Fast Re-Tx: Mobile Latency Optimization with Cross Layer Analysis

Haokun Luo
University of Michigan
{haokun}@umich.edu

ABSTRACT

Bad network performance could hurt user experience and downgrade application reputation. The discontinuous burst of short data transmission is the most common traffic pattern in the cellular network. That implies that the devices are frequently switching between idle state and data transmission state. Based on our observations, significant latency appears during the initial period of the data transmission. To identify the root cause of the latency problem, we use diagnostic cross layer monitoring tool, *Qualcomm eXtensible Diagnostic Monitor* (QxDM), to enable the visibility of detailed data link layer (i.e. *Radio Link Control* (RLC) layer) data transmission information. We also designed a novel cross-layer mapping mechanism to correlate both transport layer behavior with data link layer, and found that RLC layer protocol's inactive response to packet loss leads to unnecessary delays in the both layers. We propose a RLC *Fast Re-Tx (Retransmission)* mechanism to avoid sluggish reaction to packet loss, and further reduce the latency during the initial network connection. Based on our real trace analysis, the *Fast Re-Tx* mechanism could reduce the latency by up to *35.69%* over initial network connection.

General Terms

Mobile, Measurement

Keywords

RRC state machine, Cross Layer Analysis, RLC Fast Retransmission, Latency Optmization

1. INTRODUCTION

3G cellular data networks have recently witnessed rapid growth, especially due to the emergence of smartphones. In this paper, we focus on the *Universal Mobile Telecommunications System* (UMTS) 3G network, which is among the most popular 3G mobile communication technologies. However, the bottleneck of the internet resides in the first hop of the network, and large amount of lower layer retransmission could cause significant latency [11]. 3G network has a bad reputation at the initial period of network connection [7], and significant delay will hurt user experience badly. In this paper, we are going to allocate the fundamental cause of the abnormal latency, and propose a modified protocol solution.

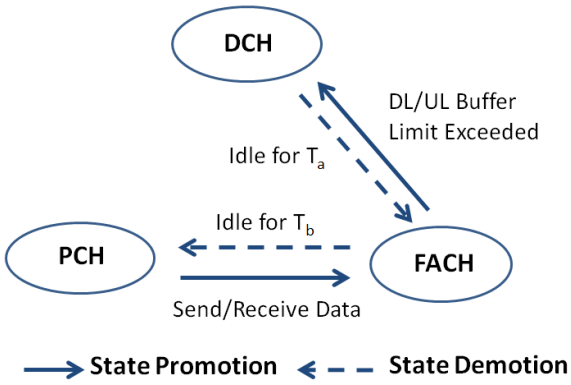


Figure 1: RRC State Machine for the 3G UMTS network for T-Mobile

3G network systems operate under radio resource constraints. To efficiently utilize the limited radio resources, UMTS introduces for each *user equipment* (UE, i.e. smartphone) a *radio resource control* (RRC) state machine, such as in Figure 1, that determines radio resource usage affecting device energy consumption and user experience [5]. UE will stay in one of three states, each with different amount of allocated radio resources. The transitions between states also have significant impact on the UMTS system.

The network topology provides a nature isolation between different layers. The design of transport layer protocol doesn't require the knowledge of lower layer information. However, the abstraction of the design could lead to sub-optimal scenario, i.e. large significant latency will occur during the RRC state transition. Since the root cause of the abnormal delay behavior resides in the data link layer (i.e. RLC, radio link control, layer), the visibility of lower layer information will help us identify the root cause of the bizarre behavior in upper layer.

Previous studies [9, 12, 10, 8] don't have accessibility to detailed link layer data transmission information, so they have to treat the lower layer as a black box, and use inference technique to determine and control RRC states with less

accuracy. In our study, the QxDM tool is able to capture the fine grained data transmission and context information (i.e. signal strength, RLC protocol configuration) in both upper and lower layers. To the best of our knowledge, our study is the first cross-layer analysis that have both ground truth knowledge in transport layer and data link layer. Thanks to our cross-layer visibility, we identify the root cause of the unexpected latency as the RLC protocol's lagging response to the packet loss signal. We propose *Fast Re-Tx* as the improved RLC layer mechanism to reduce the latency over the initial data transmission. We summarize our contributes here:

- **Novel cross-layer mapping mechanism to fully correlate multiple network layers.** We develop the first cross-layer mapping algorithm that accurately maps the TCP/UDP packets to the corresponding *packet data unit* (PDU) in the RLC layer. Our mapping accuracy is on average 99.8%.
- **Root cause analysis on the performance issue and propose feasible solutions.** Based on the cross-layer mechanism, we could perform an in-depth analysis on RLC layer behavior during the initial data transmission period. We found that the RLC sender doesn't reactive to PDUs loss signal — duplicate *acknowledgement* (ACK) PDUs, and it leads to upper layer packet delays, and even TCP *retransmission timeout* (RTO). Therefore, we propose RLC *Fast Re-Tx* to actively respond to duplicate RLC ACK PDUs to reduce the latency over problematic initial data transmission period. We evaluate the improved mechanism using real time traces, and estimate the RLC *round-trip time* (RTT) could drop up to 35.69% over FACH state.

