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

Rajeev Kumar†, Andrea Francini*, Shivendra Panwar†, and Sameerkumar Sharma*
† Department of Electrical and Computer Engineering, New York University, Brooklyn (NY), USA
* Autonomic Networks and Mobile Services Research Department, Nokia Bell Labs, Murray Hill (NJ), USA

***Abstract*—5G is expected to expand the set of interactive applications enabled by the mobile network. Low end-to-end packet latency is a fundamental requirement for proper operation of those applications, but may be compromised in packet buffers shared with elastic applications that use greedy TCP sources for end-to-end transport. The accumulation of queuing delay by the elastic application degrades the latency for the interactive application, however light the throughput of the latter may be (as is the case with online gaming and over-the top voice). Buffer sharing is unavoidable in the RLC layer of the 3GPP RAN stack. To minimize its negative effect on interactive applications, we split the buffering between the RLC and PDCP layers. Then we equip the PDCP buffer with per-flow queues, and apply to the RLC buffer a new dynamic sizing mechanism that enforces the lowest queuing delay that is compatible with the existing configuration of the RLC connection. On a cellular link shared with a greedy TCP flow, our dynamic sizing solution can reduce the queuing delay of PING packets by up to two orders of magnitude compared to the default configuration with an RLC-only buffer of fixed size.**

***Index Terms*—5G; Latency; Buffer management; RLC; RAN**

I. Introduction

The design of the fifth-generation (5G) wireless network addresses the diverse requirements of a broad set of new applications, further extending the prominent role of data communications within society and industry at large. Low and ultra-low packet latency are key enablers for many of those applications, especially when they include interactivity features [1], [2]. For example, applications like autonomous driving, augmented reality, and virtual reality require end-to-end packet latencies in the 1-10 ms range. Online gaming, multi-party voice/video calls, and remotely controlled mechanical devices require latencies in the 10-50 ms range [3].

Meeting the stringent requirements of interactive applications is particularly challenging in a wireless network, due to the variability of the wireless link bandwidth. Every drop in link bandwidth increases the transmission time of individual packets, and therefore the latency of all packets that are already queued in the link buffer or still in transit between the data source and the buffer. The buffer of a wireless link brings two types of contributions to the end-to-end packet latency of an application flow. *Endogenous delay* results from the accumulation of packets of the application flow in the link buffer when the link bandwidth is lower than the transmission rate of the application server. *Exogenous delay* occurs when packets of the application flow find the link buffer loaded with packets of other applications. Endogenous delay may grow large for applications that try to maximize their utilization of the data path bandwidth, typically over TCP transport. Exogenous delay is particularly harmful to low-bandwidth interactive applications, as it inflates their latencies despite the negligible load that they impose on the network.

In this paper we focus on the minimization of exogenous delay for an interactive flow that shares a mobile wireless link (radio bearer) with at least one greedy TCP flow. This scenario is common when a single mobile device (UE) provides cellular access to multiple terminals (mobile hotspot use case), but is also experienced on a smartphone or tablet that runs an over-the-top (OTT) conversational application concurrently with a large file download. The standard quality-of-service (QoS) framework of 3GPP mobile networks allows the isolation and prioritization of application flows by mapping them onto dedicated bearers with different QoS Class Identifiers (QCIs). However, the diversification of the mapping is only limited to applications that are managed by the service provider, such as Voice over Long-Term Evolution (VoLTE), whereas OTT applications typically end up sharing the bandwidth and buffer of a common (default) bearer.

Intra-bearer flow queuing can be an effective solution: associating flows with dedicated queues makes the exogenous delay depend only on the number of active flows within the bearer, and not on the total number of packets that are stored in the bearer buffer. This is true even when the queue scheduler of the bearer is completely unaware of the latency requirements of the flows it serves, for example because the mapping of flows onto queues is purely stochastic and not controlled by any form of signaling or provisioning [4]. Incidentally we argue that flow queuing is also an enabler for the minimization of endogenous delay, because the queue scheduler enforces fairness among the active flows independently of the behavior of the respective sources. Therefore each source can be designed around the specific needs of its application without being constrained by the ruling TCP framework for distributed congestion control and fairness enforcement.

