Energy Efficient Scheduler Design in Wireless Networks

Jeongjoon Lee[1], Catherine Rosenberg[1], and Edwin K. P. Chong[2]

[1]School of Electrical and Computer Engineering
Purdue University, West Lafayette, IN 47907-1285
jjlee@purdue.edu, cath@ecn.purdue.edu

[2]Dept. of Electrical and Computer Engineering
Colorado State University, Fort Collins, CO 80523-1373
echong@engr.colostate.edu

Abstract. Minimizing energy consumption is crucial for portable wireless communication systems because they operate on a limited battery supply. For example, the IEEE 802.11 standard includes a mechanism called power-saving mode (PSM), which allows a network interface on a mobile station to enter a sleep state whenever possible to reduce its energy consumption. However, the scheme needs to be complemented by an efficient scheduling mechanism to maximize the energy savings. The purpose of this paper is to show how well-designed scheduling disciplines along with a power-saving mode can achieve significant improvement in saving energy consumption.

We propose a generic wireless system composed of an access point and several portable stations that offer a PSM to its users. Our PSM is different from that of the IEEE 802.11 in the sense that to save more energy our PSM is driven and controlled by the access point, while the process is much more under the control of the stations in IEEE 802.11. We study how to design the downlink scheduler in the access point so as to minimize the total energy consumption of all the portable stations that are in PSM. We formulate two scheduling problems (in a static context and in a dynamic one) and propose several heuristic scheduling policies to solve them. Extensive simulations show the significant improvement on energy saving of our policies over more traditional scheduling schemes.

1. Introduction

Wireless stations have become an integral part of our day-to-day lives. A key concern in the design and development of such wireless stations is energy consumption. Although advances in battery technology and low-power circuit design have been significant, dramatic improvements in battery technology to meet the energy needs for future mobile stations are not expected in the near future [1]. As a result, minimizing power consumption is a major factor driving the design of mobile stations and of the protocols therein. As the authors of [2] show, a large part of the power drain of a mobile station connected to a wireless LAN can be attributed to its wireless LAN card.

While this paper is not specifically on IEEE 802.11, it is interesting to discuss this wireless LAN technology as it is the most important example of data-oriented wireless technology to date. In IEEE 802.11, a wireless network interface can be in either the *awake* or *doze* states. In the awake state, there are three different modes—TX (transmit), RX (receive), and IDLE (idle)—and each consumes different amounts of energy. In the doze state, there are two different modes—SLEEP (sleep) and OFF (power off)—and the wireless network interface consumes much less energy in this state than in the awake state. Many researchers have closely studied through measurements the source of power drainage in wireless LAN cards, and have observed that the power consumption in idle mode is significant and even comparable to the receive-mode power consumption ([2], [3], [4], [5]). It has been shown by Stemm et al. [2] that leaving a wireless LAN card in sleep mode whenever possible can dramatically reduce its power consumption. The IEEE 802.11 standard [6] recently introduced a power-saving mode (PSM), which offers an opportunity for stations that need it to reduce their power consumption by going into sleep mode on a regular basis. In the PSM specified in IEEE 802.11, the Access Point (AP) sends a Traffic Indication Map (TIM) containing information about its downlink queues (one per PSM station) at the beginning of each Beacon Period (BP). The TIM is the way to inform the PSM stations that care to listen if the AP has packets to deliver to them. It is then left to the station to decide when to request the sending of its packets.

We propose a PSM for a generic wireless LAN, which is different in many respects from that of IEEE 802.11. In particular, we give a more central role to the AP since we believe that this will be necessary for offering QoS to some users and for taking full advantage of a power-saving mode. The purpose of this paper is to show how being in PSM yields considerable energy savings for a station as compared to the Constantly Awake Mode (CAM), when an efficient scheduling mechanism is used at the AP to manage the access to the downlink.

In IEEE 802.11, every station (STA) that is associated with the AP in the PSM is assigned an association identifier (AID) during the association process, and the AID maps to a single bit in the TIM that indicates if there is any packet queued in the AP for that station. However, the TIM does not indicate the order of transmissions, the number of packets queued or to be transmitted, or the time when the station should awake or go into sleep mode. In addition to the Single Bit TIMs (SBT) used in the 802.11 PSM, Stine et al. [7] propose three alternative directory protocols, Multiple Bit TIMs (MBT), Single Address Lists (SAL), and Two Address Lists (TAL), that may be used by a central node (e.g., the AP) to coordinate the transmission of data and the dozing cycles of PSM STAs. For example, the MBT uses multiple bits of the TIM identifying the number of exchanges in which a node participates. At the cost of these additional bits, the nodes may doze in the time preceding their packets, assuming an implicit order. The operation of the SAL is similar to the MBT with the difference that the addresses of the receiving nodes are listed in order. The TAL consists of two addresses for each packet transmission indicating which pairs of nodes are to awaken for each transmission in an ad-hoc network. While they also mention some scheduling issues, they mainly focus on directory protocols and simply use the SPT (Shortest Processing Time) scheduler [8], and hence this work is different from ours. Krashinsky et al. [9] study, via simulation, the performance-energy tradeoff when the IEEE 802.11 PSM is used and show that the Round Trip Time (RTT) of a TCP connection can increase substantially with the PSM. They also find that the power consumption used to listen for TIMs can dominate the total energy consumption if the network is accessed only sporadically, and hence they propose to adapt the sleep duration depending on the past activity of the station to allow the network interface to sleep for longer periods of time when there is no activity, thereby reducing the energy consumed to listen to beacons. This work is different from ours in spirit. They focus on reducing the power consumption due to listening to beacons when the STAs are very sporadically active, while we focus on minimizing the overall total power consumption for all the levels of activities of the STAs. In addition, they are trying to reduce the power consumption individually for each STA by varying its sleep period, while our approach tries to reduce the overall power consumption of all the STAs in PSM by using the proper scheduling scheme at the AP.

