Analytical Models for Energy Consumption in Infrastructure WLAN STAs Carrying TCP Traffic

Pranav Agrawal*, Anurag Kumar†, Joy Kuri*, Manoj K. Panda†, Vishnu Navda‡ and 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 develop analytical models for estimating the energy spent by stations (STAs) in infrastructure WLANs when performing TCP controlled file downloads. We focus on the energy spent in radio communication when the STAs are in the Continuously Active Mode (CAM), or in the static Power Save Mode (PSM). Our approach is to develop accurate models for obtaining the fraction of times the STA radios spend in idling, receiving and transmitting. We discuss two traffic models for each mode of operation: (i) each STA performs one large file download, and (ii) the STAs perform short file transfers. We evaluate the rate of STA energy expenditure with long file downloads, and show that static PSM is worse than just using CAM. For short file downloads we compute the number of file downloads that can be completed with given battery capacity, and show that PSM performs better than CAM for this case. We provide a validation of our analytical models using the NS-2 simulator.**

I. Introduction

With the advent of wireless technology in hand-held devices, saving energy incurred due to wireless protocols is of prime importance. To design efficient power management policies for wireless adapters, it is often required to know the energy spent by wireless stations (STAs) running different classes of applications. Since users often use TCP based applications, in this paper we characterize the energy spent by STAs while running TCP controlled data transfers. We focus only on the energy spent in radio communication, and we evaluate this by obtaining the fraction of times the STA radio stays in different states, i.e., idle, transmit and receive.

In normal mode of operation, also called as 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 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 generally the Power Save Mode (PSM). In this mode, an STA can be in any one of the two states: *active state* and *sleep state*. *Contribution:* In this paper we analyze various scenarios in which several stations (STAs) are associated with a single Access Point (AP). In each scenario each STA is considered to be either in the PSM or the CAM and downloading files via the AP. The file server is considered to be located on the high speed Ethernet link connected to the AP, which makes the propagation delay between the AP and server negligible. Our aim is to model the energy spent for radio communication by the STAs for the following two types of TCP traffic in either mode of operation of the STA:

1) N STAs downloading long files over TCP – In this scenario, the average rate of energy expenditure is analyzed. To evaluate this, we obtain the fraction of times the radios of STAs stay in different states, i.e., idle, receive and transmit.

2) N STAs downloading short files over TCP – In this scenario, we consider a constant number of users downloading short files over TCP. In between two downloads, there is a short period of inactivity or *think time*. To analyze this scenario, a Processor Sharing (PS) model is used to model the download rates provided to the files by IEEE 802.11 MAC. The PS service rate is obtained from the analysis of long file downloads mentioned in the previous paragraph. The processor sharing model has been previously explored in literature, like in [1], [2]. But here we do the analysis of the energy consumption using this model. And also, due to the role of beacons, the analysis is different in PSM, even though the underlying model is processor sharing.

This paper is organized as follows: Section II discusses the previous literature in the area of WLAN energy modeling. Section III provides an overview of the PSM and the queuing structure at the AP. Section IV states the modeling assumptions. In Section V, we analyze the scenario of TCP long transfers for both CAM and PSM. In Section VI, we analyze short file transfers. Finally, Section VII concludes the paper.

II. Related Work

Anastasi et al. [3] consider a single STA in PSM downloading a file over TCP in the presence of N saturated STAs. The authors evaluate the expected energy spent by the STA to download the file as a function of N. This is indirectly obtained by evaluating the contention time required to send a PS-POLL frame. In our work, we do not consider saturated STAs, instead, they are considered to be downloading files over TCP. Further, we have assumed a more realistic PSM protocol, which will be explained later.

Lei and Nilsson [4] consider STAs in PSM carrying downlink traffic in which the inter-arrival time between packets at the AP is exponentially distributed. They obtain the average packet delay due to queueing and the PSM protocol. They also

978-1-4244-5489-1/10/ $26.00

obtain lower and upper bounds on the average percentage of time a STA stays in sleep state. Baek and Choi [5] consider the same scenario as in [4] and evaluate the variance and exact expression for the average percentage of time a STA stays in sleep state. Si et al. [6] consider STAs in PSM mode, carrying downlink and uplink traffic. They describe the model, using which aggregate throughput and power consumption are obtained. We note that the models in [4], [5] and [6] assume Poisson arrivals of packets into the AP or STA, and hence, do not correctly model the traffic generated by TCP controlled file downloads.

In all the above papers [3], [4], [5] and [6], the authors consider PSM protocol implementation which is not practical in the presence of download type background traffic. They consider the following sequence of frame exchanges: First, the PSM STA sends the PS-POLL frame through contention, then after SIFS, the AP sends the data packet and after SIFS again, the STA sends the MAC ACK. So the AP does not contend to send data. In the presence of traffic from the AP to other STAs, when the AP receives the PS-POLL frame, some packets might be present already in the NIC queue of the AP, and these packets need to be sent first. We have analyzed the PSM protocol in the presence of traffic from other STAs, which is a more realistic scenario, and so we have considered an implementation of the PSM protocol in which the AP *contends* to send data to PSM STAs.

Krashinsky and Balakrishnan [7] consider a single STA in PSM doing very short file transfers (order of tens of KBs). They observe 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 [7] is further improved upon by Quiao and Shin [8]. They estimate the RTT of current TCP connection and using this information, the sleep wake schedule is made more efficient. However, the scope of their work is limited to only one STA, while here, we analyze the effect of background traffic, which plays a dominant role in determining energy consumption and delay for a file transfer.

Yong et al. [9] propose a way to minimize energy and delay by scheduling and informing the schedule to STAs through beacon frames. Tan et al. [10] propose to take advantage of *throttling* done by the TCP server in media streaming applications. Zanella and Pellegrini et al. [11] and Wang et al. [12] consider N saturated STAs in CAM mode and analyze the energy spent by them in radio communication. While in our paper, STAs are not considered to be saturated, but they are considered to be downloading files over TCP. Baiamonte et al. [13] propose to make use of the NAV set in RTS and CTS, using which the non intended receiver can switch to low power state during the upcoming transmission.

III. PSM - Overview

In this section, we discuss the implementation of the PSM protocol that we have considered, and the queueing structure at the AP for the packets destined for the PSM clients. When any STA switches to Power Save Mode (PSM), it goes to sleep state (switches off its radio) and also informs the AP about it. Packets arriving at the AP for PSM STAs are stored in separate queues maintained for each PSM STA; we call them *PSM Queues*, see Fig. 1. There is a NIC queue in which MAC PDUs are enqueued for transmission by the PHY layer. The AP sends beacon frames periodically, through which it informs PSM STAs about the packets enqueued for them. PSM STAs also wake up periodically to listen to the beacon frame. If, on receiving a beacon, an STA sees an indication that there is a packet enqueued for it, then it contends for the medium to send the PS-POLL frame. In reply to the PS-POLL frame, the AP immediately sends a MAC ACK. *This behavior is in contrast to earlier studies where it is assumed that a data packet is sent immediately.* It is not practical to assume that the AP can immediately send a 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 instant when AP receives the PS-POLL frame.

