature. There are various modes for the RLC layer. For instance, there is the unacknowledged mode (UM), that will not perform retransmissions, and the acknowledged mode (AM), which contains retransmission buffers (per-DRB) in case the data is not ACKed by the receiver.

The algorithm that determines what data goes into the air is called Radio Data Scheduler (RDS), and is inside the eNB MAC layer. It decides how to fill in time and frequency each slot, according to some policy (e.g., round-robin or proportional fair). It uses as input the BSR messages that come from the RLC layer of each connected UE to prepare the part of the frame that will be used by a UE to transmit uplink data. The eNodeB MAC layer uses similar messages (transmission opportunity, TxOpp) from its own RLC layer to prepare the frequency/time blocks that will be used to send downlink (from the UE perspective) data.

This model discourages the usage of AQM algorithms, as we argued in the Section II, because dropping one or more packets at RLC layer with AQM algorithms invalidates the BSR information that the MAC uses to do the appropriate scheduling. So, in the majority of cases a FIFO queue is used: as the networking community knows since RFC 2308, uncontrolled FIFO queues lead to problems such as flow lockout and a difficult sizing process, which induce increasing latency. Recently, an algorithm named DynRLC has been proposed [6] to dynamically control the RLC buffer size on the eNB. The idea is to define an RLC queueing time target, i.e., a DeLay Threshold (DLT), and sizing the RLC buffer to keep the queueing delay under DLT. DynRLC periodically estimates the average queueing time and defines the current size of the RLC buffer according to DLT.

IV. Adding TC on top of the 3GPP stack

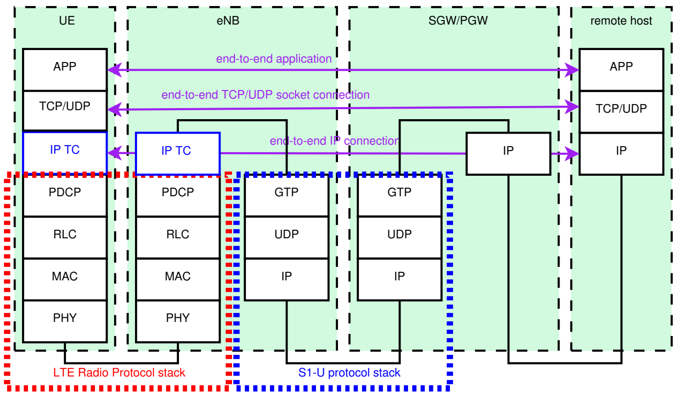


Fig. 1: LTE-EPC data plane protocol stack with the introduction of TC on top of the LTE model.

In this section, we present our approach to fighting bufferbloat in LTE, which consists in placing the IP TC infrastructure on top of the LTE stack and adequately handling the flow control between the two. On most UE implementation, the 3GPP model for the Radio Access Network (RAN) stays inside the Link layer of the TCP/IP stack. Therefore, the implementation is limited to the flow control between the operating system and the LTE firmware, a practice that has already been proposed in the literature, as we reviewed in Section II. For what regards eNB, to manage downlink flows, the infrastructure can be added as an independent application, following the recent trend on Multi-access Edge Computing [11].

As shown by Figure 1, the TC is introduced both on UEs and eNBs. On the UE side, IP packets generated by a local application are sent to TC and enqueued into the (single) qdisc virtually linked to the LTE device. Android devices, having a Linux kernel, already include the traffic control infrastructure, as shown in [1].

On the eNB side, instead, the approach we use to introduce TC is rather distinctive. We think about the TC layer as a standalone application, which receives GTP decapsulated packets and then enqueues them into the qdisc corresponding to their destination UE. Over a single LTE device on the eNB, there are multiple scheduling qdiscs installed, one for each UE attached to the eNB. In the Linux kernel, this is possible by using a classifier qdisc (i.e., multi-queue aware qdisc) that can differentiate the packets based on their Radio Network Temporary Identifier (RNTI), enqueuing them in the appropriate child scheduling qdisc. Each child qdisc manages the packets destined to a particular transmission queue on the device, which is represented by the RLC buffer. We note that our approach exploits and extends the use of in-kernel, already existing, multi-queue aware qdiscs. The proposed method also requires the introduction of a flow control mechanism between the TC and the RLC buffers. In particular, we apply the flow control mechanism between each pair of RLC buffer and its connected qdisc. The flow control involves a soft-stopping mechanism (the qdisc do not move packets to the RLC queue based on the dynamic quota defined by BQL algorithm), as we explained in Section III. We have also implemented the hard-stopping mechanism (the RLC buffer stops the qdisc when it can not enqueue more bytes), but in all our simulations it did not come into play. Indeed, the BQL quota was always less than the RLC buffer size.

