UNIVERSITY OF CALIFORNIA, SAN DIEGO

MAC Layer Power Management Schemes for Efficient Energy-Delay Tradeoffs in Wireless Local Area Networks

A dissertation submitted in partial satisfaction of the

requirements for the degree Doctor of Philosophy

in

Electrical and Computer Engineering (Computer Engineering)

by

Mahasweta Sarkar

Committee in charge:

Professor Rene L. Cruz, Chair
Professor Anthony Acampora
Professor Ramesh Rao
Professor Alex C. Snoeren
Professor Geoffrey Voelker

2006

The dissertation of Mahasweta Sarkar is approved, and

it is acceptable in quality and form for publication on

microfilm:

Chair

University of California, San Diego

2006

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

This dissertation would not have been possible without the help of many people. Above all, I would like to thank my adviser, Dr. Rene Cruz. None of my dissertation research would have been possible without his guidance, unflinching support, remarkably perceptive suggestions and his boundless passion for research. My interactions with him have taught me a myriad different things ranging from problem solving to interpersonal communications. His numerous brilliant ideas have paved the way for my research work and I am truly indebted to him for that.

I would also like to thank faculty members and students involved in the Adaptive Systems group especially Dr. Ramesh Rao for introducing me to the big and beautiful world of 802.11. To Vikram, Pavan, Ashay and Arvind I owe the largest debt. I have immensely benefited from their feedback, suggestions and guidance especially during the initial years of my research at UCSD.

The five years that I spent at UCSD will be the most memorable years in my entire student life and I single handedly owe it to all my friends – Manjiri, Prasad, NP, Patrick, Kameswari, Jenny, Naomi, Ranjita, Preeti, Kanishka, Krishna, Pani, Shoubhik and above all Mrunal.

I will be forever grateful to Sreerupa, Ankhi and Pallavi for providing me with the emotional and mental support whenever my world seemed to be falling apart. To my sister Tukai and my parents I owe a world of gratitude for always being there for me in every possible way, especially to my Mom for being my best friend in every sense of the term. Maa - I owe everything I have ever achieved in life to you. To Aarjish – you are my sunshine. Last but definitely not the least, Sandip - thank you for everything. I doubt I could do justice to the difference you have made to my life by even attempting to list them and though I don’t say it often enough, I am so glad to have you in my life.

Finally, I would like to dedicate this dissertation to Sandip, Maa and Baba.

VITA

September 1975	Born, Kolkata, India.
1996	B.Sc (Hons), Chemistry, Calcutta University, India.
2000	B.S (Summa Cum Laude), Computer Science, San Diego State University, San Diego.
2006	Ph.D., Electrical and Computer Engineering (Computer Engineering), University of California, San Diego.

PUBLICATIONS

Mahasweta Sarkar and Rene Cruz, "Analysis of Power Management for Energy and Delay Trade-off in a WLAN" *Conference on Information Sciences and Systems (CISS), Princeton, March 2004*.

Mahasweta Sarkar and Rene Cruz, "An Adaptive Sleep Algorithm for Efficient Power Management in WLANs" *IEEE 61st Semiannual Vehicular Technology Conference, Stockholm, Sweden, May 2005.*

Mahasweta Sarkar and Rene Cruz, "A MAC Layer Dynamic Power Management Scheme for Multiple Users in a WLAN" *submitted to International Wireless Communications and Mobile Computing Conference (IWCMC), Vancouver, Canada, July 2006.*

Mahasweta Sarkar and Rene Cruz, "A MAC Layer Power Management Scheme for Efficient Energy Delay Tradeoff in a WLAN" *submitted to Journal of Computer Communications, 2005.*

ABSTRACT OF THE DISSERTATION

MAC Layer Power Management Schemes for Efficient Energy-Delay Tradeoffs in Wireless Local Area Networks

by

Mahasweta Sarkar
Doctor of Philosophy in Electrical and Computer Engineering
(Computer Engineering)
University of California, San Diego, 2006
Professor Rene. L Cruz, Chair

In order to minimize power consumption and thereby prolong the system lifetime of battery powered wireless devices, it makes sense for such devices to transit to a very low power "Sleep" state when they are not communicating with their peers.

However the main challenge of the sleep mechanism lies in the wireless nodes inability to "wake up" as soon as a packet arrives for it during its sleep state. This leads to an obvious tradeoff between power saving and packet delay. In this dissertation we are interested in the problem of optimizing the timing and duration of sleep states of wireless nodes in an infrastructure WLAN scenario with the objective of minimizing average overall system power consumption with respect to a QoS constraint. The QoS parameter we have focused on is average packet delay.

We first considered a simple model comprising of a single transmitter and a single receiver. We formulated this as an optimization problem and solved it numerically using

dynamic programming. We were able to derive closed form expressions for the optimal sleep duration for a given packet delay, as well as the associated minimal rate of power consumption. We extended the model to a multi-user system. Results indicated that the optimal policy for a specific node was a function of its buffer length as well as the sleep states of the other nodes in the system.

We next considered the problem of scheduling multiple streams with either same or different packet delay constraints. We proposed an adaptive sleep-scheduling algorithm based on a heuristic derived from the results observed in the dynamic programming formulation. Our algorithm has three different sleep scheduling schemes – Round Robin scheme, Shortest Sleep First scheme and the Steep Descent method.

We compare and contrast each of these schemes to the theoretical lower bound that we have previously computed by the method of dynamic programming. Results show that our algorithm performs favorably when compared to the lower bound. Further, in order to evaluate the performance of the Steep Descent method we construct a simplified simulation of the 802.11 sleep scheduling algorithm. Our simulation results show a substantial improvement in the performance of our new sleep scheduling policy when compared to the static sleep schedule of 802.11 especially in scenarios where traffic conditions change slowly over time which we model as a three state Markov process.

1 Introduction

1.1 Overview

The dream of ubiquitous and universal information access is fast becoming a reality. Cell phones with Internet browsing capabilities, Bluetooth connected Personal Digital Assistants (PDAs), research on wireless sensor networks by both academia and industry; Wireless Local Area Networks (WLANs) in coffee shops, restaurants, airports and hotels are on the rise.

For convenience sake, these wireless devices have to be portable and tiny yet powerful. To conform to the portability criteria these devices have to operate over wireless links. This limits the energy supply of such devices to built-in batteries. Unfortunately, battery technology is not progressing at the pace silicon processing technology is, but is expected to improve by a modest 20% over the next 4 years or so [1]. Recent trends in research and commercial efforts have thus rightfully shifted their emphasis from ever-increasing throughput alone to issues related to power efficiency. Energy and power consumption are becoming more formidable and important constraints and are being tackled at all levels of design from architecture and hardware to protocols and algorithms [2][3][27][29].

1.2 Where is the Bottleneck?

In order to develop an energy efficient system, we first have to identify the main components that form the bottleneck in terms of energy consumption. Contrary to popular belief, the communication functionalities predominate over the computation

functionality of most wireless devices [4]. In addition, the relative importance of the communication energy is expected to increase in the future [4]. While the energy consumption of digital circuits and processors are rapidly decreasing due to Moore's law and ingenious design techniques, the communication subsystem does not follow this trend. Wireless communications are therefore expected to become a bottleneck in terms of energy consumption and inevitably system viability. This has given rise to a variety of power management techniques at every level in system design. At one end of the spectrum, radio designers have created very efficient radio architectures [30], implementations and control algorithms while at the other end, network and application designers have proposed new protocols and algorithms that focus on energy efficiency for tasks such as routing [9][28], medium access control [26][31], source coding, error coding [38][39], encryption etc.

1.3 Related Work

In the recent past power management for large classes of applications have been an active area of research. The oldest and most straightforward technique to reduce the energy consumption of a system is to shut down unused parts. Shutdown-based power management has been explored for hard disks [32][33], displays, and communication modules [34][35][36], amongst others. For processors, it has been incorporated in the kernel of Real-Time Operating Systems (RTOS) [18]. More information can be found, for example, in the comprehensive overviews published by Benini *et al.* [2], and Lorch and Smith [37].

Chip-level power management features have been implemented in mainstream commercial microprocessors [11][12]. Techniques for automatic synthesis of chip-level power management logic are surveyed in [13].

In general, power management policies can be classified into predictive, adaptive and stochastic schemes. At the system level, the most common power management policy is the predictive scheme that uses past history of the workload to predict future idle periods. The goal is to predict when the idle period will be "long enough", so that the component can be placed into low power state. The simplest form of predictive technique is the timeout policy implemented in most operating systems. Timeouts assume that the component is very likely to remain idle if it has already been idle for a certain timeout period. This policy however wastes power while waiting for the timeout to expire [14][15].

Predictive policies [16][17][18] improve upon the timeout scheme by forcing the transition to a low power state as soon as a component becomes idle if the predictor estimates that the idle period will last long enough. An incorrect estimate can cause both performance and energy penalties. The distribution of idle and busy periods for an interactive terminal is represented as a time series in [19] and approximated with a least square regression model. The regression model is used for predicting the duration of future idle periods.

In [20], an improvement over the prediction algorithm of [19] is presented where idleness prediction is based on a weighted sum of the duration of past idle periods with

geometrically decaying weights. The policy is augmented by a technique that reduces the likelihood of multiple mis-predictions.

In contrast, stochastic control approaches formulates policy optimization as an optimization problem under uncertainty. Stochastic models use distributions to describe the times between arrivals of user requests (inter-arrival times), the length of time it takes for a device to service a user's request and the time it takes for a device to transition between its power states. The system model for stochastic optimization can be described either with just memoryless distributions (exponential or geometric) [21][22] or with general distributions [23][24]. Power management policies can also be classified into two categories by the manner in which decisions are made: discrete time [21][22] or event driven [23][24]. In addition, policies can be stationary (the same policy applies at any point in time) or non stationary (the policy changes over time). All stochastic approaches except for the discrete adaptive approach presented in [22] are stationary. The optimality of stochastic approaches depends on the accuracy of the system model and the algorithm used to compute the solution. In both the discrete and the event-driven approaches optimality of the algorithm can be guaranteed since the underlying theoretical model is based on Markov chains. Approaches based on discrete time setting [21][22] require policy evaluation even in the low power state thus wasting energy. On the other hand, event driven models based on exponential distributions [25] show little or no power savings when implemented in real systems since the exponential model does not describe well the request inter-arrival times [23][24].

1.4 Thesis Contribution

In this dissertation we address the issue of the inevitable tradeoff between power management and packet delay for wireless devices. We focus at the MAC layer of the protocol stack and study the "sleep" mechanism of wireless data devices. We observed that battery powered wireless devices (e.g. as in 802.11 based WLANS, sensor networks, ad hoc networks, RFIDs, paging systems) often undergo dormant phases when they do not need to be communicating with each other or with a central base station. Since these nodes are battery powered, they are severely energy constrained. Hence in order to save energy and thereby prolong the system lifetime, it makes sense for these nodes to transit to a "sleep" state when they are not communicating with their peers. Studies in [9] have shown that being in an idle state (instead of being in the sleep state, during the non communication phases) entails energy consumption that is comparable to being in a receiving or transmitting state. A WaveLAN card running at 11 Mbps consumes about 15 times more energy in the idle state than when the card is in sleep state. Table 1.1 shows the energy consumptions of a TR1000 radio operating under different modes. The numbers in the table further emphasizes the benefits of operating in the sleep state with regards to power savings.

Table 1.1: Power Consumption of the TR1000 radio in different modes

Transmit (Tx)	Receive (Rx)	Idle	Sleep
14.88 mW	12.5mW	12.36 mW	0.016 mW

We define a sleeping node as a wireless node that has switched off its receiving, transmitting and channel sensing circuitry. Thus, while in the sleep state the wireless node has no packet processing capabilities. However since the major power hungry circuitries are turned off during the sleep state, the wireless node consumes very little power while in this state. If a wireless node remains in the sleep state for too long it can result in delay penalties for the packets that accumulate for it at the central node (Access Point or similar). On the other hand, it can save substantial power for itself if it remains in the sleep state for longer durations. Thus an "optimal" policy may be desired which enables the wireless node to save maximum power by sleeping for the longest duration while still not violating the system's packet delay constraint. In this dissertation we are interested in the problem of optimizing the timing and duration of sleep states of a wireless node with the objective of minimizing power consumption with respect to a QoS constraint. The QoS parameter we have focused on is average packet delay.

The main challenge of the sleep mechanism lies in the wireless nodes incapability to "wake up" as soon as a packet arrives for it during its sleep state. Various protocols have been suggested to by-pass this problem [4][8][10]. Most of these protocols utilize a separate wakeup radio that essentially fulfils the role of an out-of-band paging channel. This radio periodically listens for wakeup beacons and could be designed to be less power hungry than the main data radio. In this thesis, we take a different view point in addressing the above problem. We try to 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.

To help gain a better understanding of the general problem, we first consider a simple model comprising of a single transmitter and a single receiver. The receiver is the wireless node whose mode we wish to control. The transmitter can give commands to the receiver regarding its sleep state, and forwards incoming streaming data to the receiver appropriately. We consider packet transmission from the transmitter (e.g a wired Access Point (AP)) to a wireless node over a static and perfect channel. To reduce its power consumption, the wireless node is capable of going to sleep. While in the sleep state, it is incapable of receiving packets from the transmitter or the AP (Note that we use the words transmitter and Access Point interchangeably throughout this thesis). The transmitter has the provision to buffer packets for the wireless node while it sleeps. These packets are transmitted to the wireless node when it wakes up. Clearly, the packets incur delay whenever the wireless node goes to sleep. On one hand the wireless node can greatly reduce its power consumption by sleeping for long periods of time. On the other hand, the longer it sleeps the greater is the delay incurred by its packets which have to be buffered at the transmitter during these long sleep durations. In our problem, we have a constraint on the maximum average delay that the data packets can tolerate at the transmitter, namely $\boldsymbol{D_{max}}$. Given this constraint of $\boldsymbol{D_{max}}$ we seek optimal sleep durations for the wireless node that minimizes its average power consumption.

We formulated this as an optimization problem and solved it numerically using dynamic programming (DP) [41][42]. The solutions from the numerical calculations strongly suggest that the optimal policy (that which minimizes average power consumption subject to an average packet delay constraint) is such that the transmitter should only command the receiver to sleep when there is no data queued at the

transmitter. The system thus behaves as a single server queue with vacations. We were able to derive closed form expressions for the optimal sleep duration, as well as the associated minimal rate of power consumption [43].

We extended the 1-user (one receiver-one transmitter) case to a multi-user (multiple receivers – one transmitter) scenario and used similar methodology as the 1-user case to obtain optimal sleep scheduling policies [44]. However, obtaining optimal results using the procedure of solving optimization problems proved to be very tedious and time consuming as the number of users in the system increased. As we increased the number of receivers in the system the state space for the optimization problem grew exponentially which made the run time of the numeric calculation unfit for any real time application. This motivated us to develop a simple algorithm that is fast, scalable realistic, and moreover adaptive to various traffic conditions.

In this thesis we perform dynamic power management on delay constrained wireless nodes by scheduling the nodes to sleep for an "optimal" duration such that they meet any "reasonable" average packet delay constraint while consuming the minimal power possible. We present three different sleep prediction schemes and scheduling strategies, namely the Round Robin scheme, the Shortest Sleep First scheme and the Steep Descent method. Unlike most other algorithms suggested to address similar concerns, none of our schemes require the knowledge of past packet arrival history of a wireless node. Our algorithm computes the sleep duration of a wireless node as a function of its average packet delay constraint and its current buffer load. It learns the traffic behavior over time and adapts its sleep-scheduling mechanism according to the traffic

trend coupled with the average packet delay constraint. We compare and contrast each of these schemes to the optimal results obtained by means of dynamic programming. We derive a theoretical lower bound on the average power consumption of a wireless node subjected to a specific average packet delay constraint and show that our sleep scheduling schemes perform sufficiently close to the lower bound. We also compare the Steep Descent method to the static sleep policy as specified in the 802.11 standard [40] and show that for certain traffic patterns our scheme outperforms the static sleep scheme.

1.5 Thesis Organization

This thesis is organized as follows: Chapter 2 presents the elements of the system model and the assumptions that we use throughout this work. Chapter 3 discusses the mathematical formulation of our optimization problem and outlines the solution method using Dynamic Programming (DP) equations for a single user case. It analyses the results obtained numerically from the DP and describes the closed form expressions derived for the optimal sleep duration. In Chapter 4 we describe how the power-delay optimization problem can be applied to a system comprising of multiple wireless nodes served by a single transmitter (or Access Point). We show and analyze the results that were acquired from solving this optimization problem numerically using the technique of Dynamic Programming for a system comprising of two wireless nodes communicating with a single transmitter. We also obtain a lower bound on the power consumption of a wireless system for a given average packet delay constraint. Chapter 5 introduces our adaptive algorithm – the "Round Robin" Scheme and the "Shortest Sleep First" scheme. Comparisons are made between the power-delay tradeoff achieved by this scheme and

those acquired from the DP formulation of the optimization problem. Chapter 6 compares and contrasts a static sleep scheduling scheme with a dynamic sleep scheduling scheme which we refer to as the “Steep Descent Method”. We summarize the thesis and conclude with some closing remarks in Chapter 1.

2 System Model

There are several aspects of the system model that we have applied in this research that are common throughout. Among these are the basic packet arrival model, packet service model, basic channel model and some of the system constraints. These common elements along with some of the notation that we adopt in this thesis are described in this chapter.

2.1 System Components

As depicted in **Figure 2.1**, we consider a system comprising of a single transmitter or Access Point (AP) and a few wireless devices which we refer to as wireless nodes. We also refer to these wireless nodes as "users" of the system. As mentioned in Chapter 1, we shall use the term transmitter and AP interchangeably. These wireless nodes can be laptops, PDAs, sensor nodes or any wireless node that can receive and transmit data. Data for the various wireless nodes arrive at the transmitter at the beginning of every slot. The transmitter can either transmit that data to the respective node instantly or buffer the data for the time being with the intent of transmitting it at a later point in time. Thus the transmitter has the capability to buffer data for the nodes. The transmitter transmits data to the nodes over a wireless channel. We discuss the channel characteristics in a later section.

2.2 Time Model, Packet Arrival and Service Model

We consider a discrete time framework in which time is divided into equal length intervals referred to as time slots or simply slots. These slots are indexed by integers. Data arrives randomly to the transmitter at the beginning of every slot, to be transmitted to the wireless nodes. We assume that the arrival process $\{a_n\}$ is a sequence of independent and identically distributed random variable with mean **λ.** Given this assumption, arrivals at the current slot are independent of arrivals in past slots so the arrival process is said to be *memoryless*. We also assume that the arrival process is independent of the channel and independent of arrivals for other nodes. We model the arrivals 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. Thus the transmitter has the capability to queue packets for the wireless node. For each node we define a_n to be a random variable that represents the number of packets that arrive at the AP 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 AP at the end of slot n, also called the *backlog* of slot n. We require that

$u_n <= x_{n-1} + a_n$.

