HTML5 Connectivity Methods and Mobile Power Consumption

Giridhar D. Mandyam
Qualcomm Innovation Center
5775 Morehouse Drive
San Diego, CA 92121 USA
mandyam@quicinc.com

Navid Ehsan
Qualcomm Technologies Inc.
5775 Morehouse Drive
San Diego, CA 92121 USA
nehsan@qti.qualcomm.com

***Abstract*—HTML5 has introduced new connectivity paradigms into the browser, which will allow web developers to leverage connectivity beyond HTTP-based stateless transactions. However, this kind of connectivity requires awareness on the part of the web developer when the execution environment is a mobile browser. Specifically, handheld devices and associated limited battery life should be considered when leveraging new web connectivity enablers. In this work, two recent technologies (WebSockets and WebRTC) will be examined with respect to power consumption. Based on this analysis, recommendations for future standardization work will be discussed.**

Keywords-WebSockets; battery life; AJAX; WebRTC

I. INTRODUCTION

Web services have found their place as a suitable compliment to traditional TCP/IP networking transactions, as they have been able to secure some level of interoperability and therefore allow clients to communicate with servers using protocols that leverage conventional semantics. Web services provide an alternative to binary protocols, which often do not lend themselves to simple interoperability. As a result, browser-based applications have been able to leverage HTTP (hypertext protocol) networking transactions to communicate with servers and as a result provide a more dynamic experience to the end user.

There are standards-based HTTP communications protocols such as SOAP, Simple Object Access Protocol [1]. In addition, there are HTTP-based communications methods that leverage AJAX (asynchronous Javascript and XML [2]) due to their simplicity and universality. AJAX services are normally designed using Representational State Transfer [3], or REST, principles (i.e. the term "RESTful" to characterize an AJAX-based service following REST methodology). AJAX services that are RESTful are in general stateless. This means that a single request coupled with a corresponding response completes the networking transaction. However, certain classes of services require persistent IP-based connections (e.g. instant messaging, opaque data streaming, IP-based telephony). This includes applications that need to listen for server-originated data push. With AJAX this often results in continuous client-originated polling.

HTML5 has introduced new connectivity methods that can provide the web developer with a way to address interactive or real-time communications with better performance than technologies such as SOAP or AJAX could achieve. Two such methods are WebSockets and WebRTC.

The use of either one of these technologies in the context of a mobile browser has the potential to affect the battery life of the host device. Maximizing mobile battery life is a critical challenge in the industry, and many times the burden of minimizing power consumption falls on the mobile operating system and underlying hardware. Nevertheless, browser implementers and web developers also should bear in mind the limited power available on mobile devices and may have to design accordingly. This paper focuses on specific aspects of WebSockets and WebRTC that have the potential to greatly affect power consumption.

II. WEBSOCKETS

WebSockets establish a TCP-based session between a browser and a server, based on a client-server messaging model. Web sockets leverage traditional HTTP handshaking [4], but actually use a binary TCP framing protocol after the handshake has successfully completed. The underlying TCP session can be terminated by either endpoint, as in traditional TCP. The protocol can also run over a secure socket, which not only can ensure a level of data privacy and reliability but also provide a means of transmission through intermediaries or proxies that may not pass through WebSocket sessions otherwise.

While WebSockets provide a convenient means for web developers to maintain a persistent connection between the browser and a server, there are no in-built means of managing the connection when data is not actively being sent. In other words, standard TCP keep-alive procedures are not always supported in the browser implementation of WebSockets. Moreover the current WebSocket API (Application Programming Interface) defined in Javascript by the Worldwide Web Consortium (W3C) [5] does not provide any indication to the web developer as to whether a keep-alive mechanism is in place – only when a connection is torn down.