2. BACKGROUND

Mobile cellular networks, including 3G UMTS [1], use the Radio Resource Control (RRC) protocol to allocate resources to mobile UEs. Handsets transition between different RRC states, which vary in power consumption and bandwidth, and an individual RRC state machine is maintained for each handset. Transitions between different states occur due to traffic patterns between the UE and base station — in general, more traffic will cause a higher-power and higher-bandwidth state to be entered. The RRC protocol for these network types has been defined by 3GPP [5].

In 3G UMTS, there are three main states: DCH, which is high-power and high-bandwidth, FACH, which is low power and low bandwidth, and PCH, where no transmission is possible. If a higher-bandwidth state is needed, there is a promotion delay. Some carriers may always go directly from PCH to DCH. An example RRC state machine can be seen in Figure 1. These terms are summarized in Table 1.

The UE will initially stay in PCH without data transmission. Then any data transmission will trigger the UE to promote from PCH to FACH, and even further to DCH if the amount of transmitted data exceeds a threshold. If UE doesn't transmit any data in DCH for T_a (around 3s), it will demote to FACH state. Similarly the FACH to PCH demotion timer T_b is around 2s [9]. FACH state is a required RRC transition state as long as there is data communication over the air. Therefore, the initial data transmission delay occurs when the UE stays in FACH state. To explicitly allocate the root cause of latency issue in FACH state, we break down transitions between states into several substates in Figure 2. The PCH_STABLE substate doesn't exist since the UE cannot stay in PCH during the PCH to FACH promotion.

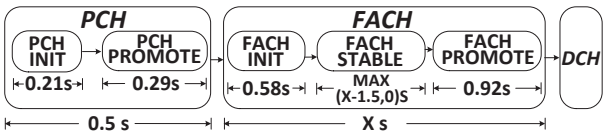


Figure 2: RRC substate definitions from observed RLC-layer behavior during state promotions. We examine these individually through QxDM to investigate root causes of latency.

RLC layer protocol is an *automatic repeat request* (ARQ) fragmentation protocol used over a cellular air interface. Data in RLC layer is transmitted in terms of PDU, and the maximum PDU size is 42 bytes in UMTS network [4]. Thus, any upper layer packet with size greater than one PDU size will be fragmented into multiple PDUs. That is the reason why one TCP or UDP packet could mapped to a series of PDUs in §4.2.

3. OBSERVATION

Using the RRC inference methodology in previous study [9], we have found that unexpected data transmission latency occurs during the FACH state. A motivating example of the problem can be seen in Figure 3. 22 sets of RRC state measurements can be seen in the top graph, including results for both large (1 KB) and empty UDP packets. For this test, DCH is induced by sending a large packet, then another packet is sent after a time interval — this interval is shown on the X-axis. The two packet sizes allow us to observe FACH, which is characterized by low latencies for small packets and higher latencies for large ones.

It is clear that the DCH state demotion timer is 2.5 to 3 seconds long, and the FACH timer is an additional three seconds, after which there is a transition to PCH. DCH is characterized by low RTTs, as there is no promotion delay. PCH has much higher RTTs, due to the promotion delay, and FACH has a lower RTT for small packets due to the promotion delay not being required. The ideal pattern that would be expected based on the RRC specification [5] can be seen in the bottom graph.

Category	Label	Description
3G UMTS - RRC States in Specification	DCH	High-power, high-bandwidth
	FACH	Low-power, low-bandwidth
	PCH	No transmission possible
3G UMTS - Observed Behavior	FACH TRANSITION	Period of high latency when transitioning from DCH to FACH
	FACH INIT	Initial FACH state interval with *frequent* link layer retransmission
	FACH PROMOTE	State transition interval from FACH to DCH with signaling overhead
	FACH STABLE	Time interval in between FACH INIT and FACH PROMOTE
	PCH INIT	Initial PCH state interval with *few* link layer retransmission
	PCH PROMOTE	State transition interval from PCH to FACH with signaling overhead
QxDM Related	RLC RETX RATIO	$\frac{\#\ of\ Retx\ PDUs\ in\ \boldsymbol{T}}{Total\ \#\ of\ PDUs\ in\ \boldsymbol{T}}$, where $\boldsymbol{T}$ is a range of time
	INTER-PACKET INTERVAL	Time between when a test packet is sent and when a high-power state is induced, for the purpose of controlling RRC state.