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. It may be noted that each PS-POLL packet permits the STA to download exactly one packet enqueued at the AP.

There are some situations in which the behavior of the entities is not specified in the standard, but is implementation-dependent. Such situations and the assumed behaviors of an STA and the AP are described here. After sending a PS-POLL, the STA marks its state as *waiting for unicast*. If before the STA receives the unicast packet, the AP transmits a beacon frame and it indicates that there are packets at the AP for this STA, then this STA will not generate another PS-POLL frame. But this may result in a deadlock when the packet that it is waiting for is lost, because then the STA will continue to be awake and will not send another PS-POLL. To prevent this situation, a timer is started when the STA sends the PS-POLL, and if the STA does not receive a packet before timer expiry, it goes to the sleep state. Subsequently in the next beacon interval, if the STA gets an indication, then it will send a PS-POLL to retrieve the packet from the AP. Further, if the beacon frame arrives at the STA when it is contending for PS-POLL, then it ignores the beacon frame, because the STA already knows that there is a packet at the AP for it.

IV. Modeling Assumptions

In this section we state the assumptions common to all the scenarios we have analyzed. We consider a single cell 802.11 WLAN with N STAs associated with a single AP. The STAs and the AP contend for the channel via the DCF

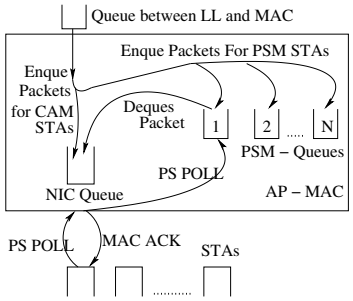


Fig. 1: Queuing Structure at AP

mechanism. We consider various scenarios in which either all the STAs are in CAM or all are in PSM. Here, we consider only TCP controlled download traffic, which means that the AP sends data packets to STAs, while STAs send TCP ACKs. We assume that the RTS/CTS mechanism is used by the AP to send data packets, while the basic access scheme is used by the STAs to send TCP ACKs. The following are our modeling assumptions:

- The WLAN is the bottle neck link of the system and the file server is assumed to be connected to the AP by a high speed LAN, which implies that the propagation delay between the AP and the TCP server can be neglected.
- In all the scenarios, at any instant, an STA has at most a single TCP connection.
- The application at the STA is such that flow control is never required and the advertised TCP window is always $W_{\max}$.
- The receivers do not implement the *delayed TCP ACK* strategy, i.e., every received packet generates a TCP ACK.
- The buffers are large enough so that there is no loss of packets due to buffer overflow at the AP or STAs.
- There are no packet losses due to the bit errors on the the wireless medium. Also, there are no packet drops due to the excessive collisions in the medium. The analysis can be extended to include packet loss due to bit errors; further, for TCP controlled transfers, the collision probability is indeed so low that the packets are rarely dropped at the MAC layer due to excessive collisions.
- We also assume that when there are k active STAs, then these STAs and the AP attempt in any slot with probability β_{k+1}, where β_{k+1} is long term attempt rate and is obtained via saturation analysis in [14].

V. Long File Transfer

In this section, we consider N STAs associated with the AP, with each one downloading a single large file over TCP. We consider two scenarios: 1) N STAs in CAM, 2) N STAs in PSM. For both the scenarios, we obtain expressions for throughput and average current drawn as a function of the number of STAs.

A. All STAs in CAM

Let $X_{ack}(t)$ be the total number of ACKs stored in all the STAs at any instant t, $X_{data}(t)$ be the number of data packets stored at the AP at t. Since the propagation delay between the AP and the server is negligible, so a data packet arrives immediately after the arrival of the TCP ACK at the AP. By assumption, $W_{\max}$ is the TCP window advertised by the receiver, so at any instant, $X_{ack}(t)+X_{data}(t) = NW_{\max}$. This implies that it is sufficient to keep track of either $X_{ack}(t)$ or $X_{data}(t)$. In the model, we assume that TCP ACKs are uniformly distributed among STAs, which is quite a valid assumption as there is no preference given to any STA. The model we use is a simplified version of the model described in [15], in which the authors consider both upload and download traffic, and evaluate the aggregate throughput. In the next section, we develop a new model for calculating energy expenditure rates.

Let us call the instants just after successful transmission of a packet on the medium, as the *success instants*, and denote the k^{th} success instant as G_k. Let the value of $X_{ack}(t)$ at instant G_k be X_k. Since, we are approximating IEEE 802.11 MAC by p-persistent model, in which every wireless entity attempts independently in every slot with probability β_k [14], where k is the number of active entities. Because of this, given the state of $X(t)$ at G_k, the future evolution of the process is independent of the past. Under the above assumptions, $\{(X_k; G_k), k \geq 0\}$ forms a Markov renewal sequence, and the process $X(t)$ forms a Markov regenerative process. The DTMC process (X_k) is shown in Figure 2. A transition from state i to $i+1$ represents the success of the AP and a transition from state i to $i-1$ represents success of some STA. Since the backoff parameters of all the STAs and the AP are same, if $X_k = i$, then at the next *success instant* the AP wins the contention with probability $1/(\min(N,i)+1)$ and one of the $\min(i,N)$ STAs wins with probability $\min(i,N)/(\min(i,N)+1)$.

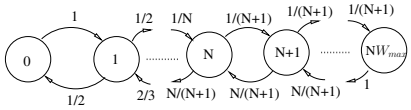


Fig. 2: DTMC of the process X_k

1) Average Current: In this section, expressions for the average current drawn by an STA are obtained. For this, we obtain the fraction of time any STA spends in different communication states. We define the following possible states:
1) Transmitting State (Tx): In this state, the STA is transmitting.
2) Receiving State (Rx): In this state, the STA is receiving. There could be following two substates of this state:

2.a) Receiving Decode state (RxD): In this state, the STA is receiving as well as decoding.

2.b) Receiving Listen state (RxLs): In this state, the STA is receiving but not decoding the data. This state is possible because of channel reservation by RTS-CTS mechanism. If the channel is reserved for two nodes, than any other node will know the length of the reservation from the Duration field in the RTS and CTS; so other node will listen to the ongoing transfer and can choose not to decode the corresponding packets. This can result in less consumption of power than in the receive decode state [16].

