POEM: Minimizing Energy Consumption for WiFi Tethering Service

Wan-Seon Lim and Kang G. Shin, *Life Fellow, IEEE, Fellow, ACM*

***Abstract*— Despite the rapidly increasing number of public WiFi hotspots, their coverage is still limited to indoor and office/business environments. WiFi tethering is thus a useful and economic means of providing on-the-go mobile users' Internet connection. One of the main problems of WiFi tethering is the excessive power consumption of mobile access points (APs), or tethered smartphones. Since the power-saving mechanism, defined and deployed in the IEEE 802.11 standard, is intended for clients only, the WiFi interface of a mobile AP never enters the sleep mode. In this paper, we propose a simple but effective system, called power-efficient mobile (POEM), which reduces energy consumption of a mobile AP by allowing its WiFi interface to sleep even during data transfer. The POEM exploits the inherent bandwidth asymmetry between the WiFi and the WWAN interfaces of a mobile AP by buffering the data packets received via the WWAN interface at the mobile AP, thereby allowing the WiFi interface to enter the sleep mode. Compared with the other power-saving solutions for WiFi tethering, the POEM is able to handle various types of applications, such as video steaming and file download, designed to support legacy clients (i.e., clients without POEM), and is also efficient in reducing the clients' energy consumption. We have implemented and conducted an extensive evaluation of the POEM's effectiveness in an android/linux-based test bed. Our experimental results show that the POEM can allow the WiFi interface of a mobile AP to sleep for up to 90% of the total transfer time without significantly affecting system throughput or end-to-end delay.**

***Index Terms*— IEEE 802.11, access point (AP), tethering, power saving, WLAN.**

I. Introduction

TETHERING is to share the Internet connection of a mobile device such as a smartphone, with other nearby devices such as tablets and laptops. Typically, the mobile device is connected to the Internet via WWAN (e.g., 3G or 4G LTE) and communicates with other devices over WiFi or Bluetooth or a physical connection such as a USB cable. If tethering is done over WiFi, then the Internet-connected mobile device (a smartphone) acts as a portable WiFi access point (AP) and other (client) devices connect to the AP with their WiFi interfaces.

WiFi tethering, *a.k.a*. a *Mobile Hotspot* or *Personal Hotspot*, is increasingly popular for users' on-the-go mobile Internet connection. WiFi can support higher bandwidth and longer communication range than Bluetooth. USB tethering requires a physical cable connection and cannot support multiple clients. Most smartphone platforms support WiFi tethering as a built-in feature so users need not install additional software for WiFi tethering. With WiFi tethering, users can go online from their WiFi-enabled devices such as tablets or laptops even when no public hotspots are available. Besides, the users can also use the Internet more securely via WiFi tethering than via an open WiFi hotspot.

Using WiFi tethering, however, significantly increases the power consumption of the mobile device that acts as an AP (which will henceforth be referred to as 'a mobile AP') since its WiFi interface cannot sleep regardless of existence/absence of on-going traffic on the WiFi channel. The IEEE 802.11 standard defines a Power Saving Mode (PSM) [1] that allows the WiFi interface to stay in a low-power (i.e., sleep) state to reduce power consumption. However, PSM is designed *for clients*, not for APs. An AP always stays in a high-power (i.e., active) state according to the IEEE 802.11 standard. Besides, since existing non-mobile APs are externally powered, most power-saving mechanisms have been proposed to enhance the efficiency of the IEEE 802.11 PSM for clients' power savings only.

Recently, Han *et al.* [2] proposed DozyAP for reducing the power consumption of a mobile AP. Their key idea is to put the WiFi interface of a mobile AP in sleep when the WiFi network is idle. However, DozyAP has the following disadvantages. First, it improves the power efficiency of a mobile AP only if the packet inter-arrival time is long. Note that in some popular applications such as video streaming and file sharing, the packet inter-arrival time can be very short. As shown in the Cisco's global mobile data traffic forecast [3] (summarized in Table I), video streaming and file sharing constitute a dominant portion of mobile data traffic. Moreover, due to its per-client polling mechanism, DozyAP requires modification of clients and increases the clients' power consumption.

WiFi Direct is specified by the WiFi Alliance for efficient device-to-device communications. In contrast to the ad-hoc mode of the IEEE 802.11 standard, which has not been successfully deployed in the market, WiFi Direct allows devices to set up an infrastructure-like network. In particular,

Manuscript received March 24, 2015; revised November 2, 2015; accepted March 25, 2016; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor S. Puthenpura. Date of publication May 12, 2016; date of current version December 15, 2016. This work was supported in part by the Daegu Gyeongbuk Institute of Science and Technology Research and Development Program within the Ministry of Science, ICT and Future Planning (MSIP), Korea, through the CPS Global Center, the Global Research Laboratory Program through the National Research Foundation within MSIP, Korea, and the Division of Computer and Network Systems through the U.S. National Science Foundation under Grants CNS-1160775 and CNS-1317411.

W.-S. Lim was with the University of Michigan, Ann Arbor, MI 48109 USA. He is now with the Electronics and Telecommunications Research Institute, Daejeon 34129, South Korea (e-mail: wnsn.lim@gmail.com).

K. G. Shin is with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: kgshin@umich.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2016.2556689

TABLE I
GLOBAL MOBILE DATA TRAFFIC (PB PER MONTH)

	2012	2013	2014	2015
Total	885	1,577	2797	4700
File Sharing	93	142	214	298
Video	455	858	1603	2835

WiFi Direct defines the concept of a P2P Group, where a P2P Group Owner (P2P GO) acts as an AP for a set of connected P2P clients, and introduces a new power-saving mechanism for P2P GO. However, most of existing WiFi devices cannot support WiFi Direct, and it is difficult to apply this new mechanism of WiFi Direct to the current WiFi devices working as a mobile AP. In addition, since WiFi direct is designed for device-to-device communication, its power saving mechanism does not exploit the inherent bandwidth asymmetry between WiFi and WWAN interfaces.

In this paper, we propose a new **Po**wer-**E**fficient **M**obile (**POEM**) AP to reduce the power consumption of mobile APs. It exploits the bandwidth asymmetry between WWAN and WiFi interfaces. In POEM, a mobile AP monitors the WiFi link utilization. If the utilization is lower than a pre-defined threshold, the mobile AP puts its WiFi interface in sleep and buffers data packets received via the WWAN interface for a certain period of time. The WiFi interface is then switched to the active mode and the buffered packets are delivered to the clients. With this buffer-and-forward mechanism, the WiFi interface of a mobile AP can stay in the sleep mode even if the packet inter-arrival time is short. As a result, POEM significantly outperforms DozyAP in terms of energy savings for various types of applications, such as file download or video streaming. Moreover, this buffer-and-forward mechanism of POEM gives the *clients* chances to reduce their energy consumption as well. Another advantage of POEM over DozyAP is that it can reduce the mobile AP's energy consumption even when legacy clients (i.e., clients without POEM) connect to the mobile AP. In POEM, the mobile AP adjusts its sleep period by considering not only the WiFi link utilization but also the existence of legacy clients.

The three main design challenges in POEM are: (1) determining an optimal length of the sleep period to maximize power savings while limiting the additional packet delay, (2) developing a method to make the mobile AP and clients enter/leave the sleep mode synchronously, and (3) enabling the power saving of a mobile AP even when it serves legacy clients. For the first challenge, we design a sleep scheduling algorithm for the mobile AP which determines how long the WiFi interface can sleep based not only on the packet inter-arrival time but also on the WiFi channel utilization. For the second challenge, we design a simple polling mechanism that makes the mobile AP and clients sleep synchronously while incurring low overhead of exchanging control frames. For the third challenge, we design an optional mode of POEM which requires *no modification* to the clients.

The rest of the paper is organized as follows. In Section II, we present the background of POEM, while Section III motivates the need for a new solution like POEM. We describe the design of POEM in Section IV and propose an extension of POEM for eliminating the *client-side* overhead in Section V. We present its detailed evaluation in Section VI and discuss the related work in Section VII. Finally, we discuss future extensions to POEM and draw conclusions in Section VIII.

II. BACKGROUND

A. WiFi Power Savings

For completeness we briefly review the main features of the IEEE 802.11 PSM [1]. The goal of the 802.11 PSM is to put the WiFi interface of a client in the active mode only when he needs to send or receive data, and in the sleep mode otherwise.

The standard 802.11 PSM is called the *static PSM*. It allows a client to sleep as much as possible by putting the client's WiFi interface in sleep if the AP does not have any buffered frames for the client. However, it causes the client to suffer long network latency and low network efficiency due to the requirement of sending a separate PS-Poll frame for each data frame to be downloaded from the AP. To overcome or alleviate this problem, most WiFi devices use an *adaptive PSM* in which a client stays active unless its WiFi interface is idle for a certain period of time (e.g., 200ms in Google Nexus One) [2]. Then, it sends a Null-Data frame with the power management flag set to 1. Upon receiving the Null-Data frame, the AP starts to buffer data frames destined for that client since the client is in the sleep mode.

