MINIMIZING ENERGY CONSUMPTION IN SENSOR NETWORKS USING A WAKEUP RADIO

BY

MATTHEW JEFFERSON MILLER

B.S., Clemson University, 2001

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2003

Urbana, Illinois

To Leigh Ann, Dad, and Mom

Acknowledgments

Thanks to my wife, Leigh Ann, for her constant support and unwavering confidence in me. I am grateful to my parents for all the support they have given me academically, financially, and spiritually through the years. This work would not have been possible without the meticulous guidance and encouragement of my adviser, Dr. Nitin Vaidya. Finally, I thank God for His blessings on me to make this work possible.

I would also like to acknowledge some financial support for this research was provided by a National Defense Science and Engineering Graduate (NDSEG) Fellowship.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ACK Acknowledgment.

API Application Programming Interface.

ATIM Ad Hoc Indication Message.

CDMA Code Division Multiple Access.

CDMA/CA Carrier Sense Multiple Access/Collision Avoidance.

CPU Central Processing Unit.

CTS Clear-To-Send.

DCF Distributed Coordination Function.

DIFS Distributed Interframe Spacing.

FTP File Transfer Protocol.

HIPERLAN HIgh PEformance Radio Local Area Network.

HTTP HyperText Transfer Protocol.

IEEE Institute of Electrical and Electronics Engineers.

IP Internet Protocol.

MAC Medium Access Control.

NAV Network Allocation Vector.

NFS Network File System.

PCF Point Coordination Function.

PLCP Physical Layer Convergence Procedure.

PSM Power Save Mechanism.

RTS Request-To-Send.

RTT Round Trip Time.

SIFS Short Interframe Spacing.

SMDP Semi-Markov Decision Process.

TCP Transmission Control Protocol.

WLAN Wireless Local Area Network.

Chapter 1

Introduction

The emergence of sensor networks presents many new challenges in wireless ad hoc networks. While the precise application of sensor networks is speculative, certain characteristics are typically assumed. First, the sensors are static after initial deployment (unless placed on a mobile entity [1]). Second, energy is scarce and it is inconvenient or impossible to replenish the energy source frequently.

Because energy should be conserved, power save protocols are needed. This problem can be addressed at each layer of the network stack. Our specific focus is the Medium Access Control (MAC) layer since this gives a fine-grained control to switch the wireless radio on and off. The fundamental question MAC-layer power save mechanisms seek to answer is: *When should a device switch to a low power mode and for how long?*

There are four major sources of energy waste at the MAC layer [2]:

Collisions When packets collide, energy is wasted because the packet, along with associated control overhead, must be retransmitted. Thus, the energy used to transmit and receive the colliding packets is wasted. Collisions are usually reduced by scheduling or deferring for random intervals from medium access.

Overhearing This occurs when a node promiscuously listens to a packet intended for a different destination. In this case, the node could sleep rather than idly listening to the channel.

Control Overhead Aside from the data packet, which has additional header bytes prepended, usually other MAC level packets add overhead. For example, an acknowledgment packet is usually required from the receiver to verify whether the data transmission was a success or

Table 1.1: Characteristics of a sensor radio.

Radio State	**Power Consumption (mW)**
Transmit	81
Receive/Idle	30
Sleep	0.003

failure.

Idle Listening This occurs when a node is listening to the channel, but not transmitting or receiving any data. This process typically consumes much more energy than if the node were to sleep.

Radios typically have four power levels corresponding to the following states: transmitting, receiving, listening, and sleeping. Typically, the power required to listen is about the same as the power to transmit and receive. The sleep power is usually one to four orders of magnitude less. For Mica Mote sensors [3], these power levels are shown in Table 1.1. Thus, a sensor should sleep as much as possible when it is not engaged in communication.

We present a protocol designed for a topology where all sensors are within range of each other. This protocol is then extended to the multiple hop and multiple flow cases. As in previous work [4, 5, 6], we assume that a second radio is available to awake neighbors. This second radio uses much less power via either a low duty cycle [5] or hardware design [6]. It is assumed the second radio is only capable of transmitting a busy tone, rather than actual data. This allows a simpler, more energy efficient design. However, it introduces a problem: each busy tone must wakeup a node's entire neighborhood since the intended receiver's identifier is not encoded on the *wakeup* channel. The main contribution of this work is selectively waking up the *primary* radio at nodes that have previously engaged in communication via rate estimation. Analytically, we derive equations to find the optimal wakeup interval to minimize the energy consumption.

In Chapter 2, we review related work. Chapter 3 describes and analyzes our proposed protocol. Chapter 4 presents simulation results. Finally, we conclude and discuss future work in Chapter 5. In Appendix A, we list the Matlab code for our analysis.

Chapter 2

Related Work

In this chapter, we present an overview of research that has been done in the realm of power saving for ad hoc networks. Our primary focus is decentralized schemes that involve the MAC layer. However, we also present other relevant work that attempts to conserve energy in wireless environments.

In Section 2.1, we describe the power save mechanism in the industry standard, IEEE 802.11, discuss its shortcomings, and review proposed modifications. Section 2.2 presents power save protocols at the MAC layer. These works are most closely related to our work. Section 2.3 gives an overview of power conservation techniques proposed at upper layers in the network stack. Finally, in Section 2.4, we discuss comprehensive surveys on wireless energy research and measurement studies on wireless devices.

Note that this categorization is not intended to be mutually exclusive; some works could easily fit into more than one category. We try to categorize the works with the most similarity.

2.1 IEEE 802.11 Power Save Mechanism

In this section, we will begin by describing the power save mechanism (PSM) specified in IEEE 802.11 [7]. From the description, it will become obvious that this PSM is not well suited for the dynamic nature of ad hoc networks. On of the major issues is the time-synchronization required of nodes in the network. In ad hoc networks, this is a difficult problem. Section 2.1.1 will present some research that addresses the synchronization issue. In Section 2.1.2, we give an overview of some research to improve the performance of 802.11 PSM.

The IEEE 802.11 specification [7] is the standard currently used by commercial Wireless Local Area Network (WLAN) cards. It specifies a MAC protocol for wireless access in both ad hoc environments, called the Distributed Coordination Function (DCF), and centralized systems, called the Point Coordination Function (PCF). Additionally, a power save mechanism is also specified in the standard.

We will now describe IEEE 802.11's PSM. Nodes are assumed to be synchronized and awake at the beginning of each *beacon interval*. After waking up, each node stays on for a period of time known as the *Ad hoc Traffic Indication Message (ATIM) window*. During the ATIM window, since all nodes are guaranteed to be on, packets are advertised that have arrived since the previous beacon interval (or could not be sent in the previous beacon interval). These advertisements take the form of ATIM packets. More formally, when a node has a packet to advertise, it sends an ATIM packet to the intended destination during the ATIM window, following the rules of IEEE 802.11's Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) mechanism. In response to receiving an ATIM packet, the destination will respond with an ATIM-Acknowledgment (ATIM-ACK) packet (unless the ATIM specified a broadcast or multicast destination address). When this ATIM handshake has occurred, both nodes will remain on after the ATIM window and attempt to send their advertised data packets before the next beacon interval, subject to CSMA/CA rules. If a node remains on after the ATIM window, it must keep its radio on until the next beacon interval. If a node does not receive an ATIM or ATIM-ACK (assuming unicast advertisements), it will enter sleep mode at the end of the ATIM window until the next beacon interval.

This process is illustrated in Figure 2.1. Nodes **A**, **B**, and **C** are all within range of each other. The dotted arrows indicate events that cause other events to occur. At t_0, an ATIM window begins and **A** has a packet to send to **B** (the other nodes do not have any data to send). Node **A** sends an ATIM to **B** and **B** replies with an ATIM-ACK. At t_1, the ATIM window ends. Since **C** did not receive an ATIM or ATIM-ACK, it returns to sleep. However, **A** and **B** will remain on and exchange the data packet and corresponding ACK. Even after this data exchange takes place, **A** and **B** will remain on until the next beacon interval at t_2. At this time, **C** wakes up again for the ATIM window (**A** and **B** also remain on during the ATIM window). At t_3, no ATIM or ATIM-ACK has been received, so all three nodes return to sleep.

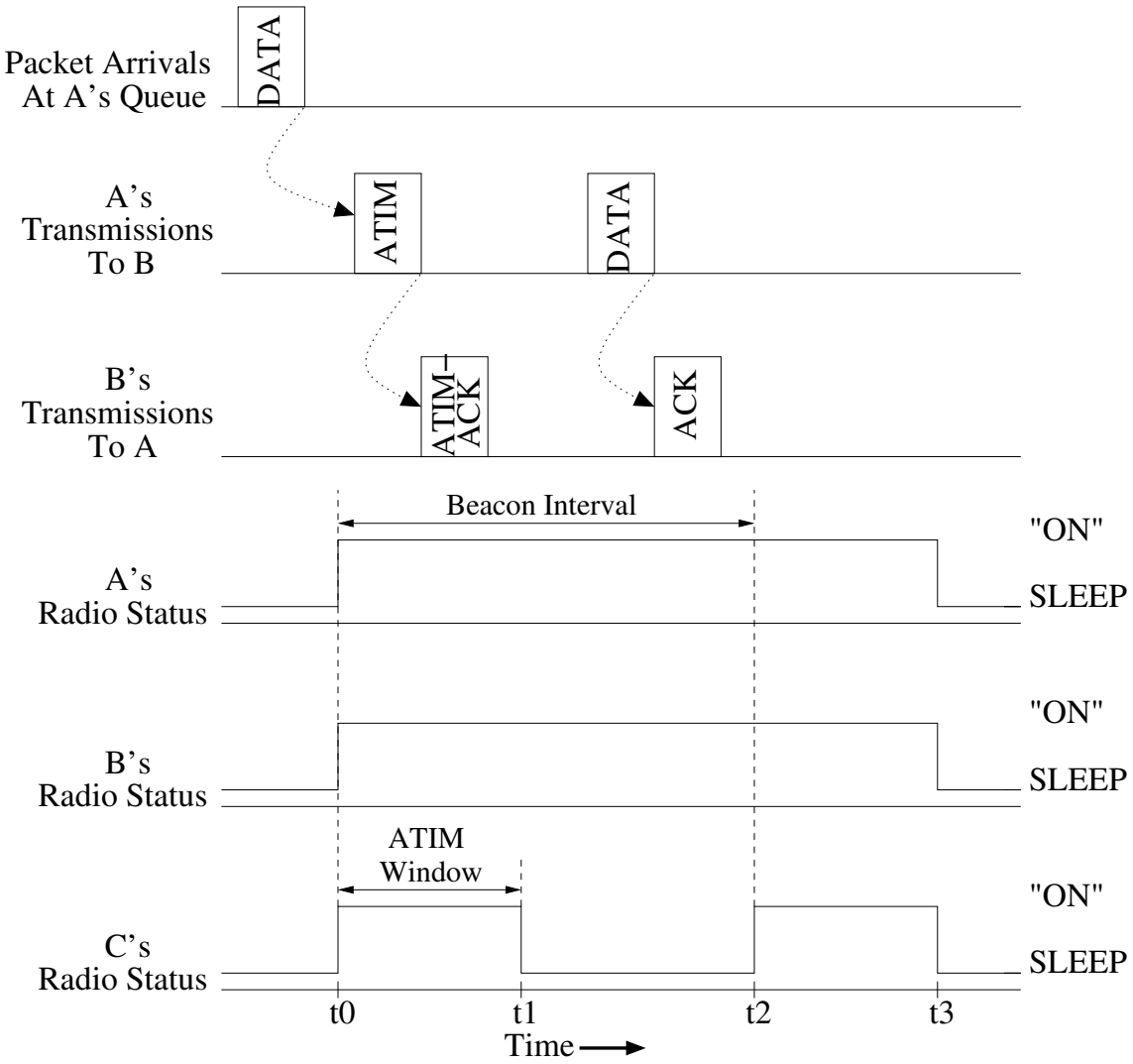

Figure 2.1: IEEE 802.11 power save mechanism.

2.1.1 Synchronization

The 802.11 specification does describe a beaconing mechanism for synchronization, but it is only useful in PCF or when the topology is always fully connected. A broad survey of time synchronization problems, proposals, and research considerations for ad hoc wireless environments is given in [8]. They enumerate the reasons why traditional wired approaches will not work. Their research proposals focus on methods to coordinate time between communicating nodes based on local clocks, rather than striving for a global timescale. The authors also suggest synchronization should be reactive, rather than proactive and adaptive to level of accuracy needed by the application.

A comprehensive analysis of synchronization in 802.11's PSM is given in [9]. They propose three protocols which allow neighbors to advertise to each other by guaranteeing some overlap in their awake windows. The first protocol calls for nodes to be awake for at least half of each beacon interval and alternate their advertisement windows at the beginning and end of intervals. This guarantees overlap but requires a lot of energy. The second approach requires the nodes to only stay awake for a long active interval once every T beacon intervals. During the other $T-1$ intervals, the node will only wakeup for the duration of an advertisement window. The final approach is quorum-based. In this approach, each node picks $2n-1$ out of n^2 intervals (where n is a specified value) in such a way that at least two chosen intervals are guaranteed to overlap with a neighbor's. Each chosen interval, the node will stay awake for the entire interval. During the other intervals, the node will only stay awake for an advertisement window. They cite a proof that guarantees this scheme will allow unsynchronized hosts to overlap. The authors note that broadcast is still difficult in such a scheme. Also, these methods require all nodes to have the same advertisement window and beacon interval lengths. This assumption hinders performance in environments with heterogeneous loads [10].