The backlog process x_{n+1}, therefore, satisfies the recursion

$x_{n+1} = x_n - u_{n+1} + a_{n+1}$ (1)

Note that we use packet as a unit of date quantity. In this work the unit of data quantity is arbitrary; however, a single unit is assumed for all data quantities. In other words, the unit of data can be a bit, byte or fixed length packet as long as the same unit is applied uniformly. However the most practical choice is probably packets and hence we refer to the Quality of Service (QoS) criteria as average packet delay. We define $p = E[a_n]$.

2.3 Node Model

The wireless node can be in one of the two following states – "Sleep" or "Awake". While in the sleep state the wireless node switches off its receiving, transmitting and channel sensing circuitries. The awake state comprises of either the transmitting, receiving or idle states. When the node is in the idle state it does not receive or transmit data but instead it remains in the channel-sensing mode. There is an energy cost associated with each state. For each node we define $\boldsymbol{P_n}$ to be the energy consumed by the wireless node in slot $\boldsymbol{n}$. Let $\boldsymbol{P_s}$ denote the energy required per slot while in the sleep state and $\boldsymbol{P_a}$ denote the energy required per slot while in the awake state. We do not distinguish between the power consumption of the transmit, receive or idle states because the actual power consumed in these three modes are not too different as depicted in Table 1.1. Also for purposes of our problem, we are interested in the relative difference in energy consumption between the sleep and awake state in general and not so much the different modes of the awake state. However there is a power and delay penalty in transitioning from the sleep to the awake state. We model the delay penalty in terms of the power penalty as well. We argue that given sufficiently high power to transition from the sleep to the awake state, the delay associated with the same transition can be minimized. Let

P_{as} denote the energy cost for switching from awake to sleep state and P_{sa} be the energy cost for transitioning from sleep to awake state. We assume $P_a >> P_s$ [7].

Note that we have deliberately avoided associating a unit with the energy values. We observe that these values will be different for different devices. It will also vary from one manufacturer to the other for the same device. What we have instead focused on are the relative values of these power consumptions which are approximately standard for a specific device even across different manufacturers [7] . In general we have followed the trend of $P_a > P_{sa} > P_{as} > P_s$.

2.4 Channel Model

Data is transmitted by the transmitter to the wireless nodes across a wireless channel. We assume that the channel is static. By that we mean that the channel condition does not change with time. We also assume that the channel condition is "good". Since the channel is static, it implies that the channel condition is always "good". While this assumption is clearly not practical, the results obtained by making it can be interpreted as an upper bound on the best performance achievable.

2.5 System State

For a system comprising of 'L' nodes we characterize the system state in each slot by the L-tuple $(x1_n, r1_n, x2_n, r2_n, \ldots, xL_n, rL_n)$, where

xM_n denotes the backlog of node M at the end of slot n, and

rM_n denotes the sleep state of node M during slot n

rM_n can take the following values:

0, implying that wireless node ***M*** is awake during slot ***n***

1, implying that wireless node ***M*** is asleep during slot ***n*** but will be awake in slot ***n+1***

k, implying that node ***M*** will be asleep during slot ***n***, ***n+1***,…,***n+k-1***, but will be awake during slot ***n+k***, where ***k*** is any positive integer.

To illustrate the above concept clearly let us look at a concrete example. Let us consider a system comprising of 2 wireless nodes. Let us consider an arbitrary time slot say 3 (i.e ***n***=3). Let the system state at the end of the 3rd time slot be denoted by (0,0,2,3). This implies that at the end of the 3rd time slot, the first node has no data buffered for it and it is in the "Awake" state. The second node has two packets buffered for it at the transmitter and is in the "Sleep" state. It will be in the "Sleep" state for slots 4 and 5 as well but will be in the "Awake" state in slot 6.

2.6 System Constraint

In this thesis we only focus on the downlink traffic, i.e we are concerned about data transmission from the transmitter to the wireless nodes. For scenarios where the average packet delay constraint for a given node is same irrespective of whether the packet is a downlink packet or an uplink packet, the problem we study and its solution does not change. For instance when a node wakes up to receive its packets from the transmitter, it can piggyback the packets that it want to transmit along with the ACK packets that it is supposed to send to the transmitter as per the protocol of WLAN standard [40]. Thus the uplink packets will then incur the same delay as the downlink

traffic. However this argument does not hold if the delay constraint on the uplink packets is different from those of the downlink ones. We consider this problem to be one that can be investigated in the future.

Channel contention is another issue that we choose to disregard in this dissertation. We chose to model our system along the lines of the Point Coordination Function (PCF) mechanism as outlined in the standard for 802.11 [40] where the Access Point controls channel access resulting in a more harmonious channel allocation scheme.

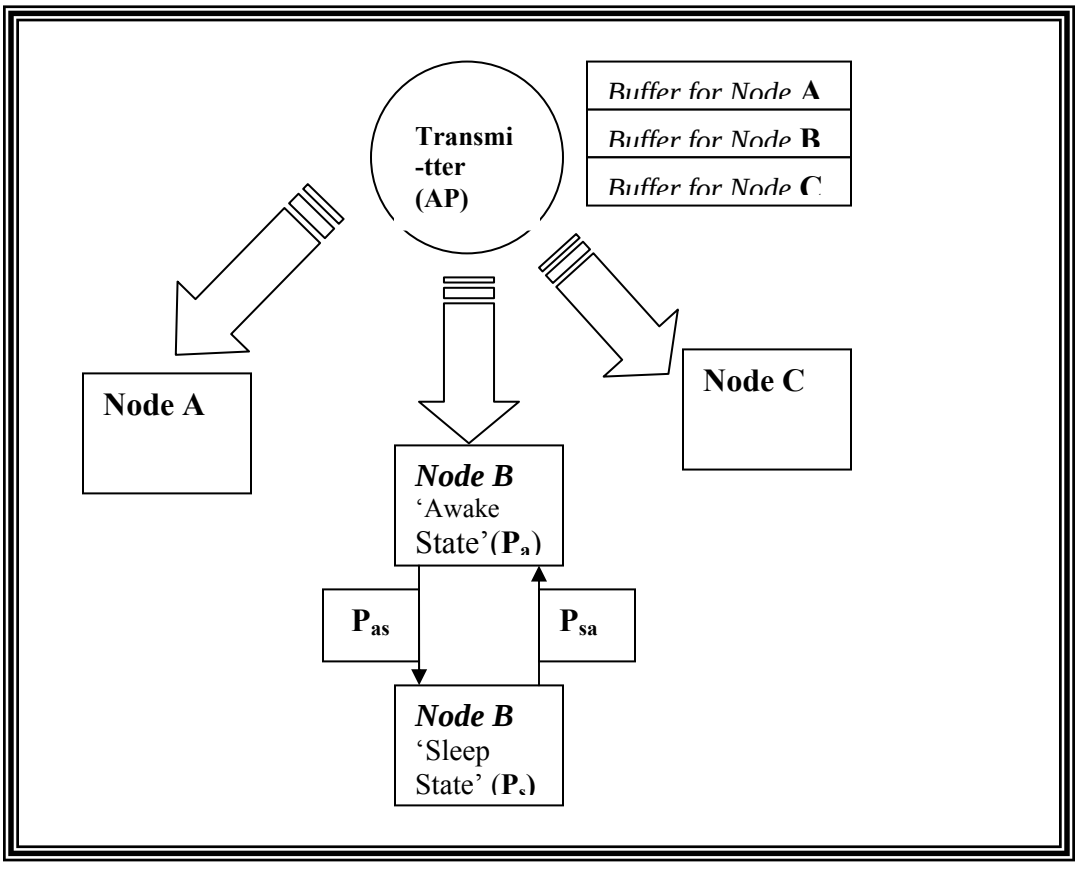

Figure 2.1: System model consisting of an AP and three wireless nodes

3 Optimal Single User Sleep Scheduling Policy

In this chapter we are generally interested in the problem of optimizing the timing and duration of sleep states of a wireless node with the objective of minimizing power with respect to a QoS constraint. The QoS parameter we have focused on is average packet delay. We try to calculate the optimal sleep duration as a function of average packet delay constraint and packet arrival rate.

To help gain a better understanding of the general problem, in this chapter we consider a simple model comprising of a single transmitter and a single receiver. The receiver is the wireless node whose mode we wish to control. The transmitter/AP can give commands to the receiver regarding its sleep state, and forward incoming streaming data to the receiver appropriately. We consider packet transmission from the transmitter to a wireless node over a static channel. To reduce its power consumption, the wireless node is capable of going to sleep. While in the sleep state, it is incapable of receiving packets from the AP. The AP has the provision to buffer packets for the wireless node while it sleeps. These packets are transmitted to the wireless node when it wakes up. Clearly, the packets incur delay whenever the wireless node goes to sleep. On one hand the wireless node can greatly reduce its power consumption by sleeping for long periods of time. On the other hand, the longer it sleeps the greater is the delay incurred by its packets which have to be buffered at the AP during these long sleep durations. In our problem, we have a constraint on the maximum average delay that the data packets can tolerate at the AP, namely D_{max}. Given this constraint of D_{max}, we seek optimal sleep

durations for the wireless node that minimizes its average power consumption. We formulated this as an optimization problem and solved it numerically using dynamic programming. The solutions from the numerical calculations strongly suggest that the optimal policy is such that the AP should only command the receiver to sleep when there is no data queued for it at the transmitter. The system thus behaves as a single server queue with vacations. We were able to derive closed form expressions for the optimal sleep duration, as well as the associated minimal rate of power consumption.

3.1 Problem Statement For the 1-user case

We require that the average delay suffered by data packets at the AP be no more than D_{max}. We let $\bar{x} = E[x_n]$. According to Little's result, assuming that the backlog process x_n is stationary, the average delay is given by $\bar{x}/p$. Thus, we require that

$$\bar{x} \le pD_{\max}. \quad (2)$$

We approach the problem, by considering it as an optimal control problem over a finite time horizon [42] say over slots 0,1,2,….,N-1. In order to reflect the average delay constraint, we consider policies ϕ, such that

$$E\left[\sum_{n=0}^{N-1} x_n\right] \le pD_{\max} N \quad (3)$$

Let A be the set of all such policies.

We thus seek to minimize the total energy expended over the finite time horizon, over the set of all policies $\phi \in \mathsf{A}$, i.e., we consider the following optimization problem:

$$\min_{\phi}\left\{E\left[\sum_{n=0}^{N-1}P_n\right]\right\}\text{subject to } \phi \in \text{A} \text{ . (4)}$$

We use a similar approach as in [45] to solve the above equation.

Let $f(D_{\max})$ be the optimal value of the objective function (4) in our problem, i.e.

$$f(D_{\max}) = \min_{\phi\in A}\left\{E\left[\sum_{n=0}^{N-1}P_n\right]\right\} \quad (5)$$

We use duality to solve the above equation, since direct evaluation of $f(D_{\max})$ appears difficult. Defining β as a "dual variable", where $\beta \geq 0$, we define the dual function

$$h(\beta) = \min_{\phi}\left\{E\left[\sum_{n=0}^{N-1}P_n + \beta\sum_{n=0}^{N-1}x_n\right]\right\} \forall \beta \geq 0 \quad (6)$$

Assuming, we can calculate $h(\beta)$, $f(D_{\max})$ can be lower bounded as follows:

$$f(D_{\max}) = \min_{\phi\in A}\left\{E\left[\sum_{n=0}^{N-1}Pn\right]\right\}$$

$$\geq \min_{\phi\in A}\left\{E\left[\sum_{n=0}^{N-1}Pn\right] + \beta\left(E\left[\sum_{n=0}^{N-1}x_n\right] - pD_{\max}N\right)\right\}$$

$$\geq \min_{\phi}\left\{E\left[\sum_{n=0}^{N-1}P_n\right] + \beta\left(E\left[\sum_{n=0}^{N-1}x_n\right] - pD_{\max}N\right)\right\}$$

$= h(\beta) - \beta(pD_{\max}N)$ so that

$$f(D_{\max}) \geq \sup_{\beta\geq 0}\left\{h(\beta) - \beta\left(pD_{\max}N\right)\right\} \quad (7)$$

For sufficiently large ***N***, using a time sharing argument, it is clear that $f(D_{\max})$ is asymptotically convex. It can thus be seen that there is no "duality gap", and therefore in this case the difference between the two sides of (7) becomes negligible as compared with ***N,*** as ***N*** gets larger.

3.2 Dynamic Programming Formulation for the 1 user case

We calculate $h(\beta)$, using the approach of dynamic programming [41]. We define,

$$J^N(x,r) = \min_{\phi} E\left[\sum_{n=0}^{N-1} P_n + \beta \sum_{n=0}^{N-1} x_n \mid x_0 = x, r_0 = r\right] \qquad (8)$$

Then $J^N(x,r)$ satisfy the following recursions:

For simplicity we assume here that

$p = probability(a_n = 1) = E[a_n]$

$$J^{N+1}(x,0) = Pa + \beta x + \min\left\{\begin{array}{l} p[J^N(x,0)] + (1-p)[J^N((x-1)^+,0)], \\ P_{as} + \min_{k\geq 1}\left\{pJ^N(x+1,k) + (1-p)J^N(x,k)\right\} \end{array}\right\} \qquad (9)$$

where $(x-1)^+ = \max(x-1,0)$

$$J^{N+1}(x,1) = Ps + Psa + \beta x + \left\{p[J^N(x,0)] + (1-p)[J^N((x-1)^+,0)]\right\} \qquad (10)$$

$$J^{N+1}(x,k) = Ps + \beta x + \left\{ p[J^N(x+1,k-1)] + (1-p)[J^N(x,k-1)] \right\} \forall k > 1 \quad (11)$$

The initial condition of the recursion is:

$$J^0(x,k) = 0, \forall x,k \quad (12)$$

We augment the dynamic programming recursions to calculate:

$$E\left[\sum_{n=0}^{N-1} P_n \mid x_0 = x, r_0 = r\right] \quad (13)$$

$$E\left[\sum_{n=0}^{N-1} x_n \mid x_0 = x, r_0 = r\right] \quad (14)$$

for the corresponding optimal policy, for each value of β.

3.3 Results

We performed a series of numerical calculations to calculate the minimum average energy required per slot that satisfied the maximum allowed average delay constraint. In order to realize a finite state space, we truncated the state space to a finite maximum backlog, $\boldsymbol{x_{max}}$ and the maximum consecutive sleep duration, $\boldsymbol{k_{max}}$. The value of $\boldsymbol{k_{max}}$ was found experimentally, for a corresponding packet arrival rate $\boldsymbol{p}$. Specifically, $\boldsymbol{k_{max}}$ is chosen sufficiently large so that its value does not affect the numerical results.

We developed a computer program, to implement the dynamic programming algorithm [42] for computing $\boldsymbol{h(\beta)}$ over a range of values of $\boldsymbol{\beta}$, given a Bernoulli arrival distribution with a packet arrival probability of $\boldsymbol{p}$ in every slot, a fixed service rate of one packet per slot and fixed energy costs of $\mathbf{P_a}$, $\mathbf{P_s}$, $\mathbf{P_{as}}$, $\mathbf{P_{sa}}$. We assumed $P_a >> P_s$ [7]. While

only finite horizon, finite state solutions can ever be exactly computed, we apply the *value iteration algorithm* described in [41], to compute good approximations to the optimal policy and corresponding average cost for the infinite horizon case. In other words, using this technique, we were able to compute policies and corresponding average power costs, that are stable as $N \to \infty$ and the size of the backlog state space, $\boldsymbol{x_{max}}$ becomes large. The results of this computation can then be used to select the optimal policy that satisfies the average delay constraint or to study the behavior of the optimal policy over a range of conditions like average packet arrival rate and average power costs.

Let, M*(x, r) denote the optimal policy for an initial system state of (x, r). We observed that as long as we had, $P_a >> P_s$, M*(x,r) was always found to be of the form :

$$\begin{aligned} M^*(x,r) &= 0; \text{when } x > 0, r = 0 \\ &= k; \text{where } k > 0, x = 0, r = 0 \\ &= r - 1; r > 0, x \geq 0 \end{aligned} \qquad (15)$$

Explicitly stating, the optimal policy commands the receiver to be in the awake state (denoted by '0') as long as there are packets buffered for it at the transmitter (x>0). It puts the receiver to sleep (for '$\boldsymbol{k}$' slots) when there are no packets buffered for the receiver (x=0). If the receiver is already in the sleep mode (r>0) then the optimal policy has no other choice but to decrement the sleep duration till the receiver is awake again (denoted by the expression 'r-1'). The value of $\boldsymbol{k}$ (the sleep duration) is dependant on $\boldsymbol{D_{max}}$ and $\boldsymbol{p}$.

We present some representative results related to specific scenarios in a graphical representation. We assume $\boldsymbol{p}$ =.1, P_a =1, P_{as}=.0001, P_{sa}=.01, P_s=.001 to obtain the results displayed in Figure 3.1 and Figure 3.2.

We see that the average power consumption per slot of a wireless node decreases as the tolerable average packet delay of the node increases. This is explained by the fact that as the average packet delay tolerance of an user increases, the transmitter can subject the user to longer sleep durations without compromising on the average delay constraint. These longer sleep durations result in lower power consumptions. Thus in Figure 3.2 we see that as the system tolerance to average packet delay increases, the sleep durations also increase. Figure 3.1 illustrates the fact that an increase in tolerable average packet delay, decreases the average power consumption per slot.

