OPSM - Opportunistic Power Save Mode for Infrastructure IEEE 802.11 WLAN

Pranav Agrawal*, Anurag Kumar†, Joy Kuri*, Manoj K. Panda†, Vishnu Navda‡, Ramachandran Ramjee‡
*Centre for Electronics Design and Technology
†Electrical Communication Engineering
Indian Institute of Science, Bangalore, India - 560 012
‡Microsoft Research India, Bangalore, India

***Abstract*—We focus on the energy spent in radio communication by the stations (STAs) in an IEEE 802.11 infrastructure WLAN. All the STAs are engaged in web browsing, which is characterized by a short file downloads over TCP, with short duration of inactivity or *think time* in between two file downloads. Under this traffic, Static PSM (SPSM) performs better than CAM, since the STAs in SPSM can switch to low power state (sleep) during think times while in CAM they have to be in the active state all the time. In spite of this gain, performance of SPSM degrades due to congestion, as the number of STAs associated with the access point (AP) increases. To address this problem, we propose an algorithm, which we call *opportunistic PSM (OPSM)*. We show through simulations that OPSM performs better than SPSM under the aforementioned TCP traffic. The performance gain achieved by OPSM over SPSM increases as the mean file size requested by the STAs or the number of STAs associated with the AP increases. We implemented OPSM in NS-2.33, and to compare the performance of OPSM and SPSM, we evaluate the number of file downloads that can be completed with a given battery capacity and the average time taken to download a file.**

I. Introduction

With the advent of wireless technology in hand-held devices, saving energy incurred due to wireless protocols is of prime importance. Users often use these devices for web browsing, so it is imperative to evaluate and improve the performance of various modes of operation that are supported by IEEE 802.11 for web traffic.

In the normal mode of operation, also called the Continuously Active Mode (CAM), an STA always keeps its radio on, so it can receive and transmit at any time. This mode of operation is energy inefficient since the STAs draw current even when they are idling. To save power during the period when there is less or no network activity, WiFi cards are provided with controls through which they can be turned off. To leverage this facility, the IEEE 802.11 standard has a feature using which STAs can turn off their radio without losing packets. This is called the Power Save Mode (PSM). In this mode, an STA can be in any one of the two states, *active state* and *sleep state*.

When an STA switches to the Power Save Mode (PSM), it goes into sleep state (switches off its radio) and also informs the AP about it. Any packets arriving at the AP for STAs in PSM are stored in separate queues (PSM queues), maintained for each STA. If the AP needs to send any packet from any of these queues, it enqueues it at the tail of its transmission queue, which we the AP's NIC queue.

There is a beacon frame, sent by the AP after every fixed time interval. This time interval between two consecutive transmission of beacon frames is called the beacon interval. In the beacon frame there is a field called traffic indication map (TIM), using this the AP informs the PSM STAs about the stored packets. PSM STAs wake up periodically to listen to beacon frames. If they find that there are packets stored for them then they switch to active state and send a PS-POLL frame. In reply to PS-POLL, the AP immediately sends a MAC ACK. On receiving the PS-POLL from a STA, the AP dequeues the HOL packet from the corresponding PSM queue and enqueues it at the tail of the NIC queue. If the PSM queue of the STA is still non-empty, then the AP sets the More Bit in the dequeued packet to indicate to the STA that there are more packets stored for it at the AP. On receiving this packet, the STA checks the More bit. If it is set then it sends another PS-POLL frame. In this way, the STA does not sleep until its PSM queue at the AP becomes empty.

Note that, earlier studies [1]–[4] consider that on receiving a PS-POLL the AP immediately sends the data packet. It is not practical to assume that the AP can immediately send the data packet in response to the PS-POLL since there might be already some packets present in the NIC queue of the AP at the time when the AP receives the PS-POLL frame. The above described implementation is closely analyzed for different TCP traffic models in our earlier work [5].

