Wireless Netw (2015) 21:963–980
DOI 10.1007/s11276-014-0798-7

Sleeping mobile AP: a novel energy efficient Wifi tethering scheme

Kyoung-Hak Jung · Jae-Pil Jeong · Young-Joo Suh

Published online: 15 October 2014

Abstract Wifi tethering enables Wifi-only devices to access the Internet by sharing the WWAN (e.g., 3G and LTE) connection of a smartphone where there is no available Wifi access point. However, the current tethering schemes have a limitation as they consume a significant portion of the battery power for providing Wifi clients with the Internet connection. In this paper, we propose a new tethering scheme that reduces the energy consumption of a mobile AP (MAP) without substantial throughput and delay degradation. To improve energy efficiency, the proposed scheme adaptively adjusts the sleep and wake-up periods based on the bandwidth asymmetric feature of the MAP. Further, it provides a longer idle time enough to put the clients into a sleep mode by combining idle periods between subsequent packets, and conserves their energy as well. Our evaluation based on the prototype implementation on commercial smartphones shows that the proposed scheme reduces the energy consumption of the MAP and the client smartphones by up to 56.0 and 8.3 %, respectively.

Keywords Energy conservation · PSM · Wifi tethering · Mobile AP · Smartphones

1 Introduction

Wifi tethering is a feature that shares the Internet connection of a smartphone with other devices in the neighborhood through Wifi. Due to the dramatic increase in mobile devices, this function is recently highly desired to enable Wifi-only devices such as laptops and tablets to access the Internet even though there is no Wifi access point,[1] and thus it is widely supported on most of recent smartphones such as iPhone 4, 4S and 5 (iOS 4.2.5 or later) [4], Windows Phone 7 and 8 [5], and most Android phones [6].

However, existing Wifi tethering schemes rapidly shorten the battery lifetime of a smartphone due to the following reason. When acting as a mobile AP (MAP), the smartphone's Wifi interface always stays in a high power state even when there is no traffic at all. In an effort to save power, Windows Phone simply disables the Wifi tethering function when no traffic activity is observed for several minutes. However, it has a drawback that a user has to go back to the smartphone and re-enable the tethering function automatically disabled whenever the user wants to use the tethering again, and therefore the user may undergo inconvenience. DozyAP [7] addresses this problem, and introduces a simple protocol that puts the Wifi interface into a sleep state only for a certain period when there is no traffic for a time longer than a pre-defined threshold (e.g., 150 ms). However, it still has two limitations. First, the Wifi interface has to remain awake during all idle intervals less than the threshold, leading to waste of energy. Especially, this energy waste could be significant because a MAP's higher speed Wifi link is bottlenecked by its relatively lower speed cellular link (e.g., 150 vs. 14.4 Mbps, where 802.11n (one Wifi antenna) and 3G HSDPA, respectively).[2] Second, although the Wifi link is highly underutilized due to the bottleneck problem, client's Wifi interface hardly goes into a sleep state since it cannot sleep

K.-H. Jung (✉) · J.-P. Jeong · Y.-J. Suh
Pohang, Korea
e-mail: yeopki81@postech.ac.kr

[1] According to a market research [1], 90 % of customers actually prefer Wifi-only devices over the 3G/LTE versions although the coverage of cellular and Wifi networks is 99 versus 49 %, respectively, in the US [2, 3].

[2] The available bandwidth of cellular links could be different depending on network service providers and locations.

in the middle of inter-packet arrivals. As a result, it unnecessarily wastes energy.

This paper proposes a new tethering solution called *sleeping mobile AP* (SMAP) that saves substantial energy of a MAP by putting its Wifi interface into a sleep state when the Wifi link is underutilized. Two important considerations should be addressed in designing a tethering scheme. The first one is that the scheme should not degrade throughput and delay performance substantially. In SMAP, packets arrived when the Wifi network is almost idle are basically buffered in a MAP and then delivered to a client later as a burst, enabling the MAP and the client to exploit idle periods for sleep. To minimize the negative impact caused by the delayed transmission, SMAP adaptively determines the wake-up and sleep durations based on the network activity and the clients' *power-save mode* (PSM) parameters. The second consideration is that a tethering scheme should not sleep without clients' agreement that they will not send any uplink traffic while the MAP sleeps. To avoid possible packet losses, SMAP needs to be implemented in the MAP and the clients, but it requires no major modification of the client by using a loadable module to make it easily deployable.

The rest of this paper is organized as follows. In Sect. 2, we provide findings on existing Wifi tethering schemes and motivate the problem. Section 3 describes the proposed SMAP for saving power followed by the packet delay analysis in Sect. 4. In Sect. 5, we present our experimental environments and evaluation results. Section 6 provides an overview of existing power save mechanisms. Finally, we conclude this paper in Sect. 7.

2 Motivation

While significant efforts to save client's energy have been expended, the same has not been actively addressed for access points (APs) because the APs are usually assumed to be supported by AC power. However, a battery-limited MAP is likely to suffer from significant energy drain like typical mobile devices, and thus it needs to be considered as well. Unfortunately, existing tethering schemes are limited in terms of energy efficiency since they are not designed with thorough consideration of the characteristics of a MAP. In this section, we address the problems with today's Wifi tethering function of commercial Android smartphones using the experimental measurements and motivate the need for a new energy efficient tethering solution by discussing opportunities for conserving power.

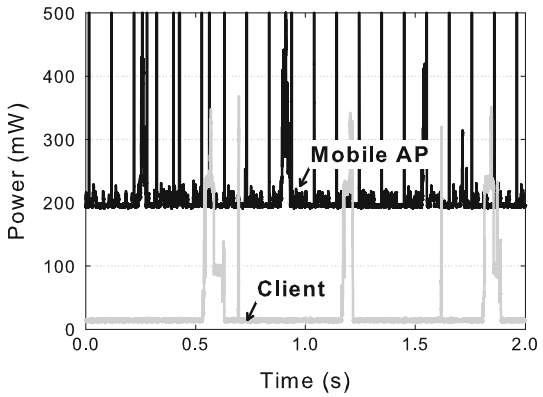


Fig. 1 Power consumption of smartphones working as a mobile AP and a client. Periodic spikes in the figures represent beacon transmission and reception over Wifi

2.1 Impact on energy consumption

Figure 1 depicts the power consumption of two Galaxy Nexus (Android 4.1.1) smartphones when one functions as a MAP that has the 3G connection of a Korean telecommunications operator (SK Telecom) and the other functions as a tethered client. The client turns on the Wifi interface only, while the MAP turns on both Wifi and 3G interfaces. During the experiment we did not generate any traffic to investigate the idle power of a MAP. To measure the amount of consumed power, we used Monsoon Power Monitor [8] which is one of the most widely used external energy meters that monitors the battery power of hand-held devices in real time. As shown in the figure, the power consumption of the client is pretty low because its Wifi interface is in a low power state for most of the time it is idle. As a result, the average power consumption is less than 30 mW. On the other hand, when the Wifi tethering is enabled, the power consumption of the MAP increases significantly. As shown in the figure, the smartphone working as the MAP operates in a high power state constantly (e.g., 200 mW) even though there is no traffic at all. This result indicates that the Wifi tethering leads to severe drop of the battery lifetime. Intuitively, the MAP's Wifi interface should be put into a low power state to reduce the use of the battery power when the network is idle, and thus an energy efficient Wifi tethering solution for a MAP is highly required to solve this problem.

Figure 2(a) represents the power measurement results of a Galaxy Nexus client that receives a 1,024 byte packet every 500 ms. Each down arrow in the figures indicates a packet transmission to the client. As shown in the figure, the client stays in a high power state and consumes more than 100 mW for most of the time. Typically, clients can enter the sleep state when there is no traffic for a certain

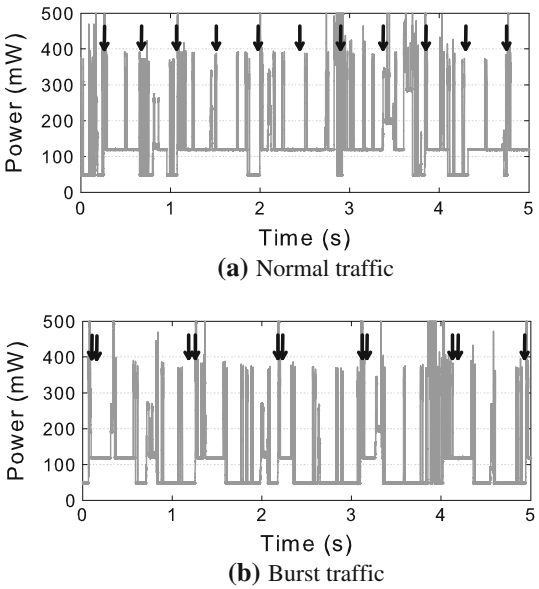


Fig. 2 Power consumption results of a Galaxy Nexus client with different traffic patterns

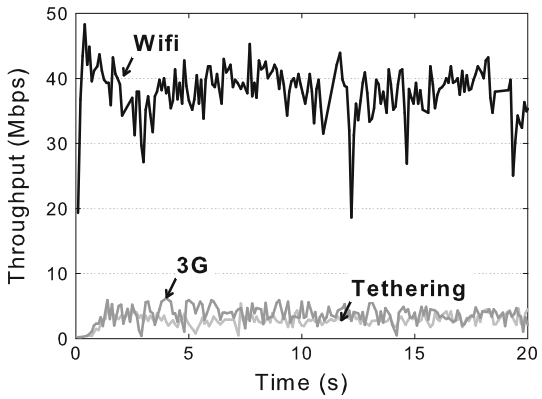


Fig. 3 Throughput measurements for a Wifi client, a 3G client, and a tethered client connected to a mobile AP with 3G Internet connection under a FTP traffic condition

period (which will be discussed in detail later). However, in this case, subsequent packet arrivals hinder the client's transition to sleep mode. This implies that the client's battery exposed to the high voltage will lose the energy quickly and also its lifetime will be shortened. On the other hand, as shown in Fig. 2(b), when the same amount of traffic is generated but transmitted at the different time in bursts, the client can have more opportunities to enter the sleep state by utilizing longer idle periods during data transfers, and thereby it consumes less energy compared to Fig. 2(a). This approach may lead to an additional delay for each packet, but the client still receives the same number of packets during the measurement. That is, it may not degrade users' QoS substantially since the increased delay of a partial of the whole packets is not typically meaningful for users. Rather, in most of applications which are not delay-sensitive, the arrival time of the last packet is more important. Based on this, it is possible to properly reshape traffic pattern by squeezing the constant bit rate traffic, and provide meaningful idle periods enough for clients to enter the sleep state.