We evaluated the proposed approach by using ns-3 simulations. To this purpose, we implemented the necessary changes to allow RLC to notify the traffic control and the BQL library when a packet is enqueued or dequeued, to support flow control and BQL. We have implemented the *wake*, *stop*, *NotifyEnq*, *NotifyDeq* notifications. In our implementation, we assume the cooperation of the LTE device for both UE and eNB. Such an approach has been followed in [6] for the eNB.

If this cooperation is not feasible, for instance in a UE with closed firmware or other issues, implementing this flow control is still possible by exploiting information about the number of bytes enqueued and transmitted by the device firmware, such as in [1].

Finally, it is important to note that the support for BQL allows keeping a backlog in TC regardless of the actual physical RLC buffer size. Also, our approach allows exploring the use of different strategies tailored to the LTE/NR cases.

V. Results

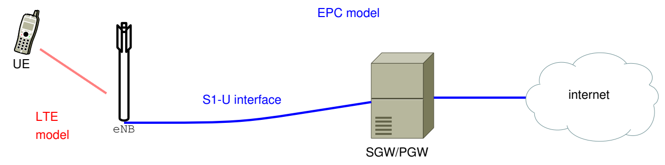


Fig. 2: The network topology used for the validation tests.

A. Simulation Settings

For all the experiments described from now on, the simple LTE topology reported in Figure 2 was used. A number of UEs are attached to the eNB which is connected to the EPC with a point-to-point link having a data rate of 10 Gbps and a delay of 10 ms. The basic idea is to evaluate the effects on the RAN performance due to reduced queueing at the RLC layer and to the introduction of flow-control and traffic-control in LTE.

We evaluate two scenarios. In the first one, a single UE is connected to the eNB; in the second one, ten UEs are connected to the eNB. UEs are in a fixed position. The RAN is configured with 25 Resource Blocks (5 MHz), and the maximum expected throughput of both UL and DL is approximately 16 Mbps in a single UE scenario. The default bearer is configured with RLC in UM mode.

In both scenarios, each UE manages four TCP flows. The basic idea is to saturate the UL and DL paths with two bulk streams (labelled as “bulk UL” and “bulk DL”) while evaluating the performance of two interactive UL and DL small flows (labelled as “small UL” and “small DL”). The bulk streams generate constant traffic load able to saturate the network paths. The small flows are generated by *on-off* traffic sources that create application data at a rate of 500 Kbps in both scenarios. The traffic patterns are defined according to [1], [12], in particular for the number of concurrent TCP connections and the proportion between UL and DL traffic. The TCP version used is New Reno and the TCP segment size is 1400 bytes. The generated traffic is not marked with any IP QoS information, nor distributed through different LTE bearers.

We compare four approaches:

- In the first one, there is no flow control and all the data is queued at the RLC layer. The RLC buffer has a physical size of 500 KB according to [13] and [1].
- In the second approach, we evaluate our proposal. The flow control is regulated by BQL. The traffic control layer is configured with an FQ-CoDel qdisc on the UEs and the eNB. On the eNB, there is one instance of FQ-CoDel for each UE attached. The RLC buffer has a physical size of 500 KB.
- In the third approach, we evaluate the proposal in [6]. The flow control is regulated by DynRLC algorithm. TC is configured with an FQ qdisc on the UEs and the eNB. On the eNB, there is one instance of FQ for each UE attached. The RLC buffer has a physical size of 500 KB.
- In the fourth approach, we evaluate the case of flow control regulated by DynRLC and TC configured with FQ-CoDel. The traffic control layer is configured with an FQ-CoDel qdisc on the UEs and the eNB. On the eNB, there is one instance of FQ-CoDel for each UE attached. The RLC buffer has a physical size of 500 KB.