Table 1: Summary of key terminology used.

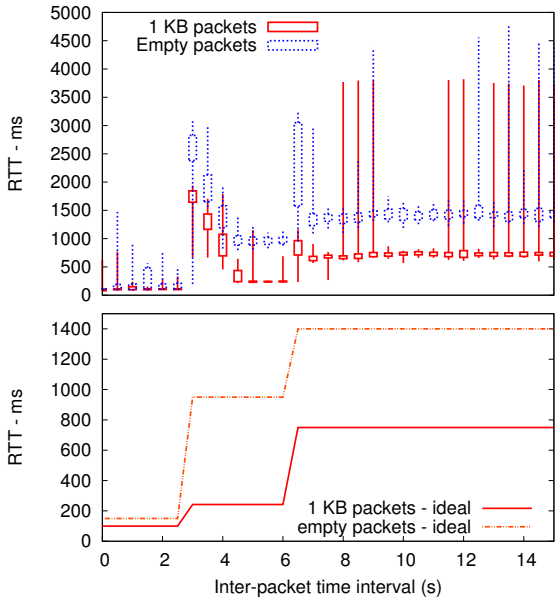


Figure 3: Top graph shows RTT as a function of time between packets, for twenty-two measurements. Corresponding ideal RRC state below; assumes behavior follows spec and disregards noise and transition delays. Unexpected latency could be observed during 3s to 5s that corresponds to FACH state. The inter-packet time interval is defined in Table 1

Comparing both graph at the same, we could clearly identify the large latency at the initial period of FACH state. In other words, all data transmissions have to experience such unexpected delays during the initial data transmission. We will identify the root cause of that problem in §5

4. CROSS-LAYER METHODOLOGY

We use QxDM to perform cross-layer analysis for several purposes. In §4.1 we describe how we collect ground truth data transmission information in the IP and RLC layers. We describe the cross-layer mapping algorithm in §4.2 that allows us to understand the impact of RLC layer behavior on higher layers, and in §4.3 we describe how we calculate the RLC retransmission ratio in QxDM.

4.1 QxDM Experiments

QxDM is a real-time data collection and diagnostic logging tool for measuring *Radio Frequency* (RF) performance in mobile devices [3]. It allows us to gain insight into lower layer RLC transmission information. The tool was used to collect data on IP packets, RLC data PDUs, and RRC states, as well as RLC control PDUs and RLC configuration information such as polling timers and retransmission limits. The experiments performed were done on two devices from two different manufacturers, which we will call M1 and M2. The M1 device runs Android OS 4.1.1, and the M2 device runs Android OS 4.0.4. This allows us to study potential device-dependent behavior.

There were several phenomena that we wished to investigate in order to uncover the root causes of behavior observed at the UDP level. We wanted to verify the existence of unexpected latencies during transitions to FACH and identify the root cause of the problem. To do so, we ran the RRC inference test for 160 consecutive hours and collected QxDM logs as well as server-side tcpdump [2] traces. We labeled our UDP packets with a sequence number, and the echo server sent back the received sequence number. We refer to this dataset as the *QxDM_UDP_Trace*.

For a second set of tests, we sent a packet train of TCP packets of size 10 KB with inter-packet timings between 3s and 5s, incremented by 0.5 seconds. These values were chosen as they fall within the range of inter-packet timings associated with unexpected FACH transition latencies and would allow us to investigate this unexpected behavior. We wished to investigate the RLC retransmission delay's influence on TCP retransmission. We refer to this dataset as *QxDM_TCP_Trace*.

4.2 Cross-Layer Mapping Algorithm

Correlating the transport layer packets with the RLC layer transmitted PDUs would allow us a transparent view of link layer behavior, especially RLC retransmissions. One major

limitation we need to address when using QxDM to analyze data is that packet logging is incomplete. Only the header and first byte of the data payload is logged for each RLC PDU. Furthermore, it is also possible for a small number of RLC PDUs to not be captured, leading to a sequence number gap. We created a mapping algorithm to address these limitations.