2.2 Impact on throughput

Figure 3 shows the bandwidth of a 3G client, an 802.11n Wifi client connected to a normal AP, and a tethered client (a client connected to a MAP with a 3G connection). From the figure, it is observed that the Wifi client achieves eight times higher throughput than the 3G client and the tethered client, on average. In addition, the tethered client may not achieve substantial throughput although it is connected to the MAP over a high-speed Wifi link. This is obvious that all packets the tethered client transmits/receives should be passed through the cellular link of the MAP. As a result, higher speed Wifi is bottlenecked by 3G connection, and the tethered client is not able to fully utilize the Wifi bandwidth. For this reason, the tethered client achieves very similar throughput to the cellular client as shown in the figure.

The above result implies that the MAP and its associated client can put their Wifi interface in the sleep state for a long period because the Wifi interfaces are bottlenecked by cellular connection which offers lower throughput. Therefore, we need to reduce the unnecessary energy waste without a substantial impact on throughput and delay with the consideration of this bandwidth discrepancy between Wifi and 3G connections.

2.3 Impact on packet delay

To study the effect of the IEEE 802.11 *power save mode* (PSM) on packet delay, we measured the packet delay between a client and a server, which are connected via a normal AP, when the server sends downlink traffic with varying the packet generation interval from 10 ms to 5 s. Figure 4 shows the observed delay result using a Galaxy Nexus smartphone. From the figure, we can find that the packet delay is bounded by approximately 300 ms and periodically rounded up to the nearest 300 ms as the packet generation interval increases.

According to the IEEE 802.11 standard [9], a client can use a PSM to conserve its power by switching its Wifi interface between an awake state and a sleep state. A PSM client staying in the sleep state periodically wakes up to listen to a beacon frame, and retrieves the buffered packets

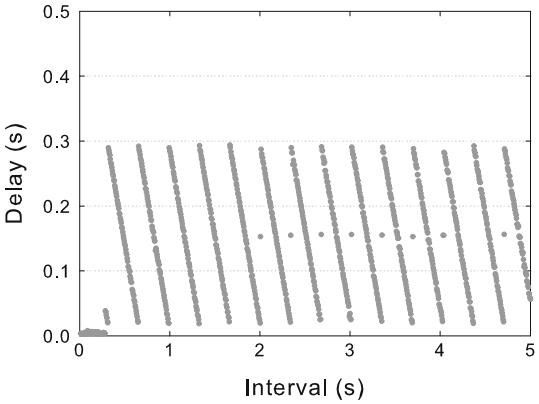


Fig. 4 Packet delay of a client under periodic downlink traffic

from its associated AP if the received beacon frame contains an identifier which indicates the presence of packets destined to itself. This PSM mechanism saves a significant amount of energy by putting the Wifi interface into a low power state for most of time, however, it may hinder users from receiving good quality of service for time-critical VoIP applications, such as Skype and FaceTime, since the PSM may cause intolerable delay. For this reason, most recent smartphones and tablets adopt the *adaptive* PSM (A-PSM) mechanism, which allows the Wifi interface enters a sleep state only when there is no network activity for a certain period. That period, called tail time, ensures that the Wifi interface does not sleep in the middle of packet inter-arrival times. Thus, as shown in the figure, the observed packet delay is greatly reduced until the packet interval reaches 100 ms. However, it is repeatedly rounded up to the nearest 300 ms with increasing the interval due to client's periodic sleep and wake-up.

The above result implies that, the MAP does not need to immediately forward the packets received from the 3G network since the client staying in a sleep state cannot receive them until it wakes up again. This is due to *listen interval* (LI), the periodic intervals that a client awakens to listen to a beacon from an AP. LI is a multiple of the *beacon interval* (BI), the interval between successive two beacons. Therefore, if the MAP can adjust the power state of its Wifi interface, it does not need to immediately wake up its Wifi interface to send data packets until the beginning of the client's next LI.

2.4 Summary

We first measured the power consumption of recent commercial smartphones when they function as a MAP, and identified that the MAP could suffer from a significant energy drain problem even though there is no traffic at all. We then showed that a client consumes energy differently according to the traffic pattern, and found a way to save power by properly reshaping the traffic into bursts without throughput degradation. While the reshaped traffic may cause an extra delay for each packet, it is hardly perceivable by users in general since the delay of partial packets may not degrade user's QoS substantially except in the case of time-critical applications.

We then illustrated that the Wifi link between a MAP and a client is significantly underutilized due to the lower bandwidth of cellular link than the Wifi link. Intuitively, the Wifi interfaces of both MAP and its client could be put into a sleep state for idle periods, and thereby provide an additional opportunity for energy saving without any impact on throughput. Finally, by using the delay measurements at the client side, we showed that a PSM client is unable to receive any packets until it wakes up at the beginning of its next LI. That means, the packet delay may not be significantly impacted although the MAP delays packet transmissions until the client's next LI.

Based on the above observations, we now present a new scheme for saving power of a MAP, called *sleeping mobile AP* (SMAP). The key idea of SMAP is to adaptively put the Wifi interface of a MAP into a low power state and to stay in this state for a certain period depending on the current traffic load. Further, it schedules the best timing to forward packets to a client with a negligible impact on the packet delay, so as to increase the available sleep time of both MAP and client. The detailed operation of SMAP will be explained in the next section.

3 Proposed protocol

As shown in Fig. 5, the overall architecture consists of two parts, the client and the MAP. The SMAP at the client side is implemented between link and IP layers to make it easily deployable and to manage the traffic with minimal overhead. On the other hand, the SMAP at the MAP side is directly implemented on both Wifi driver and firmware. It is composed of four components, *traffic monitor* (TM), *sleep scheduler* (SS), *traffic inactivity timer* (TIT), and *packet buffer* (PB). TM is a component that monitors the amount of current network activity when the Wifi interface is awake. Based on the information, SS calculates the duty cycle to determine the required time to fully deal with the current traffic load over Wifi. TIT is a timer component that determines whether there is a traffic arrival during a certain period, and PB is to buffer the packets arriving from the upper layer during sleep periods. On the other hand, the client has two components, *blocking controller* (BC) and PB as shown in the figure. BC is a component that determines whether to buffer the packets arriving from the upper

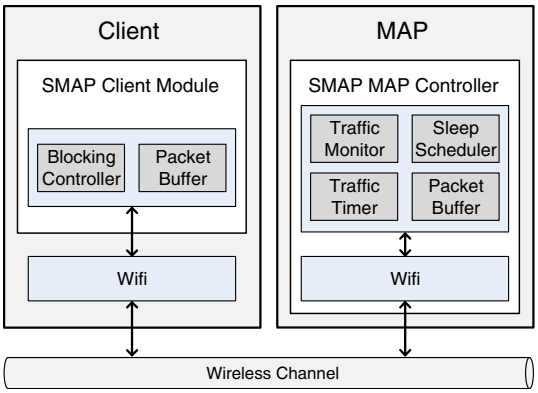


Fig. 5 SMAP architecture

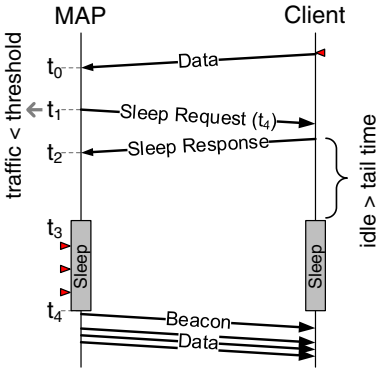


Fig. 6 SMAP basic operation (*Each triangle* on MAP and *client lines* indicates a data packet arriving from the upper layer.)

layer or not, and PB is used for the same purpose as in the MAP.

3.1 Basic operation

When tethering function is enabled, a MAP initially keeps its Wifi interface awake and continuously monitors the network activity by using TM. Based on the observed result, it calculates the current traffic load at the beginning of every BI. If it detects that the Wifi link is underutilized (e.g., the level of network activity is lower than a threshold), the MAP determines the duty cycle of the Wifi interface to coordinate the ratio of its wake-up and sleep periods using SS. Then, after a beacon transmission, the MAP keeps awake during the calculated wake-up period, and then initiates a power negotiation with clients to sleep.

For the power negotiation, the MAP sends a *sleep request* message to its client, similar to DozyAP [7], and puts its Wifi interface in sleep state if the client replies with a *sleep response* message. Especially, each sleep request message contains the sleep period (calculated by SS) for which the MAP wants to sleep. This is to safely avoid any possible packet loss with explicit guarantee that the client would not send uplink traffic until the MAP goes back to an awake state. If no sleep response is received, the MAP simply retransmits the sleep request message to the client. Note that, both sleep request and response messages are transmitted in a unicast manner. This reason is that the MAP should not enter the sleep state without explicit agreements of all clients. If not, an uplink packet can be transmitted while the MAP sleeps since some of clients may not be aware of the sleep schedule, and it will be lost after consecutive retransmission timeouts.

When the MAP puts its Wifi into a sleep state, all packets arriving from the upper layer will be buffered into PB and forwarded to the corresponding client in a burst manner after it wakes up. Then, the MAP repeats the above procedures. This enables both MAP and client to facilitate energy saving by reducing substantial idle listening periods between successive packets. For better understand, Fig. 6 shows the interactions between a MAP and a client. Here is the scenario.

- At time t_0, a MAP receives a data packet from a client while its Wifi interface stays in an awake state.
- At time t_1, the MAP finds out the underutilized Wifi link and sends a sleep request including a sleep duration. In this case, the sleep duration is t_4 as shown in the figure.
- Then, the MAP puts its Wifi into a sleep state after receiving a sleep response from the client.
- For easier deployment, SMAP is implemented as a module and loaded on the client without direct modification to the Wifi kernel driver. For this reason, the client using the normal A-PSM can go to a sleep state after its tail time expires at time t_3.
- For the subsequent packets, the MAP buffers all packets and forwards them as a burst after the sleep finishes at time t_4.

