Characterizing Radio Resource Allocation for 3G Networks

Feng Qian
University of Michigan

Zhaoguang Wang
University of Michigan

Alexandre Gerber
AT&T Labs Research

Z. Morley Mao
University of Michigan

Subhabrata Sen
AT&T Labs Research

Oliver Spatscheck
AT&T Labs Research

ABSTRACT

3G cellular data networks have recently witnessed explosive growth. In this work, we focus on UMTS, one of the most popular 3G mobile communication technologies. Our work is the first to accurately infer, for any UMTS network, the state machine (both transitions and timer values) that guides the radio resource allocation policy through a light-weight probing scheme. We systematically characterize the impact of operational state machine settings by analyzing traces collected from a commercial UMTS network, and pinpoint the inefficiencies caused by the interplay between smartphone applications and the state machine behavior. Besides basic characterizations, we explore the optimal state machine settings in terms of several critical timer values evaluated using real network traces. Our findings suggest that the fundamental limitation of the current state machine design is its *static* nature of treating all traffic according to the same inactivity timers, making it difficult to balance tradeoffs among radio resource usage efficiency, network management overhead, device radio energy consumption, and performance. To the best of our knowledge, our work is the first empirical study that employs real cellular traces to investigate the optimality of UMTS state machine configurations. Our analysis also demonstrates that traffic patterns impose significant impact on radio resource and energy consumption. In particular, We propose a simple improvement that reduces YouTube streaming energy by 80% by leveraging an existing feature called *fast dormancy* supported by the 3GPP specifications.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design – Wireless Communication; C.4 [**Performance of Systems**]: Measurement Techniques

General Terms

Measurement, Algorithms

IMC'10, November 1–3, 2010, Melbourne, Australia.

Keywords

UMTS, 3G Networks, RRC state machine, Inactivity timers, Tail effects, Smartphones, Multimedia streaming

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 UMTS (the Universal Mobile Telecommunications System) 3G network, which is among the most popular 3G mobile communication technologies. As evidence of this popularity, in Q3 2008, among the 400 million of worldwide 3G users, 300 million were UMTS subscribers [1].

Compared to WiFi, 3G systems operate under more radio resource constraints. To efficiently utilize the limited radio resources, UMTS introduces for each *user equipment* (UE, *i.e.,* a smartphone) a *radio resource control* (RRC) state machine that determines radio resource usage affecting device energy consumption and user experience. Usually a UE (user equipment) can be 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. Frequent state promotions (resource allocation) may lead to unacceptably long delays for the UE, as well as additional processing overheads for the radio access network [10, 21]. State demotions (resource release) are controlled by critical inactivity timers affecting radio resource utilization and UE energy consumption.

The current design of the RRC state machine appears to be ad-hoc with statically configured parameters. Our work systematically studies its design using real cellular traces from a large cellular ISP and analyzes the effect on important factors from both the network operator's perspectives, namely radio resource usage efficiency and management overhead, and from end-user's perspective, namely device energy consumption and application performance. We examine tradeoffs among these factors. In particular, we focus on settings of critical inactivity timer values that determine when to release radio resources after a period of inactivity.

As an example of the challenge in balancing the tradeoffs, using real UMTS cellular traces we observe that decreasing one inactivity timer by three seconds can reduce the overall radio resource usage by 40%, but increasing the number of state promotions by 31%. On the other hand, increasing the inactivity timer effectively enhances end user experience and reduces the management overhead, however at the expense of low efficiency in radio resource utilization and en-

ergy consumption. Intuitively, the optimal timer settings heavily depend on application traffic patterns and thus can benefit from *traffic awareness* via trace-driven tuning.

In this paper, we undertake a detailed exploration of the RRC state machine and its optimizations using real traces from a large cellular ISP. In particular, we make the following contributions.

1. Accurate inference of the RRC state machine. Different carriers may adopt different state machine models with varied parameters. Accurate inference is therefore the very first necessary step towards characterizing and improving the RRC state machine. We propose a novel inference technique purely based on probing from the user device. It systematically discovers the state transitions by strategically adjusting the packet dynamics. We applied our algorithm to two UMTS carriers and validated its accuracy by measuring the device power consumption.

2. Characterization of state machine behaviors. The current RRC state machine parameters are either empirically configured in an ad-hoc manner [7], or determined using analytical traffic models [20, 32]. The latter approach, however, suffers from several limitations. *(i)* The expressiveness of an analytical model is quite limited and is unlikely to capture, using a statistical distribution with a few parameters, the characteristics of real-world traffic patterns generated by millions of cellular users. *(ii)* The existence of concurrent applications accessing the network further increases the difficulty of modeling the packet dynamics.

We systematically characterize the impact of existing operational state machines by analyzing traces collected from a commercial UMTS network. We found that short data transfers severely suffer from the state promotion delay, and significant portions (up to 45.3%) of the occupation time of the high-speed dedicated transmission channel is wasted on the idle time period matching the inactivity timer value before a state demotion, which is called *tail* time [14]. We explore the optimal timer values by replaying the trace against different state machine settings with varying parameters. Our findings suggest that the fundamental limitation of the current state machine design is its *static* nature of treating all packets according to the same inactivity timer, making it difficult to balance the aforementioned tradeoffs. We also observe that applications exhibit different sensitivities to the change of inactivity timers due to their different traffic patterns. To the best of our knowledge, our work is the first empirical study that employs real cellular traces to investigate the optimality of RRC state machine configurations.

3. Analysis of multimedia streaming strategies. We study the streaming approaches employed by Pandora audio [5] and YouTube video streaming, the two most popular smartphone multimedia streaming applications contributing large traffic volume. Our analysis demonstrates that traffic patterns impose significant impact on the radio resource and energy consumption, again due to the interplay between the UE application and the RRC state machine. The current Pandora approach incurs long tail periods, which waste 50% of the dedicated channel time and 59% of the radio energy, while the YouTube strategy suffers from long dedicated channel occupation time due to bandwidth underutilization. We propose a simple improvement that saves the YouTube streaming energy by 80% by leveraging an existing feature called *fast dormancy* supported by 3GPP specifications [8, 9].

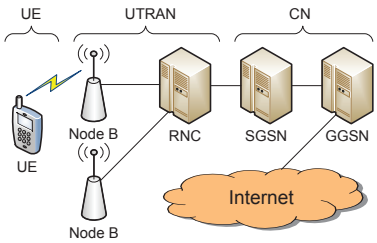


Figure 1: The UMTS architecture

4. Data preprocessing scheme for cellular data traces. We are the first to highlight the need and address the challenge of preprocessing cellular network traces, which differ from Internet traffic data in that cellular data can reflect significant delays imposed by state promotions transitioning from low power to higher power state due to radio resource allocation overhead. To extract the actual traffic patterns, *e.g.,* for the purpose of evaluating alternate designs of the RRC state machine, such delays must be eliminated.

2. BACKGROUND

This section provides sufficient background for further discussions of this paper.

2.1 The UMTS Network

As illustrated in Figure 1, the UMTS network consists of three subsystems: User Equipments (UE), UMTS Terrestrial Radio Access Network (UTRAN), and the Core Network (CN). UEs are essentially mobile handsets carried by end users. The UTRAN allows connectivity between a UE and the CN. It consists of two components: base stations, called Node-Bs, and Radio Network Controllers (RNC), which control multiple Node-Bs. Most UTRAN features such as packet scheduling, radio resource control, and handover control are implemented at the RNC. The centralized CN is the backbone of the cellular network. In particular the GGSN (Gateway GPRS Support Node) within the CN serves as a gateway hiding UMTS internal infrastructures from the external network.

2.2 The RRC State Machine

In the context of UMTS, the *radio resource* refers to WCDMA codes that are potential bottleneck resources of the network. To efficiently utilize the limited radio resources, the UMTS radio resource control (RRC) protocol introduces a state machine associated with each UE. There are typically three RRC states as described below [25, 19].

IDLE. This is the default state when a UE is turned on. The UE has not yet established an RRC connection with the RNC, thus no radio resource is allocated, and the UE cannot transfer any user data (as opposed to control data).

CELL_DCH. The RRC connection is established and a UE is usually allocated dedicated transport channels in both downlink (DL, RNC to UE) and uplink (UL, UE to RNC) direction. This state allows a UE to fully utilize radio resources for user data transmission. We refer to CELL_DCH as DCH henceforth. A UE can access HSDPA/HSUPA (High Speed Downlink/Uplink Packet Access) mode, if supported by the infrastructure, at DCH state. For HSDPA, the high speed transport channel is not dedicated, but shared by a limited number (*e.g.,* 32) of users [19]. Further, when a large

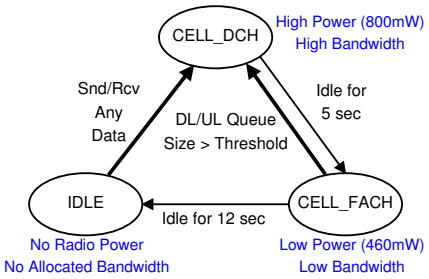


Figure 2: The RRC state machine for the 3G UMTS network of Carrier 1

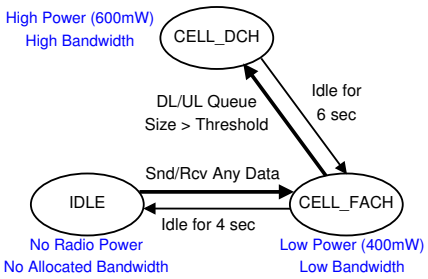


Figure 3: The RRC state machine for the 3G UMTS network of Carrier 2

Table 1: Inferred parameters for two carriers

Inactivity timer	Carrier 1	Carrier 2
α: DCH→FACH	5 sec	6 sec
β: FACH→IDLE	12 sec	4 sec
Promotion time	Carrier 1	Carrier 2
IDLE→FACH	N/A	0.6 sec
IDLE→DCH	2 sec	N/A
FACH→DCH	1.5 sec	1.3 sec
RLC Buffer threshold	Carrier 1	Carrier 2
FACH→DCH(UL)	543 ± 25 B	151 ± 14 B
FACH→DCH(DL)	475 ± 23 B	119 ± 17 B
State radio power	Carrier 1	Carrier 2
DCH/FACH/IDLE	800/460/0 mW	600/400/0 mW
Promotion radio power	Carrier 1	Carrier 2
IDLE→FACH	N/A	410 mW
IDLE→DCH	550 mW	N/A
FACH→DCH	700 mW	480 mW

Table 2: Optimize radio resources: the key tradeoff

Increase α or β timers	**Decrease** α or β timers
ΔD increases	ΔD decreases
Increase tail time (§5.2)	*Decrease* tail time
Waste radio resources	*Save* radio resources
ΔS decreases	ΔS increases
Reduce state promotions	*Increase* state promotions
Reduce RNC overhead	*Increase* RNC overhead
Improve user experiences	*Degrade* user experiences
ΔE increases	ΔE decreases
Waste UE radio energy	*Save* UE radio energy

number of UEs are in DCH state, the radio resources may be exhausted due to the lack of channelization codes in the cell. Then some UEs have to use low-speed shared channels although their RRC states are still DCH.

