Discovering RRC Transient States, Behavior and Performance Implications in Cellular Networks

Sanae Rosen[#], Haokun Luo[#], Z. Morley Mao[#], Jie Hui[*], Aaron Drake[*], Kevin Lau[*]
[#]University of Michigan, [*]T-Mobile USA Inc.[*]

ABSTRACT

In 3G and 4G cellular networks, devices transition between different *Radio Resource Control* (RRC) states in response to network traffic and according to parameters specific to network operators. These states have different energy consumption and performance (bandwidth and latency) tradeoffs. We present the first in-depth examination of the implementation of these states and their impact on performance. We devise a general technique for inferring RRC states for any network technology on end-user devices that addresses issues such as network noise and captures non-ideal behavior. In particular, we discover that at the *Radio Link Control* (RLC) layer, stages within RRC state transitions, which we call *transient states*, have a significant impact on user-perceived performance and measure this impact through our inference tool. We also investigate this behavior at the RLC layer directly by developing a cross-layer analysis technique to investigate the root causes of these phenomena. Based on our observations, we propose a new RLC *Fast Re-Tx* mechanism that could reduce the RLC data transmission latency by up to 35.69%.

1. INTRODUCTION

Cellular network technologies, such as 3G *Universal Mobile Telecommunications System* (UMTS) and 4G *Long Term Evolution* (LTE), require that devices transition between various RRC states based on the network traffic patterns of individual mobile clients. These states have different performance and energy consumption tradeoffs, as well as different state transition delays. Using high-power RRC states only when necessary allows users to experience good network performance on resource-constrained mobile devices. Although the RRC states are defined by *3rd Generation Partnership Project* (3GPP) [9, 10], many aspects of the RRC state machine, such as timers for transitioning between states, are implementation or configuration-specific, differing by device model, network operator and location. The real-world deployment of RRC state machines and the impact on performance are not well-understood.

A better understanding of RRC state behavior would be beneficial for many parties. Cellular network operators would be interested in determining how devices on their networks perform and how performance can be improved. Device manufacturers would be interested in detecting device-dependent effects — device implementation details and features such as Fast Dormancy [5] can mean different devices perform differently. Developers of major applications might be interested in understanding how network behavior can impact application performance, as work has been done to show that application behavior can decrease RRC-related performance issues [17]. Finally, researchers studying protocol optimization would find a better understanding of RRC state behavior valuable.

In this paper, we examine how RRC states are implemented, behave and perform in real-world implementations. We present a new approach to inferring RRC state transitions that addresses limitations in previous work which prevent inference from being done in non-ideal network conditions or by non-expert users. In particular, we focus on detecting performance anomalies caused by RLC-layer *transient states*, which can have a significant performance impact. We perform a study of RRC state implementations and performance on 16 network operators from 9 countries, and use RLC layer analysis to understand the root causes of behavior and performance trends discovered. We present three main categories of contributions:

- **Methodological contributions.** We have created two types of tools. First, we designed a novel generic RRC state model inference method that is robust to poor network conditions and interfering background traffic on the mobile device. It is implemented in an Android application and is designed to be usable out of the box by non expert users. We focus primarily on inferring the demotion timers of RRC states as well as the performance associated with each state. We hope to release this tool as open-source software. Second, we determined methods to effectively make use of RLC-layer traces from *Qualcomm eXtensible Diagnostic Monitor* (QxDM). We developed a novel cross-layer mapping mechanism that correlates transport layer packets with data layer link *protocol data units* (PDUs, the smallest data transmission unit in RLC). It can successfully map 99.8% of transport-layer packets to PDUs. This allows

[0]The views presented here are as individuals and do not necessarily reflect any position of T-Mobile.

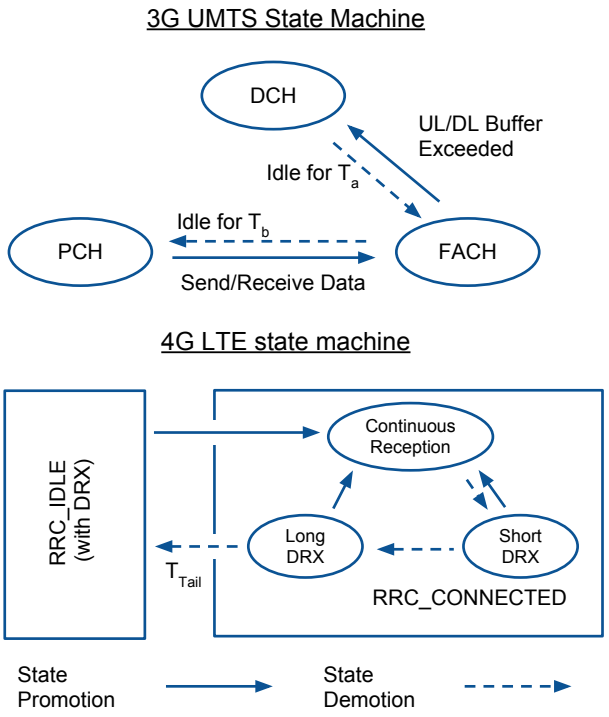


Figure 1: Possible 3G and 4G State Machines as Specified by 3GPP.

the root causes of TCP and UDP performance and packet loss issues to be accurately identified at the RLC layer.

- **New Findings on the Impact of RRC State on Performance.** We analyze the RRC state implementation and associated performance characteristics for a number of network operators, showing that our methods are useful in understanding RRC states in the real world. We present several new findings that show there are significant undetected performance problems in networks today. At the RLC layer, we show that there are frequent RLC-layer retransmission delays surrounding *transient states*, especially during state transitions to and from FACH.
- **Proposed Solution to Performance Problems Found.** We propose an improvement to how RLC currently behaves based on our improved understanding of RLC-level behavior. To reduce potentially unnecessary retransmissions that occur during poorly-performing FACH-related transient states, we propose an RLC *Fast Re-Tx* mechanism that actively responds to the RLC PDU loss signals. By simulating this fast retransmission mechanism using real QxDM traces that provide visibility of RLC traffic behavior, we find that our proposed mechanism could reduce RLC latency by up to 35.69% when these poorly-performing FACH states occur. In other states, the latency is still reduced by 2.66% on average.

2. BACKGROUND

As illustrated in Figure 2, in both 3G UMTS [1] and 4G

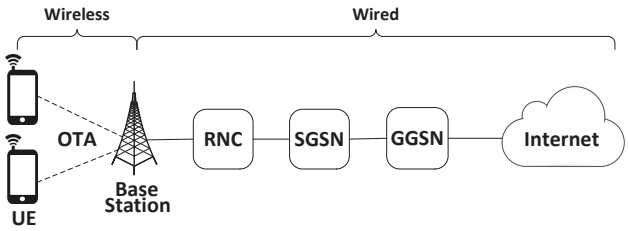


Figure 2: The general cellular network architecture

LTE networks [2], data is transmitted from *user equipment* (UE), i.e. mobile devices, to the base station (Node B in UMTS, eNB in LTE), then to the *Serving GPRS support node* (SGSN) and *Gateway GPRS support node* (GGSN), and ultimately to the server [18]. The link between the UE and the base station is known as the *over-the-air* (OTA) link, and it is the only wireless link in the network topology. The rest of the links from the base station to the internet are all wired.

Mobile cellular networks use the Radio Resource Control (RRC) protocol to allocate resources to mobile devices. 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 device 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 [9, 10].

Different carriers implement RRC state machines differently, with state transitions and different associated timers varying between carriers. However, for 2G, 3G and 4G LTE, there are a fixed set of possible states defined. We give a brief overview below; a more detailed overview can be found in previous work [16, 19].

For 3G UMTS [9], 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 at the top of Figure 1. These terms are summarized in Table 1.

For 4G [10], the state machine is more complicated, and is summarized in the bottom half of Figure 1. We are concerned mainly with transitions between RRC_CONNECTED, a higher-power state, and RRC_IDLE, a lower-power state where no data is transmitted. The other states have timers on the orders of tens or hundreds of milliseconds are are not practical to measure on end-user devices, as tools such as a power monitor are required.