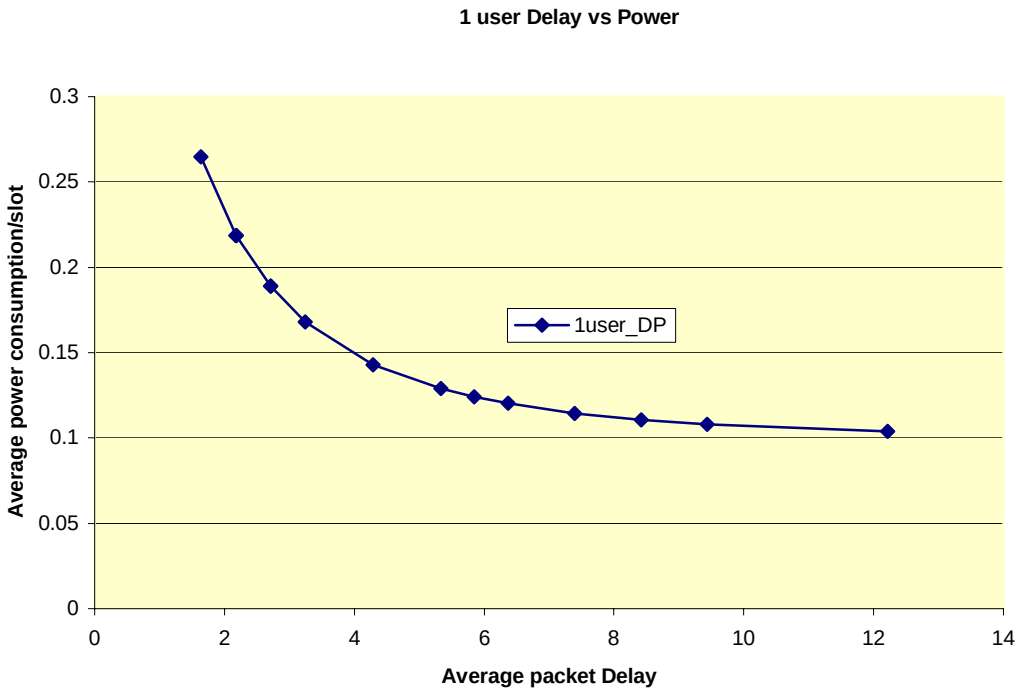

Figure 3.1: Average packet delay versus average power consumption/slot for a single user system

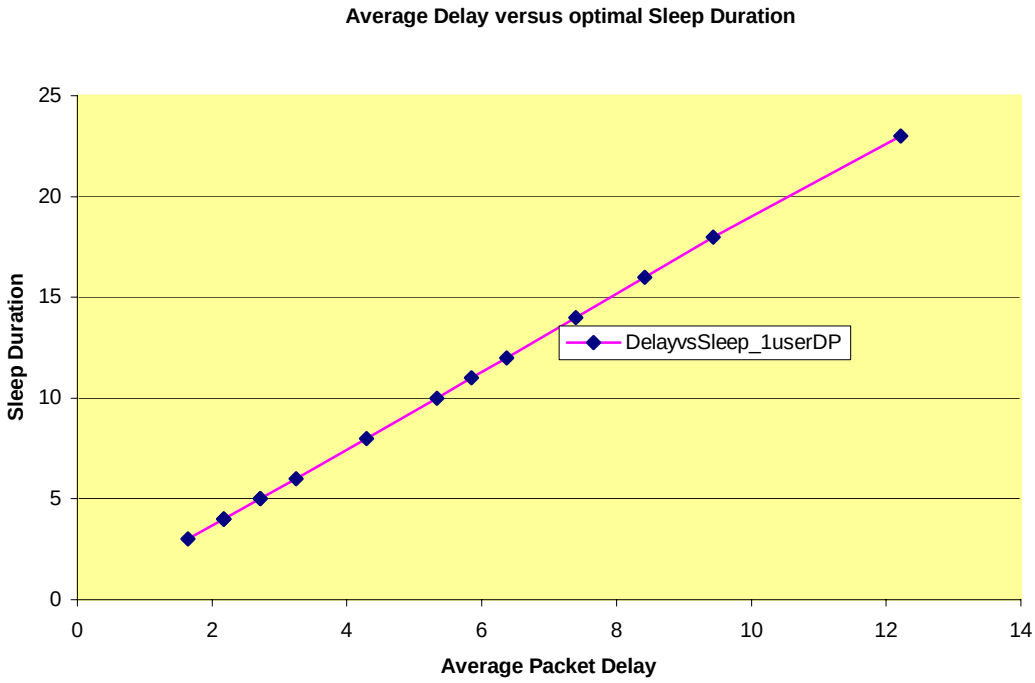

Figure 3.2: Average packet delay versus optimal sleep duration for a single user system

It is evident that in the optimal policy, the system behaves similar to a single server queue with vacations. Specifically, when the queue empties, the server goes on "vacation" for k slots. Therefore we can evaluate the performance of the optimal policy by analyzing such a queuing model. We applied a branching process argument to the policy derived from the numerical computation, to acquire two closed form equations. One of these equations, relate the average packet delay D_{max}, to the sleep duration k, of the wireless node. The other gives us the average power expended per slot under the policy obtained as per the dynamic programming formulation. We elaborate on the derivation in the next section.

$$\text{Average packet delay} = D_{max} = \frac{k(k+1)}{2k+2(1-p)^{k+1}} \quad (16)$$

$$\text{Average Energy Expended per slot} = \frac{\{(1-p)(P_{as} + P_s k + P_{sa}) + Pa\{(1-p)^{k+1} + pk\}\}}{\{k + (1-p)^{k+1}\}} \quad (17)$$

It is of interest, to note from equation (16) that the optimal sleep duration ***k*** is only a function of the average delay D_{max} and packet arrival rate ***p***. It does not depend on the associated energy costs like P_a, P_{as}, P_{sa} and P_s.

3.4 Derivation of the Closed-Form Expression

We now present an analysis of the optimal policy as outlined in (14). We shall study the optimal policy with a branching process analysis [41] to obtain a closed form expression of the average power consumption and the average delay as a function of the sleep duration ***k***.

Let us assume that the receiver goes to sleep for ***k*** slots when there are 0 packets in the buffer. Packets for the sleeping node accumulate at the AP during these ***k*** slots. When the receiver wakes up after ***k*** slots, it serves the first batch of packets. A second batch of packets may arrive while packets from the first batch are being served and so forth until the backlog is cleared and the buffer occupancy goes down to 0 again. The receiver then goes to sleep again for ***k*** slots. If there are no packet arrivals during the sleep duration (***k*** slots), the receiver goes back to sleep again for yet another ***k*** slots. Since the arrival rate ***p*** <service rate, where service rate is equal to 1 packet per slot the system is stable and hence reaches this 0 backlog state often.

We define one "Epoch" as the time period between the onset of two consecutive Sleep cycles (of duration ***k*** slots). Thus each epoch consist of several sub-epochs. The first sub-epoch comprises of the ***k*** slots during which the receiver is in the sleep state. The second sub-epoch comprises of a random number of slots required to serve the batch of packets that arrives during the ***k*** sleep slots. In general, the ***n***th sub-epoch comprises of the service time of the batch of packets that arrive during the ***(n-1)***th sub-epoch. Obviously, the receiver is asleep during the first sub-epoch and awake during the remaining sub-epochs. An epoch terminates when the receiver has no more packets to serve and is ready to go to sleep again.

Let ***L*** be a random variable that represents the duration of an epoch. Note that, ***L=k+A*** where ***A*** is a random variable representing how long the node is awake during an epoch and ***k*** is the duration of the receiver's sleep state.

Let $\boldsymbol{Z_n}$ denote the length (i.e the number of slots) of the ***n***th sub-epoch.

$$\begin{aligned} \text{Epoch Length} = L &= \sum_{n=0}^{\infty} Z_n \\ &= Z_0 + \sum_{n=1}^{\infty} Z_n \\ &= k + A \end{aligned} \qquad (18)$$

As mentioned previously, time is divided into equal unit length intervals called slots. Packets follow a Bernoulli arrival pattern, with probability ***p*** of a packet arrival in every slot. For simplicity we assume that when the receiver is awake, one packet can be served in each slot.

Note that,

$$\text{Average Power Expended per slot} = \frac{P_{as} + P_s k + P_{sa} + P_a E[A]}{E[\text{Epoch Length}]} \qquad (19)$$

$$\text{Average Backlog per slot} = \frac{\text{Average Backlog Over an Epoch}}{\text{Average Epoch Length}} = \frac{E\left[\text{Cumulative Backlog Over an Epoch}\right]}{E\left[\text{Epoch Length}\right]} \qquad (20)$$

$$E[L] = k + E[A] \qquad (21)$$

We now use branching process techniques to analyze average epoch length and average backlog over an epoch. Note that the minimum length of an epoch = ***k*** +1, since the wireless node is awake for at least one slot after it sleeps for ***k*** slots.

We define the following terms. Figure 3.3 provides a pictorial depiction of the system.

$Y_0 = \sum_{n=1}^{k} x_n$

= Cumulative backlog at the end of first subepoch

$B_0 = \sum_{n=1}^{k} a_n$

= total number of packet arrivals at the end of first subepoch

$B_1 = \sum_{n=k+1}^{k+B_0} a_n$

= total number of packets that arrived during the service time of the batch of B_0 packets

Thus, we have,

$$E[B_0] = pk$$

$$E[B_0^2] = pk + (k-1)p^2k$$

Also,

$$E[B_1 \mid B_0 = B_0] = pB_0$$

Note that if $B_0 = 0$, then

$$E[B_1 \mid B_0 = 0] = p$$

Hence,

$$E[B_1] = pE[B_0] = p(pk) = p^2k$$

Generalizing, we have

$$E[B_m] = pE[B_{m-1}] = p^{m+1}k \qquad (22)$$

Similarly

$$\begin{aligned} E[B_m^2] &= pE[B_{m-1}] + p^2E[B^2{}_{m-1}] - p^2E[B_{m-1}] \\ E[B_m^2] &= p^m[kp(1-p)] + p^2E[B^2{}_{m-1}] \end{aligned} \qquad (23)$$

After solving the above recursion we get,

$$E[B_m^2] = p^{2m}[pk + p^2k(k-1)] + p^{m+1}k(1-p^m) \qquad (24)$$

We now find a general expression for the expected cumulative backlog after each sub-epoch.

$$E\left[Ym \mid B_{m-1}\right] = (1+2+\ldots+(B_{m-1}-1)) + \frac{p(B_{m-1})(B_{m-1}+1)}{2}$$
$$= \frac{(B_{m-1})^2(p+1)}{2} + \frac{B_{m-1}(p-1)}{2} \qquad (25)$$

Therefore,

$$E\left[Y_m\right] = \frac{E\left[(B_{m-1})^2\right](p+1) + E\left[B_{m-1}\right](p-1)}{2} \forall m > 0 \quad (26)$$

Substituting values of $E[(B_{m-1})^2]$ and $E[B_{m-1}]$ and simplifying, we get,

$$E\left[Ym\right] = \frac{p^{2m}k(k-1)(p+1) + 2p^{m+1}k}{2} \forall m > 0 \quad (27)$$

Thus,

$$\sum_{m=1}^{\infty} E\left[Y_m\right] = \frac{p^2k(k+1)}{2(1-p)} \qquad (28)$$

Next, we find the average length of an epoch.

$$\text{Epoch Length} = k+1, \text{if } B_0 = 0$$
$$= k + B_0 + B_1 + \ldots, \text{otherwise} \qquad (29)$$

$$E\left[L\right] = k + 1 * \text{Probability}(B_0 = 0) + E[B_0] + E[B_1] + \ldots$$

Simplifying, we have,

$$E\left[\text{Epoch Length}\right] = \frac{k}{1-p} + (1-p)^k \quad (30)$$

Thus,

$$E[A] = E[\text{Epoch Length}] - k$$
$$= \frac{k}{1-p} + (1-p)^k - k \qquad (31)$$

Therefore,

$$\text{Average Backlog per slot} = \frac{E[Y0] + E[Y1]E[Y2] + \ldots}{E[\text{Epoch Length}]} \qquad (32)$$

Replacing with appropriate values, we get,

$$\text{Average Backlog/slot} = \frac{pk(k+1)}{2k + 2(1-p)^{k+1}} \qquad (33)$$

From Little's Formula, we get,

$$\text{Average maximum delay} = D_{\max}$$
$$= \frac{k(k+1)}{2k + 2(1-p)^{k+1}} \qquad (34)$$

$$\text{Average Power Expended/slot}$$
$$= \frac{P_{as} + P_s k + P_{sa} + P_a E[A]}{E[\text{Epoch Length}]} \qquad (35)$$

Hence replacing (35) with appropriate values we get:

$$\text{Average Power Expended/slot}$$
$$= \frac{(1-p)(P_{as} + P_s k + P_{sa}) + P_a\left((1-p)^{k+1} + pk\right)}{k + (1-p)^{k+1}} \qquad (36)$$

It is of interest, to note from equation (34) that the sleep duration $\boldsymbol{k}$ is only a function of the average delay $\boldsymbol{D_{max}}$ and packet arrival rate $\boldsymbol{p}$. It is independent of the associated energy costs like P_a, P_{as}, P_{sa} and P_s.

We have obtained a similar expression for a more general case. If, the service times of the packets are independent and identically distributed random variables, with a general distribution, then the average packet delay and average power cost expressions take the following form:

Let,

s_i = number of slots needed to serve the i^{th} packet

$T_0 = \sum_{i=1}^{B_0} s_i$ = the service for B_0 packets

$$E[s_i] = \mu$$
$$E[s_i^2] = \mu'$$

Then,

$$\text{Average Backlog/slot} = \frac{pk^2 + pk + p^2k^2 - p^2k - p^2k^2\mu}{2(1-p^2\mu^2)(k+(1-p)^k(1-p\mu))} + \frac{\mu' p^3k + \mu' p^2k - \mu p^3k^2 - 2p^3\mu^2k}{2(1-p^2\mu^2)(k+(1-p)^k(1-p\mu)} \quad (37)$$

From Little's Formula, we get

$$\text{Average Maximum Delay} = D_{\max}$$
$$= \frac{k^2 + k + pk^2 - pk - pk^2\mu - 2p^2\mu^2 k}{2(1-p^2\mu^2)(k+(1-p)^k(1-p\mu))} \qquad (38)$$
$$+ \frac{\mu' p^{2k} + \mu' pk - \mu p^2 k^2}{2(1-p^2\mu^2)(k+(1-p)^k(1-p\mu))}$$

$$\text{Average Power Expended/slot}$$
$$= \frac{(1-p\mu)(P_{as} + P_s k + P_{sa}) + P_a(kp\mu + (1-p)^k(1-p\mu))}{k+(1-p)^k(1-p\mu)} \qquad (39)$$

In the next chapter we extend this model to a system comprising of multiple wireless nodes communicating with a single transmitter or AP.

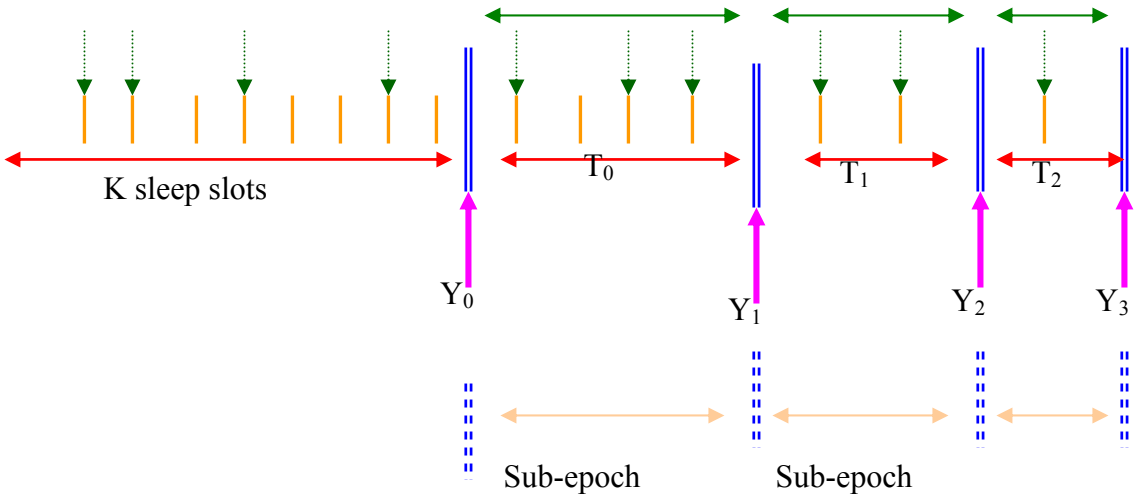

Figure 3.3:System model for the purpose of queuing analysis

4 Multiple User Transmission Policy

In the previous chapter we considered transmitting a single stream of data over a wireless link. However in most wireless systems the channel is normally shared by multiple users. Methods for accomplishing such channel sharing appear to fall into two broad categories:

1) coordinated approaches in which users share the channel in an orthogonal manner (i.e. without interfering with each other) and

2) un-coordinated approaches in which interference in allowed.

Time division multiplexing (TDM), frequency division multiplexing (FDM) and code division multiplexing (CDM) are all examples of the coordinated approach. At the other extreme we have random access techniques such as ALOHA and CSMA in which the channel is shared through contention so users do interfere with each other. In this thesis we assume the coordinated approach in which users share the channel in an orthogonal manner. Hence we do not take channel contention into account. Instead we ask the question when and for how long the transmitter (or AP) schedules the sleep state of individual nodes such that their individual average packet delay constraints can be met and overall average system power consumption can be minimized. In this chapter we consider this question in the context of two users.

4.1 Two User System

For the two-user case, we have two wireless nodes served by a single Access Point (AP). The AP has provision to buffer data packets for these two wireless nodes during their sleep durations. The system model for each node is the same as described for the one-user case. We denote the energy costs per slot associated with the sleep state for the two users as $P1_s$ and $P2_s$ respectively. The energy required by the two users per slot while in the awake state is denoted by $P1_a$ and $P2_a$. In addition, let $P1_{as}$ and $P2_{as}$ denote the energy required for switching from the awake to the sleep state and $P1_{sa}$ and $P2_{sa}$ be the energy required for switching from the sleep to the awake state for the two users respectively. We assume $P1_a >> P1_s$ and $P2_a >> P2_s$ [7].

For the two-user system we characterize the system state in each slot by the quadruple $(x1_n, r1_n, x2_n, r2_n)$, where

$x1_n$, $x2_n$ denote the backlog at the end of slot ***n,*** for user 1 and 2 respectively and

$r1_n$, $r2_n$ denote the sleep/awake state of users 1 and 2 respectively during slot ***n***

$r1_n$ and $r2_n$ can take the following values:

0; denotes that wireless node awake during slot ***n***

1; denotes that wireless node is asleep during slot ***n*** but will be awake in slot ***n+1***

k; denotes that wireless node is asleep during slot ***n***, ***n+1,…,n+k-1,*** but will be awake during slot ***n+k.***

4.2 Formulating the 2-User Optimization Problem

We consider a system comprising of two wireless nodes (or users) that are served by a single transmitter (or AP). We formulate an optimization problem similar to the one we did in the chapter 3 for a single user. The aim is to study the scheduling policy that the AP resorts to in transmitting packets to the two users. We are interested in finding answers to the following questions:

i) For a given average packet delay constraint (same for both users) which user does the AP serve first and how many packets does it serve at one go (does it empty the user's buffer or not)

ii) How long does the AP subject a user to sleep for? On what factors is this decision based?

iii) Are the sleep durations staggered such that the potential scenario where both nodes are awake at the same time can be avoided as much as possible?

iv) How do the above scenarios change if the packet delay constraints are different for the two users?

We require that the average delay suffered by data packets at the transmitter be no more than $D1_{max}$ and $D2_{max}$ 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 x1* and x2*. Thus, we require that, x1*<= $pD1_{max}$ and x2*<= $pD2_{max}$.

Similar to the one user case we also approach this problem by considering it as an optimal control problem over a finite time horizon [41] say over slots 0,1,2,….,N-1. In

order to reflect the average delay constraint, we consider policies ϕ_1 and ϕ_2 such that

$$E\left[\sum_{n=0}^{N-1} x1_n\right] \le pD1_{\max} N$$
and (40)
$$E\left[\sum_{n=0}^{N-1} x2_n\right] \le pD2_{\max} N$$

Let $\prod$ be the set of all such policies.

