An Adaptive “Sleep” Algorithm for Efficient Power Management in WLANs

Mahasweta Sarkar and Rene L Cruz
Department of Electrical and Computer Engineering
University of California, San Diego
La Jolla, CA 92093
msarkar,cruz@cwc.ucsd.edu

Abstract—In an effort to conserve energy, standard protocols for WLANs [1] have the provision for wireless nodes to “sleep” periodically. In this paper we consider the problem of optimizing the timing and duration of sleep states of wireless nodes with the objective of minimizing power consumption with respect to a QoS constraint namely average packet delay. We study downlink traffic only. First, using a dynamic programming formulation coupled with a duality argument, we solve the optimization problem numerically for a two-node system served by a single wired transmitter (Access Point). Then, using the solutions from the numerical calculations as a guideline we coin an adaptive algorithm for scheduling the sleep time and duration of multiple wireless nodes operating in an infrastructure mode served by a single Access Point. Simulation results show that the power efficiency of our algorithm is comparable to the results obtained from the dynamic programming formulation.

Keywords- power management, WLAN, sleep-wakeup policy, dynamic programming;

I. INTRODUCTION

We define a sleeping node as a wireless node that has switched off its receiving, transmitting and channel sensing circuitry. The main challenge of the sleep mechanism lies in the wireless node’s inability to “wake up” as soon as a packet arrives for it during its sleep state. Prolonged sleeping can result in delay penalties for the packets that accumulate for the wireless node at the Access Point (or similar) but it also entails substantial power savings for the wireless node. Thus in this paper we seek an “optimal” policy that enables wireless nodes to save maximum power by sleeping for the longest duration such that they do not violate their packet delay constraints. Unlike [3][4] we calculate the “optimal” sleep duration as a function of average packet delay tolerance. The wireless node is then allowed to sleep for the pre-determined “optimal” number of slots after which it has to wake up to receive its packets (if any) from the Access Point (AP).

We first consider a system comprising of two wireless nodes served by a single transmitter, operating over a static and perfect channel. We formulate this as an optimization problem and solve it numerically using dynamic programming (DP). Using the results obtained from the DP as a guideline we then design a realistic, scalable, low complexity and adaptive “sleep” scheduling algorithm for multiple wireless nodes served by a single AP operating in the infrastructure mode in a WLAN. Our algorithm predicts the sleep duration of each user based on its packet delay constraint. We consider a round robin and a non-round robin service order. Simulation results show that in terms of power efficiency, our algorithm performs close to the optimal results derived from the dynamic programming formulation.

II. SYSTEM MODEL

Time is divided into equal length intervals called slots, which are indexed by integers. Packets arrive randomly to the transmitter, to be transmitted to the wireless nodes across a static and perfect channel.

We assume that packet arrivals at the transmitter are memoryless and modeled by a Bernoulli process. Let p be the probability of a packet arriving in a slot. We also assume that a packet always arrives at the beginning of a slot. It can be served in the same slot it arrived or during a later slot. We define \( P_s \) to be the power consumed by the wireless node in slot ‘n’, \( a_n \) to be a random variable that represents the number of packets that arrive at the transmitter at the beginning of slot ‘n’, \( u_n \) to be the number of packets transmitted to the wireless node in slot ‘n’ and \( x_n \) to be the number of packets queued at the transmitter at the end of slot ‘n’, also called the backlog of slot ‘n’. The backlog process \( x_n \), therefore, satisfies the recursion

\[
x_{n+1} = x_n - a_n + u_n
\]

The wireless nodes can be in one of the two following states – “Sleep” or “Awake”. Let \( P_s \) denote the power required per slot while in “Sleep” state and \( P_a \) denote the power required per slot while in “Awake” state. In addition, let \( P_{sa} \) denote the power required for switching from Awake to Sleep state and \( P_{as} \) be the power required for switching from Sleep to Awake state. We assume \( P_{sa} \gg P_s \) [2].

In order to distinguish between the two nodes, we use the node-id along with the different variable names. For instance, \( P_{1s} \) and \( P_{2s} \) denote the power consumed by wireless nodes 1 and 2 respectively, in slot ‘n’.

We characterize the system state in each slot by the quadruple \((x_{1n}, r_{1n}, x_{2n}, r_{2n})\), where...
\( x_{1n} \) and \( x_{2n} \) = backlog for node 1 and node 2 respectively at the end of slot ‘n’, and
\( r_{1n} \) and \( r_{2n} \) = sleep state of node 1 and node 2 respectively during slot ‘n’
where \( r_{1n} \) and \( r_{2n} \) can independently take the following values:

0; \{node awake during slot ‘n’\}
1; \{node asleep during slot ‘n’ but will be awake in slot ‘n+1’\}
k; \{node asleep during slot ‘n’, ‘n+1’…..‘n+k-1’ but will be awake during slot ‘n+k’\}

III. THE OPTIMIZATION PROBLEM

We require that the average delay suffered by data packets at the transmitter be no more than \( D_1 \) and \( D_2 \) respectively for nodes 1 and 2. According to Little’s result, assuming that the backlog process is stationary, the average delay for the two nodes are given by \( x_1* \) and \( x_2* \). Thus, we require that, \( x_1* \leq pD_1 \) and \( x_2* \leq pD_2 \). We approach the problem, by considering it as an optimal control problem over a finite time horizon [7] say over slots 0,1,2,….,N-1. In order to reflect the average delay constraint, we consider policies \( \phi \), such that

\[
E\left[\sum_{n=0}^{N-1}x_{1n}\right] \leq pD_1N \quad \text{and} \quad E\left[\sum_{n=0}^{N-1}x_{2n}\right] \leq pD_2N
\]

(2)

Let \( \Lambda \) be the set of all such policies.