Determining when to fall back to a low power state can have a substantial impact on performance, as can the state transitions supported. If demotion happens early, then there will be additional promotion delays, resulting in decreased

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 - Newly-detected Transient States	FACH INIT	Initial FACH state interval with *frequent* link layer retransmission
	FACH PROMOTE	State transition interval from FACH to DCH with signalling 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 signalling overhead
3G UMTS - UDP-Layer Observed Behavior	FACH TRANSITION	Period of high latency when transitioning from DCH to FACH
4G LTE - RRC States in Specification	RRC CONNECTED	High-power, high-bandwidth
	RRC IDLE	No transmission possible
4G LTE - UDP-Layer Observed Behavior	LTE TRANSITION	Period of high latency when demoting from RRC CONNECTED to RRC IDLE
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
	UDP loss ratio	$\frac{\#\ of\ UDP\ packets\ \boldsymbol{NOT}\ received\ by\ receiver\ in\ \boldsymbol{T}}{Total\ \#\ of\ UDP\ packets\ that\ the\ sender\ transmitted\ in\ \boldsymbol{T}}$, 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 measuring timers.

Table 1: Summary of key terminology used.

user performance. If it happens late, then an unnecessary amount of power will be consumed. For this reason, we wish to understand how carriers implement RRC state machines in practice.

Furthermore, we found that network traffic often does not follow the ideal pattern expected by the RRC specification, another major motivation for understanding RRC state behavior in the real world. This occurs in particular during state transitions, where certain transitions on certain devices can lead to substantial delays, so we define terms to refer to these phenomena which we will use throughout the paper. There is often a significant additional promotion delay immediately after a state demotion, especially a demotion from DCH to FACH. We refer to this phenomenon as *FACH_TRANSITION* when it occurs immediately before FACH, and *LTE_TRANSITION* when it occurs in LTE.

To understand the root cause of this unexpected behavior, we observed behavior at the RLC layer. We break down the behavior in each state at the RLC layer into several smaller *transient states* as shown in Figure 3.

We define the FACH_PROMOTE and PCH_PROMOTE transient states based on behavior observed at the RRC layer. These states end when the new RRC state is logged by QxDM and start when the previous IP data packet was sent. We describe the experiments to determine when this period occurs in 4.1. The FACH_INIT transient state corresponds to a period of high RLC retransmissions determined experimentally from the QxDM traces. The period of FACH not covered by FACH_PROMOTE and FACH_INIT we define to be FACH_STABLE. There is no corresponding PCH_STABLE state as no data is sent in PCH. Therefore, the PCH_INIT transient state refers to the period of time in PCH during our tests where the device is not in PCH_PROMOTE. We determined that the FACH_TRANSITION delays occur primarily due to losses in the transition states FACH_INIT and FACH_PROMOTE.

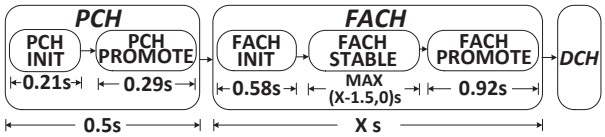


Figure 3: RRC transient state definitions from observed RLC-layer behavior during state promotions. We examine these individually through QxDM to investigate root causes of latency.

3. RRC STATE INFERENCE METHODOLOGY

The 3GPP specifications leave some implementation details, such as specific timers, to the carriers [10, 9]. It has been shown in previous work [16] that it is possible to infer RRC state transition timers using differences in RTT values for different inter-packet timings. As transitions between states involve overhead, it is possible to infer the RRC state at a given time by sending a packet to deliberately trigger a promotion to a higher state and measuring the resulting RTT. By varying the spacings between packets it is possible to determine demotion timers, and by varying packet sizes it is possible to detect promotions triggered by large data transmissions.

A large-scale survey of the RRC state machines of different carriers would be valuable in understanding how RRC state machines are implemented by carriers and perform in practice. Additionally, it would allow us to determine if, for a specific carrier, there are differences between phone models or geographic regions. It is possible that carriers may make use of different equipment vendors in different markets, leading to RRC implementation differences. Also, some aspects of RRC behavior are known to be device-dependent, such as fast dormancy [5]. RRC transitions involve both the device and base-station, so device-specific differences are possible, and in §5 we discuss some which we have detected.

We have also found that the performance trends expected

from the RRC state specification is not reflective of the performance clients experience in practice. A motivating example of the problem can be seen in Figure 4. 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, as a state transition only occurs after sufficient data is transmitted.

It is clear from Figure 4 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 can be seen in the bottom graph.

There are two main differences between the measured values and the ideal behavior. First, transitions, especially to FACH, involve long and unexpected latencies, which we will show are due to delays in promotion-related RLC-layer transient states. Second, the measurements are very noisy, especially in PCH. As a result, inferring RRC states from a small number of tests is not reliably accurate, and data processing is often needed for accurate results.

Evidently, there are several significant challenges that need to be addressed to detect these types of non-ideal RRC state behavior. First, substantial processing is needed to be able to eliminate network noise and correctly infer RRC timers — the data shown in Figure 4 is in fact less noisy than average. Furthermore, even with controlled experiments the high degree of variance in our measurements often precludes the use of the algorithm presented in previous work [16]. Likely a large increase in network traffic in the past few years has lead to the increased presence of noise in our data. It seems unlikely that, as cellular networks become more congested, inference will become easier, so a more robust RRC state inference methodology is critical

Second, due to the assumption that devices will closely fit a specific pattern of behavior, unexpected phenomena caused by lower-layer behavior, such as FACH_TRANSITION, would not be identified by an algorithm based on matching data to a known RRC state pattern only. With existing approaches, these delays would be misclassified as noise or PCH and overlooked. To detect these phenomena, a more general approach is needed. An approach agnostic to RRC state also has the benefit that only one implementation is needed for different network technologies, making a public deployment more practical. We inferred timers for nine different network technologies in our public deployment.

Finally, the existing inference algorithm cannot run as

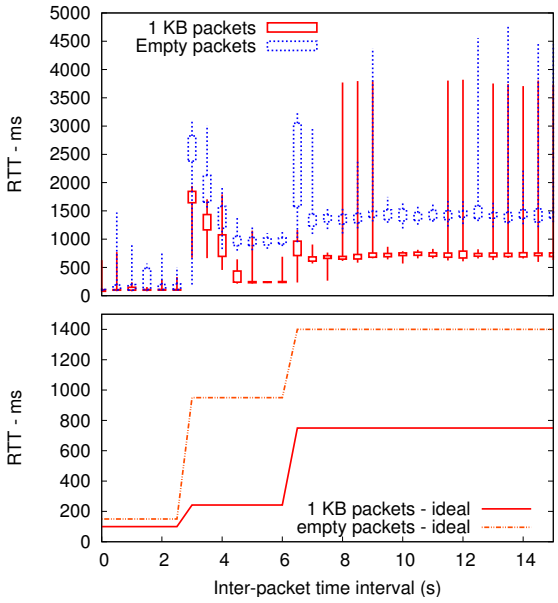


Figure 4: Top graph: 22 measurements of RTTs as a function of time between packets. Ideally, we would expect it to appear like the lower, ideal graph, with no transmissions delays.

an application on the end-user devices of non-experts, as it requires all network traffic (including background traffic not under the user's control) to be paused during these tests. In order to make a public deployment possible, it is necessary to modify the algorithm to ensure that it is robust in the presence of background traffic, does not interfere with existing applications and does not require root access.

In developing these algorithms, we made use of three test devices, covering two models of phone and two carriers. We examined T-Mobile and AT&T, using one Samsung Galaxy SIII for each carrier as well as an HTC one on T-Mobile. We used QxDM to verify the ground truth for network timings for two of these devices, as well as to verify the presence of the anomalous FACH_TRANSITION behavior and to determine the impact of RLC-layer transient states. We also validated our findings with a power monitor, as in previous work [16, 19]. Finally, in a controlled study with various devices on a single carrier, T-Mobile, using known RRC state machines (as described in §5) we verified that our inference algorithm is effective, including in higher-noise conditions. We also determined how much data is insufficient for the algorithm to work. We compare our inferred timers with the QxDM-derived ground truth in §5.1.1.

We start by describing the client design below.