A similar approach is taken in [11]. Here, the author proposes leaving the nodes up for at least 50% of each interval and including an advertisement window at the beginning and end of the active period. This guarantees discovery and then a node can determine the phase of an intended receiver and adjust its cycle accordingly. The major disadvantages of this approach are the device can only sleep for less than half the time and packet latency is increased.

The authors in [12] focus more on the lack of centralized coordination than synchronization. The problem they solve is neighbor discovery in sensor networks over a potentially large time scale where a node must be powered down most of the time to conserve energy. The basic idea is a nondeterministic protocol where a node is awake for a randomly chosen X time slots out of Y (where $X \ll Y$). Each node enters a listen or transmit mode with a specified probability such that, with high probability, neighbors will discover each other over the time interval.

In [13], a deterministic protocol for neighbor discovery is presented. Using combinatoric theory, sleep schedules are chosen such that every pair of neighbors is guaranteed to overlap for at least one slot. Thus, if a node is awake X out of Y slots, energy is reduced and all neighbors should be

able to contact each other within Y slots to synchronize their communication.

2.1.2 Proposed Modifications to 802.11 PSM

Approaches in this category propose changes to the DCF PSM of 802.11 to allow for increased time in the sleep state.

The approach in [14] is similar to PAMAS [4] (discussed in Section 2.2.1), but only uses one radio and is designed around the 802.11 standard. The simple solution is for a node to enter the sleep state whenever it overhears a Request-To-Send (RTS)/Clear-To-Send (CTS) handshake followed by a data packet for which it is not the target. This is an intuitive approach because in this scenario, it would be impossible for the node to access the medium. However, this approach may require frequent transitions between the idle and sleep state which uses a non-trivial amount of time and energy. The paper does a simple test on the extra energy required for the transitions. The values used for the energy transition are not justified by any data. Another problem is in a lightly loaded network, most of the nodes will still have to use a significant amount of energy to remain in the idle state.

The next two approaches require adjustments to the ATIM window beaconing process. In [15], the ATIM window size is adjusted dynamically per node as a function of how many packets a node has to send, the window size of neighboring nodes, and the reception of packets which indicate a neighbor could not reach the node with its current window size. Energy is further reduced by allowing nodes to sleep when all advertised packet have been received instead of remaining awake for the duration of the beacon period. One limitation of this approach is that every neighbor must explicitly advertise pending packets for a destination or the the destination will sleep before the source can transmit the data. In 802.11, the source would know the destination would be awake for the duration of the beacon interval and therefore be able to transmit data after the advertising source has finished.

The approach in [16] is to modify the header of RTS, CTS, data and ACK packets to convey the information typically provided in the ATIM window. This allows the ATIM window to be eliminated by exchanging data packets in the common awake window instead of control packets. This improves 802.11's power save in highly loaded networks because channel capacity is not wasted on the ATIM

window and the extra energy of transmitting control packets is eliminated. The latency is increased, however, because a node will power down more aggressively.

2.2 Power Conservation at the MAC Layer

We will now present research which attempts to save power at the MAC layer, primarily by selectively switching the wireless card into sleep mode. In Section 2.2.1, we investigate protocols that use multiple radios or channels. Section 2.2.2 presents work developed specifically for sensor networks. In Section 2.2.3, a popular technique is explored which selects a small subset of the nodes in the network to keep their wireless cards on, while all others enter the 802.11 PSM mode. The solutions attempt to maintain connectivity such that all nodes are within one hop of a node that is on (similar to a connected dominating set problem). Finally, Section 2.2.4 considers other protocols which attempt to adjust sleep schedules based on the traffic in the network.

2.2.1 Multiple Radio Architectures

These solutions investigate what can be gained by adding an second radio or channel to the WLAN card. The PAMAS protocol [4] adapts basic mechanisms of IEEE 802.11 [7] to a two-radio architecture. PAMAS allows a node to sleep to avoid overhearing a packet intended for a different destination or to avoid interfering with another node's reception by transmitting. The control channel is used to exchange RTS/CTS packets, emit busy tones to eliminate interference, and probe ongoing communications for their duration. Whenever a node awakes and detects another transmission, it can probe the control channel to determine how much longer this transmission will continue. Unlike our work, it ignores the idle listening problem.

A similar approach is used in [17]. Here, the nodes will transmit a receive or send busy tone upon receiving a RTS or CTS, respectively. This busy signal addresses the problem of nodes moving into an area where communication is occurring after missing the initial RTS/CTS exchange. It also reduces the probability of control packet collisions when traffic load and propagation delays are high.

The PicoRadio [18, 19, 20, 6] design uses a low-power wakeup channel. We apply this technique in our protocol. A MAC protocol has been designed which allows nodes to wakeup a neighbor

Table 2.1: Target specifications for PicoRadio hardware.

	Wakeup Radio	**Data Radio**
Transmit Power (μW)	1000	1000
Receive/Idle (μW)	50	1000
Sleep (μW)	—	0
Bitrate	$O(100)$ bps	50 kbps
Range (m)	10	10
Transition Energy, sleep→idle	—	1 μs × 1 mW
Transition Energy, idle→sleep	—	1 μs × 0 mW

when data needs to be sent. When a node wishes to send data, it encodes the neighbor's receiving channel in a beacon on the wakeup channel. The nodes then communicate over the high powered data channel of the receiver. This design uses a CDMA scheme which requires each neighbor within a 2-hop range to be assigned a unique channel and discover and maintain the channel IDs for each 1-hop neighbor, which is difficult in a distributed setting. Also, the channel ID is encoded in the wakeup signal, which increases the hardware complexity. Our approach could be adapted to similar hardware which uses a busy tone on the wakeup channel. Table 2.1 shows the target specifications for the PicoRadio hardware[1].

A wakeup channel is also used in [21]. Here, a low-power radio is integrated with a PDA. The protocol is implemented from off-the-shelf hardware. The devices register their presence with a server via a proxy. When another node wishes to communicate, the proxy will send a short wakeup packet over the low power, low bit rate channel. This will cause the high powered radio to turn on so data communication can begin. However, this protocol is designed for systems with centralized access points or proxies and not fully distributed networks.

A theoretical approach to multiple channel power save is investigated in [22]. Here a base station uses RF ID tags to wakeup devices which could be in any one of L sleep states. Each sleep state uses less power in steady state, but requires more delay and power when transitioning to the fully awake state. The idea that is a device will remain in a state at least long enough to get a positive energy gain before transitioning to the next lower power state. The base station tracks this cycle for each device and when it has data to send, it waits as long as possible before waking the device

[1]These values were obtained in an email correspondence with Brian Otis, the lead graduate student for the hardware design.

and transmitting subject to QoS requirements. When the base station wishes to wake a device up, it pages all devices in that current sleep state. The non-target devices in the paged sleep state will then start the sleep cycle again once they determine the data is not for them. This allows the size of the paging message to be on the order of the number of sleep states instead of the number of nodes.

STEM [5, 23] is a two-radio architecture which achieves energy savings by having the primary radio sleep until communication is necessary while the wakeup radio periodically listens using a low duty cycle. When a node has data to send, it begins transmitting continuously on the wakeup radio long enough to guarantee that all neighbors will receive the wakeup signal. A variant of STEM has been proposed [5] that uses a busy tone, instead of encoded data, for the wakeup signal. Our protocol is similar to STEM, but achieves greater energy savings by periodically listening on the primary channel and buffering packets.

2.2.2 Sensor Network-Specific Protocols

The works in this section design MAC protocols which exploit sensor network properties, such as static nodes, to save energy.

In [24], the authors assume a sensor network where there is one sink and many sources. Therefore, a logical tree organization is possible via beaconing. The authors propose an adaptive rate control mechanism at the MAC layer. The nodes use an RTS/CTS handshake to detect contention on the channel. Assuming the application layer is sending data at rate S, the MAC layer will send at a rate of $p \times S$, where $p \in [0, 1]$. Note that this approach requires the nodes to control the rate at which they originate data to avoid excessive drops due to buffering. Such a scheme is feasible in sensor networks because the input sensing intervals can be adjusted. When the node successfully gains access to the channel, p is increased additively by α. When the node fails to gain access, p is multiplicatively decreased by β. Priority is given to route-through traffic that originates at a node's child since the loss of a multihop packet is a bigger waste of resources than dropping a packet at the source. To further avoid collisions, when a node hears its parent finish a data transmission, it will estimate when the grandparent will finish sending data and defer from transmitting during this period. This avoids interference at the node's parent.

S-MAC [2] is a protocol developed specifically to address energy issues in sensor networks. It reduces energy consumption at the expense of fairness and latency. S-MAC uses a simple scheduling scheme to allow neighbors to sleep for long periods and synchronize wakeups. A neighborhood of nodes synchronize by one node broadcasting a duration of time it will be awake. After this period, the node will sleep for the same amount of time. Each node will follow this sleep/awake schedule also and broadcast it to their neighbors. If a node receives two different schedules, it will remain awake according to both schedules. In S-MAC, nodes enter sleep mode when a neighbor is transmitting and fragment long packets to avoid costly retransmissions. After each fragment, an ACK is sent by the receiver so nodes waking up in its vicinity will sense the transmission. S-MAC is designed to save energy on a single radio architecture. While this approach does allow packets to be buffered, it provides no mechanism to communicate with the receiver on-demand. Also, S-MAC uses a fixed sleep interval regardless of traffic.

2.2.3 Subset of Low Power Nodes

Work in this section investigates how a subset of the nodes in a system can enter a low power state without significantly degrading the performance achievable if all nodes were to remain in high power mode. The AFECA algorithm [25] allows nodes to sleep based on the size of their neighborhood. The idea is if node density is large, then more nodes can sleep without greatly increasing the latency of data flows. GAF [26] assumes the nodes have some location information and form virtual grids. The size of the grids is chosen such that the nodes in two adjacent grids are equivalent with respect to forwarding packets. Then, within each grid, a discovery protocol tries to ensure that most of the time one node remains active while the rest enter a low-power state. As mobility increases, the discovery process should be more frequent.

The goal of SPAN [27] is to save energy while not degrading the latency and throughput achievable in 802.11 without power save mode. This protocol operates between the MAC and routing layers. The system allows all nodes to enter power save mode except for elected *coordinators*. At the MAC layer, nodes periodically exchange *hello* messages which contain its set of neighbors, coordinators, and whether or not it is a coordinator. Nodes will then elect themselves coordinators if their neighbors would get better connectivity by it doing so. A random delay is introduced before

nodes declare themselves coordinators. This delay varies inversely with the amount of connectivity that would be achieved and inversely with the amount of energy remaining at the node. For fairness, the coordinators will periodically withdraw.

The LEACH protocol [28] is designed specifically for sensor networks that frequently send data to a single sink. The idea is that certain nodes are selected as *cluster-heads* and nearby sensors will send data to them at a presumably low transmit power. The cluster-heads will then aggregate the data, compress it and transmit it at potentially higher power to the data sink. Time is divided into rounds and the cluster-heads are rotated such that a node will fulfill this role on average every $\frac{1}{p}$ rounds (where p is an input parameter to the system). When a new cluster-head is chosen, it broadcasts to all nodes which have joined its cluster a TDMA schedule for transmissions and a CDMA channel to use to avoid interference with nearby clusters.

2.2.4 Adaptation to Traffic

Like our protocol, work in this section attempts to adjust nodes' wakeup schedules based on the pattern of traffic they are receiving.

T-MAC [29] extends S-MAC by adjusting the length of time sensors are awake between sleep intervals based on communication of nearby neighbors. Thus, less energy is wasted due to idle listening when traffic is light. However, T-MAC is still limited by a one-radio architecture.

In [30], energy is also saved by adjusting to traffic. The protocol works with on-demand routing and uses 802.11's PSM when a node is not engaged in sending, receiving, or forwarding data. When a node is communicating, soft-timers are used to transition the node to an idle listening mode which reduces latency and preserves throughput better than only using 802.11's power save. However, the timers do not adjust to the traffic rate, so if traffic is not frequent enough to refresh the timers, the benefits of the protocol are lost. Nodes must promiscuously listen to the packets of neighbors to determine if they are disconnected on in power save mode.

In [31], renewal theory is used to analyze schemes where a mobile device wakes up to receive packets from a base station with a queue of size zero or ∞. The analysis allows arbitrary probability distributions for the interarrival times of packets. Similarly, our protocol attempts to intelligently wakeup to reduce energy consumption. However, our protocol is different because it benefits from

a second radio. Our protocol does not need to drop packets when the queue is finite and fills up while the receiver is sleeping. With the second radio, the receiver can be awakened on-demand in this situation. Also, the renewal theory scheme (with a queue size of ∞) requires traffic to follow a two-state Markov chain model with geometric arrivals. Our work is more general since it does not limit traffic to being in one of two states. Also, our protocol can operate in an environment with contention, whereas the renewal theory work does not work with contention on the medium.

2.3 Power Conservation at Upper Layers

This section focuses on techniques that have been used above the MAC layer to reduce wasted energy. A vast research area exists in power aware routing. We cover a small subset of network layer research in Section 2.3.1. Section 2.3.2 shows how energy can be conserved at the transport layer. Finally, Section 2.3.3 gives an overview of some protocols proposed to work at the application layer.