3) Idle State (Id) - The STA is in this state, when the channel is idle; no node is transmitting.
4) Sleep State (Sl) - In this state, STA is in sleep state and draws a very small current.
Let us denote the above states as $M_1 = Tx$, $M_2 = RxD$, $M_3 = RxLs$, $M_4 = Id$, $M_5 = Sl$. Let us denote J_{M_r} the current drawn by an STA while it is in state M_r. Let us define $Q_k(t)$ as the total charge drawn by STA k in the time interval $(0, t)$, then the average current (J_{av}) drawn by STAs can be written as follows:

$$J_{av} = \frac{1}{N} \sum_{k=1}^{N} \lim_{t \to \infty} \frac{Q_k(t)}{t} \tag{1}$$

Let us define the following indicator functions for an STA k:

$$I^k_{M_r}(u) = \begin{cases} 1 & \text{if STA } k \text{ is in state } M_r \text{ at instant } u \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Now, writing $Q_k(t)$ in terms of the above indicator functions:

$$Q_k(t) = \sum_{r=1}^{5} J_{M_r} \int_0^t I^k_{M_r}(u)\ du \tag{3}$$

By substituting, Eqn. 3 in Eqn. 1,and then rearranging it, we get the following equation for average current:

$$\begin{aligned} J_{av} &= \sum_{r=1}^{5} J_{M_r} \frac{1}{N} \lim_{t \to \infty} \frac{1}{t} \int_0^t \sum_{k=1}^{N} I^k_{M_r}(u)\ du \\ &= \sum_{r=1}^{5} J_{M_r} \frac{1}{N} \lim_{t \to \infty} \frac{1}{t} \int_0^t S_{M_r}(u) du = \sum_{r=1}^{5} J_{M_r} \Phi_{M_r} \end{aligned} \tag{4}$$

where, Φ_{M_r} as the fraction of time an STA spends in state M_r and $S_{M_r}(u) = \sum_{k=1}^{N} I^k_{M_r}(u)$. A finite sum and the limit can be exchanged; so the above rearrangement is valid. Then by Markov regenerative analysis [17], one can show the following:

$$\lim_{t \to \infty} \frac{1}{t} \int_0^t S_{M_r}(u) du = \frac{\sum_{k=0}^{NW_{max}} \pi_k E_k \left[\int_{G_{k-1}}^{G_k} S_{M_r}(u)\, du \right]}{\sum_{k=0}^{NW_{max}} \pi_k E_k[T]} \tag{5}$$

The detailed expression for $E_k[T]$ and $E_k \left[\int_{G_{k-1}}^{G_k} S_{M_r}(u)\, du \right]$ is given in [18].

B. All STAs in PSM

1) $N > 5$ – Development of the Model: In this section, we consider N STAs in PSM, downloading large files over TCP. In this scenario, the AP always contends for the channel, since for every two packets (1 TCP ACK + 1 PS-POLL) sent by each of the STAs, N packets need to be transmitted by the AP. Since, no preference is given either to the AP or to the STA, the above situation is possible, only if small number of STAs contend at any time, so that $2/3$ of the packets are transmitted by the STAs and $1/3$ of them are transmitted by the AP. Since, we are assuming negligible propagation delay between the AP and server, so at any time, most of the packets of the TCP windows of the STAs are present at the AP. Because of this, the "More" bit is always set in every data packet sent; so the STAs never go to sleep. On receiving a packet, the STA has to send a PS-POLL frame and a TCP ACK. Since the PS-POLL is a MAC level packet, it is enqueued at the HOL position in the transmission queue of the STA and the TCP ACK at the tail of the queue. If the transmission queue of the STA is empty when it receives the data packet, then immediately after the reception, its transmission queue will contain two packets, PS-POLL at the HOL position and TCP-ACK behind the PS-POLL. After STA sends a PS-POLL it starts contending for TCP-ACK. If the STA queue is nonempty (it implies that the STA is contending for TCP ACK) when it receives the data packet, then the STA will first transmit the PS-POLL. To transmit the PS-POLL, the STA will not sample the new backoff, but uses the residual backoff of the TCP ACK for which it was already contending when it received the data packet. It is not possible that STA receives the data packet when it is contending for PS-POLL, because it is only after the PS-POLL is sent, a data packet arrives at the STA.

When the AP receives a PS-POLL packet from the STA, then a data packet corresponding to this STA is brought into the NIC or transmission queue of the AP. There might be some packets already in the transmission queue of the AP (the probability of this increases with N), due to which this packet will be transmitted only after the packets preceding it are transmitted. During the time when the AP transmits these preceding packets, with high probability, the STA will transmit the TCP ACK; as a result the AP always sends a data packet to an STA that has an empty transmission queue.

Since no preference is given to the AP and the AP sends a single packet per PS-POLL, the transmission queue of the AP will build up for large value of N. The following can be inferred on the basis of the above discussion: 1) A packet successfully transmitted by the AP goes to an empty STA and the total number of contending STAs increases by one; 2) There are some STAs that are contending to send PS-POLLs and some are contending to send TCP-ACKs; 3) When a STA successfully transmits a PS-POLL, the number of STAs contending for TCP-ACK increases by one; 4) When a STA successfully transmits a TCP ACK, the number of STAs contending decreases by one.

Consider the process $X(t)$ of the number of STAs with a PS-POLL at the HOL position and TCP ACK behind it, and the process $Y(t)$ of the number of STAs with only a TCP ACK. Consider the joint process $(X(t), Y(t))$, embed it at the ends of success instants. Let us denote G_k the instant when the k^{th} successful transmission ends. Let us denote (X_k, Y_k) the value of the process $(X(t), Y(t))$ at G_k. Define $T_k = G_{k+1} - G_k$. Using the same arguments of p-persistent approximation, as stated earlier in the Section V-A, $\{(X_k, Y_k); G_k, k \geq 0\}$ forms a Markov renewal sequence, and the process $(X(t), Y(t))$ forms a Markov regenerative process. The transition probabilities of the Markov chain of (X_k, Y_k) depend on the number of active STAs, and are shown in Fig. 3. In Fig. 3, the x axis represents the process X_k and the

y axis represents process Y_k, and the state space is given by $\{(x, y) : 0 \leq x + y \leq N\}$. The transition probabilities are obtained using the fact that all nodes (Wireless entities in WLAN) have equal chance to transmit; so the transition probability from (x_1, y_1) to (x_1+1, y_1), which corresponds a successful transmission by the AP, is given by $1/(x_1+y_1+1)$. Other transition probabilities are also obtained in the same way.