This paper is organized as follows. The Generic Power Management Model (GPMM) and the other assumptions we make for our study are introduced in Section 2. The problem formulation and the main results for both static and dynamic assumptions are presented in Section 3. Our conclusion and references follow.

2. Generic Power Management Model and General Assumptions

We propose a Generic Power Management Model (GPMM) that keeps some of the main features of the PSM of IEEE 802.11 while differing in other aspects. The main difference is that in GPMM the PSM is under the control of the AP rather than of the individual STAs. The GPMM is defined on the following wireless system comprising an Access Point (AP) and several stations (STA). To simplify, we assume that the downlink and the uplink are separated and that the AP is in full control of the downlink. Each STA can either be in the normal mode or in PSM (Power Saving Mode). When the STAs are in normal mode, they stay awake all the time (i.e., they never go into SLEEP mode). In the following, we assume that all the STAs are PSM-enabled. Note that, while we describe GPMM in the WLAN context, it could also be used in a satellite context.

We assume that there is a message exchange between the AP and a STA that wants to go into PSM mode (we assume in our scenario that this is done when a station associates with the AP). Once this message exchange is completed, the AP allocates some buffer space to the new PSM STA and starts buffering the incoming packets for this new PSM STA. The AP proceeds as follows.

Before the beginning of a new Beacon Period (BP) composed of *$(L+1)$* timeslots, the AP decides which packets to send in the coming BP using its own scheduling policy. The AP then prepares a TIM and sends a beacon with the TIM in the first timeslot of the BP. We assume that the TIM takes exactly one timeslot. The TIM is a bitmap consisting of one bit for each PSM STA indicating if there is at least one packet **scheduled** during the upcoming BP. That is, if there is at least one packet ready to be delivered by the AP to a PSM STA during the BP, then the corresponding bitmap is 1. Otherwise, it is 0. This is different from the TIM mechanism in IEEE 802.11, where the AP sets the TIM to 1 if there is at least one packet currently queued for the STA, regardless of whether or not the packet(s) destined to this STA will be scheduled during the corresponding BP.

We assume that the PSM STAs **have to** wake up at the beginning of **each** BP to listen to the TIM. If the bitmap for the STA is not set in the TIM, then the STA goes back immediately to sleep. Otherwise, the STA has to remain awake until the last packet scheduled for it in this BP is delivered. The STA knows if a packet it has received is the last one in the BP by inspecting the MORE DATA bit field in the packet header. If the MORE DATA bit is set to zero, it indicates that the packet is the last one to be scheduled in this BP, and hence after receiving the packet from the AP, the corresponding STA goes back into the SLEEP mode and stays in this mode until the beginning of the next BP.

We assume that the STAs are always synchronized in time with the AP. Without loss of generality, we assume that the power consumption in IDLE mode, P_{IDLE}, is equal to the power consumption in RX mode, P_{RX}, and we assume that the power consumption in SLEEP mode is negligible. We also neglect the transition delays and the energy consumption when a STA changes its states. We assume that the packets are of fixed size and that a timeslot is designed to serve exactly one packet. The service is assumed to be *gated*, i.e., packets arriving in a BP are served only in or after the next BP (this is a reasonable assumption since the AP has to prepare the TIM in advance).

We describe the differences between our model and the IEEE 802.11 as follows. In IEEE 802.11, a STA is responsible for informing the AP, in its association request, of how often the STA will be awake (i.e., of the number of BPs—henceforth called the ListenInterval—between two consecutive waking-ups). If a ListenInterval of *n* BPs has been agreed upon by the AP and the STA, the station has to awake every *n* BPs to check if the AP has any data to send to it by listening to the TIM. By having a large ListenInterval, the station can stay longer in the SLEEP mode and, hence, save more energy while trading off QoS parameters such as delay. In the IEEE standard, a PSM STA when it awakes reads the TIM and requests the delivery of its buffered packets by sending a PS-POLL packet to the AP. The AP will respond to each PS-POLL by sending one packet to the requesting STA. In this way the STA controls when it starts receiving packets. While the mechanism related to ListenInterval and PS-POLL in IEEE 802.11 gives the stations flexibility to partially control when the buffered packets will be sent so as to save energy, note that sending a PS-POLL does not guarantee the AP's rapid response. The AP may be congested, in which case the STA has to remain awake (i.e., in the IDLE mode) until the packet is eventually delivered without knowing if the delivery will be made in the next BP or in a subsequent BP. Hence, the energy saved by having a larger ListenInterval may be offset by increased IDLE-mode power consumption.

3. Main Results

3.1 Static Case

We first consider, in this section, the **static** scenario in which the number of PSM STAs within the coverage area of the AP, *M,* is fixed, and we focus on how to schedule in the most energy-efficient way a number of packets that have arrived before the BP under consideration, which we call the *first BP* to simplify the notation. This scenario is "static" in that, for each test case, once packets are queued at the beginning of our first BP, we assume that there are no more packets coming, and the scheduler's task is to send only the packets that are currently in the queue in the most power efficient manner. This scenario will help us gain insights about how to schedule in an energy-efficient way, and these insights will be used later in a more dynamic context.