B. Power-Saving for WiFi Tethering

To the best of our knowledge, DozyAP [2] is the first solution to reduce the power consumption of WiFi tethering by putting the WiFi interface of a mobile AP in sleep when possible. With DozyAP, a mobile AP can put its WiFi interface in the sleep mode to save power when the WiFi channel is idle. When the WiFi interface remains idle for a certain pre-defined time period, denoted as thresh, the mobile AP uses the sleep protocol to check if there are packets buffered at the clients. Although DozyAP is effective in reducing the power consumption of WiFi tethering under certain conditions, it has three main limitations as follows.

Limited Applicability: Because of DozyAP's sleep policy based on the idle time threshold, it can improve the power efficiency of a mobile AP only if the packet inter-arrival time is longer than the threshold. To estimate the gain of DozyAP in terms of power savings, we define an expected sleep period *ES(t)* as:

$$ES(t) = \text{sum of } \{\text{PAI larger than } t - t\}/\text{total time}, \quad (1)$$

where PAI is the packet inter-arrival interval and the total time (in ms) is the interval between the arrival of the first packet and that of the last packet.

As shown in Table I, video streaming and file sharing are the most popular contemporary applications. We measured the PAIs of video streaming by conducting simple experiments. Our testbed is composed of a mobile AP and a client. At the client, we played a 154-second video clip from YouTube while varying the video size (360p, 480p, and 720p.) We recorded

TABLE II
PACKET INTER-ARRIVAL INTERVAL WITH YOUTUBE VIDEO STREAMING

Video size	360p	480p	720p
Average PAI (ms)	11.36	11.77	5.08
Min / Max PAI (ms)	1.01 / 24007	0.01 / 358.40	0.01 / 205.11
ES(200ms) / *ES*(150ms)	0.408 / 0.414	0.041 / 0.062	0.037 / 0.052

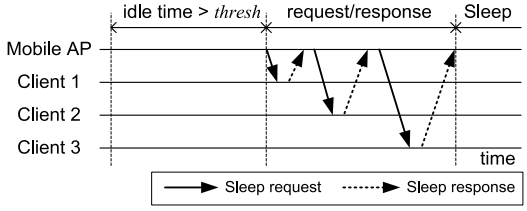

Fig. 1. Timing diagram of DozyAP when there are three clients.

the time when a packet is received from the 3G interface of the mobile AP and then calculated PAI.

Table II shows the experimental results. A Youtube client has a buffer and the packets received from the Youtube server are temporarily stored in the buffer to smooth out bandwidth variations. On starting the video streaming application, the Youtube client aggressively downloads video packets until the number of packets in the buffer reaches a certain threshold, and then stops downloading. When the number of remaining packets in the buffer becomes lower than a given threshold, the Youtube client resumes the downloading.

In case of file sharing applications, since clients continuously send/receive packets, PAI is much shorter. As an example of file sharing, we chose an FTP application and conducted experiments with it. When the client downloads a 25-MByte file from an external FTP server, ES(200ms) and ES(150ms) are just 0.016 and 0.072, respectively.

Inefficient Polling Mechanism: Fig. 1 shows a timing diagram of DozyAP when a mobile AP serves 3 clients. If the WiFi channel is congested or experiences high-level contention, it will take long to complete these frame exchanges. The problem gets worse if some of the clients are in sleep when the mobile AP sends sleep requests. In such a case, the mobile AP has to wait until the sleeping clients wake up and respond. Such a long delay reduces the mobile AP's chance to put its WiFi interface in sleep. Moreover, the sleep protocol of DozyAP increases the power consumption of clients since they have to wake up to exchange the sleep requests and replies with the mobile AP.

Client-Side Modification: DozyAP requires all clients of a mobile AP to participate in the polling procedure. In other words, if there is at least one legacy client, the mobile AP cannot enter the sleep mode for its WiFi interface. This requirement severely degrades the deployability of DozyAP.

III. MOTIVATION

As discussed above, DozyAP can save a mobile AP's power only when PAI is long due to its time-based triggering mechanism and the large overhead of its per-client polling mechanism. With DozyAP, a mobile AP may hardly put its WiFi interface in sleep in the case of file downloading from an FTP or web server and file sharing via a P2P application. In the case of TCP-based video streaming such as Youtube, one of the most popular video websites in the world, DozyAP fails to save the mobile AP's power depending on the size of streamed video used there. The experimental results showed that, even when only one 480p (standard definition) video is streamed via a mobile hotspot, there is little chance for the AP's WiFi interface to enter the sleep mode with DozyAP.

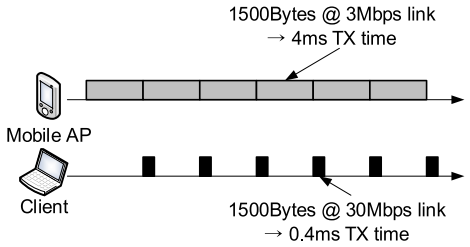

Fig. 2. Packet arrival timing at a mobile AP and a client.

Unlike DozyAP, POEM exploits the bandwidth asymmetry between the WiFi and WWAN (3G or LTE) interfaces of a mobile AP. Usually, the effective bandwidth of a WWAN is lower than that of a WiFi network such as 802.11n can be as high as 100Mbps. Fig. 2 shows a simplified timing diagram of packet arrival at a mobile AP and a client where the effective bandwidths of WWAN and WiFi links are 3Mbps and 30Mbps, respectively. In this example scenario, the mobile AP continuously receives 1500-byte packets from a server and forwards them to the client. One can see that the idle time between successive packet receptions is much larger than the transmission time on the WiFi link, i.e., the WiFi link is severely underutilized.

The key idea of POEM is summarized as follows. The mobile AP under POEM is allowed to enter the sleep mode based not only on the idle time but also on the WiFi channel utilization. When the mobile AP detects the channel utilization of the WiFi link lower than a pre-defined threshold, it checks if there are one or more clients with packets to send. If not, the mobile AP and its clients put their WiFi interface in sleep for a certain period of time. In contrast to DozyAP, the mobile AP under POEM may enter the sleep mode even when PAI is short as in the example shown in Fig. 2. This channel utilization-based sleep triggering mechanism provides the mobile AP much more opportunities to save energy than the time-based sleep triggering mechanism of DozyAP.

Moreover, to overcome the high overhead of per-client polling mechanism of DozyAP, we propose a simple yet efficient polling mechanism to be used by the mobile AP and clients before entering the sleep mode. By reducing the time overhead in exchanging control messages, POEM gives the mobile AP and clients much more chance to put their WiFi interface in sleep. Note that DozyAP increases the power consumption of clients by up to 8% over the legacy mobile AP scenario [2] due to the power consumption for exchanging

control messages. By contrast, POEM is designed to reduce the power consumption of both the mobile AP and its clients.

Finally, in order to improve the feasibility and deployability of POEM, we provide an optional mode which does not require any client-side modification. This optional mode is enabled when one or more legacy clients need to be served, and aims at reducing the AP's power consumption while limiting additional delay or QoS degradation.

IV. Design of POEM

There are two operating modes of POEM: *default* and *optional*. The default mode is activated when all the clients associated with a mobile AP are compatible with POEM (i.e., they support POEM operations). In contrast, the optional mode is designed for the legacy 802.11 stations. The default mode of POEM is detailed in this section while the optional mode will be covered in Section V.

A. Overview

We first provide a high-level overview of POEM and then detail its main components.

In the infrastructure mode WLAN (i.e., WiFi hotspot), the AP periodically sends beacon frames at a fixed beacon interval (BI), which is typically set to 100ms. In POEM, a mobile AP can send two types of beacon frames: *normal* and *extended*. Just before sending a beacon frame, the mobile AP decides on its type according to the sleep scheduling algorithm (to be described in Section IV.B). The *normal* beacon frame is the legacy beacon frame defined in the IEEE 802.11 standard. If the mobile AP sends the *normal* beacon frame, the mobile AP and its clients operate according to the IEEE 802.11 standard until the next beacon is sent/received. Channels are accessed by using the standard DCF procedure.

If the mobile AP decides to sleep, it sends the *extended* beacon frame and checks if one or more clients have uplink data frames to send by using a new polling mechanism (to be described in Section IV.C). If all clients respond with no uplink data frames to send, the mobile AP and the clients put their WiFi interfaces in sleep for K beacon intervals where K is a positive integer value acquired from the sleep scheduling algorithm. While the WiFi interfaces remain in the sleep mode, the mobile AP buffers the data packets received via its WWAN interface. Similarly, if data packets are delivered to a client's MAC layer from its upper layer, the packets are buffered until the WiFi interface wakes up.