CELL_FACH. The RRC connection is established but there is no dedicated channel allocated to a UE. Instead, the UE can only transmit user data through shared low-speed channels that are typically less than 15kbps. We refer to CELL_FACH as FACH from this point on. FACH is designed for applications requiring very low data throughput rate. It consumes much less radio resources than DCH does.

RRC states impact a UE's energy consumption. A UE at IDLE consumes almost no energy from its radio interface. The radio power consumption for DCH is 50% to 100% higher than that for FACH (Table 1). While within the same state, the radio power is fairly stable regardless of the data throughput when the signal strength is stable. Further, the RRC state machine is maintained at both the UE and the RNC. The two peer entities are always synchronized via control channels except during transient and error situations. Also note that both the downlink (DL) and the uplink (UL) use the same state machine.

In the RRC state machine, there are two types of state transitions. *State promotions*, including IDLE→FACH, IDLE→DCH, and FACH→DCH, switch from a state with lower radio resource and UE energy utilization to another state consuming more resources and UE energy. *State demotions*, consisting of DCH→FACH, FACH→IDLE, and DCH→IDLE, go in the reverse direction. Depending on the starting state, a state promotion is triggered by either any user data transmission activity, if the UE is at IDLE, or the per-UE queue size, called Radio Link Controller (RLC) buffer size, exceeding a threshold in either direction, if the UE is at FACH.

The state demotions are triggered by two inactivity timers maintained by the RNC. We denote the DCH→FACH timer as α, and the FACH→IDLE timer as β. At DCH, the RNC resets the α timer to T seconds, a fixed threshold, whenever it observes any UL/DL data frame. If there is no user data transmission activity for T seconds, the α timer times out and the state is demoted to FACH. A similar scheme is used for the β timer for the FACH→IDLE demotion.

Promotions involve more work than demotions do. In particular, state promotions incur a long "ramp-up" latency of up to 2 seconds during which tens of control messages are exchanged between a UE and the RNC for resource allocation (*e.g.,* radio bearer reconfiguration and RRC connection setup). Excessive state promotions increase the management overhead at the RNC and degrade user experience [10, 21, 28], especially for short data transfers, which we investigate in §5.1.

Figures 2 and 3 depict our inferred state machine models for two large UMTS carriers, based on our inference methodology described in §3. Their difference naturally introduces the problem of seeking the optimal state machine configuration to better balance radio resource utilization and performance. We quantitatively compare both carriers in §6.4.

2.3 Tradeoff Considerations to Optimize Resource Allocation

As discussed in §1, the RRC state machine introduces tradeoffs among radio resource utilization, UE energy consumption, end user experience, and management overheads at the RNC. We need to quantify these factors to analyze the tradeoff. Given a cellular trace and a state machine configuration C, we compute three metrics to characterize the above factors. Previous work either consider only one factor [20, 32] or focus on other metrics (*e.g.,* dropping rate due to congestion [22] and web page response time [31]), using

analytical models. We detail our methodology for computing the three metrics in §6.

- The DCH state occupation time, denoted by $D(C)$, quantifies the overall radio resources consumed by UEs on dedicated channels in DCH state. We ignore the relatively low radio resources allocated for shared low-speed channels on FACH.
- The number of state promotions, denoted by $S(C)$, is the total number of IDLE→DCH, IDLE→FACH, and FACH→DCH promotions. $S(C)$ quantifies the overhead brought by state promotions that worsen user experience and increase the management overhead at the RNC. We ignore the state demotion overhead as it is significantly smaller compared with the state promotion overhead.
- The energy consumption, denoted by $E(C)$, is the total energy consumed by radio interfaces of all UEs in the trace.

We are interested in relative changes of D, S, E when we switch to a new state machine using the same trace. Let C be the default state machine used as the comparison baseline, and let C' be a new state machine configuration. The relative change of D, denoted as ΔD, is computed by $\Delta D(C') = (D(C') - D(C))/D(C)$. We have similar definitions for ΔS and ΔE.

As we shall see throughout this paper, the key tradeoff expressed in our notations is that, for any state machine setting, increasing ΔS causes both ΔD and ΔE to decrease (there may exist exceptions when ΔS is too large). In other words, if more state promotions are allowed, then we can save more radio resources and UE energy. Ideally, we want to find a state machine configuration C' such that $\Delta D(C')$ and $\Delta E(C')$ are significantly negative, while $\Delta S(C')$ is reasonably small. This important tradeoff is summarized in Table 2.

3. INFERRING THE STATE MACHINE

We describe an end-host based probing technique for inferring the state machine, which we validate using power measurements. Accurate inference of the state machine and its parameters is the first necessary step towards characterizing and improving the RRC state machine. We study two large 3G carriers with results presented in Figures 2, 3, and Table 1, which will be used in our simulation program described in §4.

3.1 Methodology

We make the following assumptions for the inference algorithm. *(i)* There are at most three states: IDLE, FACH, and DCH. DCH is the state allowing high data rate transfer. *(ii)* The time granularity for inactivity timers is assumed to be seconds. Our algorithms can easily adapt to finer granularities. *(iii)* The state promotion delay is significantly longer than (at least two times as) a normal RTT for both DCH and FACH (less than 300 ms based on our measurements). This is reasonable due to the promotion overhead explained earlier. *(iv)* We roughly know the range of RLC buffer thresholds (64B–1KB) that trigger the FACH→DCH promotion. We detail our methodology below.

State promotion inference determines one of the two promotion procedures adopted by UMTS:
$P1$: IDLE→FACH→DCH, or $P2$: IDLE→DCH. Algorithm 1 illustrates how we distinguish between $P1$ and $P2$, where *min* and *max* denote RLC buffer sizes that does not trigger, and does trigger, the FACH→DCH promotion, respectively. Note that IDLE→DCH or IDLE→FACH always happens regardless of the RLC buffer size. The idea is to distinguish $P1$ and $P2$ by detecting the presence of the FACH→DCH promotion. We set *min* and *max* to 28 bytes (an empty UDP packet plus an IP header) and 1K bytes, respectively. If $P1$ holds, then the state is promoted to FACH after Step 2, and then further promoted to DCH at Step 3. Thus Δt includes an additional FACH→DCH promotion delay. Otherwise, for $P2$, Δt does not include the promotion delay since the state is already DCH after Step 2.

Algorithm 1 State promotion inference

1: Keep UE on IDLE.
2: UE sends *min* bytes. Server echoes *min* bytes.
3: UE sends *max* bytes. Server echoes *min* bytes.
4: UE records the RTT Δt for Step 3.
5: Report $P1$ iff $\Delta t \gg$ normal RTT. Otherwise report $P2$.

Algorithm 2 State demotion inference

1: **for** $n = 0$ to 30 **do**
2: UE sends *max* bytes. Server echoes *min* bytes.
3: UE sleeps for n sec.
4: UE sends *min* bytes. Server echoes *min* bytes.
5: UE records the RTT $\Delta t_1(i)$ for Step 4.
6: **end for**
7: **for** $n = 0$ to 30 **do**
8: UE sends *max* bytes. Server echoes *min* bytes
9: UE sleeps for n sec.
10: UE sends *max* bytes. Server echoes *min* bytes.
11: UE records the RTT $\Delta t_2(i)$ for Step 10.
12: **end for**
13: Report $D1$ iff $\Delta t_1(\cdot)$ and $\Delta t_2(\cdot)$ are similar, else report $D2$.

State demotion inference determines whether UMTS uses $D1$: DCH→IDLE or $D2$: DCH→FACH→IDLE. The inference method is shown in Algorithm 2, which consists of two experiments. The first experiment (Steps 1 to 6) comprises of 30 runs. In each run, the UE goes to DCH by sending *max* bytes (Step 2), sleeps for n seconds (Step 3), then sends *min* bytes (Step 4). Recall that *min* and *max* denote RLC buffer sizes that does not trigger, and does trigger, the FACH→DCH promotion, respectively. By increasing n from 0 to 30, we fully exercise all the states experienced by the UE at the beginning of Step 4 due to the inactivity timer effects. The second experiment (Step 7 to 12) is similar to the first one except that after Step 10, the UE always promotes to DCH. In contrast, after Step 4, if there exists a DCH→FACH demotion, the UE will be in FACH. Therefore, for $D1$ the observed RTTs for two experiments, $\Delta t_1(0..30)$ and $\Delta t_2(0..30)$, will be similar. On the other hand, for $D2$, *i.e.,* the state is demoted to FACH (the α timer), then back to IDLE (the β timer), then for $\lfloor\alpha\rfloor < i \leq \lfloor\alpha+\beta\rfloor$, the difference between $\Delta t_1(i)$ and $\Delta t_2(i)$ is roughly the FACH→DCH promotion delay.