A *work-conserving* discipline is defined as a policy that does not allow the server to be idle if there are packets in queue, and in this sense the discipline is "throughput-optimal." On the other hand, if we think of the nature of the PSM stations, which consider that saving more energy by trading off QoS parameters such as delay is acceptable, non-work-conserving disciplines are also worth investigating. Indeed, by putting themselves in PSM, the stations already agree to sacrifice QoS parameters such as delay and throughput, because while they are in SLEEP mode they cannot receive or send any packet, and this impacts the delay seen by their packets.

Under our static assumption, we define what we call a *semi-work-conserving* discipline, which is a non-work-conserving discipline defined as follows: Assume that *N* packets have to be scheduled at the beginning of the BP under consideration and that no more packets will arrive later. Due to the fixed number of timeslots in a BP (the first timeslot being reserved for the TIM and the next *L* timeslots being reserved for data transmissions), the number of BPs required to schedule all the *N* packets is $Q = \lceil N / L \rceil$. Under a work-conserving discipline, a schedule is determined regardless of *Q*, and there is no timeslot wasted until the last packet is scheduled (i.e., if there is a need to waste timeslots because *N/L* is not an integer, then the wasted timeslots are all at the end of the last BP (i.e., the Q^{th} BP).) In contrast, under our semi-work-conserving discipline, the packets do not have to be scheduled consecutively, and the only restriction is that the packets currently queued should be scheduled within *Q* BPs, i.e., the number of BPs used to send the *N* packets is the same for both the work-conserving and the semi-work-conserving disciplines.

Problem Formulation for the Static Case and Preliminary Results. We formulate the problem to find an optimal semi-work-conserving scheduling discipline in the static case as follows: find the semi-work-conserving scheduling discipline U^* that minimizes the overall power consumption of the PSM STAs, E:

$$E = \sum_{q=1}^{Q}\left(P_{RX} \times M + \sum_{j=1}^{M} E_j^q\right) = P_{RX} \times M \times Q + \sum_{q=1}^{Q}\sum_{j=1}^{M} E_j^q \quad (1)$$

where E_j^q denotes the power consumption of the j^{th} STA during the q^{th} BP. Note that the first term in the objective function in (1) is the energy spent by all the STAs to listen to the TIMs, and it is constant. Hence, our objective is equivalent to minimizing the second term, which is the energy consumption determined by scheduling. If there is no packet scheduled for the j^{th} STA during the q^{th} BP, E_j^q is 0. Otherwise, it is the product of P_{RX} and the time duration (i.e., the number of timeslots) the j^{th} station has to remain awake during the q^{th} BP (i.e., from right after receiving the TIM to the time until the transmission of its last packet in this BP).

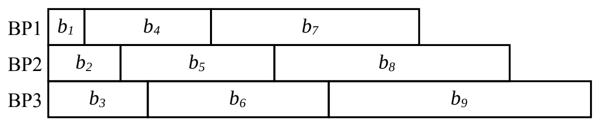
We call the group of packets that are queued and destined for one STA a *batch*. The length of the j^{th} batch, b_j, is defined as the number of packets in queue at the beginning of the first BP for STA j. Suppose that all the packets queued at the beginning of the first BP can be scheduled within one BP (i.e., $N<L$). In this case, the SPT (Shortest Processing Time) algorithm [8] is proven to be optimal. The basic idea of the SPT algorithm is as follows: group the packets that belong to a STA together as a batch first, sort the batches in the order of non-decreasing length, and then schedule the batches according to that order.

If the number of packets queued at the beginning of a BP is larger than L, then we need more than one BP to schedule all the packets. The minimum number of BPs needed to schedule all the packets currently queued is determined by $Q=\lceil N/L \rceil$, and we have to schedule all the packets within Q BPs according to our semi-work-conserving assumption. Note that for each BP, the number of packets to be scheduled can at most be L. We refer to this as the *timeslot restriction*. We also define the *length* of a schedule as the largest sum of length of batches scheduled in a BP. By definition, this length is at most L under the timeslot restriction.

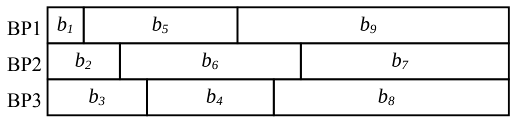
To derive some useful properties, we assume for the time being that while we use $Q=\lceil N/L \rceil$ number of BPs to schedule N packets, there is **no timeslot restriction**, i.e., the length of a schedule is not assumed to be restricted to be less than or equal to L. We will return to the general case with the timeslot restriction later. When there is no timeslot restriction, we introduce what we call the ESPT (Extended SPT) algorithm, which was first introduced as a variation of the SPT algorithm and proven to be optimal in [10] for the job-scheduling problem to minimize the mean flow time when jobs are assigned to multiple identical machines, where the flow time is defined as the duration from the time that a job is ready to the time that the processing of the last operation of the job is completed. Since our problem without timeslot restriction can be exactly transformed into this job-scheduling problem, we can prove that the ESPT algorithm is also optimal for our problem without timeslot restriction. We describe the ESPT algorithm as follows.