We thus seek to minimize the total energy expended over the finite time horizon, over the set of all policies ϕ' belonging to $\prod$, i.e., we consider the following optimization problem:

$$\min_{\phi'}\left\{E\left[\sum_{n=0}^{N-1} P1_n + P2_n\right]\right\} \text{subject to } \phi' \in \prod \quad (41)$$

We use a similar approach as in [42] to solve the above equation. Let
$f(D1_{\max}, D2_{\max})$ be the optimal value of the objective function, in our problem, i.e.

$$f(D1_{\max}, D2_{\max}) = \min_{\phi 1 \in \prod}\left\{E\left[\sum_{n=0}^{N-1}(P1_n + P2_n)\right]\right\} \quad (42)$$

We use duality to solve the above equation, since direct evaluation of $f(D1_{\max}, D2_{\max})$ appears difficult. Defining β1 and β2 as "dual variables", where β1,β2 >=0, we define the dual function

$$h(\beta 1,\beta 2)=\min_{\phi'}\left\{\begin{array}{l}E\left[\sum_{n=0}^{N-1}(P1_n+P2_n)\right]+\\ \beta 1\sum_{n=0}^{N-1}x1_n+\beta 2\sum_{n=0}^{N-1}x2_n\end{array}\right\} \quad (43)$$

$\forall \beta 1,\beta 2 >=0$

Assuming, we can calculate $h(\beta 1,\beta 2)$, $f(D1_{\max},D2_{\max})$ can be lower bounded as follows:

$$\begin{aligned}
f(D1_{\max},D2_{\max}) &= \min_{\phi'\in\prod}\left\{E\left[\sum_{n=0}^{N-1}(P1_n+P2_n)\right]\right\}\\
&\geq \min_{\phi'\in\prod}\left\{\begin{array}{l}E\left[\sum_{n=0}^{N-1}(P1_n+P2_n)\right]+\\ \beta 1\left(E\left[\sum_{n=0}^{N-1}x1_n\right]-p1D1_{\max}N\right)\\ +\beta 2\left(E\left[\sum_{n=0}^{N-1}x2_n\right]-p2D2_{\max}N\right)\end{array}\right\}\\
&\geq \min_{\phi'}\left\{\begin{array}{l}E\left[\sum_{n=0}^{N-1}(P1_n+P2_n)\right]+\\ \beta 1\left(E\left[\sum_{n=0}^{N-1}x1_n\right]\right)+\beta 2\left(E\left[\sum_{n=0}^{N-1}x2_n\right]\right)\\ -\beta 1\,p1D1_{\max}N-\beta 2\,p2D2_{\max}N\end{array}\right\}\\
&= h(\beta 1,\beta 2)-\beta 1(p1D1_{\max}N)-\beta 2(p2D2_{\max}N)
\end{aligned} \quad (44)$$

so that

$$f(D1_{\max},D2_{\max})\geq \sup_{\beta 1,\beta 2\geq 0}\left\{\begin{array}{l}h(\beta 1,\beta 2)-\\ \beta 1(p1D1_{\max}N)-\\ \beta 2(p2D2_{\max}N)\end{array}\right\} \quad (45)$$

As in the single user case, for sufficiently large N, using a time sharing argument, it is clear that apparent that *f* is convex. It can thus be seen that there is no "duality gap", and therefore in this case (45) is an equality.

4.3 Dynamic Programming Formulation

We calculate $h(\beta 1, \beta 2)$, using the approach of dynamic programming [42]. We define,

$$J^N(x1, r1, x2, r2) = \min_{\phi 1} E\left[\sum_{n=0}^{N-1}\left(P1_n + P2_n\right) + \beta 1\sum_{n=0}^{N-1} x1_n + \beta 2\sum_{n=0}^{N-1} x2_n \;\middle|\; x1_0 = x1, r1_0 = r1, x2_0 = x2, r2_0 = r2\right] \quad (46)$$

For simplicity we assume here that

$$p1 = probability(a1_n = 1) = E[a1_n] =$$
$$p2 = probability(a2_n = 1) = E[a2_n].$$

Then $J^N(x1, r1, x2, r2)$ satisfies the following recursions:

$$
\begin{aligned}
J^{N+1}(x1,0x2,0) = {} & Pa1 + Pa2 + \beta 1x1 + \beta 2x2 + \\
\min \Bigg[& \left\{ \begin{array}{l} p1\,p2\,J^N(x1,0,(x2+1),0) + \\ p1(1-p2)J^N(x1,0,x2,0) + \\ (1-p1)\,p2\,J^N((x1-1)^+,0,x2,0) + \\ (1-p1)(1-p2)J^N((x1-1)^+,0,x2,0) \end{array} \right\}; \\
& \left\{ \begin{array}{l} p1\,p2\,J^N(x1+1,0,x2,0) + \\ p1(1-p2)J^N((x1+1),0,(x2-1)^+,0) + \\ (1-p1)\,p2\,J^N(x1,0,x2,0) + \\ (1-p1)(1-p2)J^N(x1,0,(x2-1)^+,0) \end{array} \right\}; \\
& Pas1 + \\
& \min_{\forall k \geq 1} \left\{ \begin{array}{l} p1\,p2\,J^N(x1+1,k,x2,0) + \\ p1(1-p2)J^N(x1+1,k,(x2-1)^+,0) + \\ (1-p1)\,p2\,J^N(x1,k,x2,0) + \\ (1-p1)(1-p2)J^N(x1,k,(x2-1)^+,0) \end{array} \right\}; \\
& Pas2 + \\
& \min_{\forall k \geq 1} \left\{ \begin{array}{l} p1\,p2\,J^N(x1,0,x2+1,k) + \\ p1(1-p2)J^N(x1,0,x2,k) + \\ (1-p1)\,p2\,J^N((x1-1)^+,0,x2+1,k) + \\ (1-p1)(1-p2)J^N((x1-1)^+,0,x2,k) \end{array} \right\}; \\
& Pas1 + Pas2 + \\
& \min_{\substack{\forall k \geq 1, \\ \forall m \geq 1}} \left\{ \begin{array}{l} p1\,p2\,J^N(x1+1,k,x2+1,m) + \\ p1(1-p2)J^N(x1+1,k,x2,m) + \\ (1-p1)\,p2\,J^N(x1,k,x2+1,m) + \\ (1-p1)(1-p2)J^N(x1,k,x2,m) \end{array} \right\} \\
& \Bigg]
\end{aligned}
\qquad (47)
$$

$$J^{N+1}(x1,1,x2,0) = Ps1 + Pa2 + Psa1 + \beta 1x1 + \beta 2x2 +$$
$$\min \left[\begin{array}{l} \left\{ \begin{array}{l} p1\,p2\,J^N(x1,0,(x2+1),0) + \\ p1\,p2\,J^N(x1+1,0,x2,0) + \\ p1(1-p2)J^N(x1,0,x2,0) + \\ p1(1-p2)J^N(x1+1,0,(x2-1)^+,0) + \\ (1-p1)\,p2\,J^N(x1,0,x2,0) + \\ (1-p1)\,p2\,J^N((x1-1)^+,0,x2,0) + \\ (1-p1)(1-p2)J^N((x1-1)^+,0,x2,0) + \\ (1-p1)(1-p2)J^N(x1,0,(x2-1)^+,0) \end{array} \right\}; \\ Pas2 + \\ \min_{\forall k \geq 1} \left\{ \begin{array}{l} p1\,p2\,J^N(x1,0,x2+1,k) + \\ p1(1-p2)J^N(x1,0,x2,k) + \\ (1-p1)\,p2\,J^N((x1-1)^+,0,x2+1,k) + \\ (1-p1)(1-p2)J^N((x1-1)^+,0,x2,k) \end{array} \right\} \end{array} \right] \qquad (48)$$

$$J^{N+1}(x1,0,x2,1) = Pa1 + Ps2 + Psa2 + \beta 1x1 + \beta 2x2 +$$
$$\min \left[\begin{array}{l} \left\{ \begin{array}{l} p1\,p2\,J^N(x1,0,(x2+1),0) + \\ p1\,p2\,J^N(x1+1,0,x2,0) + \\ p1(1-p2)J^N(x1,0,x2,0) + \\ p1(1-p2)J^N(x1+1,0,(x2-1)^+,0) + \\ (1-p1)\,p2\,J^N(x1,0,x2,0) + \\ (1-p1)\,p2\,J^N((x1-1)^+,0,x2+1,0) + \\ (1-p1)(1-p2)J^N(x1,0,(x2-1)^+,0) + \\ (1-p1)(1-p2)J^N((x1-1)^+,0,x2,0) \end{array} \right\}; \\ Pas1 + \\ \min_{\forall k \geq 1} \left\{ \begin{array}{l} p1\,p2\,J^N(x1+1,k,x2,0) + \\ p1(1-p2)J^N(x1+1,k,(x2-1)^+,0) + \\ (1-p1)\,p2\,J^N(x1,k,x2,0) + \\ (1-p1)(1-p2)J^N(x1,k,(x2-1)^+,0) \end{array} \right\} \end{array} \right] \qquad (49)$$

$$J^{N+1}(x1,k,x2,0) = Ps1 + Pa2 + \beta 1x1 + \beta 2x2 +$$
$$\min\left[\begin{array}{l}\left\{\begin{array}{l} p1\,p2\,J^N(x1+1,k-1,x2,0) + \\ p1(1-p2)J^N(x1+1,k-1,(x2-1)^+,0) + \\ (1-p1)\,p2\,J^N(x1,k-1,x2,0) + \\ (1-p1)(1-p2)J^N(x1,k-1,(x2-1)^+,0)\end{array}\right\};; \\ Pas2 + \\ \min_{\forall m\geq 1}\left\{\begin{array}{l} p1\,p2\,J^N(x1+1,k-1,x2+1,m) + \\ p1(1-p2)J^N(x1+1,k-1,x2,m) + \\ (1-p1)\,p2\,J^N(x1,k-1,x2+1,m) + \\ (1-p1)(1-p2)J^N(x1,k-1,x2,m)\end{array}\right\}\end{array}\right] \quad (50)$$

$$J^{N+1}(x1,0,x2,k) = Pa1 + Ps2 + \beta 1x1 + \beta 2x2 +$$
$$\min\left[\begin{array}{l}\left\{\begin{array}{l} p1\,p2\,J^N(x1,0,x2+1,k-1) + \\ p1(1-p2)J^N(x1,0,x2,k-1) + \\ (1-p1)\,p2\,J^N((x1-1)^+,0,x2+1,k-1) + \\ (1-p1)(1-p2)J^N((x1-1)^+,0,x2,k-1)\end{array}\right\}; \\ Pas1 + \\ \min_{\forall m\geq 1}\left\{\begin{array}{l} p1\,p2\,J^N(x1+1,m,x2+1,k-1) + \\ p1(1-p2)J^N(x1+1,m,x2,k-1) + \\ (1-p1)\,p2\,J^N(x1,m,x2+1,k-1) + \\ (1-p1)(1-p2)J^N(x1,m,x2,k-1)\end{array}\right\}\end{array}\right] \quad (51)$$

$$J^{N+1}(x1,0,x2,1) = Ps1 + Ps2 + Psa1 + Psa2 + \beta 1x1 + \beta 2x2 +$$
$$\min\left[\begin{array}{l}\left\{\begin{array}{l} p1\,p2\,J^N(x1,0,(x2+1),0) + \\ p1(1-p2)J^N(x1,0,x2,0) + \\ (1-p1)\,p2\,J^N((x1-1)^+,0,x2+1,0) + \\ (1-p1)(1-p2)J^N((x1-1)^+,0,x2,0)\end{array}\right\}; \\ \left\{\begin{array}{l} p1\,p2\,J^N(x1+1,0,x2,0) + \\ p1(1-p2)J^N(x1+1,0,(x2-1)^+,0) + \\ (1-p1)\,p2\,J^N(x1,0,x2,0) + \\ (1-p1)(1-p2)J^N(x1,0,(x2-1)^+,0)\end{array}\right\}\end{array}\right] \tag{52}$$

$$J^{N+1}(x1,1,x2,k)_{\forall k>1} =$$
$$Ps1 + Ps2 + Psa1 + \beta 1x1 + \beta 2x2 + \tag{53}$$
$$\left[\begin{array}{l}\left\{\begin{array}{l} p1\,p2\,J^N(x1,0,(x2+1),k-1) + \\ p1(1-p2)J^N(x1,0,x2,k-1) + \\ (1-p1)\,p2\,J^N((x1-1)^+,0,x2+1,k-1) + \\ (1-p1)(1-p2)J^N((x1-1)^+,0,x2,k-1)\end{array}\right\} \\ \\ \end{array}\right]$$

$$J^{N+1}(x1,k,x2,1)_{\forall k>1} =$$
$$Ps1 + Ps2 + Psa2 + \beta 1x1 + \beta 2x2 + \tag{54}$$
$$\left[\begin{array}{l}\left\{\begin{array}{l} p1\,p2\,J^N(x1+1,k-1,x2,0) + \\ p1(1-p2)J^N(x1+1,k-1,(x2-1)^+,0) + \\ (1-p1)\,p2\,J^N(x1,k-1,x2,0) + \\ (1-p1)(1-p2)J^N(x1,k-1,(x2-1)^+,0)\end{array}\right\} \\ \\ \end{array}\right]$$

$$J^{N+1}(x1,k,x2,m)_{\forall k,m>1} =$$
$$Ps1 + Ps2 + \beta 1x1 + \beta 2x2 + \qquad (55)$$
$$\left[\begin{Bmatrix} p1\,p2\,J^N((x1+1),k-1,(x2+1),m-1)+ \\ p1(1-p2)J^N(x1+1,k-1,x2,m-1)+ \\ (1-p1)\,p2\,J^N(x1,k-1,x2+1,m-1)+ \\ (1-p1)(1-p2)J^N(x1,k-1,x2,m-1) \end{Bmatrix}\right]$$

where $(x\text{-}1)^{+} = \max(x\text{-}1,0)$.

The initial condition of the recursion is:

$$J^0(x1,r1,x2,r2) = 0, \forall x1,r1,x2,r2 \qquad (56)$$

Qualitatively, $J^N(x1,r1,x2,r2)$ denotes the optimal value of the cost function, over a horizon of ***N*** slots, given that wireless nodes 1 and 2 had a backlog of ***x1*** and ***x2*** packets respectively at the end of the first slot of the time horizon and the nodes were in state ***r1*** and ***r2*** respectively, during that slot.

The above equations enumerate all the different possible combinations of the initial state of the two users. For instance equation (47) calculates the optimal value of the cost function of the system when the initial condition of the two users are as follows: user 1 is in the awake state (sleep state 0) with x1 packets in its buffer and user 2 is also in the awake state with x2 packets in its buffer. Under such an initial state, the AP can choose to execute one of the following actions:

i) Keep both users awake but transmit only to user 1.

ii) Keep both users awake but transmit only to user 2.

iii) Put user1 to sleep and keep user 2 awake (and hence transmit to user 2)

iv) Put user 2 to sleep and keep user 1 awake (and hence transmit to user 1)

v) Put both users to sleep.

The DP chooses the option that yields the lowest value for the cost function.

We augment the dynamic programming recursions to calculate:

$$E\left[\sum_{n=0}^{N-1}(P1_n + P2_n) \mid x1_0 = x1, r1_0 = r1, x2_0 = x2, r2_0 = r2\right] \qquad (57)$$

$$E\left[\sum_{n=0}^{N-1} x1_n \mid x1_0 = x1, r1_0 = r1\right] \quad (58) \text{ and}$$

$$E\left[\sum_{n=0}^{N-1} x2_n \mid x2_0 = x2, r2_0 = r2\right] \qquad (59)$$

for the corresponding optimal policy, for each value of $\beta 1$ and $\beta 2$.

4.4 Results for the 2 user DP case

We performed a series of numerical calculations to evaluate the minimum average energy required per slot that satisfied the maximum allowed average packet delay. As with the 1-user case, in order to realize a finite state space, we constrained $\boldsymbol{a_n}$ as well as $\boldsymbol{u_n}$ for both users, to take on only integer values. In addition, we truncated the state space of both users to a finite maximum backlog, $\boldsymbol{x_{max}}$ and the maximum consecutive sleep duration, $\boldsymbol{k_{max}}$. The value of $\boldsymbol{k_{max}}$ was found experimentally, for the corresponding packet arrival rate $\boldsymbol{p}$.

We developed a computer program, to implement the dynamic programming algorithm [42] for computing ***h(β1 , β2)*** over a range of values of β1 and β2 given a Bernoulli arrival distribution with a packet arrival probability of ***p1*** and ***p2*** in every slot, a fixed service rate of one packet per slot and fixed power costs of $P1_a$, $P2_a$, $P1_s$, $P2_s$, $P1_{as}$, $P2_{as}$, $P1_{sa}$, $P2_{sa}$. We assumed $P1_a << P1_s$ and $P2_a << P2_s$ [7]. While only finite horizon, finite state solutions can ever be exactly computed, we apply the ***value iteration algorithm*** described in [41], to compute good approximations to the optimal policy and corresponding average cost for the infinite horizon case.

In other words, using this technique, we were able to compute policies and corresponding average power costs, that are stable as $N \to \infty$ and the size of the backlog state space, x_{max}, becomes large. The results of this computation can then be used to select the optimal policy that satisfies the average delay constraint or to study the behavior of the optimal policy over a range of conditions (e.g, packet arrival rates and power costs).

4.5 Results

We present some representative results related to specific scenarios in a graphical representation. The results represent 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=.1, P_a=1, P_{as}=0.0001, P_{sa} =0.01, P_s=0.001 to obtain the results displayed in Figure 4.1. We observe that power consumption of the system is inversely proportional to the average packet delay of the wireless nodes. The graph shows that for a higher average packet delay (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.