Inferring other parameters. Given the process of inferring the state transitions, it is easy to infer related parameters. First, the inactivity timers can be directly obtained from $\Delta t_1(\cdot)$ and $\Delta t_2(\cdot)$ computed by Algorithm 2. For the case where the demotion is DCH→FACH→IDLE, we can deduce α and β from the fact that $\Delta t_1(0...\lfloor\alpha+\beta\rfloor)$ are smaller than $\Delta t_1(\lceil\alpha+\beta\rceil...30)$, and $\Delta t_2(0...\lfloor\alpha\rfloor)$ are smaller than

$\Delta t_2(\lceil\alpha\rceil...30)$. This is because in the first experiment in Algorithm 2, a state promotion (IDLE→FACH or IDLE→DCH) will not happen until $n \geq \lceil\alpha+\beta\rceil$, while in the second experiment, a state promotion from IDLE or FACH happens when $n \geq \lceil\alpha\rceil$. Similarly, for the case where the demotion is DCH→IDLE, let the only inactivity timer be γ. Then we will observe that $\Delta t_1(0...\lfloor\gamma\rfloor)$ are much smaller than $\Delta t_1(\lceil\gamma\rceil...30)$. Second, to infer the promotion delay X→Y, we measure the entire RTT including the promotion, then subtract from it the normal RTT (*i.e.,* the RTT not including the promotion) on state Y. Finally, using the promotion delay as an indicator, we infer the RLC buffer threshold by performing binary search for the packet size that exactly triggers the FACH→DCH promotion, in each direction.

3.2 Results on State Machine Inference

We present the inference results for state machines used by two large UMTS carriers: Carrier 1 and 2. For each carrier, we repeat Algorithm 1 and Algorithm 2 for three times, ensuring that in each experiment *(i)* the server does not experience a timeout; *(ii)* in tcpdump trace, we never observe other user data transmission that may trigger a state transition; *(iii)* the 3G connection is never dropped. The entire experiment is discarded if any of these conditions is violated.

For the state promotion inference, the normal RTTs for Carrier 1 and 2 are less than 0.3 sec, and the measured Δt values in Algorithm 1 are 0.2 sec for Carrier 1, and 1.5 sec for Carrier 2, for all three trials. Based on Algorithm 1, we conclude that the promotion procedures for Carrier 1 and Carrier 2 are IDLE→DCH and IDLE→FACH→DCH, respectively. For the state demotion inference, we notice the qualitative difference between $\Delta t_1(5...16)$ in Figure 4(a) and $\Delta t_2(5...16)$ in Figure 4(b), indicating that the state demotion procedure for Carrier 1 is DCH→FACH→IDLE. Similarly, Figure 5(a) and Figure 5(b) imply that Carrier 2 also uses DCH→FACH→IDLE, due to the obvious difference between $\Delta t_1(6..9)$ and $\Delta t_2(6..9)$.

We note that for Carrier 1, $\Delta t_1(17...30)$ and $\Delta t_2(17...30)$ are roughly the same, because in Algorithm 2, for $17 \leq n \leq 30$, either sending *min* bytes (Step 4) or sending *max* bytes (Step 10) triggers an IDLE→DCH promotion, which is the only promotion transition for Carrier 1. In contrast, for Carrier 2, $\Delta t_1(10...30)$ is smaller than $\Delta t_2(10...30)$. Carrier 2 may perform two types of promotions depending on the RLC buffer size, therefore for $10 \leq n \leq 30$ in Algorithm 2, sending *min* bytes in Step 4 and sending *max* bytes in Step 10 will trigger IDLE→FACH and IDLE→FACH→DCH, respectively, resulting in different promotion delays. This observation does not affect the inference results of Algorithm 2 for either carrier.

Given the $\Delta t_1(\cdot)$ and $\Delta t_2(\cdot)$ values computed by Algorithm 2, it is easy to infer α and β by following the logic described in §3.1. The inference results are $(\alpha,\beta) = (5sec, 12sec)$ for Carrier 1 and $(\alpha,\beta) = (6sec, 4sec)$ for Carrier 2. To infer the RLC buffer thresholds, we repeat the experiments 30 times and summarize the results in Table 1[1]. Although their variances are higher than those of other parameters, we found that they have very small impact on our measurement results described later.

We investigated the stability of the state machine and found that the state transitions and timer values do not change over the four months of our study (July to November 2009). They are also independent of the time of the day, the Access Point Name (APN), or the location[2]. Also the state promotion delays are largely static.

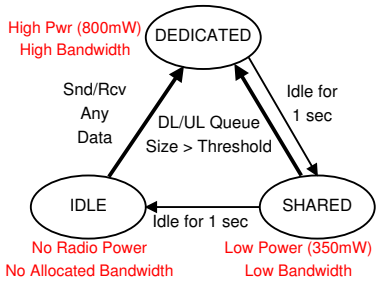


Figure 6: The RRC state machine for the 2G GPRS/EDGE network of Carrier 2

For 2G (GPRS/EDGE) networks, there exists a similar RRC state machine model. The three RRC states are "IDLE", "CELL_SHARED", and "CELL_DEDICATED" [6], corresponding to IDLE, FACH, and DCH in the 3G case, respectively. We applied our inference methodology (using a smaller increment of n in Algorithm 2) on Carrier 2's 2G network, and show the inference results in Figure 6. We observe that the inactivity timers (1 sec) are much shorter, therefore resulting in better efficiency of radio resource utilization and UE energy consumption. The negative impact of short timers is more frequent state transitions. The state promotion delays of IDLE→DEDICATED and SHARED→DEDICATED are both 0.5 sec.

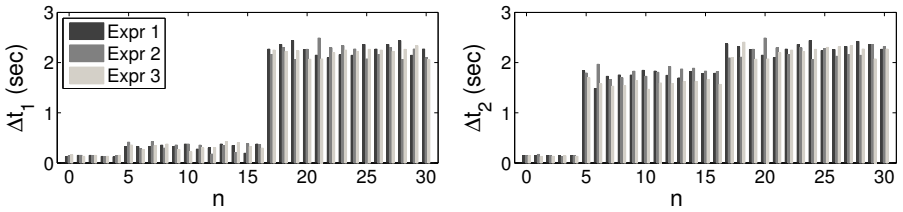
3.3 Validation using Energy Consumption

As described in §2, a UE's radio energy consumption differs for each state, a property we may use to infer the state machine. However, accurately measuring energy consumption requires special monitoring equipments. So we use it as validation for our inference algorithms, which only require UE-based probing.

We set up experiments to confirm the inactivity timers and state promotion delays for Carrier 1 by monitoring UE's energy consumption as follows. The battery of an HTC TyTN II smartphone is attached to a hardware power meter [4], which is also connected via USB to a PC that records fine-grained power measurements by sampling the current drawn from the battery at a frequency of 500 Hz. Figure 7 shows one representative experimental run of the validation. During probing, we keep the UE's LCD at the same brightness level, turn off GPS and WiFi, and disable all network activities. After keeping the smartphone in this inactive state for 20 sec, we send a UDP packet at $t = 23.8s$ thus triggering an IDLE→DCH promotion that takes approximately 2 sec as inferred in §3.1. From $t = 26.1s$, the phone remains at the high-power DCH state for about 5 sec, then switches to the low-power FACH state at $t = 31.5s$. Finally at $t = 44.1s$, the phone returns to the IDLE state. The measured inactivity timer values are longer than the inferred ones by about 10%, likely due to the synchronization overhead between the RNC and the UE. We similarly verified

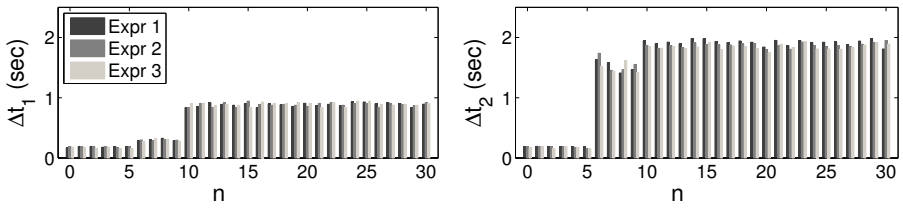
[1]Results in Table 1 were measured in November 2009.

[2]We did probing at two locations that are 600 miles apart in the U.S., using an HTC TyTN II smartphone and a Sierra 3G Air card to perform our experiment.



(a) Algorithm 2 ($\Delta t_1(\cdot)$) (b) Algorithm 2 ($\Delta t_2(\cdot)$)

Figure 4: State machine inference results for Carrier 1



(a) Algorithm 2 ($\Delta t_1(\cdot)$) (b) Algorithm 2 ($\Delta t_2(\cdot)$)

Figure 5: State machine inference results for Carrier 2

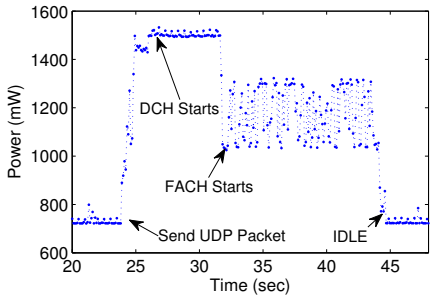


Figure 7: Validation using power measurements

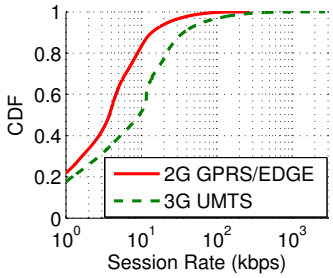


Figure 10: Session rate distribution over sessions longer than 100ms for 2G and 3G data

the FACH→DCH promotion delay, and validated Carrier 2's state machines for both 2G and 3G.

Using the IDLE power as baseline, we compute the power consumption of the 3G radio interface as shown in Table 1[3]. We also infer the RLC buffer thresholds for the FACH→DCH promotion by performing binary search. Instead of using the promotion delay as described in §3.1, we use energy as an indicator to search for the RLC buffer threshold that exactly triggers the promotion, for each direction. The validation results are consistent with our inference results.

4. 3G MEASUREMENT DATA

This section discusses the traffic data used in our study (§5 and §6). We first describe the raw dataset in §4.1, then in §4.2, we explain our preprocessing methodology to eliminate the side effects brought by the RRC state machine on the dataset. This enables us to use the preprocessed trace to experiment with different state machine configurations. In §4.3, we describe the extraction procedure for application-specific data for further analysis.