2.3.1 Network Layer

Power aware routing focuses on modifications that can be made to ad hoc routing protocols to conserve energy. Since we are primarily considering MAC layer solutions to energy consumption, this section is only intended to be a brief overview of some basic ideas in this vast field. The seminal paper in this area is [32]. The authors recognize that most work in ad hoc routing only considers the shortest path metric in choosing paths. They propose that energy-related metrics should also be considered such as: minimizing the energy needed to transmit and receive packets, maximizing the time it will take for the network to partition and minimizing the variance in power levels at the nodes.

Another work proposes link error rates should be considered in the path selection [33]. This is intuitive because a high error rate for a link will require more retransmissions and hence more energy. The authors of [34] present some practical modifications to existing ad hoc routing protocols to achieve better energy usage. They basic idea is to include power information in control packets to allow the source to choose a path based on minimizing the power. However, for such a scheme to work, the energy information needs to be tracked frequently for changes. Therefore, they propose

disabling all route caching and that nodes actively snoop the transmissions of others to try to inform sources when a lower energy path becomes available. The disadvantage of their approach is snooping is impractical when the MAC layer is practicing power save and the elimination of cached routes would probably cause more broadcast packets during the route discovery process. The only tests they consider are those with very low mobility.

2.3.2 Transport Layer

There has been relatively little work at this layer compared to the MAC, network, and application layers. Research at this layer focuses on centralized networks, rather than ad hoc networks. In [35], the authors show how 802.11's power save mode can hinder latency when short Transmission Control Protocol (TCP) transfers occur. Since 802.11 calls for a 100 *ms* sleep period between traffic advertisements, when the Round Trip Time (RTT) of the connection is short, the latency will be rounded up with a granularity of 100 *ms* during the slow-start phase. This occurs because the access point will finish sending buffered data before new data arrives from the server. This continues until the send window is sufficiently large to keep enough in-flight data. The solution they propose it to allow the device to stay awake for a short time after sending data in case the RTT is small. The sleep time will also increase as the interval between data transmissions at the mobile node grows. They derive a method such that the slowdown experienced is bounded by $(1+p)$ times the base RTT.

2.3.3 Application Layer

Due to the large amount of work in this area (particularly among operating systems groups), we only present a brief overview.

In [36], the interarrival time of HyperText Transfer Protocol (HTTP) and Telnet requests is modeled by a Pareto, rather than exponential, distribution. The goal is to determine when the WLAN card be be turned off by considering the interarrival time of user requests. Therefore, it is assumed the mobile initiates all connections in which it is involved. As the time between requests increases, the scheme transitions through time-indexed Semi-Markov Decision Process (SMDP) model states. At each state, it will turn off with probability p, where p is a monotonically

increasing value with respect to the states.

The authors of [37] propose a mechanism to allow applications to adapt to wireless delays. In their paradigm, the access point is a slave and the mobile device is the master. The access point will buffer data and the mobile will periodically wakeup and query for data.

In [38], the authors propose an Application Programming Interface (API) which allows applications to give hints about communication patterns. Additionally, history of previous access patterns is maintained. Based on this information, the WLAN card intelligently decides whether to enter power save mode. The tests mainly consider network file systems. It is unclear how the system would perform if applications with different access patterns are using the WLAN card simultaneously (e.g., Network File System (NFS), File Transfer Protocol (FTP), and HTTP all operating at the same time).

2.4 Surveys on Wireless Energy Consumption

These papers just present overviews of considerations that should be made with respect to energy in wireless environments. Primarily, this gives a starting point for readers interested in a more comprehensive survey of research in this area. Section 2.4.1 gives references to works which measure the energy consumption of wireless devices.

In [39], the authors look specifically at sensor networks. First, they analyze how the power is used in such a device. To reduce energy at the radio, they propose short-circuiting packet forwarding to avoid the use of the Central Processing Unit (CPU) as well as modulation schemes and error control schemes. At upper levels, they suggest load balancing, overhead reduction, and only turning the nodes on a small fraction of the time at the expense of increased latency.

An overview of power-saving mechanisms in 802.11 and another standard, HIgh PErformance Radio Local Area Network (HIPERLAN), is given in [10]. For 802.11, they conclude the ATIM window and beacon intervals must be dynamic as different topologies and traffic loads make static values impractical. In [40], energy research at every layer of the protocol stack is investigated. It provides a good survey for readers interested in the broad spectrum of energy research directions.

2.4.1 Energy Consumption Measurements

In [41], some power measurements are done on cards before the 802.11 standard was widely implemented. The basic conclusion is the idle state is responsible for a majority of the energy usage of the cards and therefore techniques are necessary to enter low power states as much as possible.

Both [42] and [43] report similar results of oscilloscope measurements of 802.11 WLAN cards. 802.11's power save mode is not considered, but they do note that the cards tend to enter an unspecified state when a node overhears a packet transmission for which it is not a target. This state uses less power than the idle state, but still significantly more than the sleep state. The authors also determine that the promiscuous receive mode used in many ad hoc routing protocols is expensive in terms of energy compared to simply discarding the data. The study also looks at 2 Mbps and 11 Mbps cards. At higher bit rates, the promiscuous receiving is not much more expensive than discarding the packet. It is noted that with higher bit rates, the fixed control overhead usually dominates since this data must be sent at the lower bit rate. Broadcast packets also must be sent at the lowest bit rate, making them much more expensive than unicast packets in terms of energy when multiple bit rates must be considered.

Chapter 3

Protocol Description and Analysis

3.1 Triggered Wakeups with Queuing

As mentioned previously, two channels are assumed: primary and wakeup. The primary channel is used for sending data and control packets, whereas the wakeup channel is used to wakeup neighbors. For the rest of the paper, we assume that the wakeup radio achieves low power consumption via a duty cycle. That is, a node will listen for a busy tone on the wakeup channel for τ_1 time, then sleep for τ_2 time ($\tau_1 \ll \tau_2$). The sender of a wakeup signal must transmit for $2\tau_1 + \tau_2$ time to guarantee all neighbors hear the wakeup signal. The duty cycle of the wakeup channel is defined as $\frac{\tau_1}{\tau_1+\tau_2}$. Thus, a lower duty cycle reduces idle listening energy, but increases the delay to wake a node's neighborhood. A queue threshold, L, is specified for the protocol. This threshold could be used to control delay or limit the storage usage on a sensor. For simplicity, L is expressed in packets and all data packets are the same size[1]. When the queue holds L packets, a wakeup signal must be sent so the queue size can be reduced by transmitting packets to a receiver immediately. We refer to this as a *full wakeup* because all sensors within one hop of the sender, after detecting the signal, must wakeup their primary radio and listen on the primary channel until a *filter packet* is sent (on the primary channel) to indicate which sensor's radio should remain on for reception. The other neighbors then return to sleep. To avoid costly full wakeups, a sensor estimates the rate at which it is sending data and tries to schedule a *triggered wakeup* with a receiver T seconds after its previous data transmission. Figure 3.1 illustrates this concept with a fixed T value. The dotted

[1] Alternatively, L could be specified in bytes.

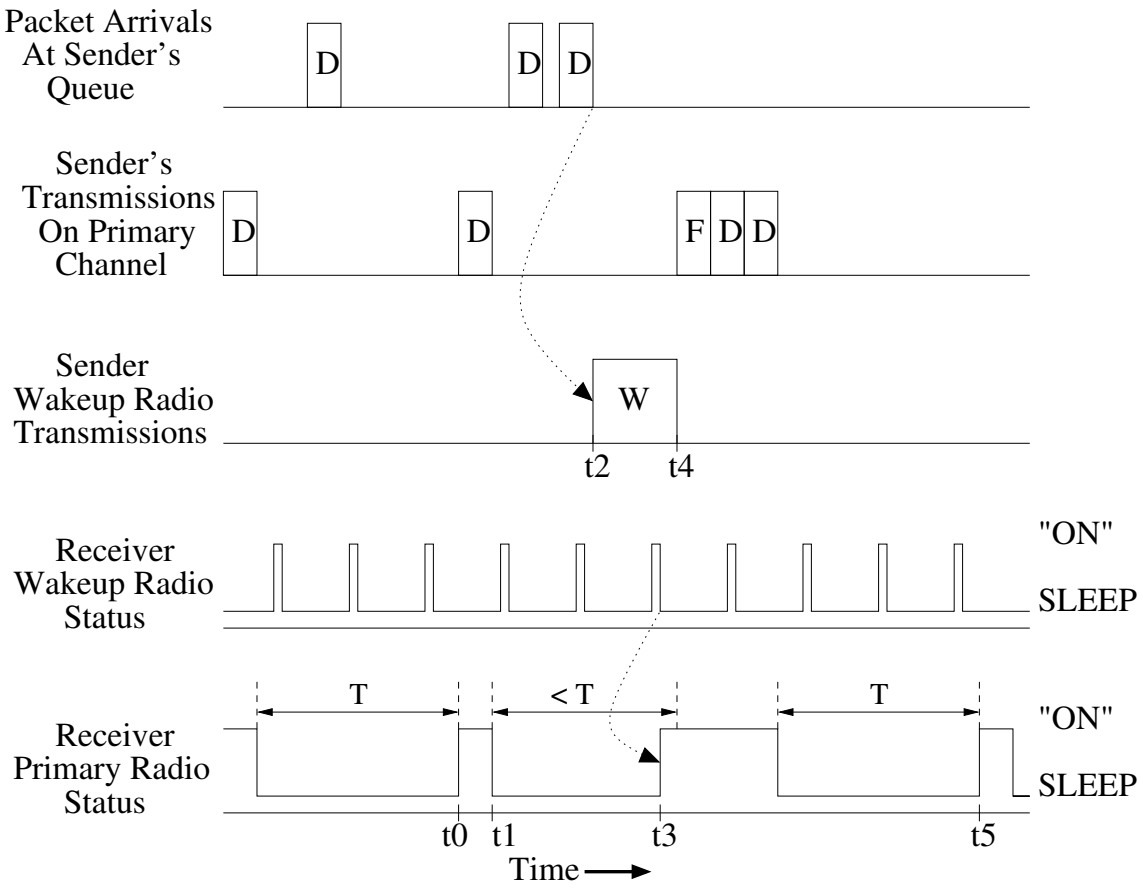

Figure 3.1: Static T and $L = 2$ (**D** = data packet, **F** = filter packet, **W** = wakeup signal).

arrows represent a "causes" relationship between events. At t_0, a triggered wakeup occurs T time after the last transmission, even though the sender's queue contains less than L packets. A full wakeup begins at t_2 because the sender's queue reaches size L. At t_4, all neighbors are guaranteed to have their primary radios on, so a filter packet (shown as **F** in the figure) and L data packets (shown as **D**) are sent on the primary channel. Unlike the figure, our protocol will dynamically adjust T since the rate is not known in advance and may vary with time.

Intuitively, if T is too small, the sender and receiver waste energy by waking up when the queue has no packets. This is called an *empty triggered wakeup* and shown in Figure 3.1 at t_5. An empty triggered wakeup results in idle listening because the primary radios stay on long enough, say T_{thresh} duration, to make sure no data is available (T_{thresh} is not shown in Figure 3.1). Thus, they are on for T_{thresh} after doing a triggered wakeup or sending/receiving a packet. If no data is

sent/received within T_{thresh} time, the primary radios return to sleep. Our goal is to find the optimal T value, T_{opt}, for a given data rate, which minimizes the energy consumption. We assume that both the sender and receiver sleep until a triggered or full wakeup occurs. However, the protocol could easily be modified for the case where the sender is always awake (e.g., a base station).

Initially, no triggered wakeup is scheduled and a full wakeup occurs when the queue contains L packets[2]. Another timer needs to be used to make sure a packet does not remain in the queue indefinitely if the sender stops generating packets[3]. The sender will piggyback its chosen T value (in *ms* in our simulations) on each data packet sent. The sender and receiver will then schedule a triggered wakeup T time in the future, taking into account transmission delay. If no more data is sent or received for T_{thresh} time, the sensors will return to sleep and wakeup $T - T_{thresh}$ time later. A minimum value, T_{min}, is specified for T such that $T_{min} > T_{thresh}$. We describe how T is adjusted in Section 3.2.1.

From our protocol description, we see that STEM [5] is a special case of our protocol with $T = \infty$ and $L = 1$. In STEM, each packet arrival causes a full wakeup, whereas our protocol avoids some full wakeups by using triggered wakeups. Our protocol is different from T-MAC [29], and similar protocols, which adjust the time a radio is on once it enters the idle state. Our protocol tries to sleep as soon as possible after data communication and predict when it should next wakeup based on previous traffic patterns.

Note that our protocol is different from previous work which attempts to adjust how long nodes stay awake after a communication event before returning to sleep (i.e., adjust T_{thresh} in our protocol). This technique is popular in research to efficiently spin-down hard disks [44, 36, 45]. Figure 3.2 shows this technique applied to our protocol with no triggered wakeups. The dotted arrows show a "causes" relationship between events. At t_1, the receiver sleeps because $(t_1 - t_0) = T$, where T is the specified threshold for when to return to sleep (note, this is a different definition than the T used in throughout the rest of the paper and corresponds to T_{thresh} in our protocol). Thus, the next packet to arrive, at t_2, causes a full wakeup.

[2] L is not necessarily equal to the capacity of the queue.

[3] The simulated flows do not test this because packets never cease being generated.