The WebSocket protocol as defined by the IETF allows for keep-alive traffic via specialized PING/PONG frames (see Sections 5.5.2-5.5.3, [4]). WebSocket signaling corresponding to a PING or PONG frame can be used as a means for keeping the TCP connection alive. These messages may be sent by either communications endpoint, but most browser-based implementations do not send PING's. The PING frame can contain application data, but since it is a control frame it is expected that such instances are rare. A PONG frame is sent in response to PING frames, but the standard allows for PONG frames to be sent unsolicited. An unsolicited PONG frame has no expected response. Note that since there is no explicit guidance as to how user agents should implement the PING/PONG mechanism, web developers may not always rely on this feature to maintain a TCP connection. As a result, developers are sometimes following their own paradigms in keeping the connection alive, e.g. sending application layer keep-alive messages at regular intervals to prevent the WebSocket connection from timing out.

Particularly for cellular connectivity, application-maintained keep-alive mechanisms can have undesired effects; especially considering recent efforts in cellular standards bodies to allow for wireless devices to request transition to low power states of operation. An example of such an enhancement to the UMTS (Universal Mobile Telephony Standard) system that allows for a wireless device (or User Equipment i.e. UE) to request for a transition of the radio access state to a low-power idling mode is known as "Fast Dormancy" [6]. Unfortunately, there is no direct tie-in between how a UE radio transitions to Fast Dormancy, or for any keep-alive mechanism that either the browser or even Javascript running within the browser may implement.

A. Fast Dormancy and Keep-alive Data

UMTS (Universal Mobile Telephony System, sometimes also known as Wideband CDMA i.e. WCDMA) has become one of the most widely-deployed cellular technologies in the "3rd-Generation" family. It was designed to not only provide circuit-switched digital voice services in a spectrally-efficient manner, but also high-speed data for wireless devices. The over-the-air communications interface between a UE (a mobile station, i.e. User Equipment) and Node B (base station) is defined as a layered protocol (Layer 1 being the physical layer, Layer 2 the medium access control layer, and Layer 3 the signaling layer). The IP networking layer is generally considered above Layer 3 in the UMTS protocol model.

The mechanism by which physical layer resources are maintained is radio resource control (RRC). RRC L3 messaging is exchanged between the UE and Node B over-the-air in conjunction with the two basic operating modes of a UE: idle and connected. The idle mode is the default mode of the UE (e.g. when a mobile device is powered up), and usually occurs before the UE has discovered a network to which it can attach. Upon discovering a network by tuning to the transmission of one or more Node B's, the UE can then move into connected mode.

Connected mode includes several RRC sub-states, each having a correspondence to the Layer 1 physical channels that the UE is accessing. The RRC state transitions are normally controlled by the UMTS Radio Access Network (UTRAN), which encompasses the network hierarchy including the actual Node B's that the UE communicates with. The URA (UTRAN Registration Area) defines an area that could encompass one or more Node B's. The UE reads a common Node B-transmitted downstream channel known as the broadcast channel to determine the URA, and must register with the network when it moves into a new URA by sending a message over the upstream common channel known as the RACH (Random Access Channel). This is known as the URA Update procedure.

The Cell_PCH RRC mode occurs when a UE has no dedicated physical resources for a specific connection, but is instead reading a common signaling channel transmitted from the Node B (i.e. the Paging Channel, or PCH). The UE is only accessible via messages transmitted on the PCH. The UE must register upon moving from one Node B to another – this process follows the Cell Update procedure.

The Cell_FACH mode occurs when the UE still has no dedicated connection resources, but is capable of transmitting upstream messages via a common physical channel (RACH) and receiving signaling messages via a downstream common channel known as the FACH (Forward Access Channel). The common channels can also be leveraged to send and receive small amounts of user data, but since these channels use common physical resources the available data throughput tends to be small.

Finally, the Cell_DCH RRC mode occurs when dedicated upstream and downstream channels are provided to the UE. This is normally when TCP/IP sessions can take place. Cell_DCH is usually when power consumption at the UE is at a maximum, as the high-power components in the UE's radio circuitry must be powered on continuously to maintain the necessary link conditions to allow for a TCP/IP session to take place with minimal disruption.