[3]The power values in Table 1 were measured in good signal strength conditions. [29] shows that signal strength may have significant impact on a UE's radio power consumption.

4.1 The Raw Dataset

Our dataset is a large TCP header packet trace collected from Carrier 1 on January 29, 2010 in the normal course of operations. The collection point is one GGSN that primarily serves 3G UMTS users but also 2G GPRS users. Non-3G traffic are filtered by excluding SGSNs that exclusively serve 2G users. Sampling was performed on a per GTP (GPRS Tunneling Protocol) session basis with a sampling rate of 1:16, so that all packets in both directions from a sampled GTP session were captured. Our trace contains 278 million TCP packets (162 GB data) of 3G traffic continuously captured in 3 hours. Due to concerns of large traffic volume and user privacy issues, we only recorded TCP/IP headers and a 64-bit timestamp for each packet, but no subscriber IDs or phone numbers.

Subsequently, we extract *sessions* (they are different from GTP sessions) from the trace, with each session consisting of all packets transferred by the same UE identified through the private client IP address present in the trace. It is known that cellular public IPs change very quickly [12]. But private IPs of Carrier 1 are much less dynamic, changing only at the interval of tens of minutes. Multiple TCP flows from concurrent applications may be mixed in the same session. We use a threshold of 60 sec of idle time to decide that a session has terminated. Changing this value to other values such as 45 or 75 sec does not qualitatively affect the anal-

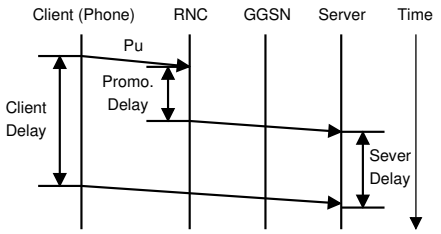


Figure 8: State promotion triggered by an UL packet

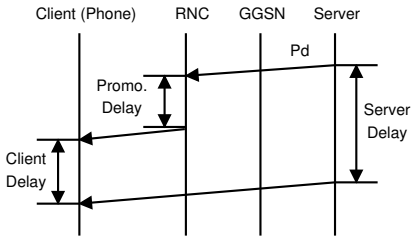


Figure 9: State promotion triggered by a DL packet

ysis results. Besides, as shown in Figure 10, we do observe qualitative difference of session rate (total bytes divided by the duration of a session) between 3G and 2G. Finally, we extracted about 1.0 million 3G sessions from the raw trace.

We discuss three limitations of our dataset. First, similar to previous measurements of wired network using passive trace [33, 26], the finite duration (3 hours) may influence distributions of state machine characteristics. Second, the dataset does not contain UDP traffic. A recent study [26] indicates that UDP accounts for less than 8% of the traffic volume for a wired VPN link. We expect the percentage of UDP traffic for cellular networks to be even smaller because currently UDP-based streaming and gaming applications on smartphones are not as popular as those on PCs. The third limitation relates to the data collection location (GGSN). We address this issue next in §4.2. Overall, we believe that the above limitations do not qualitatively affect our state machine analysis results.

4.2 Data Preprocessing

One issue regarding our cellular dataset is that the packet timestamps recorded by the GGSN are affected by delays introduced by RRC state promotions that may take up to 2 seconds to finish. Ideally we would like to capture traffic traces directly at the RNC to understand the network interaction between UEs and the radio access network so that we can use the trace to experiment with different state machine configurations. However, the data collection point, GGSN, is at the upstream from the RNC where state promotions take place. There are two scenarios for this impact. First, if an uplink packet P_u from client (UE) to the server triggers a state promotion, *e.g.,* from IDLE to DCH, which happens *before* P_u is captured by the GGSN. Thus, the GGSN records a later timestamp than the actual arrival time of P_u as shown in Figure 8. In this case, we need to shift backward in time for packet P_u and all its subsequent packets within this session by the promotion delay to preserve the correct time spacing among packets. This case is detected by simulating the state machine for each session.

The second case depicted in Figure 9 is similar except that the state promotion is triggered by a downlink packet P_d from the server to the client, so that the promotion takes place *after* the GGSN observes P_d. The packet P_d received by the client will be delayed by the state promotion, therefore the echoing uplink packet from the client and any other subsequent packets should have their timestamps shifted back by the promotion delay to exclude the effect of the promotion delay on the inter-packet time spacing.

To verify the accuracy of this preprocessing procedure, we performed controlled experiments shown in Figures 8 and 9. From the UE (client), starting from IDLE, we send two UDP packets P_1 and P_2 (two uplink packets sent in a row in Figure 8, their interval observed at the client is less than α) where P_1 triggers a state promotion, then the difference between the latency between P_1 and P_2 of the client and that of the server is approximately the state promotion delay. Similarly we verified the downlink case.

The main purpose of this data preprocessing is to eliminate the effects brought by the non-trivial promotion time while keeping other packet dynamics unchanged. It enables us to experiment with different state machine parameters or to apply a different state machine on the preprocessed trace in our further analysis. Given a session extracted in §4.1, we replay it against the original state machine of Carrier 1. If a state promotion is detected, then we shift relevant packets accordingly to remove the promotion delay. The preprocessing also detects sessions that violate the state machine. For example, we assume the state is FACH after P_k is transmitted. An uplink packet P_{k+1} triggers a FACH→DCH promotion, but the inter-arrival time between P_k and P_{k+1} is less than the FACH→DCH promotion delay. We expect that such violations are mostly caused by the variation of the RLC buffer thresholds (our simulation program assumes that they have fixed values). Sessions violating the state machine account for only 3% of the total sessions, and are not used in our subsequent data analysis.

Table 3: Cellular traces of five applications

Application	Sessions	Bytes	Description
Email-1	15K	1.4 GB	Email provider 1
Email-2	11K	536 MB	Email provider 2
Sync	4.9K	62 MB	Synchronization service
Stream	2.7K	395 MB	Pandora audio streaming [5]
Map	1.8K	60 MB	Interactive mapping

4.3 Application-specific Data Extraction

From the preprocessed trace, we select five traffic types each corresponding to a particular application as shown in Table 3. For each application, we extract sessions in which at least 95% of the packets have either the source or the destination as one fixed server IP, thus eliminating coexistence of other applications as one session may contain multiple TCP flows of concurrent applications. For example, "Sync" consists of 4.9K sessions of a popular synchronization service. All its sessions access the same server that synchronizes emails, calendar events, contacts *etc.* between PCs and a UE using push-based notification mechanism. We use the five datasets for per-application analysis in §5.2 and §6.3 to study how application traffic patterns affect the tradeoff described in §2.3.

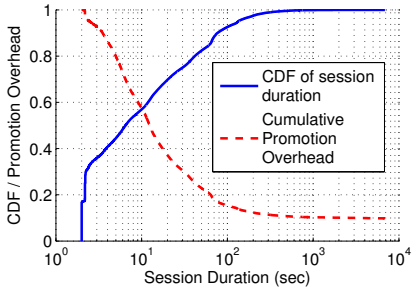


Figure 11: Cumulative promotion overhead

5. IMPACT OF RRC STATE MACHINE

In this section, we study two main negative effects of the RRC state machine, and quantify them using our datasets provided by Carrier 1. As far as we know, this is the first study that uses real cellular traces to understand how the two factors of the RRC state machine, the state promotion overhead (§5.1) and the tail effects (§5.2), impact performance, energy efficiency, and radio resource utilization. These two factors pose key tradeoffs that we attempt to balance in this paper. We investigate how multimedia streaming traffic pattern affects radio resource utilization in §5.3.

5.1 State Promotion Overhead

As discussed in §2 and §3, the RRC state promotion may incur a long latency due to control message exchanges for resource allocation at the RNC. A large number of state promotions increase management overheads at the RNC and worsen user experience [10]. They have particularly negative performance impact on short sessions. For example, starting from the IDLE state, usually it takes less than 10 sec to transfer a 200KB file under normal signal strength conditions. In such a scenario, the constant overhead of 2 sec (the IDLE→DCH promotion delay) accounts for at least 20% of the total transfer time. In other UMTS networks with 3.4kbps SRB (Signalling Radio Bearer), such a promotion time for packet session setup may take even longer, up to 4 seconds [19]. It is also known that signaling DoS attacks that maliciously trigger frequent state transitions can potentially overload the control plane and detrimentally affect the 3G infrastructure [21].

We statistically quantify the state promotion overhead using our dataset. Given Σ, the set of sessions extracted in §4.1 and preprocessed in §4.2, we compute its *average promotion overhead* $R(\Sigma)$, defined as the fraction of the total promotion delay relative to the total duration of all sessions. The duration of a session is defined as the timestamp difference between the first and the last packet in that session.

$$R(\Sigma) = \frac{\sum_{s\in\Sigma}\{2.0N_{\text{Idle-DCH}}(s) + 1.5N_{\text{FACH-DCH}}(s)\}}{\sum_{s\in\Sigma}\{T(s) + 2.0N_{\text{Idle-DCH}}(s) + 1.5N_{\text{FACH-DCH}}(s)\}}$$

Here 1.5 sec and 2.0 sec are the two promotion delays described in Table 1, $N_{\text{Idle-DCH}}(s)$ and $N_{\text{FACH-DCH}}(s)$ denote for session s the number of IDLE→DCH and FACH→DCH promotions, respectively. And $T(s)$ is the session duration after preprocessing (excluding promotion delays). Then in Figure 11, we plot the CDF of total session duration (including promotion delays) and the cumulative promotion overhead function $CP(x)$, defined as $R(\Sigma')$ where Σ' contains all sessions whose session durations (including promotion delays) are less than x. For example, we have CDF(10) = 0.57 and $CP(10) = 0.57$. They indicate that within the dataset, 57% of the sessions are at most 10 sec, and their average promotion overhead $R(\Sigma')$ is 57%. Clearly, Figure 11 indicates that shorter sessions, which contribute to the vast majority of sessions observed, suffer more severely from the promotion delay, as $CP(x)$ is a monotonically decreasing function of x.