The above algorithm is designed to reduce energy consumption with the minimal adverse impact on the network performance. Of course, it significantly reduces power consumption by introducing a sleep cycle mechanism into the power-hungry Wifi interface, but sometimes limits the energy saving since the continuous traffic monitoring process may be unnecessary if the MAP has no traffic to transmit. To solve this problem, the MAP additionally uses *traffic inactivity timer* (TIT). TIT is a timer that is used to check whether any packet arrives from the client or the cellular network for a certain time period (TIT threshold). Thus, whenever the MAP receives packets before its TIT expires, it simply restarts the TIT. However, if the TIT expires (that means, there is no traffic activity for

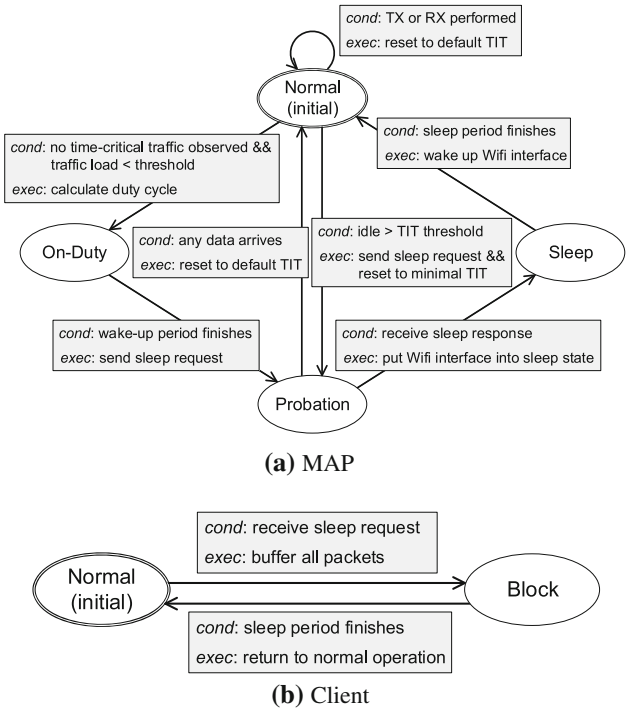


Fig. 7 Transition diagram for a MAP and a client

the time period of TIT threshold), the MAP immediately initiates the power negotiation to put the Wifi interface into a sleep state. This enables the MAP to enter the sleep state more quickly and stay in a low power state for longer time.

However, this imposes two challenges. First, if there is traffic of time-critical applications, the service quality could be degraded due to intolerable packet delay [10]. Fortunately, there has been much work in identifying those traffic from the traffic aggregate [11–13]. With the help of the existing solutions, the MAP can appropriately decide the transition to the sleep state only when no time-critical packets are included in the current traffic. Second, without a careful coordination of wake-up and sleep time, throughput and delay performance can be negatively impacted. To avoid the performance degradation, the MAP adaptively determines the threshold, which will be discussed in detail later (in Sect. 3.3).

3.2 State transition

Figures 7(a) and 8 show the state transition diagram of a MAP and its corresponding pseudo-code, respectively. Note that, the pseudo-code does not show all of the operations for the MAP as the undescribed parts are the same as the standard procedure defined in 802.11. As shown in Fig. 7(a), a MAP has four states: *normal*, *on-duty*, *probation*, and *sleep*. The normal state is the initial state of the MAP. It basically remains in this state when the Wifi tethering function is enabled. In the normal state, the MAP's Wifi interface is in an awake state and it can transmit and receive packets normally. At the same time, it additionally monitors the current traffic activity over the Wifi link while it is awake, so as to estimate the wake-up period required to support the current traffic and to find an opportunity to save power. If the MAP observes no time-critical traffic and the underutilized Wifi link (e.g., *no time-critical traffic && traffic load < threshold*) (line 20), it calculates the duty cycle and enters the on-duty state (lines 21–22). In the on-duty state, the MAP starts a sleep timer set to the wake-up period and exchanges any data packet with the client normally. When the timer expires, the MAP sends a sleep request and then enters the probation state to wait for the client's response (lines 27–29). If the MAP receives a sleep response message, the MAP transits to the sleep state and puts the Wifi interface into a low power state (lines 36–39); if it does not receive the expected sleep response, it retransmits a sleep request. After the sleep period finishes, the MAP changes from the sleep state to the normal state (lines 42–45), and then sends a burst of the packets buffered while it sleeps.

Meanwhile, the MAP in the normal state also constantly checks the presence of data packets using TIT while monitoring traffic load. If the TIT expires (that is, Wifi interface is idle for a time period larger than the TIT threshold), the MAP immediately enters the probation state after sending a sleep request, as shown in the figure (lines 48–51). This is because the TIT expiration indicates that the MAP no longer needs to stay awake since there is no packet to send. After changing to the probation state, the MAP goes to the sleep state if receiving the client's reply, and then goes back to the normal state when it wakes up.

Initially, the MAP uses 150 ms as a default TIT threshold as in DozyAP [7], but re-sets the minimum inter-frame gap[3] (e.g., 1,023 backoff slots plus a succeeding DIFS time) (line 50) when no traffic is observed during the TIT threshold as shown in the figures, so as to provide more opportunities for saving power. By minimizing the TIT threshold, a longer sleep duration is available, ensuring that the MAP and the client can increase the sleep time. Obviously, the TIT threshold is initialized to the default value (150 ms) if a MAP newly receives a packet to transmit (line 8).

Figures 7(b) and 9 represent the state transition diagram and the pseudo-code of a client, respectively. Unlike the MAP, the client has only two states: *normal* and *block*, as shown in Fig. 7(b). In the normal state, the client communicates with the associated MAP normally. If the client

[3] TIT threshold of a MAP should be larger than the maximum backoff period to receive the clients' uplink packets in a conservative manner.

```
1  # Called when SMAP protocol starts
2  procedure BeginSMAP( )
3     Start at the normal state
4     Start TM and TIT # execute TM and TIT
5
6  # On receiving data packets
7  procedure ReceiveDataPackets (packet)
8     Re-set the default TIT threshold
9     if (the current state is the sleep)
10       Buffer packet
11    else
12       Forward packet
13       if (the current state is the probation)
14          Change to the normal state
15          Start TIT
16
17 # Called when the MAP sends a periodic beacon frame
18 procedure SendBeacon ( )
19    Send a beacon frame
20    if (no time-critical and low traffic is observed)
21       Calculate the duty cycle # compute wake-up/sleep time
22       Change to the on-duty state
23       Stop TIT
24       Start a sleep timer with the wake-up period
25
26 # Called when the required wake-up period finishes
27 procedure SleepTimerExpired ( )
28    SendSleepRequest (sleep period of the duty cycle)
29    Change to the probation state
30
31 # Called when the MAP wants to sleep
32 procedure SendSleepRequest (Dur)
33    Send a sleep request message with Dur
34
35 # Called when receiving a sleep response
36 procedure ReceiveSleepResponse (Dur in Sleep Request)
37    Stop TM and TIT
38    Change to the sleep state
39    Put Wifi interface into a low power for Dur
40
41 # Called when the current sleep period finishes
42 procedure SleepExpired ( )
43    Put Wifi interface into a high power
44    Start TM and TIT
45    Change to the normal state
46
47 # Called when there is no traffic until TIT expires
48 procedure TITExpired ( )
49    SendSleepRequest (remaining time until the next LI)
50    Reset TIT to the minimum IFS value
51    Change the probation state
```

Fig. 8 SMAP algorithm of a MAP

receives a sleep request from the MAP and it can agree the request (if there is no packet to send), it replies with a sleep response message and enters the block state (lines 7–9). In this state, the client does not send packets to the MAP anymore until the sleep duration in the sleep request finishes, by buffering data packets arriving from the upper layer. This forces the client's tail time to be expired, and thereby provides an opportunity to save power. After the

```
1  # Called when SMAP protocol starts
2  procedure BeginSMAP( )
3     Change to the normal state
4
5  # Called when receiving a sleep request
6  procedure ReceiveSleepRequest (packet)
7     Change to the block state
8     Start a sleep timer for the duration in packet
9     Buffer all packets arriving from the upper layer
10    Send a sleep response message
11
12 # Called when the current sleep period finishes
13 procedure SleepTimerExpired ( )
14    Change to the normal state
15    if (buffered packets exist)
16       Send buffered packets to the MAP
```

Fig. 9 SMAP algorithm of a client

sleep duration finishes (lines 13–15), the client is able to return to the normal state and operate normally according to the 802.11 standard operation.

3.3 Calculation of duty cycle

As mentioned in the previous section, the sleep cycle of a MAP imposes the challenges to deciding the wake-up and the sleep periods without affecting the throughput and delay performance. To solve those problems, the MAP adaptively calculates the duty cycle every BI using the monitored traffic activity, and starts sleep and wake-up processes. Note that, the duty cycle is the ratio of the wake-up time to the total period. Therefore, the MAP first estimates the required wake-up time to successfully deal with the current traffic load, and then calculates the sleep time of the duty cycle. However, the duty cycle calculation is challenging because the network condition (i.e., traffic pattern and mobility) is likely to change over time. One possible solution is to compute the duty cycle using the entire history of the monitored traffic. This is easy to do since the computation only requires the number of packets, the packet size, and the initial time where the observation began. Unfortunately, this scheme can degrade the user QoS as it takes a long time to adjust to changes due to the old history. In other words, the network throughput (energy saving) will be degraded since the duty cycle cannot be quickly adjusted when the traffic load is drastically increased (decreased).

To make the duty cycle more responsive to changes, SMAP computes it over a sliding window. This means that, the old history will be eventually removed from the history window and no longer included in the duty cycle calculation. However, if the history window does not capture a large enough sample of the traffic patterns, it may fluctuate

and not reach the sweet point. Therefore, SMAP uses one-second history window as the training period (that is, 10 BIs) which hits the sweet point across all the traffic patterns that we studied, so as to adaptively calculate the duty cycle over time.

Based on this, the MAP monitors the ongoing traffic using TM over Wifi link and records the results (the number of packets for each device in the neighborhood and the average packet length) while it awakens. Then, based on (8) in [14], the MAP calculates the expected transmission probability τ_i for device i over Wifi as

$$\tau_i = \frac{C_i F}{Q} \tag{1}$$

where Q is the total period for which the MAP performed traffic monitoring, C_i is the number of packets that device i transmitted, and F is the slot time.

Next, let P_{tr} and P_s be the total probability that at least one packet transmission is initiated and a packet transmission is successful over the shared medium, respectively. Then, they can be derived as

$$P_{tr} = 1 - \prod_{i=0}^{N-1} (1 - \tau_i) \tag{2}$$

$$P_s = \frac{\sum_{x \in G} \tau_x \prod_{y \in G - \{x\}} (1 - \tau_y)}{P_{tr}} \tag{3}$$

where N and G are the total number of devices in the neighborhood and a set of those devices, respectively.

