Preempting State Promotions to Improve Application Performance in Mobile Broadband Networks

Kristian R. Evensen, Džiugas Baltrūnas, Simone Ferlin-Oliveira, Amund Kvalbein
Simula Research Laboratory, Norway
{kristrev, dziugas, ferlin, amundk}@simula.no

ABSTRACT

Mobile broadband is one of the most common ways of connecting to the Internet. A mobile broadband network is stateful, and a device is allocated different radio resources depending on the state. State promotions take up to three seconds, and the promotions, or just being in the "wrong" state, can have a severe effect on user experience.

Existing work has mostly focused on optimising state changes in order to reduce resource usage in the network, as well as battery consumption on devices. In this paper, we look at how explicitly requesting state promotions can be used to improve application performance. Our technique is evaluated using an application that retrieves data through the common HTTP protocol, in real-world 3G networks. We show that by preempting state changes, the delay caused by state promotions are removed and the transfer time is significantly reduced.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General - Data communications

Keywords

Latency, Mobile broadband, RRC state, State preemption, Optimization, HTTP

1. INTRODUCTION

Connecting to the Internet through Mobile broadband (MBB) is increasing rapidly in popularity. The network vendor Ericsson reports that there are currently 1.1 billion MBB subscriptions world-wide [3], and they expect that 85 % of the world's population will have access to 3G networks by 2017. Also, by the same year, Ericsson expects the number of MBB subscriptions to reach 5 billion.

Unlike traditional IP networks, MBB networks are stateful. Each user equipment (UE) that is connected to a MBB network is placed in one of multiple Radio Resource Control (RRC) states, each allocated a different amount of radio resources. More radio resources means more available bandwidth and lower latency. The network controls state transitions, typically based on inactivity timers (for state demotion) and a data rate threshold (state promotion).

RRC State promotions introduce a delay of up to three seconds [14]. This is one order of magnitude higher than normal observed latencies, and can have a significant impact on the user experience. While a state promotion is ongoing, no IP traffic from the UE is forwarded through the network. As a consequence, an operation like loading a web page takes considerably longer than if the UE already was in the "correct" state.

Latency is important in order to ensure a good user experience [7,10], and even small delays cause a reduction in user satisfaction. For example, when Google increased the number of search results per page to 30, this caused an increase in page load time of 0.6 seconds (from 0.3 to 0.9) [9]. As a result, people conducted 25 % fewer searches. An experiment performed by Microsoft's consumer search engine team [18], showed that a 2 second slowdown per search query changed queries per user by -1.8 % and revenue per user by -4.3 %.

Existing work [11, 13, 14] has mostly focused on inferring the RRC state machine or optimising the "tail effect", the period of inactivity before the first state demotion timer expires. In this paper, we evaluate the potential gain of preempting the state change. A State Promotion Daemon (SPD) is running on the UE, and applications can request RRC state leases. In other words, the SPD gives applications more control over the current RRC state, instead of depending on the network.

When a request is received, the daemon sends enough traffic to exceed the data rate threshold, attempting a state promotion, unless the connection is already in the "correct" state. Leases can be renewed, and at least one lease has to be active for the daemon to attempt staying in an RRC state. By using the SPD, state promotion logic can be moved out of the applications. A developer only has to decide if and when an application should request and maintain a RRC state lease.

One potential use of the daemon is a browser requesting a RRC state lease when the user starts typing an address, speeding up the perceived loading time. Another example is to improve the performance of applications generating thin streams [12], as these applications will in many cases not generate enough data to exceed the rate threshold. An example of such an application is an SSH client, which could

MobiArch'13, October 4, 2013, Miami, Florida, USA

http://dx.doi.org/10.1145/2505906.2505911.

maintain a state lease for as long as a connection is active, improving the user experience due to lower latencies.

We have focused on state transitions in 3G-networks, i.e., UMTS. However, our approach will work with other, similar technologies (like LTE) as well. In order to demonstrate the effect of preempting the state transition, we have evaluated the impact it has on the time it takes to complete an HTTP GET-request, in real-world 3G-networks. According to Google [16], the average web page is 320 KB. By sending 0.3 % more data in order to preempt the state promotion, the effect of the promotion was mitigated. 95 % of the requests were completed at least two seconds faster.

The rest of the paper is structured as follows. In section 2, we will give a more detailed description of the UMTS-specific RRC state machine, as well as an introduction to related work. Section 3 explains the architecture of the State Promotion Daemon, while section 4 contains the results of our HTTP-experiments. Section 5 summarizes our work and provides some ideas for future work.