Table 4: Duplicated DNS queries and responses due to an IDLE→DCH promotion in Windows XP

Time (s)	Direction	Details of DNS query/response
0.000	Uplink	Std. query A www.eecs.umich.edu
0.989	Uplink	Std. query A www.eecs.umich.edu
1.989	Uplink	Std. query A www.eecs.umich.edu
2.111	Downlink	Std. query response A 141.212.113.110
2.112	Downlink	Std. query response A 141.212.113.110
2.121	Downlink	Std. query response A 141.212.113.110

Our second observation is that, the RRC state promotion delay may be longer than application-specific timeout values, thus causing unnecessary timeouts leading to increased traffic volume and server overhead. For example, by default, Windows uses $t = 1, 2, 4, 8$ sec as timeout values for DNS queries [2]. Whenever a DNS query triggers an IDLE→DCH promotion that takes 2 seconds, the UE always experiences the first two timeouts. Therefore, two additional identical DNS queries are unnecessarily transmitted, and three identical responses simultaneously return to the UE, as shown in Table 4. This often happens in web browsing when a user clicks a link (triggering a DNS query) after an idle period. The problem can be addressed by using a large timeout value after a long idle period for UMTS 3G connection.

5.2 The Tail Effects

One straightforward way to alleviate the state promotion overhead is to increase inactivity timer values. For example, consider two sessions that transfer data on DCH during time=0 to time=3 sec and from $t = 10s$ to $t = 14s$. We can eliminate the state promotion at $t = 10s$ by increasing α to at least 7 sec. However, this decreases the DCH utilization efficiency as the UE occupies the dedicated channel from $t = 3s$ to $t = 10s$ without any data transmission activity. Furthermore, this worsens the negative impact of *tail effects*, which waste radio resources and UE's radio energy.

We define a *tail* as the idle time period matching the inactivity timer value before a state demotion [14]. It can never be used to transfer any data. In the above example, if $\alpha < 7sec$, then there exists a DCH tail from $t = 3$ to $t = 3 + \alpha$. Otherwise the period between $t = 3$ and $t = 10$ does not belong to a tail since there is no state demotion.

During a tail time, a UE still occupies transmission channels and WCDMA codes, and its radio power consumption is kept at the corresponding level of the state. In typical UMTS networks, each UE is allocated dedicated channels whose radio resources are completely wasted during the tail time. For HSDPA [19] described in §2, although the high speed transport channel is shared by a limited number of UEs, occupying it during the tail time can potentially prevent other UEs from using the high speed channel. More importantly, tail time wastes considerable amount of UE radio energy regardless of whether the channel is shared or dedicated. Reducing tail time incurs more frequent state transitions, a key tradeoff we explored in §2.3.

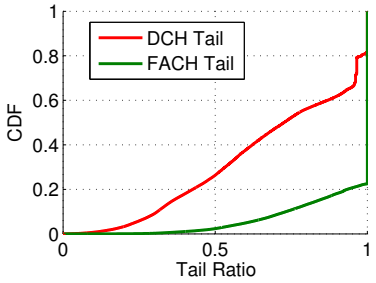


Figure 12: Distribution of $P_{\text{DCH-Tail}}$ and $P_{\text{FACH-Tail}}$ across all sessions

Table 5: Breakdown of RRC state occupation / transition time and the tail ratios

Occupation/Trans. Time	
P_{DCH}	44.5%
P_{FACH}	48.0%
$P_{\text{IDLE}\rightarrow\text{DCH}}$	6.8%
$P_{\text{FACH}\rightarrow\text{DCH}}$	0.7%
Tail Ratios	
$P_{\text{DCH-Tail}}$	45.3%
$P_{\text{FACH-Tail}}$	86.1%

Overall measurement. Table 5 provides overall statistics for all sessions. The first four rows in Table 5 break down state occupation/transition time of all sessions into four categories (they add up to 100%): on DCH/FACH states, or in IDLE→DCH/ FACH→DCH promotions. In the other two rows, $P_{\text{DCH-Tail}}$ is the DCH tail ratio, defined as the total DCH tail time as a percentage of the total DCH time. We define $P_{\text{FACH-Tail}}$, the FACH tail ratio, in a similar way. Figure 12 plots the CDFs of $P_{\text{DCH-Tail}}$ and $P_{\text{FACH-Tail}}$ on a per-session basis. The results indicate that tail time wastes considerable radio resources and hence UE's radio energy. It is surprising that the overall tail ratio for DCH and FACH are 45% and 86% respectively, and that more than 70% of the sessions have $P_{\text{DCH-Tail}}$ higher than 50%.

We also found that although the state occupation time of FACH is 48%, it transfers only 0.29% of the total traffic volume. The vast majority of the bytes (99.71%) are transferred in DCH. In other words, due to its low RLC buffer thresholds (§2), low throughput, and long tail, the efficiency of FACH is extremely low.

There exists strong correlations between session size (*i.e.,* the number of bytes of a session) and state machine characteristics. In particular, Figure 13 indicates that large sessions tend to have high fraction of DCH occupation time (P_{DCH}) as well as low P_{FACH} and $P_{\text{PROMO}} = P_{\text{IDLE}\rightarrow\text{DCH}} + P_{\text{FACH}\rightarrow\text{DCH}}$ values. Also as shown in Figure 14, as session size increases, both DCH and FACH tail ratios statistically decrease. In fact, small sessions are short. Their tail time periods, which are at least 5 sec and 12 sec for DCH and FACH, respectively, are comparable to or even longer than the total session duration, thus causing high $P_{\text{DCH-Tail}}$ and $P_{\text{FACH-Tail}}$ values. Another reason is that, we observe large sessions are more likely to perform continuous data transfers (*e.g.,* file downloading and multimedia streaming) that constantly occupy DCH, while small sessions tend to transfer data intermittently.

Per-application measurement. Figure 15 plots the per-session DCH tail distributions for the five applications shown in Table 3. The map application has low DCH tail ratios as its traffic patterns are non-consecutive data bursts interleaved with short pauses that are usually shorter than the α timer, so a UE always occupies DCH. On the other hand, the Sync application has inefficient DCH utilizations indicated by high $P_{\text{DCH-Tail}}$ values, since most of its sessions are very short. Usually a Sync session consists of a single data burst incurring a 5-sec DCH tail and a 12-sec FACH tail.

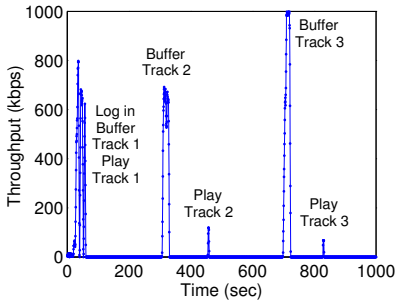


Figure 16: Pandora streaming (first 1k sec)

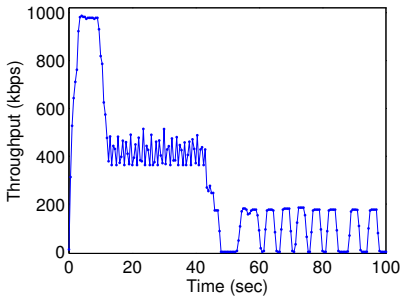


Figure 17: YouTube streaming (first 100 sec)

5.3 Streaming Traffic Pattern and Tails

To investigate streaming traffic tails, we use an Android G2 phone of Carrier 2 to collect tcpdump traces for Pandora audio streaming [5], which is the "Streaming" application in Table 3, and YouTube video streaming[4]. They are the two most popular smartphone multimedia streaming applications. Both of them use TCP.

Pandora [5] is a music recommendation and Internet radio service. A user enter her favorite song or artist (called a "radio station"), then Pandora automatically streams similar music. We collected a 30-min Pandora trace by logging onto one author's Pandora account, selecting a pre-defined station, and then listening to seven tracks (songs). The traffic pattern is shown in Figure 16. Before a track is over, the content of the next track is buffered in one burst utilizing the maximal bandwidth (indicated by "Buffer Track x" in Figure 16). Then at the exact moment of switching to the next track, a small traffic burst is generated (indicated by "Play Track x" in Figure 16). This explains why Pandora has high tail ratios as each short burst incurs a tail. Based on our measurement, in the example shown in Figure 16, 50.1% of the DCH time and 59.2% of the radio energy are wasted on tails.

[4]The two applications' streaming behaviors may be different on other platforms (*e.g.,* iPhone).

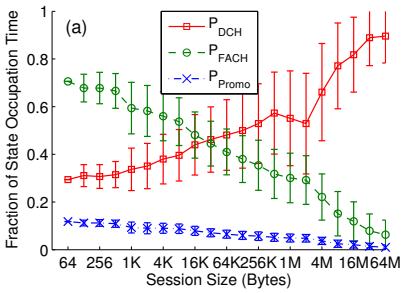


Figure 13: Session size vs. state occupation time

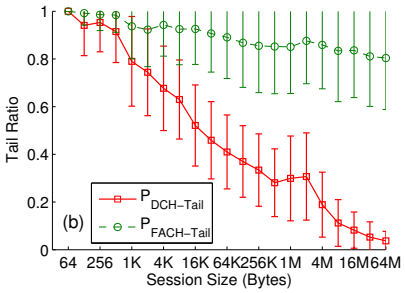


Figure 14: Session size vs.tail ratios

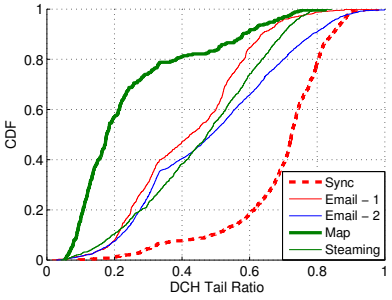


Figure 15: CDF of DCH tail ratios for different apps

YouTube employs a different streaming procedure consisting of three phases shown in Figure 17. *(i)* For the first 10 sec, the maximal bandwidth is utilized for data transmission. *(ii)* The throughput is kept at around 400kbps in the next 30 sec. *(iii)* The remaining content is transmitted intermittently with the inter-burst time between 3 to 5 sec and the throughput for each burst of 200 kbps. Shorter video clips may only experience the first one or two phases. We found that YouTube traffic incurs almost no tail (except for one tail in the end) since nearly all packet inter-arrival times (IATs) are less than the α timer value. However, its drawback is under-utilization of network bandwidth causing its long DCH occupation time. Further, sending data slowly may significantly waste UE's energy since on DCH a UE's radio power is relatively stable ($\pm$ 50mW as measured by a power meter) regardless of the bit rate when the signal strength is stable.