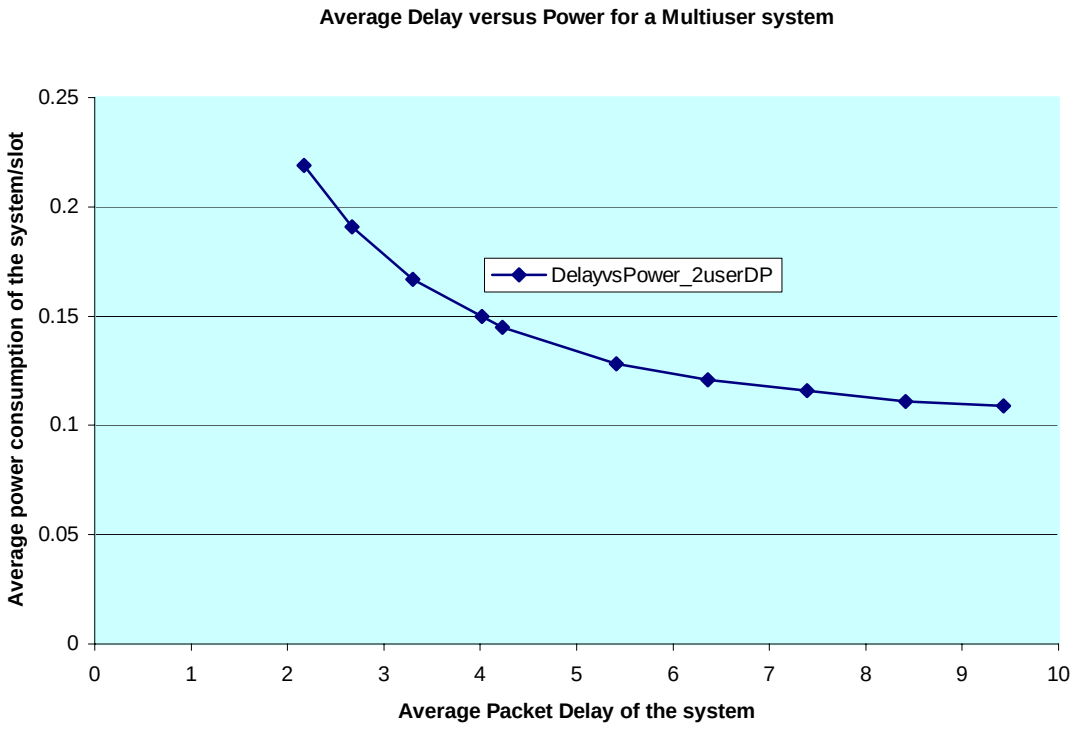

Figure 4.1: Graph representing average system packet delay to the average system power consumption per slot

The numerical results also provided us with insights to the questions outlined in section 4.2. For instance, under appropriate circumstances we saw that the policy subjected the nodes to unequal sleep durations such that their wake-up time would not overlap and unnecessary power consumption by a node having to wait its turn to be served, can be avoided. Unfortunately, unlike the 1-user case, we were unable to formalize a well-defined service policy for the 2-user case. When and for how long the

nodes were scheduled to sleep, varied widely depending on a number of parameters. For instance if both users had packets buffered for them at the AP and both of them were in the “awake” state, the DP chose different actions under different scenarios. Sometimes it subjected one node to sleep (most often the node with lesser number of packets), sometimes it had both nodes remain in the Awake state (mainly noticed if the difference between the buffer sizes of the two nodes was “small”). Even if the DP chose to put a node to sleep, the sleep duration varied widely depending on the relative difference between the buffer sizes of the two users. Thus it was very difficult to formalize the optimal policy for the 2-user system. However, we did gain a lot of insight into possible “optimal” actions that were later incorporated in our adaptive sleep-scheduling algorithm.

The other point worth noting is that the power-delay curve for the 2 user system almost fits snugly to the curve obtained for the 1-user system. This allows us to coin a lower bound for the average power consumption of a system for a specific average packet delay.

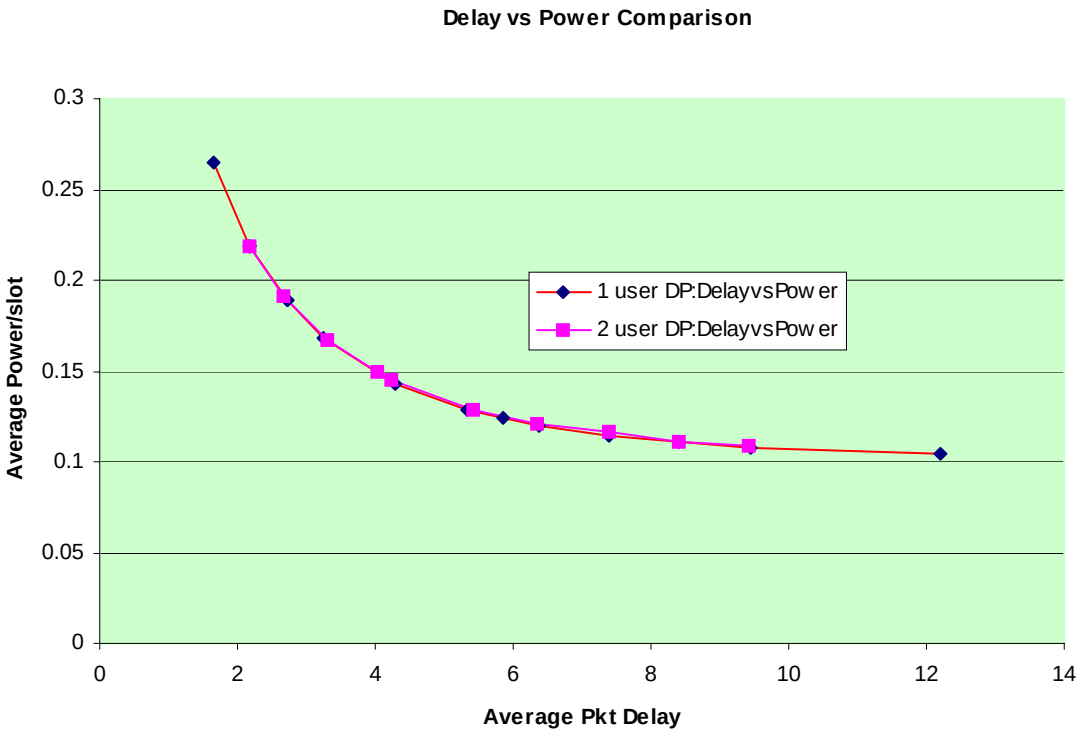

Figure 4.2: Comparing the energy - delay tradeoff for a 1 user and a 2-user system

4.6 A Lower Bound

We claim that the result of the minimization problem stated in equation (4) in Chapter 3, for a system comprising of a single wireless node can be used as a lower bound for the power consumption of a multi-node wireless system.

Let there be L wireless nodes in the system denoted by '1', '2',…'L'. Let,

$g(p,P_a,P_s,P_{as},P_{sa})$ denote the minimum average power consumption for a user in the single user case,

$P_j(n)$ denote the energy consumed by node 'j' in slot 'n'

$X_j(n)$ denote the backlog of user 'j' and the end of slot 'n'.

Let the total amount of energy consumed by node 'j' over a time horizon of N slots be

$$W_j = \sum_{n=1}^{N} P_j(n) \quad (60)$$

For a multi-node system we wish to minimize the overall average power consumption of the system while conforming to the average packet delay constraint of every node. Mathematically speaking we consider the following optimization problem:

$$\min_{\phi'} \left\{ E[\sum_{j=1}^{L} W_j] \right\} \text{subject to} \quad \mathrm{E}\left[\sum_{\mathrm{n}=1}^{\mathrm{N}} \mathrm{x_j(n)} \right] \le \lambda_j (D_{\max})_j N, \forall j = 1,2,..L \quad (61)$$

Theorem 1:

$$\min_{\phi'} \left\{ E[\sum_{j=1}^{L} W_j] \right\} \text{subject to}$$

$$\mathrm{E}\left[\sum_{\mathrm{n}=1}^{\mathrm{N}} \mathrm{x_j(n)} \right] \le \lambda_j (D_{\max})_j N, \forall j = 1,2,..L$$

$$\ge \sum_{j=1}^{L} g(p_j, (P_a)_j, (P_s)_j, (P_{as})_j, (P_{sa})_j) \forall j = 1,2,..L$$

Proof:

$$\text{Let } C = \min_{\phi'} \left\{ E[\sum_{j=1}^{L} W_j] \right\} \text{subject to } E\left[\sum_{n=1}^{N} x_j(n) \right] \le \lambda_j (D_{\max})_j N, \quad \forall j = 1,2,...L \quad (62)$$

Then,

$$C \ge \left\{ \sum_{j=1}^{L} \min \left[\begin{array}{l} \{E[W_j]\} \text{subject to} \\ \mathrm{E}\left[\sum_{\mathrm{n}=1}^{\mathrm{N}} \mathrm{x_i(n)} \right] \le \lambda_i (D_{\max})_i N, \\ \forall i = 1,2,...L \end{array} \right] \right\}$$

$$\geq \left\{ \sum_{j=1}^{L} \min \left[\begin{array}{l} E[W_j] \text{subject to} \\ \mathrm{E}\left[\sum_{n=1}^{N} x_j(n) \right] \leq \lambda_j (D_{\max})_j N \end{array} \right] \right\}$$

$$= \sum_{j=1}^{L} g\left(p_j, (P_a)_j, (P_s)_j, (P_{as})_j, (P_{sa})_j \right) \quad (63)$$

Acquiring the "optimal" sleep duration for a wireless node given an average packet delay constraint using the technique of solving a minimization problem is rather cumbersome for a multi-node system. Even for a system comprising of two wireless nodes the problem becomes hard to tackle. This is mainly because implementing the Dynamic Programming recursions (the technique that we use to solve the minimization problem since direct evaluation of the problem is hard) requires maintaining state spaces of each wireless node. The state spaces grow exponentially as the number of wireless nodes in the system increase. This makes the run time of the dynamic programming implementation to be painfully slow. Thus, although the above technique gives us the optimal solution to our problem, in practice it is hard to achieve this solution. Moreover, this optimization formulation requires perfect knowledge of packet arrival rate of a wireless node that is again hard to acquire realistically. These limitations led us to devise a more realistic algorithm that would be simple, fast, scalable and feasible yet which would meet the average packet delay constraint of the wireless system while consuming the least amount of power possible. In the following chapters we describe these algorithms and compare the energy-delay tradeoff of these algorithms to the theoretical lower bound that we have computed.

5 An Adaptive Algorithm – The "Round Robin" Scheme and the "Shortest Sleep First" Scheme

In the previous chapter, we observed that under various initial buffer occupancies (different values of $x1_0$ and $x2_0$) and initial sleep states (different values of $r1_0$ and $r2_0$) 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 sleep-scheduling algorithm. The aim of our algorithm is to minimize the overall average power consumption of a system comprising of multiple wireless nodes operating in an infrastructure mode in a WLAN where each node is subjected to a packet delay constraint. We consider a model comprising of multiple wireless nodes served by a single wired transmitter (or AP). As before, we study downlink traffic only. We also assume static and perfect channel conditions.

5.1 The Round Robin Scheme

We consider two different service orders - a Round Robin scheme and a non-Round Robin scheme which we refer to as the "Shortest Sleep First (SSF)" scheme. The difference between the two schemes lies in the order in which the AP transmits packets to the wireless nodes. The other functionalities are common to both schemes. In this section

we describe the common functionalities as well as the functionalities unique to the Round Robin scheme in detail.

Let us consider a system comprising of 'L' wireless nodes served by a single Access Point. Let the wireless nodes be arbitrarily indexed by the IDs '1', '2',…'L'. Let the tolerable average packet delay of the nodes be denoted by $\boldsymbol{D_1}$, $\boldsymbol{D_2}$,…., $\boldsymbol{D_L}$ respectively. Time is divided into discrete units which we refer to as 'slots.' At time t=0, we assume for concreteness that there are no packets buffered for any node and that the nodes are subjected to sleep for arbitrary sleep durations of $\boldsymbol{s_1}$, $\boldsymbol{s_2}$, …., $\boldsymbol{s_L}$ slots respectively.

In every slot, the AP has a state associated with it. . We denote the states of the AP by numbers '1','2',.. 'L'. The state of the AP designates the index of the node that is currently eligible for service. In the Round Robin scheme, the state of the AP cycles through the states in round robin order, i.e. 1,2, …, L, 1,2,…, L, 1, 2. …. The state of the AP may remain the same for consecutive slots. For concreteness, we assume that the state of the AP at time t=0 is 1. If the state of the AP is '*m*' in a slot, then we say that node '*m*' is the *"current node"* in that slot. For the Round Robin scheme, we call node 'm+1' the *"next node"* in that slot, where this is understood to be node 1 in the case where $m=L$.

In every slot a sequence of tasks are performed for every node in the system (except for task (4) which is performed on the current node). These tasks are executed in both the Round Robin scheme as well as the Shortest Sleep First scheme. These actions are sequentially outlined below:

1) **Packet Arrival:** Packets arrive (if at all) for a node at the beginning of a slot.

2) **Buffer Update:** Every node has two buffers assigned to it. Packets that arrive for a node when the node is in the sleep state, are placed in buffer '***S***'. Packets that arrive for the node when the node is awake but the state of the AP is not equal to the node ID are also placed in buffer '***S***'. Packets that arrive for a node when the node is in the awake state and while the state of the AP is equal to the node ID are placed in buffer '***R***'.

3) **Set "Early" bit :** If a node in the awake state and the state of the AP is not equal to the node ID, then we set a parameter which we refer to as "early" to the Boolean value "true". It is otherwise set to "false". We discuss the impact of this parameter on calculating the sleep duration of a particular node later in this paper.

4a) **Serve the current node if possible:** If the current node is in the awake state then the AP transmits a packet from the ***S*** buffer for the current node provided that the ***S*** buffer is not empty. If the ***S*** buffer is empty, depending on the sleep state of the next node, we may serve a packet from the ***R*** buffer. Otherwise if the current node is asleep, then the AP does not transmit a packet to any node. We discuss in details the service order and the service policy followed by the AP later in this section.

4b) **Calculate Next Sleep duration of the current node if needed**: It is first determined if the current state of the AP should be changed. If so, the current node will be put to sleep (assuming that the current node was assigned a sleep duration greater than 0. If not, the current node remains in the awake state but the state of the AP might still change). The AP calculates the sleep duration based on various factors which are discussed separately later in this section.

4c) **Put the node to sleep:** If it was determined in the previous step that the state of the AP should be changed, the current node is then put to sleep for the calculated number of slots (which might be 0 as well). Various book-keeping updates are performed when a node is put to sleep. For instance contents of buffer ***R*** are transferred to buffer ***S*** of the current node. Specifically in the SSF scheme, the vector ***V*** is re-arranged. We discuss this in detail in the next section.

5) **Calculate packet delay:** At the end of every slot, and for each node, the AP calculates a weighted average packet delay. For a particular node 'm', this value is referred to as the calculated packet delay of node 'm' and is denoted by $(CD)_m$.

We now discuss the process in which we calculate ***(CD).***

Let $\check{D}_i$ denote the delay of the '***i***'th packet in 'N' slots.

(CD) of that packet can then be recursively defined as :

$$(CD) = (1-\omega)(CD) + \omega \check{D}_i,$$

where ω is a parameter that represents the reciprocal of the "memory window".

6) **Update the vector *V*:** This functionality is executed at the end of every slot in the SSF scheme only. We discuss this functionality in details in next section.

5.1.1 Service Order

In the Round Robin (RR) scheme, we serve the nodes in a fixed, pre-determined sequential order. The service sequence is chosen arbitrarily. As mentioned earlier in the previous section the AP has a state associated with it in every slot. The state of the AP

designates the index of the node that is currently eligible for service. In the Round Robin scheme, the state of the AP strictly cycles through the states in round robin order, i.e. 1,2, …, L, 1,2,…, L, 1, 2. …. The state of the AP may remain the same for consecutive slots. For concreteness, we assume that the state of the AP at time t=0 is 1.

In the Round Robin scheme the decision to change the state of the AP from 'm' to 'm+1'(where 'm+1' can be the node with ID '1' when m=L) is based on the buffer occupancy of node 'm' and the sleep/awake state of the next node. We discuss the decision of when to change the state of the AP in the following sub-section.

5.1.2 Service Policy

When the current node is in the awake state, there are primarily two tasks that have to be performed. First, the current node has to receive the packets that have been buffered for it at the AP. In this section we discuss the first task. We postpone the discussion of the second task until the next sub-section. We describe the actions followed by the AP while transmitting packets to a particular node.

Let us assume that node 'm' is the current node, and that it is currently in the awake state. As mentioned previously, node 'm' has two buffers which we refer to as '$\boldsymbol{S_m}$' and '$\boldsymbol{R_m}$' respectively. Recall that any packet that arrives for the current node when it is in the awake state will be stored in buffer $\boldsymbol{R_m}$. In every slot, there are two scenarios that can possibly arise:

I. Buffer $\boldsymbol{S_m}$ can have greater than 0 packets

II. Buffer $\boldsymbol{S_m}$ can have 0 packets

We now discuss the actions executed by the AP under the two different scenarios.

I. Buffer S_m has greater than 0 packets:

In the above scenario, the AP first checks the sleep/awake state of the next node. Again, there can be two possible scenarios:

a) Next node can be in the sleep state

b) Next node can be in the awake state

Based on the above scenario, the AP executes different actions which we discuss below.

a) Next node is in the sleep state:

In such a case, the AP transmits a packet to the current node 'm' from its buffer S_m. If both buffers S_m and R_m are empty, the AP switches its state from the current node thus marking the end of the service duration of current node. The current node 'm' transits to the sleep state (if possible).

b) Next node is in the awake state:

In such a case, the AP transmits a packet to the current node 'm' from its buffer S_m. If buffer S_m becomes empty, the AP switches its state from the current node, thus marking the end of the service duration of the current node. Note that unlike the previous case, the AP does not check to see the status of buffer R_m in this case. The current node 'm' transits to the sleep state (if possible).

II. Buffer S_m has 0 packets:

If the AP encounters a situation where buffer S_m is empty, it checks for two things :

- the sleep/awake status of the next node and
- the buffer R_m of the current node

We now enumerate the actions executed by the AP for each of the three possible scenarios enumerated below.

a) Next node **is in the awake state and R_m. has greater than 0 packets:**

In the above scenario, the AP transfers the packets from buffer R_m to buffer S_m and transmits one packet to the current node. If the number of packets in S_m then reduces to either 0 or is less than the number of packets in $S_{next\ node}$, then the AP switches its state from the current node thus marking the end of the service duration of current node.

b) Next node is in sleep state and R_m.has greater than 0 packets:

In this case, the AP transfers the packets from buffer R_m to buffer S_m and transmits one packet to the current node. If the number of packets in S_m reduces to 0 then the AP switches its state from the current node, thus marking the end of the service duration of the current node.

c) R_m has 0 packets:

In this scenario the AP switches its state from the current node thus marking the end of the service duration of the current node.

5.1.3 Calculating the next sleep duration

In section 5.1.2 we said that when the current node is in the awake state, there are primarily two tasks that have to be performed. First, the current node has to receive the packets that have been buffered for it at the AP. After the AP has transmitted all or some of the data packets to the current node, it might choose to switch its state to another node thus marking the end of the service duration of the current node. In this case we need to determine the sleep duration of the current node. If this sleep duration is zero, this corresponds to keeping the current node awake. Otherwise, the current node is put to sleep for the calculated sleep duration. Our algorithm calculates this sleep duration through an adaptive "trial and error" mechanism. It examines the average delay suffered by the data packets buffered for the current node in the recent past and assigns a sleep duration based on that observation. In addition, in the Round Robin scheme the algorithm also takes into account the sleep duration of the other nodes in the system that are scheduled to be served before the node under consideration along with the number of data packets already buffered for them.

In order to motivate how we calculate the sleep duration, consider the first slot in the current cycle where the AP state was set to the current node. We call this slot the beginning of the current cycle. There are 3 scenarios.