2. BACKGROUND

This section describes the UMTS-specific state machine, as well as introducing related work.

2.1 The RRC state machine

UMTS networks consist of three subsystems - the UE, the UMTS Terrestrial Radio Access Network (UTRAN) and the Core Network. The purpose of the UTRAN is to enable a UE to connect to the backbone, and thereby for example the Internet. A UTRAN consists of base stations and Radio Network Controllers (RNC). Each RNC maintains one RRC state for every UE that is controlled by it, and the RNC is responsible for initiating state transitions. The UE is aware of which RRC state it is in, this information is transmitted from the RNC and always kept in sync.

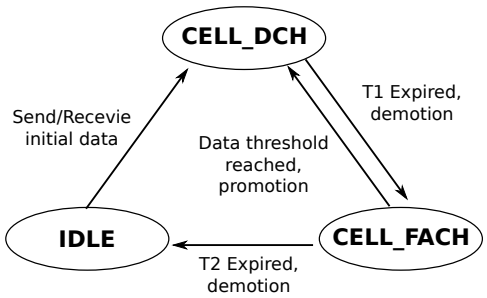

Figure 1: Example of a RRC state machine configuration, where UEs in IDLE are always promoted to CELL_DCH.

There are typically three RRC states [5], IDLE, CELL_FACH and CELL_DCH, and one common configuration of the RRC state machine is shown in figure 1. RRC state changes are controlled by two timers and a data rate threshold. If the UE is quiet, i.e., not sending any data packets, it is usually in the IDLE (sometimes referred to as DISCONNECTED) state. When there is an IP packet waiting to be sent or received by the UE, it is moved to CELL_FACH or CELL_DCH, depending on the network configuration. When in CELL_FACH, the UE can then be further promoted depending on if the data rate exceeds a given threshold. A typical value for this threshold is 1 KB/s, however, when measuring we have seen as little as 1 Kb/s in some networks. The state promotion time is roughly two seconds from IDLE to CELL_FACH or CELL_DCH, and one second from CELL_FACH to CELL_DCH.

CELL_FACH is intended for low-bandwidth traffic, and consists of a single channel shared between a limited number (typically 32) of UEs [5]. When in CELL_FACH, the UE can receive data at up to 64 Kb/s, and send data at 8-16 Kb/s. CELL_DCH provides the UE with a dedicated high-speed channel, supporting a theoretical maximum of 42 Mb/s downlink and 500-600 Kb/s uplink [13]. The latency difference between the two states is in the range of several hundred milliseconds.

Demotions are controlled by timers, commonly referred to as T1 and T2. If a UE is in CELL_DCH and has been quiet for T1 seconds, it is demoted to CELL_FACH. T2 is the timeout from CELL_FACH to IDLE. A demotion is more or less instant and has little effect on latency. As with the data rate threshold, the T1 and T2 timers are configurable. For example, in one of the Norwegian MBB networks, T1 is 10 seconds and T2 is 20 seconds.

Though our work is focused on UMTS, it can easily be adapted to work in LTE networks. LTE is the next generation of MBB and also uses an RRC state machine, but there are only two main states, IDLE and CONNECTED [1]. However, the latter has three microstates: Continuous Reception (CR), Short DRX and Long DRX. CR offers the most capacity, then Short DRX and finally Long DRX. State promotions are decided by the current data rate, while the demotions are controlled by three inactivity timers, similar to UMTS.

2.2 Related work

In MBB networks, application delays and, hence, also the user experience is directly related to radio resource management. Several previous contributions have focused on inferring the operation of the RRC state machine, and on tuning parameters in order to give more power efficient operation (for the UE) [11, 13, 14].

In [11], the authors develop a tool called 3G3T to deduce the RRC state machine transition parameters, e.g., the thresholds and inactivity timers used to promote or demote the UE from one state to another. Another method for deducing RRC state machine transitions is introduced in [13]. Their method is based on sending probing packets and observing changes in the RTT. The inferred values are combined with packet traces from a UMTS network to propose better parameter values for the demotion timers. In this work, we rely on their proposed method for inferring RRC states if these are not available directly from the modem. In [14], the authors combine RRC state inference with trace captures from several popular mobile applications, in order to profile application performance and optimize energy usage. In contrast to this work, their focus is on energy conservation. In many scenarios when applications run on less power-constrained devices, performance will be more important.