The cross-layer mapping algorithm maps complete IP packets (known as SDUs, or *Service Data Units*) to the corresponding fragmented RLC payload (or PDU). PDUs have a fixed size which we denote as S_{PDU}. The basic idea is that we find the first byte of the RLC PDU that matches the first byte of an SDU. We then skip forward from that byte in the SDU by S_{PDU}, and check if that next byte matches the next RLC PDU. As well, if the sequence number difference D between two consecutive PDUs is greater than 1, then some RLC PDUs are missing from the QxDM trace. Then, we skip ahead by $S_{PDU} \times D$ instead. We know our mapping is complete if every PDU's first byte maps an SDU, with the SDU bytes being the appropriate distance apart and with the ordering of PDUs and SDUs being conserved. Otherwise, no mapping was discovered. Algorithm 1 describes this mapping mechanism in detail.

However, this mapping mechanism cannot recover all packets in every case. In particular, this is true when missing PDUs are at the beginning or end of the mapped RLC list, but those cases occur rarely in the QxDM traces. We evaluate the accuracy of this improved mapping algorithm by observing the percentage of mapped IP packets in both *QxDM_TCP_Trace* and *QxDM_TCP_Trace*. We were able to correctly map 99.8% of packets.

Algorithm 1 Map the SDU to the corresponding PDU list

```
Function Cross-Layer-Mapping(SDU, PDU_LIST):
// point to the current mapped PDU in PDU_LIST
Initialize PDU_INDEX as 0
// point to the current mapped byte in SDU)
Initialize SDU_MAPPED_INDEX as 0
Initialize MAPPED_PDU_LIST as empty list
while PDU_INDEX is less than length of PDU_list do
  while SDU_MAPPED_INDEX < SDU size do
    if SDU[SDU_MAPPED_INDEX] equals to CUR_PDU then
      // Success to map current byte
      Append CUR_PDU to MAPPED_PDU_LIST
      CUR_PDU = PDU_LIST[PDU_INDEX]
      factor is seq_num between CUR_PDU and LAST_PDU
      Set LAST_PDU as CUR_PDU
      if cur_PDU has LI field in header then
        Increase SDU_MAPPED_INDEX by the value of LI field
      else
        Increase SDU_MAPPED_INDEX by the size of CUR_PDU multiples factor
      end if
      if CUR_PDU's HE field indicate last PDU then
        Break
      end if
    else
      // fail to map the current byte
      Reset MAPPED_PDU_LIST as empty list
      Reset SDU_MAPPED_INDEX as 0
      Break
    end if
    Increase PDU_INDEX by 1
  end while
  if SDU_MAPPED_INDEX equals to SDU size then
    // Success to find a mapping!
    return MAPPED_PDU_LIST
  end if
end while
return NOT FOUND
```

4.3 RLC Retransmission Calculation

The RLC layer includes a retransmission mechanism to improve the reliability of transmissions over the lossy data transmission channel. However, this may not eliminate unnecessary retransmissions, as it acts independent of upper layers. This is especially problematic for TCP retransmission timeouts. We describe in this section how we determine when RLC retransmission has occurred from the QxDM logs.

Each RLC sequence number uniquely identifies subsequent RLC PDUs, so we can determine RLC retransmissions based on duplicate sequence numbers. One complication, however, is that sequence numbers wrap around every 4096 PDUs. To avoid over-counting duplicate sequence numbers from a different cycle, a smaller window size of 512 PDUs is set to count how often sequence numbers reappear. The number of RLC retransmissions is the cumulative count of the repeated sequence number for a given list of RLC PDUs [4].

The RLC retransmission ratio measures the relative frequency of RLC retransmissions over a range of time as defined in Table 1. First, we describe how to determine the retransmision ratio for a RRC state. The RRC state for each RLC PDU is determined by backtracking to the most recent RRC state QxDM log entry. RLC-layer retransmitted PDU counts are broken down based on the RRC states where they were transmitted. By dividing the total number of retransmitted PDUs by the total number of PDUs in each RRC state, the retransmission ratio can be calculated. We also calculate the RLC layer retransmission ratio for specific inter-packet times (as described in Table 1) by counting retransmissions over a certain inter-packet time transmission period.