In summary, our analysis of Figure 16 and Figure 17 again implies that application traffic patterns can cause significant impact on radio resource and energy consumptions. The Pandora's approach incurs long tail periods while the YouTube streaming strategy suffers from long DCH occupation time and low energy efficiency due to bandwidth under-utilization. We propose a more energy-efficient approach for YouTube streaming in §7.

6. TUNING INACTIVITY TIMERS

Given the earlier observation that the inactivity timer values determine the key tradeoff (§2.3), we discuss how to optimize inactivity timers by trace-driven tuning. We describe our methodology and evaluation metrics in §6.1, followed by the results in §6.2 and for different applications in §6.3. We mainly focus on Carrier 1, with Carrier 2's findings briefly covered in §6.4.

6.1 Methodology and Evaluation Metrics

Given the moderate size of the search space for both inactivity timers, we exhaustively enumerate all combinations of the α (DCH→FACH) timer, and the β (FACH→IDLE) timer and evaluate each combination empirically by replaying all sessions for the corresponding RRC state machine. This is similar to, but simpler than, the simulation procedure described in §4.2 as the trace is state machine independent after preprocessing. As described in §2.3, we use three metrics to characterize the tradeoff.

- $\Delta E(\alpha,\beta) = (E(\alpha,\beta) - E(A,B))/E(A,B)$: the relative change in UE radio energy consumption relative to that of the default timer setting.

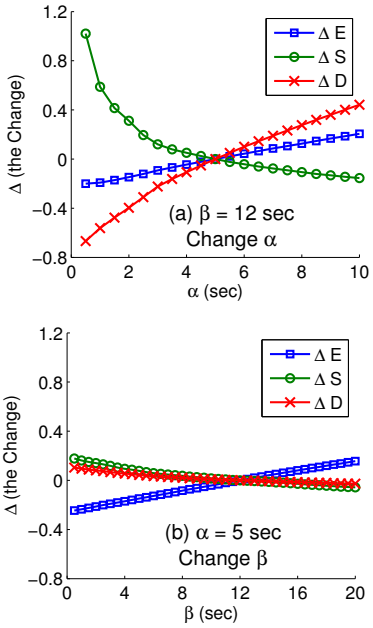


Figure 21: Impact of changing one timer (α or β). The other timer (β or α) is set to the default value

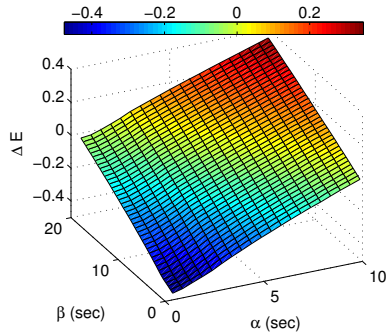
- $\Delta S(\alpha,\beta) = (S(\alpha,\beta) - S(A,B))/S(A,B)$: the relative change in the number of state promotions.
- $\Delta D(\alpha,\beta) = (D(\alpha,\beta) - D(A,B))/D(A,B)$: the relative change in the total DCH time.

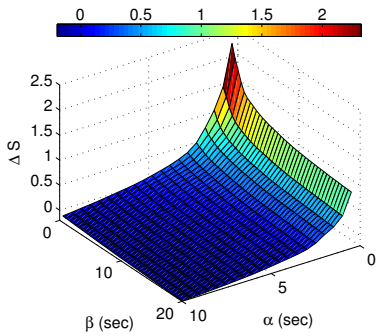
where $A = 5sec$ and $B = 12sec$ correspond to the default inactivity timer values for Carrier 1 (Table 1).

6.2 Overall Results

We visualize the distributions of ΔE, ΔS, ΔD in Figure 18, Figure 19, and Figure 20, respectively. Each plot contains $|\{0.5s, 1s, ..., 10s\} \times \{0.5s, 1s, ..., 20s\}| = 800$ samples of (α,β) pairs. We first fit them to quadratic functions for $\alpha,\beta \geq 1$ using multi-linear regression, with the residuals of less than 0.01, 0.08 (for $\alpha,\beta > 2sec$), and 0.03 for ΔE, ΔS, and ΔD, respectively. Clearly, the regression model is traffic-dependent.

$$\Delta E(\alpha,\beta) = -0.544 + 0.061\alpha + 0.026\beta - 0.001\alpha^2$$
$$\Delta S(\alpha,\beta) = 0.652 - 0.123\alpha - 0.023\beta + 0.005\alpha^2 + 0.001\alpha\beta$$
$$\Delta D(\alpha,\beta) = -0.618 + 0.171\alpha - 0.011\beta - 0.006\alpha^2$$





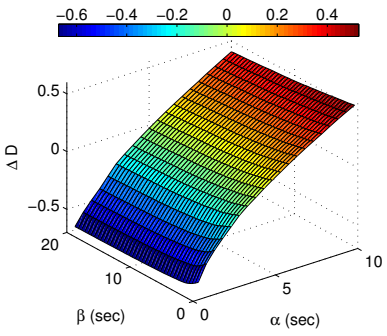


Figure 18: Impact of (α, β) on ΔE **Figure 19: Impact of (α, β) on ΔS** **Figure 20: Impact of (α, β) on ΔD**

We observe that ΔE is close to a linear function of α and β. Clearly, decreasing inactivity timer values reduces tail time, thus cutting a UE's energy consumption. Usually a UE is at DCH after transferring some data. Assuming this, the UE will *first* experience the α and then the β tail. Therefore α contributes more than β does in determining ΔE. For ΔS and ΔD, they can also be approximately regarded as linear functions of α and β when $\alpha, \beta > 2sec$, and the contribution of α is much larger than that of β as well. This is visualized in the 2D plots of Figure 21(a) and (b), where we fix one timer to the default value and change the other timer, then study its impact on the three metrics.

The linear fitting for ΔS does not hold when α is small ($< 2sec$) due to skewed distribution of packet inter-arrival times. Aggressively decreasing α causes excessive number of state promotions, which increase processing overheads at the RNC, worsen the application performance, and cause additional energy overhead since a promotion may consume as much as 87.5% of the power of the DCH state. We observe in Figure 21(a) that $|d\Delta S/d\alpha| > |d\Delta D/d\alpha|$ and $|d\Delta S/d\alpha| > |d\Delta E/d\alpha|$ for $\alpha < 5sec$, meaning that as we reduce α, the incurred state promotion overhead grows faster than the saved DCH time and energy do. This implies the fundamental limitation of current timeout scheme: it is difficult to well balance the tradeoff among ΔD, ΔE, and ΔS as timers are globally and statically set to constant values.

6.3 Per-application Results

Applications exhibit traffic patterns with application-specific packet dynamics, thus with different responses to changes in inactivity timer values. We next study the impact of the α timer on ΔE, ΔS, and ΔD for the five applications described in §4.3. We vary α and fix β at the default value of 12 sec given the relatively small impact of β.

Figures 22, 23, and 24 illustrate the effect of α on ΔE, ΔS, and ΔD, respectively, for the four applications (the curves for "Email-2" are very similar to those for "Email-1"). For the interactive map application, when α is small, decreasing α causes considerable increase of ΔS up to 470% (Figure 23). Accordingly, its ΔE increases as α decreases due to additional energy consumed by state promotions (Figure 22). As described in §5.2, the map traffic pattern consists of non-consecutive data bursts interleaved with short user-injected pauses. Therefore a small value of α comparable to or shorter than most pause durations will trigger a tremendous increase of FACH→DCH promotions. Figures 22 and 24 also indicate that short sessions with a single data burst (the Sync application) benefit more from small α values in terms of energy and radio resource savings. Our observations suggest that dynamically setting timers according to application traffic patterns may better balance the tradeoff.

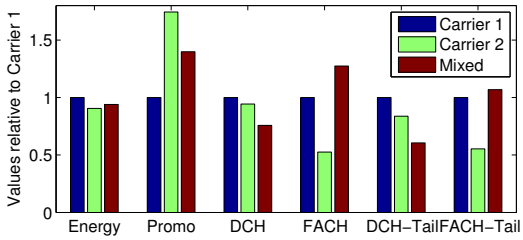


Figure 25: Compare state machines for two carriers

6.4 Comparing to Carrier 2's State Machine

Recall that the major difference between the state machine for Carrier 1 (Figure 2) and that for Carrier 2 (Figure 3) is the state promotion from IDLE, from which upon any user triggered network activity, Carrier 1 directly enters into DCH while Carrier 2 may do either IDLE→FACH or IDLE→FACH→DCH, depending on the packet size. By employing the same simulation approach described in §6.1 for Carrier 2 we observe qualitatively similar trends in terms of the way ΔE, ΔS, and ΔD response to changes of α and β. Also α plays a more important role than β does in controlling the three metrics.

Figure 25 quantitatively compares both carriers. "Carrier 1" and "Carrier 2" are the real state machine settings adopted by both carriers as described in Table 1. "Mixed" corresponds to an artificial scheme using Carrier 2's state transition model but Carrier 1's parameters. We characterize each scheme using six metrics with Carrier 1 serving as the comparison baseline (its Y values are always 1).

The results in Figure 25 verify again the tradeoff discussed in §2.3. First, by comparing "Carrier 1" and "Mixed", we find that changing IDLE→DCH to IDLE→FACH→DCH decreases DCH and DCH tail time by 24% and 39%, respectively, but at the cost of increased number of state promotions by 40%, since for Carrier 2, when a UE at IDLE has non-trivial amount of data (greater than the RLC buffer threshold) to transfer, it always experiences two state promotions to DCH. Our second observation is derived by comparing "Carrier 2" and "Mixed". Their FACH→IDLE timers are significantly different (4 sec for "Carrier 2" and 12 sec for "Mix"), resulting in considerable disparities of their FACH (and FACH tail) times.