In this paper we consider a general scenario in which some STAs are associated to a single Access Point (AP) and are engaged in web browsing. Web browsing is characterized by users requesting short file transfers, and in between two requests there is a short period of inactivity or *think time*. In our earlier work [5], we compared the performance of CAM and PSM. We showed using analysis and simulation that, PSM performs better than CAM in the scenario in which all the STAs are downloading short files with think times. This happens because, when the STAs are in PSM they can go to sleep during the inactivity periods, which saves energy, while in CAM the STAs stay in active state all the time. Although PSM performs better than CAM, its performance still degrades with increasing number of STAs. When the number of STAs increases, throughput share of an individual STA decreases

978-1-4244-6826-3/10/$26.00 ©2010 IEEE

which increases the download time and hence the energy expenditure. In this paper we propose an improvement on the static power save mode of IEEE 802.11, which we call an opportunistic power save mode (OPSM). We will show through simulations that OPSM is more energy efficient than SPSM. OPSM gives better performance than PSM, only when the round trip propagation delay (RTPD) from the AP to TCP server is very small and the file size requested by the STAs are large enough to cause congestion at the AP.

This paper is organized as follows: Section II discusses the previous literature. Section III gives the description of OPSM. In Section IV, we present the simulation results. Finally, Section V concludes the paper.

II. Related Work

There have been many schemes proposed in the literature to improve the performance of PSM. Generally, the focus of these studies is on either scheduling of packets or the waking up the STAs in PSM such that it minimizes energy and delay.

In Yong et al. [6], time is sliced into slots and the AP schedules the transmission of the packets and informs this schedule to the STAs through beacon frames. The STAs wake up at the specified slot as indicated in the schedule, receive their packets and go to sleep again. Lee et al. [7] proposes heuristic scheduling policies of packets at the AP, to minimize energy and the average delay.

Perez-Costa and Camps-Mur [8] estimates the interarrival time of packets at the AP and sends the PS-POLL to retrieve the packet based on the estimate.

Gan and Lin [9] observes the contention among PS-POLLs just immediately after the beacon interval. It improves the performance of the system through scheduling the wake-up of the PSM STAs.

In all the above works, the AP does *packet level* scheduling. Since we consider the STAs to be downloading files, so here we are concerned with scheduling files instead of packets. Further, in our work, the AP does not play a role in scheduling, but the STAs themselves cooperate to schedule their file downloads, which makes our work different. However, in our work the AP need to send one extra bit in addition to the TIM in the beacon frame, which indicates whether its NIC queue is empty or not.

Nath et al. [10] minimize the energy by transmitting multiple beacons, one for every STA associated to the AP. Each STA estimates the RTT of the current TCP connection and sends this information to the AP, based on this information the AP schedules the beacon frames to the STAs.

Krashinsky and Balakrishnan [11] consider a single STA in PSM doing short file transfers. They observed that web transfers incur large delays, because of the interaction between TCP slow start, RTT and PSM. To bound the delay, the authors propose a bounded slow down (BSD) protocol in which a web page can experience a delay not more than a specified percentage (p) of the actual normal delay (without PSM). BSD [11] is further improved upon by Quiao and Shin [12]. They estimates the RTT of current TCP connection and using this information, the sleep wake schedule is made more efficient.

Anastasi et al. [1] consider a single STA in PSM downloading a file over TCP in the presence of N active STAs and evaluated the expected energy spent by the STA to download the file as a function of N.

Lei and Nilsson [2], Baek and Choi [3] and Si et al. [4] analyzed the performance of PSM for different traffic models.

Tan et al. [13] propose to take advantage of throttling done by the TCP server in media streaming applications.

III. OPSM - Description

Wireless stations often engage in web browsing, which are characterized by short file downloads with short durations of think times in between two downloads. If the number of such STAs is associated with a single AP increases, then an STA takes longer to download a file, which increases the energy expenditure. This is because the AP does not give preference to any STA: so with an increasing number of STAs downloading simultaneously, the throughput share of an individual STA decreases.