5. ROOT-CAUSE ANALYSIS WITH QXDM

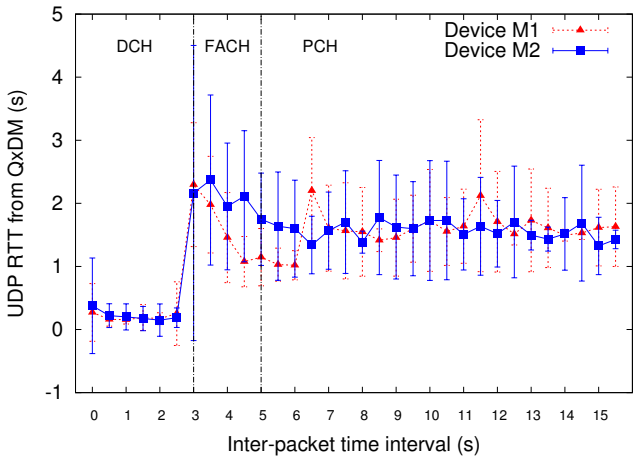
We examine the root cause of the unexpected latencies when transitioning between RRC states. We make use of data from *QxDM_UDP_Trace* and *QxDM_TCP_Trace*. To describe the root causes, we apply the terminology defined in §2. Based on our measurements of the FACH and PCH promotion times, we determine FACH_PROMOTE covers the time period of 0.92 seconds before DCH promotion occurs in *QxDM_UDP_Trace* and *QxDM_TCP_Trace*, and similarly PCH_PROMOTE covers the time period of 0.21 seconds before FACH promotion occurs as defined in Table 1.

To demonstrate the strong correlation between data link layer retransmissions and higher layer latencies, we calculate the RLC layer retransmission ratio for each inter-packet time interval using the methodology described in §4.3. We insert the inter-packet time for the test performed in the packet payload for data in the *QxDM_UDP_Trace*, so we can directly correlate the RLC PDUs with the associated timing after we apply the cross-layer mapping algorithm from §4.2. We show the high retransmission ratio over FACH state in Figure 4(b), which matches the latency behavior in Figure 4(a). The similar patterns imply that root cause of FACH latency issue resides in the data link layer.

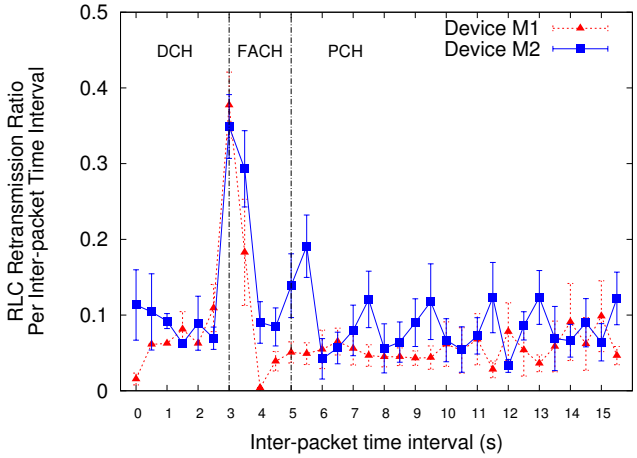
There are two possible RLC retransmission behaviors that lead to delays at the transport layer. First, retransmission may be necessary due to noisy channel conditions during FACH. Based on the *QxDM_UDP_Trace* and the *QxDM_TCP_Trace*, we can see that RLC retransmission ratios are particularly high during FACH. A possible solution to this problem is for the application to batch data transmissions to reduce the frequency of RRC state promotions, a solution that has been shown to be beneficial in eliminating other transmission-related delays [9]. Second, retransmissions can become unnecessarily delayed due to a slow response to the PDU loss signal (i.e., duplicated ACKs). In particular, this can effect TCP transmissions, as the relationship between TCP retransmissions and RLC retransmissions leads to delays. We propose a solution to address this latter problem in §6.

Using the RLC retransmission calculation methodology described in §4.3, we calculate the RLC transmission ratio for each state and each promotion or demotion period. We show these results in Figure 5. The retransmission ratios are particularly high in FACH_INIT and FACH_PROMOTE, and for the M2 device is especially high during FACH_INIT. These observations are consistent with the abnormal delays we observed when transitioning from DCH to FACH. A possible explanation is the difference in hardware used. The M1 and M2 devices use two different generations of chipsets, and the newer one supports a larger range of network types [6]. This chipset dependency might also explain why this phenomenon was not observed in earlier work, on older devices, which likely used yet another chipset. However, a detailed examination of hardware differences is out of the scope of the paper.

We also examined performance in TCP. A TCP RTO can



(a) UDP RTT Result



(b) RLC Retransmission Result

Figure 4: RLC layer retransmission ratio has a strong correlation with the delay over FACH state. Measurements on 1 KB packets.

cause a RTT delay as the congestion window size decreases and the device falls back to the slow start phase [14]. We use our mapping algorithm to correlate TCP retransmission behavior with RLC retransmissions. We found that the current RLC protocol responds sluggishly to the duplicate ACKs that signal that a PDU has been lost (see Figure 6). This leads to a TCP RTO which introduces more latency in the transport layer.