Using Layer 3 signaling based mechanisms, the UTRAN is able to transition the UE from one RRC state to another. However, many UE manufacturers implemented their own heuristics as to when it would be best for a UE to move to a low power mode like idle based upon observed networking activity. Many UE's leveraged a Layer 3 message known as SCRI (Signaling Connection Release Indication) to trigger the UTRAN to release dedicated physical layer resources and transition the mobile to idle mode. The SCRI message was originally meant to indicate that an unrecoverable error had occurred at the UE, but since many UE's used this mechanism for power savings there was much variability in how it was implemented. For instance in [7], HuaWei describes a test of various smartphones in which even phones from the same manufacturer would display times of 3 to 10 seconds after a data transmission was complete before sending the SCRI.

Fast Dormancy was a feature standardized as part of the Release 8 version of UMTS [6]. With this mechanism, the UE is allowed to provide an SCRI with a new Information Element (IE) that includes a cause for the SCRI to be sent. The value in this IE can now indicate that the UE has determined that the data session has terminated, and dedicated physical layer resources may be released. Based upon receipt of this message,

the UTRAN can choose to transition this UE to Idle, Cell_PCH, or URA_PCH modes.

Although the URA_PCH mode and idle mode are roughly equivalent when comparing UE power consumption for typical mobile handheld devices, the necessary UTRAN signaling to sustain a transition from Cell_DCH to Idle is significantly greater than that to transition from Cell_DCH to URA_PCH. The network can further inhibit multiple SCRI's sent in succession by broadcasting an inhibit timer (defined from 0 up to a maximum of 120 seconds).

An application-layer keep-alive mechanism if not executed properly could actually work against Fast Dormancy heuristics in the UE. However, given typical mobile implementation and functionality partitioning, it is almost impossible for a web developer to know how Fast Dormancy is implemented in a UE.

B. *Transitioning Between AJAX and WebSockets*

Recent enhancements to web standards have now introduced a battery API [9] to browsers, thus allowing for both the browser and web developer to have greater control over networking transactions in light of the current battery conditions. Support for this API ensures that the web runtime engine at very least has access to the battery state of the device on which it is running. Since it is assumed that the web runtime engine does not have access to Fast Dormancy timer values, other mechanisms are necessary that allow for continued WebSocket communication while ensuring keep-alive techniques do not inadvertently prevent the UE from transitioning to a low power state.

Recall that the WebSocket protocol provides a signaling mechanism through the PING and PONG messages that allows for some form of keep-alive. Moreover, the web developer can create a keep-alive mechanism by simply sending messages at a regular interval to the server. If the keep-alive interval selected is too small, then it could be problematic in terms of battery life as power savings mechanisms such as Fast Dormancy may never be possible. On the other hand, an interval that is too long could result in termination of the underlying TCP session due to mechanisms such as Fast Dormancy releasing dedicated radio resources. This could result in loss of session continuity between the web application and server.

One possibility is to be able to at least "downgrade" a WebSocket session to an AJAX session when maintaining the WebSocket session is either not desirable or possible. For instance, a web runtime engine upon detecting that the battery level is low abandons any kind of rapid keep-alive mechanism and transitions to a stateless AJAX mechanism (using the XML HTTP Request, or XHR, API in Javascript [10]) for sending messages from the client to the server. Granted, upon doing so the web application will have to resort to the kind of polling mechanism WebSockets was designed to avoid in order to receive messages from the server, but even the polling frequency can be reduced in light of low battery life.

C. *Experimental Results*

In order to verify the potential improvements in battery life, a web application running in Javascript was implemented using the Mozilla Firefox browser running on a Lenovo T420. Cellular connectivity was over AT&T's UMTS/HSPA network. The Firefox implementation of WebSockets and the W3C battery API were leveraged in a Javascript application that adaptively switched between a WebSocket connection and AJAX connection when communicating with a server. The server was a PHP server that services both WebSocket and AJAX requests.