We are not the first to observe that state promotion constitutes a significant portion of the transfer delay in MBB networks. In [4], the authors focus on the time it takes from the user makes a request to first byte is received. In the cellular network they measured, state transition time constituted up to 68 % of the time to the first byte.

3. STATE PROMOTION DAEMON

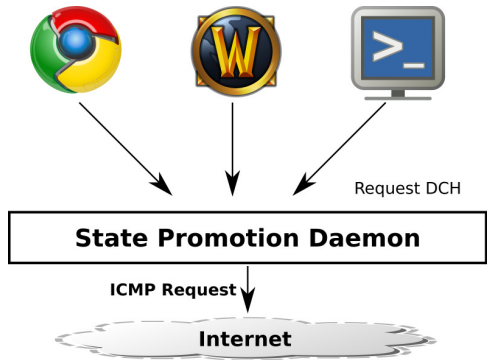

Figure 2: Overview of the architecture of the SPD.

The SPD, shown in figure 2, enables applications to have more control over the RRC state. It is designed not to rely on OS-specific features, thus, it is platform-independent. Communication between applications and the daemon relies on standard network sockets, and the daemon allows applications to explicitly request state promotions. This is done in order to avoid the UE occupying unnecessary radio resources and the daemon sending redundant traffic. Because CELL_DCH is the preferred state for data transfers, the daemon is currently focused on promoting connections to CELL_DCH.

In the next two subsections, we describe the SPD's two modes of operation. Then, we discuss some potential challenges and implementation choices.

3.1 Startup

At startup, the SPD performs two actions. The first is to determine if the RRC can be read from the modem or not, which is done by comparing the modem model against a database of known devices. Information about the RRC state is normally not available by default, except, to the best of our knowledge, on modems made by Sierra Wireless. Here, the state is available through a normal AT-command. On modems containing chipsets from Qualcomm and Huawei, the RRC state can be read by accessing the modem's diagnostic mode. Libraries like libqcdm [19] simplify this process. Qualcomm-chipsets are used in for example most of Huawei's modems released up to now, ZTE-modems and many Android-phones. If the RRC state is not exported by the modem, it is inferred based on RTT using a method similar to the one presented in [13]. The RTT difference between CELL_FACH and CELL_DCH is typically several hundred milliseconds.

The second action is to determine the T1 and T2 timers, as well as the amount of data that has to be sent in order to trigger a state promotion. These values are assumed to be static [17] and are inferred using the state promotion and demotion algorithms described in [13]. The timers are used by the SPD to avoid sending unnecessary data, by spacing promotion requests out in time.

3.2 Operation

After the startup phase, the SPD creates a UDP socket used to communicate with applications. For an application to request an RRC state lease, and instruct the daemon to potentially request a state promotion, it sends a START_LEASE-message to the SPD. If the daemon detects that the 3G connection is already in CELL_DCH when a request arrives, it looks at the T1 timer and current RRC state. If a) less than two times the CELL_DCH RTT (measured during startup) is left of T1, or b) the 3G connection is not in CELL_DCH, the daemon will attempt to trigger a state promotion. Otherwise, the promotion request is delayed until a) is true. The RRC leases are only valid for a configurable time, but can be renewed. As long as there is at least one active lease, the SPD attempts to stay in CELL_DCH. Applications can explicitly release leases by sending a STOP_LEASE-message.

The promotion request is an ICMP request with a given payload length (determined during the startup phase), in order to exceed the data threshold limit required for a RRC state promotion. Many well-known servers, for example the Open DNS public DNS servers, reply to ICMP requests with a larger payload length than the default value, and can be used as targets. By sending ICMP requests, we support the three ways a RNC measures the data rate: uplink, downlink or the sum of uplink and downlink. The success of the state promotion is detected by reading it from the modem, or by comparing the RTT of a default size ICMP request against the CELL_FACH RTT measured during startup. If it is significantly lower, the daemon assumes success. If a promotion fails, another attempt is made after T1 / 2 seconds.

3.3 Discussion

The SPD is designed to be platform independent and as non-intrusive as possible. In order to accomplish criteria one, we decided to use UDP sockets for communication between applications and the daemon. UDP sockets are supported by all modern platforms. Another possibility would have been to use a socket option. Applications could mark sockets as low latency, and as long as such a socket exists, the SPD would attempt to keep the 3G connection in CELL_DCH. However, adding socket options requires modification to the OS kernel. This not possible on for example Windows and most mobile operating systems.