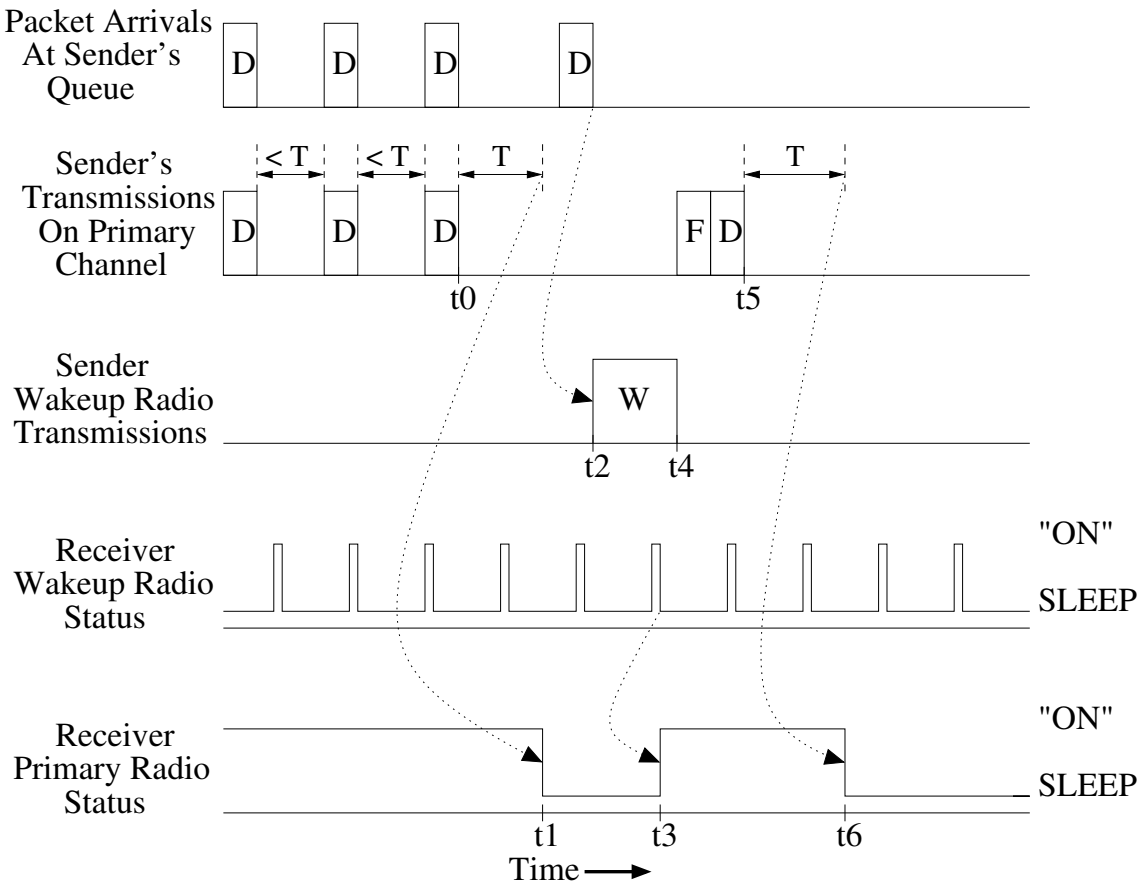

Figure 3.2: Protocol with idle timeout (**D** = data packet, **F** = filter packet, **W** = wakeup signal).

3.2 Energy Analysis of Triggered Wakeups

To find T_{opt}, we derive equations for the expected energy consumption per bit. We make some simplifying assumptions in the analysis. First, it is assumed that there is one sender transmitting to one receiver among N sensors. The remaining $N-2$ nodes do not send or receive any data. Second, we assume once a sensor starts sending a wakeup signal or does a triggered wakeup, only packets in the queue *at the beginning of the wakeup* are sent. Thus, exactly L packets are sent for a full wakeup and at most $L-1$ packets are sent for a triggered wakeup. We remove this constraint in the simulations. We leave out idle energy consumed during backoff periods and due to collisions to keep the analysis tractable. These terms are relatively independent of the T value in our protocol. Our parameters, shown in Table 3.1, are based on Mica Motes and 802.11 (we use the power values shown in Table 1.1). The RTS, CTS, and ACK packet sizes and contents

Table 3.1: Protocol parameter values.

Parameter	**Value**
Physical Layer Header ($PLCP$)	4 bytes
Network Layer Header (IP)	20 bytes
MAC Layer Header (MAC)	32 bytes
Data Size ($DATA_{size}$)	30 bytes
Bytes in each Data Packet	$DATA_{size} + MAC + PLCP + IP$
Bytes in a Filter Packet	$33 + PLCP$
Bytes in a RTS Packet	$20 + PLCP$
Bytes in a CTS Packet	$14 + PLCP$
Bytes in a ACK Packet	$14 + PLCP$
Bitrate	40 kbps
T_{thresh}	20 ms
T_{min}	50 ms
T_{DIFS}	50 μs
T_{SIFS}	10 μs
T_{prop}	2 μs
T_{trans_on}	0 μs
T_{trans_off}	0 μs
τ_1	1 ms
τ_2	299 ms

are unmodified from the 802.11 standard. The average power used while sleeping for the two-radio architecture, P_{sleep}, is set according to Equation 3.1. When a node is sleeping, its data radio will only consume $Radio_{sleep\_power}$. The wakeup radio will consume $Radio_{sleep\_power}$ for $\frac{\tau_2}{\tau_1+\tau_2}$ of the time and $Radio_{idle\_power}$ for the remaining $\frac{\tau_1}{\tau_1+\tau_2}$ of the time.

$$P_{sleep} = Radio_{sleep\_power}\left(\frac{\tau_2}{\tau_1+\tau_2}+1\right) + Radio_{idle\_power}\left(\frac{\tau_1}{\tau_1+\tau_2}\right) \tag{3.1}$$

We set τ_1 and τ_2 to be 1 ms and 299 ms, respectively. These values are similar to those used in [46] and give a duty cycle of $\frac{1}{300}$. With these power levels and the selected wakeup radio duty cycle, according to Equation 3.1, P_{sleep} is:

$$P_{sleep} = 0.003\left(\frac{299}{300}+1\right) + 30\left(\frac{1}{300}\right) \approx 0.106 \text{ mW} \tag{3.2}$$

E_{bit} is the cumulative energy used by all nodes (in Joules) per data bit delivered. Recall that L is our queue threshold and N is the number of sensors. Let R be the packet arrival rate. The interarrival time of packets is assumed to have an exponential distribution. Later, we consider time-varying rates. First, we derive p_f, the probability a full wakeup occurs. Let X be the length of time until the L-th packet arrival and Y be the number of packet arrivals that occur over time T (i.e., $Y \sim \text{Poisson}(\lambda = RT)$).

$$\begin{aligned} \Pr[X \geq T] &= \Pr[Y < L \text{ on the interval } [0, T)] \\ p_f &= \Pr[X < T] \\ &= 1 - \Pr[X \geq T] \\ &= 1 - \sum_{i=0}^{L-1} \frac{(RT)^i}{i!} e^{-RT} \end{aligned} \tag{3.3}$$

Equation 3.3 comes from the Poisson distribution [47]. Let p_e be the probability of an empty triggered wakeup and $p_{\overline{f+e}}$ be the probability of a non-empty triggered wakeup. We have:

$$p_e = e^{-RT} \tag{3.4}$$

$$p_{\overline{f+e}} = \sum_{i=1}^{L-1} \frac{(RT)^i}{i!} e^{-RT} \tag{3.5}$$

Next, let $Q_{\overline{f+e}}$ be the expected number of packets in the queue at time T for a non-empty triggered wakeup. Thus,

$$\begin{aligned} Q_{\overline{f+e}} &= \frac{\sum_{i=1}^{L-1} i \frac{(RT)^i}{i!} e^{-RT}}{\sum_{i=1}^{L-1} \frac{(RT)^i}{i!} e^{-RT}} \\ &= \frac{\sum_{i=1}^{L-1} i \frac{(RT)^i}{i!}}{\sum_{i=1}^{L-1} \frac{(RT)^i}{i!}} \end{aligned} \tag{3.6}$$

We need to find T_{sleep_full}, the expected sleep time given a full wakeup occurs. Let Z be the expected time of the L-th packet arrival. The gamma distribution models the waiting time until the L-th event occurs for events that follow a a Poisson distribution [48]. Thus, $T_{sleep\_full} = \text{Ex}[Z|Z \leq$

$T]$ and $Z \sim \text{Gamma}\left(\alpha = L, \beta = \frac{1}{R}\right)$. We let $f(z)$ denote the probability density function of the gamma distribution [48]:

$$f(z) = \frac{z^{\alpha-1}e^{-z/\beta}}{\Gamma(\alpha)\beta^{\alpha}} \tag{3.7}$$

Where the complete gamma function, $\Gamma(L)$, is defined to be [49]:

$$\Gamma(L) = \int_0^{\infty} x^{L-1}e^{-x}\ dx \tag{3.8}$$

For T_{sleep_full}, we have:

$$\begin{aligned} T_{sleep\_full} &= \frac{\int_0^T zf(z)\ dz}{\int_0^T f(z)\ dz} \\ &= \frac{\int_0^T \frac{R^L}{\Gamma(L)} z^L e^{-Rz}\ dz}{\int_0^T \frac{R^L}{\Gamma(L)} z^{L-1} e^{-Rz}\ dz} \\ &= \frac{\int_0^T z^L e^{-Rz}\ dz}{\int_0^T z^{L-1} e^{-Rz}\ dz} \end{aligned} \tag{3.9}$$

Now, we can express the expected energy consumed for each type of wakeup. Let

$$E_{pkt} = E_{MAC\_RX} + E_{data\_RX} + E_{MAC\_TX} + E_{data\_TX} \tag{3.10}$$

be the energy required to send *and* receive one packet, where E_{data_TX} and E_{data_RX} is the energy to send and receive a data packet, respectively. E_{MAC_TX} is the energy consumed by the transmitter to send an RTS and receive a CTS and ACK. Similarly, E_{MAC_RX} is the energy consumed by the receiver to get an RTS and send a CTS and ACK. Thus, we have:

$$E_{MAC\_TX} = E_{DIFS} + 3E_{SIFS} + 4E_{prop} + E_{RTS\_TX} + E_{CTS\_RX} + E_{ACK\_RX} \tag{3.11}$$

$$E_{MAC\_RX} = E_{DIFS} + 3E_{SIFS} + 4E_{prop} + E_{RTS\_RX} + E_{CTS\_TX} + E_{ACK\_TX} \tag{3.12}$$

where E_{DIFS} and E_{SIFS} are the idle energy consumed during DIFS and SIFS durations, respectively, and E_{prop} is the idle energy during propagation delays.

Thus, for an empty triggered wake, the energy is:

$$E_{empty} = 2\left(E_{thresh} + E_{trans\_on} + E_{trans\_off}\right) + NP_{sleep}T \tag{3.13}$$

where E_{thresh} is the energy needed to listen to the channel for T_{thresh} time. Equation 3.13 follows from the fact that both the sender and receiver must wakeup and keep their data radios on for T_{thresh} time and all N nodes have slept T time since the last wakeup.

For non-empty triggered wakeups:

$$E_{triggered} = 2E_{trans\_on} + Q\overline{f+e}E_{pkt} + 2E_{thresh} + NP_{sleep}T + 2E_{trans\_off} \tag{3.14}$$

Equation 3.14 is similar to Equation 3.13 except that extra energy is consumed to send $Q\overline{f+e}$ packets.

For a full wakeup, the equation is:

$$\begin{aligned} E_{full} \;=\; & E_{wake\_TX} + (N-1)E_{wake\_RX} + NE_{trans\_on} + NE_{DIFS} + \\ & E_{filter\_TX} + (N-1)E_{filter\_RX} + 2NE_{prop} + LE_{pkt} + \\ & 2E_{thresh} + NE_{trans\_off} + NP_{sleep}T_{sleep\_full} \end{aligned} \tag{3.15}$$

Equation 3.15 states that the sender must transmit a wakeup signal and $(N-1)$ receivers must listen to it. Then, all N nodes must turn their data radio on and wait until the filter packet is received. All the other nodes can return their data radios to a sleep state, but the sender and receiver remain on to exchange L packets. Finally, each node has slept T_{sleep_full} time since the last wakeup, unlike in Equations 3.13 and 3.14 where the nodes slept T time since the last wakeup.

Thus, the expected energy consumed per bit is:

$$E_{bit} = \frac{p_f E_{full} + p_{\overline{f+e}} E_{triggered} + p_e E_{empty}}{Data_{size} \times 8 \times (p_f L + p_{\overline{f+e}} Q_{\overline{f+e}})} \tag{3.16}$$

Using Equation 3.16, Figure 3.3 shows E_{bit} as a function of T for $R = 1.0$, $L = 2$, and $N = 8$.