According to the 3GPP RLC specification, the sender only retransmits the PDU once it receives a STATUS LIST (i.e., non-acknowledged) control PDU from the receiver, ignoring duplicate ACKs [4]. These duplicate ACKs are strong hints for PDU losses. If the lost RLC PDUs were to be transmitted after 0.5s rather than 2.8s, then the TCP RTO could be avoided, reducing latency by more than 2.3s. We propose a RLC *Fast Re-Tx* mechanism, where unacknowledged PDUs are retransmitted once three duplicate RLC PDU ACKs are received by the sender. The resulting faster reaction to lost signals would reduce both RLC latency and latency in the

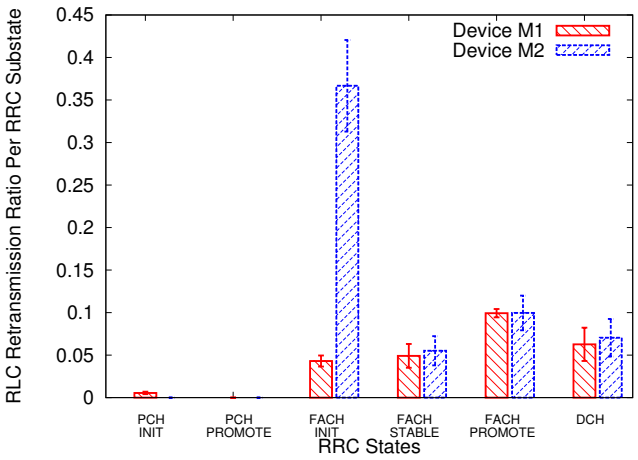


Figure 5: Significant RLC retransmission ratio over stable FACH state and FACH promotion state is consistent with higher-layer measurements and suggests that FACH_INIT has a significant impact on the state transition delay.

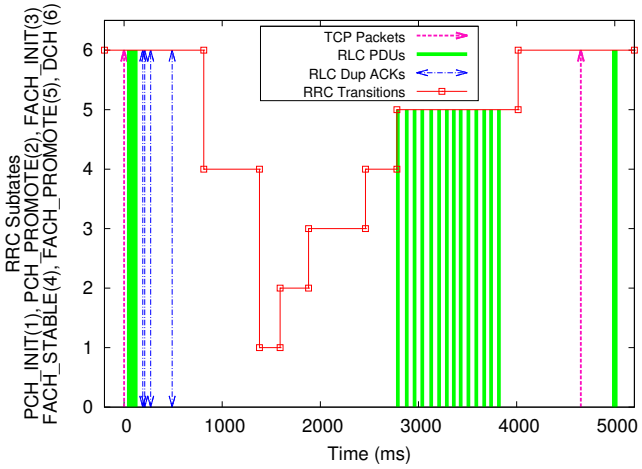


Figure 6: TCP RTO is caused by delays in RLC PDU retransmission. Duplicate ACKs suggest a PDU has been lost. By responding to this probable loss in a timely manner, RLC transmission latency can be reduced and sometimes TCP RTOs can even be avoided.

transport layer. We will evaluate this mechanism in §6

6. EVALUATION

As we have shown in §5, negative interactions between RLC-level retransmissions and TCP-level retransmissions result in increased delay. In this section, we propose a mechanism for reducing this delay by implementing RLC fast retransmission to allow the RLC layer to respond faster to PDU loss.

First, we explain how we measure RTTs in the RLC layer. As STATUS PDUs (ACK or NACK) are not generated by every received PDU, and are only triggered if a polling request is received or one or more PDUs are missing in the receiver buffer [4], it is difficult to estimate RTTs directly. QxDM traces only include client-side information, so we have no data on when the server receives each PDU. We estimate the RTTs of RLC PDUs based on the timestamp difference between the most recent sender polling request and the most recently received ACK. The RLC configuration limits the maximum polling request frequency to 500ms. A previous mobile RTT estimation study shows that the autocorrelation coefficient between two RTT measurements within 500 ms is more than 0.6 [13], so this estimate is reasonable.

6.1 Cost-Benefit Analysis

To evaluate the proposed *Fast Re-Tx* mechanism, we perform a cost-benefit analysis using the *QxDM_UDP_Trace* and *QxDM_TCP_Trace*. In our proposal, when the sender receives 3 duplicate ACKs, the device predicts that all the previous unacknowledged PDUs have not been received by the receiver, and transmits all of them right away. However, there is a chance that the PDUs are actually delayed by radio channel contention. We validate the actual cause of the delays using the QxDM traces, based on whether the sender receives an ACK or a NACK after the *Fast Re-Tx* occurs. A future ACK implies all the PDUs are delayed over the channel, while a NACK could indicate the PDUs are lost [4].