Finally, based on (12) in [14], the normalized duty cycle S required to serve the current traffic while contenting with N devices can be obtained as

$$S = \frac{P_s P_{tr} T_s}{(1 - P_{tr})F + P_{tr} P_s T_s + P_{tr}(1 - P_s) T_c} \tag{4}$$

where T_s and T_c are the average channel busy time when a packet is successfully transmitted and collided, respectively. Thus, $T_s = RTS + SIFS + CTS + SIFS + L + SIFS + ACK + DIFS$ and $T_c = RTS + DIFS$ can be derived since RTS/CTS (or CTS-to-self) needs to be used by 802.11g/n clients to protect subsequent 11g/n transmission, where L is the average packet length (in slot unit).

However, since S ranging between 0 and 1 just represents the ratio of the wake-up time to the total period, it does not indicates the actual time duration. To obtain the wake-up time (as well as the sleep time), the MAP determines the total period, which is the latter goal in this section. This should be carefully determined because the longer total time leads to longer sleep duration and more energy saving, but also increases the packet delay. To minimize the negative impact on the delay, the MAP decides the total period based on client's LI. As discussed in Sect. 2.3, a PSM client stays in a lower state when the network is idle, and it wakes up every LI to retrieve any possible packets buffered at the AP. Based on this, the MAP determines the total period using the remaining time until the beginning of the client's next LI, LI_{remain}, so as to reduce the extra packet delay for the PSM client.

Using those results, the MAP calculates the actual wake-up and the sleep period as follows.

$$T_{wake-up} = S \times LI_{remain} \tag{5}$$

$$T_{sleep} = (1 - S) \times LI_{remain} \tag{6}$$

Figure 10 shows an illustrative example when a MAP and a client communicate with each other. In the figure, each reverse triangle indicates a packet arrival. The MAP first monitors the current traffic load over Wifi while awake (time t_0–t_1). As shown in the figure, the MAP simply forwards a received packet to the corresponding client when it is active. Then, the MAP calculates the duty cycle using the monitored results at the next BI at time t_1. As discussed, the wake-up and sleep durations are determined based on the traffic load and the client's LI. Assume that the normalized duty cycle is 0.2 according to Eq. (4) in Sect. 3.3, and the LI is 3 (the client wakes up every 3rd beacon, as shown in the figure). Then, the MAP decides to keep awake for 60 ms (20 % of 300 ms). During the wake-up period, it normally transmits/receives packets. However, if the wake-up period finishes and there is no packet to send, the MAP transits to the on-duty state and starts the sleep request-response process to sleep until the beginning of the next LI (time t_4). After that, it goes to the sleep state, and all subsequent packets are buffered at the MAP. This allows the client to enter a low power state by forcing its tail time to be expired. Then, when the client's next LI starts at time t_4, both the MAP and the client wake up at the same time, and all of the buffered packets are transmitted to the client as a burst during the new wake-up period.

If the proposed scheme is not applied, the MAP and the client cannot enter the sleep state. However, with the help of duty cycle and traffic reshaping, the MAP and the client have an opportunity to sleep regardless of traffic activity, and thereby conserve their power as shown in the figure.

3.4 Supporting multiple clients

Although multiple clients are associated to the same MAP, SMAP can operate normally. If there is traffic activity, the MAP calculates the transmission probability for each client, the total probability that at least one packet transmission is initiated, and the normalized duty cycle by using the Eqs. (1)–(4), so as to obtain appropriate wake-up and sleep periods, as discussed in Sect. 3.3. Based on the results, the MAP sends a sleep request message to each of clients in a

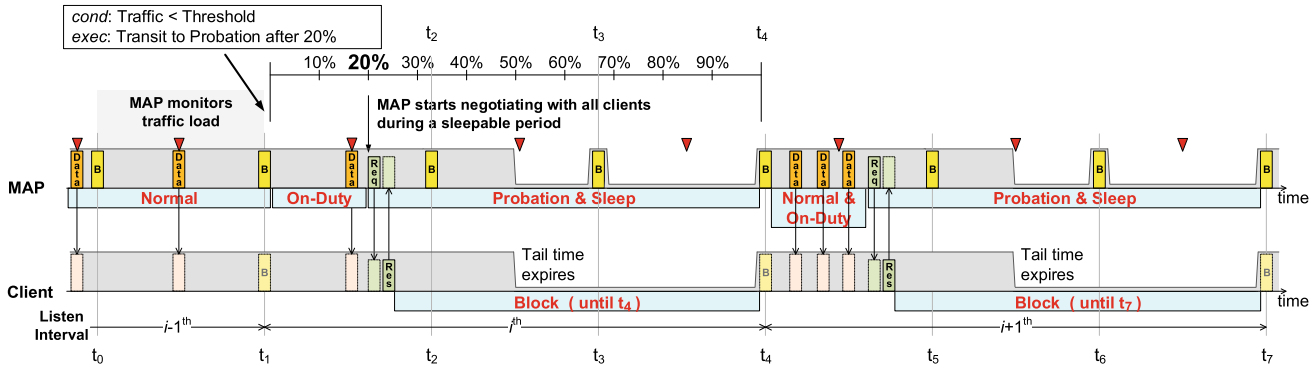


Fig. 10 An illustrative example of a MAP and a client operation (*B*: beacon, *Data*: data packet, *Req*: sleep request, *Res*: sleep response)

unicast manner. The MAP will enter a sleep state when receiving the corresponding sleep response messages from all of its clients, ensuring that all clients agree not to send any packets until it wakes up. However, this will spend more energy and time due to separate negotiations with all of clients. This problem can be more important when multiple clients use different PSM methods. Constantly awake mode (CAM) aiming at maximizing the network throughput constantly keeps awake to immediately deal with possible uplink/downlink traffic, but PSM and A-PSM put the client's Wifi interface into a sleep state when no traffic exists, as discussed in Sect. 2.3. Thus, it is more difficult for the MAP to negotiate with all clients in the coexistence environment, because some of clients may not be unable to participate in the power negotiation due to their own sleep process. For example, with CAM clients, the MAP can immediately start the power negotiation and enter a sleep state when it wants to sleep since the CAM clients always stay in an awake state. On the other hand, when CAM and PSM (or A-PSM) clients coexist, the MAP may not quickly enter the sleep state since it cannot finish the power negotiation with all clients until all of the PSM clients wake up again. As a result, the energy saving of the MAP can be degraded. However, this problem is not serious in most cases because the Wifi tethering is usually used for personal Internet access when there is no available AP. That is, only a few client devices are likely to connect to a MAP (we will discuss the impact of multiple clients later in Sect. 5), and thus substantial performance degradation due to multiple clients may not be induced.

Meanwhile, if multiple clients wake up at the same time for receiving the beacon frame to check for the presence of any buffered packets, they have to contend with each other to retrieve the buffered packets and some of them have to wait in a high power state while one client retrieves a packet. This problem is already addressed and aligned with the AP virtualization technique in [15], but it is hard to be applicable to the MAP. While the virtualization functionality is widely available in most AP (e.g., an Atheros chipset allows up to four virtual APs), it is not included in most of smartphones, and therefore substantial modification needs to be required to leverage the virtual AP function. To mitigate this problem, SMAP exploits the client's LI information. Most of recent smartphones use a multiple of BIs as the LI to reduce energy waste due to the frequent wake-up to receive beacons, and they wake up at least every 2nd beacon. The MAP forces clients to synchronize to different BI by evenly distributing them when they try to associate with the MAP. This reduces the number of clients that wake up at the same time, and thereby mitigates energy consumption caused by contention among them.

4 Delay analysis

In this section, we analyze the delay when a MAP uses the sleep cycle of the proposed scheme, based on M/G/1 queuing model. Depending on the current traffic activity, a MAP transits between awake and sleep states. Let L_n, A_n, and R_n be the queue length after a MAP sends/receives nth packet, the number of packets enqueued during the service time of nth packet, and the number of packets arrived during the sleep duration of a MAP, respectively. In addition, let Q_n be the number of packets arrived before TIT expires, while the queue is empty ($Q_n = 0$ implies that MAP enters a sleep state due to TIT expiration), it is possible to express L_{n+1} as a function of L_n, Q_n, and A_{n+1}, and the sequence $\{L_n; n = 1, 2, \ldots\}$ constitutes a Markov chain [16]. Suppose that p is the probability that the MAP with an empty queue receives $n + 1$th packet when it is in the wake-up period of its duty cycle, then L_{n+1} can be described in the following four cases.

- Case I: As shown in Fig. 11(a), if one or more packets already remain in the MAP's queue ($L_n > 0$) after the nth packet departs, and if zero or more packets additionally arrive during the service time of $n + 1$st

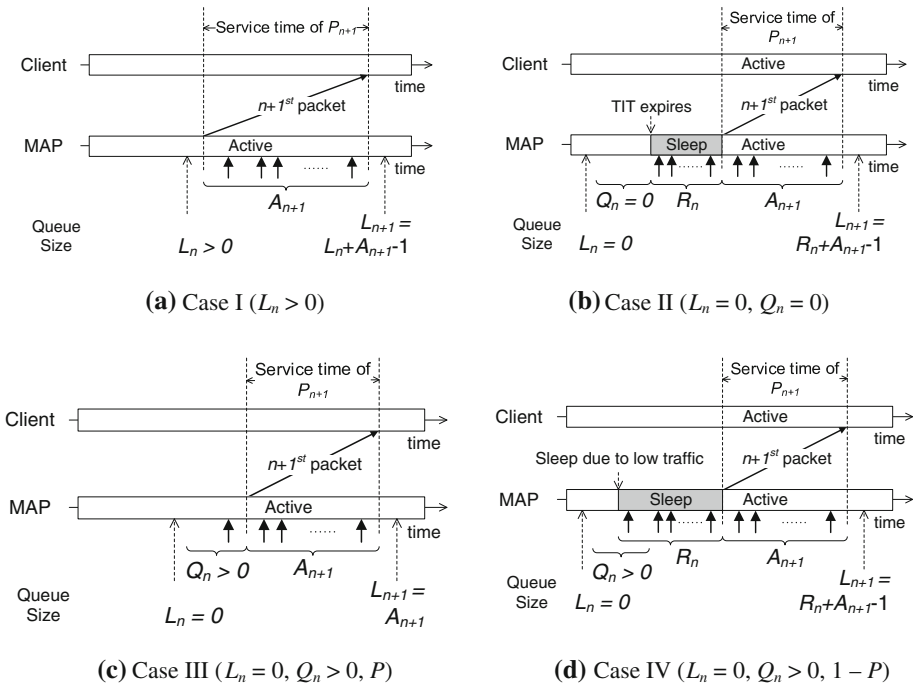


Fig. 11 The number of queue length at a MAP after sending $n + 1$st packet to a client

packet ($A_{n+1} \geq 0$), then the number of buffered packets after $n + 1$th packet is serviced equals to $L_n + A_{n+1} - 1$.