Unfortunately the implementation of flow queuing is not possible where it would be most efficient. In the 3GPP mobile radio access network (RAN) stack, the radio link control (RLC) layer is the natural venue for instrumenting the bearer buffer because of the retransmission capabilities of its acknowledged mode (RLC-AM) [5]. The buffer is needed to hold data units that are waiting for acknowledgment of their successful transmission. To avoid wasting precious radio resources, the protocol data units (PDUs) in the RLC retransmission buffer do not always map one-to-one with the original IP packets. The payload of an RLC PDU may belong to one IP packet or to multiple IP packets, possibly from different IP flows. The latter case precludes the realization

of flow queuing in the buffer of any RLC instance that tries to saturate the payload of every PDU, making the packet data convergence protocol (PDCP) layer the first feasible option.

We consider a per-bearer arrangement where a flow-queuing PDCP buffer is flow-controlled by a shared-queue RLC buffer, and study solutions for minimizing the accumulation of exogenous delay in the RLC buffer. We evaluate the solutions on the OpenAirInterface (OAI) platform [6], observing that the exogenous delay can be effectively controlled by dynamic adjustment of the RLC buffer size. The substantial latency reduction that we obtain for low-bandwidth interactive applications has no negative effect on the throughput of the elastic applications that share the same radio bearer.

This paper is organized as follows. Section II presents latency measurements from the default bearer configuration in OAI, where all buffering occurs in the RLC layer, and from a combination of PDCP flow queues with a smaller, but static, RLC buffer. We describe our new algorithm for dynamic sizing of the RLC buffer in Section III. We study the effects of RAN configuration parameters and bearer bandwidth in Sections IV and V, respectively. Section VI concludes the paper.

II. Bearer Buffer: RLC-Only and PDCP Flow Queues

On a generic link, flow queuing isolates the traffic flows of concurrent applications because it ensures that the distribution of link bandwidth depends on the number of flows with queued packets and not on the rate and pattern of their packet arrivals. When the link is congested, each flow has its packets stored in a dedicated queue and a scheduler serves the busy queues based on the preferred fairness criterion.

Stochastic fairness queuing (SFQ) is a simple way of running a flow queuing arrangement without a priori knowledge of the nature and requirements of the application traffic to be handled [4]. In SFQ every incoming packet is mapped onto a queue by hashing the fields of its header that are unique to the respective flow. This way all packets of a flow end up in the same queue. A round-robin scheduler visits the busy queues with even frequency for extraction and then transmission of their packets. At every visit, the scheduler may extract a fixed number of queued packets (irrespective of their sizes) or a fixed number of bytes (for exact throughput fairness). The use of a hash function eliminates the need for provisioning a packet classification rule for every new flow that is established, and for removing it when the flow is torn down. Hash collisions where packets of distinct flows end up in the same queue are always possible, but their likelihood can be limited by proper sizing of the hash value range.

Flow queuing eliminates the need to carefully size the overall buffer space based on the prevalent bandwidth-delay product (BDP) of the flows that share the buffer. As opposed to single-queue buffers, flow-queue buffers can be sized liberally without risking excessive exogenous delay. When stored packets have exhausted the buffer space allocated to the flow queues and a new packet arrives, flow isolation is best enforced if a packet is removed from the longest queue in the set [7].

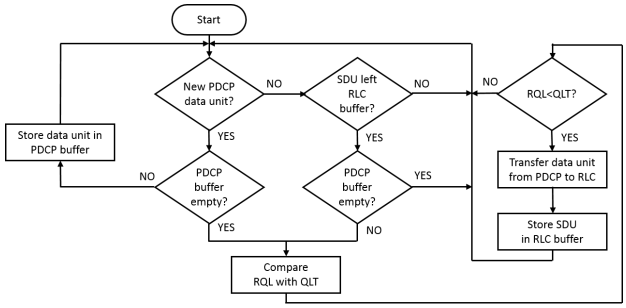
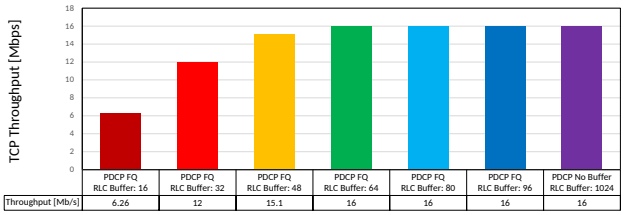


Fig. 1: Interaction between PDCP flow queues and RLC buffer.