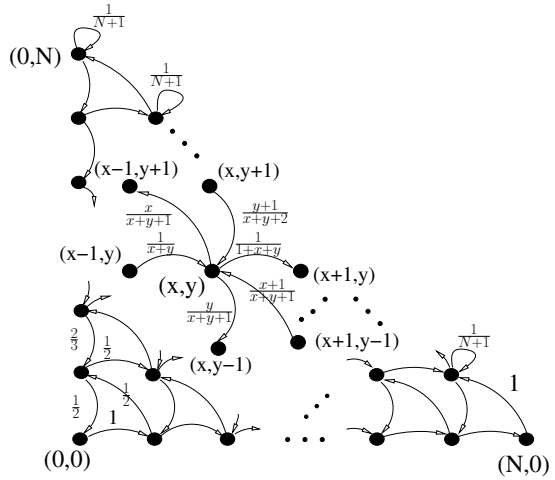


Fig. 3: 2-Dimensional DTMC of the process (X_k, Y_k)

2) $N > 5$ – Aggregate download throughput: Let the number of successful attempts by the AP in the k^{th} cycle be denoted by H_k; it could be either 1 or 0. Let $H(t)$ denote the number of successful attempts made by the AP in $(0, t)$. By Markov regenerative analysis [17] the following can be written,

$$\Theta_N = \lim_{t\to\infty} \frac{H(t)}{t} = \frac{\sum_{j=0}^{N-1}\sum_{i=0}^{N-j-1} \pi_{i,j}\frac{1}{i+j+1} + \sum_{i,j:i+j=N, i\neq N} \pi_{i,j}\frac{1}{N+1}}{\sum_{j=0}^{N}\sum_{i=0}^{N} \pi_{i,j} E_{i,j}[T]} \quad (6)$$

$\pi_{i,j}$ is the stationary probability of the process (X_k, Y_k). $E_{i,j}[T]$ is the expected time until the next success, starting with the state (i, j) and its detailed derivation is given in [18].

3) Average Current with N STAs in PSM: Equations 1– 4 remain valid for this scenario also. The expression for various fractions is given by the following equation:

$$\Phi_{M_r} = \frac{1}{N}\lim_{t\to\infty}\frac{1}{t}\int_0^t S_{M_r}(u)du = \frac{1}{N}\frac{\sum_{i=0}^{N}\sum_{j=0}^{N}\pi_{i,j}E_{i,j}\left[\int_{G_{k-1}}^{G_k} S_{M_r}(u)du\right]}{\sum_{i=0}^{N}\sum_{j=0}^{N}\pi_{i,j}E_{i,j}[T]} \quad (7)$$

The detailed expression for $E_{i,j}[T]$ and $E_{i,j}\left[\int_{G_{k-1}}^{G_k} S_{M_r}(u)du\right]$ is given in [18].

C. Simulation Results – Long Files

Simulation results are obtained using ns-2.33 [19] and the various parameters used are taken from the 802.11b standard

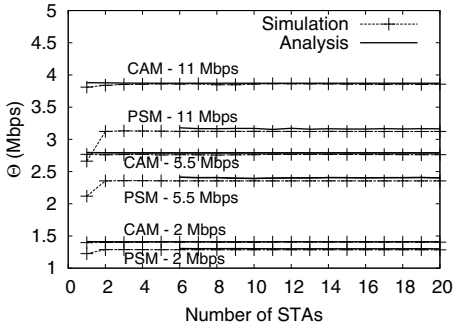
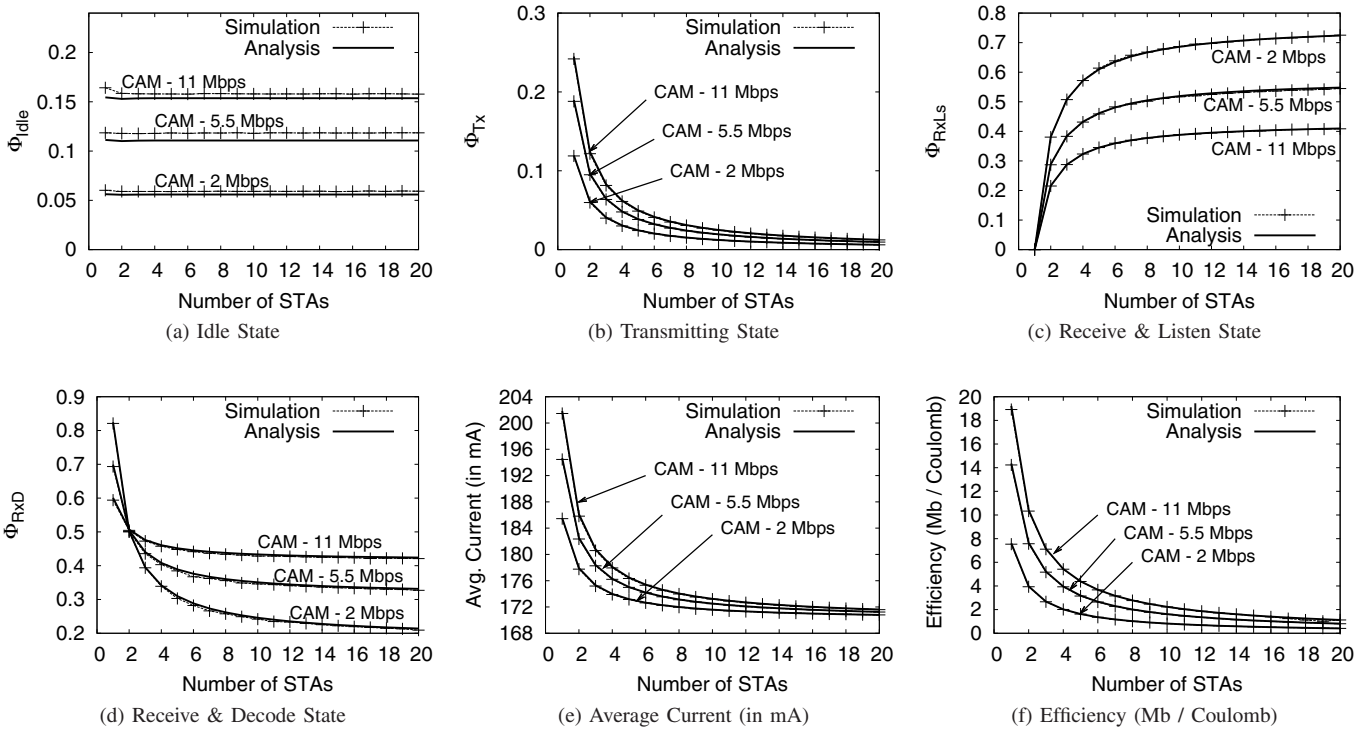


Fig. 6: Aggregate Throughput - (CAM & PSM)