To perform the cost/benefit analysis, we categorize all duplicate ACK scenarios into 4 cases based on the prediction accuracy, which is the ratio of the number of PDU divided by the total number of *Fast Re-Tx* PDU. We call them — Win, Draw_Plus, Draw_Minus, and Loss. Draw_Plus occurs when the sender receives a NACK and more than 50% of the PDUs sent in *Fast Re-Tx* would have to be retransmitted anyway (according to the traces.) This is illustrated in Figure 7(b) — there is a decrease in latency. The "win" case is similar, but occurs where the TCP RTO was avoided entirely, leading to further latency reductions as shown in Figure 7(a). "Draw_Minus" occurs when less than 50% of the PDUs retransmitted through our proposed mechanism would have been retransmitted anyway — unneccessary retransmissions have occurred. This case is shown in Figure 7(c). If all PDUs were successfully delivered to the receiver, we refer to this case as "Loss", as shown in Figure7(d). Redundant data was transmitted and there were no latency gains.

As we can see from Table 2, almost exactly 75% of the time there would be a clear benefit from RLC *Fast Re-Tx*. The rest of the time more cellular traffic could be introduced and the RLC layer throughput could decrease. However, the throughput impact is less significant than the existing delays during FACH, since the throughput is much lower than the bandwidth over the FACH state [9]. Using the *QxDM_UDP_Trace* and *QxDM_TCP_Trace* and the previous RLC RTT estimation mechanism, we estimate that the overall RLC delay would be reduced by up to 35.69%

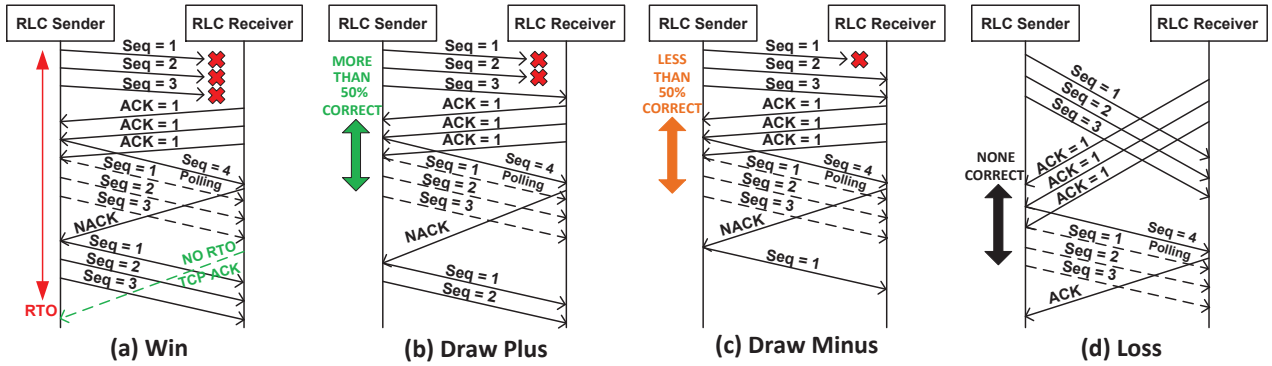


Figure 7: *Win*: RLC Fast Re-Tx avoids a TCP RTO. *Draw Plus*: Prediction accuracy is more than 50%. *Draw Minus*: Prediction accuracy is less than 50%. *Loss*: All predictions are wrong.

Case Name	Total Case Ratio (%)	Average RLC RTT	
		Delta (ms)	Delta Ratio (%)
Win	10.32±1.89	-0.08	0.21
Draw_Plus*	64.69±8.32	-1.178	3.07
Draw_Minus	20.63±3.45	+0.365	0.95
Loss	4.36±0.06	+0.057	0.15

* The Draw_Plus case excludes the percentage of Win

Table 2: The RLC *Fast Re-Tx* occurrence percentage and impact on average RLC RTT value. Delta is the difference between average RTT of each case and the average RLC RTT in the QxDM traces. Delta Ratio is defined as the delta RTT divided by the average RTT.

during FACH_INIT and FACH_PROMOTE. This is not at the expense of performance in other states — in fact there is an overall average benefit of 2.66%. Although further testing would be required to ensure that devices benefit from this mechanism in a real implementation, these results suggest that this approach is a promising potential solution to the problem of negative interactions between TCP and RLC retransmission mechanisms, especially in the poorly-performing FACH state.

7. DISCUSSION

7.1 Future Work

The current study is primarily focus on the latency improve over the RLC layer. It is also possible to measure the delay improvement over application layer directly, especially for the initial period of the data transmission. Another possible direction is to evaluation the energy efficiency for the improved RLC protocols with fast retransmission mechanism. Since QxDM provides device data transmission power information, we could also evaluate the energy saving over each RRC state and state transitions.