Now consider an STA with slightly modified behavior. Before starting a file download, if it finds that the AP is serving any other STA (assuming that somehow it gets this knowledge), then it goes to a low power state (sleep state), and starts its download after the AP finishes serving the other STA. If all the STAs associated with the AP follow this strategy, then it will result in only one STA downloading a file at any instant. Because of this, throughput is not shared among STAs during a file download, and this reduces the download time of a file and in turn reduces the energy. We call this strategy as *Opportunistic PSM* because the STA waits for the opportunity to download its file when it has to spend least energy, which is possible only when the AP is not serving any other STA.

Yet there is a problem with the above setting: Consider that there are a large number of STAs browsing the web and they are associated with a single AP. Now consider a situation, a file transfer is going on, and during this file transfer, several STAs complete their think times. According to the modified behavior, the STAs initiate their file downloads immediately after the completion of the ongoing file transfer. This rules out the possibility of saving energy, since that requires only one STA downloading a file at a time. To tackle this situation, STAs should wait for random durations of time instead of immediately initiating their file download after the completion of an ongoing transfer. The STA which samples the least waiting time will start its download and the other STAs which sampled larger waiting times will find the AP serving an STA, hence they will go to sleep state again. So, we can expect that with high probability, only one STA gets service at any instant.

Now the question is how the STAs can get to know whether any other STA is downloading or not. There are many ways to find this out: One way is by sniffing the channel for a duration; if there is no activity on the channel, then it can be concluded that the AP is not serving any STA. Nevertheless, sniffing the channel wastes some energy. Another efficient method which

we consider here is to add a bit to the beacon header, indicating whether the NIC queue of the AP is empty or not. Since the beacon frames are transmitted periodically by the AP, adding 1 bit to it will not produce much extra overhead. Through this bit, STAs can get the information whether the AP is currently serving any STA or not. We will call this bit as the *NIC Queue Empty (NIE)* bit. To make sure that the AP is not serving any other STA, the AP sets the NIE bit to 0, only if it did not transmit any packet to any STA in the last beacon interval.

Note that the scheme described above works only for small values of round trip propagation delay (RTPD) between the file server and the AP. Since OPSM requires that the AP should indicate in every beacon frame it transmits during a file transfer that an STA is downloading a file, so the AP should transmit at least one packet in every beacon interval during a file transfer, this is true only for smaller values of RTPD with high probability. If the RTPD becomes higher, it is possible that all the packets for the STA might be in flight and the AP does not transmit packet in a beacon interval; as a result the AP will indicate to other STAs that it is not serving any STA. So other STAs may also start their file downloads even when a file transfer is going on, which results in higher energy consumption. Another reason why OPSM does not work for higher RTPD is that the STA has to remain idle when the AP does not have packet to send since the packets might be in flight due to positive RTPD, while in SPSM the STA goes to sleep state during this duration. This results in OPSM performing poorly for higher RTPDs.

In OPSM, the STA can be either in CAM or PSM, and the STA switches from one mode to another. In CAM, the STA stays in the active state always while in PSM, the STA can be in active or in sleep state. If a STA switches from PSM to CAM or vice versa, it informs the AP about it. When the AP gets the information that the STA has switched from PSM to CAM, it transfers all packets queued in the corresponding PSM queue of the STA into its transmission queue (NIC queue), and from here onward, it enques all the packets destined for the STA directly into its NIC queue. When the STA switches from CAM to PSM, the AP starts enqueuing all packets destined to the PSM STA in the corresponding PSM queue.

The next section presents the detailed implementation of the OPSM.

A. Implementation Details of OPSM in NS-2

There are two modes of operations PSM and CAM. Under PSM, the STA can be in one of the following two states:

- SLEEP PSM (S_{PSM}) – In this state, the STA is in sleep state.
- ACTIVE PSM (A_{PSM}) – In this state, the STA is in the active state and expects a beacon frame from the AP.

Under CAM, the STA can be in the following two states:

- TIMER ON (T_{CAM}) – In this state, the STA is in active state and is counting down a timer; if the STA remains idle (neither receives nor transmits a packet) before its expiry, then it switches its mode from CAM to PSM.