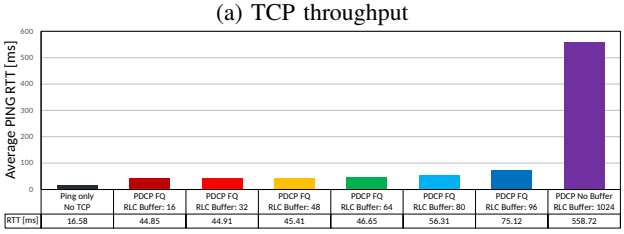
In the mobile RAN protocol stack, it would be ideal for the minimization of queuing delay and end-to-end latency if there was only one queuing stage, and if that stage could be arranged with flow queues. However, efficient use of the wireless link bandwidth imposes the instantiation of a single FIFO queue per radio bearer in the RLC layer, so flow queuing can only be implemented in a higher layer, best if adjacent to RLC. A bearer that pursues the minimization of exogenous queuing delay should therefore combine the shared FIFO queue in the RLC layer with a set of flow queues in the PDCP layer.

A meaningful measure of network latency for interactive applications is the time between the transmission of a request message by the client and the arrival to the client of the last byte of the server response. The PING round-trip time (RTT) is a reliable tool for assessing the minimization of exogenous queuing delay in the context of that measure. On the one hand, it captures well the properties of the bi-directional data path between client and server that contribute to network latency equally for all applications (propagation latency and exogenous queuing delay). On the other hand, it is practically transparent to the endogenous queuing delay and to the application-specific handling of data-path bandwidth (since the request and response messages consist of small single packets). We use the *iperf* tool to instantiate the persistent TCP flow from network server to mobile client that causes the accumulation of exogenous delay in the bearer buffer.

The default configuration of RLC-AM in OAI has a buffer space of 1024 service data units (SDUs) and no packet buffer in the PDCP layer (one RLC-AM SDU encapsulates one IP packet). We extend OAI with a second configuration where the RLC-AM buffer size is smaller but fixed (ranging from 16 to 96 SDUs in 16-unit increments) and a set of PDCP flow queues absorbs extra packets when the lossless RLC-AM buffer is saturated. The opportunity of transferring one or more packets from the PDCP buffer to the RLC-AM buffer is evaluated every time an SDU leaves the RLC-AM buffer. The round-robin scheduler of the PDCP flow queues extracts a single packet, whatever its size, per queue visit. The flow chart of Fig. 1 shows the conditions that control the transfer of data units from the PDCP flow queues to the RLC buffer in the OAI extension. The final decision is based on the comparison of the current *RLC queue length* (*RQL*) with the *queue length threshold* (*QLT*) that sets the buffer size. The algorithm is the same also for the unacknowledged mode of RLC (RLC-UM).



(a) TCP throughput



(b) PING RTT

Fig. 2: Buffer configuration effect on TCP throughput and PING RTT.

We run all experiments on a portable LTE testbed that includes one Ettus B210 Universal Soft Radio Platform (USRP) as the radio head of the LTE radio access network (RAN) and five Intel i7 Next Units of Computing (NUCs) with the following roles: application/content cloud (one NUC), hosting the application servers; central cloud (one NUC), hosting the LTE Evolved Packet Core (EPC); mobile edge cloud (one NUC), hosting the LTE radio access network (RAN); and two end-user devices, hosting the application clients (one NUC each, with respective USB-dongle LTE adapter) [8]. The baseline configuration of the radio interface yields a data throughput of about 16 Mb/s when a single UE is active.

Figure 2 shows the *iperf* TCP throughput and PING RTT measured when the two applications run concurrently over the same LTE bearer. The results yield the following observations: (i) the minimum PING RTT achievable is in the 15 ms range; (ii) the default configuration with large RLC buffer and no PDCP buffer adds 540 ms of average queuing delay, which is not acceptable for interactive applications; (iii) flow queuing in PDCP lowers the RTT drastically; and (iv) the choice of size for the RLC-AM buffer is critical (the TCP throughput suffers if the buffer is too small and interactivity may be compromised if the buffer is too large). There appears to be an optimal RLC buffer size that reconciles full TCP throughput with low PING RTT (between 48 and 64 SDUs in this particular experiment). Such an optimal size clearly depends on the bandwidth available to the radio bearer and should be determined automatically, as described next.