- Case II: While the MAP is in a sleep state ($L_n = 0$, $Q_n = 0$) after TIT expires, one or more packets including $n + 1$st packet could arrive ($R_n \geq 1$), and zero or more packets further arrive during the service time of $n + 1$st packet ($A_{n+1} \geq 0$), then the number of buffered packets after $n + 1$th packet is serviced equals to $R_n + A_{n+1} - 1$. Figure 11(b) shows an example when the MAP receives packets after TIT expires.
- Case III: As shown in Fig. 11(c), there is no packet in the queue ($L_n = 0$) when nth packet departs. If $n + 1$st packet arrives before TIT expires ($1 - \Pr[Q_n = 0]$) and the MAP's wake-up period finishes (p), and if zero or more packets arrive during the service time of the $n + 1$st packet ($A_{n+1} \geq 0$), then the number of buffered packets after $n + 1$th packet is serviced equals to is A_{n+1}.
- Case IV: There is no packet in the queue ($L_n = 0$) when nth packet departs. If $n + 1$st packet arrives before TIT expires ($1 - \Pr[Q_n = 0]$), but the MAP already enters a sleep state due to underutilized Wifi link ($1 - p$), and if zero or more packets arrive during the service time of the $n + 1$st packet ($A_{n+1} \geq 0$), then the number of packets arrived after $n + 1$th packet is serviced equals to is $R_n + A_{n+1} - 1$ (see Fig. 11(d)).

As a result, the queue length after $n + 1$th packet is serviced can be derived as:

$$L_{n+1} = \begin{cases} L_n + A_{n+1} - 1, & \text{for } L_n > 0 \\ R_n + A_{n+1} - 1, & \text{for } L_n = 0,\ Q_n = 0 \\ A_{n+1}, & \text{for } L_n = 0,\ Q_n > 0,\ p \\ R_n + A_{n+1} - 1, & \text{for } L_n = 0,\ Q_n > 0,\ 1 - p. \end{cases} \tag{7}$$

Meanwhile, let π_k and L be the probability of the queue length being k and the queue length in the steady state, respectively. Then π_k can be denoted as $\pi_k = \sum_{j=0}^{\infty} \Pr[L_n = j] \times \Pr[L_{n+1} = k | L_n = j]$ and also $\pi_k = \Pr[L = k]$ since the next queue length is basically influenced by the previous length, where $\sum_{j=0}^{\infty} \Pr[L_n = j] = 1$. Based on the above equations, for four cases, π_k can be described as:

$$\pi_k = \begin{cases} \sum_{j=1}^{k+1} \pi_j \Pr[A_n = k - j + 1], & \text{for } j > 0 \\ \Pr[Q_n = 0] \times \left(\sum_{l=1}^{k+1} \Pr[R_n = l] \Pr[A_n = k - l + 1] \right), & \text{for } j = 0,\ Q_n = 0 \\ (1 - \Pr[Q_n = 0]) \times \Pr[A_n = k] \times p, & \text{for } j = 0,\ Q_n > 0,\ p \\ \Pr[Q_n = 0] \times \left(\sum_{l=1}^{k+1} \Pr[R_n = l] \Pr[A_n = k - l + 1] \right) \times (1 - p), & \text{for } j = 0,\ Q_n > 0, 1 - p. \end{cases} \tag{8}$$

where $k = L_n + 1$, $j = L_n$. By combining the above four cases into a single equation, π_k is expressed as:

$$\pi_k = \pi_0 \left[p(1-q_0)a_k + (1-p+pq_0)\left(\sum_{l=1}^{k+1} r_l a_{k-l+1}\right)\right] + \sum_{j=1}^{k+1} \pi_j a_{k-j+1} \tag{9}$$

where $q_m = \Pr[Q_n = m]$, $a_k = \Pr[A_n = k]$, and $r_l = \Pr[R_n = l]$ for m and $k \geq 0$, and $l \geq 1$. Let the PGF of π_k be $\Pi(z) = \sum_{k=0}^{\infty} z^k \pi_k$, then it is represented as:

$$\Pi(z) = p(1-q_0)\pi_0 \sum_{k=0}^{\infty} z^k a_k + (1-p+pq_0)\pi_0 \sum_{l=1}^{\infty} z^{l-1} r_l \times \sum_{k'=0}^{\infty} z^{k'} a_{k'} + \left[\frac{\Pi(z)-\pi_0}{z}\right] \sum_{k'=0}^{\infty} z^{k'} a_{k'}. \tag{10}$$

Let the PGF of a_k be $\mathrm{A}(z) = \sum_{k=0}^{\infty} a_k z^k$. PGF is closely related to LST that can represents with respect to the time domain. Suppose that the LST of service time t_x distribution can be represented as $f_x^*(s)$, the packet arrival rate has a Poisson distribution with rate λ_a which is independent with t_x. Then, according to [17, 18], $A(z) = f_x^*(\mathrm{s}) = f_x^*(\lambda_a - \lambda_a z)$.

Similarly, let $\mathrm{R}(z) = \sum_{l=0}^{\infty} r_l z^l$ be the PGF of r_l distribution. Then, $\Pi(z)$ can be derived as:

$$\Pi(z) = \frac{\pi_0[1-p(1-q_0)z-(1-p+pq_0)R(z)]f_x^*(\lambda_a-\lambda_a z)}{f_x^*(\lambda_a-\lambda_a z)-z} \tag{11}$$

Since $\Pi(1) = 1$, we can derive π_0 from (11) as:

$$\pi_0 = \frac{1-\rho}{p(1-q_0)+(1-p+pq_0)E[R]} \tag{12}$$

where ρ be the λ_a/λ_x. Substituting (12) into (11), we have

$$\Pi(z) = \frac{(1-\rho)[1-p(1-q_0)z-(1-p+pq_0)R(z)]f_x^*(\lambda_a-\lambda_a z)}{[p(1-q_0)+(1-p+pq_0)E[R]][f_x^*(\lambda_a-\lambda_a z)-z]}. \tag{13}$$

According to [16], assume that the traffic arrival follows the Poisson distribution, the TIT threshold has the density function $f_I(t)$, mean $1/\lambda_I$, variance V_I, and the LST $f_I^*(\mathrm{s})$. Then,

$$q_0 = \int_{t=0}^{\infty} \left[\frac{e^{-\lambda_a t}(\lambda_a t)^0}{0!}\right] f_I(t)dt = f_I^*(\lambda_a) = e^{-\lambda_a/\lambda_I} \tag{14}$$

Meanwhile, suppose that the LST of the packet delay t_d is $f_d^*(s)$. Then, using the differentiation in the s-domain property, the expected packet delay $E[t_d]$ can be obtained since the differentiation of $\Pi(1-s/\lambda_a)$ by s corresponds to the time domain [19]. Thus,

$$\begin{aligned} E[t_d] &= -\frac{df_d^*(s)}{ds}\Big|_{s=0} \\ &= -\frac{d\Pi(1-s/\lambda_a)}{ds}\Big|_{s=0} \\ &= \frac{(1-p+pe^{-\lambda_a/\lambda_I})\lambda_a}{2\lambda_D[p(1-e^{-\lambda_a/\lambda_I})(1-e^{-\lambda_a/\lambda_D})\lambda_D+(1-p+pe^{-\lambda_a/\lambda_I})\lambda_a]} \\ &\quad + \frac{(1-V_x\lambda_x^2)\lambda_a - 2\lambda_x}{2(\lambda_a-\lambda_x)\lambda_x}. \end{aligned} \tag{15}$$

Here, λ_x, λ_a, λ_I, λ_D, and V_x represent the service rate ($1/t_x$), packet arrival rate, TIT expiration rate ($1/TIT$), sleep rate ($\frac{1}{LI\times[1-p]}$), and the variance of service rate ($1/\lambda_x^2$). p is basically determined by the current traffic load (e.g., channel utilization), and we assume that p follows exponential distribution with λ_a/λ_x since p is determined according to the traffic load. Figure 12 shows the variation of the expected delay with respect to λ_a. While the packet delay usually increases with increasing λ_a, this is not the case with a low traffic load as shown in the figure. Observe that the expected packet delay of SMAP is higher than the normal MAP, but lower than DozyAP with a low λ_a. While DozyAP can sleep for up to 500 ms when no traffic arrives for traffic inactivity threshold, SMAP stays in a low power state until the client's next LI, which is much less than the sleep duration of DozyAP. However, SMAP shows higher delay than DozyAP as λ_a increases. Again, this is because SMAP buffers packets and sends a burst by combining small idle periods caused by bandwidth discrepancy between Wifi and 3G. Of course, this trend does not continue when λ_a becomes much higher than the service rate λ_x.

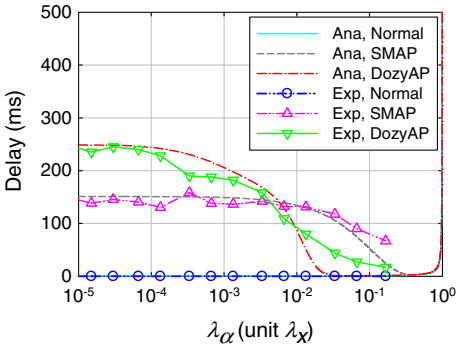


Fig. 12 Analysis and experimental delay results as a function of the packet arrival rate λ_a (*TIT threshold* 150 ms, *Sleep duration (DozyAP)* 500 ms [7], *Sleep duration (SMAP)* 300 ms)

5 Performance evaluation

5.1 Evaluation methodology

For experimental evaluation, we implemented SMAP on Samsung Galaxy Nexus smartphones that have a detachable battery, a Broadcom 802.11a/b/g/n network interface (version 5.90.195.61) [20], and a Wifi tethering function of Android 4.1.1 (kernel version 3.0.31).