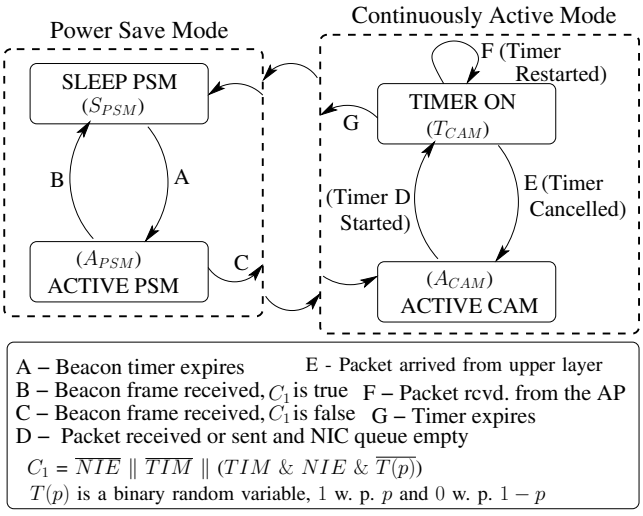


Fig. 1: State Machine

- ACTIVE CAM (A_{CAM}) – In this state, the STA is in active mode and it stays in this state while it is contending to transmit any packet. Note that, this state is different from T_{CAM}, as in this state the STA is not counting down the timer to switch to PSM.

The following points describe the behavior of the STA from any point of time when the STA is in state SLEEP PSM (S_{PSM}) (see Fig. 1).

- When the STA is in state SLEEP PSM, a timer periodically brings the STA to state ACTIVE PSM ($S_{PSM} \rightarrow A_{PSM}$), so that the STA can hear the beacon frame transmitted by the AP.
- When the STA is in state ACTIVE PSM (A_{PSM}), it expects a beacon frame from the AP. After receiving a beacon, the STA can go to either state S_{PSM} or A_{CAM}, depending on the contents of the beacon frame. Following points describes the possible cases:
  - There is no packet at the AP (TIM bit unset). This means that the STA has not requested any file, so it switches to sleep state ($A_{PSM} \rightarrow S_{PSM}$).
  - Some other STA is downloading a file, which is indicated by NIE bit being unset. In this case, irrespective of TIM bit set or unset, the STA goes back to sleep state again ($A_{PSM} \rightarrow S_{PSM}$).
  - When both TIM and NIE bits are set, then it means that there are packets at the AP for this STA and the AP is not serving to any other STA. The STA can start downloading, but as described earlier there might be many STAs which are waiting for the AP to become idle. To prevent them from starting downloads together, the STA samples a binary random variable $T(p)$; If 0 is sampled, then the STA goes to sleep state ($A_{PSM} \rightarrow S_{PSM}$). If it samples 1, then the STA sends a PS-POLL frame to the STA with power management bit unset and it switches its mode from PSM to CAM ($A_{PSM} \rightarrow A_{CAM}$). Through the power management bit, the AP gets to know that the

STA has switched to CAM, so it dequeues all the packets from its PSM queue and enques them at the tail of its NIC queue. $T(p)$ is the random variable which is 1 with probability p and 0 with probability $1-p$.

Define $C_1 := \overline{TIM} \;||\; \overline{NIE} \;||\; (TIM \;\&\; NIE \;\&\; \overline{T(p)})$. The points above can be summarized as follows

$$\begin{aligned} S_{PSM} &\rightarrow A_{PSM} \quad \text{Beacon timer expires} \\ A_{PSM} &\rightarrow S_{PSM} \quad C_1 \text{ is true} \\ A_{PSM} &\rightarrow A_{CAM} \quad C_1 \text{ is false} \end{aligned} \tag{1}$$

The following points describe the behavior of the STA in CAM:

- When the STA is in state ACTIVE CAM (A_{CAM}) (see Fig. 1), for every reception of packet from the AP and on every packet sent by it, the STA checks if its NIC queue is empty. If it is, then it starts a timer with time out value (T_{C-P}) and switches to state T_{CAM}.
- When the STA is in state T_{CAM}, and it receives any packet from the AP, then it restarts the timer and remains in the same state, since the STA has to be remain idle for T_{C-P} duration before switching to PSM.
- When the STA is in state T_{CAM}, and it receives any packet from the upper layer then it cancels the timer and switches to state A_{CAM}. This is because the STA cannot remain idle, as it has to contend for a packet.
- When the STA is in state T_{CAM} and the timer expires, this means that the STA did not receive a packet from the upper layer, and neither did it receive a packet from the AP within duration T_{C-P}, due to which timer expires. The STA now switches its mode from CAM to PSM and enters in the state (S_{PSM}). While switching to PSM, it sends a PS-POLL frame with power management bit set, indicating that, it has switched to PSM; So from here onward, the AP starts enqueuing packets for it in the corresponding PSM queue.

If any packet comes from the upper layer while the STA is in sleep state of PSM, then the STA immediately switches to active state and starts contending for the packet. After sending the packet, it goes to sleep state again. If in case the beacon frame arrives in while the STA is contending for a packet then the STA switches to CAM (A_{CAM}) if the condition C_1 is false. However if the C_1 is true then the STA does not go sleep state again, till it sends the packet it is contending for.

IV. Simulation Results

We compare the performace of OPSM with that of the more realistic implementation of SPSM, considered in our earlier work [5].

We have implemented OPSM in the NS-2.33 simulator. In the simulation, we consider a constant number of users downloading short files over TCP with *think times*. We assume that all the files downloaded by the STA utilize a single TCP connection, which means that the TCP connection is established for the first file while for rest of the files, the same connection is reused. For every file, an HTTP request packet is sent by STAs to initiate the transmission [14]. To generate HTTP traffic in NS-2, we used PACKMIME [15]. Details about the simulation setup are given below:

- Round trip propagation delay (RTPD) from the AP to server is taken as $0\,ms$ in one set of simulations and $20\,ms$ in another.
- At any instant, an STA has at most a single TCP connection.
- The receivers do not implement the *delayed TCP ACKs*.
- There is no loss of packets due to buffer overflow at the AP or STAs.
- There are no packet losses due to bit errors on the wireless medium.

The file size is taken to be exponentially distributed with mean $400KB$ and $1000KB$, and the think time is taken to be exponentially distributed with mean $5s$. The beacon interval is taken as $100ms$. As the beacon frame is sent by contention so an STA has to be awake for a duration to be able to listen to it. This duration is taken as $5ms$. So, after an STA receives the beacon frame, it starts a timer, which wakes it up after the time out value of $95ms$. Time out value of timer for the STA to move from CAM to PSM (T_{C-P}) is taken as $100ms$. $T(p)$ is 1 with probability .1 and 0 with probability .9. Generally the radio chip draws different values of current while it is transmitting, idling, receiving and sleeping (low power state). These values of current are taken from the specifications of the Intel PRO/Wireless 2011 card [16] and they are: transmitting current = $300mA$, Idle current = $170mA$, Receive current = $170mA$, Sleep current = $10mA$.

We use the following two metrics to compare OPSM with SPSM:

- Average number of file downloads for a given battery capcity – Here, the battery capacity is taken to be the maximum charge that can drawn from it. So the number of files that can be completed in a given battery capacity is obtained by dividing the battery capacity (100 Coulombs) by the average charge drawn in downloading a file. Average charge drawn per file is obtained by dividing the total charge drawn by all the STAs in a given time interval by the total number of files downloaded by all of them in the same time interval.
- Average sojourn time – This is defined as the total time taken by all the files downloaded in a given interval divided by the total number of files downloaded in the the same interval. Here, the time taken to download a files is taken as the time difference between the instant the STA starts contending to send the HTTP request packet and the instant it receives the last packet of the file.

Fig. 2 shows the comparison of the average number of files that can be downloaded in a given battery capacity (100 Coulombs). We can see that STAs in OPSM can download more files than STAs in SPSM. Fig. 4 shows the percentage increase of the number of file downloads in OPSM over SPSM in a given battery capacity (100 Coulombs). We can see

that with increase in the number of STAs or increase in the file size, the gain achieved by OPSM increases and it can go to up to more than 200%.