1. Sort the batches so that they are in non-decreasing order of lengths (i.e., $b_{[1]} \le b_{[2]} \le \ldots \le b_{[M]}$). Note that we will use square brackets *[.]* to denote the position in the order, i.e., the symbol *[1]* in the expression $b_{[1]}$ means the batch in first position in the order. Since we sort the batches in non-decreasing order, we have $b_{[k]} \le b_{[l]}$ if $k<l$.
2. Partition the set of batches ordered as above to obtain the sets R_r where
$$R_r = \left\{b_{[M-(r-1)Q]}, b_{[M-(r-1)Q-1]}, b_{[M-(r-1)Q-2]}, \ldots, b_{[M-(r-1)Q-(Q-1)]}\right\} \text{ and } r \in \{1,2,\ldots,\lceil M/Q \rceil\}. \quad (2)$$
We define this set as the set of batches of *rank r*, i.e., the set of rank 1 consists of the Q largest batches, the set of rank 2 consists of the Q largest batches among the remaining batches, and so on.
3. An ESPT schedule may now be obtained by assigning each of the batches of rank 1 to be the last of a different Q BP. Next, each of the batches of rank 2 is assigned to a different BP, and so on. This construction can be interpreted as a *Q-batches-at-a-time* assignment procedure.

Let us give an example. Suppose we have 9 PSM STAs that have $b_j=j$, j=1, 2,…, 9, and L=15. Then the total number of packets, N, is 45, and we have $Q=\lceil 45/15 \rceil = 3$. Hence, we have 3 sets of different ranks: $R_1=\{b_9, b_8, b_7\}$, $R_2=\{b_6, b_5, b_4\}$, and $R_3=\{b_3, b_2, b_1\}$, and in this case a power-consumption-optimal schedule using the ESPT algorithm can be obtained as shown in Figure 1(a). Note that there are 3! ways of assigning 3 batches of each rank to 3 BPs, and hence there are $(3!)^3$ different ESPT schedules that achieve the optimal power consumption (e.g., Figure 1(b) shows another optimal schedule that could have been found by the ESPT algorithm.). Likewise, if there are M STAs, and Q BPs are required to schedule all the packets, then the total number of optimal schedules obtained by the ESPT algorithm is $(Q!)^{\lfloor M/Q \rfloor}(M-\lfloor M/Q \rfloor * Q)!$. The only



(a) A power-consumption-optimal ESPT schedule



(b) Another power-consumption-optimal ESPT schedule that meets timeslot restriction

Figure 1. Examples of power-consumption-optimal ESPT schedules (In the figure, each row shows the order of the batch schedule from left to right for each BP; we do not care of the order of the BPs)

restriction is that to achieve the optimal solution, two batches that belong to the same rank cannot be assigned to the same BP. We call this the *rank restriction* rule.

NP-Completeness of our Problem in the General Case. Consider now the problem with timeslot restriction, which means that the length of a schedule S, L_S, must satisfy $L_S \le L$. As we have demonstrated in the last example, while the ESPT algorithm is very simple and efficient in finding power-consumption-optimal schedules when there is no timeslot restriction, the optimal schedules obtained have widely different schedule lengths. For example, the schedule shown in Figure 1(a) has a length of 18, which does not meet the timeslot restriction, while the schedule shown in Figure 1(b) has a length of 15. Hence, the first step we could do is to find out if there exists a solution among those $(Q!)^{\lfloor M/Q \rfloor}(M - \lfloor M/Q \rfloor * Q)!$ power-consumption-optimal schedules that satisfies $L_S \le L$. Unfortunately it is not easy to determine if an optimal solution to our problem is among those schedules or not. We prove by the following theorem that this problem is NP-complete.

Theorem 1. The problem to find a power-consumption-optimal schedule that satisfies $L_S \le L$ among the ESPT schedules without timeslot restriction is NP-hard.
Proof: An NP-hard set-partitioning problem can be reduced to this problem (see [11] for more details).█

Energy Efficient Semi-Work-Conserving (EES) Scheduler. Since our general problem turns out to be NP-hard, we propose a heuristic algorithm to solve it. We call our heuristic algorithm the EES (Energy Efficient Semi-work-conserving). It consists of two parts: first, in a heuristic way we try to find a schedule that satisfies $L_S \le L$ among those $(Q!)^{\lfloor M/Q \rfloor}(M - \lfloor M/Q \rfloor * Q)!$ power-consumption-optimal schedules without timeslot restriction. Second, if the heuristic is not able to find such a schedule, it performs batch splitting (i.e., a batch is sent over more than one BP). Note that batch splitting cannot improve the overall power consumption; hence, we perform batch splitting in a way that tries not to increase the total power consumption.

The basic idea is as follows. To find a schedule that satisfies $L_S \le L$ among the power-consumption-optimal schedules without timeslot restriction, we try to normalize the schedule length of each BP. To do this, for each batch we first compute the difference from the smallest batch among the batches of the same rank, i.e., the 'length difference' d_j for the batch b_j that is in rank r, is computed as $d_j = b_j - \min_k\{b_k;\ b_k \in R_r\}$. Then we partition the d_j's into Q BPs such that $\max\{\Sigma_{j\in BP1} d_j,\ \Sigma_{j\in BP2} d_j, \ldots,\ \Sigma_{j\in BPQ} d_j\}$ is minimized. In partitioning the d_j's, we sort them in non-increasing order and assign the largest one in order to the BP with the smallest sum of length differences. In addition, it is necessary to retain the original rank restriction so that each BP is assigned exactly one batch from each rank. The details of the EES algorithm are given below.