To avoid affecting the overall system, as well as the network, we chose to make a daemon rather than a normal application. As the RRC states are per UE, and not per connection on the UE, we believe that having a system-wide "state controller" is the most sensible approach. Also, the daemon is based on events and does not send any data unless instructed to (except during startup). In other words, the daemon is idle most of the time, only waking up when either a lease request arrives or a timer expires.

Even though the daemon itself generates very little data, and uses the T1 timer to limit the number of promotion requests, it has no control over how applications choose to use it. An application can attempt to force CELL_DCH at all times, even if it is not needed. Applications using the SPD should follow normal best practices. For example, staying on CELL_DCH consumes more battery on the UE and generates more data, so it is not in a developers best interest to remain on CELL_DCH. Another option for controlling how the SPD is used, is to limit it to only accept requests from some well-known applications.

In terms of consumption of additional radio resources, which is also a limited resource, we believe that the daemon will mostly be used by applications that would anyway

cause a CELL_DCH promotion. Thus, the radio resource usage will in many cases be the same as without the SPD.

An alternative approach for deciding when to trigger a state promotion, could be based on the work presented in [6]. In this paper, the authors have measured that traffic generated when the screen is on is less delay tolerant than when the screen is off. Instead of applications requesting state promotions, more intelligence could be added to SPD. It could for example listen for system-wide events like typing or other gestures, as well as monitor the currently active application(s), and make a state promotion decision.

4. EVALUATION

To demonstrate the performance gain of preempting RRC state promotions, we built an application retrieving data over HTTP. HTTP is the foundation of the world wide web and a core building block of many delay-sensitive services. One example is the auto suggestion-feature offered by many search engines. They normally send HTTP GET-requests (to retrieve suggestions) as the user types. Also, the request/response pattern used in a typical HTTP-session is very common in other application layer protocols.

4.1 Test setup

Our testbed consisted of a laptop running Ubuntu 12.04, equipped with Huawei E173-modems. The modems were connected to the three MBB networks that are available in Norway, enabling us to analyze the potential gain of preempting RRC state promotions across different providers. As shown in [2], the networks use different RRC state configurations.

The SPD is currently implemented in Python. Since Huawei E173 is equipped with a Qualcomm-chipset, the RRC state could be read from the modem. Our measurement application is also written in Python and made use of *httplib* to retrieve data using HTTP. We measured the effect of preempting state promotion when the initial state was IDLE, and 200 tests were made per ISP for each test parameter combination. The difference between IDLE and CELL_FACH, in terms of state promotion, is mostly that the promotion time is one seconds less when initial state is CELL_FACH.

4.2 HTTP transfer time

According to a survey conducted by Google [16], where they analysed 4.2 billion web pages, the average size of a complete (including all elements) web page is 320 KB, while 90 % of the elements fit within 16 KB. Google has also observed that the fraction of requests sent over cold and warm TCP connections was roughly 33%/67% [15], respectively. A cold TCP connection is defined to be a new TCP connection, while a warm connection is a persistent connection.

Based on these observations, we conducted four sets of measurements, one for each combination of connection type and download file size. The measurements were performed during night, in a business park, to reduce the influence of cross-traffic, and out HTTP server was placed close to all three networks. We only present results from one operator, but the observations are valid for all three. Even though the T1 and T2 timers, as well as the data rate threshold differed, there were no significant differences in the gain of preemption, Thus, for these networks, the configuration does not affect the potential gain of preempting state promotion.

	Avg.	Median	90%	95%	Std. dev
Cold, Non-preemptive	3.33	3.27	3.64	3.72	0.20
Cold, Preemptive	1.39	1.33	1.46	1.80	0.26
Warm, Non-preemptive	3.30	3.28	3.50	3.64	0.15
Warm, Preemptive	1.22	1.19	1.31	1.38	0.18

Table 1: Time (in seconds) it took to complete the 320 KB transfers.

4.2.1 *Large requests*

Large requests are defined as those requests where 320 KB of data was retrieved from the server. The results are summarized in table 1, and there is a significant gain when the state promotion was preempted. The delay introduced by state promotion is eliminated, and the average transfer time was reduced by approximately two seconds. For example, the average transfer time with warm connections and preemption was 1.22 seconds, compared to 3.30 seconds without preemption.