3.1 RRC Inference Client Design

Data is collected client-side, and a server script generates an RRC state model for each device based on all data collected to date. This allows for a relatively lightweight client application as well as allowing the RRC state model to be improved over time. We use the example of 3G UMTS

Algorithm 1 Client-side data collection

```
Function ClientDemotionTest():
for n = 0 to 30 inclusive do
    for i = 0 to 9 do
        Send max bytes, server echoes min bytes.
        Sleep 500n ms.
        Send ten max bytes, server echoes min byte for each.
        Record average RTT and packets lost (7s timeout).
        Sleep 500n ms.
        Send ten min bytes, server echoes min byte for each.
        Record average RTT and packets lost (7s timeout).
        if No other traffic sent then
            record RTTs and losses
            break from inner loop
        end if
    end for
end for
```

below, but this approach applies to LTE and *Enhanced Data rates for GSM Evolution* (EDGE) as well.

The client design is based on previous work [16], although substantial changes were needed to enable a real-world deployment and for anomalous transition states to be detected. Algorithm 1 summarizes the client design. In previous work, DCH is induced by sending a 1 KB UDP packet, then another UDP packet after a specified period of time, which is either empty or 1 KB, so that state transitions dependent on the quantity of data sent could be identified. The server echoes back an empty packet so the RTT can be calculated — an empty packet is chosen to minimize the effect of queueing on the return path. Differences in RTT allow the RRC state for each inter-packet timing to be determined, and thus the RRC state timers.

In previous work, a straightforward classification approach was used. If the *round-trip time* (RTT) is small, it is in DCH; if there is a long delay it is in PCH, as the delay implies a promotion has occurred. If the delay of the small packet is substantially smaller than the large one, it is likely in FACH, where a promotion to DCH occurs only when the data in the buffer exceeds a certain size. Similar packet timing patterns occur for other technologies; for LTE, RRC_CONNECTED follows a similar pattern to DCH, and RRC_IDLE is similar to PCH. Our classification approach is similar, but allows us to detect unexpected phenomena — for example, due to transition delays we can no longer safely assume that DCH is associated with the longest RTT.

There are several modifications that need to be made to this algorithm. Network characteristics and RRC implementations have changed since the previous study was performed in 2010 [16], so some assumptions made then do not hold. In particular, anomalous transition behavior has become common and we wish to study this phenomenon. Changes also need to be made so that the tool can be used as part of a public deployment, as there were numerous practical limitations to the tool from previous work.

First, we modify how we collect data to ensure that timers are accurately inferred. Inter-packet timings are increased at half-second intervals and only measure from 0 to 15 seconds, instead of increasing by second intervals over 30 seconds. Since the previous study, timers have become shorter and half-seconds now make a significant difference. As well, for efficiency we measure the timings for large and small packets together. Each individual subtest for a specific inter-packet timing n is performed as follows: a 1 KB packet is sent to induce DCH, then there is a n-second delay, then 1 KB test packets are sent, then there is another n-second delay, then empty test packets are sent. We send 10 packets for each test to count losses, as well as record the signal strength values. This data is sent to the server for processing.

We collect data for all inter-packet timings, as opposed to taking a binary-search approach, because RRC state is often not apparent from a single sample. For timers of 1100 and 1500, for example, the device appears to be in PCH, but it might instead be experiencing high network delays or be in an anomalous transition state.

Changes are also made to make the algorithm suitable for a public deployment in an Android application. We need to ensure no interfering network traffic is sent by the device that might unexpectedly alter the RRC state. We also do not want users to need to do anything beyond installing the application. We make use of global packet count information in /proc/net/dev to verify that no interfering traffic has been sent for our tests. After 15 failed tries to measure an inter-packet interval, we record the test was unsuccessful, as the device may be under heavy use at the time.

In the best case, where every try succeeds, the test will take just under 8 minutes. In practice, in controlled experiments it usually took twenty to thirty minutes.

We also record the effects of RRC state on higher-level protocols. We record tests in the middle of each inferred RRC state. We record the impact of RRC state on DNS lookups, TCP handshakes, and HTTP connections.

This methodology is only suitable for timers at the granularity of half-seconds. For 3G this is not a problem, as timers are generally on the order of seconds. For LTE, the timer for the transition between RRC_CONNECTED and RRC_IDLE is similarly on the order of seconds. However, some of the timers for the states within RRC_CONNECTED can be as small as 20 ms, and even the RRC_IDLE cycle length requires measurements several times a second to detect [19].

We have explored various approaches to measuring these timers. There are three main challenges: there is a great deal of noise in the network; the Android framework introduces additional unexpected delays; and measuring these short timers increases the amount of traffic that needs to be sent by several orders of magnitude. We were able to substantially increase the accuracy of our RTT measurements by collecting tcpdump [4] traces, but this was insufficient. We found that even after sending packets continuously from one device for days, it was not possible to reliably and accurately infer these small packet timers.

We also investigated the possibility of performing all measurements passively. By monitoring packet counters

in /proc/net/dev/, it is possible to see when packets are being sent and received by existing applications on the device, thus eliminating the cost in terms of data overhead. We monitored the packets sent and received on an actively used device for a week, and found that the range of inter-packet timing intervals was not sufficiently varied to reach any useful conclusions during that time. As continuously monitoring background traffic in this manner can have a significant impact on battery life, this does not appear to be a feasible solution.

UMTS demotion patterns are detected in the same manner as in previous work [16]. We do not currently infer the buffer sizes that need to be exceeded to trigger a transition from FACH to DCH, but this could be done using a binary-search approach, by sending packets of various sizes in order to determine at what point the RTT changes substantially.

3.2 RRC Inference Server Design

Aside from storing our results, the server's main role is generating the RRC state models. Upon uploading a new set of data, a new model is generated for that device ID. To deal with the higher level of noise in our data as well as to be able to analyze all network types and detect unexpected behavior, it was necessary to take an entirely different approach to the server-side analysis from what was done in previous work [16].

In the ideal case, the data would closely fit the ideal RRC state behavior. In practice, even in the best case it looks more like Figure 4, where there are spikes in latency around RRC transmissions, as well as noise in high-latency RRC states. There may be far fewer measurements for a device, and the data collected is often noisier. Furthermore, we would like to observe when devices behave in unexpected ways.

To create a reasonable RRC model under adverse conditions, we first remove noise from the data, and then divide into timer ranges where RRTs are similar. Finally, we label the states detected where possible. We treat 1 KB packets differently from empty packets. The FACH state is characterized by these packets having very different values and therefore we wish to ensure these differences are preserved.

Filtering noise is not always straightforward, as interesting phenomena such as high round-trip times surrounding state transitions can look like intermittent delays, and vice-versa. We experimented with several noise filtering approaches on four devices with different manufacturers and carriers to develop our filtering approach before verifying with a larger number of devices. Taking the average value of all tests was not effective, as these values are often skewed by high amounts of noise. Taking median values is more effective, but it is also sensitive to periodic noise spikes, especially for small numbers of tests. We also tried taking the minimum value of a set of tests, the intuition being that the network is likely to add delays but not subtract them, but this also resulted in noisy data.