(given in Table I). Data rate is taken as 2 Mbps to transmit control frames. To transmit data frames and MAC Header, data rate is taken as 2, 5.5, 11 Mbps. The TCP packet size is taken as 1500B and the RTS threshold taken as 300B, which means that the TCP ACK is sent using basic access and the data packet is sent using the RTS/CTS scheme. The values of current in different states of the radio are taken from the specifications of the Intel PRO/Wireless 2011 card [20]. Comparison of the throughput obtained in CAM and PSM is shown in Fig. 6. It can be seen that the aggregate throughput obtained by the STAs in PSM is less than that in CAM. The reason for this is the overhead of extra PS-POLL in case of PSM. Throughput in CAM is obtained using the model developed in Section V-A, and its analysis is shown in [18]. Figures 4 and 5 show the comparison of analytical and simulation results for fraction of times, average current and efficiency. Efficiency is obtained by dividing the throughput (in Mbps) by average current (in mA), which is equivalent to data downloaded (in Mb) per Coulomb of charge drawn by an STA. Figures 4a and 5a show the fraction of time an STA remains in the idle state for CAM and PSM respectively. It remains constant with number of STAs increasing. The time for which the channel remains in the idle state per data packet transmitted can be divided into three parts; 1) Time spent in backoff, this depends on the number of STAs contending, 2) Inter frame time, SIFS and DIFS, this remains constant, 3) Time spent in idling during collision, EIFS, this depends on the number of nodes contending. The throughput and the number of contending nodes [21] do not change with number of STAs. So the time spent in decrementing backoff counter and the number of collisions per data packet transmitted also do not change. The interframe time, DIFS and SIFS, is constant for a data frame. It can be inferred that a data packet is associated with a constant idle time, irrespective of the number of STAs. Since the transmission and receive times of frames depend only on the data rate, so the fraction of time an STA stays in idle state remains constant.

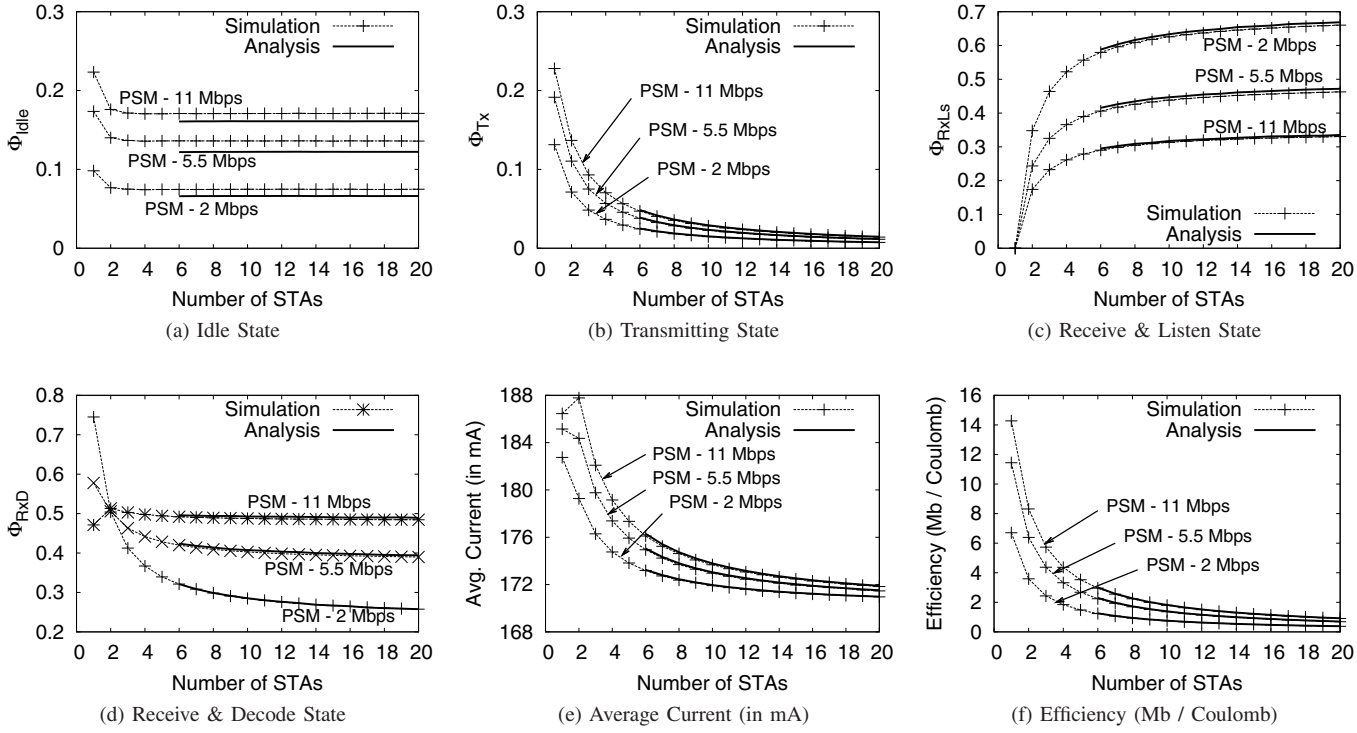
The throughput share of a single STA decreases as the number of STAs increases. This implies that as the number of STAs increases, an STA spends more time in receiving data frames for other STAs. An STA stays in Receive & Listen state (RxLs) state while it is receiving data frames for other STAs. So, the fraction of time an STA stays in RxLs state increases



(a) Idle State (b) Transmitting State (c) Receive & Listen State

(d) Receive & Decode State (e) Average Current (in mA) (f) Efficiency (Mb / Coulomb)

Fig. 4: Continuously Active Mode



(a) Idle State (b) Transmitting State (c) Receive & Listen State

(d) Receive & Decode State (e) Average Current (in mA) (f) Efficiency (Mb / Coulomb)

Fig. 5: Power Save Mode

(Figs. 4c and 5c) with increasing number of STAs. Also, because of this reason, the fraction of time any STA remains in Receive & Decode (Figs. 4b and 5b) and transmitting state decreases (Figs. 4d and 5d) with increasing number of STAs.

An STA transmits the following frames per data frame it receives: CTS, MAC ACK and TCP ACK. Being control frames, CTS, MAC ACK are transmitted at 2 Mbps, so the transmission time of CTS and MAC ACK does not change with data rates. Also, the transmission time of TCP ACK does not vary much with changing data rates because of its size (98 bytes). As the data rate increases, the throughout also increases, which implies that if we consider a time interval then with increasing data rates, we can pack more data packets in it. When the number of transmissions of data frame increases, then the number of transmissions of aforementioned frames increases also. Since the transmission time of these frames does not change with data rate, so the total transmission time increases, which leads to increase in the fraction of time an STA stays in transmission state with increasing data rate (Figs. 4b and 5b).

Recalling, a data frame is associated with constant idle time. As the data rate increases, in a given time interval, the number of data packets transmitted also increases. So with increasing data rate, the idle duration in the time interval increases, hence the fraction of time an STA spends in the idle state increases (Figs. 4a and 5a).