In POEM, the mobile AP has three main modules: (1) *traffic monitoring*, (2) *sleep scheduling*, and (3) *sleep polling*. These modules are detailed next.

B. Traffic Monitoring

The traffic monitoring module checks every received or transmitted packet via the WiFi interface in order to update the existence of a delay-sensitive application (E_d), the total transmission time of the mobile AP and its clients (T_m), and the total transmission time of other WiFi devices on the same channel (T_o). These parameters are reported to the sleeping module upon request.

Existence of a Delay-Sensitive Application: Since sleep may incur an additional delay in delivering packets, the sleep duration should be carefully chosen according to the type of application and the delay requirement thereof. If delay-sensitive applications, such as VoIP, video conference, and online gaming, are being run on the devices of mobile hotspot users, the sleep mechanism of POEM may degrade their *Quality-of-Experience* (QoE). The traffic monitoring module checks the existence of packets generated by delay-sensitive applications at each beacon interval. There have been three main approaches to the classification of traffic type: checking the port number at the transport layer [4], accessing the payload of every data packet [5], [6], and using transport/flow-layer statistics [7], [8]. Comparison of these approaches is not in the scope of this paper and *any* of them can be used in POEM. We assume that the monitoring module can detect a packet with data of a delay-sensitive application.[1]

The value of E_d is refreshed at every beacon interval. When the monitoring module detects the transmission of a beacon frame, it resets E_d to 0. Then, the monitoring module checks each of incoming packets (i.e., the packets received via the WWAN interface and destined for hotspot clients) and outgoing packets (i.e., the packet received via the WiFi interface). If there is a packet with data from a delay-sensitive application, E_d is set to 1.

WiFi Channel Utilization: The traffic monitoring module maintains the total transmission time of mobile hotspot (T_m) and the total transmission time of overheard packets (T_o) to estimate the channel utilization. Every time when the monitoring module detects the transmission of an uplink or downlink data frame for the mobile AP (i.e., either the mobile AP sends a data frame or it receives a data frame) over the WiFi channel, it updates T_m as:

$$T_m = T'_m + (T_{data} + tSIFS + T_{ack}) \tag{2}$$

where T'_m is the previous value of T_m; T_{data} and T_{ack} are transmission times for data and ACK frame, respectively; and *tSIFS* is the length of SIFS. (If RTS and/or CTS frames are detected before the data transmission, we add their transmission times.) Note that the WiFi channel is shared with other WiFi devices working on the same channel. When the traffic monitoring module overhears the transmission of a data/control frame from WiFi devices that do not belong to the mobile hotspot, it updates the total transmission time of overheard packets (T_o) as:

$$T_o = T'_o + T_{frame} \tag{3}$$

where T'_o is the previous value of T_o and T_{frame} are transmission times for the overheard frame.

C. Sleep Scheduling

A mobile AP can be in one of 3 possible states: *active*, *idle sleep*, and *forced sleep*. We define two types sleep states,

[1] Alternatively, POEM can be manually enabled/disabled by the mobile AP owner according to the clients' requirements. Since a mobile hotspot is part of a personal network and usually serves only a few clients, such a manual configuration is usually not a problem.

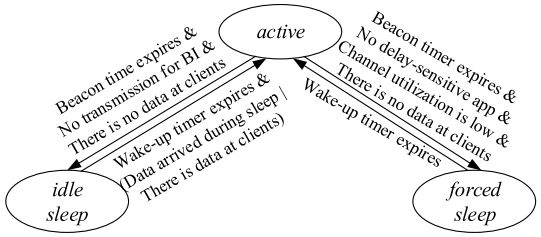

Fig. 3. State-transition diagram of a mobile AP.

idle sleep and *forced sleep*, since in POEM the mobile AP is allowed to put its WiFi interface in sleep not only when the network is idle but also when the network is underutilized. The AP's state may change when it sends a beacon frame to its clients. Upon expiration of the periodic beacon timer or the wake-up timer, the sleep scheduling module obtains the values of E_d, T_m, and T_o from the traffic monitoring module.

Fig. 3 shows a state-transition diagram of the mobile AP. In the *active* state, the mobile AP accesses the channel according to the standard DCF procedure and sends beacon frames at fixed beacon intervals (BIs). When the network is idle, the mobile AP moves to the *idle sleep* state from the *active* state. On the other hand, when the network is underutilized, it moves to the *forced sleep* state. In both *idle sleep* and *force sleep* states, the mobile AP puts its WiFi interface in sleep for K beacon intervals. Whenever the state changes, K', the previous value of K, is updated accordingly. If the previous state was *active* then K' is reset to 0.

Fig. 4 shows the detailed operation of the sleep scheduling module in three different states.

Active → Idle Sleep State Transition: When the beacon timer expires in the *active* state, the sleep scheduling module first checks if there is no downlink packet to the clients and T_m is zero (i.e., there has been no traffic in the mobile hotspot at least for BI). If yes, the sleep scheduling module sets the sleep duration K to K_{min} as shown in the left side of Fig. 4(a). Then, it checks if the clients have one or more uplink data frames to send using the sleep polling mechanism. If some of clients have uplink data frames to send, the mobile AP will stay in the *active* state. Otherwise, the mobile AP moves to *idle sleep* state and puts its WiFi interface in sleep for K beacon intervals.

Active → Forced Sleep State Transition: As shown in the right side of Fig. 4(a), if there is a downlink packet or T_m is nonzero, the mobile AP checks if there are one or more delay-sensitive applications (i.e., $E_d = 1$). If yes, then the AP doesn't move to the sleep mode in order to meet the users' QoE. Otherwise, the mobile AP may move to the *forced sleep* state based on T_m and T_o. Let U_m denote the estimated channel utilization of the mobile hotspot. Since packets were buffered for $K' \times$ BI, U_m is calculated to be $T_m/(K' \times \mathrm{BI} + \mathrm{BI})$. For example, if the previous sleep interval K' is 1, T_m is 20ms, and BI is 100ms, then U_m becomes 0.1. The estimated channel utilization, U_o, of other devices is simply calculated as T_o/BI since there is no

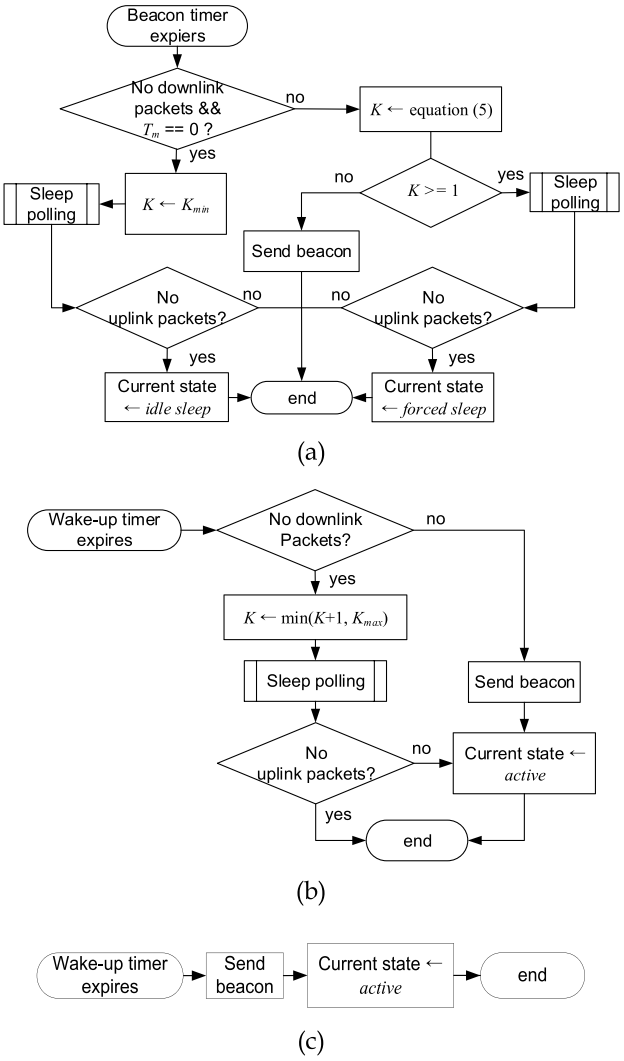

Fig. 4. Flowcharts of the sleep scheduling module in different states: (a) active state, (b) idle sleep state, and (c) forced sleep state.

packet buffering. After calculating U_m and U_o, T_m and T_o are reset to 0.