Battery life can be extended by falling back to AJAX as long as the interval between AJAX messages is sufficiently large to as to allow for Fast Dormancy to be utilized. This was borne out in several trials. As an example, starting at nearly full charge the battery level reduced from 99% to 94% over the course of ten minutes with a WebSocket connection along with keep-alive messaging sent every 3 seconds. When the application transitioned to AJAX using the same message payload as was used for keep-alive but with a 20 second inter-message interval, the battery life reduced from 94% to 92% over ten minutes. Therefore a managed transition from WebSockets to AJAX can not only keep client-server sessions alive when the WebSocket networking session is terminated for any reason, but can also improve battery life.

III. WEBRTC

The W3C along with the IETF have been actively developing the necessary specifications to enabler browser-to-browser real-time communications. WebRTC, or Web Real Time Communications, refers to the set of standards currently under development that will enable this feature ([12]). In turn, a new class of web applications should be possible that were once only available to end users in the form of native applications.

WebRTC leverages an out-of-band signaling channel between endpoints for negotiation of the parameters of the call ("Communications are coordinated via a signaling channel which is provided by unspecified means", Section 5 of [12]). Nevertheless, the IETF along with the W3C leverages an offer/answer protocol based on the Session Description Protocol (SDP) [13]. In particular, SDP allows for each endpoint of a WebRTC session to agree upon a suitable choice of multimedia codecs. This section of the paper focuses on the potential impact of the codec on power consumption in a mobile device.

Among the requirements for WebRTC is that when a session is operating over a cellular network, it must be possible to leverage quality-of-service (QoS) mechanisms in the radio access network (see Section 4.2.7, [14]). This can have consequences for mobile power consumption. In this context, QoS mechanisms in UMTS Long-Term Evolution (LTE) cellular systems are considered.

LTE is departure from WCDMA in that the LTE Physical Layer is based on orthogonal frequency division multiplexing (OFDM). A discussion on the physical layer of LTE is beyond the scope of this paper. Nevertheless, physical layer channels

for uplink and downlink signaling and traffic are still applicable. Among the physical layer channels are

- Physical downlink control channel (PDCCH) – This channel provides control information to UE's such as link allocations, uplink transmission acknowledgments, and signaling.
- Physical downlink shared channel (PDSCH) – This channel provides user data for specific UE's. It is 'shared' because it is allocated at different instances of time to different UE's.
- Physical uplink control channel (PUCCH) – This channel is sent by the UE to the LTE base station (eNodeB) to provide control messaging such as scheduling requests (so as to receive physical layer resources for transmission)
- Physical uplink shared channel (PUSCH) – This channel carries user traffic from the UE to the eNodeB.

In LTE voice services are supported via VoIP solutions. This is commonly known as Voice-over-LTE (VoLTE). VoLTE systems leverage QoS mechanisms in LTE. WebRTC would ideally also leverage identical features of LTE, but it is possible to deploy VoIP services on cellular networks without QoS. Such services are also known as over-the-top, or OTT.

LTE QoS for voice services requires regular allocation of physical layer resources so as to ensure VoIP packets are sent with minimal delay and guaranteed data rate. Since resources are scheduled in LTE (i.e. data sent over discrete time periods on the PDSCH or PUSCH), as opposed to being permanently assigned to UE-eNodeB connections, the scheduling mechanisms of LTE are critical to QoS.

Unlike other cellular systems that leverage variable-rate coding, voice services in LTE (along with UMTS) systems is fixed rate. As a result, an endpoint in a voice call alternates between a "talk" and "listen" state. During the "talk" state, the endpoint of a call will generate voice traffic every 20 milliseconds (ms). During the "listen" state, the endpoint will still transmit a Silence Insertion Descriptor (SID) packet every 160 ms. However, the endpoint does not transmit traffic between SID packets and therefore from a UE's perspective it does not need its radio transmitter continuously active. Whether an endpoint is in a "talk" or "listen" state depends on whether voice activity is detected at that endpoint. The "talk" state is where the UE expends power at the highest rate, as both its radio transmitter and receiver hardware is active.