Since the fraction of time an STA stays in idle and transmission state increases, so the fraction of time during which the STA stays in receive state (RxLs + RXD) decreases with increasing data rate. This can be obtained by adding the values shown by Figs. 4c and 4d for CAM and Figs. 5c and 5d for PSM.

With the number of STAs increasing, the fraction of time an STA spends in transmitting state decreases and transmit current is more than the idle and receive current (Tab. I); so, with the number of STAs increasing, the average current decreases (Fig. 4e for CAM and Fig. 5e for PSM) and converges to idle current for large number of STAs. Since the throughput of a single STA decreases as $1/N$ and the average current converges to a constant value, so the efficiency as defined above decreases as $1/N$. On comparing Fig. 5f and Fig. 4f, it is clear that for the long file transfer case, CAM has higher efficiency than PSM, it is because of the overhead of PS-POLL in case of PSM.

VI. Short Files

Short file downloads and *think times* between downloads is the typical behavior of a user browsing the Internet. We assume that all the files are part of 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 used. For every file, an HTTP request packet is sent by STAs to initiate the transmission [22].

With the number of STAs increasing, the aggregate throughput of the AP does not change, as observed in the previous section, and this throughput is equally shared by all the

TABLE I: Parameters

Parameter used			
Parameter	Value	Parameter	Value
EIFS Time	364 μs	RTS Size	20 bytes
SIFS Time	10μs	PS-POLL Size	20 bytes
DIFS Time	50μs	CTS Size	14 bytes
System Slot time	20μs	MAC ACK Size	14 bytes
PLCP Header time	144 μs	IP Header	20 bytes
PHY Header time	48 μs	TCP data size	1500 bytes
MAC Header Size	34 bytes	TCP Header Size	20 bytes
J_{Id} J_{RxD}, J_{RxLs}	170 mA	J_{Tx}	300 mA
J_{Sl}	10 mA		

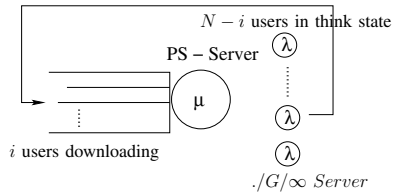


Fig. 7: Closed queueing network model for short file downloads. The service of file downloads at the WLAN is modeled by a processor sharing server; the bars behind the PS server represent the residual file sizes.

STAs. Thus, the AP can be modeled as a Processor Sharing (PS) server and the think time can be modeled as the time spent at a $./G/\infty$ server. This is analogous to the Closed Queueing Network model in which there is a constant number of customers alternating between the Processor Sharing server (AP) and at a $./G/\infty$ server as shown in Fig. 7. Think time is considered to be exponentially distributed with mean $\frac{1}{\lambda}$ and file size distribution is taken as exponentially distributed with mean L. So the service time of a single file being downloaded alone is exponentially distributed with mean $\frac{1}{\mu} = \frac{L}{\Theta}$, where Θ is the aggregate throughput seen by STAs downloading large files, as obtained in the previous section. For this scenario, we are interested in obtaining two metrics:

1) Average charge ($E[Q_f]$) per file – It is defined as the total charge drawn by all the STAs in a given interval divided by the total number of files downloaded by all of them in the same interval.

2) Average sojourn time ($E[S]$) – It 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 same interval. Here, the time taken to download a files is taken as the time difference between the instant the STA starts contending for the HTTP request packet and the instant it receives the last packet of the file.

A. All STAs in CAM

If $X(t)$ is the number of ongoing short files transfers at t, then the number of STAs in the think state at t will be $N - X(t)$. $X(t)$ is a CTMC, because service time and think times are exponentially distributed. Fig. 8 shows the transition

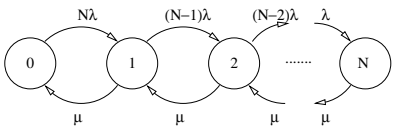


Fig. 8: Transition rate diagram of $X(t)$

rate diagram of $X(t)$.

1) Expected charge drawn by a STA per short file downloaded ($E[Q_f]$) : Define $J_{k,a}$ (derived in Section V-A) as the average current drawn by k STAs when they are downloading long files and let $J_{k,p}$ be the average current drawn by a STA listening (not doing any activity) to the traffic of k STAs downloading long files (derived in [18]). Let $I_{\{X(t)=k\}}$ be the indicator function indicating k STAs active at any instant t. Let $Q_j(t)$ be the total charge drawn, and $n_j(t)$ be the number of downloads completed by STA j in time interval $(0,t)$. Our aim is to evaluate the average charge drawn by STAs per file, which is given by the following equation:

$$E[Q_f] = \lim_{t\to\infty} \frac{\sum_{j=1}^{N} Q_j(t)}{\sum_{j=1}^{N} n_j(t)} = \lim_{t\to\infty} \frac{\frac{\sum_{j=1}^{N} Q_j(t)}{t}}{\frac{\sum_{j=1}^{N} n_j(t)}{t}} \tag{8}$$

Note that if the limits (as $t \to \infty$) in the numerator and the denominator exist, then these are, respectively, the rate of consumption of charge in all the STAs, and the total rate of transfer of short files (over all the STAs). Now, if there are exactly k STAs active and downloading throughout the interval $(0,t)$ and $N-k$ in think state during that duration, then the following expression gives the total charge drawn by all the STAs in the time interval $(0,t)$:

$$\int_0^t [kJ_{k,a} + (N-k)J_{k,p}]I_{\{X(t)=k\}}dt \tag{9}$$

On summing the above expression over all k, we get the sum of the charge drawn by all the STAs, in the time interval $(0,t)$. After summing the above expression and then substituting it in Eqn. 8, and then applying the limit on it, we get the following equation:

$$E[Q_f] = \frac{\sum_{k=0}^{N} \pi_k[kJ_{k,a} + (N-k)J_{k,p}]}{\sum_{k=0}^{N} \pi_k(N-k)\lambda} \tag{10}$$

where π_k is the stationary probability of k STAs downloading files and $(N-k)$ STAs in think state.

2) Expected Sojourn Time E[S]: Using Little's Theorem [17] , following expression can be written for expected sojourn time:

$$E[S] = \frac{\sum_{k=0}^{N} k\pi_k}{\sum_{k=0}^{N} \pi_k(N-k)\lambda} \tag{11}$$

B. All STAs in PSM

In this scenario, STAs are in PSM. So when the user is in the think state, the STA goes to sleep. When the user requests a file, the STA wakes up and sends a HTTP request packet and then again goes back to sleep. Since we are assuming the server to be local to the LAN, so packets from the server in response to the request, arrive immediately at the AP. This information is sent to the STA in the next beacon frame. This means that the STA starts getting service at the beginning of the next beacon interval. After this, the STA is assumed to remain awake till the whole file is downloaded. The interaction between TCP slow start and PSM [7] can be ignored in our case. It is a reasonable approximation because the RTT between the AP and the TCP server is negligible in our case; so data packets arrive immediately in response to the TCP ACK, and due to this, the STA does not go to sleep state. Further, we consider file downloads in the presence of download type traffic to other STAs. This decreases the net throughput to a single STA. Hence, the sojourn time of the file increases, and so the time spent in slow start becomes less dominant.