We use a policy under which all the packets buffered during sleep (i.e., during $K \times \mathrm{BI}$) should be transmitted over the WiFi link for BI, the duration of the next *active* state. Thus, K must satisfy the following condition:

$$(K+1)^* U_m + U_o < 1. \tag{4}$$

Since K is an integer, it becomes

$$K = \min\left(\left\lfloor \frac{1-U_o}{U_m} \right\rfloor - 1, K_{max}\right) \tag{5}$$

where K_{max} is an upper bound of K.[2]

After determining the value of K, if K is equal to, or larger than 1, it moves to the forced sleep state and stays in the sleep

[2]Note that (4) is based on the assumption that the channel utilization remains unchanged for $K+1$ beacon intervals. In reality, this assumption may not hold since the traffic pattern on the channel is dynamic. However, since there is no practical way to accurately estimate the traffic pattern during sleep, we made this assumption.

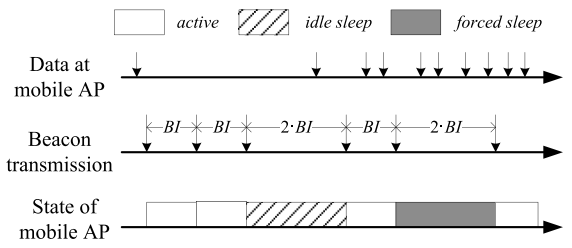

Fig. 5. Beacon transmission timing and change of mobile AP's state.

mode for K beacon intervals. In the *idle sleep* or *forced sleep* state, the mobile AP uses the wake-up timer while turning off the beacon timer.

Operations in Idle Sleep State: Fig. 4(b) shows how the sleep scheduling module operates when the wake-up timer expires in the *idle sleep* state. First, the scheduling module checks if there has been no frame transmission during the last sleep period. If yes, K increases up to K_{max}, i.e.,

$$K = \min(K + 1, K_{max}). \tag{6}$$

Then, the sleep polling mechanism is executed. If none of clients has any uplink frame to send, then the AP's state does not change.

Operations in Forced Sleep State: Fig. 4(c) depicts the sleep scheduling module in the *forced sleep* state. In this case, in order to quickly deliver the frames buffered during the sleep, the AP always moves to the *active* state.

Fig. 5 shows an example of beacon transmissions and change of the mobile AP's state. In this example we assume clients do not have uplink data to send and there is no delay-sensitive application. The initial state of the mobile AP is *active*. When the mobile AP sends the second beacon frame, the network idle time is longer than BI, thus changing the state to *idle sleep*. At that time, K is set to 1. While being in the sleep state, the AP does not receive any data from its WWAN interface. Therefore, the state remains to be *idle sleep* and the value of K becomes 2. Then, the mobile AP receives a data packet during from its WWAN interface the sleep period. Now, the state changes to *active* and the mobile AP delivers the received packet to the client. When the mobile AP sends the fifth beacon frame, the mobile AP changes its state to *forced sleep* with K set to 2. After sleeping for 2×BI, the state changes to *active*.

D. Sleep Polling

In order to initiate the sleep polling procedure, we use an extended beacon (EB) frame. The EB frame has the same format as the legacy 802.11 beacon frame except for the additional leader address (LA) field and sleep interval (SI) field. Before sending an EB frame, the mobile AP randomly selects one of its clients as the leader and put his MAC address in the LA field of the EB frame. It also records the value of K in the SI field.

After SIFS and receiving the EB frame, each client responds as follows[3]:

[3]We adopt the NAK-based polling which is an effective way to reduce the time overhead of getting feedback from multiple Wi-Fi clients [26].

1: Check who is the leader
2: **if** I am the leader **then**
3: **if** I do not have uplink frame **then** send an ACK
4: **else** send a NAK
5: **elseif** I am not the leader **then**
6: **if** I do not have uplink frame **then** do nothing
7: **else** send a NAK

As a result, only if all clients do not have any uplink frame to send, the mobile AP can receive an ACK from the leader. Otherwise, the mobile AP receives a NAK or a corrupted frame. In the former case, the mobile AP sends a *sleep success* (SS) frame while in the latter case, it sends a *sleep cancel* (SC) frame. The mobile AP may not receive any frame when EB frame is lost due to bad channel condition or collision. In such a case, the mobile AP will retransmit the EB frame until the number of retransmission attempts for a given EB frame reaches 4.

Note that the beacon frame transmission is mandatory in WiFi networks, so our polling mechanism requires an additional time to transmit two control frames only, becoming much more efficient than DozyAP that relies on the per-client polling mechanism.

E. Clients' Power Savings

The primary goal of POEM is to enable a mobile AP's WiFi interface to sleep as much as possible to save energy. We have thus far focused on the operation of a mobile AP. In POEM, WiFi clients are required to report the existence of uplink frames to send via the sleep polling procedure, and if the mobile AP decides to sleep (i.e., it sends a SS frame), they also enter the sleep mode. While the mobile AP is in the sleep mode, the clients need not wake up since they cannot receive nor send data frames during the AP's sleep. However, while the mobile AP stays up, some of clients may enter the sleep mode if they don't have active sessions.

POEM allows clients to exploit their own PSM mechanisms (e.g., the *static* or *adaptive* PSM described in Section II.A). A notable requirement is that, if a client enters the sleep mode while the mobile AP is in *active* state, it should periodically wake up to receive every beacon frame sent from the mobile AP. This is to guarantee that all clients participate in the sleep polling procedure.

It is worth noting that POEM can reduce clients' power consumption compared to the legacy mobile AP. In contrast to DozyAP, the mobile AP under POEM may enter the sleep mode even when PAI is short. This provides both the clients and the mobile AP much more opportunities to enter the sleep mode. Moreover, even when the network is idle, DozyAP increases the clients' power consumption by 8% due to its sleep request-reply protocol [2]. With the help of our new polling mechanism that requires a fixed time slot, the clients under POEM can reduce power consumption in exchanging control messages as compared to DozyAP.

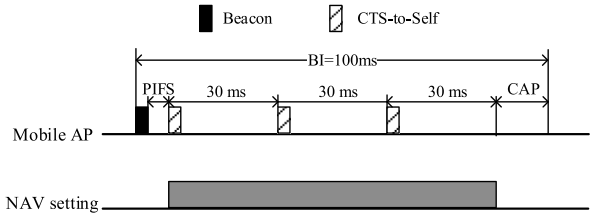

Fig. 6. An example of sleep management in sPOEM.

V. Simple POEM

It is difficult to deploy POEM as is since it requires client modifications for the coordination with the mobile AP. To improve the feasibility and deployability of POEM, we propose a *simple* POEM (sPOEM), an optional mode of POEM for avoiding the client-side modification. sPOEM is enabled when there are one or more legacy clients in the mobile hotspot. Since sPOEM inherits the basic features of POEM, here we only describe the new features of sPOEM.

A. Sleep Management

The sleep polling procedure described in Section IV.D cannot be used in sPOEM as it requires clients' cooperation. The purposes of the sleep polling are to 1) check the existence of uplink packets and 2) make clients stop sending uplink packets during the sleep interval. In sPOEM, the mobile AP does not consider the existence of uplink packets for sleep decisions. To block uplink transmissions from its clients, the mobile AP uses CTS-to-Self frames as follows.

In the beginning of *idle sleep* or *forced sleep* state, the mobile AP sends a CTS-to-Self PIFS after sending a beacon frame. WiFi devices receiving the CTS-to-Self frame set their *Network Allocation Vector* (NAV) to the Duration field of the CTS-to-Self frame. Let T_{CS} denote the value of the Duration field. The WiFi devices receiving the CTS-to-Self frame are not allowed to transmit frames during T_{CS}. The mobile AP is able to send the CTS-to-Self frames at a T_{CS} interval as long as the blocking time by CTS-to-Self frames does not exceed BI. Let N_{CS} denote the number of CTS-to-Self frame transmissions in BI. Fig. 6 shows an example of sleep management in sPOEM where BI is 100ms, and T_{CS} and N_{CS} are 30ms and 3, respectively.

When K is larger than 1, the mobile AP repeats this procedure during the sleep interval. Note that, in contrast to the *default* mode, the mobile AP sends a beacon frame every BI regardless of K value. This is because some WiFi clients perform AP scanning when they lose a certain number of beacon frames. The period between the time when NAV expires and the next beacon transmission is referred to as the *channel access period* (CAP). During a CAP, any WiFi device is allowed to transmit packets on the channel.

The values of T_{CS} and N_{CS} are decided according to BI and CAP. Since the maximum value of Duration field is 32767 (roughly 32 ms) in the IEEE 802.11 standard, N_{CS} and N_{CS} becomes

$$N_{CS} = \lceil (\mathrm{BI} - \mathrm{CAP})/\ 32 \rceil. \tag{7}$$

$$T_{CS} = (\mathrm{BI} - \mathrm{CAP})/N_{CS}. \tag{8}$$

B. Sleep Scheduling

Since CTS-to-Self frames' blocking of transmissions affects all transmissions from WiFi devices on the same channel, the sleep decision should be made differently from the default mode.

Active → Idle Sleep State Transition: sPOEM adopts the same policy to decide on the value of K for *idle sleep* state with the *default* mode. The mobile AP moves from *active* to *idle sleep* state when both T_m and T_o are 0, i.e., there is no transmission on the same channel.