But for RTPD of $20ms$ and small number STAs, OPSM fails to perform better. This can be seen in the curve for $400KB$ file size in Fig. 2b. This happens because, in the case of SPSM, the STA goes to sleep when the packets are in flight, while in OPSM, the STA stays in CAM, thus it remains in the active state even when the AP does not have packets to send. Nevertheless for higher number of STAs OPSM performs better than SPSM.

Another characteristic that need to be noted here is that the performance of OPSM degrades with increasing number of STAs. This is because the STA has to wake up periodically to hear every beacon frame, and this corresponds to a fixed charge that is drawn by all the STAs, irrespective of the number of STAs associated with the AP. But when the number of STAs increases, the number of files a STA can download in a given time interval decreases. Therefore the charge drawn per file due to just listening to the beacon frames increases. Also, the STA spends more time in sleep state with increasing number of STAs. This increases the charge drawn in sleep state per file download. Although the STA draws a small current (10 mA), but the duration it stays in sleep state is significant so is the charge drawn during sleep state. Another reason for the degrading performance is the retransmissions of the packets by the TCP sender due to TCP timeouts while the packets are waiting at the AP. This increases with increasing number of STAs, since an STA has to wait more before it can download its packets.

Ideally there should not be any difference in the sojourn times in PSM and OPSM because in PSM, the system behaves like a processor sharing queue, while in OPSM it behaves like a server serving the customers one by one in random order. Using the Little's Theorem, it can be shown that the expected sojourn time of a customer does not change with the order of service of customers.

In comparison to SPSM, OPSM does not have the overheads of PS-POLLs hence the throughput achieved by OPSM is more than that of the SPSM. So we expect the sojourn time in OPSM to be lower than that in SPSM. But, to implement the random order of service, some time is wasted in idling. A STA can start its download only at the beginning of the beacon interval with probability p, so it is with some positive probability, none of the STAs begin their download. Thus some beacon interval are wasted and this increases the sojourn time. For RTPD of $0ms$ and mean file size of $1000KB$ (see Fig. 3a), we can observe that the sojourn time of the STAs is lower than that of the SPSM irrespective of the number of STAs; This is because the time which is wasted in attempting to download the file is dominated by the decrease in download time due the higher throughput. For RTPD of $0ms$ and mean file size of $400KB$, the sojourn times files are almost equal.

Fig. 3b shows that sojourn times for RTPD of $20ms$. It shows that OPSM has slightly higher average sojourn time than SPSM. Nevertheless, the increase is not significant, keeping in mind the energy saving achieved.

V. Conclusion

We have shown that our proposed scheme leads to considerable energy savings and it increases with increasing congestion at the AP. The reason behind the improved performance is that OPSM only permits one download at any time, due to which an STA gets the maximum throughput and this results in least energy consumption. In SPSM, this is not the case, since it allows simultaneous downloads, which leads to longer file download times and hence consumes more energy than OPSM. Our future work is to design a generic policy for the STAs that works better for higher RTPD also.

References

[1] G. Anastasi, M. Conti, E. Gregori, and A. Passarella, "Saving energy in wi-fi hotspots through 802.11 psm: An analytical model," in *2nd Workshop on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt'04)*, March, 2004, pp. 227–236.

[2] H. Lei and A. A. Nilsson, "An m/g/1 queue with bulk service model for power management in wireless lans," in *PE-WASUN '05: Proceedings of the 2nd ACM international workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*. New York, NY, USA: ACM, 2005, pp. 92–98.

[3] S. Baek and B. D. Choi, "Performance analysis of power save mode in ieee 802.11 infrastructure wireless local area network," *Journal of Industrial and Management Optimization (JIMO)*, vol. 5, no. 3, pp. 481–492, August, 2009.

[4] P. Si, H. Ji, F. Yu, and G. Yue, "Ieee 802.11 dcf psm model and a novel downlink access scheme," in *Wireless Communications and Networking Conference, 2008. WCNC 2008. IEEE*, 31 2008-April 3 2008, pp. 1397–1401.