III. Dynamic Sizing of the RLC-AM Buffer

When the PDCP layer hands a data unit to the RLC layer, the latter encapsulates it as an RLC SDU and stores it in the RLC SDU buffer. It also copies segments of the SDU into subsequent RLC PDUs, so that the payload of every PDU is fully utilized (at least as long as the PDCP layer keeps supplying new data units). As a result, one RLC SDU may be fully contained in one RLC PDU or spread over multiple contiguous PDUs. The payload of one PDU may come from a single SDU or from multiple SDUs.

RLC PDUs are stored in a dedicated buffer. When the medium access control (MAC) layer finds a transmission opportunity for the bearer, it fulfills it with an adequate number of bytes from RLC PDUs in the RLC PDU buffer. In RLC-UM, the MAC transfer of the last byte of a PDU triggers the removal of the PDU from the buffer. In RLC-AM, instead, the removal occurs only after the RLC-AM transmitter receives a *status PDU* that acknowledges the successful delivery of the *data PDU* across the wireless link. In both cases, an RLC SDU is removed from the SDU buffer only after all the PDUs into which it was segmented have left the PDU buffer.

Now we describe our algorithm for dynamic control of the buffer size QLT of Fig. 1. We aim at minimizing the queuing delay in the RLC buffer while preserving the throughput of an oversized static buffer. As detailed in the pseudocode of Algorithm 1, the general idea is to use the queuing delay of the RLC SDU buffer to modulate QLT. We call *queuing delay* the time that the SDU spends in the RLC SDU buffer. Every sample is computed as the difference between the time of removal of the SDU from the buffer and the time of its arrival. The definition implies that an arrival timestamp is associated with every SDU in the buffer. We note that the queuing delay tends to be larger in RLC-AM than RLC-UM, because RLC-AM adds the time needed for arrival of the status PDU that acknowledges the successful delivery of the data PDU.

Algorithm 1: Sizing of RLC buffer based on SDU delay

```
1:  QLT = QLT_max, t_prev = t_now, T = 200 ms
2:  while T > 0 do
3:    if RLC SDU removed from buffer then
4:      Collect delay sample for removed SDU
5:      Use new delay sample to update DLM
6:      if t_now > (t_prev + T) then
7:        if DLM < DLT then
8:          QLT = QLT + α · (QLT_max − QLT)
9:        else
10:         QLT = QLT − α · (QLT − QLT_min)
11:       end if
12:       t_prev = t_now
13:     end if
14:   end if
15: end while
```

When the chosen *delay metric* DLM is smaller than the desired *delay threshold* DLT, QLT grows by a fraction α of its distance from the maximum value allowed QLT_{max}. When it is larger, it drops by the same fraction of its distance from the minimum value allowed QLT_{min}. QLT_{min} and QLT_{max} depend on the minimum and maximum throughput expected for the radio bearer. We impose a minimum time interval $T = 200$ ms between consecutive QLT updates to ensure that the effects of a QLT adjustment are well captured by the measurement that drives the next adjustment (t_{now} is the current time, t_{prev} is the time of the previous QLT update). DLM is a function of the delay samples collected from the removed SDUs. Here we consider the following four options for the definition of DLM: (i) minimum, (ii) maximum, and (iii) average of the samples collected since the previous QLT update (called respectively the *interval minimum* τ_{min}, the *interval maximum* τ_{max}, and

the *interval average* $\overline{\tau}$), and (iv) exponential weighted moving average of all delay samples ($EWMA(\tau)$).