Active → Forced Sleep State Transition: For the *forced sleep* state, it uses a different policy to decide on the value of K from the *default* mode. In particular, since all transmissions were blocked for $K' \times \mathrm{BI}, U_o$ is calculated as $T_o/(K' \times \mathrm{BI} + \mathrm{BI})$ while U_m is obtained in the same way as in the *default* mode. We let K satisfy the following condition:

$$(K+1) * (U_m + U_o) < 1. \tag{9}$$

This is to guarantee that all blocked transmissions during the sleep mode will be transmitted over the WiFi link for BI. Comparing (4) and (9), we can see that sPOEM gives less opportunity for mobile APs to sleep. Since K is an integer, it becomes

$$K = \min\left(\left\lfloor \frac{1}{U_m + U_o} \right\rfloor - 1, K_{max} \right). \tag{10}$$

Operations in Idle Sleep State: The mobile AP overhears the channel for CAP to check the channel is still idle or not. If the channel is not idle or the mobile AP has any downlink packets, the state changes to *active* state. Otherwise, K is increased up to K_{max} according to (6).

Operations in Forced Sleep State: As in the default mode, the AP always moves to *active* state.

Compared to the *default* mode, sPOEM has less chances to move to sleep state, may increase delays of uplink transmissions, and incurs a larger overhead due to frequent sleep and wake-up. Nevertheless, this optional mode has a distinct advantage of feasibility and deployability. As we will see in the next section, sPOEM works well in many scenarios and outperforms DozyAP in certain scenarios.

VI. Evaluation

A. Implementation

We have implemented a prototype of POEM to evaluate its effectiveness. Described below is a brief overview of our implementation. We used five smartphones (LG-D802) during experiments. One of smartphone works as a mobile AP and other four smartphones work as client.

One practical challenge in implementing the sleep scheduling module is to make the WiFi interface sleep (i.e., consume less power) when the mobile AP is in the *idle sleep* or *forced sleep* state. Unfortunately, most of WiFi drivers do not provide the API to put the WiFi interface in sleep when it works as an AP, and thus we could not implement sleep and wake-up operations of POEM.

In our implementation, when the mobile AP's state changes to *idle sleep* or *forced sleep*, the mobile AP records the sleep

interval in a log file and blocks transmissions via the WiFi interface until the state changes to *active*. After finishing each experiment, we estimated how much energy can be saved by POEM as follows. We set the WiFi interface of the smartphone to the station mode, not the AP mode, and connect it to an AP since we can make the WiFi interface sleep in the station mode. Then, we enabled the WiFi PSM of the smartphone and let its WiFi interface sleep and awake according to the history of sleep events stored in the log file and measured the power consumption rate of the mobile AP by using the Monsoon Power Monitor [10].

B. Experimental Setup

We compare the performance of POEM and sPOEM against DozyAP and the standard 802.11 under which the mobile AP's WiFi interface is always awake. DozyAP has 5 system parameters: *thresh*, *min*, *max*, *thresh_l*, and *long*. If the WiFi channel is idle for thresh time units, then the mobile AP enters the sleep mode. In this case, the length of sleep time lies between *min* and *max*. If the WiFi channel is idle for *thresh_l*, then the mobile AP periodically sleeps for a time period of long. We set the parameters as in [2]: *thresh*=150ms, *min*=100ms, *max*=500ms, *thresh_l*=3sec, and *long*=500ms. For POEM, we set K_{min} and K_{max} to 1 and 5, respectively. The PHY modes of WiFi interfaces are set to 802.11n. The transmission rate is adjusted according to the default adaptation mechanism of the legacy WiFi driver. The beacon interval is set to 100ms and the RTS/CTS exchange is disabled. We do not change any parameters for the mobile AP's LTE connection.

We conducted experiments with LTE and 802.11n according to chipset types of our devices. One may speculate that POEM loses its advantage when the speed of WWAN increases with the advance of technology. However, the gap between the speed of WWAN and WiFi will continue to exist. For example, the maximum PHY data rate of LTE-advanced is 3Gbps while that of 802.11ac is 6.93Gbps. Moreover, the effective bandwidth between the correspondent host (e.g., server) and the mobile AP depends on not only the wireless link of LTE but also every link on the path.

The following three applications are used in our experiments.

- Video streaming: The Chrome browser is used as a YouTube streaming client. We test two different video sizes: 480p and 720p (which will be referred to as SD video and HD video, respectively). The running time of the video is 154 seconds. Here Youtube streaming is not regarded as a delay-sensitive application.

- Music streaming: As in the Youtube streaming, the Chrome browser is used as a client application for the Google Music service which plays a 244-second song.

- FTP download: In this case, clients download a 25-MByte file from an external FTP server.

- Chat: For chat we collected 10-minute traces from a simple chat application. We replayed the traces using tcpreplay [24] for a meaningful comparison.

- Web browsing: We collected 10-minute traces from a web browser. We let the users read news from the New York Times website.

To assess the performance of POEM, we evaluate sleep time, energy consumption, average round trip time (RTT) and total transfer time. Sleep time is defined as the ratio of the total sleep time of the mobile AP's WiFi interface to the total application time. For video and music streaming, the total application time is defined as the time to play the video or audio source. For chat and FTP download, the total application time is the time between the first message and the last message and the time spent for downloading the file, respectively. Energy consumption is the energy consumed by the mobile AP. During experiments, we turned off the screen and killed all applications of the mobile AP. Average RTT is the average RTT of TCP packets between the server and the clients. Since we do not fully implement the sleep and wake up operations due to restricted programmability of the Android WiFi driver, RTT does not include the delay overhead of the sleep and wake-up operations. However, it is reported in the literature that the delay overhead is negligible compared to RTT (e.g., lower than 2ms in [25]). Total transfer time is the time elapsed between the time when the mobile AP sends the first packet via the WiFi interface and the time when the mobile AP sends the last packet via the WiFi interface.

C. Experimental Results

Sleep Time: The sleep time is the most intuitive metric for testing the efficiency of POEM. Note that the standard 802.11 does not allow the mobile AP to put its interface in sleep. Therefore, we compare the sleep times of DozyAP, POEM, and sPOEM.

Fig. 7 shows the measurement results. In the case of SD video streaming for one client, DozyAP, POEM, and sPOEM achieve similar sleep times as shown in Fig. 7(a). When there is one SD video stream, the network idle time constitutes a large portion of the total application time (i.e., 154 seconds) and thus DozyAP's sleep mechanism works well. POEM and sPOEM allow the mobile AP's WiFi interface to sleep a bit longer since its sleep is triggered based not only on the idle time (*idle sleep*) but also on the WiFi channel utilization (*forced sleep*). As the number of clients increases, the sleep time decreases for DozyAP due to the reduced the network idle time. In contrast to DozyAP, POEM and sPOEM guarantee long sleeps for multi-client cases. Since sPOEM does not support tight synchronization between the mobile AP and clients, it is not as good as POEM in terms of energy savings. However, it still shows better results than DozyAP.

For HD video streaming, the network idle time is reduced much more than the SD video streaming. As shown in Fig. 7(b), with DozyAP, the mobile AP cannot frequently put its WiFi interface in sleep even when there is only one client. By contrast, POEM guarantees over 45% of sleep time in all the experimental scenarios we considered, thanks to its channel utilization-based sleep policy. sPOEM is also shown to outperform DozyAP.

As shown in Fig. 7(c), the mobile AP can sleep much more in the case of music streaming than the video streaming.

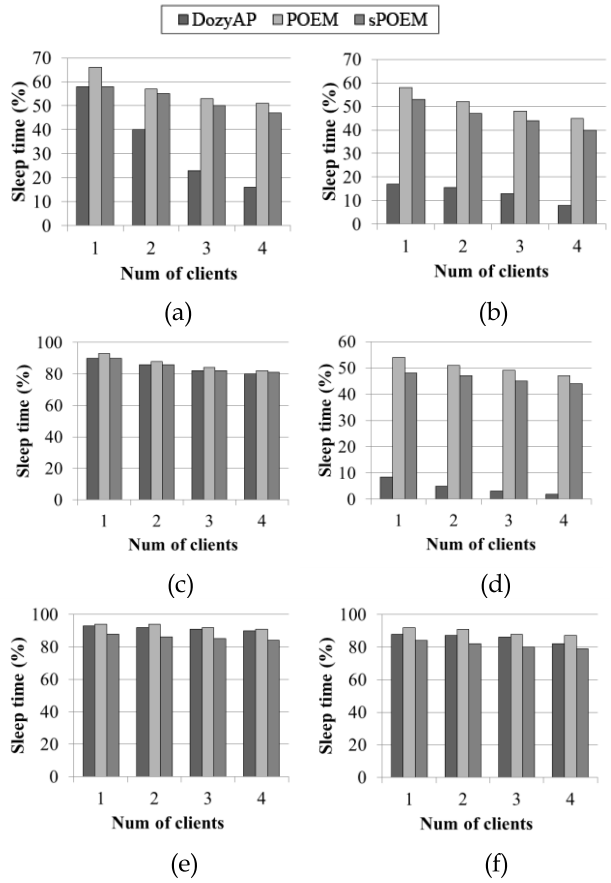

Fig. 7. Sleep time of mobile AP's WiFi interface: (a) SD video streaming, (b) HD video streaming, (c) Music streaming (d) FTP download, (e) Chat, and (f) Web browsing.