EES (Energy Efficient Semi-Work-Conserving) algorithm

Step 1: Group all the packets for a given station j in a batch b_j. If the total number of packets currently queued is less than or equal to L, then schedule the batches in non-decreasing order (SPT); done. Else, go to Step 2.
Step 2: Create the sets R_r's as in (2).
Step 3: For each batch $b_j \in R_r$, $\forall r$, calculate the length difference d_j as $d_j = b_j - \min_k\{b_k;\ b_k \in R_r\}$.
Step 4: Assign batches to the Q BPs according to the following rules.
- (i) Assign the batch with the largest length difference to the BP with the smallest sum of the length differences as long as the next rule is not violated.
- (ii) The batch $b_j \in R_r$ can only be assigned to BP l, if l has no other batch from R_r already assigned to it.
- (iii) If there is more than one BP that has the smallest sum of the length differences and that meets the rule (ii), then assign the batch to the BP with the smallest sum of lengths of batches, to break the tie.

Step 5: Once every batch is assigned, check if each BP has the sum of the lengths of its batches equal to at most L. If yes, go to Step 7. Else, go to Step 6.
Step 6: Consider the BPs that have the sum of the lengths of their batches greater than L.
- (i) Sort the batches in the BP in non-increasing order of length.

(ii) Reassign the batches in that order to that BP until the sum of lengths of the assigned batches is equal to *L*. Split a batch if it is needed. Add unassigned (possibly split) batches to a temporary set of batches *W*. Do this for all the BPs that have the sum of lengths of batches greater than *L*.

(iii) If $W=\emptyset$, then go to step 7. Otherwise, for the batches currently in *W* do the following.

a. Take the batch with the largest length in *W*.

b. Assign this batch, say b_d, to the BP, say B_i that has the smallest number of batches among the BPs that have the sum of lengths of batches less than *L*. If there is more than one BP with the same smallest number of batches, then try the one with the smallest sum of lengths of batches first, to break tie. If, by assigning b_d to B_i, the sum of lengths of batches for B_i, say $Z(B_i)$ becomes greater than *L*, then split the batch b_d into batches $b_{d(c)}$ and $b_{d(\sim c)}$. The split batches $b_{d(c)}$ and $b_{d(\sim c)}$ have their lengths $b_d-(Z(B_i)-L)$ and $Z(B_i)-L$ correspondingly, and $b_{d(c)}$ is assigned to B_i. Add the remaining batch $b_{d(\sim c)}$ to *W*. Go to iii).

Step 7: For each BP, schedule the batches in the SPT order; done.

We illustrate our heuristic by applying it to the same example we have shown in Figure 1. After partitioning the batches into the sets of ranks, we compute the length difference d_j for each batch in each rank. For example, for the batches of rank 1, we compute the length differences $d_9=b_9-b_7=2$, $d_8=b_8-b_7=1$, and $d_7=b_7-b_7=0$. Then, we sort the batches in the order of non-increasing length differences and start to assign each of them in order to the one of 3 BPs with the smallest sum of length differences. For example, we first assign b_9, b_6, and b_3 to each BP since the corresponding differences are equal to 2. The sum of the length differences for each BP is then updated to 2. Then b_8, b_5, and b_2, each of which has a length difference of 1, are assigned to the 3 BPs. When we assign b_8, we cannot assign it to BP1 since b_9, which is of the same rank as b_8, is already assigned to it. Thus, b_8 could only be assigned to BP2 or BP3. Since both have the sum of the length differences of 2, there is a tie. To break the tie between BP2 and BP3, we choose the BP with the smaller sum of the lengths of batches; hence, BP3 is chosen since it has the sum of the length of its batches as 3 while BP2 has 6. Next, to assign b_5, we choose BP1 since BP1 has the smallest sum of the length differences among the BPs that meet the rank-restriction rule. We continue to assign the batches in this way, and after assigning all the batches, we schedule the batches in each BP in non-decreasing order. This schedule corresponds to the one shown in Figure 1(b). Note that the length of this power-consumption-optimal schedule is 15, and this meets timeslot restriction. (If it had not, we start batch splitting as detailed in the algorithm)

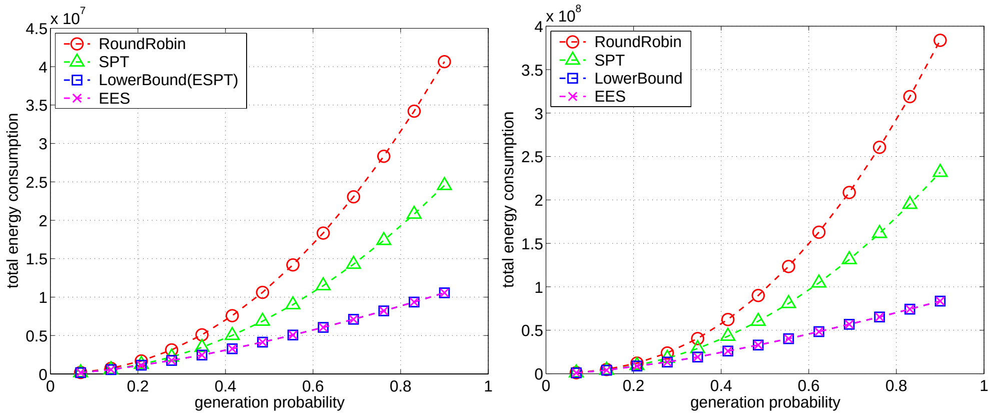
Simulation Results. We illustrate the performance of our EES algorithm by simulation. Since we assume the *static* case scenario, for each test case, the number of STAs (*M*) is fixed, the number of packets currently queued (*N*) is selected at the beginning of the first BP, and then the scheduling is done only for those *N* packets that are currently in the queue assuming that no more packets will come. To generate a large number of test cases, we generate the batches for the stations as follows. For each STA, we generate a batch of size *b* ($0\le b\le L+1$) with probability p_b:

$$p_b = \binom{L+1}{b} p^b (1-p)^{L+1-b} \quad (3)$$

where *p* is called the *generation probability*. Hence the average batch per STA is of length $p(L+1)$. We assume that all the STAs are independent and identically distributed (i.i.d.), and therefore $X=pM(L+1)$ packets on average are waiting to be scheduled at the beginning of the first BP.