There were two approaches that did work well, however,

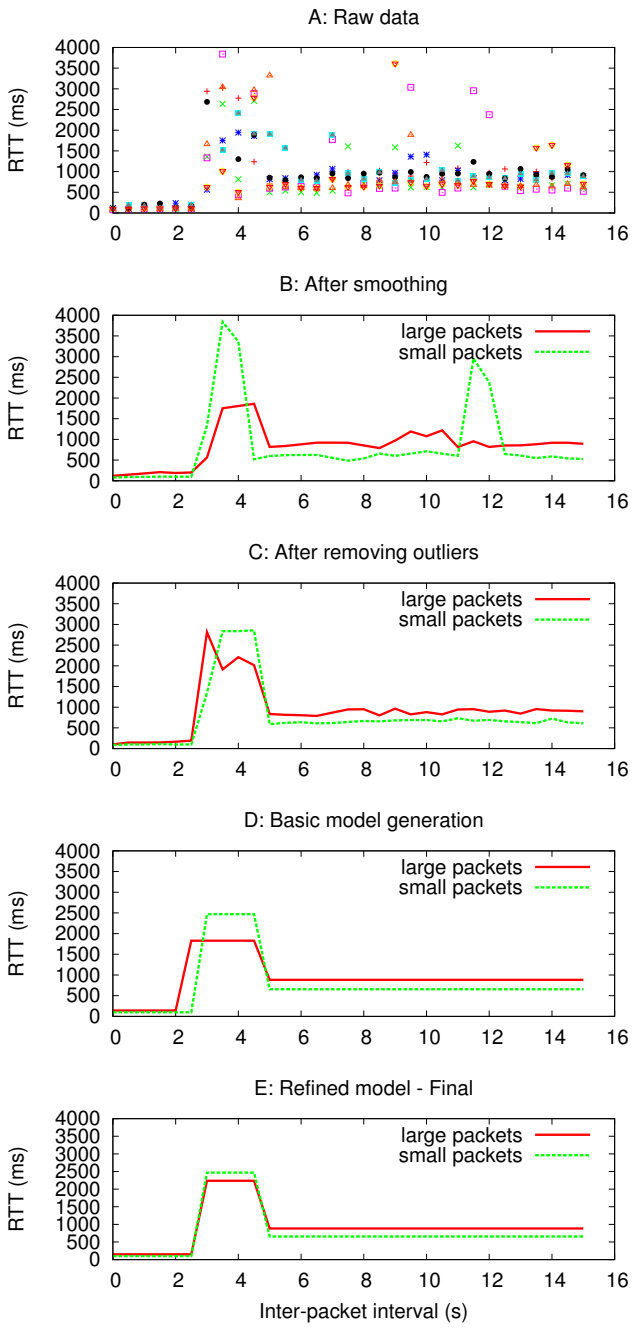


Figure 5: Steps in extracting RRC model and unexpected behavior trends from raw data. Note that the raw data follows a non-ideal pattern. Step B: modified running average function that preserves state transitions applied to one test only. Step C: all tests averaged without outliers removed. Step E: DCH and PCH state extracted; in between them is RRC_TRANSITION rather than FACH — the anomalous transition period occurs throughout the entire FACH state.

which are complementary. The first is to remove outliers. If there are three tests, with round-trip times 132, 145 and 370, then most likely the third one is due to intermittent

delay. We disregard all values that are more than half a standard deviation from the mean, or one standard deviation when less than ten tests were performed, then average the remaining values. This significantly reduces noise in our data in most cases. However, for very noisy data and few tests this still often leaves noise spikes. If there are three tests, and two coincidentally happen to have latency spikes at the same time, then this spike will not be filtered out.

A modified running average function applied to the data is also effective. We call this our "smoothing" function, which is applied to individual tests. For every data point, we first consider the difference between the points on either side of that point. If this difference is more than a quarter of the data point in question, we leave it as is. Otherwise, the data point is updated to be the average of the points before and after it. This has two results: first, we filter latency spikes, and second, we do not smooth large jumps in the data which are characteristic of state transitions.

These two approaches complement each other. Smoothing works well when the probability of a latency spike for each data point is independent, but works poorly where there are consistent RTT delays for several consecutive tests. Outlier removal works well when a specific test has latency spikes that other tests do not, and especially when a specific test has much longer RTTs than the others. It does not work well for very high levels of random noise.

After experimenting with various orderings for these two approaches on a number of devices, we found that smoothing each test individually before removing outliers between the tests performs best. If we remove outliers first, then for noisy data we may inadvertently filter out the correct, low-noise values. We then smooth the data again at the end to make the data to process cleaner. The results of these steps can be seen in Figure 5, parts B and C. B shows the results of smoothing a single test only.

After these pre-processing steps, we create the model. At first, we continue to treat small and large packets separately. Starting with the smallest interpacket interval, we divide the data into segments with approximately constant round-trip times. The results of this can be seen in Figure 5, part D.

Then, we compare the two models generated — that for the large packets, and that for the small packets. Although the average RTT for the two models differs, the beginning and end of each segment should be the same for both. There are two cases where the boundaries of segments may differ after model generation that need to be corrected. First, with low amounts of data, it sometimes happens that the beginning and end of a segment for each model is off by half a second. An example can be seen in Figure 5, part D. In this case, we find which segment division will result in the smallest average error and apply that to both models. Second, there are some states where behavior for one packet size or the other does not change. In particular, in the FACH state it is possible that the RTT for small packets is very close to that in DCH state. In this case, we split the larger segment.

Finally, we label the states. We do not use the same assumptions in doing so as in previous work as we find these do not always apply to our data. In particular, we do not assume that in PCH the 1KB packets and empty packets have very similar delays, as we find that this is frequently not the case. Instead, we determine states based on the relative average RTTs of each segment. For example, FACH (by definition) has latencies for empty packets close to those in DCH, whereas PCH does not. We also account for the fact that we expect to see states appear in a particular order as we increase inter-packet timings. For example, a carrier implementing the UMTS specification will never demote to FACH after demoting to PCH with no packets being sent.

This step was first developed and validated on a small number of devices whose behavior was verified using QxDM. It was also validated using an internal deployment on a carrier with known RRC state machine timers.

We look for the following states: high-power (DCH-like), low-power (PCH-like), and FACH-like. We also look for latency spikes transitioning between these states, like that seen in Figure 5. Anything that does not fit one of these patterns we label as "Anomalous" and flag for manual investigation. We err on the side of flagging models as anomalous to ensure that all other RRC states are accurate. We describe these results in §5.

4. CROSS-LAYER ANALYSIS 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 [7]. It allows us to gain insight into lower layer RLC transmission information. It is a Windows based monitoring application. When I perform control experiments and real application measurements, I plug in the device to the desktop or laptop with QxDM software installed. 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.

Once the experiments finished, we filtered out the real-time monitoring information related to IP packets, RLC PDUs (protocol data unit, the smallest data transmission unit in RLC layer), and RRC states in Table 2, and dumped the results into a log file. The 0x11EB log entry includes IP headers, IP payloads, and its customized header. Since

QxDM Log ID	Description
0x11EB	IP data packets
0x4132	WCDMA RLC downlink acknowledge mode configuration
0x4133	WCDMA RLC uplink acknowledge mode configuration
0x413B	WCDMA RLC uplink acknowledge mode PDU
0x418B	WCDMA Flexible RLC downlink acknowledge mode PDU
0x4125	WCDMA RRC states

Table 2: QxDM log entries used in cross layer analysis

large IP packets will be fragmented into smaller segments, the customer header could indicate the segment index of the whole IP packet. The 0x4132 and 0x4133 unveil the RLC AM (acknowledge mode) configurations, i.e. the polling function timers, the retransmission limit for a single PDU, and etc. The 0x413B, and 0x418B provides RLC PDU header and first byte payload information for both data PDUs and control PDUs (or STATUS PDUs) in both uplink and downlink directions. We wrote a QxDM log parser to aggregate the filtered entries, and apply cross layer analysis to understand the correlation between different layers. We conducted all the experiments on two devices – Galaxy S3 with Android OS 4.1.1 and HTC One S with Android OS 4.0.4, to observe any device dependent behavior.

QxDM provides precise RRC state information to indicate the current radio channel state. RLC data PDU and status PDU will assist us in cross layer mapping to the transport layer data, i.e. TCP/UDP. Other context information could also be found in the log, i.e. signal strength and physical layer transmission bit rate. The QxDM log is essentially a list of log entries formatted as a uniformed header with timestamp, a detailed log entry description, and a raw hex dump result.

Table of QxDM entries

First, we validate the RRC inference methodology discussed above. We calculate the RRC demotion timers directly from QxDM logs. According to the 3GPP spec [9], the sender receives a *radio bearer reconfiguration* (RBR) message from the Node B (*i.e.* the base station) when it demotes from DCH to FACH. It receives a *physical channel reconfiguration* (PCR) message upon demoting from FACH to PCH. By measuring the timestamp difference between the last IP packet and the corresponding RBR or PCR message we confirm the inferred RRC demotion timer values.

Second, we wished to investigate the root causes of behavior observed at the UDP layer. In particular, we wished to uncover the relationship between delays during demotions to and from FACH and RLC-layer transient states. To do so, we ran the RRC inference test for 160 consecutive hours and collected QxDM logs as well as server-side tcpdump [4] traces. We labelled 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. In that case, we skip ahead by $S_{PDU} \times D$ instead. We know our mapping is complete if every PDU's first byte maps to 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 2 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 the *QxDM_TCP_Trace* and the *QxDM_TCP_Trace*. We were able to correctly map 99.8% of packets.