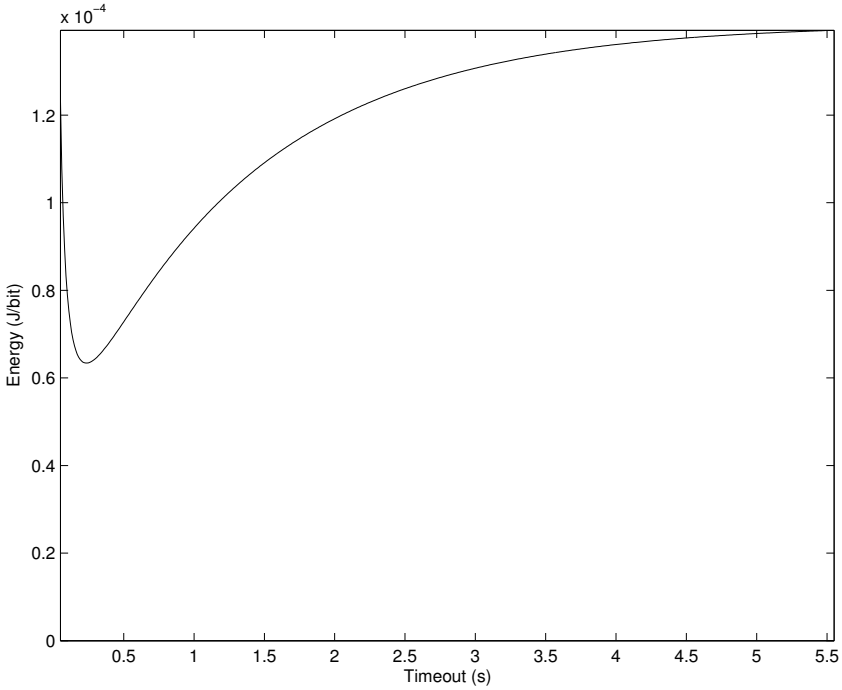

Figure 3.3: Energy versus timeout.

Note that E_{bit} is minimized at $T = T_{opt} \approx 0.235$. Clearly, choosing $T = T_{opt}$ should minimize energy consumption.

Figure 3.4 shows energy savings we can expect when T_{opt} is used compared to setting $T = \infty$ (i.e., a full wakeup occurs every time the queue fills up). The graph shows how the energy savings changes with R, L, and N, based on our analysis. The horizontal axis is the value of the changing parameter (i.e., R, L, or N) while the other two parameters stay fixed. The fixed values are: $R = 1.0$, $L = 2$, $N = 8$. For example, when R and L stay fixed and $N = 40$, T_{opt} gives about a 75% improvement (i.e., it uses only 25% of the energy per bit that $T = \infty$ uses). As another example, when R and N stay fixed and $L = 40$, there is only about a 30% improvement for reasons discussed below.

From Figure 3.4, we can observe how each of the parameters affects the energy savings. The energy savings remains almost constant as R changes. At very low rates, the ratio asymptotically approaches one because the sleep time between packets is so large that the energy spend sleeping between packets dominates the difference in energy to do a full wakeup versus a triggered wakeup. Otherwise, the ratio remains constant at R changes due to the fact that the rate primarily functions

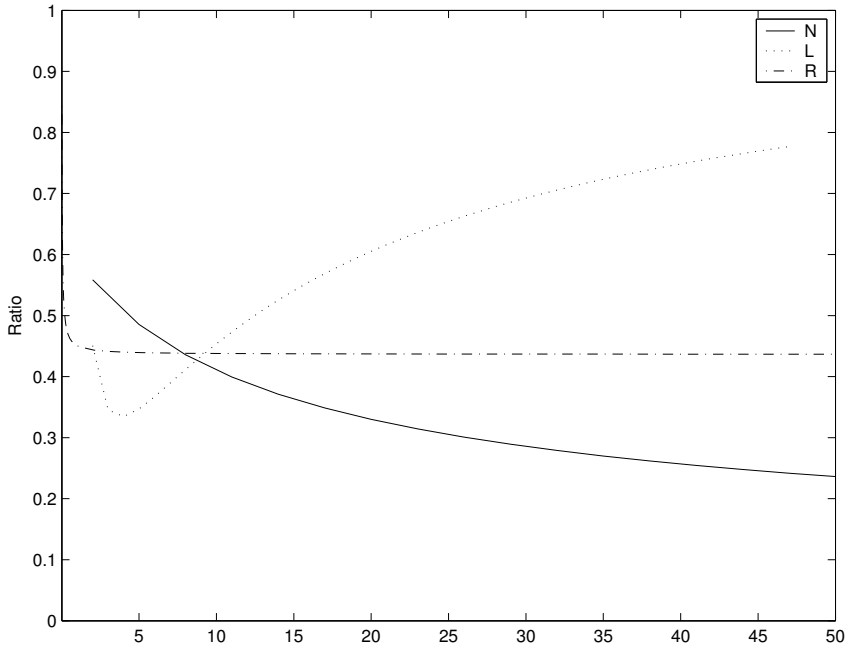

Figure 3.4: $\frac{E_{bit} \text{ when } T=T_{opt}}{E_{bit} \text{ when } T=\infty}$ ratio.

as a scaling factor that does not change the relative energy difference. As N changes, T_{opt} results in more energy savings because the entire neighborhood will only awake with probability p_f when packets are sent. When $T = \infty$, the neighborhood will awake every time packets are sent, obviously resulting in increased relative energy usage when N is large. In general, increasing L results in less energy savings because the full wakeup costs are amortized over more packets. Note, however, this trend is reversed when L is between about 2 and 5. This can be attributed to the fact that, despite the increased amortization of full wakeup costs, there is a large variance in when the L-th packet arrival occurs when L is small. Thus, there will be more full wakeups and empty triggered wakeups (i.e., $p_{\overline{f+e}}$, from Equation 3.5, will be small). For example, when $L = 2$, $p_{\overline{f+e}}$ is about 0.2, but when L increases to 5, $p_{\overline{f+e}}$ increases to about 0.6. Therefore, when L is small, the $p_{\overline{f+e}}$ factor dominates the relative energy savings, but as L grows larger, the amortization factor begins to dominate.

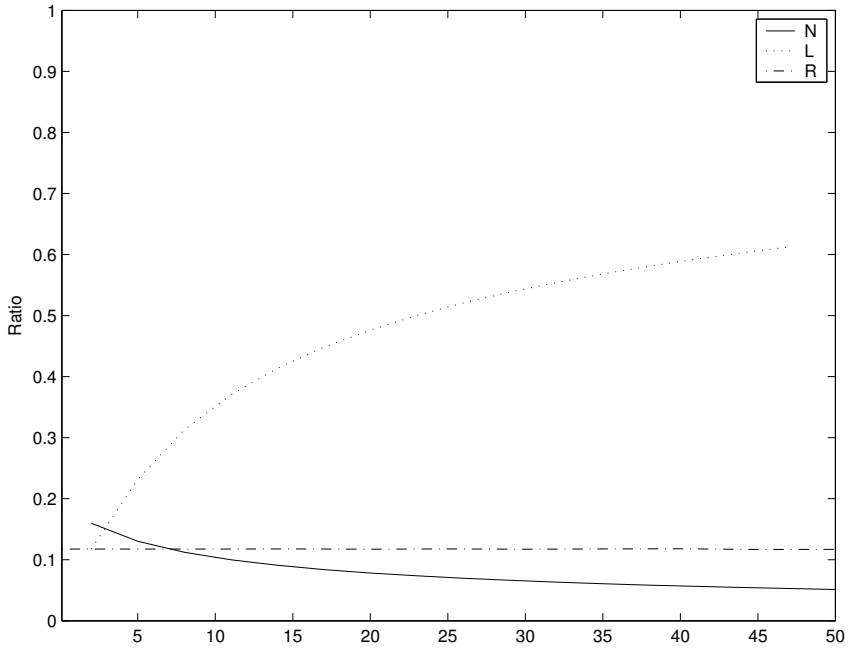

Figure 3.5: $\frac{T_{opt}}{L/R}$ ratio.

3.2.1 Adjusting T

The sender estimates its sending rate via a weighted average of the interarrival time of packets. The estimate, $R_{est} = \frac{1}{t_{est}}$, is updated according to the equation:

$$t_{est} = \rho t_{est} + (1 - \rho) t_{diff} \tag{3.17}$$

where t_{diff} is the most recent sample of interarrival time.

Figure 3.5 shows how the ratio $\frac{T_{opt}}{L/R}$ (where L/R is the expected time for the queue to reach L packets) changes with R, L, and N, based on our analysis. The horizontal axis is the value of the changing parameter (i.e., R, L, or N) while the other two parameters stay fixed. The fixed values are: $R = 1.0$, $L = 2$, $N = 8$. From Figure 3.5, when L and N are fixed, we observe that for some constant γ,

$$T_{opt} = \gamma \frac{L}{R} \tag{3.18}$$

where γ is independent of R. However, Figure 3.5 also shows that γ is not constant if L and N are

dynamic in the network. In our evaluation, we assume L and N are known in advance. Thus, γ is calculated as a function of L and N.

Chapter 4

Experimental Results

We implemented our protocol from Section 3.2 in *ns-2* [50] by modifying the 802.11 MAC and physical layer code in *ns-2*. Eight sensor nodes were placed within range of each other and a random sender and receiver were chosen to begin communicating with Poisson traffic at rate R. The remaining six nodes did not send or receive any data. We tested several R values of 0.2, 0.5, 1.0, and 2.0 packets per second. The resulting T_{opt} values were always greater than T_{min}. Unlike the analysis, packets were sent if they arrived after a wakeup occurred. We set $L = 2$ to demonstrate the simplest case of our protocol: rather than sending a packet immediately, we try to delay it until a triggered wakeup occurs. Each data point is averaged over ten runs and error bars show standard deviation. The simulation time was such that the expected number of packets sent was the same regardless of rate (unless otherwise noted, this value was set to 200). Thus, if the desired expected number of packets is P, the test for rate R_i would be run for $\frac{P}{R_i}$ time. The values in Tables 1.1 and 3.1 were used when applicable.

4.1 Effects of ρ

First, we investigate how ρ, from Equation 3.17, affects energy consumption. Recall that ρ is the weight given to the previous, cumulative interarrival estimate when a new arrival occurs to obtain a new interarrival time estimate. Intuitively, if ρ is large, the rate estimate is slow to adjust to rate variations, but more robust to occasional outliers. If ρ is small, T will adjust quickly to rate changes, but occasionally erroneously adjust too much in response to outliers.

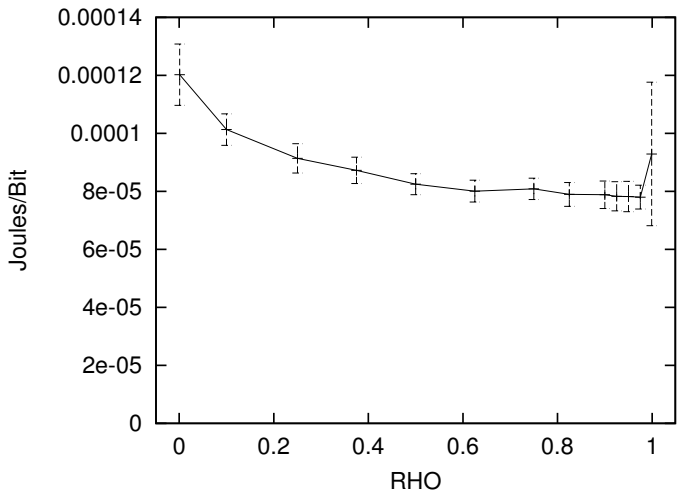

Figure 4.1: Effects of ρ when $R = 0.2$.

4.1.1 Traffic with a Low Rate

Figure 4.1 shows how the energy consumption changes with ρ at a low rate. We see that energy consumption remains about the same when ρ is in the range of 0.6 to 0.975. If ρ is close to 1, there is large variance since the rate estimate is primarily based on the first sample. Because the interarrival time of packets follows an exponential distribution, variance is greater at a low rate (i.e., large interarrival times). If ρ is chosen to be too small, the rate estimate will not make sufficient use of previous estimates and choose T based on the most recent samples.

4.1.2 Traffic with a Higher Rate

Figure 4.2 shows how the energy consumption changes at a higher traffic rate. Once again, if ρ is close to 1, there is a large variance in the energy consumption. In general, there is less variance at a higher rate since the interarrival time of packets follow an exponential distribution. This allows the energy consumption to remain about the same, even at small values of ρ (i.e., each triggered wakeup is based only on the previous sampled packet interarrival time). Since the variance is relatively low, basing our triggered wakeup time on only one sample does not degrade performance like it does when the rate is low (and hence packet interarrival time variance is high).

Based on these results, we use $\rho = 0.9$ for subsequent tests, unless stated otherwise. We omit results for $R = 0.5$ and $R = 1.0$ because they show similar trends to Figure 4.1, but note that

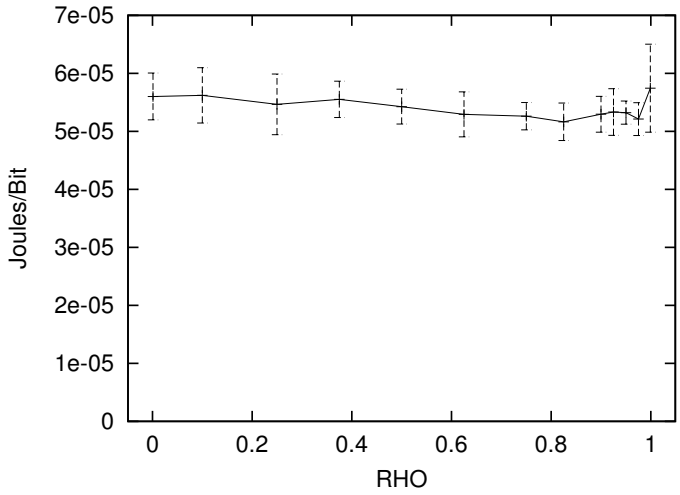

Figure 4.2: Effects of ρ when $R = 2.0$.

the lowest energy consumption occurred within the same range of ρ as for $R = 0.2$. Therefore, we expect $\rho = 0.9$ to give us the best energy consumption at each rate.