1) Current node was early (early bit was set in this slot

2) Current node was not early, with no packets in buffer

3) Current node was not early with packets in buffer

If the current node was not early, we say it was "On time." Ideally we want to avoid scenarios 1) and 2) because they result in unnecessary power consumption. Thus, in our algorithm, under the circumstances where a node woke up "early" or was on time but had no packets to receive when its turn came up in the service order, the algorithm prolongs the sleep duration of that node in the next cycle in an effort to avoid the above scenarios that lead to unnecessary power consumption. However prolonging the sleep duration of a node is also dependent on the value of an important parameter **δ**. (We provide a rigorous definition of **δ** shortly). When δ is positive this implies that the actual average packet delay of the node is below the tolerable average packet delay, and our algorithm tends to increase the sleep duration of the node in its next cycle by a "few" slots. When δ is negative this implies that the average packet delay of the node is higher than the tolerable average packet delay, and our algorithm decreases the sleep duration of the node, again by a "few" slots. The magnitude by which the sleep duration is increased or decreased in every cycle is governed by another parameter K. We now provide a more

precise description of our algorithm which calculates the sleep duration of a particular node.

Let us assume that node ‘m’ is the node whose sleep duration we wish to calculate. Recall that the tolerable delay for node m is given by D_m,, and $(CD)_m$ is the estimate of the average packet delay for packets for node ‘m’ that is calculated at the end of every slot.

1> Calculate δ:

The algorithm calculates δ_m which is defined as the quantity

$(D_m - (CD)_m)$.

δ_m can be equal to, greater than or less than 0.

2> Check “early” bit:

We then check to see if the node transitioned to its awake state “early” by checking the parameter “early” which we described in the beginning of the next section.

3> Determine value of ‘K’:

We define **K** to be a parameter that can acquire different positive values based on the conditions outlined in Table 5.1.

Table 5.1 shows the action that the algorithm takes on altering the sleep duration of a node in the next cycle under different conditions which are considered in the first slot of the current cycle.

Table 5.1: Sleep Prediction Scheme for Round Robin Policy

	Early	On time with no packets	On time with packets
$\delta > 0$	Lengthen sleep **K = α1**	Lengthen sleep **K= α2**	Lengthen sleep **K= α3**
$\delta < 0$	Shorten sleep **K= α4**	Shorten sleep **K= α5**	Shorten sleep **K= α6**

For example, if we assume node 'm' was on time at the beginning of the current cycle, and if node m had no packets buffered for it at the beginning of the current cycle, and if δ_m is positive, then the parameter **K** acquires the value of **α3.**

4> Calculate (*I*):

For the node 'm', we define the parameter - **Increment = (I_m) = K$(\delta_m)^2$**.

The parameter K is positive, so that (I_m) will always acquire positive values. The current sleep duration of node 'm' is increased or decreased by (I_m) number of slots in the next cycle, in accordance with the sign of δ_m. Thus, for the particular node 'm', we perform the following operation:

If $(\delta_m) < 0$, then we set (I_m) to $(I_m)(-1) = (- I_m)$.

5> Update "Calculated Sleep Duration" (CSD):

Initially, at time t=0, $\mathbf{(CSD)_m}$ is set to an arbitrary value. We update **CSD** of node 'm' as follows.

$$\mathbf{(CSD)_m} = \max\{\ \mathbf{(CSD)_m} + \boldsymbol{I_m}\ ,\ 0\}$$

6> Calculate SSB (**S**um of **S**leep duration and **B**uffer length):

At any time t>0, we define the term (**SSB**) of node 'm' as follows:

$$(SSB)_m^{\ t} = (\text{buffer length})_m^{\ t} + (\text{sleep duration})_m^{\ t}$$

i.e., $(SSB)_m^{\ t}$ denotes the sum of the number of packets buffered for node 'm' at time t and the number of remaining slots for which node m shall be in the sleep state. Since we are calculating the sleep duration of node 'm', we calculate the SSB of every node in the system except for node 'm'.

7>Calculate max(SSB) :

Let $\boldsymbol{Z_m} = \max\ \{(SSB)_j^{\ t}\}\ \forall j = 1,2,..L,\ j \neq$ 'm'

8> Calculate ASD (Actual **S**leep **D**uration)**:**

We define $\mathbf{(ASD)_m}$ of node 'm' as the actual number of slots that the node is subjected to sleep for. For node 'm', our algorithm calculates $\mathbf{(ASD)_m}$ as follows:

$$\mathbf{(ASD)_m} = (\boldsymbol{Z_m})\ \text{slots, if } \mathbf{(CSD)_m} < \boldsymbol{Z_m}$$

$(\mathbf{ASD})_m = (\mathbf{CSD})_m$, otherwise

The node 'm' is then subjected to sleep for $(\mathbf{ASD})_m$ slots. In every cycle, when a node transitions to the sleep state, the algorithm executes the eight steps mentioned above. Obviously the value of $\mathbf{CSD}_m$ is updated in every cycle.

5.2 Shortest Sleep First (SSF) Scheme

In terms of average power consumption versus average packet delay, the Round Robin Scheme that we discussed in Section 5.1 performs fairly close to the theoretical lower bound that we have acquired through the process of dynamic programming as outlined in Chapter 3 and 4 for the symmetric case where all nodes have identical arrival statistics and the same tolerable average delay values. However we realized that the Round Robin Scheme is inappropriate for cases where all the wireless nodes in the system do not have the same average packet delay constraint or have very different arrival statistics. Thus we designed a variation of the Round Robin scheme. We refer to this new scheme as the Shortest Sleep First **(SSF)** scheme. The variation lies in the sequence of states that the AP acquires. Loosely speaking, unlike the Round Robin scheme the AP does not cycle through its various states in a fixed round robin order. On the contrary, the state of the AP is dictated by the sleep/awake duration of the wireless nodes in the system. We provide a rigorous description of the algorithm shortly. We retain all the other functionalities of the Round Robin scheme. In the following section we only describe the unique feature of the SSF scheme which is the Service Order. In order to avoid repetition we refrain from discussing the other functionalities of the SSF scheme.

Similar to the Round Robin scheme, let us consider a system comprising of 'L' wireless nodes served by a single Access Point. Let the wireless nodes be arbitrarily indexed by the IDs '1', '2',…'L'. Let the tolerable average packet delay of the nodes be denoted by $\boldsymbol{D_1}$, $\boldsymbol{D_2}$,…., $\boldsymbol{D_L}$ respectively. It is to be noted that the SSF scheme is most effective when $\boldsymbol{D_1}$, $\boldsymbol{D_2}$,…., $\boldsymbol{D_L}$ are all not equal or the arrival statistics for the traffic to the nodes are substantially different..

5.2.1 Service Order

Unlike the Round Robin Scheme, the SSF scheme has no arbitrarily pre-determined fixed order in which nodes are served. In other words, unlike the Round Robin scheme, the states of the AP in the SSF scheme does not cycle through '1','2',..'L','1','2',…,'L' sequentially in a round robin fashion.

In the SSF scheme, we maintain a vector $\boldsymbol{V}$ which comprises of an ordered arrangement of the sleep/awake durations of every node in the system. The sleep/awake durations in the vector $\boldsymbol{V}$ are ordered in ascending order. Thus in some slot '$\boldsymbol{q}$', vector $\boldsymbol{V}$ can assume the following configuration:

$$\boldsymbol{V_q}=\{\hat{S}^q_c, \hat{S}^q_g, \hat{S}^q_{L....}, \hat{S}^q_m\},$$

where $\hat{S}^q_i$ denotes the sleep/awake duration of node 'i' in slot '$\boldsymbol{q}$'. It is worth mentioning that $\hat{S}$ values corresponds to the 'r' values defined in the system state in Chapters 3 and 4.

Note that $\hat{S}^q_i$ can acquire values greater than, less than or equal to 0. A value less than or equal to 0 denotes the awake state while a positive value denotes the sleep state.

For example, $\hat{S}^q_i = 0$, implies that in slot 'q', node 'i' transitioned to the awake state from the sleep state. Similarly,

$\hat{S}^q_i = -2$ implies that node 'i' is awake in slot 'q' and has been awake in slots 'q-1' and 'q-2' as well.

$\hat{S}^q_i = 2$ implies that node 'i' is in the sleep state in slot 'q' and will continue doing so in slots 'q+1'. It will transition to the awake state in slot 'q+2'.

For further clarification, $\boldsymbol{V_q}=\{\hat{S}^q_c, \hat{S}^q_g, \hat{S}^q_{L....}, \hat{S}^q_m\}$, implies that in slot 'q', the sleep/awake duration of node with ID '*a*' is less than that of node with ID '*g*' and so on. It is evident that in slot 'q', node with ID '*a*' has the lowest sleep/awake duration while node with ID '*m*' has the highest sleep/awake duration.

In the beginning of any slot 'q', the state of the AP for that slot corresponds to the node ID of the first entry in vector $\boldsymbol{V_q}$. The state of the AP designates the index of the node that is eligible for service in slot 'q'. If the state of the AP is '*i*' in slot 'q', then we say that node '*i*' is the "current node" in slot 'q'. Also, in slot 'q', the second entry in vector $\boldsymbol{V_q}$ denotes the "next node" that is scheduled to receive service.

At the end of every slot 'q', the AP decreases the sleep/awake duration of all the wireless nodes in its system by one slot and updates the vector $\boldsymbol{V_q}$ with these values. This updated version serves as the new vector for slot 'q+1', i.e $\boldsymbol{V_{q+1}}$ is thus created. As per the service policy described in section 5.1.2, when an AP decides to switch state from the current node, it reorders the vector $\boldsymbol{V}$ at the end of that slot. Thus explicitly stating, the

vector ***V*** is updated in every slot but reordered only in slots where the AP switches its state.

During the re-ordering procedure, it might be the case that two nodes have the same sleep/awake duration. The AP then assigns higher precedence to the node that has the least amount of packets in its ***S*** buffer. This decision might apparently seem counter-intuitive but this approach turns out to be the most power-efficient choice. This decision was decided upon by examining the policy generated by the 2-user dynamic programming formulation under a similar circumstance.

If further the number of packets buffered for two nodes in their respective ***S*** buffers are equal then the AP assigns a higher precedence to the node which has consumed higher power till that point in time. If a tie exists even after the above classifications, then precedence of one node is chosen randomly over the other.

5.2.2 Calculating Next Sleep Duration

We use almost the same method as described in Section 5.1.2 of the Round Robin scheme to calculate the sleep duration of a node. Steps 1>through 5> through as described in Section 5.1.3 are followed as is. The ASD of a node is assigned its CSD value. Note that we do not calculate the SSB value of a node.

5.3 Discussion on the Results of the Algorithm

We wrote a computer program to implement both schemes of our 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. We tested our algorithm for

wireless local area networks comprising of up to ten wireless nodes. Recall that in chapter 4 we had acquired a lower bound on the average power consumption of a WLAN system subjected to an average packet delay constraint. We compare the results derived from the two schemes (Round Robin and SSF) of our algorithm to that lower bound. Figure 5.1 shows the delay versus power consumption tradeoff for the three schemes (DP, RR, SSF) for a system comprising of two wireless nodes served by a single AP/transmitter. The curve representing the results acquired from the DP formulation offers the best average delay-average power tradeoff. The Round Robin and SSF scheme does consume higher average power for a given average packet delay than the DP formulation. However, our algorithm is still very useful in a system comprising of multiple wireless nodes. What is lost in performance is gained in simplicity, scalability and speed. Moreover, our algorithm does not require a-priori knowledge of average packet arrival rate of any user. It has the ability to "learn" the traffic pattern and make "intelligent" decisions on subjecting the nodes to appropriate sleep durations. Another key merit of our algorithm over the DP formulation is its scalability. The number of users in the system has insignificant bearing on the run time of our algorithm. This is not true for the DP formulation. For every user that gets added to the system, the time to calculate the optimal action in each slot increases exponentially.

We now focus our discussion on comparing and contrasting the power efficiency of the Round Robin scheme to the SSF scheme. As is evident from Figure 5.1, under the condition where every wireless node in the system has the same average packet delay constraint the Round Robin scheme consumes less power on average than the SSF scheme for a given average packet delay. This can be explained by the fact that the

Round Robin scheme knows a-priori the order in which nodes will be served. It uses this information coupled with the knowledge of the service policy of the AP to calculate sleep durations for nodes such that unnecessary energy expenditure by a node is greatly minimized. For instance, the number of slots for which a node is in the awake state but the node ID does not correspond to the state of the AP in those slots, is greatly reduced thus minimizing unnecessary power consumption by a node. This improvisation renders the Round Robin scheme to be more power efficient than the SSF scheme where such an optimization is not possible.

However, the SSF scheme turns out to be more effective when the packet delay constraint of all users in the system are not equal. Especially under scenarios where huge disparity exists between the delay constraints of the users, the Round Robin scheme simply cannot meet the individual packet delay constraints. In the SSF, scheme, in contrast, a node with a large tolerable delay is allowed to sleep throughout several sleep cycles for nodes with small tolerable average delays, thus conforming to the packet delay constraint of every node in the system.

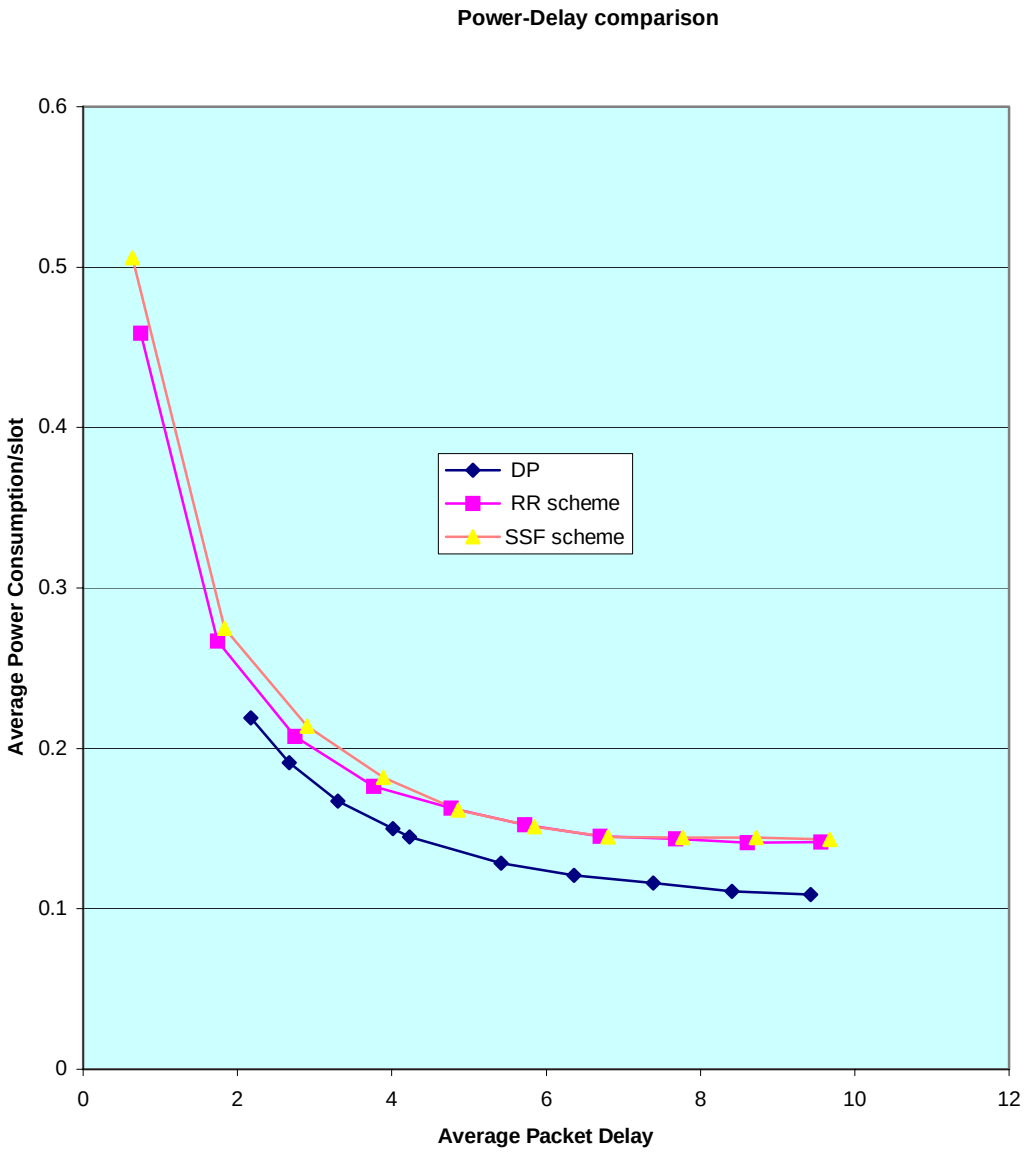

Figure 5.1: Comparison of power consumption for the DP, RR and SSF scheme

6 A Comparative Study of a Static and a Dynamic Sleep Scheduling Algorithm

As stated in Chapter 1, the issue of designing power efficient wireless devices has been approached at various levels. At one end of the spectrum, radio designers have created very efficient radio architectures, implementations and control algorithms while at the other end, network and application designers have proposed new protocols and algorithms that focus on energy efficiency for tasks such as routing, medium access control, source coding, error coding, encryption etc. It has been observed that, though radios and networks are designed to achieve a specified peak performance, it is often the case that the required performance is significantly less. For example, a wireless link to a PDA might be able to support streaming video; however it might only be used to upload emails. The bandwidth requirement in the case of email uploads is much less compared to what the link is capable of handling. We observe that such slack in system requirement can be used to our advantage in designing power efficient wireless communication systems. For example, in scenarios where the system does not need to perform at its peak rate, energy can be saved by slowing down transmission rate or turning off the radio for a while.

The above idea forms the corner stone of this chapter. We address the issue of designing power efficient wireless communication systems by developing a power efficient MAC layer sleep-scheduling algorithm. Instead of designing a rigid algorithm based on the worst-case operating conditions alone, we have designed an adaptive, learning algorithm that takes advantage of the changing requirements of the system and

acts accordingly. This strategy avoids superfluous power consumption without sacrificing system performance.

6.1 A Brief Overview of Power Management Scheme in 802.11

The power saving mode (PSM) in the Infrastructure model of an 802.11 [40] system has the provision for the wireless nodes to notify the Access Point (AP), at the time of association, of their sleep duration. The AP then allocates enough resources to support the sleeping nodes during their sleep states. The nodes wake up after their stipulated sleep duration to hear the beacon frame, transmitted by the AP. The beacon has information regarding packets buffered for the sleeping nodes at the AP. If the wireless node realizes (from the beacon message) that the AP has packets buffered for it, it stays awake for the entire duration of the beacon transmission and requests the AP to deliver its packets until its buffer is emptied. However, the standard does not specify any algorithm or scheme that the wireless nodes should follow to decide upon the duration of their sleep state. Usually the nodes choose a sleep period that is equal to their average packet delay tolerance. We refer to this scheme as the **"Sleep Equals Delay"** or **SED** scheme. This conservative scheme guarantees to meet any "reasonable" pre specified average packet delay at the cost of substantial power consumption. This led us to devise a simple, centralized, dynamic, scalable and adaptive algorithm that is a variation of the SSF scheme that we described in Chapter 5. We refer to this algorithm as the "Steep Descent Method" (SDM). Our algorithm adapts dynamically to the packet arrival rate and the average packet delay constraint of the wireless nodes. For a given average packet delay constraint a wireless node running our algorithm consumes much less power than when it

runs the SED scheme especially under non uniform traffic conditions. The simulation results prove the power efficiency of our adaptive sleep algorithm over the static SED scheme. In addition, we also compare the SDM scheme to the lower bound on power consumption derived in Chapter 4.6. Results show that the SDM scheme performs fairly close to the lower bound in terms of power consumption.