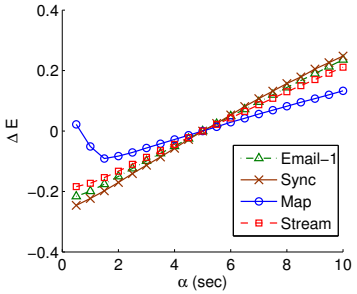


Figure 22: Impact of α on ΔE (β is set to the default) for four apps.

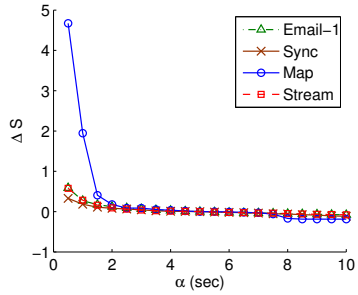


Figure 23: Impact of α on ΔS (β is set to the default) for four apps.

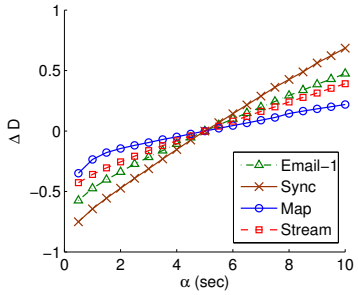


Figure 24: Impact of α on ΔD (β is set to the default) for four apps.

6.5 Summary

We tune the two inactivity timer values for the RRC state machine of Carrier 1. Table 6 summarizes our derived timer values under different constraints of ΔS. Column 2 to 4 correspond to three optimization objectives: energy-saving biased (minimize $0.75\Delta E + 0.25\Delta D$), no bias (minimize $0.5\Delta E + 0.5\Delta D$), and radio-resource-saving biased (minimize $0.25\Delta E + 0.75\Delta D$). The coefficients are empirically chosen to weight the energy and/or the radio resource consumption.

We highlight our findings in this section as follows. *(i)* ΔE, ΔS, and ΔD are approximately linear functions of α and β when they are not very small. The α timer imposes much higher impact on the three metrics than the β timer does. *(ii)* Very small α timer values ($< 2sec$) cause significant increase of the state promotion overhead. *(iii)* Applications have different sensitivities to changes of inactivity timers due to their different traffic patterns. *(iv)* It is difficult to well balance the tradeoff among ΔD, ΔE, and ΔS since the state promotion overhead grows faster than saved DCH time and energy do when we reduce the timers. The fundamental reason is that timers are globally and statically set to constant values.

7. IMPROVE CURRENT TIMER SCHEME

We explore approaches that improve the current inactivity timer scheme whose limitations are revealed in §6.

7.1 Shaping Traffic Patterns

UE applications alter traffic patterns based on the state machine behavior in order to reduce the tail time. We describe two such approaches.

Batching and Prefetching. In [14], the authors discuss two traffic shaping techniques: batching and prefetching. For delay-tolerant applications such as Email and RSS feeds, their transfers can be *batched* to reduce the tail time. In the scenario of web searching, a UE can avoid tails caused by a user's idle time with high probability by *prefetching* the top search results. [14] proposes an algorithm called *TailEnder* that schedules transfers to minimize the energy consumption while meeting user-specified deadlines by batching or prefetching. Their simulation indicates that TailEnder can transfer 60% more RSS feed updates and download search results for more than 50% of web queries, compared to using the default scheme. Similar schemes for delay-tolerant applications in cellular environment were proposed for, for example, offloading 3G data transfers to WiFi [13] and scheduling communication during periods of strong signal strength [29].

Proposed Traffic shaping scheme for YouTube. Recall that in §5.3, we pinpoint the energy inefficiency of YouTube traffic caused by under-utilizing the available bandwidth (Figure 17). To overcome such inefficiency, we reshape the traffic using a hypothetical streaming scheme called *chunk mode*, as illustrated in Figure 26(a). The video content is split into n chunks $C_1, ..., C_n$, each transmitted at the highest bit rate. We model the traffic pattern of chunk mode transfer as follows. Let $L = 9.7MB$ be the size of a 10-minute video and let $M = 800kbps$ be the maximal throughput. For each chunk, it takes T_{SS} seconds for the TCP slow start[5] to ramp up to the throughput of M. By considering delayed ACKs and letting the RTT be 250ms (as measured by the ping latency to YouTube), we compute T_{SS} at 1.3 sec during which $L_{SS} = 60KB$ of data is transferred. The total transfer time for the n chunks (excluding tails) is $T = (T_{SS} + (\frac{L}{n} - L_{SS})/M)n$, and the DCH tail and FACH tail time are $n\alpha$ and $n\beta$, respectively. The whole transfer incurs n IDLE→FACH promotions and n FACH→DCH promotions for Carrier 2's UMTS network.

Based on the above parameters, we compute ΔD and ΔE and plot them in Figure 26(b) and (c), respectively. Note that the energy E consists of three components: the state promotion energy, the DCH non-tail energy, and the tail energy of DCH/FACH. In each plot, the two curves "YouTube" and "Chunk Mode" correspond to streaming schemes of current YouTube and the chunk mode, respectively. We describe the "Chunk + FD" curve in §7.2.

As indicated by Figure 26, compared to current YouTube streaming strategy, the chunk mode saves DCH time and energy by up to 80%. Transferring the whole video in one chunk ($n = 1$) is the most effective. However, measurement results show that users often do not watch the entire video [18]. Therefore when n is small, it may cause unnecessary data transfers if a user only watches part of the video. This problem can be addressed by increasing n and transferring data chunks according to the playing progress of the video. However, resources saved by the chunk mode decrease as n increases due to the tail effect.

To summarize, for some applications, shaping their traffic patterns based on the prior knowledge of the RRC state machine brings significant savings of radio resources and UE

[5]A slow start is necessary since the interval between consecutive chunks is longer than one TCP retransmission timeout [11]. Using TCP keep alive can avoid slow starts but it consumes more energy due to the tail effect.

Table 6: Optimal timer values under constraints of ΔS and different objective functions

Constraint	Obj: $\min\{0.75\Delta E + 0.25\Delta D\}$ $(\alpha, \beta, \Delta E, \Delta D)$	Obj: $\min\{0.50\Delta E + 0.50\Delta D\}$ $(\alpha, \beta, \Delta E, \Delta D)$	Obj: $\min\{0.25\Delta E + 0.75\Delta D\}$ $(\alpha, \beta, \Delta E, \Delta D)$
$\Delta S < -0.1$	$(8.0, 11.5, +0.12, +0.28)$	$(6.5, 18.0, +0.18, +0.12)$	$(6.5, 18.0, +0.18, +0.12)$
$\Delta S < 0.0$	$(6.0, 7.0, -0.06, +0.13)$	$(4.0, 19.0, +0.10, -0.13)$	$(4.0, 19.0, +0.10, -0.13)$
$\Delta S < 0.1$	$(4.0, 8.0, -0.13, -0.09)$	$(3.0, 14.5, -0.04, -0.23)$	$(3.0, 14.5, -0.04, -0.23)$
$\Delta S < 0.2$	$(3.5, 4.5, -0.23, -0.11)$	$(2.5, 11.5, -0.13, -0.31)$	$(2.5, 11.5, -0.13, -0.31)$
$\Delta S < 0.3$	$(3.0, 3.0, -0.30, -0.16)$	$(2.5, 6.0, -0.26, -0.28)$	$(2.0, 13.5, -0.11, -0.40)$
$\Delta S < 0.4$	$(2.5, 3.0, -0.33, -0.25)$	$(2.0, 7.5, -0.25, -0.38)$	$(1.5, 13.5, -0.14, -0.48)$
$\Delta S < 0.5$	$(2.0, 4.0, -0.34, -0.35)$	$(1.5, 8.0, -0.26, -0.46)$	$(1.5, 8.0, -0.26, -0.46)$

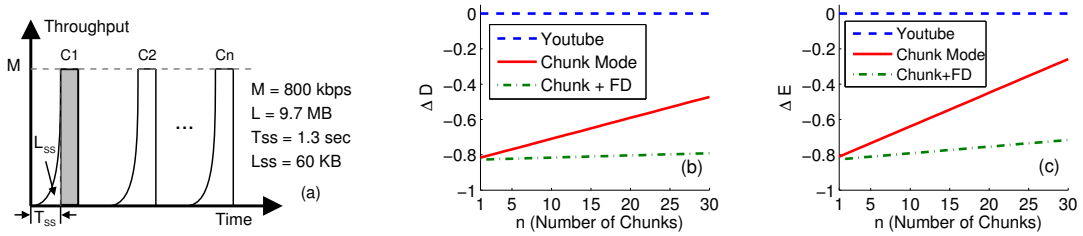


Figure 26: (a) Streaming in chunk mode and its evaluations of (b) ΔD and (c) ΔE

energy. However, this technique has limited applicability. For example, it is difficult to be applied to delay-sensitive applications such as web browsing.

7.2 Dynamic Timers and Fast Dormancy

Dynamic Timer Scheme. Our per-application study in §5.2 and §6.3 suggests that *dynamically* changing timers can potentially better balance the tradeoff. Ideally, this can be achieved by the RNC that adjusts timers at a per-UE basis according to observed traffic patterns. However, such an approach requires significant changes to the RNC, which currently does not recognize IP and its upper layers. The computational overhead is also a concern as the RNC has to identify traffic patterns and compute the appropriate timer values for all connected UEs. The third challenge is that for each UE, the traffic observed by the RNC may originate from multiple applications that concurrently access the network, thus making identifying traffic patterns even harder.