For each test case, once the packets are generated, as described above, at the beginning of the first BP, the number of BPs required to send all the packets currently queued is calculated as $Q=\lceil N/L \rceil$, where *N* is the number of packets effectively generated, and the scheduler starts its schedule. Once the scheduler has scheduled all the packets in *Q* BPs, the total energy consumption is calculated, and the test case is done. We calculate the energy consumption for each test case by summing up the energy consumption of all the STAs. In calculating the energy consumption, we assume that every STA consumes exactly one unit of energy to receive a packet or to stay awake idle in a timeslot. For every BP, the energy consumption spent by all the STAs to listen to the TIM is added up to the total energy consumption.

We vary *p*, the generation probability, from 0 to 1, and for each value of *p*, we generate 10,000 test cases. For each test case, we generate a (possibly different) number of packets, create the schedule with EES, and compute the total energy consumption. We repeat this process for 20 different random seeds. Then for a given value of *p*, we get the total energy consumption for 200,000 test cases and take their average. Figure 2(a) shows the total average energy consumption as a function of *p* for EES as well as for other scheduling disciplines (Round Robin (RR) and SPT), for *M*=10 and *L*=20. Figure 2(b) shows results for *M*=50 and *L*=50. We have run



(a) When M=10 and L=20 (b) When M=50 and L=50

Figure 2. STATIC CASE: Total energy consumption comparison

extensive simulations for different *M*'s and *L*'s, and have seen the same trend as exhibited in these two figures. It is clear on these figures that EES outperforms both RR and SPT, especially for large *p*.

Another important aspect is how close the performance of our algorithm is to the optimal schedule, since our algorithm is a heuristic and therefore is not necessarily optimal. Since finding the optimal value is computationally burdensome, we instead compare the performance of our algorithm with that of the ESPT algorithm, which is free from the timeslot restriction. Note that the performance of the ESPT algorithm without timeslot restriction is a lower bound that can be achieved by the EES algorithm for some test cases. The performance of ESPT is also given in Figure 2. Note that the performance of the optimal solution to our problem is in between the performance of our EES algorithm and that of the ESPT algorithm, and hence by measuring the performance difference between our algorithm and ESPT, we get a sense of how close the performance of our EES algorithm is to that of the optimal solution. Hence, the results shown in Figure 2 confirm that our EES scheduler performs extremely well.

3.2 Dynamic Case

We formulate our scheduling problem for the dynamic case in this section. We assume that packets arrive continually over time, and the AP decides a schedule at the beginning of every BP based on the packets that are in the buffers. Our problem formulation is as follows: find an optimal scheduling discipline U^* that minimizes the average total power consumption of all the PSM STAs per BP, i.e.,

$$\text{minimize } \overline{E} = \lim_{P\to\infty} \frac{1}{P} \sum_{q=1}^{P} \sum_{j=1}^{M} E_j^q \tag{4}$$

In the q^{th} BP, a station *j* would spend at least one energy unit to receive the TIM and of at most *L+1* energy units if the AP has scheduled one of its packets to be sent in the last timeslot of the BP (i.e., $\forall j, q, \;\; 1 \le E_j^q \le L+1$). Hence the total energy (in units of energy) spent in a BP by a STA does not only depend upon the number of packets that have been scheduled by the AP for that station but also upon the position of the last packet scheduled for that station in the BP.

Based on what we have learned in the static case, we develop two heuristic scheduling disciplines and compare their performances with more traditional scheduling disciplines. The first of the two heuristic policies is a work-conserving scheduling discipline called LPTSPT (Longest Processing Time in selecting stations, Shortest Processing Time in scheduling packets). We show that without increasing the complexity of the algorithm compared to SPT (Shortest Processing Time), which is known to perform fairly well [7], LPTSPT performs better than or at least equal to SPT. The second is a non-work-conserving scheduling discipline called DEES (Dynamic Energy Efficient Semi-work-conserving), an extension to the dynamic case of the EES algorithm that we developed for the static case. We describe these policies in the next subsections and then evaluate their performances by simulation.

LPTSPT. From the static case in which we showed that in some special cases, the SPT algorithm was optimal, we have learned two important rules to schedule in a way to minimize energy consumption: 1) Allocating adjacent timeslots to the packets that belong to a STA (i.e., treating them as a batch). 2) Scheduling the batches within a BP in the order of non-decreasing length. What LPTSPT does further as compared to SPT is to reduce the number of STAs to be scheduled during a BP so as to minimize the IDLE mode energy consumption.

The basic idea is as follows. Assume that the number of packets queued at the beginning of a BP, say *N*, is larger than the number of timeslots to schedule during a BP, say *L* (i.e., $N>L$). Then, the scheduler needs to select *L* out of *N* packets to schedule during this BP. Note that any work conserving scheduler must schedule *L* packets. In this case, we have two decisions to take: 1) How to select *L* packets (in particular from which stations)? 2) After having selected *L* packets, how to schedule them? For the second decision, we already have an answer: schedule the batches in the order of non-decreasing length. Let us first discuss what SPT does. SPT would order the batches in non-decreasing order (i.e., $b_1 \le b_2 \le \ldots \le b_M$) and would index them such that STA_1 is the one with the smallest batch. SPT would then choose STAs 1 to *j* where *j* is chosen such that: $\sum_{i=1}^{j-1} b_i < L \le \sum_{i=1}^{j} b_i$, would if necessary truncate b_j and allocate timeslots starting with the station having the shortest batch (could be station *j* because its batch has been truncated).