Link allocations, controlled by the eNodeB in LTE, are done on the basis of transmission time interval's (TTI's). A TTI is nominally 1 ms. A UE is also allowed to turn off its receiver when there are no pending uplink or downlink transmissions periodically, also known as discontinuous reception (DRX). DRX is another mechanism by which the UE can attain power savings. Those periods when the UE's receiver and/or transmitter are switched on are referred to as "active time". In addition, short and long DRX periods are defined (known as short and long cycles). The "on duration" for DRX, i.e. the time when the UE is actively monitoring the downlink, is typically 1-4 ms. The inactivity timer, which is the amount of time the UE must continue to monitor the downlink after a transmission or reception of data or control traffic is also typically 1-4 ms.

In addition, LTE (like UMTS) link reliability is enhanced by an hybrid automatic repeat request (HARQ) mechanism ("hybrid" meaning reliability is ensured through a combination of retransmissions and error correcting coding). With respect to DRX, the UE must monitor the PDCCH to determine when a retransmission is necessary prior to shutting off its receiver.

A. Power State Modelling

The UE is required to monitor PDCCH during "active time". During the time when UE is not required to monitor PDCCH, it may choose to turn off some hardware components in order to save power. The extent of the power saving depends on the length of the off cycle. UE power depends on many different factors such as modem components, application, and radio frequency components (RF). For the purpose of this study we use a simple power model as introduced in [15].

We assume "light sleep" and "deep sleep" power states. Deep sleep is typically used for idle mode DRX, and the UE can turn off modem and RF. The problem is that warm-up time is large from this state and therefore not suitable for real-time applications. For small DRX cycles (typically < 40msec) the UE can use light sleep to save power. In light sleep, the UE is likely to just use a low power state for RF. The power level in light sleep is typically much higher when compared to deep sleep, but the transition time in and out of light sleep is much shorter.

In this paper we assume the following power states:

- Power state 1: RRC connected. The UE is actively monitoring downlink (DL) and is also transmitting on the uplink (UL).
- Power state 2: RRC connected. The UE is actively monitoring DL, but there are no uplink transmissions. Transmit (Tx) chain is turned off in this state.
- Power state 3 (light sleep): The UE is not monitoring DL or transmitting on UL. Both Receive (Rx) and Tx are turned off. The modem and RF are put in "light sleep" mode.
- Power state 4 (deep sleep): The UE is not monitoring DL or transmitting on UL. RF and modem are both shut down. The only modem power consumption is due to leakage.

The above power states and the state transitions are illustrated in Figure 1 (dotted transitions are not relevant to VoLTE power consumption).

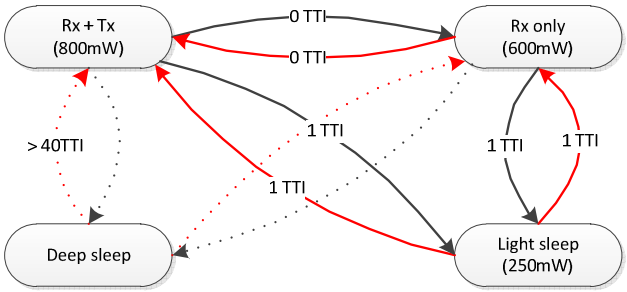

Figure 1 – Modem power states

We define κ_{ij} to be the time (in ms) to transition from power state *i* to power state *j*. In particular, for the purpose of studying short DRX we assume the following:

κ_{21}: Time it takes (in ms) to transition from power state 2 to power state 1. This is the time it takes to turn on the Tx chain. This is on the order of microseconds, and can be assumed to be negligible.

κ_{31}: Time it takes (in ms) to transition from power state 3 (light sleep) to power state 1 (ready to transmit and receive). We assume this will take 2 ms.

κ_{32}: Time it takes (in ms) to transition from power state 3 (light sleep) to power state 2 (ready to receive). We assume this will take 2 ms.

Other transition times are illustrated in Figure 1.