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

Algorithm 2 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
```

logs.

Each RLC sequence number uniquely identifies each RLC PDU, so we can determine RLC retransmissions based on duplicate sequence numbers. One complication is that sequence numbers wrap around every 4096 PDUs. To avoid over-counting duplicate sequence numbers from a different cycle, a 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 [8].

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 retransmission 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 §3) by counting retransmissions over a certain inter-packet time transmission period.

5. EVALUATION RESULTS

First, we describe our findings from our client-based RRC inference approach applied to devices applied to devices in controlled experiments, and next describe how we used QxDM-based cross-layer analysis to verify our findings and investigate root causes of our observations, in particular the impact of transient states. Next, in §5.2 we describe the results of the public deployment of our inference tool to examine the representativeness of our findings for RRC states worldwide. Our results are summarized in Table 3. For privacy reasons, we anonymize all data that might identify users when collecting data.

5.1 Controlled Experiments

We collected three types of data from our controlled experiments: results from a small number of test devices on various carriers, to develop and validate our analysis methodology;results from a deployment within T-Mobile, to measure RRC states and performance characteristics; and results from QxDM traces to validate theT-mobile results and investigate root causes of the phenomena we detected. We primarily discuss the second two sets of results in this section.

The inference measurements from theT-mobile deployment are shown in Table 4. These timers have changed since being measured in previous work [16], underscoring the usefulness of a tool that can be used to crowdsource the taking of continuous measurements. What is of particular interest is the increase in latency when transitioning from DCH to FACH for UMTS. This latency generally lasts from 1 to 2.5s. A similar pattern was occasionally seen when promoting to DCH, but generally lasts no more than half a second. Likewise, this delay sometimes occurred in LTE when demoting to RRC_IDLE, although it was only observed in our public deployment. An example of this phenomenon can be seen in Figure 4. This delay is of particular concern because the associated RTTs are often significantly longer even than those in PCH. In some cases, the user derives no benefit from the low-power FACH state at all, as this phenomenon lasts long enough that the expected FACH behavior is never observed.

We examined the impact of RRC state on several higher-

§	Figures & Tables	Data Source	Description
5.1	Table 4	Controlled experiments using T-Mobile devices	RRC state timers inferred
5.1	Figure 12	Controlled experiments using T-Mobile devices	Unexpected high-latency transitions detected; behavior device-dependent
5.1	Table 5	Controlled experiments using T-Mobile devices	Impact on higher-layer protocols measured. DNS, TCP handshakes affected by RRC state; HTTP independent
5.1.1	Table 4	*QxDM_TCP_Trace* & *QxDM_UDP_Trace*	RRC state timer validation in QXDM
5.1.1	Figure 6, Figure 7, Figure 8	*QxDM_TCP_Trace* & *QxDM_UDP_Trace*	Root cause analysis of unexpected latencies using cross-layer mapping mechanism
5.1.1	Figure 9, Figure 6	*QxDM_UDP_Trace*	Analyze root cause of UDP packet loss behavior
5.2	Figure 10	Public deployment	Inferred RRC implementations globally; varies by carrier
5.2	Figures 12 and Figure 13	Public deployment	Confirm delays and losses during state transitions a common problem
5.2	Figure 11	Public deployment	Device-dependence of RRC performance

Table 3: Summary of key results

Network Type	DCH/ RRC CONNECTED (s)	FACH (s)	Transition delay?	# devices observed
UMTS	3	2	Yes*	12
HSDPA	3	None	No	1
LTE	10	None	No	3
EDGE	2	None	No	3
QxDM Ground Truth				
UMTS	3.2	2.3	Yes	2

* For a subset of devices

Table 4: Inferred timers for controlled experiments on a major US carrier; ground truth for UMTS from QxDM

Protocol	DCH (ms)	FACH TRANSITION (ms)	FACH (ms)	PCH (ms)
DNS lookup (UDP)	449 ± 881	1656 ± 1157	776 ± 948	1209 ± 1350
TCP handshake	229 ± 191	1524 ± 1371	1114 ± 1096	936 ± 674
HTTP	3120 ± 6465	3530 ± 4905	2926 ± 1609	3338 ± 6031

Table 5: Comparison of 3G performance across different RRC states for higher-level protocols. FACH TRANSITION refers to a period of high latency when packets are sent during promotions to FACH due to RLC-layer transient state behavior. Note RRC state dependence of DNS and TCP in particular.

level protocols as well. We investigated the time to complete a DNS lookup, a TCP handshake, and the loading of a small webpage over HTTP to illustrate the impact of promotion overhead. For the DNS test, we generated a random, invalid URL, as we found that on some devices the DNS cache could not be flushed. For TCP, we measured the time to perform a TCP handshake with `www.google.com`, and we measured the time to access `www.google.com` through HTTP. The results are summarized in Table 5.

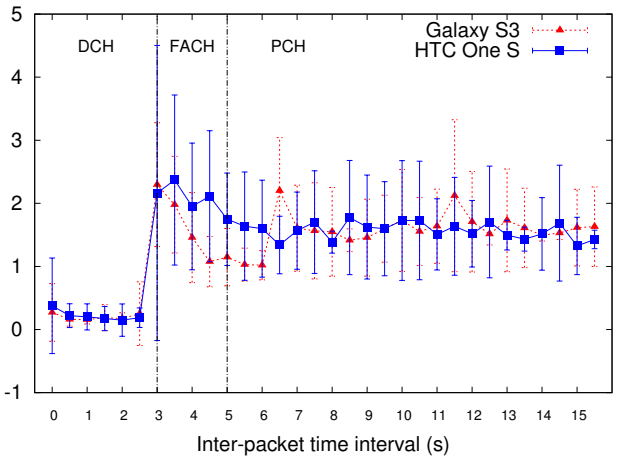
In general, there was a very high variance for the data collected in each of the states. TCP handshakes and DNS lookups, like raw UDP packets, appear to be affected by the RRC state of the initial packet sent, in large part because these tests involve few packets. For HTTP, however, the impact of the starting RRC state was not statistically significant. This is also unsurprising, as the RRC state only affects the first packet sent. FACH_TRANSITION appears to have a performance impact, however. RRC state would primarily impact frequent, low-bandwidth transmissions, such as applications that perform periodic polling, as observed in previous work [17].

5.1.1 Cross-Layer Root-Cause Analysis with QxDM

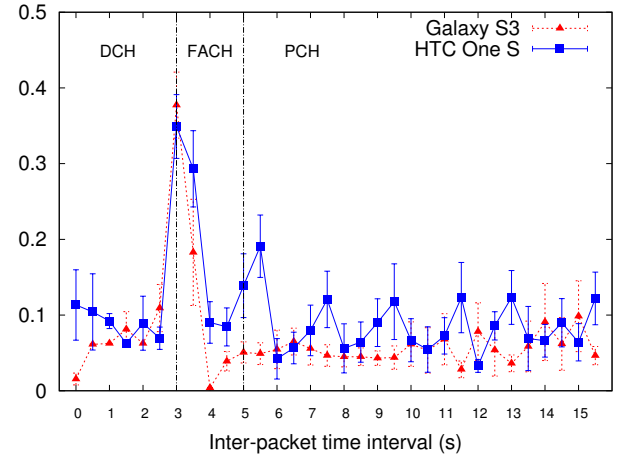
We first validate the inferred RRC timers using the method described in §4.1. The time difference between the last IP packet and the associated RBR message is around 3.2 seconds. Similarly, the PCH demotion timer is approximately 2.3 seconds, based on the time difference between the last IP packet in FACH and the associated PCR message. The first timer result matches the one in Table 4 exactly, rounding to the nearest half-second (as that is the granularity at which the measurements were taken). The second timer matches closely as well.