Let $X(t)$ denote the number of STAs in the download state at time t, and X_k, $k \geq 0$ the value of $X(t)$ embedded at the beacon instants. Since the file sizes and think times are taken to be exponentially distributed, so the process X_k is a DTMC. The transitions of the Markov chain are governed by the number of files completing transfer and the number of users completing their think times in the beacon interval. To simplify the calculation of transition probabilities, we assume that the users who complete their downloads start their think times from the next beacon interval, so that the number of users that complete their think times in a beacon interval do not depend on the number of users who complete their transfers in the same beacon interval. This assumption is justified since the beacon interval is generally 100 ms and the think time is generally of the order of seconds; hence the probability of a user completing its think time within one beacon interval is very small.

1) Transition probabilities of the Markov Chain: Let N be the total number of STAs associated with the AP, and b the duration of the beacon interval. Since we assume that the think times of STAs are exponentially distributed with mean $\frac{1}{\lambda}$, so the probability that a user finishes his think time within interval of b is $1-e^{-\lambda b}$. If there are i customers downloading files at the start of a beacon interval, then $N-i$ users are in think state, and then the probability that k users finish their think times within the beacon interval is $Bi(N-i,k) = \binom{N-i}{k}(1-e^{-\lambda b})^k(e^{-\lambda b})^{N-i-k}$. Let $q(i,m,b)$ be the probability that m users complete their downloads out of i active users within the time interval of b. This probability depends on the mean file length; we have assumed the files to be exponentially distributed with mean $L = \frac{\Theta}{\mu}$, where Θ is the throughput obtained in the previous section for the large file download case. Let us denote $p_{i,j}$ the probability that there will be j users downloading files at X_{k+1}, given that there were i users downloading files at X_k. The transition probability ($p_{i,j}$) of X_k can be written as follows:

$$\begin{aligned} p_{i,j} &= \sum_{m=\max(0,i-j)}^{\min(i,N-j)} q(i,m,b)Bi(N-i,j-i+m) \\ &\qquad\qquad 0 < i,j \leq N,\ j = 0 \\ p_{0,j} &= Bi(N,j) \qquad\qquad 0 \leq j \leq N \end{aligned} \tag{12}$$

where, $q(i,m,b)$ is given by the following equation:

$$q(i,m,b) = \begin{cases} \frac{\mu^m e^{-\mu b} b^m}{m!} & m < i \\ 1 - \sum_{s=0}^{m-1} \frac{\mu^s e^{-\mu b} b^s}{s!} & m = i \end{cases} \quad (13)$$

2) Calculation of Sojourn Time: Using Little's Theorem, the following expression can be written for the expected sojourn time:

$$E[S] = \frac{\sum_{k=0}^{N} k\pi_k}{\lim_{t\to\infty} \frac{1}{t} \sum_{j=1}^{N} n_j(t)} \quad (14)$$

where $n_j(t)$ is the number of downloads completed by the user j in $(0,t)$ and π_k is the stationary probability of $X(t)$. The above expression only account for the time for which the STA stays in active state. It does not accounts for the time duration between the instant it sends the HTTP request and the next beacon instant. The expected value of this duration is $\frac{b}{2}$. By definition, this duration also is need to included in the sojourn time. So the total sojourn time of the file is the sum of the above expression (Eqn. 14) and $\frac{b}{2}$.

3) Calculation of average charge drawn per file: In this scenario, STAs download a file and then move to the Think state. During think time, STAs stay in sleep state except when they wake up to listen to the Beacon Frames. As the Beacon frame is sent through contention, so STAs have to be awake for some duration to be able to listen to it. Let us call this duration T_{Lb}. The mean number of times STAs come to the active state during think time is equal to the expected think time divided by the beacon interval ($\frac{1}{b\lambda}$). So the mean total duration for which STAs stay in active state during think time is $T_{Lb}(\frac{1}{b\lambda})$. Using the Equations 8 – 10, and taking the current drawn by STAs in think state as J_{Sl}, the following equation for the expected charge drawn per file can be written:

$$\begin{aligned} E[Q_f] =& \frac{\sum_{k=0}^{N} [kJ_k + (N-k)J_{Sl}]\pi_k}{\lim_{t\to\infty} \frac{1}{t} \sum_{j=1}^{N} n_j(t)} \\ &+ J_{Id} T_{Lb} \left(\frac{1}{b\lambda}\right) - J_{Sl} T_{Lb} \left(\frac{1}{b\lambda}\right) \end{aligned} \quad (15)$$

where, J_k is the average current drawn by the k STAs, which are downloading files. It is to be noted that we have not modeled long files transfer in PSM scenario for $2 \leq N \leq 5$, so to evaluate J_2 to J_5, we extended the model of PSM for $N > 5$. However, we have modeled PSM for $N = 1$, and is shown in the [18], and the value of J_1 is obtained using it. $\lim_{t\to\infty} \frac{1}{t} \sum_{j=1}^{N} n_j(t)$ is derived in [18].

C. Simulation Results - Short Files

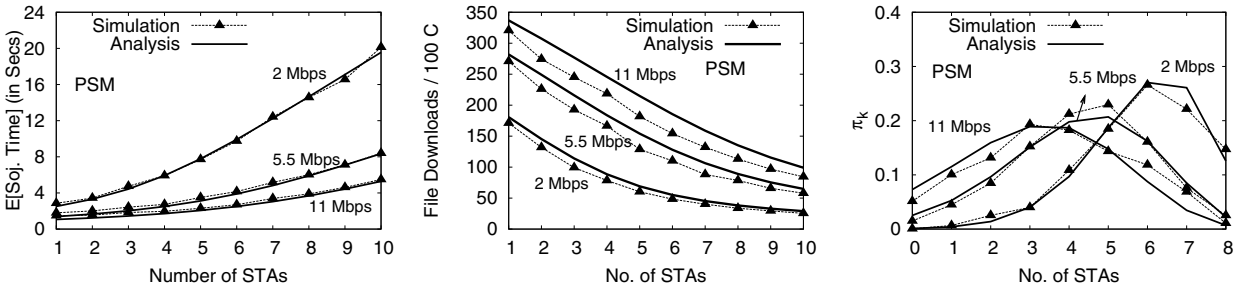
Simulation results are obtained using ns-2.33 and the other parameters are same as stated earlier in Section V-C. To generate HTTP traffic in ns, we used PACKMIME [23]. The file size is taken to be exponentially distributed with mean $400KB$, and the think time is taken to be exponentially distributed with mean $5\,Secs$. The beacon interval is taken as $100\,ms$ and the time duration for which an STA switches to CAM, to listen to beacon frame, is taken as $5\,ms$.