For the first decision, LPTSPT does the following. While any work-conserving scheduler has to schedule exactly *L* packets (since $N>L$), to save more power it is key to reduce the number of STAs spending IDLE-mode energy. To this end, LPTSPT selects batches in the order of non-increasing length; i.e., it will select (assuming the same ordering as above) stations *M* to *j* where *j* is chosen such that: $\sum_{i=j+1}^{M} b_i < L \le \sum_{i=j}^{M} b_i$. Then if necessary LPTSPT would truncate batch b_j and would then allocate timeslots in the same way as SPT. When $N \le L$, LPTSPT does the same as SPT, since all the packets currently queued are to be scheduled. Both algorithms schedule the batches in the same order of non-decreasing length in this case.

We present below the details of our LPTSPT algorithm.

LPTSPT (Longest Processing Time in selecting stations, Shortest Processing Time in scheduling packets) algorithm

Step 1: Group all the packets for a given station *j* in a batch b_j.

Step 2: If the total number of packets currently queued is less than or equal to *L*, then schedule the batches in the order of non-decreasing length; done. Else, go to Step 3.

Step 3: Sort the batches in the order of non-increasing length. Take as many batches as necessary to get *L* packets or more. If necessary, truncate the smallest selected batch. For the selected batches, schedule the batches in the order of non-decreasing length; done.

Note that the LPTSPT algorithm involves at most one more sorting than the SPT algorithm, and hence the overall complexity remains $O(MlogM)$, which is the same as the SPT algorithm.

DEES. In the previous section, we have developed a heuristic non-work-conserving scheduling algorithm called EES for the static case. EES determines a schedule over *Q* BPs, the number of BPs required to send the *N* packets without putting any restriction on where the empty timeslot can be. By generalizing EES to the dynamic case, we allow our scheduler to not necessarily fill up the current BP even if $N>L$. This makes DEES a non-work-conserving scheduler, and this will impact the delay performance. However, by nature the PSM stations consider that saving energy is of prime importance and hence are prepared to trade off QoS parameters such as delay for energy savings. Hence, non-work-conserving schedulers are also worth considering.

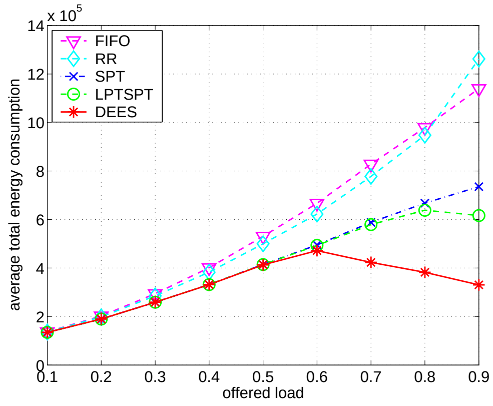
We now present the DEES algorithm. Before the beginning of a BP, DEES decides the schedule for all the packets currently queued as if it needed to schedule all of them during $Q=\lceil N/L \rceil$ BPs starting from the current BP (in fact it will redo the same thing before each BP). Hence, out of this schedule, only what has been decided for the first BP will be used. Note that Steps from 1 to 6 of the DEES algorithm are the same as in the EES algorithm, and the only difference is Step 7.

DEES (Dynamic Energy Efficient Semi-work-conserving) algorithm

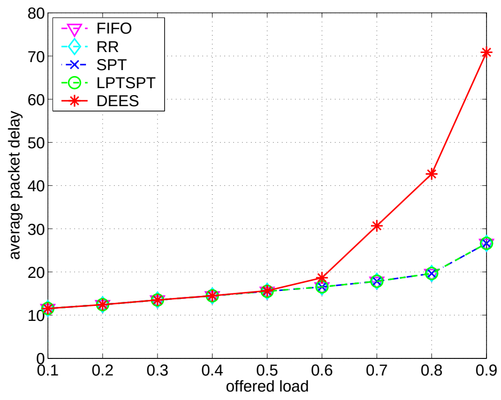
Step 1 ~ Step 6: Same as in the EES algorithm.

Step 7: Among the *Q* BPs that have been scheduled, choose the BP with the longest length. Schedule the batches in that BP in the SPT order; done.

The reason we adopt the strategy described in step 7 is as follows. At the end of step 6, we have allocated packets to the *Q* BPs. Only one out of these BP schedules will be used since the scheduler will start the process anew at the beginning of the next BP. There are several ways to select that BP schedule. In general, if we select the BP schedule with the least number of packets, then we should achieve lower energy consumption. However, this strategy could severely aggravate the delay performance and cause buffer overflows, i.e., it threatens the whole system stability. In contrast, choosing the BP schedule with the largest number of packets limits the impact on delay and on the system stability. For example, if the number of packets currently queued is $2L<N\le 3L$, then $Q=3$ and we have to choose one BP schedule among 3 BP schedules. Since we choose the one with the largest number of packets, the number of packets that will be served is at least $2L/3$. Likewise, if *(Q-*



(a) Total energy consumption



(b) Average packet delay

Figure 3. DYNAMIC CASE: Total energy consumption and average packet delay versus offered load (when M=10, L=20)

$1)L<N\leq QL$, by choosing the one with the largest number of packets, the scheduler serves at least $(Q-1)L/Q$ packets in a BP. As the queue length increases, Q increases, and the number of packets to be scheduled in a BP (i.e., $(Q-1)L/Q$) increases as well. Hence if the system becomes backlogged, the scheduler becomes less and less non-work-conserving.