As shown in Fig. 5, SMAP is implemented into the client and the MAP. The client's SMAP is implemented between the link and the IP layers. It blocks packets arriving from the upper layer (IP layer) using a packet buffer if it replied to a sleep request with a sleep response, and then it forwards all buffered packets to the lower layer (link layer) later when the sleep duration finishes. The MAP's SMAP, on the other hand, is directly modified from the Wifi driver and kernel OS to implement the power negotiation and duty cycle calculation algorithms. Once receipt of the sleep response, the MAP buffers the succeeding packets which arrive from the upper layer until the sleep duration finishes, so as not to send packets received from the cellular interface over Wifi while it sleeps.

However, there are two difficulties concerning the implementation of SMAP. Firstly, the power state transition of the MAP's Wifi interface is practically difficult since this part is implemented in the firmware, not the Wifi driver and the kernel OS. To solve this problem, we modified the firmware based on a reverse engineering method [21] that provides a fundamental solution for the modification of the Wifi firmware in Android smartphones. In analyzing the firmware, we found that most of functions in the firmware have a special wrapper which checks a special function table and jumps to the relevant code if necessary. Based this finding, we changed the Wifi driver and the firmware to make the wrapper point to new codes in the Wifi driver with the fixed memset, and implemented the power state transition of the Wifi interface. Secondly, to ensure periodic beacon transmission during the sleep duration, we implemented an additional timer in SMAP. When the MAP enters a sleep state, it checks whether or not the sleep duration is longer than the time remaining until the beginning of the next BI. If not longer, the MAP simply enters the sleep state without any additional operations, and it wakes up when the sleep duration finishes. Otherwise, the MAP starts the timer which will expire before the next beacon transmission, and then it wakes up when the timer expires so as to transmit a beacon. After sending the beacon, the MAP enters the sleep state again if the previous sleep duration is not finished.

In evaluation, five client phones were placed within 1 m from the MAP phone, and connected to Monsoon Power Monitor, an external real-time power monitoring device [8] for measuring the power consumed by smartphones. To isolate the power consumption of Wifi interface, we initialized all of the smartphones in a factory default setting with no application running in the background, disabled all other communication interfaces such as GPS, Bluetooth, and NFC, and turned down the backlight. In addition, we measured the packet delay, which is another important performance metric. We developed a simple Java program and installed it on both a client connected to a MAP and the corresponding server to measure end-to-end delay accurately. This Java program records a timestamp at both end for every packet to calculate the delay. To reduce time error introduced by different system clock, we ran the NTP daemon on the server and synchronized the clock of the client with the server.

For comparison, we measured the power consumption and the packet delay of a normal MAP, DozyAP, and SMAP using four different applications, web browsing, music and video streaming, and FTP. Among them, web browsing traffic is significantly affected by the behavior of users, and thus we used the real traces collected from real users [22] to get reliable results. The traces have the history of the Internet usage for 24 iPhone 3GS users, including users' unique ID, the timestamp, and a specific URL address that the users visited via a 3G network. Among them, we chose one trace of the most active user to represent a baseline for achievable performance in a conservative manner, and put it into our Java program to send and receive packets according to the same time and order as the trace. In music traffic, we used Google Play Music (a popular application sharing and playing the music on Android) and played a 262 s long music song. In video traffic, we executed Youtube application and watched a 153 s video clip. In FTP, we download a 22.6 MB data file using AndFTP application on Android.

Figure 13 shows the traffic patterns of four applications which is used in our experiments. To capture the traffic on a tethered client, we used tcpdump, a common packet analyzer. From web traffic in Fig. 13(a), sporadic spikes can be observed for the measurement period. The reason is that, once a user sends out a page request to a web server and downloads the corresponding data, then the user reads the page for some amount of time, and thus there are several idle periods between web page retrievals. On the other hand, music data is transmitted in one burst as shown in Fig. 13(b), since a music content is continuously transmitted until the whole-file is delivered. However, once the transfer completes before playback finishes, a long idle period will last because the user simply listens to the remaining time of play without further network activity. In video streaming, a long idle period can be seen between consecutive bursts from Fig. 13(c). Typically, a video streaming server would send a burst of data for the initial

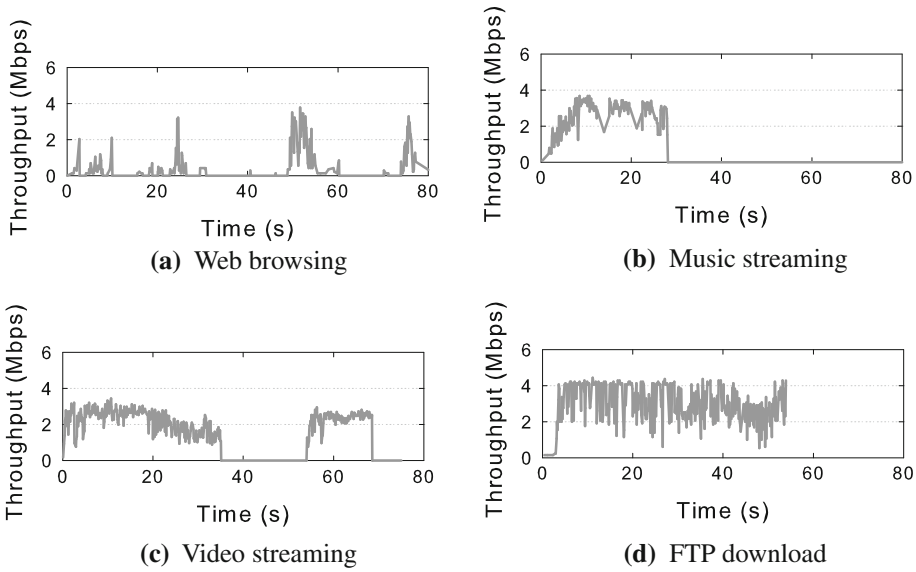


Fig. 13 Traffic patterns by throughput of four different applications

buffering, and stop it if the client's playback buffer becomes full at some point. Obviously, if the buffer lies within the underflow threshold, the server sends a data again until the buffer fullness is reached. In Fig. 13(d), FTP traffic shows a similar trend to music traffic. From the figure, one burst can be observed during the measurement period because the FTP traffic source will not stop the data transmission till the whole file is delivered. Thus, there is no a long idle period exists during the measurement period. Based on those traffic, each experiment is lasted until the application is finished (e.g., 180 s of web browsing trace, 262 and 153 s of music/video data, and 22.6 MB of data file).

5.2 Energy and delay of a MAP

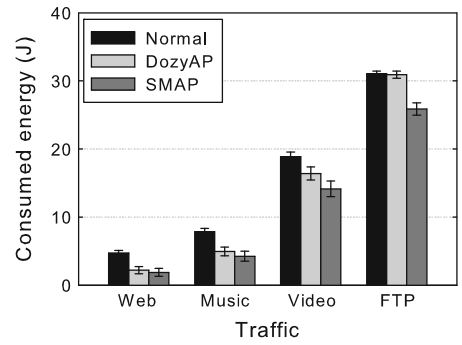
Figure 14(a) shows the amount of energy consumed by normal MAP, DozyAP, and SMAP. The results for the four applications show that SMAP can save 17–56 % energy compared to normal MAP. Furthermore, SMAP saves 6–16 % of energy consumed by DozyAP which employs an energy saving mechanism for a MAP. The main reason is that DozyAP can sleep only when the traffic does not exist for a certain period, whereas SMAP exploits not only idle periods due to no traffic activity, but also small idle periods between successive packets to sleep longer.

Unfortunately, SMAP does not outperform DozyAP substantially in web traffic, compared to the other applications. However, this is because the web traffic is typically the low volume of data. The main advantage of SMAP is to provide meaningful time periods enough to sleep by combining small idle periods between consecutive packet transfers. As shown in Fig. 13(a), web traffic contains only a few small idle periods whereas it has a lot of long idle periods due to the user's thinking time. For this reason, SMAP may not substantially achieve energy saving more than DozyAP.

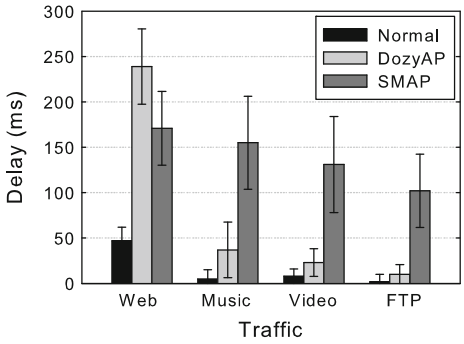
Figure 14(b) shows the packet delay performance. In the figure, SMAP tends to incur extra delay compared to the other schemes. The reason is that SMAP buffers packets and sends them with a burst to provide meaningful idle period for sleep. However, this extra delay is hardly perceivable by users because the user's QoS is typically determined by the arrival time of the last packet (that is, finish time) rather than the delay of each packet. Figure 14(c) shows the finish time of normal MAP, DozyAP, and SMAP for the four traffic types of applications. As shown in the figure, SMAP achieves very compatible finish time for all applications although it increases the average packet delay. From Fig. 14(a–c), SMAP saves energy more than the other schemes without substantial delay increase.