6.2 Adaptive Sleep Algorithm : STEEP DESCENT METHOD (SDM)

For the SDM scheme, we adhere to the system model that we described in Chapter 2. Specifically, we characterize each node in any particular slot 'i', by the tuple (x_m (i), r_m (i)), where

x_m (i) = backlog of node 'm' at the end of slot 'i', and

r_m(i) = state of node 'm' during slot 'i'

r_m can independently take the following values in any slot 'i':

0; {node 'm' is awake during slot 'i'}

-h; {node 'm' is awake during slot 'i' and has been awake in the previous h-1 consecutive slots}

1; {node 'm' is asleep during slot 'i' but will be awake in slot 'i+1'}

k; {node 'm' asleep during slot 'i', 'i+1'….'i+k-1' but will be awake during slot 'i+k'}

We now describe the algorithm in detail:

Let us consider a system comprising of 'L' wireless nodes served by a single Access Point. Let the wireless nodes be arbitrarily indexed by the IDs '1', '2',…'L'. Let the tolerable average packet delay of the nodes be denoted by $\boldsymbol{D_1}$, $\boldsymbol{D_2}$,…., $\boldsymbol{D_L}$ respectively. Time is divided into discrete units which we refer to as 'slots.' At time t=0, we assume for concreteness that there are no packets buffered for any node and that the nodes are subjected to sleep for durations that equals their respective packet delay constraints, namely $\boldsymbol{D_1}$, $\boldsymbol{D_2}$, …., $\boldsymbol{D_L}$ slots respectively.

In every slot, the AP has a state associated with it. We denote the states of the AP by numbers '1','2',.. 'L'. The state of the AP designates the index of the node that is currently eligible for service. The state of the AP may remain the same for consecutive slots. If the state of the AP is '*m*' in slot 'q', then we say that node '*m*' is the "*current node*" in slot 'q'. The state that the AP would acquire next in some slot 'q+y' corresponds to the ID of the "*next node*" in slot 'q'. We define the concept of a "*next node*" more rigorously in the next section. In every slot a sequence of tasks are performed for every node in the system (except for task (4) which is performed on the current node). These actions are sequentially outlined below:

1) **Packet Arrival:** Packets arrive (if at all) for a node at the beginning of a slot.

2) **Buffer Update:** Every node has two buffers assigned to it. Packets that arrive for a node when the node is in the sleep state, are placed in buffer '***S***'. Packets that arrive for the node when the node is awake but the state of the AP is not equal to the node ID are also placed in buffer ***S***. Packets that arrive for a node when the node is in the awake state and while the state of the AP is equal to the node ID are placed in buffer '***R***'.

3a) **Serve the current node if possible:** If the current node is in the awake state then the AP transmits a packet from the ***S*** buffer for the current node provided that the ***S*** buffer is not empty. If the S buffer is empty, depending on the sleep state of the next node, the AP may serve a packet from the ***R*** buffer. Otherwise if the current node is asleep, then the AP does not transmit a packet to any node. We discuss in details the service order and the service policy followed by the AP later in this chapter.

3b) **Calculate Next Sleep duration of the current node if needed**: It is first determined if the current state of the AP should be changed. If so, the current node will be put to sleep. The AP calculates the sleep duration based on various factors which are discussed separately later in this chapter.

3c) **Put the current node to sleep:** If it was determined in the previous step that the state of the AP should be changed, the current node is then put to sleep for the calculated number of slots. Various book-keeping updates are performed when a node is put to sleep. For instance contents of buffer ***R*** are transferred to buffer ***S*** of the current node, the vector ***V*** is re-arranged. We discuss this in detail shortly.

4)**Calculate packet delay:** At the end of every slot, and for each node, the AP calculates a weighted average packet delay. This value is referred to as the calculated packet delay ***(CD)*** and for a node 'm' it is denoted by $\boldsymbol{(CD)_m}$.

We now discuss the process in which we calculate ***(CD)***.

For node m,

$$(CD)_m(q) = \frac{\text{(Weighted Average Backlog)}_m\text{(q)}}{\text{(Weighted Average Packet Arrival Rate)}_m\text{(q)}}$$

where $(CD)_m(q)$ denotes the weighted average packet delay of node 'm' in slot 'q'.

Recall that x_m (q) denotes the backlog of node ‘m’ at the end of slot ‘q’. Let us define A_m(q) as the number of packets that arrive for node ‘m’ in slot ‘q’.

Let $(CB)_m$(q) of node ‘m’ in slot ‘q’ be recursively defined as :

$$(CB)_m(q) = (CB)_m(q)(1-\alpha) + \alpha x_m(q)$$

Also, let $(PA)_m$(q) of node ‘m’ in slot ‘q’ be defined as:

$$(PA)_m(q) = (PA)_m(q)(1-\alpha) + \alpha A_m(q)$$

where α is a parameter that represents the reciprocal of the “memory window”.

Note that at time t=0,

$(CB)_m = (PA)_m = 0$, ∀m.

Thus in slot ‘q’, $(CD)_m$ (q) is calculated as follows:

$$(CD)_m(q) = \frac{(CB)_m(q)}{(PA)_m(q)}$$

When $(PA)_m$ (q) is equal to zero, then $(CD)_m$(q) is set to zero as well.

5) **Update the vector *V*:** This functionality is executed at the end of every slot and we discuss this functionality in detail shortly.

6.2.1 Service Order

We recall that $r_m(q)$ denotes the state of node 'm' in slot 'q'. In every slot we maintain a vector $\boldsymbol{V}$ that comprises of the 'r' values of every node in the system arranged in ascending order. Thus in some slot '$\boldsymbol{q}$', vector $\boldsymbol{V}$ can assume the following configuration:

$\boldsymbol{V_q=\{r_m(q), r_n(q),\ldots,r_a(q)\}}$, where m,n,…a$\in${1,2,…,L}

In the beginning of any slot 'q', the state of the AP for that slot corresponds to the node ID of the first entry in vector $\boldsymbol{V_q}$. Since the state of the AP designates the index of the node that is eligible for service in slot 'q', thus if the state of the AP is '*i*' in slot 'q', then we say that node '*i*' is the "current node" in slot 'q'. Also, in slot 'q', the second entry in vector $\boldsymbol{V_q}$ denotes the "next node" that is scheduled to receive service.

At the end of every slot 'q', the AP decreases the 'r' value of all the wireless nodes in its system by one slot and updates the vector $\boldsymbol{V_q}$ with these new 'r' values. This updated vector acts as the new vector for slot 'q+1'. $\boldsymbol{V_{q+1}}$ is thus created. As per the service policy (which we describe shortly), when an AP decides to switch state from the current node, it reorders the vector $\boldsymbol{V}$ at the end of that slot. Thus explicitly stating, the vector $\boldsymbol{V}$ is updated in every slot but reordered only in slots where the AP switches its state.

During the re-ordering procedure, it might be the case that two nodes have the same 'r' value. The AP then assigns higher precedence to the node with a lower 'x' value. This decision might apparently seem counter-intuitive but it turns out to be the most power-efficient choice. This decision was decided upon by examining the policy

generated by the 2-user dynamic programming formulation under similar circumstance [44].

If further the 'x' values of two nodes are equal then the AP assigns a higher precedence to the node that has consumed higher power till that point in time. If a tie exists even after the above classifications, then precedence of one node is chosen randomly over the other.

6.2.2 Service Policy

The service policy for the Steep Descent Method is exactly the same as the Round Robin and the SSF scheme described in Chapter 5. However for continuity sake, we describe the service policy here again.

When the current node is in the awake state, there are primarily two tasks that have to be performed. First, the current node has to receive the packets that have been buffered for it at the AP. In this section we discuss the first task. We postpone the discussion of the second task until the next sub-section. We describe the actions followed by the AP while transmitting packets to a particular node.

Let us assume that node 'm' is the current node, and that it is currently in the awake state. As mentioned previously, node 'm' has two buffers which we refer to as '$\boldsymbol{S_m}$' and '$\boldsymbol{R_m}$' respectively. Recall that any packet that arrives for the current node when it is in the awake state will be stored in buffer $\boldsymbol{R_m}$. In every slot, there are two scenarios that can possibly arise:

I. Buffer $\boldsymbol{S_m}$ can have greater than 0 packets

II. Buffer $\boldsymbol{S_m}$ can have 0 packets

We now discuss the actions executed by the AP under the two different scenarios.

I. Buffer $\boldsymbol{S_m}$ has greater than 0 packets:

In the above scenario, the AP first checks the state of the next node. Again, there can be two possible scenarios:

a) Next node can be in the sleep state

b) Next node can be in the awake state

Based on the above scenario, the AP executes different actions which we discuss below.

a) Next node is in the sleep state:

In such a case, the AP transmits a packet to the current node 'm' from its buffer $\boldsymbol{S_m}$. If both buffers $\boldsymbol{S_m}$ and $\boldsymbol{R_m}$ are empty, the AP switches its state from the current node thus marking the end of the service duration of current node. The current node 'm' transits to the sleep state.

b) Next node is in the awake state:

In such a case, the AP transmits a packet to the current node 'm' from its buffer $\boldsymbol{S_m}$. If buffer $\boldsymbol{S_m}$ becomes empty, the AP switches its state from the current node, thus marking the end of the service duration of the current node. Note that unlike the previous case, the AP does not check to see the status of buffer $\boldsymbol{R_m}$ in this case. The current node 'm' transits to the sleep state.

II. Buffer S_m has 0 packets:

If the AP encounters a situation where buffer $\boldsymbol{S_m}$ is empty, it checks for two things –

- the state of the next node and
- the buffer $\boldsymbol{R_m}$ of the current node

We now enumerate the actions executed by the AP for each of the three possible scenarios enumerated below.

a) Next node is in the awake state and $R_{m.}$ has greater than 0 packets:

In the above scenario, the AP transfers the packets from buffer $\boldsymbol{R_m}$ to buffer $\boldsymbol{S_m}$ and transmits one packet to the current node. If the number of packets in $\boldsymbol{S_m}$ then reduces to either 0 or is less than the number of packets in $\boldsymbol{S_{next\ node}}$, then the AP switches its state from the current node thus marking the end of the service duration of current node.

b) Next node is in sleep state and $R_{m.}$ has greater than 0 packets:

In this case, the AP transfers the packets from buffer $\boldsymbol{R_m}$ to buffer $\boldsymbol{S_m}$ and transmits one packet to the current node. If the number of packets in $\boldsymbol{S_m}$ then reduces to 0 then the AP switches its state from the current node, thus marking the end of the service duration of the current node.

c) R_m has 0 packets:

In this scenario the AP switches its state from the current node thus marking the end of the service duration of the current node.

6.2.3 Calculating the Next Sleep Duration

In section 6.2.2 we said that when the current node is in the awake state, there are primarily two tasks that have to be performed. First, the current node has to receive the packets that have been buffered for it at the AP. After the AP has transmitted all or some of the data packets to the current node, it might choose to switch its state to another node thus marking the end of the service duration of the current node. In this case we need to determine the sleep duration of the current node. We define a "cycle" for a particular node 'm' as the number of consecutive slots for which it remains in the awake state followed by the number of consecutive slots for which it remains in the sleep state. Our algorithm calculates this sleep duration through an adaptive "trial and error" mechanism. It examines the average delay suffered by the data packets buffered for the current node in the recent past and compares it to the nodes tolerable delay value. It assigns a sleep duration based on this observation. If the $(\boldsymbol{CD})_{\boldsymbol{m}}$ value for node 'm' is "almost equal" to its $\boldsymbol{D_m}$ value, then the algorithm assigns the same sleep duration for node 'm' in the next cycle as well. If however the $(\boldsymbol{CD})_{\boldsymbol{m}}$ value for node 'm' is "slightly less" than its $\boldsymbol{D_m}$ value, then the algorithm assigns a sleep duration for node 'm' that is "slightly longer" than the current sleep duration in the next cycle. Otherwise, it assigns a "longer" sleep duration for node 'm' in the next cycle. We use parameters '**K**' and '**Y**' (which we define shortly) as tools to control the increment of a node's sleep duration. We now provide a more precise description of our algorithm that calculates the sleep duration of a particular node.

Let us assume that node ‘m’ is the node whose sleep duration we wish to calculate. Recall that the tolerable delay for node ‘m’ is given by $\boldsymbol{D_m}$, and $\boldsymbol{(CD)_m}$ is the average packet delay of node ‘m’ that is calculated at the end of every slot.

1> Calculate δ:

The algorithm calculates $\boldsymbol{\delta_m}$ which is defined as the quantity

$\boldsymbol{(D_m - (CD)_m)}$.

$\boldsymbol{\delta_m}$ can be equal to, greater than or less than 0.

2> Determine Maximum Sleep Duration:

We define ‘$\boldsymbol{M}$’ to be a parameter that can take positive values only. The maximum number of consecutive slots a node ‘m’ can sleep for is then defined as :

(MAXSLEEP)m = $\boldsymbol{M(D_m)}$

3> Determine values of ‘K’ and ‘Y’:

We define K and Y to be two parameters that can acquire different positive values based on different δ values as outlined in

Table 6.1. The table also shows the action that the algorithm takes on altering the sleep duration of a node in the next cycle under different conditions which are considered in the first slot of the current cycle.

Table 6.1: Sleep Prediction Protocol for the SDM Scheme

	$\gamma 1 \leq \delta \leq \gamma 2$	$\gamma 2 < \delta \leq \gamma 3$	$\delta > \gamma 3$
$\delta \geq 0$	Lengthen sleep **K = α1** **Y = £1**	Lengthen sleep **K= α2** **Y = £2**	Lengthen sleep **K= α3** **Y = £1**
$\delta < 0$	Shorten sleep	Shorten sleep	Shorten sleep

For example, if we assume node 'm' had a positive value for δ_m at the beginning of the current cycle, and if the δ_m value was γ', where $\gamma 1 \leq \gamma' \leq \gamma 2$, then the parameters **K** and **Y** would acquire the values of **α1** and **£1** respectively.

4> Calculate (*I*):

For a node 'm', we define the parameter - **Increment = (I_m) = $\mathbf{K}(\delta_m)^2 + \mathbf{Y}$**

5> Update "Calculated Sleep Duration" (CSD):

Initially, at time t=0, $\mathbf{(CSD)_m}$ is set to $\boldsymbol{D_m}$.

We update **CSD** of node 'm' as follows.

$\mathbf{(CSD)_m} = \max\{ \mathbf{(CSD)_m} + \boldsymbol{I_m} , \text{MAXSLEEP}\}$; if $\boldsymbol{\delta \geq 0}$

$= \boldsymbol{D_m}$, otherwise

The node 'm' is then subjected to sleep for $\mathbf{(CSD)_m}$ slots. In every cycle, when a node transitions to the sleep state, the algorithm executes the five steps mentioned above. Obviously the value of $\mathbf{(CSD)_m}$ is updated in every cycle.

6.3 The Static "SLEEP EQUALS DELAY" (SED) Algorithm

A conservative mechanism to facilitate sleeping of a wireless node while still meeting an average packet delay constraint is to allow the node to sleep for a duration no longer than the desired average packet delay requirement. We refer to such a scheme as the "Sleep Equals Delay" (SED) scheme. This scheme is often implemented in commercial WLANs. Given our system model where time is divided into discrete units called slots with a packet arrival rate λ ($0<=\lambda<=1$) in every slot, the above mechanism guarantees that any "practically feasible" average packet delay requirement of a system will be met.

For fairness, the functionalities of the SED scheme, namely the tasks that are performed in every slot, the service order and the service policy are exactly the same as described for the SDM scheme in section 6.2. The only functionality of the SED scheme that is different from the SDM scheme is the procedure by which "Next Sleep Duration"

for a particular node is calculated. To prevent repetition we refrain from re-stating the entire SED algorithm. Instead we only discuss the procedure by which the next sleep duration of a particular node is calculated. In the SED scheme, for any node 'm',

$$(\mathbf{CSD})_m = D_m, \ \forall \delta$$

i.e every node is subjected to a sleep duration equal to its maximum tolerable packet delay in every cycle. The sleep duration is independent of the value of δ or λ.

In the next section, we show via our simulation results that such a static sleep scheduling mechanism though guarantees to deliver the required QoS, is not always the most power efficient scheme especially for non uniform traffic arrivals.

6.4 Simulation Setup and Results

We wrote a computer program to implement both the adaptive and the static "Sleep Equals Delay" (SED) algorithm. We first compare the power consumption of our Adaptive SDM scheme to the lower bound on the average power consumption that we had derived by solving an optimization problem in Chapter 4.6. We consider a Bernoulli traffic arrival pattern with λ=0.1. We assume the following values for the several energy costs: P_a= 1.0, P_{as}= 0.0001, P_{sa}=0.01 and P_s=.001. The values of these energy costs are on a relative scale to each other and have been acquired from [7]. Figure 6.1 represents a comparison between the overall average power consumption per slot of a system comprising of two wireless nodes that are served by a single AP and the lower bound on average power consumption as computed in Chapter 4.6. The X-axis represents the

average packet delay and the Y-axis represents the average power consumption per slot for the entire system. We see from Figure 6.1 that the overall average power consumption of a system running our SDM scheme is fairly close to the theoretical lower bound on power consumption. What is lost in terms of power consumption by our algorithm, is gained in terms of ease of implementation, simplicity, speed and most importantly scalability.

We next compare the power consumption of a system comprising of four wireless nodes executing the SED scheme with the power consumption of a similar system executing the adaptive SDM scheme. In order to demonstrate how our adaptive algorithm dynamically adapts to the packet arrival rate, we consider the following Markov Modulated Bernoulli traffic arrival pattern pictorially depicted in Figure 6.2.

Let us consider a packet generator that is capable of being in one of the three possible states – ON1, ON2 and ON3. Each state is associated with a particular value of λ, say $\lambda 1$, $\lambda 2$ and $\lambda 3$ respectively. The packet generator has the capability to switch from one state to another with specific transition probabilities. Figure 6.2 represents the transition probabilities and the respective values of $\lambda 1$, $\lambda 2$ and $\lambda 3$ that we use in our simulation.