Next, 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*. Making use of the terminology defined in §2, we examine each of the *transient states* at the RLC layer to understand the root causes of this behavior. Based on our measurements of the FACH and PCH promotion times, we determine that FACH_PROMOTE covers the time period of 0.92 seconds before promotion to DCH occurs in *QxDM_UDP_Trace* and *QxDM_TCP_Trace*. 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 inter-packet timing after we apply the cross-layer mapping algorithm from §4.2. In Figure 6(b) the high retransmission ration in FACH can be seen, which matches the latency behavior in Figure 6(a). The similar patterns imply that the root cause of the FACH latency issue resides in the data link layer.



(a) UDP RTT Result



(b) RLC Retransmission Result

Figure 6: RLC layer retransmission ratio has a strong correlation with the delay over FACH state. Measurements on 1 KB packets.

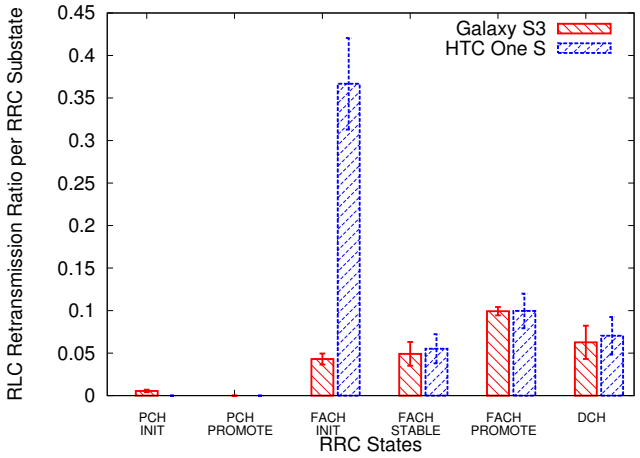


Figure 7: The significant RLC retransmission ratio over the stable FACH state and FACH promotion state is consistent with higher-layer measurements and suggests that FACH_INIT has a significant impact on the FACH transition delay.

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 [16]. 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 transient state. We show these results in Figure 7. The retransmission ratios are particularly high in FACH_INIT and FACH_PROMOTE. For theHTC device, the ratio is especially high during FACH_INIT. These observations are consistent with the abnormal delays we observed when transitioning from DCH to FACH at the UDP layer. The higher loss rates for the device made byHTC are consistent with the differences between devices found in the public study in §5.2. A possible explanation is the difference in hardware used.The HTC device uses Snapdragon S3 MSM8260, while the Galaxy S3 contains the next-generation Snapdragon S4 MSM8960 and supports a larger range of network types [11]. This chipset dependency might also explain why this phenomenon was not observed in earlier work, on older devices, which likely used yet another chipset. In our public study we found that there are a number of chipsets from various companies that exhibit this behavior. However, a detailed examination of hardware differences is out of the scope of the paper.

We also examined performance in TCP. A TCP *retransmission timeout* (RTO) can cause a RTT delay as the congestion window size decreases and the device falls back to the slow start phase [28]. 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 8). 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 [8]. 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

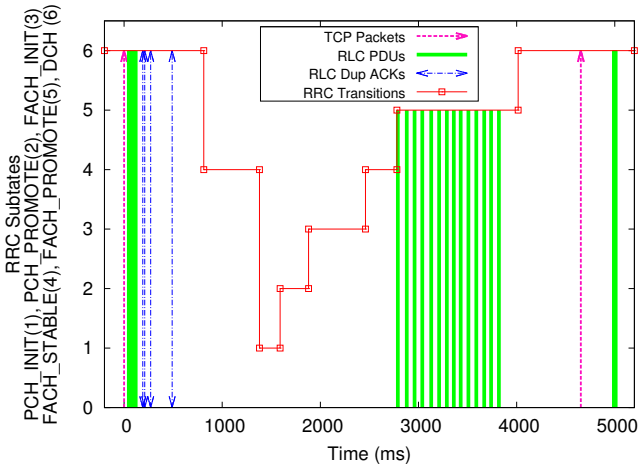


Figure 8: TCP RTO is caused by delays in RLC PDU retransmission. Duplicate ACKs suggest a PDU has been lost. By responding to this probable loss in a timely manner, RLC transmission latency can be reduced and sometimes TCP RTOs can even be avoided.

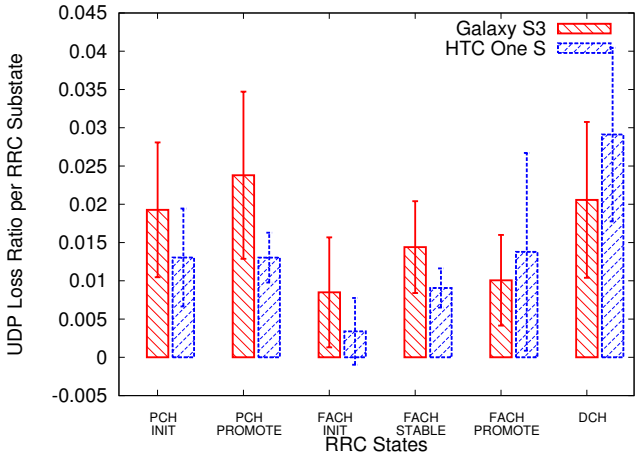


Figure 9: UDP loss dependence on the device and on RRC transition states

	OTA (%)		Wired (%)
	Reset	**Exceed Limit**	
Samsung SIII	3.67±3.14	0.57±1.01	95.77±3.01
HTC One	3.32±1.31	1.17±1.16	95.56±1.37

Table 6: Most UDP packet losses happen in the wired network, not the OTA (or wireless) channel.

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 transport layer. We evaluate this mechanism in §6

Finally, we examine UDP losses in each RRC state and identify root causes of these losses. We label each packet with a unique "sequence number" to map UDP packets sent from the client to the server-side tcpdump trace — a loss occurs when a packet never appears at the server. We consider the RRC state or RLC-layer transient state associated with each packet to be the state when the packet is sent, as defined in Table 1. Packet loss ratios are the ratios between the number of packets lost from each state and those sent. As can be seen in Figure 9, these losses are highly device-dependent as well, with the HTC device having a higher loss ratio in DCH and FACH, and theSamsung device being more lossy in PCH. As devices perform most transmissions in DCH and FACH, the Samsung device is less lossy overall.

To identify the root cause of these UDP losses, we use the cross-layer mapping algorithm to understand RLC-layer transient state behavior. We wish to understand if the UDP packets are getting lost over the OTA link or the wired channel as defined in §2. Since QxDM only captures the data transmission in UTRAN, SDU losses not found in QxDM traces are lost over the wired channel. According to the 3GPP specification [8], RLC PDUs are lost between the handset and the Node-B due either to senders retransmitting the PDUs more times than a pre-defined limit or due to the sender being reset by the receiver. After mapping each UDP PDU to the corresponding RLC PDU, we check if the sender received a reset control PDU from the receiver or if the number of retransmitted PDUs exceeds a predefined limit. If neither case applies, we know that the the UDP packet was lost over the internet. Most UDP packets are lost over the wired channel, as can be seen in Figure 6. This is due to the RLC layer ARQ (Automatic Repeat reQuest) mechanism that limits losses in the RLC layer.

5.2 Public Deployment of the Inference Tool

To get a broader and more representative set of data, we integrated the coarse-grained RTT measurement functionality into Mobiperf [3], an existing network performance measurement application. We did not collect data on the impact of RRC state on higher-level protocols, however.

130 devices on 41 carriers in 25 countries have installed the test, based on hashed phone identifiers (to preserve user anonymity). However, we only have complete data on 27 devices, covering 9 countries, 16 carriers and 20 distinct device models from 6 manufacturers.

We excluded devices with less than three tests run in order to ensure we could effectively filter noise. Second, a number of devices never uploaded accurate data on the network technology used — either because they never updated from the original application version that did not collect this data, or because their device always returns the value "UNKNOWN". We exclude these devices from any findings that require knowing the specific network technology. We therefore focus on 27 devices with complete data in this section — these cover 9 countries, 16 carriers and 20 distinct device models from six manufacturers.

In this section, as we are observing higher-level behavior, we do not discuss the transient states directly. Instead, we refer to the behavior caused by delays during transient states, resulting in UDP-layer delays when sending packets during the transient states associated with a state demotion, as FACH_TRANSITION or LTE_TRANSITION. The former