5.3 Energy consumption of a client

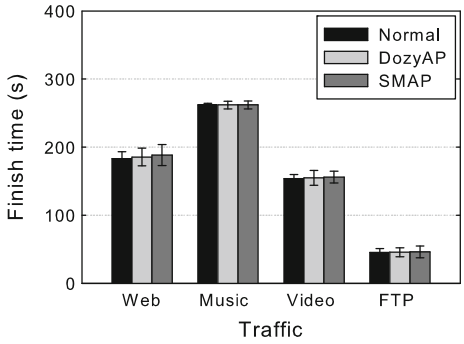
Figure 15 shows the energy consumption of a client for the four traffic patterns. Compared to normal MAP, SMAP reduces the energy consumption by up to 8.3 % while DozyAP increases the energy consumption by up to 3 %. This reason is that the client in DozyAP has to additionally receive a sleep request and reply a sleep response to explicitly show the agreement on the request. On the other hand, unlike DozyAP or normal MAP cannot enable the client to enter a sleep state under intensive traffic condition, SMAP provides a longer idle time enough to sleep by combing small idle periods between successive packets. This produces the result that SMAP client's energy is saved. To better understand this effect, we measured sequence numbers of FTP traffic, which is shown in



(a) Energy consumption



(b) Delay



(c) Finish time

Fig. 14 Energy consumption, delay, and finish time measurements for four applications

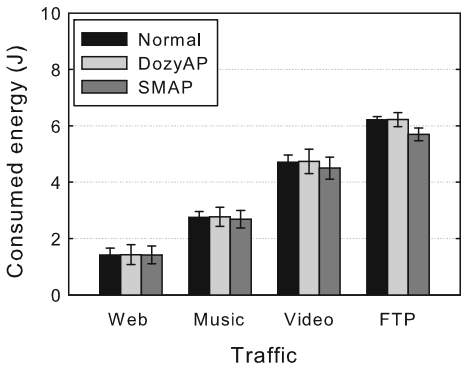


Fig. 15 Energy consumption of a client for four traffic patterns

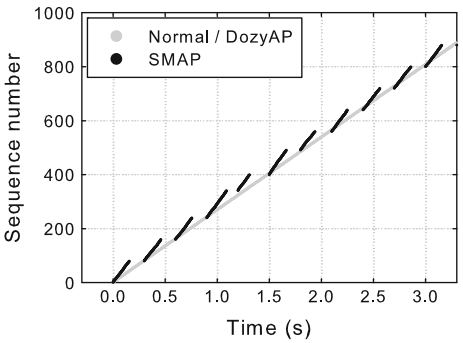
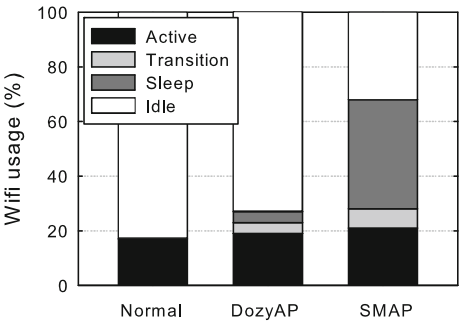
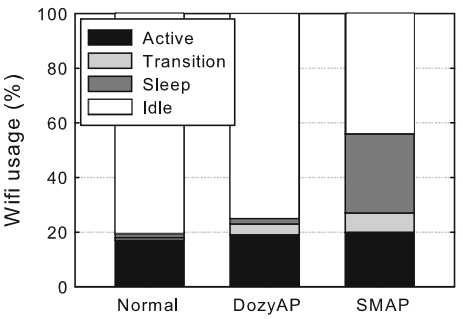


Fig. 16 Time-Sequence graph of FTP traffic downloading by a tethered client



(a) MAP



(b) Client

Fig. 17 Wifi usage of a MAP and a client under FTP traffic

Fig. 16. As shown in the figure, the ongoing packets of normal MAP and DozyAP are evenly distributed in the stream. In contrast, SMAP makes the traffic more burst while the overall throughput keeps unchanged. This burst in SMAP provides potentials to save energy by putting the Wifi interface into a low power state during idle periods. Thus, the client can sleep during the idle periods, wake up at the beginning of the next LI and receive packets, and thus the energy consumption on its Wifi interface can be reduced.

Figure 17 shows the proportion of time that Wifi interface spent at each state. To present an unbiased result, we captured 30 s in the middle of FTP download. As shown in

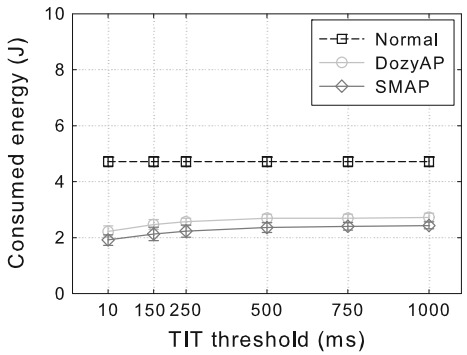


Fig. 18 Energy consumption of a MAP with varying the TIT threshold between 10 and 1,000 ms (Web traffic is used for this experiment)

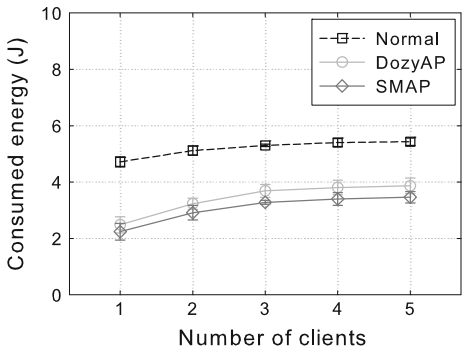


Fig. 19 Energy performance of the MAP with varying the number of clients (Web traffic is used for this experiment)

Fig. 17(a), normal MAP never sleeps and spends most of the time in idle. Although DozyAP employs a power saving mechanism, it seldom goes to the sleep state due to the frequent traffic activity. On the other hand, SMAP enables the Wifi interface to stay in the sleep state longer than others. We can see the similar trend at the client side. As shown in Fig. 17(b), the client in normal MAP or in DozyAP hardly transits to the sleep state because the subsequent packet always arrives before the tail time expires due to the bulk download of FTP traffic. However, SMAP provides longer idle periods enough for the tail time to expire, and thereby the client can put its Wifi interface into the sleep state more frequently.

5.4 Impact of TIT threshold

Figure 18 shows the energy consumption when the default TIT threshold is varied between 10 ms and 1,000 ms under Web traffic condition. In the figure, we can see that DozyAP and SMAP save the energy by 42.2–53.0 % and 48.3–59.4 % compared to normal MAP, respectively. There are a couple of trends to discuss from the results. First, regardless of the TIT threshold values, there is no change in energy consumed by normal MAP. This is obvious that normal MAP does not use the TIT threshold for a sleep mode, and thus it represents constant energy consumption while increasing the TIT threshold. On the other hand, both DozyAP and SMAP employ the TIT threshold to enter a sleep state when no traffic exists for the threshold, and thereby save significant energy. Second, as the TIT threshold increases, the amount of energy conserved by DozyAP and SMAP gradually decreases. The reason is that a higher TIT threshold leads to a more infrequent sleep transition since a MAP puts its Wifi interface into a low power state when there is no traffic activity for a certain interval longer than the TIT threshold. Of course, as discussed in Sect. 3.2, the MAP resets the minimum inter-frame gap when the previous sleep was successful so as to reduce the awake state until the subsequent TIT expiration. However, the default TIT threshold can affect only the first TIT expiration since the TIT threshold is reset to the default value (e.g., 150 ms) whenever any traffic arrives. For this reason, the variation depending on the TIT threshold is not so significant.

5.5 Impact of multiple clients

In Fig. 19, we measured the energy consumed when multiple clients are associated with a MAP. We vary the number of clients from 1 to 5 under Web traffic condition and measure the energy consumed at the MAP. As shown in the figure, DozyAP and SMAP that reduce the amount of time the MAP remains in a high power state consume less energy than normal MAP. Furthermore, by combining small idle periods between subsequent packets, SMAP is able to save more energy than DozyAP. As a result, SMAP saves 36.3–52.6 % and 9.9–11.0 % more energy than normal MAP and DozyAP, respectively.

In the figure, the energy consumption gradually increases as increasing the number of clients. The first reason is that multiple clients tend to increase the amount of traffic load, forcing the MAP to stay awake longer to deal with the increased traffic activity. The second reason is that the power negotiation becomes more difficult as the number of clients increases. As discussed in Sect. 3.1, the MAP has to receive agreements from all associated clients to enter a sleep state. However, it is more difficult to perform the sleep request-response process in a network environment where multiple clients exist because each of the clients may use a power save mechanism which hampers immediate power negotiation between a MAP and clients. Thus, the transition to the sleep mode will be delayed until all of the clients agree not to send any packet while the MAP

sleeps, and finally the amount of energy saving is decreased as shown in the figure.

6 Related work

Energy efficient Wifi operation is a very hot topic for more than a decade, and thus extensive research has been conducted to save power of Wifi devices. In previous studies, however, energy saving of the AP has not been considered important since it is typically assumed to be supported by constant AC power. However, with the increasing popularity of Wifi tethering, energy of an AP or a MAP with limited battery power also needs to be conserved. Considering the scenario in Fig. 1, a MAP constantly remains in a high power state regardless of the traffic activity, and thus it may deplete its battery power rapidly. Nevertheless, very little work for saving MAP's power has been reported in the literature. One study is DozyAP [7], in which the energy consumption of a MAP is reduced by putting its Wifi interface into a low power state when all its associated clients agree not to send any traffic while the MAP sleeps. This is facilitated in DozyAP by introducing "sleep request" and "sleep response" messages, and a new protocol on both AP and client devices. As discussed in Sect. 1, however, DozyAP is limited by energy saving since it can enter a sleep state only when no traffic exists at all for a certain period.

Unlike an AP or a MAP, on the other hand, extensive studies on client's power saving have been done. They can be classified into several approaches such as packet scheduling, MAC parameter adaptation, contention avoidance, and sleep scheduling. In packet scheduling approach, packets are scheduled in a bursty basis so that the sleep time of Wifi clients can be maximized [23–26]. In PSM-throttling [23], the authors consider a significant amount of energy wasted on idle channel listening. The streaming servers reshape the traffic into periodic bursts using bandwidth throttling, and the client can turn its interface on and off at the right time according to the burst and idle phases. [24] turns the network interface off and saves its energy while the level of the playback buffer is above a certain threshold and a streaming multimedia application consumes the backlog. In [25], the authors employ a web proxy at the AP performing a data transfer from a server by splitting TCP flows, and maximize the energy saving by regulating the relaying of the packets back to the client while minimizing the flow delay. In [26], the authors employ a pair of proxies, located on the client and server. To minimize the energy consumption, the mobile clients carry out automatic updates for dynamic contents browsing on its cache storing the responses to the previous requests, and reduce the number of wireless data transfers to and from clients. The above schemes are quite related to SMAP since it also use packet scheduling approach that reshapes the original traffic. However, the main difference is that SMAP considers the bandwidth discrepancy between Wifi and 3G links, an essential characteristic of a MAP, whereas the existing schemes exploit the RTT or buffer/cache storage to save power.

MAC parameter adaptation approach reduces unnecessary energy consumption of clients by adjusting their MAC parameters [27–29]. In [27], the authors address the problem that frequent periodic scanning process could incur high energy consumption, and propose to adaptively increase the scanning intervals depending on the amount of time the client remains in disconnected or idle state. C-PSM [28] selects optimal PSM parameters (e.g., BI and listening interval) based on traffic patterns and reduces the number of simultaneous clients' wakeups by employing a wake-up schedule to increase energy efficiency. STPM [29] adaptively determines its power management policy based on network traffic patterns, the costs (e.g., time and energy) required to transit between *constantly awake mode* (CAM) and PSM, and the expected costs after the transition.