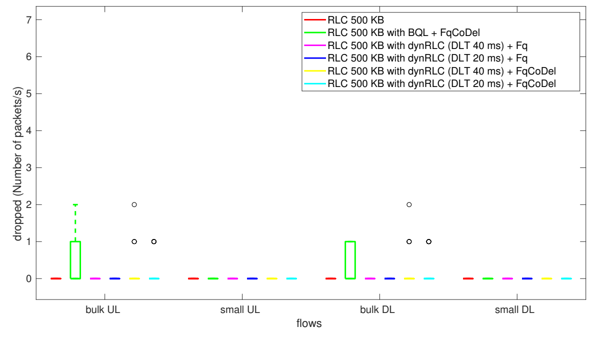
The parameters of BQL and FQ-CoDel are set to their default values. DynRLC is evaluated with two values of DLT defined according to [6].

B. Single UE scenario

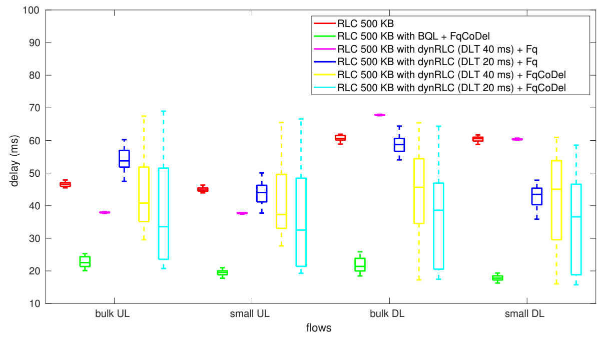
In this scenario, we compare the four approaches listed above in the case only one UE is attached to the eNB. We aim to show that adding flow control allows better backlog management. Indeed, the TC packet scheduler can manage the backlog in order to reduce experienced delay as well as differentiate between small and bulk flows carried by the same bearer.

Results are reported in Figure 3. The presence of an AQM-based queue disc at the traffic-control level causes some packet drops in the bulk UL and bulk DL flows. The dropping is more consistent when the flow control is regulated by BQL (Figure 3a).

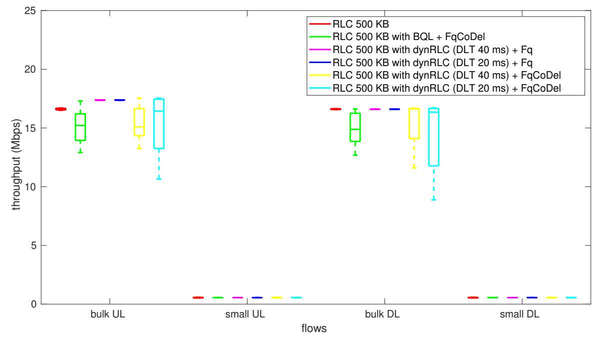
The one-way delay is reported in Figure 3b. In case of flow control regulated by BQL, all the flows have a reduced delay (median values reduced of 50% for the UL flows and of 60% for the DL flows) and FQ-CoDel provides a differentiation among flows with slightly reduced and constant delay for the



(a) Dropped



(b) Delay



(c) Throughput

Fig. 3: Evaluation of the impact of flow control and TC on LTE performance in single UE scenario.

small UL and small DL flows. DynRLC performs as expected in the DL path providing differentiation between bulk DL and small DL flows.However, in case DLT is 40 ms there is an increase in the bulk DL delay. In the UL path, there is differentiation among the flows in case DLT is 20 ms (with an increase in the bulk UL delay) while no differentiation in case DLT is 40 ms. The addition of FQ-CoDel induces further delay reduction among the flows in particular in case DLT is 20 ms. However, the experienced delays are highly variable.

The throughput (Figure 3c) is slightly reduced for the bulk UL and bulk DL when FQ-CoDel is used, according to the performance of AQMs with TCP flows, while the throughput is preserved for the small UL and small DL flows.

C. Multiple UEs scenario

This scenario aims to evaluate the effect of the introduction of TC when multiple UEs compete for the resources. Indeed, the eNB scheduler in this scenario divides the available bandwidth between ten UEs.

The results for the first UE are reported in Figure 4. The other UEs have no significant differences in performance. The presence of TC helps differentiate and manage the different IP flows on the UEs (UL path), and for each UE on the eNB (DL path).