Next we compare the four candidates for the definition of *DLM* with respect to their effectiveness in driving the control of the RLC buffer size. We repeat the single-bearer experiment of Fig. 3 for each candidate and for each of seven fixed values of *DLT* (ranging from 30 ms to 60 ms in 5 ms increments). We plot the TCP throughput results in Fig. 3(a) and the delay results (both SDU queuing delay and PING RTT averages) in Fig. 3(b). Then we compare in Fig. 3(c) the average SDU delay obtained by each *DLM* candidate with the smallest *DLT* value that yields full throughput (16 Mb/s). Figure 4 shows the cumulative distribution function (CDF) of the PING RTT samples collected in the same experiments of Fig. 3(c). We conclude that the interval average $\overline{\tau}$ is the best *DLM* candidate based on the following elements:

- The interval average $\overline{\tau}$ achieves the lowest SDU delay at full throughput, with an improvement of almost 9 ms versus the runner-up.
- The interval average $\overline{\tau}$ is the only metric that maintains a consistent relationship between the objective (the threshold *DLT*) and the output (the average of the SDU delay samples) of the buffer size control. This is testified by the linearity of the SDU delay plot in Fig. 3(b) and by the tight concentration of the PING RTT distribution in Fig. 4. This outcome is not surprising because of the similarity of definition between the *DLM* candidate and the control-loop output, and because the link conditions are stationary throughout the experiment. In any case it confirms that the control works as expected under the (favorable) conditions of the experiment, as opposed to the other *DLM* candidates.
- The interval minimum τ_{min} and maximum τ_{max} yield unpredictable outcomes because they are dominated by statistical outliers. This is especially true for τ_{min} (Fig. 4).
- Through proper configuration of the coefficient that defines the degree of weighting decrease, $EWMA(\tau)$ could in theory approximate the performance of the interval average. However, in order to maintain a stable memory depth as the throughput of the bearer fluctuates, the coefficient would also have to be automatically controlled, adding complexity to the overall solution.

IV. Effect of Configuration Parameters

The lowest value of *DLT* that yields maximum throughput for the interval average $\overline{\tau}$ is the one that the buffer size control should use for minimization of the SDU delay. However, it is not obvious what it should be. Indeed, the value of 40 ms in Fig. 3(a) is determined by the combination of several configuration parameters from different layers of the RAN stack, so the optimal value changes with the set of parameters.

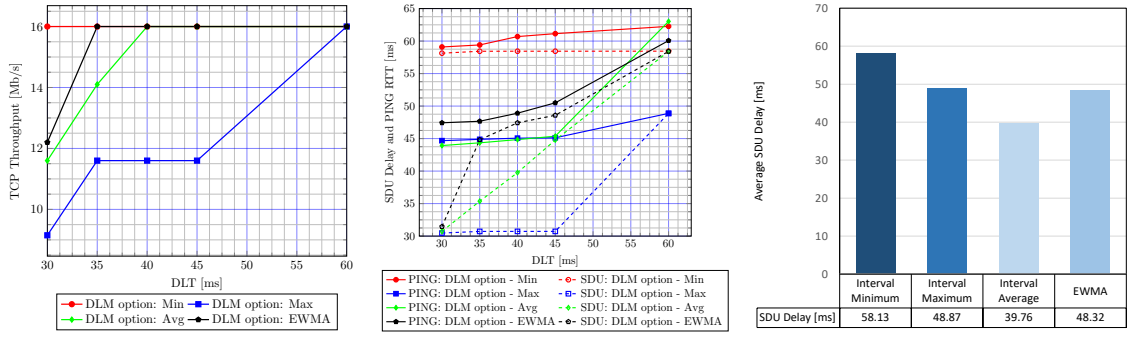
The optimal *DLT* value must be compatible with the constraints that the RLC mode of operation (UM or AM) imposes on the removal of SDUs from the SDU buffer. We focus on RLC-AM because it is the typical choice for RLC configuration of the default bearer. RLC-AM has a timer called *tStatusProhibit* [9] that enforces a minimum time between consecutive transmissions of status PDUs from the RLC receiver to the transmitter. The duration γ of the timer is part of the overall bearer configuration and ranges from 0 ms (no gap enforced) to hundreds of milliseconds. Lower values mean lower latency for downlink data, but also lower throughput for uplink data, because the transmission of uplink data PDUs is more frequently preempted by status PDUs acknowledging downlink data.