4.2 Comparison of Different Protocols

For comparison, we evaluated several protocols:

Rate Estimation (RATE EST) Our proposed protocol; γ is analytically calculated to be 0.1175.

Static Optimal (OPT) T is statically set to be T_{opt}, calculated analytically using the given rate. Thus, RATE EST estimates the rate dynamically, whereas OPT "magically" knows the rate.

T = ∞ (INFINITY) In this case, packets are only sent by full wakeups. Triggered wakeups never occur.

STEM This is STEM with a busy tone [5]. As mentioned in Chapter 3, STEM is a special case of our protocol.

4.2.1 Energy Comparison

Figure 4.3 plots the energy consumption of the protocols with rate on the horizontal axis. This

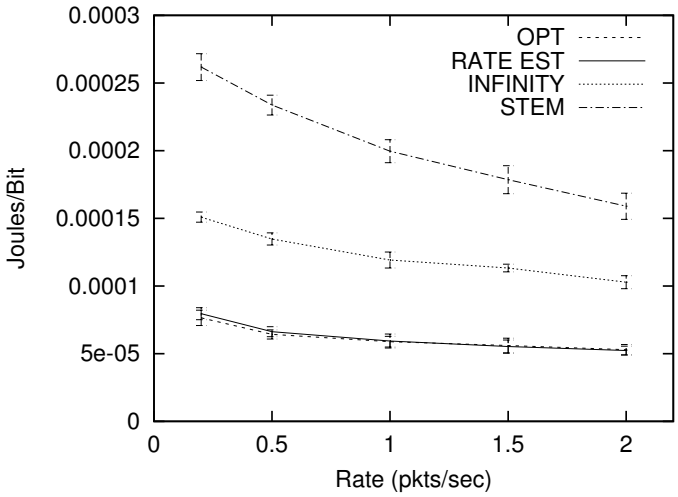

Figure 4.3: Energy consumption of protocols.

shows that regardless of rate, our protocol and the static optimal result in comparable energy consumption (the two curves almost overlap), which is significantly lower than the other protocols.

As shown in Figure 4.3, the rate estimation represents about 70% improvement over STEM regardless of rate. When compared to setting $T = \infty$, the rate estimation shows about 50% improvement. This shows the need to schedule triggered wakeups even if the full wakeup cost is amortized over multiple packets. In general, all the protocols improve as the rate increases. This is due to the decrease in sleep time between packets at higher rates.

Figure 4.3 also compares favorably to the analytical expectation shown in Figure 3.3. Both show the energy per bit to be about 60 μJ when $R = 1.0$. The 50% improvement over $T = \infty$ is also close to what is predicted in Figure 3.4.

4.2.2 Latency Comparison

Our protocol's performance is even better when the average packet latency is considered in Figure 4.4. Again, the rate estimation and the optimal performance overlap. Rate estimation shows more than 70% improvment compared to $T = \infty$. Thus, we can see our protocol performs much better than setting $T = \infty$ for both energy consumption *and* average packet latency.

As expected, STEM's latency is nearly constant at each rate. This latency corresponds to the time to do a full wakeup and shows virtually no variance. At higher rates, our protocol performs

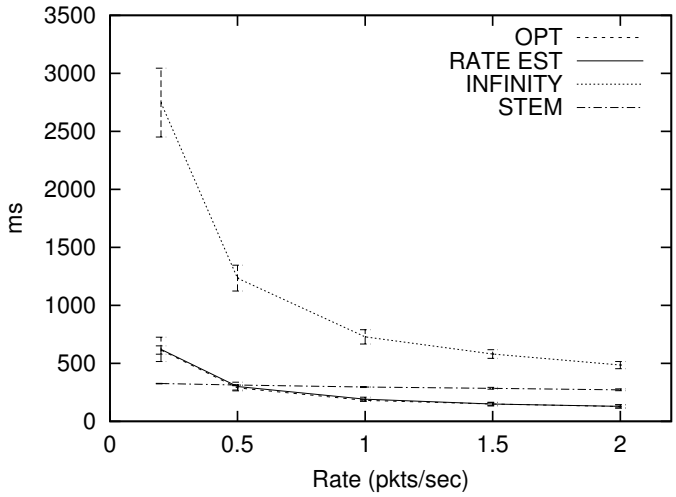

Figure 4.4: Latency of protocols.

better than STEM since T_{opt} is less than the time required to do a full wakeup. At low rates, T_{opt} is larger than the time to send a wakeup signal, which is why STEM has a lower latency at low rates. Note that $T = \infty$ will asymptotically approach STEM's latency, but never do better. On the other hand, because our protocol can avoid full wakeups, the latency will continue to be reduced until T reaches T_{min}. Thus, the theoretical bound as R increases of the ratio the latency of rate estimation to $T = \infty$ and STEM is $\frac{T_{min}}{2\tau_1+\tau_2}$ which is about 0.166 with our experimental setup (i.e., an 83.4% improvement).

We can easily check to see that the experimental results for the latency at $T = \infty$ are close to their expected value. In this case, we expect the average per packet latency for $L = 2$ to be:

$$\frac{(1/R + 2\tau_1 + \tau_2) + (2\tau_1 + \tau_2)}{2} \tag{4.1}$$

because the first packet will wait, on average, the expected interarrival time between packets $(1/R)$ plus the time to send a wakeup signal $(2\tau_1 + \tau_2)$, whereas the second packet will only have to wait long enough to send a wakeup signal $(2\tau_1 + \tau_2)$. We then divide the total latency by 2 since $L = 2$ to obtain Equation 4.1. Table 4.1 shows the analytical expectation for the latency with $T = \infty$ according to Equation 4.1 matches relatively well with the observed experimental values. At a higher rate, Equation 4.1 is not as accurate since it does not account for packets which arrive

Table 4.1: Measured latency for $T = \infty$ versus analysis for single hop, single flow scenario (in ms).

Rate	**Analytical Expectation**	**Measured Latency**	
		Avg.	Std. Dev.
0.2	2801	2747	297
0.5	1301	1235	112
1.0	801	728	62
1.5	634	580	37
2.0	551	485	30

during the wakeup and are sent without incurring the wakeup delay. At a higher rate, this happens more frequently, thereby reducing the average packet latency.

4.3 Effects of Traffic with a Time-Variant Rate

A major strength of our protocol is the ability to adjust to traffic on-demand as the sending rate changes. To test the dynamic adaptation of our protocol, R was changed from 0.2 to 2.0 packets per second and back periodically.

We use α to refer to the frequency with which the rate changes. More specifically, α is the expected number of packets generated at the current rate before switching to the other rate. For example, if $\alpha = 10$, packets are generated at $R = 0.2$ for $\frac{10}{0.2} = 50$ seconds, then at $R = 2.0$ for $\frac{10}{2} = 5$ seconds. This behavior was repeated for several cycles until the total expected number of packets sent per rate was 500. For the static optimal, we ran two separate scenarios at the different rates and averaged the results. This essentially represents the best energy consumption possible if the protocol adjusted to rate changes immediately.

4.3.1 Energy Comparison

Figure 4.5 plots energy consumption with α on the horizontal axis. As expected, rate estimation does better when the sender spends a long time at a fixed rate before switching rates. When rate change is infrequent, rate estimation uses only about 5% more energy than the static optimal. When $\rho = 0.6$, rate estimation converges more quickly toward the optimal since it is more responsive to rate change. STEM and $T = \infty$ stay relatively constant and use significantly more energy than

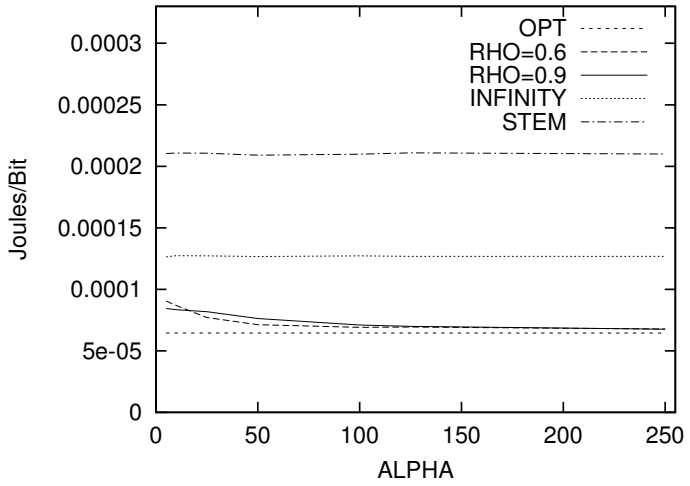

Figure 4.5: Energy consumption for traffic with time-variant rates.

our protocol.

Even in the worse case, where the rate is changing very frequently, our protocol only uses about 30-40% more energy than the optimal (depending on the ρ value). By comparison, $T = \infty$ always uses about 95% more energy than the optimal and STEM always uses over three times as much energy as the optimal.

4.3.2 Latency Comparison

The trends for the latency with time-variant traffic, shown in Figure 4.6, are similar to those seen in Figure 4.4. STEM stays near constant at the time to do a wakeup. The rate estimation protocol converges to the optimal when the sender spends a long time at a fixed rate before switching rates. The latency of the optimal energy protocol remains constant since α does not affect the optimal in our experiments. The latency of the rate estimation and optimal protocols is slightly above STEM because if we were to average the latencies of $R = 0.2$ and $R = 2.0$ from Figure 4.4, they are slightly higher than STEM for reasons discussed in Section 4.2.2.

Again, we can verify that the experimental results for $T = \infty$ match the expected average per packet latency. From Equation 4.1, we can average the expected latency at $R = 0.2$ (2801 ms) and $R = 2.0$ (551 ms) to get 1676 ms. This is close the experimental average values, which are observed to be between 1557 and 1633 ms.

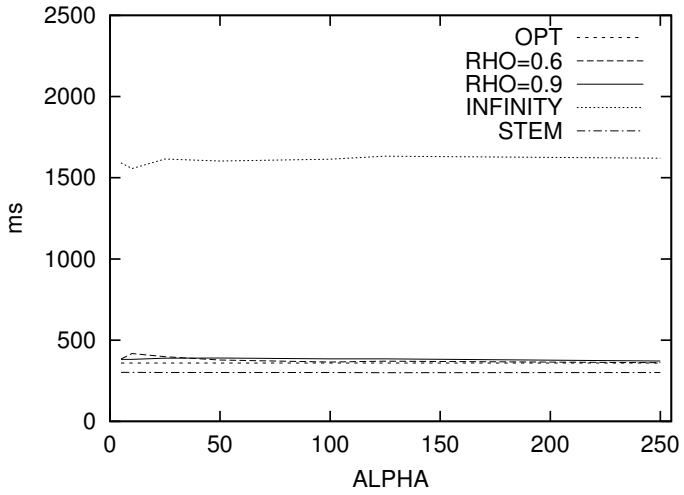

Figure 4.6: Latency for traffic with time-variant rates.

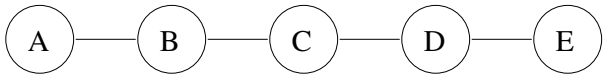

Figure 4.7: Topology for testing multiple hop performance.

4.4 Multiple Hop Performance

After examining the performance of our protocol in a single-hop, single-flow scenario, we wanted to see how it would perform in more complex scenarios. The first we tested was the multiple hop scenario in Figure 4.7. Traffic was sent from **A** to **E**, the rate was varied, and the simulation time was varied inversely with the rate such that the expected number of packets generated during a simulation run (set to be 200) remained constant.

Each node in the topology had seven neighbors, to allow comparison to the tests in Section 4.2. Note that neighbors not on the data path are not shown in Figure 4.7. Thus, the two end nodes in the topology have six neighbors and the intermediate nodes have five neighbors. The neighbors not on the data path were placed such that they could overhear exactly one node on the data path (i.e., nodes **A**, **B**, **C**, **D**, **E** do not share any neighbors other than those shown in Figure 4.7). In the discussion, the term *downstream neighbor* refers to a node that is closer to the destination than the current node (e.g., **C** is a downstream neighbor of **B**). The term *upstream neighbor* refers to a node that is closer to the source than the current node (e.g., **C** is an upstream neighbor of **D**).

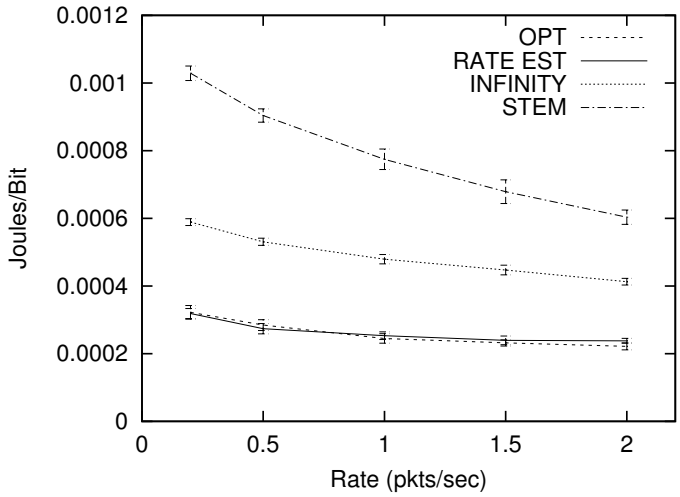

Figure 4.8: Energy consumption for multiple hop scenario.