7.2 Limitation

Since we could not modify the handset's NIC (network interface card) driver, we only evaluate the modified trace information based on the QxDM traces. Currently we assume the fast retransmitted RLC PDUs guarantee to be received on the base station, but the actual channel is lossy and the actual benefit might be lower than the simulated results. In addition, since the modification of RLC protocol requires hardware modification, the deployment of the modified protocol could be slower than a software solution.

8. RELATED WORK

One of the previous cross layer study combined inferred RRC state information with collected device power trace to imply application behaviors leading to unnecessary energy consumption [10]. Their primarily focus on the application level energy optimization instead of latency. The QxDM in our study provides more fine grained performance, latency and power information. We could have more detailed insights to understand data link layer root cause for abnormal latency and inefficient energy consumption behaviors in upper layer.

Related to our RLC fast retransmission proposal, a previous study provides cross comparison analysis over the TCP and RLC protocols, and optimizes the default protocol parameters [8]. Their primary goal is to improve the performance behavior by introducing TCP congestion control mechanism into the RLC layer. However, their traces are purely generated from simulation software, whereas we use real traffic traces.

9. CONCLUSIONS

From the RRC state inference model, we evaluated the latency across each RRC state and state transitions. We observed the extra latency in the FACH state using RRC inference methodology. Utilizing the QxDM monitoring tool, we have a better visibility over RLC layer traffic information. We wrote a QxDM log parser and analyzer to cross mapping the transport layer and data link layer information, and identified the root cause of abnormal delays cause by the imperfection in the RLC layer protocol. We proposed a RLC

Fast Re-Tx mechanism to actively response to the PDU loss, and evaluated the delay cost-benefit over real-time traces. The latency reduction is up to 35.69% over FACH state, and it could also reduce the overall latency by 2.66%. Therefore, the RLC *Fast Re-Tx* mechanism could enhance the user mobile experience by introducing less delays, especially during the initial data transmission period.

10. ACKNOWLEDGMENTS

We would like to thank Yihua Guo, Sanae Rosen, and Z. Morley Mao for supporting and commenting on this paper.

11. REFERENCES

[1] 3GPP UMTS (3G). http://www.3gpp.org/UMTS.
[2] tcpdump. http://www.tcpdump.org/.
[3] QxDM Professional Proven Diagnostic Tool. http://www.qualcomm.com/solutions/testing/diagnostics-software/, 2012.
[4] 3GPP TS 25.322: Radio Link Control (RLC) - UMTS, 2013.
[5] 3GPP TS 35.331: Radio Resource Control (RRC) - UMTS, 2013.
[6] Snapdragon (System on Chip). http://en.wikipedia.org/wiki/Snapdragon_(system_on_chip), 2013.
[7] Why are 3G Networks so Slow? http://blog.ioshints.info/2013/04/why-are-3g-networks-so-slow.html, 2013.
[8] Alcaraz, Juan J., Fernando Cerdn, and Joan Garca-Haro. Optimizing TCP and RLC interaction in the UMTS Radio Access Network. *IEEE Network Magazine*, 2006.
[9] Feng Qian, Zhaoguang Wang, Alexandre Gerber, Z. Morley Mao, and Subhabrata Sen and Oliver Spatscheck. Characterizing Radio Resource Allocation for 3G Networks. In *Proc. ACM IMC*, 2010.
[10] Feng Qian, Zhaoguang Wang, Alexandre Gerber, Z. Morley Mao, Subhabrata Sen, and Oliver Spatscheck. Profiling Resource Usage for Mobile Applications: A Cross-layer Approach. In *Proc. ACM MobiSys*, 2011.
[11] Haiqing Jiang, Zeyu Liu, Yaogong Wang, Kyunghan Lee, and Injong Rhee. Understanding Bufferbloat in Cellular Networks. In *Proc. of the 2012 ACM SIGCOMM workshop on Cellular networks*, 2012.
[12] Junxian Huang, Feng Qian, Alexandre Gerber, Z. Morley Mao, Subhabrata Sen, and Oliver Spatscheck. A Close Examination of Perfomance and Power Characteristics of 4G LTE Networks. In *Proc. ACM MobiSys*, 2012.
[13] Qiang Xu, Sanjeev Mehrotra, Z. Morley Mao, and Jin Li. PROTEUS: Network Performance Forecast for Real-Time, Interactive Mobile Applications. In *Proc. ACM MobiSys*, 2013.
[14] Vern Paxson and Mark Allman . Computing TCPs retransmission timer, 2000.