Figures 10a and 9a show the comparison of sojourn times obtained using analysis and simulation, for PSM and CAM. It can be seen that the delay incurred in downloading files for CAM is slightly lesser than in PSM. This is due to lesser throughput achieved in PSM than in CAM. Figures 10b and 9b give the comparison of the simulation and the analytical values of the number of downloads that can be completed in a given battery capacity. Here, battery capacity is taken in the form of 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 ($100\,Coulomb$) by the expected charge drawn in downloading a file.

It is clear from Fig. 10b and Fig. 9b that Static PSM is more efficient than CAM. The reason behind this is that the PSM STA goes to sleep state when it is not downloading anything; this is not the case with CAM. The PSM will perform even better if the think time between the downloads increases, since then the CAM will be wasting more energy during idling. Further improvement in the PSM is possible by increasing the beacon interval, so that the STA does not have to wake up at every beacon instant, but it will increase the delay. It is to be noted, that with the number of STAs increasing the number of file downloads that can be completed in a given battery capacity decreases, because in this case while downloading its own file STA has to overhear the frame destined to other STAs also. Figs. 9c and 10c shows the stationary probability of n stations receiving service, when there are a total of $N = 8$ STAs associated with the AP. It is clear from the figures that there is considerable probability of more than one STA being active. Our future work will be focussing on this problem of decreasing efficiency with increasing number of STAs associated with the AP.

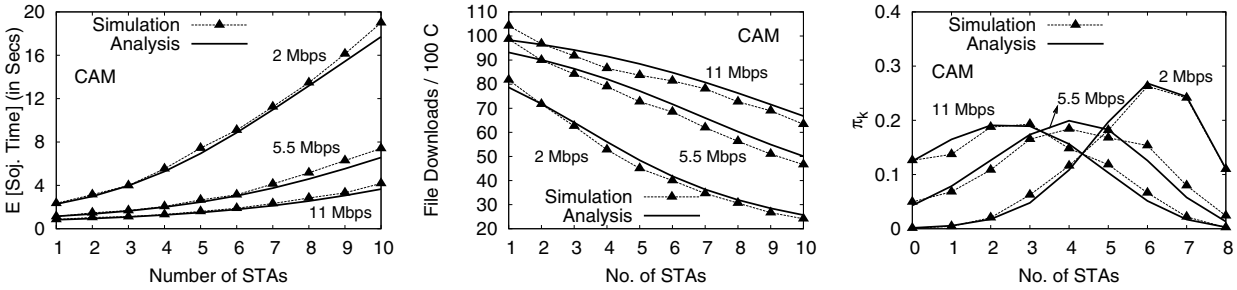
VII. Conclusion

In this paper, our contribution is two fold; firstly, we have modeled the energy consumption of TCP controlled large file transfers in CAM and in PSM, in the presence of download type TCP background traffic; this is absent in the literature. Secondly, we modeled the energy consumption of TCP controlled short file transfers when all the STAs are in CAM and in PSM. We have seen that our analytical results match quite well with the simulation results, which shows the correctness of our analysis. We have also shown that the PSM performs better than the CAM when the user remains inactive for some time in between active periods. However, if there is no inactivity, then the performance of the PSM starts to degrade and it performs worse than the CAM, as evident from the large file download case. In the future work, we will study the performance of PSM STA downloading short files, in the presence of CAM STAs carrying similar traffic. Further, we will study Adaptive PSM which will have features of both CAM and PSM; it does not have the extra overhead of PS-POLL and also can go to sleep state if user is not active for a certain time.



(a) Expected Soj. Time (b) No. of File Transfers / 100 C (c) π_k $(N = 8)$

Fig. 9: Power Save Mode: Short Files over TCP



(a) Expected Soj. Time (b) No. of File Transfers / 100 C (c) π_k $(N = 8)$

Fig. 10: Continuously Active Mode: Short Files over TCP

REFERENCES

[1] R. Litjens, F. Roijers, J. L. van den Berg, R. J. Boucherie, and M. Fleuren, "Performance analysis of wireless lans: an integrated packet/flow level approach," http://eprints.eemcs.utwente.nl/3496/, Enschede, Memorandum 1676, 2003.

[2] D. Miorandi, A. A. Kherani, and E. Altman, "A queueing model for http traffic over ieee 802.11 wlans," *Comput. Netw.*, vol. 50, no. 1, pp. 63–79, 2006.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] F. D. P. Andrea Zanella, "Mathematical analysis of ieee 802.11 energy efficiency," in *Proc. Int. Symp. The 7th International Symposium on Wireless Personal Multimedia Communications(WPMC2004)*, 12–15 September 2004, pp. p.V1–97–p.V1–101.

[12] X. Wang, J. Yin, and D. P. Agrawal, "Analysis and optimization of the energy efficiency in the 802.11 dcf," *Mob. Netw. Appl.*, vol. 11, no. 2, pp. 279–286, 2006.

[13] V. Baiamonte and C.-F. Chiasserini, "Saving energy during channel contention in 802.11 wlans," *Mob. Netw. Appl.*, vol. 11, no. 2, pp. 287–296, 2006.

[14] A. Kumar, E. Altman, D. Miorandi, and M. Goyal, "New insights from a fixed point analysis of single cell ieee 802.11 wlans," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 3, March 2005, pp. 1550–1561 vol. 3.

[15] R. Bruno, M. Conti, and E. Gregori, "Performance modelling and measurements of tcp transfer throughput in 802.11-based wlan," in *MSWiM '06: Proceedings of the 9th ACM international symposium on Modeling analysis and simulation of wireless and mobile systems*. New York, NY, USA: ACM, 2006, pp. 4–11.

[16] "Power consumption and energy efficiency comparisons of wlan products," Atheros. [Online]. Available: www.atheros.com/pt/whitepapers/atheros_power_whitepaper.pdf

[17] R. W. Wolff, *Stochastic Modeling and the Theory of Queues (Paperback)*. Prentice Hall, 1989.

[18] 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," *CoRR*, vol. abs/0909.3717, 2009.

[19] "The network simulator ns-2." http://www.isi.edu/nsnam/ns/.

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

[21] G. Kuriakose, S. Harsha, A. Kumar, and V. Sharma, "Analytical models for capacity estimation of ieee 802.11 wlans using dcf for internet applications," in *Wireless Networks, Springer*, vol. 15, 2009, pp. 259–277 vol. 3.

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

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