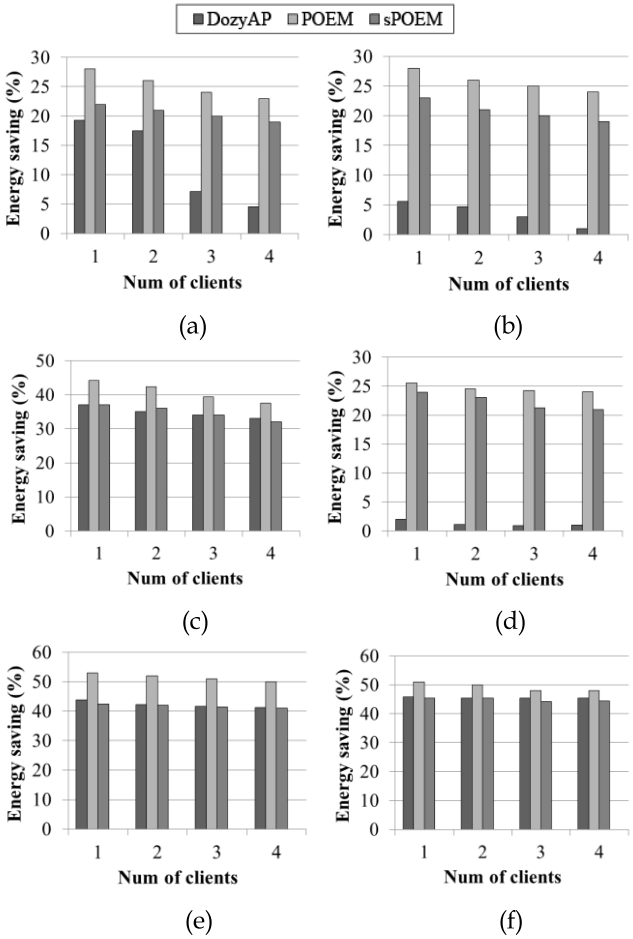

Fig. 8. Mobile AP's energy consumption: (a) SD video streaming, (b) HD video streaming, (c) Music streaming (d) FTP download, (e) Chat, and (f) Web browsing.

When the client application starts playing a song, it aggressively fetches packets that contain data needed to play back the song. Then, until the end of the song, only a small number of control packets are exchanged between the server and the client. For example, the music data fetching time is less than 4 seconds in the one-client scenario, and is less than 11 seconds in the four-clients scenario. Since the total application time in the music streaming is 244 seconds in our experiments, the network idle time is quite long, so DozyAP, POEM, and sPOEM can achieve a long sleep time.

In the case of FTP download, the FTP clients continuously request the FTP server to send data packets by sending TCP ACKs, and hence no long network idle time exists as in the video or music streaming scenario. As shown in Fig. 7(d), DozyAP exhibits about 8% of sleep time in case of one client. Due to the long latency of a LTE network, DozyAP can get a chance to enter the sleep mode for the time period between the transmission of a TCP ACK and the reception of the TCP data packet. However, as the number of FTP flows increases, the sleep time decreases. In contrast to DozyAP, POEM and sPOEM achieve over 40% of sleep time in all cases.

Figs. 7(e) and (f) show sleep time for chat and web browsing, respectively. Since traffic is infrequent in these cases, DozyAP, POEM, and sPOEM can achieve a long sleep time as in the music streaming.

Energy Savings: Fig. 8 shows the average energy savings of the mobile AP. For the video streaming, POEM can reduce power by 22.1–27.3% over the legacy 802.11 mobile hotspot. Compared to DozyAP, POEM can reduce power by 10.1–26.1%. For the music streaming, chat, and web browsing POEM's energy-savings over the legacy system becomes much greater. For the FTP download case, the average power can be reduced by about 25% regardless of the number of clients, compared to both the legacy system and DozyAP. For all cases sPOEM consumes about 4–10% more energy than POEM due to its inefficient sleep and wakeup operations.

It is worth noting that we measured the total energy consumption of the entire system, including the energy consumed by LTE interface and Android OS. If we only consider the energy consumption of WiFi, the energy-savings percentage will be even higher.

Round Trip Time and Total Transfer Time: Making the WiFi interface enter the sleep mode incurs an additional network delay. We evaluate the effects of POEM on the average RTT (Fig. 9) and the total transfer time (Fig. 10). In the case of Youtube streaming, POEM increases the average RTT by 4.2–10.6%, compared to the legacy system, and by 3.4–9.5%, compared to DozyAP. One can see that the

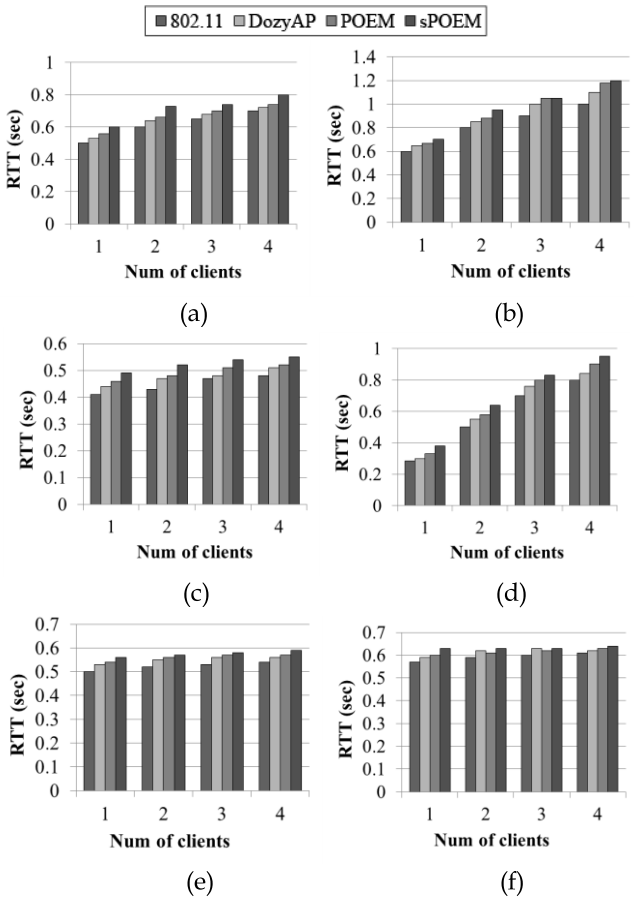

Fig. 9. Average RTT of TCP packets: (a) SD video streaming, (b) HD video streaming, (c) Music streaming (d) FTP download, (e) Chat, and (f) Web browsing.

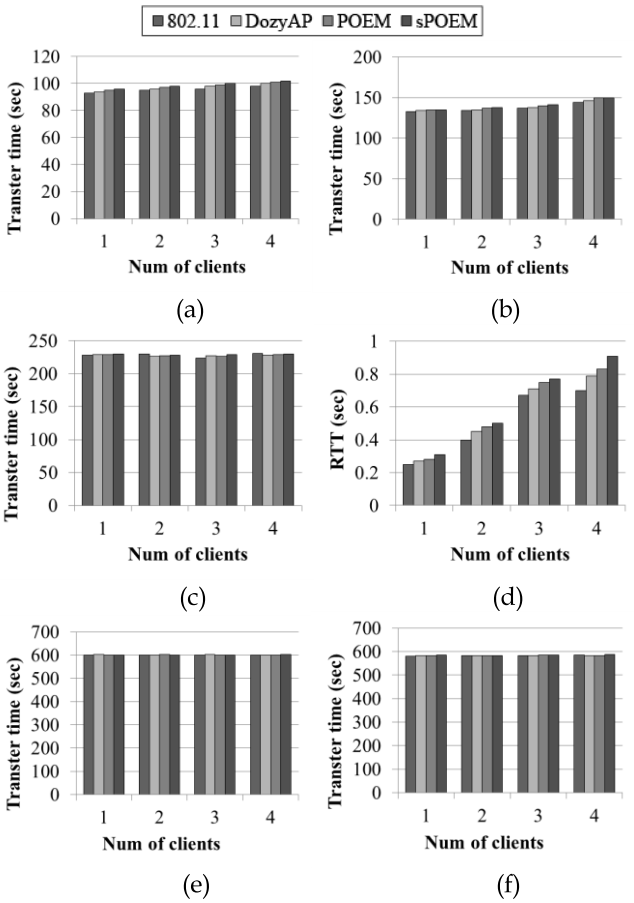

Fig. 10. Total transfer time: (a) SD video streaming, (b) HD video streaming, (c) Music streaming (d) FTP download, (e) Chat, and (f) Web browsing.

increasing rate of RTT is much smaller than the power-saving rate. In addition, an increase of RTT does not directly affect the users' QoE for the Youtube streaming. As shown in Figs. 10(a) and (b), the increasing rate of the total transfer time under POEM is limited, and the total transmission time does not exceed the total application time (154 seconds) in all cases. During experiments, the decoding buffer at each Youtube client never becomes empty. Consequently, applying POEM increases the network latency slightly but does not reduce the users' QoE, i.e., *users can still see the video without any disruption*. sPOEM shows longer RTT and total transmission time than POEM due to retransmissions at the clients. However, the increased latency is small in all cases.