refers to delays when demoting to FACH, and the latter to delays when demoting to RRC_IDLE.

First, we examine the inferred RRC states for the devices for which data has been collected and examine how these states vary across carriers. For LTE, we have data from two carriers, both within the U.S. Both have T_{Tail} timers of 10 seconds. Some devices exhibited a spike in latency when transitioning between states. There appears to be some device dependency for this effect under LTE as well. A comparison of the latencies and packet losses for two different devices using the same carrier and the same RRC state transition timers can be seen in Figure 11.

The UMTS timers are more varied, and we have more data on a wider range of carriers than for LTE. The timers are summarized in Figure 10. Each distinct RRC state machine for a carrier is counted only once regardless of the number of devices collecting data. Timers are generally closer to 3 seconds when FACH is used and 7 seconds when it is not. We do not in general have enough data to compare carriers across devices or locations, but when we do, state machine timers are consistent within one second. However, we found that one carrier which uses two different UMTS versions in two different locations has two different sets of timers. We found that devices usually enter FACH both during demotion and promotion, or not at all, with one exception where the device promotes directly to DCH but enters FACH during demotion.

Substantial delays when demoting from DCH to FACH were common. It is less immediately clear that this is device-dependent behavior, though, for two reasons. First, there are more carriers in this dataset but few devices per carrier, so comparing different devices with the same carrier is often not possible. We did observe for one carrier, Verizon, that not all devices exhibited this behavior, however. Secondly, there were two cases where, for a specific model of phone, some devices exhibited this behavior and some did not. After some investigation, it seems that different chipsets are used in different markets for the same model of phone, which might explain this discrepancy. We also determined that several different chipsets from several different manufacturers exhibit this behavior, so the problem is not due to a single poor implementation. However, without more information about each device we cannot be confident in our claim that this is entirely device-specific behavior.

We also examined the performance characteristics associated with different RRC states. As most of our data was for LTE or UMTS, we focus on these two technologies. For UMTS, we collected data from 12 devices. Between 16 and 143 tests were run per device (with an average of 70 tests run). We excluded a large number of devices with less than three tests each — the distribution of measurements per device is very irregular. For LTE, we used 9 devices with an average of 34 tests each, ranging from 4 to 110 tests.

We first examined the average promotion delays associated with different RRC states, shown in Figure 12. These

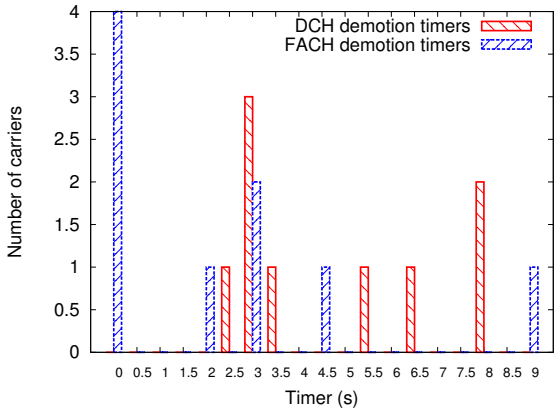


Figure 10: Distribution of RRC 3G timers, for nine UMTS RRC state machines. Note cluster around 2-3.5s.

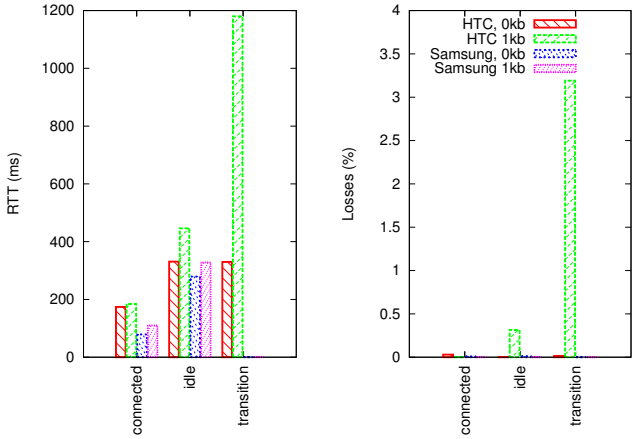


Figure 11: RTTs and loss rates for two device models on the same LTE network, based on 39 tests for the Samsung SIII and 94 for the HTC One. Note that only the HTC device exhibits an additional delay around state demotion. The "Transition" data is during a period of high latency when sending data around a demotion to RRC_IDLE.

are calculated based on the difference in RTT between the highest-power, lowest-latency state and every other RRC state. We see that the pattern we observed for thecontrolled T-mobile dataset holds here as well. Furthermore, LTE generally performs better than 3G. It can also be seen here that latencies when transitioning between DCH and FACH are quite substantial, and often higher than in PCH.

We also examined trends in packet loss in different RRC states for these carriers, as shown in Figure 13, again comparing 3G and LTE. Unsurprisingly, losses were generally low in high-power states (DCH or RRC_CONNECTED) and are high in low-power states. This is especially true in LTE. Although in LTE the state transition delay results in less of an increase in latency, it does appear to result in relatively high packet loss. Furthermore, FACH does not perform well in terms of packet loss either.

Overall, it can be seen that the phenomena we have

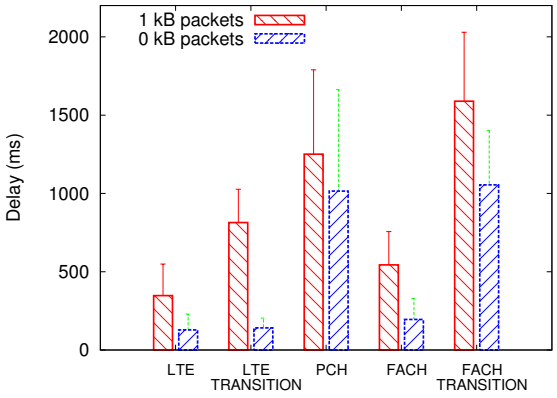


Figure 12: Promotion delays for each state for all carriers surveyed, calculated by subtracting the RTT in the highest-power state from the RTTs in states where promotions are required. The anomalous behavior FACH_TRANSITION and LTE_TRANSITION, when it occurs, results in significant delays.

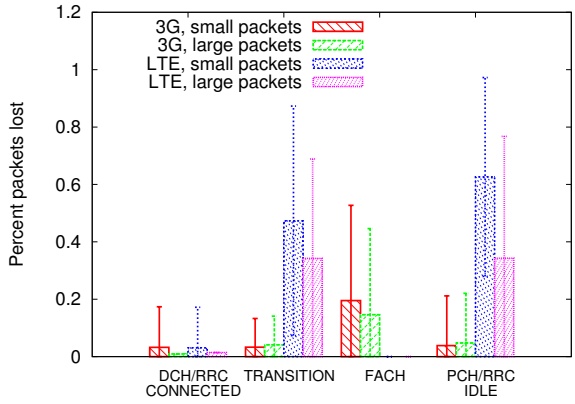


Figure 13: Packets lost per state as a percent. Note high losses during transition and RRC_IDLE for LTE.

observed in our controlled experiments onT-Mobile devices are of more universal relevance, and more generally that our client-based inference methodology is an effective way to understand trends in RRC state implementations and performance more broadly. Our inference technique has additionally allowed us to measure the performance characteristics of networks on consumer devices, without requiring any expert knowledge to collect the needed data.

6. PROPOSED RLC FAST RETRANSMISSION

As we have shown in §5.1.1, 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 packet 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 [8], 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 [25], 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 [8].

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 14(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 14(a). "Draw_Minus" occurs when less than 50% of the PDUs retransmitted through our proposed mechanism would have been retransmitted anyway — unnecessary retransmissions have occurred. This case is shown in Figure 14(c). If all PDUs were successfully delivered to the receiver, we refer to this case as "Loss", as shown in Figure14(d). Redundant data was transmitted and there were no latency gains.

As we can see from Table 7, 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 [16]. 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%*

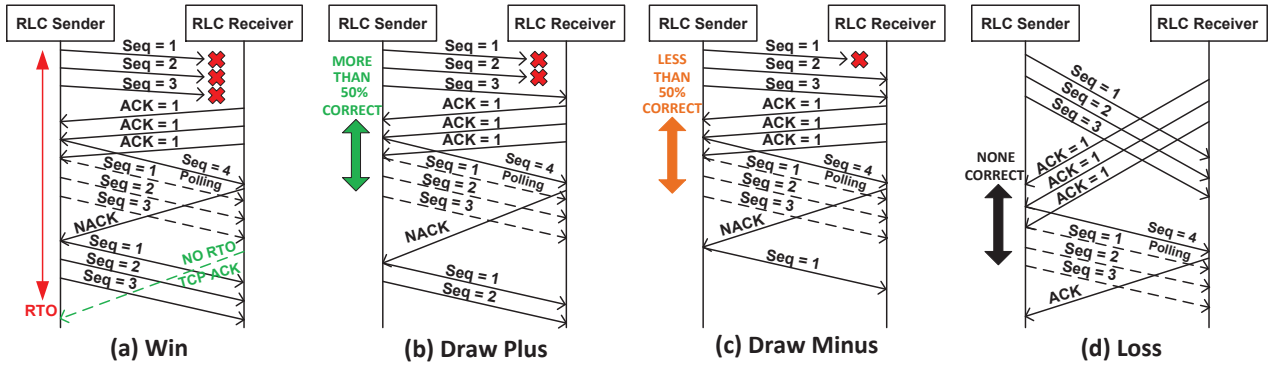


Figure 14: *Win*: RLC Fast Re-Tx avoids a TCP RTO. *Draw Plus*: Prediction accuracy is more than 50%. *Draw Minus*: Prediction accuracy is less than 50%. *Loss*: All predictions are wrong.

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 7: 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_TRANSITION. 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 transition period.