Contention avoidance approach basically isolates traffic to reduce the energy consumption caused by contention between clients [15, 30, 31]. In [30], the authors address the impact of background traffic on the energy consumption of clients, reduces the energy consumption due to the contention by adopting the mechanism of time division and adjusting the power state of the clients. In NAPman [15], the authors address the issues of unnecessary retransmissions and unfairness due to competing between background and PSM traffic, and propose an energy-aware fair scheduling algorithm to minimize the client's energy and unnecessary retransmissions by distinguishing between traffic of a PSM client and that of CAM clients. Furthermore, in SleepWell [31], the authors consider that power consumption tends to increase due to high contention when a large number of PSM clients and APs share the channel. To solve this problem, the APs stagger wake up of PSM clients to reduce contention by spreading APs' beacons.

In sleep scheduling approach, client's sleep duration is dynamically adjusted based on traffic patterns, so as to extend the battery lifetime without substantial delay increase [32–35]. In BSD [32], the authors address the problem that mobile devices unnecessarily spend their energy to wake up periodically during long idle periods under typical Web browsing workloads. The PSM clients reduce the energy consumption by extending the period of low power state while guaranteeing $(1 + p)$ times longer than the measured RTT. In SPSM [33], the authors consider the tradeoff relationship between energy conservation and delay performance. To minimize the energy consumption while guaranteeing a desired delay performance

for each user, a client determines its actions (e.g., wake up or sleep) based on a penalty function. SiFi [34] reduces the energy consumption during the silence period of a VoIP call by predicting the length of future silence periods based on historical data. In [35], the authors propose micro power management (μPM), which predicts the arrival time of next incoming frame and outgoing frame, and enters power saving modes by exploiting short idle intervals between MAC frames.

The existing schemes in the above approaches basically use the traffic patterns or user preferences to reduce the adverse impact on delay. However, it is not easy to estimate the sleep duration accurately because those (e.g., traffic pattern and user preference) can vary considerably depending on the time. On the other hand, SMAP calculates the sleep duration by using the current traffic load and client's listen interval which is a fixed PSM parameter, and thus it can provide more reliable performance than existing schemes.

7 Conclusions

Power saving for Wifi devices is an important issue for battery-driven mobile devices as Wifi interface consumes a significant portion compared to other system components. To save energy of power-hungry mobile devices, PSM and its variants have been proposed, but most of them just focused on the client side. With increasing the popularity of Wifi tethering, it is imperative to have a new solution to save energy of both MAP and client. This paper presents a new tethering solution called *sleeping mobile AP* (SMAP) that increases the battery lifetime by exploiting the feature of asymmetric bandwidth of a MAP. Further, SMAP minimizes the delay increase by adaptively determining the duty cycle using the current traffic load as well as client's PSM parameters. The experimental measurements prove that SMAP can significantly increase the battery lifetime of a MAP and a client with small extra delay. Also, SMAP reduces the energy consumption of the client as well. Our future work is to increase the energy saving gain of smartphones with consideration of the characteristics of the cellular interface.

Acknowledgments This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2013R1A1A2065379).

References

1. Chetan Sharma Consulting. http://www.chetansharma.com/.
2. Yetim, O. B., & Martonosi, M. (2012). Adaptive usage of cellular and WiFi bandwidth: An optimal scheduling formulation. In *CHANTS*, 2012.
3. Rahmati, A., & Zhong, L. (2007). Context-for-wireless: Context-sensitive energy-efficient wireless data transfer. In *ACM MobiSys* 2007.
4. Apple Ios. https://developer.apple.com/technologies/ios/.
5. Windows Phone. www.windowsphone.com/.
6. Android Phone. http://www.android.com/.
7. Han, H., Liu, Y, Shen, G., Zhang, Y., & Li, Q. (2012). DozyAP: Power-efficient Wi-Fi tethering. In *ACM MobiSys*, 2012.
8. Monsoon Solutions Inc. http://www.msoon.com/LabEquipment/PowerMonitor/.
9. IEEE 802.11, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, Standard, IEEE, Aug. 1999.
10. Namboodiri, V., & Gao, L. (2010). Energy-efficient VoIP over wireless LANs. *IEEE TMC, 9*(4), 566–581.
11. Bonfiglio, D., Mellia, M. Meo, M., Rossi, D., & Tofanelli, P. (2007). Revealing Skype traffic: When randomness plays with you. In *ACM SIGCOMM*, 2007.
12. Wang, X., Chen, S., & Jajodia, S. (2005). Tracking anonymous peer-to-peer VoIP calls on the Internet. In *ACM CCS*, 2005.
13. Li, B., Ma, M., & Jin, Z. (2010). A VoIP traffic identification scheme based on host and flow behavior analysis. *Journal of Network and Systems Management, 19*(1), 111–129.
14. Bianchi, G. (2010). Performance analysis of the IEEE 802.11 distributed coordination function. *IEEE JSAC, 18*(3), 535–547.
15. Rozner, E., Navda, V., Ramjee, R., & Rayanchu, S. (2010). NAPman: Network-assisted power management for WiFi Devices. In *ACM MobySys*, 2010.
16. Yang, S.-R., & Lin, Y.-B. (2005). Modeling UMTS discontinuous reception mechanism. *IEEE TWC, 4*(1), 312–319.
17. Daigle, J. N. (1992). *Queueing theory for telecommunications*. Reading, MA: Addison-Wesley.
18. Takagi, H. (1991). *Queueing analysis: Vol. 1, vacation and priority systems*. North Holland, Amsterdam.
19. Heidemann, D. (1994). Queue length and delay distributions at traffic signals. *Transportation Research Part B*, *28*(5), 377–389.
20. Broadcom, BCM4330. http://www.broadcom.com/.
21. http://bcmon.blogspot.com/.
22. Shepard, C., Rahmati, A., Tossell, C., Zhong, L., & Kortum, P. (2010). LiveLab: measuring wireless networks and smartphone users in the field. In *ACM SIGMETRICS performance evaluation review*, December 2010.
23. Tan, E., Guo, L., Chen, S., & Zhang, X. (2007). PSM-throttling: Minimizing energy consumption for bulk data communications in WLANs. In *ICNP*, 2007.
24. Bertozzi, D., Benini, L., & Ricco, B. (2002). Power aware network interface management for streaming multimedia. In *IEEE WCNC*, 2002.
25. Ding, N., Pathak, A., Koutsonikolas, D., Shepard, C., Hu, Y. C., & Zhong, L. (2012). Realizing the full potential of PSM using proxying. In *IEEE Infocom*, 2012.
26. Armstrong, O. T, Amza, C., & deLara, E. (2006). Efficient and transparent dynamic content updates for mobile clients. In *ACM MobiSys*, 2006.
27. Gupta, A., & Mohapatra, P. (2007). Energy consumption and conservation in WiFi based phones: A measurement-based study. In *IEEE SECON*, 2007.
28. Xie, Y., Luo, X., & Chang, R. K. C. (2009). Centralized PSM: An AP-centric power saving mode for 802.11 infrastructure networks. In *SARNOFF*, 2009.
29. Edmund, M., Nightingale, E., & Flinn, J. (2003). Self-tuning wireless network power management. In *ACM MobiCom*, 2003.
30. Heand Y., & Yuan, R. (2009). A novel scheduled power saving mechanism for 802.11 Wireless LANs. In *IEEE TMC*, 2009.

31. Manweiler, J., & Choudhury, R. R. (2011). Avoiding the rush hours: WiFi energy management via traffic isolation. In *ACM MobySys*, 2011.
32. Krashinsky, R., & Balakrishnan, H. (2002). Miniminzing energy for wireless web access using bounded slowdown. In *ACM MobiCom*, 2002.
33. Qiao, D., & Shin, K. (2005). Smart power-saving mode for IEEE 802.11 wireless LANs. In *IEEE Infocom*, 2005.
34. Pyles, A. J., Ren, Z., Zhou, G., & Liu, X. (2011). SiFi: Exploiting VoIP silence for WiFi energy savings in smart phones. In *UbiComp*, 2011.
35. Liu J., & Zhong, L. (2008). Micro power management of active 802.11 interfaces. In *ACM MobiSys*, 2008.

Kyoung-Hak Jung received the B.S. degrees in Computer Engineering from Dankook University, Seoul, Korea, in 2007, and his Ph.D. degree in Computer Science and Engineering from Pohang University of Science and Technology (POSTECH), Pohang, Korea, in 2014. He is currently a research engineer in the Pohang Information Research Lab (PIRL) at POSTECH. His research interests include delay tolerant networks, software radio, power management, and wireless LAN MAC protocol.

Jae-Pil Jeong received the B.S. degrees in Computer Engineering from Kyoungpook National University, Daegu, Korea, in 2009. He is currently a Ph.D. student in the Department of Computer Science and Engineering at Pohang University of Science and Technology (POSTECH). His research interests include wireless LAN MAC protocol, wireless mesh networks, LTE network, and video streaming over wireless networks.

Young-Joo Suh received his B.S. and M.S. degrees in Electronics Engineering from Hanyang University, Seoul, Korea, in 1985 and 1987 respectively, and his Ph.D. degree in Electrical and Computer Engineering from Georgia Institute of Technology, Atlanta, Georgia, in 1996. He is currently a professor in the Department of Computer Science and Engineering at the Pohang University of Science and Technology (POSTECH), Pohang, Korea. From 1988 to 1990, he was a research engineer at the Central Research Center of LG Electronics Inc, Seoul, Korea. From 1990 to 1993, he was an assistant professor in the Department of Computer Science and Engineering at Chung-Cheong College, Korea. After receiving Ph.D. he worked as a postdoctoral researcher in the Computer Systems Research Laboratory in the School of Electrical and Computer Engineering at the Georgia Institute of Technology from 1996 to 1997. From 1997 to 1998, he was a research fellow of the Real-Time Computing Laboratory in the Department of Electrical Engineering and Computer Science at the University of Michigan, Ann Arbor, Michigan. His current research interests include wireless LAN MAC protocol, mobility management, ad hoc networks, 4G Wireless Mobile Networks, etc. Dr Suh is a member of the IEEE and the IEEE Communications Society.