We consider the following optimization problem:

\[
\min_{\phi} \left\{ E\left[\sum_{n=0}^{N-1}(P_{1n} + P_{2n})\right] \right\} \quad \text{subject to} \quad \phi \in \Lambda
\]

(3)

We use a similar approach as in [6] to solve the above equation. Using a dynamic programming formulation coupled with a duality argument, we numerically solve the above optimization problem.

We present some representative results related to specific scenarios in a graphical representation. The results represents the scenarios where at time \( t=0 \) both nodes have 0 packets buffered for them at the AP and both nodes are in the “Awake” state. We assume \( p=0.1, P_{a1}=0.0001, P_{as}=0.01, P_{sa}=0.001 \) to obtain the results displayed in Figure 1. We observe that power consumption of the system is inversely proportional to the average packet delay tolerance of the wireless nodes. The graph shows that for a higher average packet delay tolerance (denoted by the X-axis), the power consumed per slot (denoted by the Y-axis) is lower. This can be explained by the fact that a higher average packet delay tolerance enables the wireless nodes to be in the sleep state for longer durations. This leads to lower average power consumption by the nodes.

IV. MULTIPLE-NODE ADAPTIVE SLEEP ALGORITHM

Under various initial buffer occupancies \((x_{10}, x_{20})\) and initial sleep states \((r_{10}, r_{20})\) of the two nodes, the DP generated different optimal sleep schedules and sleep durations for each of the two nodes, which minimized the overall system power cost. We studied the different sleep policies generated by the DP under different initial conditions of the two nodes and used them as a guideline to coin our own algorithm for minimizing the power consumption of multiple wireless nodes subject to packet delay constraints. We consider a model where the wireless nodes are served by a single wired transmitter (or AP). As before, we study the downlink traffic only. We also assume static and perfect channel conditions.

We consider two different service orders – a round robin scheme and a non-round robin scheme. For the most part, the algorithm is same for both the schemes. We now describe the algorithm in details.

Let the wireless nodes be denoted by ‘a’, ‘b’,…., ‘n’. Let the packet delay constraints of the nodes be \( D_a, D_b, \ldots, D_n \). At time \( t=0 \), we assume that there are no packets buffered for any node. We randomly assign sleep durations \( s_a, s_b, \ldots, s_n \) to all the nodes respectively.

A. Service Order

For the round robin scheme, we serve the nodes in a fixed, predetermined order, say node ‘a’, followed by node ‘b’ and so on. For the non-round robin scheme, we arrange the nodes in ascending order of their sleep durations at the beginning of every slot [note that the state 0 denotes the active/awake state]. Service is rendered (if possible) to the node that has the least sleep duration namely 0. If all nodes are asleep, we wait for a node to wake up before service can be rendered.

B. Calculating the next sleep duration

When a node wakes up, there are two tasks that has to be performed. First, the node has to receive the packets that have
been buffered for it at the AP during its sleep state. After it has received all the data packets that have been buffered for itself at the AP, the node shall revert back to the sleep state. The second task comprises of determining the duration of this sleep state. Our adaptive algorithm calculates this sleep duration based on the delay suffered by the buffered data packets (the ones that the node received during the current cycle). The details of adaptively determining the sleep duration are as follows.

Let us assume that node ‘a’ was the first to wake up after its sleep duration of $s_a$ slots.

1> After $s_a$ slots, we calculate the exponential moving time average delay of the packets buffered for node ‘a’. We call this the calculated delay = $CDa$

2> Let ( $Da – CDa$ ) = $\delta a$

$\delta a$ can be equal to 0 , greater than 0 or less than 0. Let ‘k’ be a parameter. k can take positive or negative values. We assign ($s_a + k(\delta a)^2$) to be the next sleep duration for node ‘a’.

Generally speaking, we follow the following rule for calculating the sleep duration in the next cycle:

**Next Sleep Duration = Current Sleep Duration + k*(\delta)^2**

The aim is to calculate the sleep duration of a node in such a way so that the node expends minimal energy in staying awake while not violating its delay constraint. Our algorithm “learns” to predict this “optimal” sleep duration by a trial and error mechanism. We observe that during the “learning phase”, a node might either wake up “early” or “on-time”. We define the notion of a node waking up “early” for the two service schemes as follows. In the non-round robin scheme, if a node’s - say node ‘a’ - sleep duration was predicted in such a way that node ‘a’ wakes up while another node ‘x’ is in service and node ‘x’ still has packets at the AP for it to receive, then we say that node ‘a’ woke up “early”. In the round robin scheme, if a node’s sleep duration was predicted in such a way that it woke up before its due service turn, we say that the node woke up “early”. We also consider two other cases where a> a node does not wake up “early” (i.e its sleep duration was predicted such that it did not wake up while another node was in service or before its own service turn) – i.e we say, it woke up “on-time” but does not have packets to receive and b> a node wakes up “on-time” and have packets to receive. Table I explains our action and the magnitude of ‘k’ that we choose under the above scenarios.