Figure 6.3 represents the overall average power consumption per slot for a system comprising of four wireless nodes that are served by a single AP. We compare the power consumption of such a system when it operates under the SED scheme and then again, under the adaptive SDM scheme. The result in Figure 6.3 has been generated using the Markovian traffic model as depicted in Figure 6.2. In Figure 6.3, the X-axis represents

the average packet delay of the system while the Y-axis denotes the average power consumption per slot of the entire system. The data on the graph wes generated by running the simulation for 300,000 slots and averaged over 10 runs. The values of the different energy parameters that we used in our simulation are as follows: P_a=1, P_{as}=0.0001, P_{sa}=0.01, P_s=0.001. It is evident from Figure 6.3, that the adaptive SDM algorithm consumes significantly less power than the SED scheme. This can be attributed to the fact that the SDM scheme has the capability to assign sleep durations to a node, based on its individual packet arrival rate and average packet delay constraint. It also has the ability to dynamically adapt to changes in packet arrival rates. The adaptive algorithm exploits this feature to subject a node to longer sleep durations when the packet arrival rate for it is lower, thus minimizing the power consumption. The static SED scheme has no such provision and thus cannot utilize the varying traffic condition to its advantage.

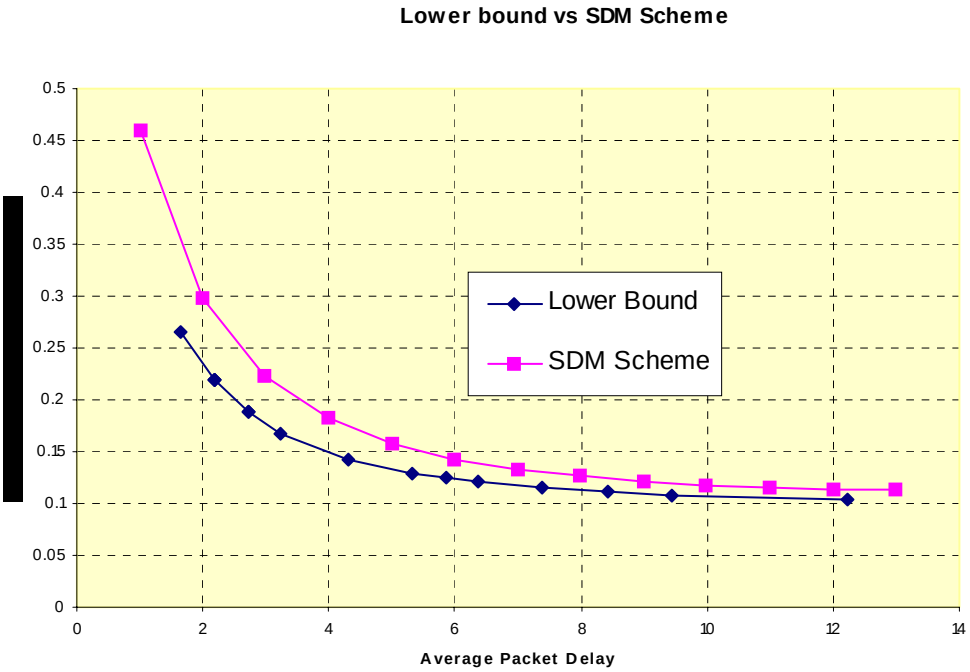

Figure 6.1: Comparison of Power consumption between the SDM scheme and the lower bound

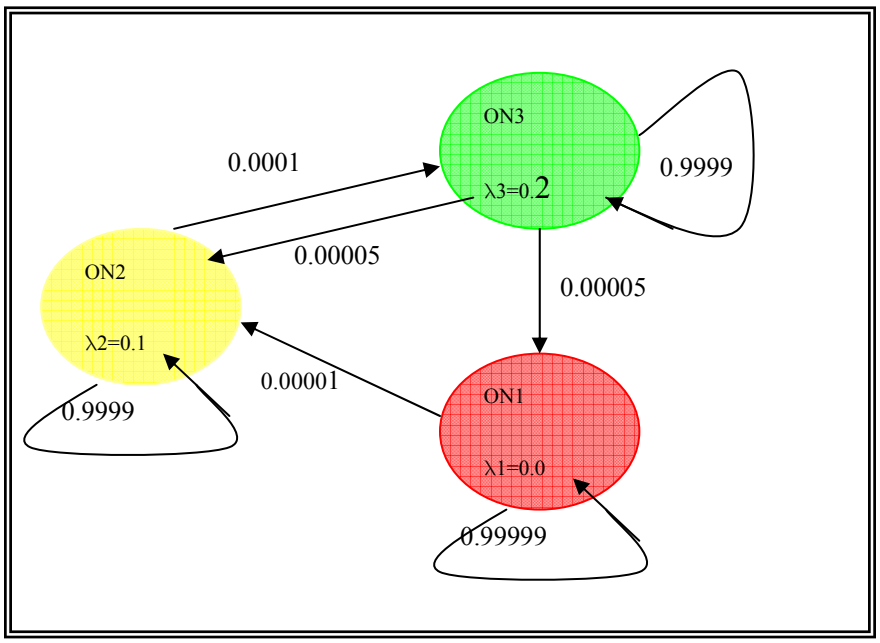

Figure 6.2: A Markov Modulated Bernoulli Packet Arrival Model

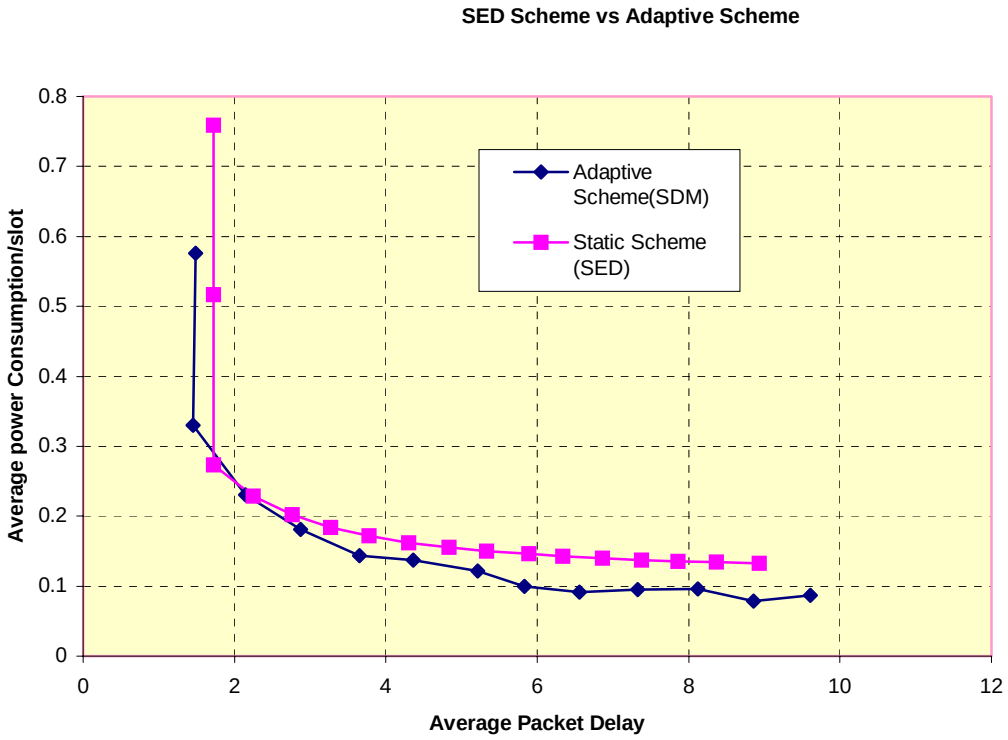

Figure 6.3: Comparison of power consumption between the SED scheme and the SDM scheme for a 4 user system with a MMBP traffic arrival model

7 Closing Remarks

7.1 Research Summary

There are numerous existing and upcoming wireless applications (e.g. 802.11 based WLANS, sensor networks, ad hoc networks, RFIDs, paging systems) that require a number of mostly-dormant wireless nodes to be communicating with each other or with a central base station, intermittently. Since these nodes are battery powered, they are severely energy constrained. Hence in order to save energy and thereby prolong the system lifetime, it makes sense for these nodes to transit to a "sleep" state when they are not communicating with their peers. In this dissertation we are interested in the problem of optimizing the timing and duration of sleep states of a wireless node with the objective of minimizing power consumption with respect to a QoS constraint. The QoS parameter we have focused on is average packet delay.

Unlike other approaches [10][4] that uses a separate, less power hungry wakeup radio that essentially fulfils the role of an out-of-band paging channel designed to periodically listen for wakeup beacons we perform dynamic power management on wireless nodes by subjecting the nodes to sleep for an "optimal" duration. We present three algorithms that do not require the knowledge of current or past packet arrival history of a wireless node. Our algorithms compute the sleep duration of a wireless node as a function of its average packet delay constraint. It learns the traffic behavior over time and adapts its sleep-scheduling mechanism according to the traffic trend coupled with the average packet delay constraint. The wireless node is then allowed to sleep for the pre-

determined “optimal” number of slots. Our adaptive schemes dynamically subject a wireless node to sleep durations that minimize power consumption while meeting an average packet delay constraint. We calculate a theoretical lower bound on the average power consumption of such delay constrained wireless nodes and show that our schemes perform favorably close to the lower bound. In addition, we compare one of our schemes to the static sleep mechanism [40] deployed by most 802.11 vendors and show that for irregular, bursty traffic our scheme outperforms the static sleep scheduling protocol.

7.2 Future Directions

The scheduling framework and approach we have developed in this dissertation is fairly general and allows us to study a broad class of “sleep” scheduling problems. Two major extensions of our work are listed below.

- **Energy Efficient Uplink Transmission :**

In our model we focus on the downlink traffic alone, i.e packet transmission from the transmitter to the receiver. Adding a packet transmission scheme for the uplink traffic which has the same average packet delay constraint as the downlink traffic will be a trivial task for the scheduling schemes we have developed. Whenever a wireless node is in the awake state and ready to receive its packets from the transmitter, it can also send the packets to the transmitter in those same slots. However, if the packet delay constraint for the uplink traffic is different from that of the downlink traffic then such a scheme will not be efficient in meeting the delay constraint. This aspect of minimizing power consumption while meeting the average packet delay constraint for both the uplink and downlink traffic can pose to be a very interesting problem.

- **A Time Varying Channel :**

In our framework, we have assumed a static and perfect channel condition that does not vary with time. Adding a more realistic channel condition – one that is time varying – is desirable. It will also add a new dimension to the problem. Scheduling the sleep time and duration shall not only be a function of the packet delay constraint of a node but also the channel condition as well.

Bibliography

[1] Kanishka Lahiri, A.Raghunathan, Sujit Dey, D.Panigrahi, "Battery driven system design: A new frontier in low power design", *Proc., Intl. Conf. on VLSI Design*, Bangalore, India, Jan 2002.

[2] L.Benini, A.Bogliolo, G. De Micheli, "A survey of design techniques for system level dynamic power management." *Trans. On VLSI Systems, Vol8, No.3*, June 2000.

[3] M.Pedram "Power Optimization and management in embedded systems" *Proc. ASP-DAC 2001*, Yokohama, Japan, Feb2001.

[4] Curt Schurgers, "Energy Aware Communication Systems", *Ph.D. dissertation, Electrical Engineering, University of California, Los Angeles*, Nov. 2002.

[5] Balaji Prabhakar, E.U Biyikoglu, Abbas El Gamal, "Energy-efficient Transmission over a Wireless Link via Lazy Packet Scheduling" *Proc. Infocom'01*, Anchorage, AK, April 2001.

[6] Tajana Simunic, "Energy Efficient System Design and Utilization" *Ph.D. dissertation, Stanford University,* February2001.

[7] L.M. Feeney and M.Nilsson, "Investigating the energy consumption of a wireless network interface in an ad hoc networking environment," *Proceedings of IEEE INFOCOM*, April 2001.

[8] R. Zheng and R. Kravets, "On demand power management for ad-hoc network" *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies*, May 2003.

[9] B.Chen, K.Jamieson, H.Balakrishnan, and R.Morris "Span: An energy-efficient coordination algorithm for topology maintenance in ad-hoc wireless networks," In *Proc. of ACM/IEEE 7th Intl. Conference on Mobile Computing and Networking (MobiCom)*, July 2001.

[10] A.K. Salkintzis and C.Chamaz, "An in-band power saving protocol for mobile data networks," *IEEE Trans. on Communications*, Vol.46, pp1194-1205, September 1998.

[11] S.Gary, P.Ippolito, "PowerPC 603, a microprocessor for portable computers," *IEEE Design and Test of Computers*, vol. 11, no.4, pp. 14-23, Win. 1994.

[12] Ge. Debnath, K. Debnath, R.Fernando, "The Pentium processor-90/100 microarchitecture and low power circuit design," *International Conference on VLSI Design*, pp.185-190, India, 1995.

[13] A.Karlin, M. Manesse, L. McGeoch and S. Owicki, "Competitive randomized Algorithms for Nonuniform Problems", *Algorithmica*, pp. 542-571, 1994.

[14] D. Ramanathan, R. Gupta, "System Level Online Power Managemnet Algorithms", *Design, Automation and Test in Europe*, pp 606-611, 2000.

[15] E. Chung, L. Benini and G. De Micheli, "Dynamic Power Management using Adaptive learning Trees", *International Conference on Computer Aided Design*, 1999.

[16] F. Douglis, P.Krishnan and B.Bershad, "Adaptive Disk Spin-down Policies forMobile Computers", *Second USENIX Symposium on Mobile and Location-Independent Computing*, pp.121-137, 1995.

[17] L.Benini, R. Hodgson and P. Seigel, "System-Level Power Estimation and Optimization" *International Symposium on Low Power Electronics and Design*, pp. 173-178, 1998.

[18] M.Srivastava, A. Chadrakasan, R. Brodersen, "Predictive system shutdown and other architectural techniques for energy efficient programmable computation," *IEEE Transactions on VLSI Systems*, vol.4, no.1, pp.42-55, March 1996 (10).

[19] C.H. Hwang and A. Wu, "A Predictive System Shutdown Method for Energy Saving of Event Driven Computation", *International Conference on Computer Aided Design,* pp. 28-32, 1997.

[20] L. Benini, G. Paleologo, A.Bogliolo and G.De Micheli, "Policy Optimization for Dynamic Power Managemnet " *IEEE Transactions on Computer aided Design*, vol 18, no. 6, pp. 813-833, June 1999.

[21] E.Chung, L. Benini and G. DeMicheli, "Dynamic Power Management for non-stationary service requests," *Design Automation and Test in Europe*, pp77-81, 1999.

[22] T.Simunic, L.Benini, and G. De Micheli, "Event Driven Power Management", *International symposium on System Synthesis*, pp.18-23, 1999.

[23] T. Simunic, L.Benini, G. De Micheli, "Energy Efficient Design of Portable Wireless Devices" *International Symposium on Low Power Electronics and Design*, pp. 49-54, 2000.

[24] Q.Qiu and M. Pedram, "Dynamic Power management based on continuous-time Markov decision processes," *Design Automation Conference*, pp.555-561, 1999.

[25] Q.Qiu and M.Pedram, "Dynamic Power Managemnet of Complex Using Generalized Stochastic Petri Nets," *Design Automation Conference*, pp. 352-356, 2000.

[26] C. F. Chiasserini, P. Nuggehalli, V. Srinivasan, R. R. Rao, "Energy Efficient Communication Protocols" *Proceedings of DAC*, New Orleans, LA, June 2002.

[27] P. Nuggehalli, V. Srinivasan and R. R. Rao, **"**Delay Constrained Energy Efficient Transmission Strategies for Wireless Devices"**,** *Proceedings of Infocom 2002*, New York City, NY.

[28] V. Srinivasan, P. Nuggehalli and R. R. Rao, "An Energy Aware Protocol for Wireless Networks", *Proceedings of IEEE VTC 2001*, Rhode Island, Greece.

[29] V. Srinivasan, C.F. Chiasserini, P. Nuggehalli and R. R. Rao, "Optimal Rate Allocation and Traffic Splits for Energy Efficient Routing in Ad Hoc Networks", *Proceedings of Infocom 2002*, New York City, NY.

[30] K.Sekar, K.Lahiri, A.Raghunathan and S.Dey, "FLEXBUS: A high-performance system-on-chip communication architecture with a dynamically configurable topology", *Proceedings Design Automation Conference*, Anaheim, CA.

[31] Arvind Santhanam, R.L Cruz, "Optimal Routing, Link Scheduling and Power Control in Wireless Multihop Networks," *Proc. Of IEEE Infocom 2003*, San Francisco, CA.

[32] Douglis, F., Krishnan, P., March, B., "Thwarting the power-hungry disk," *Proc. USENIX'94*, San Francisco, CA, pp. 293-306, Jan. 1994.

[33] Helmbold, D., Long, D., Sherrod, B., "A dynamic disk spin-down technique for mobile computing," *Proc. MobiCom'96*, New York, NY, pp. 130-142, Nov. 1996.

[34] Badrinath, B., Sudame, P., "To send or not to send: implementing deferred transmissions in a mobile host," *Proc. International Conference on Distributed Computing Systems*, Hong Kong, pp. 327-333, May 1996.

[35] Stemm, M., Katz, R. "Measuring and reducing energy consumption of network interfaces in hand-held devices," *IEICE Transactions on Communications*, Vol.E80-B, No.8, p. 1125-31, 1997.

[36] Kravets, R., Krishnan, P. "Power management techniques for mobile communication," *Proc. MobiCom'98*, Dallas, TX, pp. 157-168, Oct. 1998.

[37] Lorch, J., Smith, A., "Software strategies for portable computer energy management," *IEEE Personal Communications,* Vol.5, No.3, pp. 60-73, June 1998.

[38] Lettieri, P., Schurgers, C., Srivastava, M., "Adaptive link layer strategies for energy efficient wireless networking," *Wireless Networks*, Vol.5, No.5, Baltzer, pp.339-355, 1999.

[39] Zorzi, M., Rao, R., "Error control and energy consumption in communications for nomadic computing," *IEEE Trans. on Computers*, Vol.46, No.3, pp. 279-289, March 1997.

[40] IEEE Computer Society, "IEEE standard 802.11:wireless LAN medium access control (MAC) and physical layer (PHY) specifications." *The Institute of Electrical and Electronics Engineers*, New York, NY, 1997.

[41] Sennott, L.I. Stochastic Dynamic Programming and the Control of Queueing Systems, New York: Wiley,1999.

[42] Bertsekas, D.P Dynamic Programming and Stochastic Control, Vol.126, NJ:Prentice Hall, 1987.

[43] M.Sarkar, R.L Cruz, "Analysis of Power Management for Energy and Delay Tradeoff in a WLAN," *Proceedings of the Conference on Information Sciences and Systems*, Princeton, New Jersey, March 2004.

[44] M.Sarkar, R.L Cruz, "An Adaptive Sleep Algorithm for Efficient Power Management in WLAN," *Proceedings of IEEE Vehicular Technology Conference*, Spring, Stockholm, Sweden, May 2005

[45] B.E. Collins and R.L. Cruz, "Transmission Policies for Time Varying Channels with Average Delay Constraints" *Proc.1999 Allerton Conference on Communications,Control and Comp.*, Monticello, IL, 1999.