Fast Dormancy. Let us revisit the fundamental reason why inactivity timers are necessary. Since the RNC has no easy way of predicting a UE's network idle time, it conservatively appends a tail to every network usage period. This naturally gives rise to the idea of letting UE applications determine the end of a network usage period (*i.e.,* predicting a tail) by leveraging detailed application-specific knowledge. Applications try to *predict* tails. Once an imminent tail is predicted, a UE notifies the RNC, which then immediately releases the allocated resources. Based on this simple intuition, a feature called *fast dormancy* has been proposed to be included in 3GPP Release 7 [8] and Release 8 [9]. A UE sends a special RRC message to the RNC through the control channel. Upon receiving the message, the RNC releases the RRC connection and lets the UE go to IDLE (or a hibernating PCH state with a lower promotion delay). This feature is currently supported by several handsets [9].

We use the YouTube example to demonstrate a typical scenario where fast dormancy can be applied. Recall the chunk mode streaming scheme shown in Figure 26. In order to eliminate tails, the YouTube application would invoke fast dormancy to demote the state to IDLE immediately after a chunk is received (assuming no concurrent network activity exists). This corresponds to the "Chunk + FD" curve in Figure 26(b) and (c), which indicate that by eliminating the tails, fast dormancy can keep ΔD and ΔE almost constant regardless of n, the number of chunks. Recall that a large n prevents unnecessary data transfers in common cases [18] where a user watches part of the video.

Based on measuring the device power consumption, we do observe that Google Nexus One [3], one of the newly released phones in 2010, adopts fast dormancy in an *application-agnostic* manner: the UE goes to IDLE faster than normal phones do for the same carrier. Or equivalently, the Nexus One uses shorter inactivity timers controlled by the device in order to improve the battery life. The incurred drawbacks, again, are the extra state promotions that cause additional RNC processing overhead and worsen user experience [10, 21].

To the best of our knowledge, however, no individual smartphone application today can invoke fast dormancy based on its traffic pattern, partly due to two reasons. First, for user-interactive applications (*e.g.,* web browsing), accurately predicting a long idle period is non-trivial as user behavior injects randomness to the packet timing. Second, there lacks OS support that provides a simple programming interface for invoking fast dormancy. In particular, the concurrency problem presents a challenge. It is not feasible that applications independently predict the tail and invoke fast dormancy since state transitions are determined by the aggregated traffic of all applications. The OS should schedule concurrent applications and invoke fast dormancy only if the combined idle period predicted by all applications is long enough. Bridging the gap between the application and the fast dormancy support is our on-going work [27].

8. RELATED WORK

To the best of our knowledge, our work is the first comprehensive study that characterizes the RRC state machine and optimizes radio resource allocation using realistic traffic patterns. We have discussed related work of improving current timeout scheme (TailEnder and fast dormancy) in §7.

The problem of choosing the optimal inactivity timer val-

ues has also been studied in several previous work. Among them, Chuah *et al.* [16] studied the impact of inactivity timers on UMTS network capacity by simulating the performance of web browsing. Some [20, 32] proposed analytical models to measure the energy consumption of user devices under different timer values. In addition, some [22, 31] also discussed the influence of different timeout values on both service quality and energy consumption. Several other projects also studied network resource management [17, 30]. However, all this prior work was evaluated based on simulation using particular traffic models. In fact, real traffic patterns depend highly on user behavior among other factors and are not easily captured using analytic models. We therefore use real traffic traces for evaluation to ensure the applicability of our work to real network settings.

Researchers also proposed ideas on setting timeout values dynamically. In [23] timeouts for the inactivity timers are decided dynamically for radio resources and computation resources. However, they only addressed the problem from the perspective of network capacity as to reducing the call blocking and dropping rate. In their scenario, the same timer values are applied *globally* to all UEs at any given time. The dynamic timer scheme described in §7.2, however, is to save radio resources and UE's energy. Therefore the timer is customized for each UE.

Inferring the state machine used in radio resource control is another interesting topic. Previous work [15, 24] introduced 3G Transition Triggering Tool to infer RRC state machine parameters and to measure one-way delays in different RRC states. But their approach is based on a fixed state transition model and only infers the corresponding parameters (*e.g.,* inactivity timers). Beyond their work, we also considered and inferred different state transition models configured by two commercial UMTS carriers.

9. CONCLUDING REMARKS

In this paper, we undertook a detailed exploration of the RRC state machine, which guides the radio resource allocation policy in 3G UMTS network, by analyzing real cellular traces and measuring from real smartphones. We found that the RRC state machine may cause considerable performance inefficiency due to the state promotion overhead, as well as cause significant radio resource and UE energy inefficiency due to the tail effects. These two factors form the key tradeoff that is difficult to balance by the current inactivity timer designs. The fundamental reason is that the timers are globally and statically set to constant values that cannot adapt to the diversity of traffic patterns generated by different applications. We believe that addressing this problem requires the knowledge of UE applications, which can proactively alter traffic patterns based on the state machine behavior, or cooperate with the radio access network in allocating radio resources (*i.e.,* the fast dormancy approach). We will explore both approaches in our future work.

10. REFERENCES

[1] 300 Million UMTS Subscribers. `http://www.3gpp.org/300-million-UMTS-subscribers`.

[2] DNSQueryTimeouts. `http://technet.microsoft.com/en-us/library/cc977482.aspx`.

[3] Google Nexus One Phone. `http://www.google.com/phone/static/en_US-nexusone_tech_specs.html`.

[4] Monsoon power monitor. `http://www.msoon.com/`.

[5] Pandora Radio. `http://www.pandora.com`.

[6] GERAN RRC state-machine. 3GPP GAHW-000027, 2000.

[7] System Parameter Recommendations to Optimize PS Data User Experience and UE Battery Life. Engineering Services Group, Qualcomm, 2007.

[8] UE "Fast Dormancy" behavior. 3GPP discussion and decision notes R2-075251, 2007.

[9] Configuration of fast dormancy in release 8. 3GPP discussion and decision notes RP-090960, 2009.

[10] System impact of poor proprietary fast dormancy. 3GPP discussion and decision notes RP-090941, 2009.

[11] M. Allman, V. Paxson, and W. R. Stevens. TCP Congestion Control. RFC 2581, 1999.

[12] M. Balakrishnan, I. Mohomed, and V. Ramasubramanian. Where's That Phone?: Geolocating IP Addresses on 3G Networks. In *IMC*, 2009.

[13] A. Balasubramanian, R. Mahajan, and A. Venkataramani. Augmenting Mobile 3G Using WiFi. In *Mobisys*, 2010.

[14] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani. Energy Consumption in Mobile Phones: A Measurement Study and Implications for Network Applications. In *IMC*, 2009.

[15] A. Barbuzzi, F. Ricciato, and G. Boggia. Discovering Parameter Setting in 3G Networks via Active Measurements. *Communications Letters, IEEE*, 12(10), 2008.

[16] M. Chuah, W. Luo, and X. Zhang. Impacts of Inactivity Timer Values on UMTS System Capacity. In *Wireless Communications and Networking Conference*, 2002.

[17] M. Ghaderi, A. Sridharan, H. Zang, D. Towsley, and R. Cruz. Tcp-aware resource allocation in cdma networks. In *Proceedings of ACM MOBICOM*, Los Angeles, CA, USA, September 2006.

[18] L. Guo, E. Tan, S. Chen, Z. Xiao, O. Spatscheck, and X. Zhang. Delving into Internet Streaming Media Delivery: A Quality and Resource Utilization Perspective. In *IMC*, 2006.

[19] H. Holma and A. Toskala. HSDPA/HSUPA for UMTS: High Speed Radio Access for Mobile Communications. John Wiley and Sons, Inc., 2006.

[20] C.-C. Lee, J.-H. Yeh, and J.-C. Chen. Impact of inactivity timer on energy consumption in WCDMA and cdma2000. In *Wireless Telecommunications Symposium*, 2004.

[21] P. P. C. Lee, T. Bu, and T. Woo. On the Detection of Signaling DoS Attacks on 3G Wireless Networks. In *INFOCOM*, 2007.

[22] F. Liers, C. Burkhardt, and A. Mitschele-Thiel. Static RRC Timeouts for Various Traffic Scenarios. In *PIMRC*, 2007.

[23] F. Liers and A. Mitschele-Thiel. UMTS data capacity improvements employing dynamic RRC timeouts. In *PIMRC*, 2005.

[24] P. Perala, A. Barbuzzi, G. Boggia, and K. Pentikousis. Theory and Practice of RRC State Transitions in UMTS Networks. In *Proc. of IEEE Broadband Wireless Access Workshop*, 2009.

[25] J. Perez-Romero, O. Sallent, R. Agusti, and M. Diaz-Guerra. Radio resource management strategies in UMTS. John Wiley and Sons, Inc., 2005.

[26] F. Qian, A. Gerber, Z. M. Mao, S. Sen, O. Spatscheck, and W. Willinger. TCP Revisited: A Fresh Look at TCP in the Wild. In *IMC*, 2009.

[27] F. Qian, Z. Wang, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck. TOP: Tail Optimization Protocol for Cellular Radio Resource Allocation. In *ICNP*, 2010.

[28] P. Rodriguez and V. Fridman. Performance of PEPs in CellularWireless Networks. In *WCW*, 2003.

[29] A. Schulman, V. Navda, R. Ramjee, N. Spring, P. Deshpande, C. Grunewald, K. Jain, and V. Padmanabhan. Bartendr: A Practical Approach to Energy-aware Cellular Data Scheduling. In *Mobicom*, 2010.

[30] A. Sridharan, R. Subbaraman, and R. Guerin. Distributed Uplink Scheduling in CDMA Networks. In *Proceedings of IFIP-Networking 2007*, May 2007.

[31] A. Talukdar and M. Cudak. Radio resource control protocol configuration for optimum Web browsing. In *IEEE VTC*, 2002.

[32] J.-H. Yeh, J.-C. Chen, and C.-C. Lee. Comparative Analysis of Energy-Saving Techniques in 3GPP and 3GPP2 Systems. *IEEE transactions on vehicular technology*, 58(1), 2009.

[33] Y. Zhang, L. Breslau, V. Paxson, and S. Shenker. On the Characteristics and Origins of Internet Flow Rates. In *SIGCOMM*, 2002.