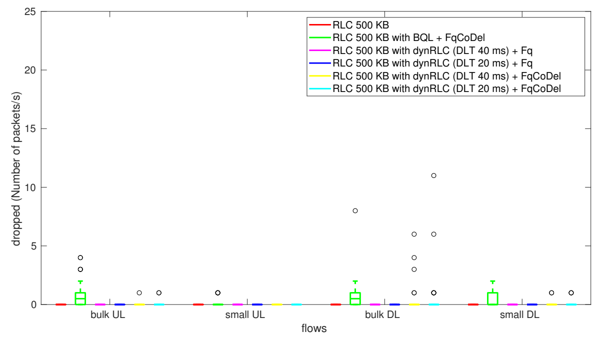
The FQ-CoDel qdiscs on both UE and eNB drop some packets in particular of the bulk UL and bulk DL flows to control their sending rate (Figure 4a). Since the per-UE expected throughput is smaller than in the single UE scenario, the small flows can now contribute significantly to network congestion. Indeed, occasionally dropping occurs in case of small UL and small DL flow, more consistent in case of flow control regulate by BQL, due to FQ-CoDel which try to keep reduced latency.

The one-way delay is reported in Figure 4b. In case of flow control regulated by BQL, all the flows have a reduced delay (median values reduced of 70%), and the qdiscs provide differentiation for the small flows in a similar way as the single UE scenario. We observe that the delay of all the flows for all the UEs is just slightly higher than the value of the single UE scenario. Conversely, without flow-control, the delay is twice higher for the UL flows and almost twice higher for the DL flows compared to those of the single UE scenario. The fact that the flows delay remains rather stable even in presence of a competing UEs is due to the FQ-CoDel scheduling algorithm which manages the queues based on the actual queueing time. DynRLC performs as expected in the DL path providing differentiation between bulk DL and small DL in all cases. The addition of FQ-CoDel induces further improvements and in case DLT 20 ms DynRLC perform as BQL. In the UL path, in this scenario, DynRLC shows worsening of the experienced delay in all cases. Indeed, DynRLC is designed for the DL path based on timeout triggered notifications of the BSR from the UE to eNB. This approach shows worsening of UL performance since the algorithm is not able to cut the UL RLC buffer size.

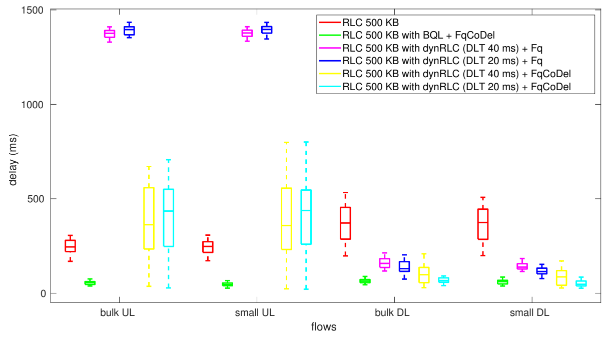
The throughput is reduced for the bulk UL and bulk DL flows in cases FQ-CoDel (a more significative reduction in case DL flows with DynRLC), while the throughput is preserved and more stable for the small UL and small DL flows (Figure 4c).

VI. Conclusion

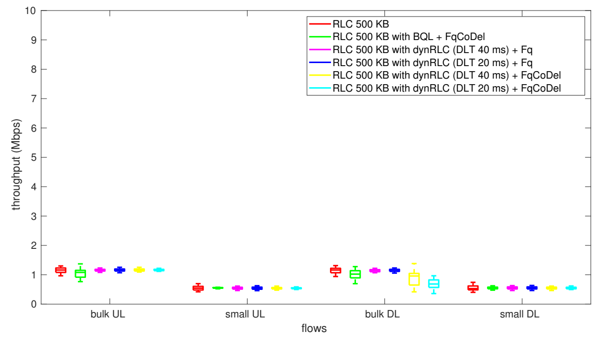
In this paper, we investigate the bufferbloat problem in 3GPP protocol stacks proposing an innovative approach to keep stable and reduced network latency. The approach relieves the network operator from network management to open/close bearers. The idea is to connect the IP TC infrastructure on top of the 3GPP model and define a form of flow control between the layers. For the first time, we evaluate an algorithm, called BQL, as a means to dynamically move



(a) Dropped



(b) Delay



(c) Throughput

Fig. 4: Evaluation of the impact of flow control and TC on LTE performance of one UE in a multiple UEs scenario. Other UEs present very similar results.

the flow control threshold (i.e., the amount of data that can be stored in RLC) in both UE and eNB. We compared the performance of BQL to another algorithm, called DynRLC, designed and presented in literature for dynamically sizing the RLC buffer on the eNB, as a means to reduce the RLC buffer size. We evaluated DynRLC also at the UE side, and tested it for the first time in a multiple UEs scenario.