Note that in power state 1, the UE may also be decoding the PDSCH and/or transmitting on the PUSCH. Also in power state 2, UE may be decoding the PDSCH. This can be modeled by adding the additional processing power P_{PDSCH} and/or P_{PUSCH} to the power consumption in any state during the corresponding TTI. For now, we assume due to the small size of VoIP packets the additional overhead for processing PDSCH and PUSCH are negligible. We also assume that all transmissions use the same transmit power.

B. *Scheduling in LTE*

In LTE, uplink allocations are accomplished by UE-initiated scheduling requests (SR's) followed by scheduling grants on the downlink. There are two types of scheduling that are applicable to voice services over LTE: dynamic scheduling and semi-persistent scheduling.

1) *Dynamic Scheduling (DS)*

In this case, the eNodeB configures SR opportunities for the UE (SR periodicity and offset). If UE has a packet to transmit, it is indicated in the corresponding SR transmission. The UE then receives a grant in PDCCH and 4msec later transmits on PUSCH.

In dynamic scheduling, each UL grant for TTI *n* (*n* being a discrete time index) has to be included in the corresponding PDCCH (at time n-4). Furthermore, the UE will have to request a grant by sending SR. Dynamic scheduling is illustrated in Figure 2. As can be seen in Figure 2, there is a delay between the time UE sends the SR and the time the grant is sent in the PDCCH. We call this "SR delay" and assume this to be 2 ms.

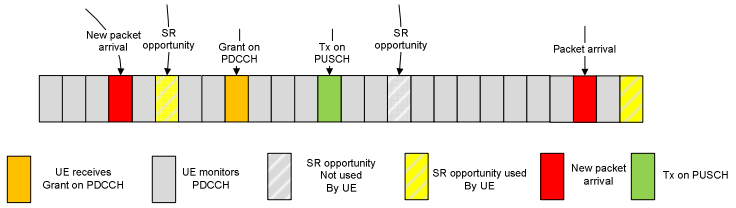

Figure 2 - Dynamic Scheduling

2) *Semi-Persistent Scheduling (SPS)*

Semi-Persistent Scheduling was mainly proposed in order to increase VoIP capacity. It reduces the load on PDCCH and removes the limitation on the control channel. The eNodeB can configure SPS grants for UL and/or DL direction. From a power point of view, there is no difference between SPS and DS for downlink data. Therefore, for the purpose of this study we focus on uplink semi-persistent scheduling.

SPS has an advantage from the UE point of view due to its effect on power saving as well. When SPS is used with DRX, it reduces the number of TTIs when the UE has to be awake and as a result reduces the average power consumption. In this case, UE does not have to request UL grants through an SR and dynamic allocation (see Figure 3).

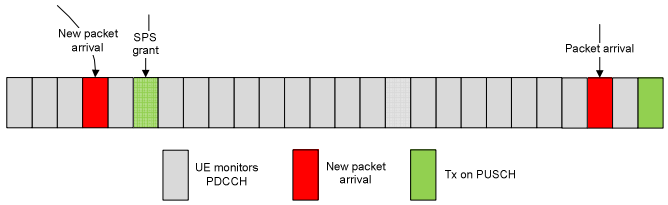

Figure 3 - Semi-Persistent Scheduling

C. *Power Analysis for Voice Services over LTE*

For the purposes of this discussion, we shall focus on the "talk" state and provide an analysis of the difference between voice services leveraging DS and SPS scheduling. This has implications for WebRTC services in the same manner as for VoLTE. We shall also assume the following:

- The probability of the first voice packet transmission being unsuccessfully received and decoded is 10%.
- The maximum number of HARQ retransmissions is 1 (which achieves a packet error rate of 1% based on the previous assumption).
- The on duration is 4 ms. The inactivity timer is 4 ms.
- The power when both Rx and Tx are on is 800 mW (denoted as P_1), where "mW" is short for milliwatts. The power when only the Rx is on is 600 mW (P_2). The power during light sleep (no Rx or Tx) is 250 mW (P_3).