7. RELATED WORK

Previous work has been done on measuring the power and performance characteristics of 3G RRC state machines [16] as well as 4G LTE networks [19, 22] in controlled environments, as well as specific features of those networks such as DRX [21]. Work has also been done on exploring the implications on application performance of RRC state and inadequacies in how mobile applications deal with RRC state [17].

We expand upon this work by implementing an RRC inference method that can be integrated into a measurement tool intended for non-experts. It can automatically infer the RRC state machine of any network type, allowing anomalous or unexpected behavior to be measured, unlike previous work that assumes that devices follow the ideal behavior defined in the specification. We also focus on *transient states* we observe at the RLC level related to promotions and demotions, and their impact on higher-layer performance. Work by Souders [27] measures RRC state machines on client devices, but being a browser-based solution it is not able to account for background activity on the phone or varying network types, measure performance data as accurately as an application can, or observe RLC-level behavior.

Other work has been done on optimizing the use of the cellular network based on 3G or 4G specific phenomena. RadioJockey [24] investigates how to effectively trigger fast dormancy based on network traffic patterns, and TailEnder [23] and work by Deng et al. [26] propose a method of scheduling data transfers to minimize energy consumption without impacting user-perceived performance.

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 [20]. 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.

More generally, work has been done to measure performance characteristics of cellular networks and networks on mobile devices. The Livelab project [14] also makes use of users running a measurement app on their phones. A wide range of findings on how users interact with mobile devices have been published, including measuring web usage in the wild [13]. Work by Halepovic et al., [15] presents a method of passively measuring HTTP transaction latency. Work by Gember et al., [12] determines how to accurately measure user-perceived performance on user devices. JamLogger [6] is an ongoing project to collect general performance and user activity on mobile devices. Our work is complementary to these studies.

8. CONCLUSION

In this paper, we have presented new techniques for detecting and analyzing RRC state behavior and performance. We have created a tool suitable for automatically inferring RRC state machines on end-user devices without requiring expert knowledge to run. We have demonstrated that it is

effective in understanding RRC state implementations and their performance characteristics and how these vary by carrier and device model. We have also shown that this tool is effective at uncovering previously unknown anomalous behavior, in particular that related to state transitions. Finally, we used RLC-level analysis and a novel cross-layer analysis technique to uncover some underlying causes of these unexpected phenomena, and show how they relate to RLC-layer *transient states*. We proposed RLC *Fast Re-Tx* mechanism that reduces the RLC latency up to 35.69% during FACH_TRANSITION.

9. REFERENCES

[1] 3GPP LTE. http://www.3gpp.org/LTE.
[2] 3GPP UMTS (3G). http://www.3gpp.org/UMTS.
[3] MobiPerf. http://mobiperf.com.
[4] tcpdump. http://www.tcpdump.org/.
[5] Configuration of fast dormancy in release 8. 3GPP discussion and decision notes RP-090960, 2009.
[6] NU JamLogger: A Study of User Activity and System Performance on Mobile Architectures. http://www.ece.northwestern.edu/microarchitecture/jamlogger/, 2009.
[7] QxDM Professional Proven Diagnostic Tool for Evaluating Handset and Network Performance. http://www.qualcomm.com/media/documents/files/qxdm-professional-qualcomm-extensible-diagnostic-monitor.pdf, 2012.
[8] 3GPP TS 25.322: Radio Link Control (RLC) - UMTS, 2013.
[9] 3GPP TS 35.331: Radio Resource Control (RRC) - UMTS, 2013.
[10] 3GPP TS 36.331: Radio Resource Control (RRC) - LTE, 2013.
[11] Snapdragon (System on Chip). http://en.wikipedia.org/wiki/Snapdragon_(system_on_chip), 2013.
[12] A. Gember, A. Akella, J. Pang, A. Varshavsky, and R. Caceres. Obtaining Representative Measurements of Cellular Network Performance. 2012.
[13] C. C. Tossell, P. Kortum, A. Rahmati, C. Shepard, and L. Zhong. Characterizing web use on smartphones. 2012.
[14] C. Shepard, A. Rahmati, C. Tossell, L. Zhong, and P. Kortum. LiveLab: Measuring Wireless Networks and Smartphone Users in the Field. 2010.
[15] E. Halepovic, J. Pang, and O. Spatscheck. Can you GET Me Now? Estimating the Time-to-First-Byte of HTTP Transactions with Passive Measurements. 2012.
[16] F. Qian, Z. Wang, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck. Characterizing Radio Resource Allocation for 3G Networks. In *Proc. ACM IMC*, 2010.
[17] F. Qian, Z. Wang, A. Gerber,Z. M. Mao, S. Sen, and O. Spatscheck. Profiling Resource Usage for Mobile Applications: A Cross-layer Approach. In *Proc. ACM MobiSys*, 2011.
[18] H. Holma, and A. Toskala. *WCDMA for UMTS: HSPA Evolution and LTE*. John Wiley and Sons, Inc., 2007.
[19] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck. A Close Examination of Perfomance and Power Characteristics of 4G LTE Networks. In *Proc. ACM MobiSys*, 2012.
[20] J. J. Alcaraz, F. Cerdn, and J. Garca-Haro. Optimizing TCP and RLC interaction in the UMTS Radio Access Network. 2006.
[21] J. Wigard, T. Kolding, L. Dalsgaard, and C. Coletti. On the User Performance of LTE UE Power Savings Schemes with Discontinuous Reception in LTE. 2009.
[22] L. Zhou, H. Xu, H. Tian, Y. Gao, L. Du, and L. Chen. Performance Analysis of Power Saving Mechanism with Adjustable DRX Cycles in 3GPP LTE. 2008.
[23] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani. Energy Consumption in Mobile Phones: A Measurement Study and Implications for Network Applications. 2009.
[24] P. K. Athivarapu, R. Bhagwan, S. Guha, V. Navda, R. Ramjee, D. Arora, V. N. Padmanabhan, and G. Varghese. RadioJockey: Mining Program Execution to Optimize Cellular Radio Usage. 2012.
[25] Q. Xu, S. Mehrotra, Z. M. Mao, and J. Li. PROTEUS: Network Performance Forecast for Real-Time, Interactive Mobile Applications. In *Proc. ACM MobiSys*, 2013.
[26] S. Deng, and H. Balakrishnan. Traffic-Aware Techniques to Reduce 3G/LTE Wireless Energy Consumption. 2012.
[27] S. Souders. Making a mobile connection. http://www.stevesouders.com/blog/2011/09/21/making-a-mobile-connection/, 2011.
[28] V. Paxson, and M. Allman . Computing TCPs retransmission timer, 2000.