Figure 4.8 shows energy consumption follows a similar trend to that of Figure 4.3. It is reasonable to expect that the results from Figure 4.3 would be scaled by a factor of about four since there are now four links per packet delivery rather than one. However, there are some effects due to the dependence of packet arrivals on a link. For example, in STEM, if **A** is able to send two packets during a wakeup instead of just one, then each node along the path will also send two packets during the wakeup of their downstream neighbor instead of one. Thus, STEM and $T = \infty$ only increase energy consumption by a factor of about 3.9. This factor is obtained by taking the ratio of the results in Figure 4.8 to those in Figure 4.3. However, rate estimation and the static optimal are negatively affected by the link dependence. If **A** causes a full wakeup, then the full wakeup will cascade down the entire path instead of being independent at each hop. This effect does not cause major degradation, however, with the energy consumption increased by a factor of 4 to 4.6 over the single hop case depending on the rate.

We note an issue that occurred with our protocol when it was implemented on the multiple hop topology. When multiple packets are sent to the next hop, they come in a quick burst. Therefore, the receiver's rate estimation calculates a very short interarrival time. This problem could be addressed by ignoring interarrival times that are smaller than a certain threshold. This modification is not implemented in the simulations.

Figure 4.9 shows the latency of the protocols. This shows a similar trend to Figure 4.4. However,

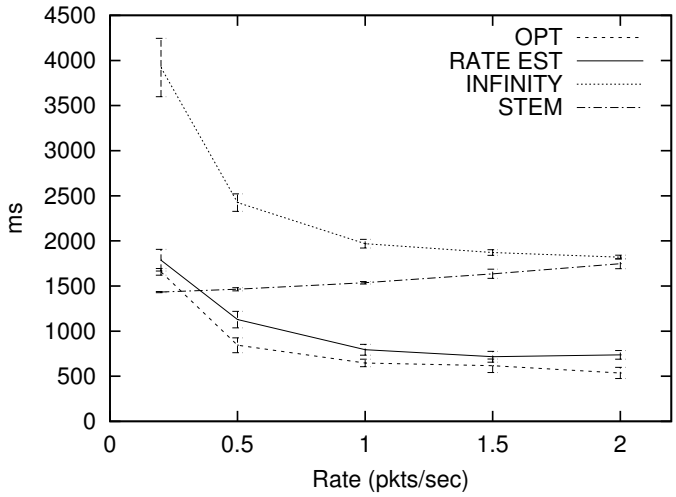

Figure 4.9: Latency for multiple hop scenario.

like the energy consumption results, these results do not always scale by a factor of four. At a low rate, STEM does show a 4.4 times increase (when compared to the single hop case) because at each hop, the packet must wait for the wakeup to occur and a small amount of contention is now induced (e.g., when **A** tries to send to **B** and **C** tries to send to **D**, one must defer their transmission to avoid a collision). However, the protocols which use $L = 2$ show less of an increase because the intermediate hops can immediately send packets to their downstream neighbor when they receive L packets from their upstream neighbor. For example, at $R = 0.2$ for $T = \infty$, the latency increases by a factor of only 1.41. In the single hop case, the first packet to arrive in the empty queue for $T = \infty$ has to wait, on average, 5 seconds plus the wakeup time. However, now both packets only have to wait about a wakeup time at each intermediate node. Similarly, rate estimation and the static optimal have their latency increase by a factor of only about 2.9 due to the decreased latency on full wakeups.

If we use Equation 4.1 to verify the latency for $T = \infty$ at $R = 0.2$, the first hop still has an expected latency of 2801 *ms* (as shown in Section 4.2.2). However, the next three links have an average latency of only 301 *ms*. Thus, the overall expected latency is $(2801 + 3 \times 301) = 3704$ *ms*, which is well within the deviation shown in Figure 4.9.

At a higher rate, the burstiness at downstream links does not help the $L = 2$ protocols as much because the interarrival time of packets is smaller. Also, at a higher rate, all the protocols show an

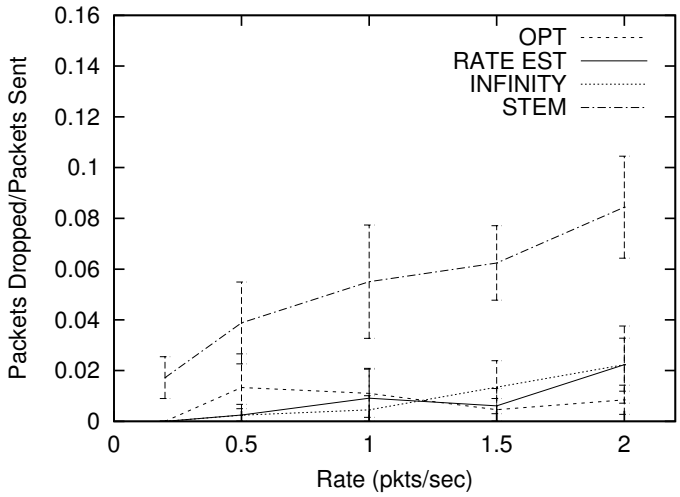

Figure 4.10: Packet drop percentage for multiple hop scenario.

increase in latency, relative to the single hop case, significantly greater than at a low rate. This is due to the increased contention on the links as the rate increases.

The effects of increased contention are also seen in the data packet drop rate, shown in Figure 4.10. As the rate increases, more drops occur due to the medium being increasingly busy. Packet drops generally occur because a sender miscalculates when the receiver will be up and does not receive a response to its RTS. In *ns-2*, a data packet is dropped after an RTS for the packet has been retransmitted seven times without receiving a CTS. For the protocols with triggered wakeups, this can occur when a packet is lost and hence the nodes believe they should wakeup at different times. For example, a sender could transmit a data packet telling the receiver to wakeup T_{new} seconds after reception. However, if the receiver does not receive the packet due to a collision and the sender is not able to retransmit the packet before the pair's scheduled sleep times, the receiver may believe it is supposed to wakeup T_{old} time after it estimates the sender has turned off.

Packet drops can occur during a full wakeup due to excessive retransmissions also. For example, **B** could begin transmitting a wakeup signal for **C** and shortly thereafter, **A** begins transmitting a wakeup signal for **B**. Thus, **B** may not receive **A**'s wakeup signal because it never listened on the wakeup channel during that time. Thus, **A** could believe **B** to be awake when it finishes transmitting the wakeup signal even though **B** has already finished its data transmission to **C** and returned to sleep without ever hearing **A**'s wakeup signal. STEM is most affected by this since it

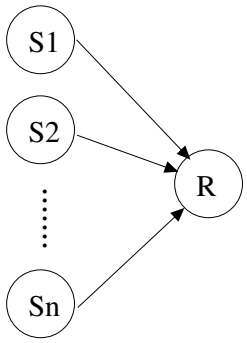

Figure 4.11: Topology for testing multiple senders.

does about twice as many wakeups as $T = \infty$ for $L = 2$. This is because STEM does a wakeup for every packet whereas $T = \infty$ does a wakeup for every other packet.

4.5 Multiple Flow Performance

In this section, we look at the affects of having multiple flows on the protocols. We do not consider the static optimal in these scenarios since it only represents the optimal in the single flow case. In these scenarios, we doubled the number of nodes to be 16. This is because we needed to increase the number of flows beyond eight to show interesting behavior. This is also the reason the energy values are higher in these scenarios when there is only one flow than in previous sections. For each scenario, connections between nodes were chosen to have a rate probabilistically. More specifically, each link has a rate of $R = 0.2$ or $R = 1.0$ with a probability of 0.5. Thus, the average rate per link is $R = 0.6$, or one packet every 1.67 seconds (even though this specific rate is never chosen for a link). This was done to show how the rate estimation performs when nodes are sending or receiving at multiple rates to or from different neighbors. Each simulation run lasts 500 seconds. Because each scenario could have links with two different rates, there was no way to normalize the simulation time such that the expected number of packets generated per link remained constant and each link contended for medium access throughout the entire simulation run.

4.5.1 Multiple Senders

In this scenario, we used the topology shown in Figure 4.11. The parameter we varied is the

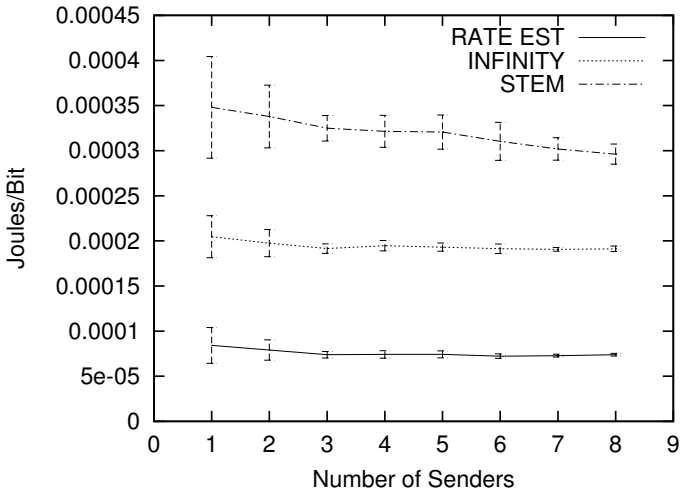

Figure 4.12: Energy consumption for multiple sender scenario.

number of senders. The overall number of nodes in the scenario remains constant at 16 and all nodes are within range of each other. The rate estimation protocol is not affected at the sender, because each sender has only one flow, but the receiver must adopt multiple schedules and respond to wakeups from multiple senders.

Figure 4.12 shows that the energy consumption of each protocol remains relatively constant as the number of senders increases. This implies that the extra energy incurred by an additional flow is compensated by the extra packets that are delivered. The rate estimation protocol represents about a 60% improvement over $T = \infty$ and a 75% improvement over STEM. This is consistent with the results discussed in Section 4.2.1. The rate estimation protocol does slightly better in the multiple sender scenario because $N = 16$ instead of 8. As explained in reference to Figure 3.4, the larger N makes full wakeups more expensive.

We present the latency for the protocols in the multiple sender scenario in Figure 4.13. We can see that the latency remains relatively constant for the rate estimation protocol and $T = \infty$ regardless of the number of senders. The latency for these protocols is the same as the interpolated latency for $R = 0.6$ in Figure 4.4. STEM shows a slight increase as the number of senders increases due to the increased contention caused by more wakeups occurring for the receiver. All protocols show a much greater variance when the number of senders is low. This is due to the large difference in latency according to which rates are probabilistically chosen on the links. For example, when

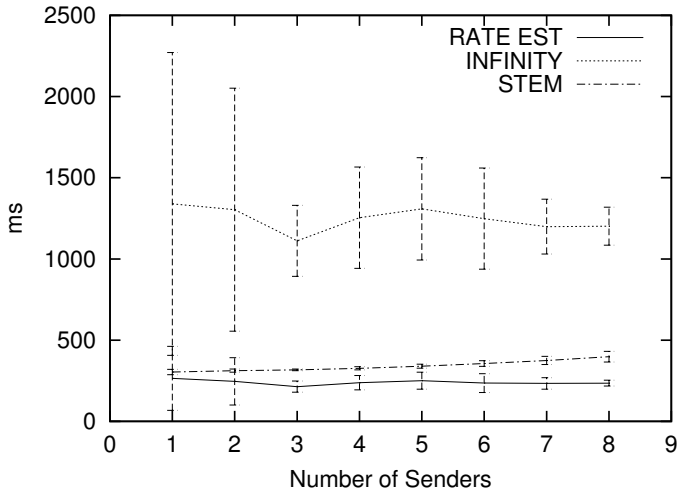

Figure 4.13: Latency for multiple sender scenario.

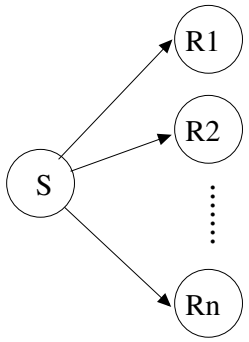

Figure 4.14: Topology for testing multiple receivers.

there is only one sender, about half the scenarios will have $R = 0.2$ for the connection, while the other half will have $R = 1.0$, which results in drastically different average latencies.

Very few packet drops occurred in this scenario. The packet loss was less than 0.5% regardless of the protocol or number of senders. Because there is only one receiver, even if the sender begins sending RTS packets when it incorrectly predicts the receiver will be on, they may still get a response if the receiver is up for a different sender.