If we examine a 20 ms DRX cycle with dynamic scheduling, we can focus on when (a) no HARQ retransmission

is necessary, and (b) an HARQ retransmission occurs. This is depicted in Figure 4 in terms of discrete TTI's. With respect to the power states that were defined in Section III.A, the probability of the UE being in a particular power state is summarized in Table 1.

The total power can be calculated as

$$P_{Talk\_DS} = 0.105P_1 + .52P_2 + .375P_3 = 489.75 \text{ mW}$$

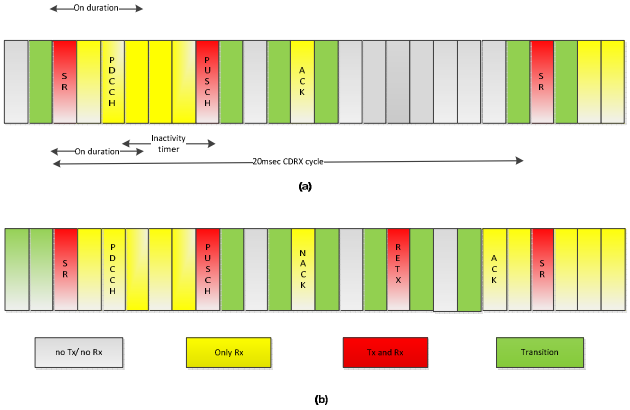

Figure 4 - Dynamic Scheduling, (a) No HARQ retransmission, (b) One Retransmission

State	Definition	First transmission successful	Retransmission required	Total Fraction of Time
State 1	Rx + Tx	0.1	0.15	0.105
State 2	Rx only - Tx off	0.5	0.7	0.52
State 3	no Rx - no Tx (light sleep)	0.4	0.15	0.375

Table 1 - Fraction of Time Spent in Each Power State for DS

A similar analysis can be performed for SPS. SPS with a 20 ms DRX cycle is depicted in Figure 5, with the percentage of time spent in each power state shown in Table 2. The total power P_{Talk_SPS} is calculated to be 394 mW. SPS resulted in nearly a 20% reduction in power consumption.

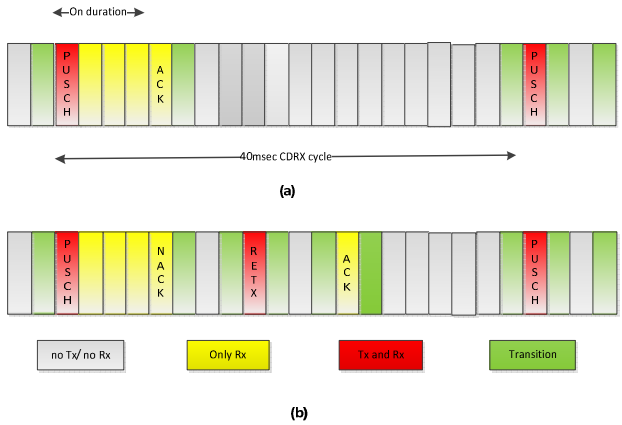

Figure 5 – Semi-Persistent Scheduling, (a) No HARQ retransmission, (b) One Retransmission

State	Definition	First transmission successful	Retransmission required	Total Fraction of Time
State 1	Rx + Tx	0.05	0.1	0.055
State 2	Rx only - Tx off	0.3	0.55	0.325
State 3	no Rx - no Tx (light sleep)	0.65	0.35	0.62

Table 2 - Fraction of Time Spent in Each Power State for SPS

D. Implications for WebRTC

SPS is clearly beneficial for mobile power consumption when IP-based voice services are active. However, OTT voice providers are not expected to leverage SPS. This is because the cellular network provider's network allocation mechanisms require that UE's leveraging SPS are known when the voice session is initiated. This allows the eNodeB scheduler to be QoS-aware. OTT services do not normally leverage QoS in a cellular operators' network. If WebRTC sessions also do not leverage operator QoS, then it is probable that DS will be employed for WebRTC sessions. As has been shown, the increase in power consumption is significant.