In our evaluation, performed with the ns-3 simulator, BQL overcomes DynRLC. BQL efficacy leads to a significant performance increase of TC, which can perform packet scheduling on flows directed on the same bearer, preserving small flows over bulk streams.

Future works include the evaluation of the effectiveness of TC with BQL (or any other innovative algorithm to dynamically define a quota of suitable RLC buffer) in more dynamic scenarios, with variable channel conditions as well as handoff between base stations. We also plan to investigate the interoperability of this proposal in existing networks, with different RDS.

Acknowledgment

This work was partially funded by Spanish MINECO grant TEC2017-88373-R (5G-REFINE) and Generalitat de Catalunya grant 2017 SGR 1195.

References

[1] Y. Guo, F. Qian, Q. A. Chen, Z. M. Mao, and S. Sen, "Understanding On-device Bufferbloat for Cellular Upload," in *Proceedings of the 2016 Internet Measurement Conference*, ser. IMC '16. New York, NY, USA: ACM, 2016, pp. 303–317. [Online]. Available: http://doi.acm.org/10.1145/2987443.2987490

[2] P. Imputato and S. Avallone, "An analysis of the impact of network device buffers on packet schedulers through experiments and simulations," *Simulation Modelling Practice and Theory*, vol. 80, pp. 1 – 18, 2018.

[3] T. Høiland-Jørgensen, P. McKenney, D. Taht, J. Gettys, and E. Dumazet, "FlowQueue-CoDel," https://tools.ietf.org/html/rfc8290, January 2018.

[4] G. Piro, N. Baldo, and M. Miozzo, "An LTE module for the Ns-3 Network Simulator," in *Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques*, ser. SIMUTools '11. ICST, Brussels, Belgium, Belgium: ICST, 2011, pp. 415–422. [Online]. Available: http://dl.acm.org/citation.cfm?id=2151054.2151129

[5] P. Imputato and S. Avallone, "Design and Implementation of the Traffic Control Module in Ns-3," in *Proceedings of the Workshop on Ns-3*, ser. WNS3 '16. New York, NY, USA: ACM, 2016, pp. 1–8. [Online]. Available: http://doi.acm.org/10.1145/2915371.2915382

[6] R. Kumar, A. Francini, S. Panwar, and S. Sharma, "Dynamic control of rlc buffer size for latency minimization in mobile ran," in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, April 2018, pp. 1–6.

[7] Y. Dai, V. Wijeratne, Y. Chen, and J. Schormans, "Channel Quality Aware Active Queue Management in Cellular Networks," in *2017 9th Computer Science and Electronic Engineering (CEEC)*, Sept 2017, pp. 183–188.

[8] H. Jiang, Y. Wang, K. Lee, and I. Rhee, "Tackling Bufferbloat in 3g/4g Networks," in *Proceedings of the 2012 Internet Measurement Conference*, ser. IMC '12. New York, NY, USA: ACM, 2012, pp. 329–342. [Online]. Available: http://doi.acm.org/10.1145/2398776.2398810

[9] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, August 1993. [Online]. Available: http://dx.doi.org/10.1109/90.251892

[10] K. Nichols, V. Jacobson, A. McGregor, and J. Iyengar, "Controlled Delay Active Queue Management," IETF, draft-ietf-aqm-codel-07, March 2017.

[11] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing - A key technology towards 5G." ETSI White Paper.

[12] X. Zhang, Y. Xu, H. Hu, Y. Liu, Z. Guo, and Y. Wang, "Profiling Skype video calls: Rate control and video quality," *2012 Proceedings IEEE INFOCOM*, pp. 621–629, 2012.

[13] R. Bestak, P. Godlewski, and P. Martins, "RLC buffer occupancy when using a TCP connection over UMTS," in *The 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, vol. 3, Sept 2002, pp. 1161–1165 vol.3.