When the *tStatusProhibit* timer expires, the RLC receiver must request a grant for uplink transmission of the status PDU, unless a grant already exists for transmission of other application data or TCP ACKs. New grant requests can only be transmitted at fixed time intervals defined by a *scheduling request periodicity* (SRP) parameter that in LTE ranges from 1 to 80 ms [10]. After receiving the request the uplink scheduler must find and reserve a transmission time for the status PDU.

Figure 5 shows a typical pattern of SDU departures from the RLC-AM buffer, where blocks of consecutive SDUs (about 40 in this particular case) are removed from the buffer at fixed time intervals (here the period is 30 ms). The arrival of a status PDU triggers each block removal. In the example, γ is set at 25 ms; the MAC scheduling protocol is likely responsible for the majority of the extra 5 ms. Over the course of an active connection we have observed occasionally larger gaps between burst transmissions, never shorter ones.

DLT must not only be larger than γ, but also include extra room to account for the variability of the SRP effect on the gap between consecutive transmissions of status PDUs. Also *DLT* applies to the queuing delay of RLC SDUs, not just the inter-arrival time of status PDUs. An SDU that misses its first opportunity for buffer removal must wait for an entire new inter-PDU period before it receives the next opportunity. As a consequence, even when optimized the average SDU queuing delay tends to be larger than the average inter-arrival time of the status PDUs. We conclude that, rather than derive *DLT* from the configured values of γ and SRP, it is simpler for the RLC transmitter to maintain a moving average of the time gap Δ_{sPDU} between consecutive arrivals of status PDUs, and then apply a multiplication factor β to the average: $DLT = \beta \cdot EWMA(\Delta_{sPDU})$. The initial value of the EWMA should be set at least as large as $\gamma + SRP$, whether the values of the two parameters are known a priori or just estimated.

Figure 6 shows the dependency of the TCP throughput and average PING RTT on the multiplication factor β (ranging from 1 to 2.5 in 0.25 increments) and on γ (from 0 ms to 25 ms in 5 ms increments). The effect of γ on the delay-throughput trade-off is evident: both metrics grow with it. Overall, a good performance compromise can be obtained with $\beta = 1.75$ and $\gamma = 10$ ms, with throughput at 15.4 Mb/s (only 3.8% below maximum) and average PING RTT at 31 ms (only 38% higher than the minimum achievable with no time restrictions on the transmission of consecutive status PDUs ($\gamma = 0$ ms)). With the γ value of the default configuration (25 ms), a similar throughput of 15.5 Mb/s is achieved with $\beta = 1.25$, together with an average PING RTT of 46 ms (about 100% higher than



(a) TCP throughput (b) SDU delay and PING RTT (c) Average SDU delay at full throughput

Fig. 3: Effect of *DLM* candidates on throughput and latency.

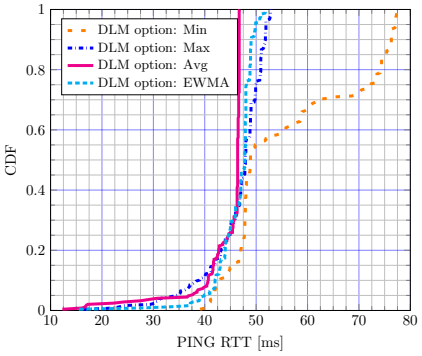


Fig. 4: CDF of PING RTT with lowest full-throughput DLT.

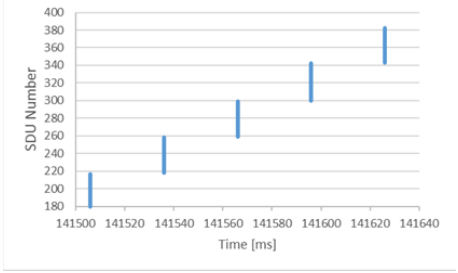
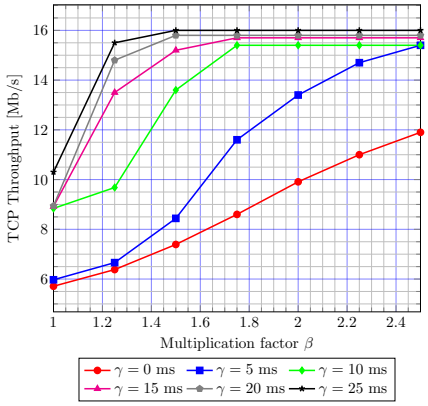
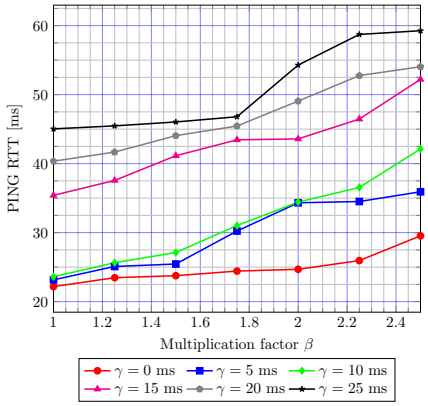