IV. Conclusions and Directions for Future Work

HTML5-enabled browsers have great potential to enable a new class of developers for mobile devices. Leveraging HTML and Javascript has the potential to achieve the "write once – run everywhere" ideal for application development. However, both web developers and browser vendors need to become more aware of power limitations on handheld devices and how new connectivity features in HTML5 impact battery life. In this paper, two new features – WebSockets and WebRTC – were analyzed with respect to potentially devastating impacts to power consumption. The W3C can address these kinds of issues in several meaningful ways, among them:

- Development of best practices for mobile web developers, specifically focused on power consumption
- Ensure that the W3C battery API [9] meets a minimum performance standard that would allow web developers to accurately query battery status
- Indication to web developers as to whether cellular QoS is applicable to a browser-originated persistent connectivity session. This could be an explicit notification through a Network Information API [16] or implicitly derived based on existing Javascript API's (e.g. monitoring throughput or power consumption over extended periods of time).
- Explicit metrics regarding the state of the connection propagated through Javascript. Much like the Navigation Timing API [17] that measures page loading speed, similar web performance metrics that measure connectivity statistics for persistent connections would help web developers design applications to account for connectivity situations where battery life could be affected.

REFERENCES

[1] Mitra, Nilo ed. *SOAP Version 1.2 Part 0: Primer*. The World Wide Web Consortium (W3C). W3C Working Draft. December 17, 2001.

[2] Garrett, Jesse James. "AJAX: A New Approach to Web Applications." http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications. February 18, 2005.

[3] Higgins, Bill. "Ajax and REST, Part 1." IBM developerWorks. http://www.ibm.com/developerworks/xml/library/wa-ajaxarch/. October 2, 2006.

[4] Fette, I. and A. Melnikov. *RFC 6455: The WebSocket Protocol*. The Internet Engineering Task Force. December 2011.

[5] Hickson, Ian ed. *The WebSocket API*. The World Wide Web Consortium (W3C). W3C Working Draft. May 24, 2012.

[6] The GSM Association. *Fast Dormancy Best Practices, Version 1.0*. July 27, 2011.

[7] HuaWei Inc. "Behavior Analysis of Smartphone, Version 1.1." 2010.

[8] Chiasserini, Carla-Fabiana and Ramesh R. Rao. "Improving Battery Performance by Using Traffic Shaping Techniques." *IEEE Journal on Selected Areas in Communications*. Vol. 19. No. 7. July 2001. pp. 1385-1394.

[9] Kostiainen, Anssi and Mounir Lamouri ed. *Battery Status API*. The World Wide Web Consortium (W3C). W3C Candidate Recommendation. May 8, 2012.

[10] Van Kesteren, Anne ed. *XML HTTP Request, Level 2*. The World Wide Web Consortium (W3C). W3C Working Draft. January 17, 2012.

[11] Tamplin, J. and T. Yoshino. "A Multiplexing Extension for WebSockets." IETF Draft draft-ietf-hybi-websocket-multiplexing-03. The Internet Engineering Task Force. July 4, 2012.

[12] Bergkvist, Adam et al. *WebRTC 1.0: Real-Time Communications Between Browsers*. The World Wide Web Consortium (W3C). W3C Editors Draft. September 25, 2012.

[13] Uberti, J. and C. Jennings. "Javascript Session Establishment Protocol". IETF Draft draft-ietf-rtcweb-jsep-01. The Internet Engineering Task Force. June 4, 2012.

[14] Holmberg, C. et al. "Web Real-Time Communication Use-Cases and Requirements". IETF Draft draft-ietf-rtcweb-use-cases-an-requirements-09. The Internet Engineering Task Force. June 27, 2012.

[15] Nokia. "DRX Parameters in LTE." R2-071825. 3rd Generation Partnership Project (3GPP). March 2007.

[16] Chitturi, Suresh and Robin Berjon ed. *The Network Information API*. The World Wide Web Consortium (W3C). First Public Working Draft. June 7, 2011.

[17] Wang, Zhiheng ed. *Navigation Timing*. The World Wide Web Consortium (W3C). W3C Proposed Recommendation. July 26, 2012.