A similar analysis can be applied to the music streaming. The average RTT is increased when POEM or sPOEM is used but the increasing rate is not high. Moreover, we can see that the effect of POEM and sPOEM on the total transfer time is very limited. Recall that the server aggressively sends packets containing actual data when the streaming session is started, and it then periodically sends a small number of control packets to maintain the streaming session. Therefore, buffering at the mobile AP during the sleep period hardly affects the total transfer time.

Unlike the video and music streaming, the performance of FTP download is very sensitive to the increase of network latency. With POEM, the average RTT is increased by 2.6–12.1% compared to the legacy system and by 2.1–9.1% compared to DozyAP. Note that the major portion of RTT is not the additional delay caused by buffering at the mobile AP during the sleep period but the delay between the FTP server and the mobile AP. The increase of RTT directly affects the total transfer time. However, we can see that the power-saving rate for FTP download is much more noticeable than the increasing rate of the total transfer time. With sPOEM, the RTT is increased by about 8% compared to POEM.

In case of the chat and web browsing, DozyAP, POEM, and sPOEM slightly increase RTT. However, such small increases of RTT hardly affect users' QoE.

Clients' Power Consumption: We also measured the power consumption of clients. In our experiments, the clients basically use the adaptive PSM. The time threshold for entering the sleep mode is set to 200ms. In contrast to the mobile AP, we intentionally did not turn off the screens of clients. Fig. 11 shows the experimental results. DozyAP is found to increase the clients' power consumption, compared to the legacy 802.11. This is because the clients have to wake up when they exchange sleep requests and replies with the mobile AP. POEM exhibits the best performance in terms of clients' power consumption. Although POEM has a polling mechanism between the mobile AP and clients, the time overhead of exchanging control frames is much less than that of DozyAP. This result indicates that the energy-savings by

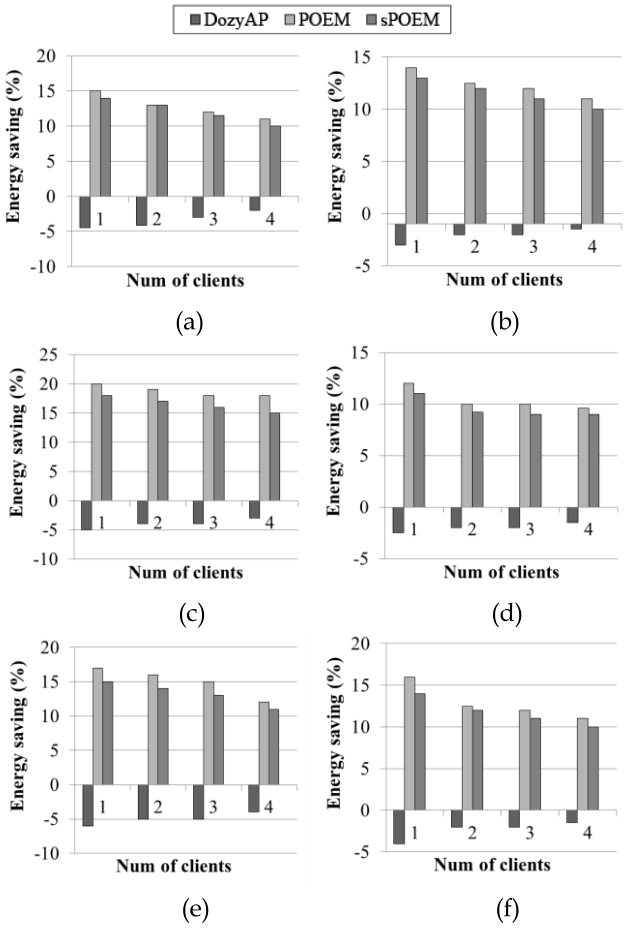

Fig. 11. Clients' average energy consumption: (a) SD video streaming, (b) HD video streaming, (c) Music streaming (d) FTP download, (e) Chat, and (f) Web browsing.

TABLE III
MOBILE AP'S ENERGY SAVING WITH MIXED TRAFFIC WORKLOADS (%)

Traffic type	DozyAP	POEM	sPOEM
480p + Music	7.29	23.89	18.11
480p + FTP	3.38	22.56	20.20
FTP + Music	5.04	23.54	21.07

the efficient sleep policy of POEM significantly outweighs the energy consumption by the polling mechanism. With sPOEM, we can also reduce the energy consumption of clients compared to the legacy 802.11 and DozyAP.

Mixed Traffic Workloads: We measured the energy consumption of the mobile AP's WiFi interface when clients send/receive a mixed type of traffic. In this experiment, two different applications are executed on each of four clients. Due to space limitation, we only present the results of three traffic combinations: 480p video + Music, 480p video + FTP, and FTP + Music (Table III). In all cases, POEM achieves the best performance in terms of reduction of energy consumption.

Impact of Background Traffic: Lastly, we measured the energy consumption of the mobile AP in the presence of background traffic in order to evaluate the effects

TABLE IV
MOBILE AP'S ENERGY SAVING WITH BACKGROUND TRAFFIC (%)

Traffic type	DozyAP	POEM	sPOEM
SD video	3.99	20.45	17.70
Music	34.56	36.21	33.21
FTP	0.31	20.45	1.31

TABLE V
THROUGHPUT OF BACKGROUND TRAFFIC (Mbps)

Traffic type	802.11	DozyAP	POEM	sPOEM
SD video	12.21	11.2	12.0	11.11
Music	15.14	12.22	14.31	13.12
FTP	11.51	9.41	11.1	11.5

of POEM and sPOEM on background traffic. For this, we placed 3 additional laptops nearby our mobile AP network. One laptop works as an AP and other laptops work as the corresponding clients. They use the same channel as the mobile AP networks for WiFi communication. Each of two clients generates a TCP flow to the AP. In the mobile AP network, two clients are used in this experiment.

Tables IV and V show the mobile AP's energy savings and the average throughput of clients that generate background traffic, respectively. From Table IV, we can see that sPOEM is not effective in case of FTP. This is because, when the network is congested, the mobile AP in sPOEM does not try to sleep in order to avoid degrading the performance of background traffic. In contrast, POEM reduces the AP's power consumption regardless of the existence of background traffic. From Table V, we can see that POEM and sPOEM hardly affect other traffic on the same channel.

VII. RELATED WORK

A. Adaptive PSM

There has been extensive research into the optimization of clients' PSM behavior to improve efficiency. Krashinsky and Balakrishnan [14] proposed the bounded slowdown (BSD) protocol to overcome the long delay problem of static PSM in IEEE 802.11. BSD aims to minimize energy consumption while guaranteeing that a connection's RTT does not increase by more than a certain factor over its base RTT. The authors of [15] developed a history-based approach that utilizes the past history to predict the amount of time the network interface spends in sleep state. Wei *et al.* [16] proposed linear prediction-based strategies that reduce the energy consumption to receive multimedia streams by judiciously transitioning the WNIC to the sleep state during no-data intervals in the multimedia stream. These client-side solutions and POEM can be used together with minimal modification to the solutions.

B. AP-Assisted Power Savings

There have been several approaches that allow an AP to help its clients save power. The authors of [17] proposed

a centralized PSM where the AP decides on system parameters, such as the beacon interval, and contention window size, in order to reduce the simultaneous wakeup of clients. In LAWS [18], the AP arranges a wakeup schedule for sleeping clients such that the number of woken-up stations in each beacon interval is balanced. This arrangement reduces the probability of collision, so the client can save more power. Similarly, NAPman [19] considers inter-client beacon staggering to reduce the channel contention among clients in order to improve energy-efficiency. The authors of [20] formulated a downlink scheduling optimization problem aimed at saving energy and proposed heuristic scheduling policies to solve the problem. POEM and these AP-assisted power saving solutions share the fact that the AP's decision affects the sleep interval of clients. However, unlike POEM, these solutions cannot reduce the AP's power consumption that POEM is capable of.