Fig. 5: SDU departures from RLC-AM buffer.



(a) TCP throughput



(b) Average PING RTT

Fig. 6: Effect of β and γ on throughput and latency.

the minimum measured). These results confirm that our control algorithm matches the best performance that we previously achieved by manually fine-tuning the size of the RLC-AM buffer (see Fig. 2) and indicate that it is possible to further reduce the latency of select radio bearers by proper configuration of other RAN stack parameters. This capability can be instantiated in 5G networks by assigning the select radio bearers to network slices that support low-latency services and applications [8]. Slices with milder latency requirements can retain the default bearer configuration (both air interface and RLC buffer) for higher spectral efficiency.

V. Sensitivity to Bearer Bandwidth

Our scheme for dynamic sizing of the RLC buffer should enforce the same queuing delay irrespective of the bandwidth that is available to the radio bearer, at least when that bandwidth is stationary. We design our last experiment to assess compliance with this requirement.

We modulate the bandwidth of the bearer optimized with dynamic RLC-AM buffer sizing by attaching a second UE to the same LTE cell. The second UE runs the default RLC-AM configuration and downloads UDP traffic from a source that transmits at a fixed rate, ranging from 0 to 12 Mb/s in subsequent runs of the experiment. Assuming that the total cell capacity is fixed (we expect it to be close to 16 Mb/s), the bandwidth of the first bearer should drop from 16 Mb/s to about 8 Mb/s as the downlink throughput of the second UE grows from 0 to 8 Mb/s. As the transmission rate of the UDP source approaches and then exceeds 8 Mb/s, the actual bandwidth split between the two bearers is controlled by the MAC scheduler and depends on the respective channel conditions, which we try to equalize. Both UEs are stationary.

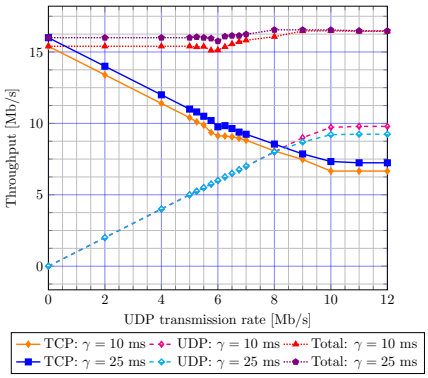
To limit the number of data points, we only look at two

values of the *tStatusProhibit* timer duration γ: 10 and 25 ms. As usual, PING traffic shares the optimized bearer with TCP traffic. Figure 7(a) plots the TCP throughput on the optimized bearer, the UDP throughput on the second bearer, and the total of the two. Figure 7(b) plots the average SDU delay and PING RTT measured on the optimized bearer. The results provide the following insights:

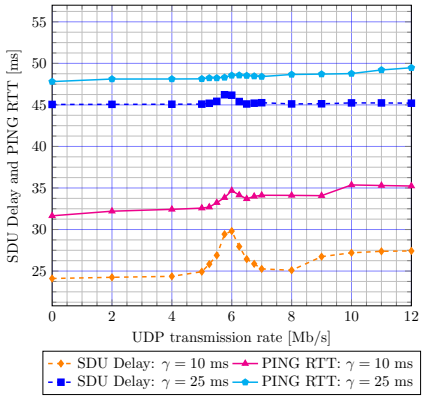
- The smaller γ lowers the TCP (and total) throughput because the increased volume of the status PDU traffic delays the uplink transmission of TCP ACKs (at 8 Mb/s, we estimate the ratio between status PDUs and TCP ACKs at about 1:3.33 with γ = 10 ms and about 1:8.33 with γ = 25 ms). However the throughput drop is practically negligible compared to the extent of the latency improvement (ranging from 10 to 20 ms in Fig. 7(b)).
- The average SDU delay is generally insensitive to the bearer bandwidth when γ = 25 ms. Instead with γ = 10 ms it grows by about 4 ms going from one end to the other of the bearer bandwidth range. We conjecture that the reduced frequency of TCP ACK transmissions increases the uplink scheduling delay of the status PDUs and therefore the value of DLT. When the ACKs are most frequent (on the left-hand side of the plot), it is more likely for a status PDU to grab a scheduling grant originally requested for a queued TCP ACK and thus mitigate the SRP effect. The likelihood of such events drops linearly with the frequency of the TCP ACKs. Indeed, in separate experiments where the optimized bearer downloads UDP traffic and no ACKs are transmitted upstream (not plotted here), the SDU delay is larger but also insensitive to the bearer bandwidth.
- The TCP throughput shows a a small dip when the UDP source of the second bearer transmits data at 6 Mb/s. Additional data points obtained with small variations of the UDP rate around 6 Mb/s indicate that the dip is not accidental. We also detect a local peak in the time gap between consecutive arrivals of status PDUs. We do not have a verified explanation for this odd behavior, but it could be induced by a local dip in the frequency of transfer of uplink scheduling grants from TCP ACKs to status PDUs, due to the particular timing alignment between ACK and status-PDU generation. No such effect occurs with UDP traffic on the optimized bearer.

VI. Conclusion

Low end-to-end latency is vital to the reliable operation of over-the-top (OTT) interactive applications like online gaming and video telephony, but is easily compromised in cellular networks when flows from greedy applications occupy the same wireless link (radio bearer). We designed a scheme for automatic control of the RLC buffer size in the mobile RAN that works together with a flow-queuing arrangement in the PDCP layer to minimize the latency degradation induced by link sharing. Our design yields the lowest latency that is attainable without degrading the overall link throughput. We prototyped our solution in the OpenAirInterface (OAI) RAN stack and tested it on a complete LTE network. Compared to the default OAI configuration of PDCP and RLC-AM, our solution drops the queuing delay overhead incurred by PING packets from 540 ms to 15 ms (-97%) when they share a 16 Mb/s radio bearer with a TCP flow. Since our dynamic solution keeps the latency practically insensitive to the bearer bandwidth, the improvement over any static configuration of the RLC buffer size, including those with PDCP flow queues, grows larger when the bearer bandwidth is lower.



(a) TCP, UDP, and total throughput



(b) SDU delay and PING RTT

Fig. 7: Sensitivity to bearer bandwidth.

References

[1] 5G-PPP, "5G empowering vertical industries," Tech. Rep., Feb. 2015.

[2] NGMN Alliance, "NGMN 5G initiative white paper," https://www.ngmn.org/uploads/media/NGMN_5G_White_Paper_V1_0.pdf, Tech. Rep., February 2015.

[3] C. Westphal, "Challenges in networking to support augmented reality and virtual reality," in *IETF98 - ICNRG meeting*, March 2017.

[4] P. E. McKenney, "Stochastic fairness queueing," in *Proc. of IEEE INFOCOM*, June 1990.

[5] *LTE: Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Link Control (RLC); Protocol specification*, 3GPP Std. 3GPP TS 36.322 version 13.3.0 Release 13, March 2017.

[6] N. Nikaein *et al.*, "Demo: Openairinterface: An open LTE network in a PC," in *Proc. of ACM MobiCom*, September 2014.

[7] B. Suter *et al.*, "Design considerations for supporting TCP with per-flow queueing," in *Proc. of IEEE INFOCOM*, March 1998.

[8] S. Sharma, R. Miller, and A. Francini, "A cloud-native approach to 5G network slicing," *IEEE Commun. Mag., 55(8)*, August 2017.

[9] *LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Resource Control (RRC); Protocol specification*, 3GPP Std. 3GPP TS 36.331 version 13.6.1 Release 13, July 2017.

[10] *LTE: Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures*, 3GPP Std. 3GPP TS 36.213 version 14.3.0 Release 14, June 2017.