<table>
<thead>
<tr>
<th>$\delta a$</th>
<th>Early</th>
<th>On time but 0 pkts</th>
<th>On time + pkts</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\delta a &gt; 0$</td>
<td>Lengthen sleep</td>
<td>$k &gt; 0$</td>
<td>Lengthen sleep</td>
</tr>
<tr>
<td>$\delta a &lt; 0$</td>
<td>Shorten sleep</td>
<td>$k &lt; 0$</td>
<td>Shorten sleep</td>
</tr>
</tbody>
</table>

**TABLE I. Sleep Decisions**

**C. Service Policy**

We now discuss the protocols that we follow while serving packets to a particular node. Packets are served in a “gated” fashion. Let ‘a’ be the node that has just woke up and is ready to receive its packets from the AP (or the transmitter). By “gated” service we mean that only the packets that have accumulated in the buffer at the transmitter during the sleep duration of node ‘a’ gets served. The packets that arrive while node ‘a’ is receiving service are buffered at the AP and served in the next cycle. We occasionally switch to exhaustive service as well. The condition under which we switch to an exhaustive service policy is explained later.

Let the next node awaiting service be ‘b’.

a) While node ‘a’’s packets are being served it might be the case that node ‘b’, wakes up. Under such circumstances, we finish serving node ‘a’’s packets in the gated fashion and put it to sleep. We then start serving node ‘b’.

b) It might also be the case, that node ‘a’’s packets will be completely served in the gated fashion and yet node ‘b’ would not have woken up. In that case, we revert to an exhaustive service policy for node ‘a’. If node ‘b’ does not wake up even after node ‘a’’s buffer has been emptied we put node ‘a’ to sleep for its stipulated number of slots and wait for node ‘b’ to wake up. Hence there might be a few slots for which the transmitter might remain idle.

**V. RESULTS OF THE ALGORITHM**

We wrote a computer program in the C++ programming language to implement this algorithm. Our algorithm has the capability to schedule the sleep time and duration of several wireless nodes operating in an infrastructure mode in a WLAN. However, in order to compare our results with the optimal solution as derived from the dynamic programming (DP) formulation, we specifically consider a two-node wireless node system since the dynamic programming formulation is modeled after a two-node system as well. Figure 2 shows the delay versus power consumption tradeoff for the three schemes. The DP scheme offers the best delay-power tradeoff. This was expected, since this scheme has perfect knowledge of the packet arrival rate for all the users. The non round robin scheme consumes slightly more total power per slot for the same packet delay than the optimal DP case. This is not unusual given that this algorithm executes a “trial and error” process to reach the “best” sleep policy. It also does not have any knowledge about the packet arrival rate unlike the DP formulation. The round robin scheme consumes more power than the non-round robin scheme as well as the DP scheme for the same packet delay. This is attributed to the fact that unlike the non-round-robin scheme service order is fixed here and independent of individual packet delay constraint. Hence a node which might have a smaller packet delay and hence a shorter sleep duration might be scheduled to be served after a
node which has a higher delay tolerance and hence a longer sleep duration. In this case, the previous node spends a lot of energy staying awake and awaiting service.

However, our algorithm has a number of merits over the DP that out-weigh the slight loss in performance. First of all, our algorithm is very scalable. It can schedule the sleep time and duration of several wireless nodes. Implementing the DP formulation on a two node system itself was cumbersome enough. The DP formulation requires large number of state spaces to be saved in order to make the optimal decision. As the number of users grow, the state space grows exponentially and even for as low as three users, it becomes quite unmanageable to handle the state space. This complexity is not encountered in our algorithm. Our algorithm is very easy to implement and requires no knowledge of the packet arrival rate. This makes it more practical than the DP formulation.

In this paper we have looked at the problem of optimizing the timing and duration of sleep states of wireless nodes, with the objective of minimizing power consumption with respect to an average packet delay constraint. First, we have considered a model with two wireless nodes being served by a single wired Access Point or transmitter. We formulated the above problem as an optimization problem and solved it numerically using dynamic programming for the two-node scenario. We then used these optimal scheduling schemes as a guideline to devise our own adaptive, robust and scalable algorithm. Our algorithm schedules the sleep time and predicts the sleep duration of multiple wireless nodes being served by a single wired Access Point, such that they expend minimal energy while still complying to their packet delay constraints. We considered two different service schemes – a round robin and a non-round robin scheme. We compared the power-delay tradeoff (we considered a two-node scenario for comparison purposes) of both the schemes of our algorithm with the optimal results derived from the DP formulation. The graphs show that our algorithm performs efficiently and close to the optimal solution in terms of power consumption.

VI. CONCLUSION

REFERENCES