[5] P. Agrawal, A. Kumar, J. Kuri, M. K. Panda, V. Navda, R. Ramjee, and V. N. Padmanabhan, "Analytical models for energy consumption in infrastructure wlan stas carrying tcp traffic," in *Accepted in Communication Systems and Networks and Workshops, 2010. COMSNETS 2010. Second International*. [Online]. Available: http://arxiv.org/abs/0909.3717v4

[6] Y. He, R. Yuan, X. Ma, J. Li, and C. Wang, "Scheduled psm for minimizing energy in wireless lans," in *Network Protocols, 2007. ICNP 2007. IEEE International Conference on*, Oct. 2007, pp. 154–163.

[7] J. Lee, C. Rosenberg, and E. Chong, "Energy efficient schedulers in wireless networks: design and optimization," *Mobile Networks and Applications*, vol. 11, no. 3, pp. 377–389, 2006.

[8] X. Perez-Costa and D. Camps-Mur, "Apsm: bounding the downlink delay for 802.11 power save mode," in *Communications, 2005. ICC 2005. 2005 IEEE International Conference on*, vol. 5, May 2005, pp. 3616–3622 Vol. 5.

[9] C.-H. Gan and Y.-B. Lin, "An effective power conservation scheme for ieee 802.11 wireless networks," *Vehicular Technology, IEEE Transactions on*, vol. 58, no. 4, pp. 1920–1929, May 2009.

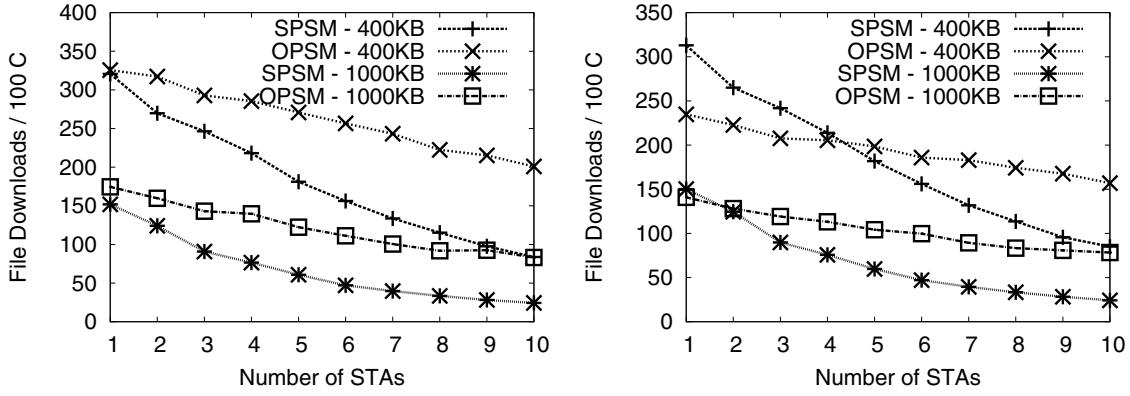
[10] S. Nath, Z. Anderson, and S. Seshan, "Choosing beacon periods to improve response times for wireless http clients," in *MobiWac '04: Proceedings of the second international workshop on Mobility management & wireless access protocols*. New York, NY, USA: ACM, 2004, pp. 43–50.

[11] R. Krashinsky and H. Balakrishnan, "Minimizing energy for wireless web access with bounded slowdown," *Wirel. Netw.*, vol. 11, no. 1-2, pp. 135–148, 2005.

[12] D. Qiao and K. Shin, "Smart power-saving mode for ieee 802.11 wireless lans," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 3, March 2005, pp. 1573–1583 vol. 3.