C. Sleep During Data Transfer

In PSM-throttling [21], a client reshapes the TCP traffic into periodic bursts, and then it sleeps and wakes up at the right time based on the predicted packet arrival time in order to minimize the energy consumption. Traffic reshaping is done in two steps, at the TCP and the MAC layers. This client-based approach may result in poor performance when there are many clients, and it also requires modification on both the TCP and MAC layers of clients. POEM aims to minimize client-side modification and support multiple clients efficiently. Catnap [22] exploits the bandwidth discrepancy between the broadband network and the WiFi network to save energy on mobile devices by combining small gaps between packets into meaningful sleep intervals. It requires modification not only on the AP and the client but also on the server. Besides, it is not suitable for video streaming applications.

D. Power-Savings of Mobile APs

Most of existing work considers power-savings of clients only [14]–[22] and little has been done on the power-savings of mobile APs. Cool-Tether [23] proposed a reverse-infrastructure mode for WiFi tethering where a device that has a WWAN interface serves as a WiFi client. However, this reverse-infrastructure mode requires the WiFi interface of the device serving as a WiFi AP to be always awake. Moreover, multiple clients cannot share a single WWAN connection. As we discussed in the previous section, DozyAP [2] is a good solution for power savings of a mobile AP, but it is useful only when the inter-packet arrival time is very long.

VIII. Conclusion

In this paper, we proposed POEM, a novel system for WiFi tethering service, to reduce the power consumption of a mobile AP. POEM utilizes the inherent bandwidth asymmetry between WiFi and WWAN interfaces of a mobile AP by buffering data packets received via the WWAN interface at the mobile AP, thereby allowing the WiFi interface to enter the sleep mode. By conducting extensive experiments, we have demonstrated the effectiveness and feasibility of POEM. Our experimental results have shown that POEM can significantly reduce the power consumption of a mobile AP and its clients at low performance loss.

References

[1] *Wireless LAN MAC and Physical Layer Specifications*, IEEE Standard 802.11, 2007.

[2] H. Han, Y. Liu, G. Shen, Y. Zhang, and Q. Li, "DozyAP: Power-efficient Wi-Fi tethering," in *Proc. MobiSys*, 2012, pp. 421–434.

[3] *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update*. [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html

[4] *Internet Assigned Numbers Authority (IANA)*. [Online]. Available: http://www.iana.org/

[5] A. W. Moore and K. Papagiannaki, "Toward the accurate identification of network applications," in *Proc. 6th Passive Active Meas. Workshop*, 2005, p. 41–54.

[6] S. Sen, O. Spatscheck, and D. Wang, "Accurate, scalable in-network identification of P2P traffic using application signatures," in *Proc. 13th WWW*, 2004, pp. 514–521.

[7] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "BLINC: Multilevel traffic classification in the dark," in *Proc. SIGCOMM*, 2005, pp. 229–240.

[8] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian, "Traffic classification on the fly," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 2, pp. 23–26, 2006.

[9] *The MadWifi Project*. [Online]. Available: http://madwifi-project.org/

[10] *Moonson Power Monitor*. [Online]. Available: http://www.msoon.com/LabEquipment/PowerMonitor/

[11] *Wireless LAN MAC and Physical Layer Specifications, MAC Quality of Service Enhancements*, IEEE Standard 802.11e, 2005.

[12] (2010). *Verizon White Paper, LTE: The Future of Mobile Broadband Technology*. [Online]. Available: https://business.verizonwireless.com/content/dam/b2b/resources/LTE_White_Paper.pdf

[13] S. Fiehe, J. Riihijärvi, and P. Mähönen, "Experimental study on performance of IEEE 802.11n and impact of interferers on the 2.4 GHz ISM band," in *Proc. 6th IWCMC*, 2010, pp. 47–51.

[14] R. Krashinsky and H. Balakrishnan, "Minimizing energy for wireless Web access with bounded slowdown," in *Proc. MobiCom*, 2002, pp. 1–12.

[15] S. Chandra, "Wireless network interface energy consumption implications of popular streaming formats," in *Proc. MMCN*, 2002, pp. 1–15.

[16] Y. Wei, S. M. Bhandarkar, and S. Chandra, "A client-side statistical prediction scheme for energy aware multimedia data streaming," *IEEE Trans. Multimedia*, vol. 8, no. 4, pp. 866–874, Aug. 2006.

[17] Y. Xie, X. Luo, and R. K. C. Chang, "Centralized PSM: An AP-centric power saving mode for 802.11 infrastructure networks," in *Proc. SARNOFF*, 2009, pp. 1–5.

[18] H.-P. Lin, S.-C. Huang, and R.-H. Jan, "A power-saving scheduling for infrastructure-mode 802.11 wireless LANs," *Comput. Commun.*, vol. 29, no. 17, pp. 3483–3492, 2006.

[19] E. Rozner, V. Navda, R. Ramjee, and S. Rayanchu, "NAPman: Network-assisted power management for WiFi devices," in *Proc. MobiSys*, 2010, pp. 91–106.

[20] J. Lee, C. Rosenberg, and E. K. P. Chong, "Energy efficient schedulers in wireless networks: Design and optimization," *Mobile Netw. Appl.*, vol. 11, no. 3, pp. 377–389, 2006.

[21] E. Tan, L. Guo, S. Chen, and X. Zhang, "PSM-throttling: Minimizing energy consumption for bulk data communications in WLANs," in *Proc. ICNP*, 2007, pp. 123–132.

[22] F. R. Dogar, P. Steenkiste, and K. Papagiannaki, "Catnap: Exploiting high bandwidth wireless interfaces to save energy for mobile devices," in *Proc. MobiSys*, 2010, pp. 107–122.

[23] A. Sharma, V. Navda, R. Ramjee, V. N. Padmanabhan, and E. M. Belding, "Cool-Tether: Energy efficient on-the-fly WiFi hot-spots using mobile phones," in *Proc. CoNEXT*, 2009, pp. 109–120.

[24] *Tcpreplay*. [Online]. Available: http://tcpreplay.synfin.net
[25] K.-Y. Jang, S. Hao, A. Sheth, and R. Govindan, "Snooze: Energy management in 802.11n WLANs," in *Proc. CoNEXT*, 2011, Art. no. 12.
[26] J. Villalón, P. Cuenca, L. Orozco-Barbosa, Y. Seok, and T. Turletti, "Cross-layer architecture for adaptive video multicast streaming over multirate wireless LANs," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 4, pp. 699–711, May 2007.

Wan-Seon Lim received the B.S., M.S., and Ph.D. degrees from the Pohang University of Science and Technology, Pohang, South Korea, in 2004, 2006, and 2010, respectively, all in computer science and engineering. He was a Research Fellow with the Department of Electrical Engineering and Computer Science, University of Michigan. He is currently a Senior Researcher of the Electronics and Telecommunication Research Institute. His research interests include future Internet, IoT, wireless LAN MAC, ad-hoc networks, and video streaming over warless networks.

Kang G. Shin is currently the Kevin and Nancy O'Connor Professor of Computer Science with the Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, MI, USA. He has supervised the completion of 77 Ph.D. students, and authored or co-authored over 840 technical articles, one textbook, and more than 30 patents or invention disclosures. His current research focuses on QoS-sensitive computing and networking and on embedded real-time and cyber-physical systems. He is a Fellow of the IEEE and ACM. He received numerous best paper awards, including the Best Paper Awards from the 2011 ACM International Conference on Mobile Computing and Networking, the 2011 IEEE International Conference on Autonomic Computing, and the 2010 and 2000 USENIX Annual Technical Conferences, the 2003 IEEE Communications Society William R. Bennett Prize Paper Award, and the 1987 Outstanding IEEE TRANSACTIONS ON AUTOMATIC CONTROL Paper Award. He also received several institutional awards, including the Research Excellence Award in 1989, the Outstanding Achievement Award in 1999, the Distinguished Faculty Achievement Award in 2001, and the Stephen Attwood Award in 2004 from The University of Michigan (the highest honor bestowed to Michigan Engineering Faculty); a Distinguished Alumni Award of the College of Engineering, Seoul National University, in 2002; the 2003 IEEE RTC Technical Achievement Award; and the 2006 Ho-Am Prize in Engineering (the highest honor bestowed to Korean-origin engineers). He has chaired several major conferences, including 2009 ACM MobiCom, 2008 IEEE SECON, 2005 ACM/USENIX MobiSys, 2000 IEEE RTAS, and 1987 IEEE RTSS. He served on the Editorial Boards, including the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS and *ACM Transactions on Embedded Systems*. He has also served or is serving on numerous government committees, such as the U.S. NSF Cyber-Physical Systems Executive Committee and the Korean Government R&D Strategy Advisory Committee. He has also co-founded a couple of startups.