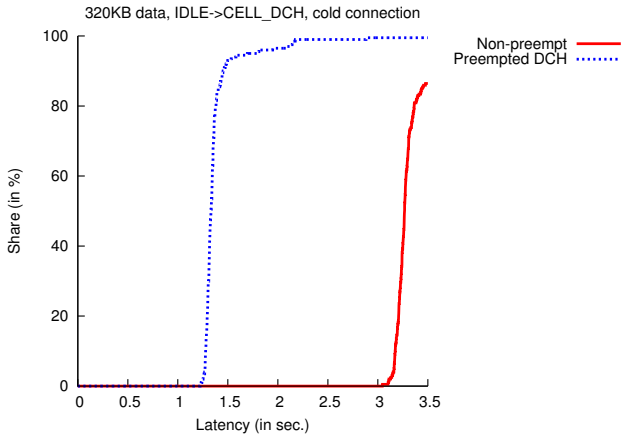

Figure 3: CDF of the transfer time for requesting and receiving 320KB of data over a cold connection, when the initial state was IDLE.

The number of bytes required to trigger a promotion to DCH in the network we used was 1 KB. In other words, by sending only 0.3% more data, a web page loaded in roughly half the time. The difference between preempting and not preempting a request sent through a cold connection is shown as a CDF in figure 3. 95% of the transfers were finished within 1.7 seconds when preemption was used. When preemption was not used, the same share was reached after 3.5 seconds. I.e., the difference was roughly equal to the state promotion time from IDLE to CELL_DCH.

Another interesting observation made from table 1, is that the additional delay caused by establishing a new connection is negligible compared to the state promotion time. The results for cold and warm connections, with or without preemption, is more or less the same.

4.2.2 *Small requests*

Data retrieved through small requests is typically the most time sensitive. Such requests will for example be used to retrieve the earlier mentioned search-engine auto-suggestions, the latest update from a live sports event or new content in a collaborative application. Even though the last two are typically used as examples of push-based services, HTTP itself

	Avg.	Median	90%	95%	Std. dev
Cold, Non-preemptive	2.22	2.14	2.53	2.65	0.30
Cold, Preemptive	0.29	0.16	0.29	1.82	0.49
Warm, Non-preemptive	2.26	2.18	2.54	2.70	0.37
Warm, Preemptive	0.15	0.12	0.19	0.25	0.14

Table 2: Time (in seconds) it took to complete the 16 KB transfers.

does not support push. Instead, push is emulated through polling techniques [8]. Also, native push-applications could for example implement techniques for predicting activity, thereby knowing when to request and maintain an RRC state lease.

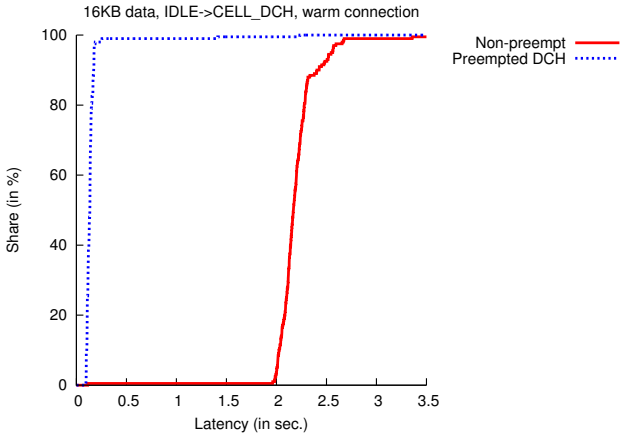

Figure 4: CDF of the transfer time for requesting and receiving 16KB of data over a persistent TCP connection, when the initial state was IDLE.

Figure 4 shows the CDF for retrieving 16 KB over a warm connection. Preempting the RRC state promotion has a significant effect on latency. 95 % of the requests were done within 0.25 seconds, compared to 2.70 seconds for non-preemption. Packet traces showed that the long tail of the preempt result, was caused by a combination of packet loss and buffering in the network.

Table 2 summarizes the results for all combinations of connection type and preemption. We see that the time difference between non-preemptive and preemptive is close to the same as for a large request. This is as expected, as preempting RRC state removes the delay added by the state promotion, which is independent of the number of bytes to be transfered. However, optimising the performance of small requests (or small amounts of data) will in many cases have a stronger impact on the user experience. Within a reasonable time limit, the initial load time for an entire web page is not that important.

5. CONCLUSION

Unlike traditional IP networks, MBB networks are stateful, and RRC state promotions can last for up to two seconds. During these promotions, no IP traffic is let through, which can have a large effect on the user experience. In this paper, we have shown that by preempting the RRC state promotions, the promotion delay can be removed.