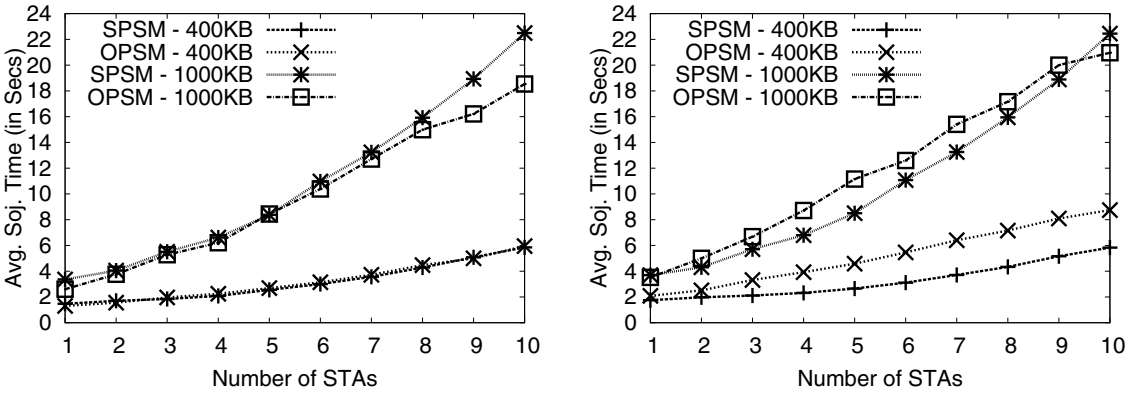
[13] E. Tan, L. Guo, S. Chen, and X. Zhang, "Psm-throttling: Minimizing energy consumption for bulk data communications in wlans," in *Network Protocols, 2007. ICNP 2007. IEEE International Conference on*, Oct. 2007, pp. 123–132.

[14] "Hypertext transfer protocol – http/1.1," 1999. [Online]. Available: http://tools.ietf.org/html/rfc2616#section-5.1.1



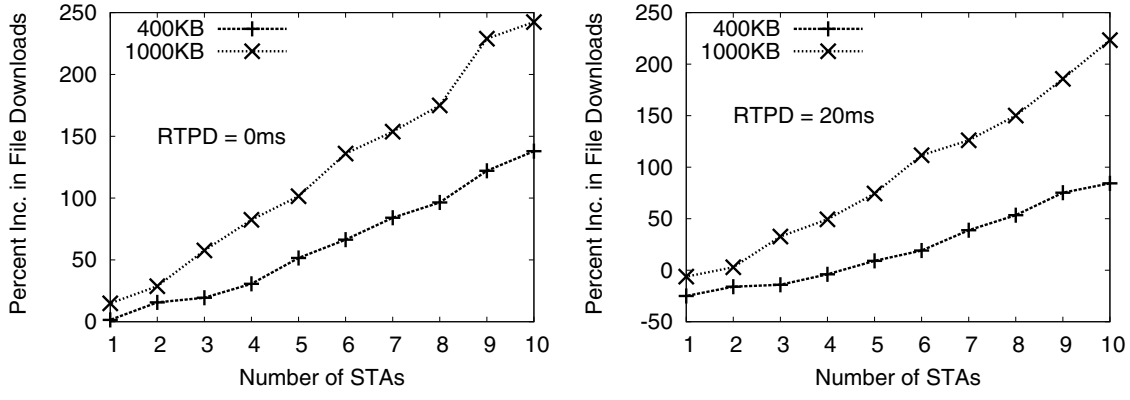
(a) RTPD = 0ms (b) RTPD = 20ms

Fig. 2: Number of Transactions per 100 Coulomb (Mean Think Time = 5s)



(a) RTPD = 0ms (b) RTPD = 20ms

Fig. 3: Average Sojourn Time (in Seconds) (Mean Think Time = 5s)



(a) RTPD = 0 ms (b) RTPD = 20 ms

Fig. 4: Percentage Increase in the Number of File Downloads in a given battery capacity (100 Coulomb) with Mean Think Time = 5s

[15] "Packmime-http: Web traffic generation." [Online]. Available: http://www.isi.edu/nsnam/ns/doc/node552.html

[16] "Intel pro/wireless 2011 lan pc card," Intel. [Online]. Available: http://download.intel.com/support/wireless/wlan/pro2011/wireless.pdf