Simulation Results. We show the performances of the LPTSPT and the DEES algorithms by simulation. We generate the traffic for each STA using a binomial process; i.e., a packet is generated in a timeslot for a given STA with probability p, and all the STAs are assumed to be i.i.d. We assume that the number of STAs (M) is fixed throughout a simulation run. We assume the system to be gated; however, the packets arrive continually over time. We simulate 20 different cases per value of p, each case being generated from different random seeds. Each case comprises 200,000 timeslots. We calculate the total energy consumption by summing up the energy consumption of all the STAs over the 200,000 timeslots for each case, and then averaging for the 20 cases. We use the total energy consumption and the average packet delay as performance measures.

We compare the performances of our algorithms with those of three other scheduling disciplines: FIFO, RR, and SPT. Figure 3(a) and (b) show the performance comparison in terms of the total energy consumption and the average packet delay as a function of the offered load $\rho=pM$ when M=10 and L=20. Let us focus on the performance of LPTSPT first. As we can see from Figure 3(a), LPTSPT and SPT outperform FIFO and RR significantly in total energy consumption. In addition, LPTSPT outperforms SPT especially when the offered load is high. Note that when the offered load is low (up to around 0.6), LPTSPT performs at least equal to or slightly better than SPT. The reason is that in most BPs, the number of packets queued is less than or equal to L, so that all the packets can be scheduled within a BP. Delay performances for all the work-conserving schedulers including the LPTSPT are the same as expected, and they are shown in Figure 3(b). From these results, we can conclude that if delay is the main issue, then LPTSPT is a better discipline than the other work-conserving schedulers introduced in this paper.

As for the DEES algorithm, it significantly outperforms other disciplines studied in this paper in total energy consumption, while trading it off for some amount of delay. When the offered load is lower than 0.6, the performance of the DEES algorithm in both the total energy consumption and the average packet delay is very close to that of LPTSPT. Again, this is because all the packets queued at the beginning of a BP are scheduled during the BP in most cases, and in this case, DEES does exactly the same as LPTSPT. As the offered load increases, we can achieve significant energy savings by using DEES. We have run extensive simulations for different M's and L's, and have seen the same trend.

At the same time, one may worry about the delay performance since it is exponentially increasing as offered load goes close to 1. However, when the offered load is around 0.6~0.8, which is a typical network operating point, DEES saves up to 40% of energy relative to LPTSPT, which is the best discipline among the work-conserving disciplines used in our simulation, at the cost of increasing the average packet delay by one more BP. Recall that since energy saving is of prime importance for energy-deprived mobile stations, DEES may indeed be an attractive option.

4. Conclusions

We have modeled a generic wireless system that offers a PSM to its users. We have formulated the problem of minimizing the total power consumption of all the portable stations that are in PSM as a downlink transmission-scheduling problem. We have proposed several heuristic algorithms under both static and dynamic assumptions and evaluated their performance by simulation. Our results confirm that the EES algorithm (in the static case) and the DEES algorithm (in the dynamic case) yield significant energy savings when compared to FIFO, Round Robin, and SPT.

Acknowledgement

This work was supported in part by the Indiana Twenty-First Century Fund through the Indiana Center for Wireless Communication and Networking and the NSF grant 0087266, and by the NSF through grants ECS-0098089, ANI-0099137, and ANI-0207892.

References

1. J. Flinn and M. Satyanarayanan, "*Energy-aware adaptation for mobile applications*," in Symposium on Operating Systems Principles, December 1999, pp. 48–63.
2. M. Stemm and R. H. Katz, "*Measuring and reducing energy consumption of network interfaces in handheld devices*," IEICE Transactions on Fundamentals of Electronics, Communications, and Computer Science, Aug. 1997, pp. 1125-1131.
3. R. Kravets and P. Krishnan, "*Power management techniques for mobile communication*," Proc. of Mobicom 98, Dallas, TX, October, 1998, pp. 157-168.
4. L. M. Feeney and M. Nilsson, "*Investigating the energy consumption of a wireless network interface in an ad hoc networking environment*," Proc. Of IEEE Infocom 2001, Vol. 3, pp. 1548 –1557.
5. J.-P. Ebert, B. Burns, and A. Wolisz, "*A trace-based approach for determining the energy consumption of a WLAN network interface*," In proc. of European Wireless 2002, Florence, Italy, Feb. 2002, pp. 230-236.
6. The editors of IEEE 802.11, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification*, 1999.
7. J. A. Stine and G. de Veciana, "*Energy efficiency of centrally controlled transmission of fixed size packets*," Proc. of WCNC 2000, pp. 765-770.
8. W. E. Smith, "*Various optimizers for single-state production*," Nav. Res. Log. Quart. 3, No. 1, March 1956.
9. R. Krashinsky and H. Balakrishnan, "*Minimizing energy for wireless web access with bounded slowdown*," Proc. of Mobicom'02, Sep. 23-28, 2002, Atlanta, GA, pp. 1-12.
10. R. W. Conway, W. L. Maxwell, and L. W. Miller, *Theory of Scheduling,* Addison-Wesley, 1967.
11. R. M. Karp, "*Reducibility among combinatorial problems*," in R. E. Miller and J. W. Thatcher (eds), Complexity of computer computations, Plenum Press, New York, 1972, pp. 85-103.