We introduce a platform-independent State Promotion Daemon, which attempts to preempt the state promotion based on application-requests. Future work includes designing automatic techniques for deciding when to request state promotions, so that applications do not have to be modified. In addition, we plan to look into the effect of RRC state transitions on different protocols. Our initial focus will be on TCP.

6. REFERENCES

[1] 3GPP. Radio Resource Control (RRC) (V10.3.0), 2011.

[2] Elmokashfi, A., Kvalbein, A., Xiang, J., and Evensen, K. R. Characterizing delays in Norwegian 3G networks. In *PAM 2012* (2012).

[3] Ericsson. Traffic and Market report June 2012 - On the Pulse of the Networked Society. `http://bitly.com/L8PMpc`, June 2012.

[4] Halepovic, E., Pang, J., and Spatscheck, O. Can you GET me now? Estimating the Time-to-First-Byte of HTTP transactions with Passive Measurements. In *Proceedings of the 12th annual conference on Internet measurement* (2012), ACM, pp. 115–121.

[5] Holma, H., and Toskala, A. HSDPA/HSUPA for UMTS: High Speed Radio Access for Mobile Communications.

[6] Huang, J., Qian, F., Mao, Z. M., Sen, S., and Spatscheck, O. Screen-off traffic characterization and optimization in 3G/4G networks. In *Proceedings of the 2012 ACM conference on Internet measurement conference* (New York, NY, USA, 2012), IMC '12, ACM, pp. 357–364.

[7] Krishnan, S., and Sitaraman, R. Video Stream Quality Impacts Viewer Behavior: Inferring Causality Using Quasi-Experimental Designs.

[8] Loreto, S., Saint-Andre, P., Salsano, S., and Wilkins, G. Known Issues and Best Practices for the Use of Long Polling and Streaming in Bidirectional HTTP. RFC 6202 (Informational), Apr. 2011.

[9] Mayer, M. In Search of... A better, faster, stronger Web. `http://bitly.com/adt3Nz`, June 2009.

[10] Miller, R. B. Response time in man-computer conversational transactions. In *Proceedings of the December 9-11, 1968, fall joint computer conference, part I* (New York, NY, USA, 1968), AFIPS '68 (Fall, part I), ACM, pp. 267–277.

[11] Perälä, P., Barbuzzi, A., Boggia, G., and Pentikousis, K. Theory and Practice of RRC State Transitions in UMTS Networks. In *Proceedings of the 5th IEEE broadband wireless access workshop* (2009), IEEE.

[12] Petlund, A., Evensen, K., Griwodz, C., and Halvorsen, P. TCP mechanisms for improving the user experience for time-dependent thin-stream applications. In *The 33rd Annual IEEE Conference on Local Computer Networks (LCN)* (2008), C. T. C. Ehab Elmallah, Mohamed Younis, Ed., IEEE.

[13] Qian, F., Wang, Z., Gerber, A., Mao, Z., Sen, S., and Spatscheck, O. Characterizing Radio Resource Allocation for 3G Networks. In *Proceedings of the 10th annual conference on Internet measurement* (2010), ACM, pp. 137–150.

[14] Qian, F., Wang, Z., Gerber, A., Mao, Z., Sen, S., and Spatscheck, O. Profiling Resource Usage for

Mobile Applications: A Cross-layer Approach. In *Proceedings of the 9th international conference on Mobile systems, applications, and services* (2011), ACM, pp. 321–334.

[15] Radhakrishnan, S., Cheng, Y., Chu, J., Jain, A., and Raghavan, B. TCP Fast Open. In *Proceedings of the 7th International Conference on emerging Networking EXperiments and Technologies (CoNEXT)* (2011).

[16] Ramachandran, S. Web metrics: Size and number of resources. `https://developers.google.com/speed/articles/web-metrics`, Mar. 2012.

[17] Ricciato, F., Coluccia, A., and D'Alconzo, A. Review: A review of DoS attack models for 3G cellular networks from a system-design perspective. *Comput. Commun. 33*, 5 (Mar. 2010), 551–558.

[18] Schurman, E., and Brutlag, J. Performance Related Changes and their User Impact. `http://bit.ly/rGtZWn`, June 2009.

[19] Williams, D. Mobile Broadband and Qualcomm Proprietary Protocols. `http://bit.ly/bjp9Hz`, Apr. 2010.