4.5.2 Multiple Receivers

In this scenario, we used the topology shown in Figure 4.14. The parameter we varied is the number of receivers. The overall number of nodes in the scenario remains constant at 16 and all nodes are

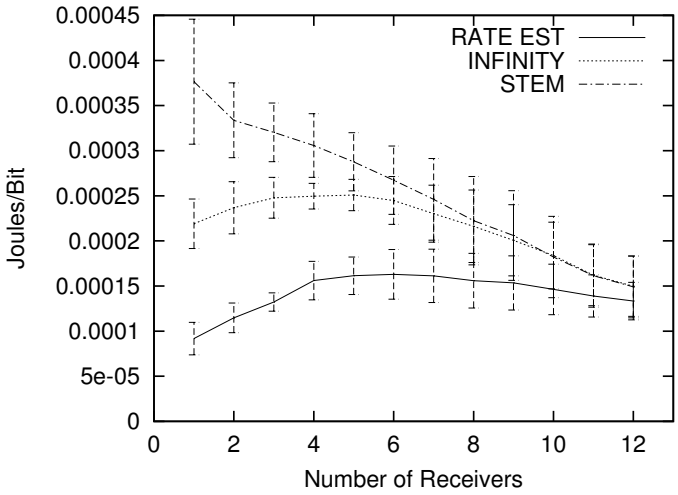

Figure 4.15: Energy consumption for multiple receiver scenario.

within range of each other. The filter packets are capable of advertising up to four addresses of receivers that should remain awake. With multiple receivers, we still update each T according to Equation 3.18, but now R refers to the cumulative rate of all flows. Thus, each destination may receive the same T value, but at different times. Also, at different times, T may be different since the rate estimate changes.

The intuition behind this scheme is as follows. Assume that there are n flows at a sender. Let L_{v_i} refer to *virtual* queue threshold for flow i, and R_i denote the rate of the flow. Thus,

$$L_{v_i} = \frac{R_i}{\sum_{i=1}^{n} R_i} L \tag{4.2}$$

where $R = \sum_{i=1}^{n} R_i$ is the cumulative rate of all flows. Therefore, $T_i = \gamma_i \frac{L_{v_i}}{R_i} = \gamma_i \frac{L}{R}$. However, γ_i cannot be calculated online since it changes with respect to L_{v_i}. Also, for $L = 2$, at least $n - 1$ of the L_{v_i} values will be less than one. This cannot be calculated in our analysis since it requires $L \geq 2$. Therefore, for γ_i, we just use the γ value for L. This does have a small affect on the protocol since the chosen value of T_i is larger than if γ_i had been used. When the number of receivers is small, the average number of full wakeups per receiver increases as the number of receivers grows instead of remaining constant.

The energy consumption of the protocols is shown in Figure 4.15. STEM's energy begins to

drop when the number of receivers is increased because the overall rate is increasing and hence the sleep time per packet is decreased. For $T = \infty$ and the rate estimation protocol, the energy starts to increase with the number of receivers because, when multiple destinations are awakened, all but one of the receivers must idly listen to the transmission. Also, as discussed previously, the rate estimation protocol results in more full wakeups per receiver as the number of receivers increases.

When the number of receivers reaches about 6 or 7, a couple of interesting events occur. First, the overall rate becomes high enough that the expected interarrival time between packets is less than the time to perform a full wakeup. Thus, the service rate is less than the arrival rate for STEM and its queue begins building up. As the queue length increases, it actually decreases the energy consumption because the probability that multiple packets can be sent from the queue on a wakeup is increased. When the number of receivers is about 9 or 10, the overall rate becomes high enough that the expected interarrival time between packets is less than half the time to perform a full wakeup. Thus, the queue begins building up for $T = \infty$ and its energy consumption converges with STEM. When this situation occurs, the queue will eventually have to start dropping packets since it has finite storage space. Also at about 6 or 7 receivers, the overall rate is high enough that T_{opt} drops below T_{min}. Thus, the rate estimation protocol begins to gradually decrease at this point as the time between triggered wakeups cannot decrease further. This decrease is because periodic wakes cannot cannot occur more frequently, yet the queue fills up faster. Thus, more packets in the queue increases the probability multiple packets can be sent when a wakeup occurs.

We gain further intuition into the protocols' behavior by looking at the latency in Figure 4.16. When the number of receivers is small, as the number of receivers increases, the latency curves show a decrease for our protocol and $T = \infty$. However, STEM and $T = \infty$ begin showing increased latency when the number of receivers gets larger and packets must spend more time in the queue. Again, STEM and $T = \infty$ converge in this metric when the number of receivers is about 10, which is the point that both protocols have a service rate larger than the arrival rate. The rate estimation protocol is able to do much better in terms of latency because it can service packets faster than the other two protocols. Specifically, since $T = T_{min}$ at a high rate and T_{min} is significantly less than the time it takes to do a full wakekup (a ratio of about $\frac{1}{6}$ in our simulations), packets wait less time in the queue.

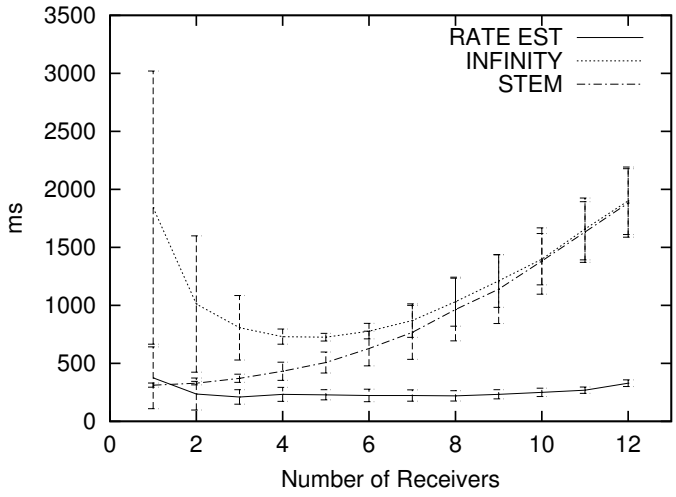

Figure 4.16: Latency for multiple receiver scenario.

Because there is only one sender, and hence no contention for transmissions, the amount of packet drops remains relatively low (i.e., less than 0.5% on average). The packet drops for STEM and $T = \infty$ come predominantly from packets left in the queue when the simulation ends (we count packets queued at the end of the simulation as dropped packets). Thus, this gives a rough estimate of how the sender's queue size is increasing with more receivers. Because the rate estimation protocol has frequent triggered wakeups, the queue does not build up in our simulations. Therefore, virtually no packets are in the queue at the end of the simulation. Figure 4.17 shows the packet drop percentage for the multiple receiver scenario.

We did observe an interesting source of MAC layer retransmissions with the rate estimation protocol. In *ns-2*, if the length of time between when a receiver sends a CTS and when the receiver gets the data packet is too long, the data packet will be dropped by the receiver. Occasionally, the sender, **S**, initiates a full wakeup for $\mathbf{R_1}$. Just before **S** sends its filter packet for $\mathbf{R_1}$, **S** does a periodic wake with $\mathbf{R_2}$ and exchanges an RTS and CTS. However, before the data packet can be sent to $\mathbf{R_2}$, the filter packet is sent for $\mathbf{R_1}$. Thus, the data packet is delayed longer than expected and when it is finally sent to $\mathbf{R_2}$, it is dropped.

Giving the filter packet priority over the data packet is a design decision in our protocol. The intuition behind this is as follows. If the filter packet gets delayed long enough, the intended receiver could return to sleep while the sender is transmitting a data packet to another destination. The

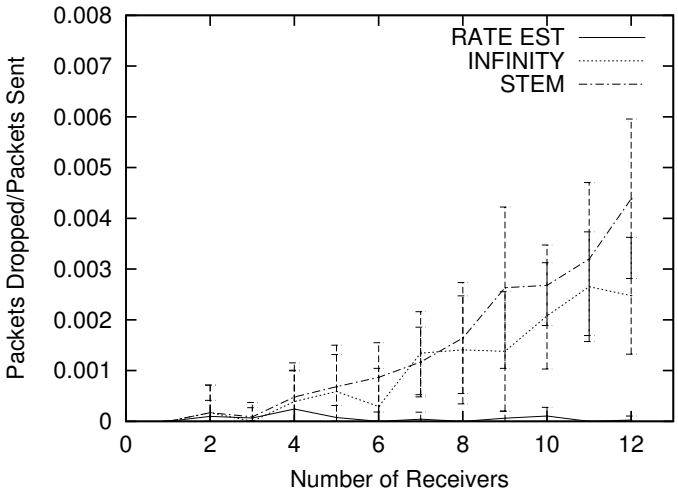

Figure 4.17: Packet drop percentage for multiple receiver scenario.

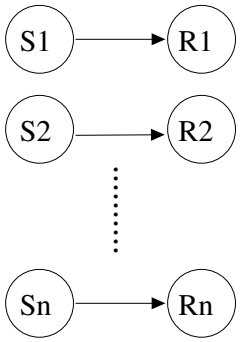

Figure 4.18: Topology for testing multiple connections.

energy cost of doing another full wakeup in this situation is greater than just retransmitting the data packet. As an alternative, when the sender begins a full wakeup, it could block the transmission of all data packets until the filter packet is sent. Another possibility is the sender could delay sending a packet for a triggered wakeup if the packet transmission would not finish before the filter packet will be sent. These two modifications are not implemented in the simulations.

4.5.3 Multiple Connections

In this scenario, we used the topology shown in Figure 4.18. The parameter we varied is the number of connections. Each connection was chosen such that a node was either a sender or receiver and only communicated with at most one other node. The overall number of nodes in the scenario

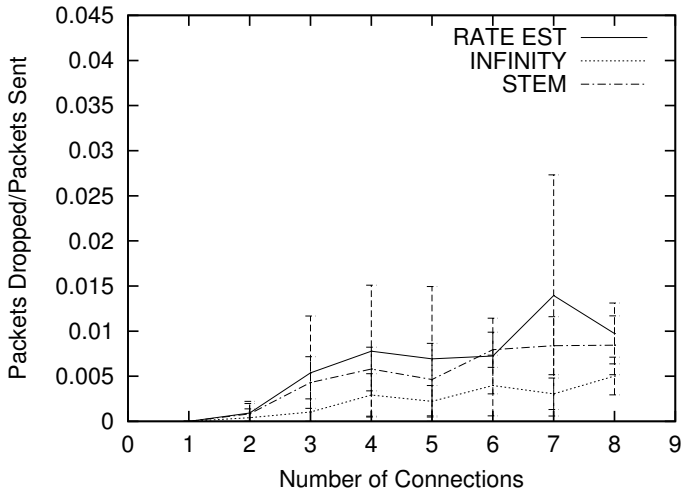

Figure 4.19: Packet drop percentage for multiple connection scenario.

remains constant at 16 and all nodes are within range of each other. Because each sender has only one flow, the results are similar to those in Section 4.5.1. The primary difference is that, since nodes are sending to different receivers, there are more packet drops because it is less likely that a receiver will be turned on to communicate with a different node if the sender-receiver pair incorrectly estimates when the other is on. Also, if two senders transmit a wakeup signal at about the same time, it is possible that the first one to transmit its filter packet will cause the other's receiver to return to sleep. For example, in Figure 4.18, if $\mathbf{S_1}$ and $\mathbf{S_2}$ begin sending a wakeup signal at about the same time, they will both contend to send the filter packet on the data channel at about the same time. Without loss of generality, assume $\mathbf{S_1}$ wins the medium access and sends a filter packet telling $\mathbf{R_1}$ to remain awake. $\mathbf{R_2}$ will overhear $\mathbf{S_1}$'s filter packet and return to sleep. Thus, when $\mathbf{S_2}$ sends its filter packet, $\mathbf{R_2}$ may not be awake to receive it.

The packet drop percentage, shown in Figure 4.19, appears to be about an order of magnitude higher in this scenario when compared to the multiple sender scenario in Section 4.5.1. The drop rate reaches about 0.4% for $T = \infty$, 0.85% for STEM, and 1% for rate estimation in the multiple connection scenario. As shown in Figure 4.20, these extra packet drops have a small effect on the energy usage. When compared to the multiple sender scenario, the multiple connection scenario shows an increase in energy of about 3-4% for STEM, 1% for $T = \infty$, and 6-10% for rate estimation. This slight increase in energy consumption seems to correlate rather strongly with the increased

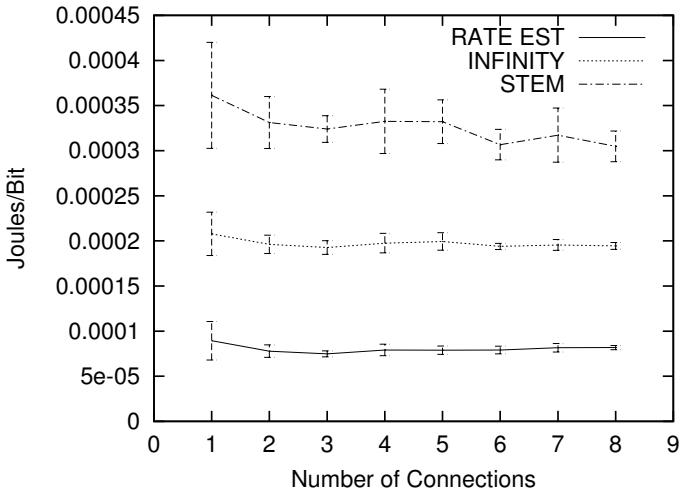

Figure 4.20: Energy consumption for multiple connection scenario.

packet drops. This is because less packets are being received despite when about the same amount of energy is used. The average packet latencies for this scenario are about the same as those in Figure 4.13.

4.6 Results Summary

To conclude this chapter, we present a brief summary of the results. First, we tested the weighted average function proposed in Equation 3.17 to determine which weights worked the best. We then evaluated our proposed rate estimation protocol against the static optimal, STEM, and setting $T = \infty$. In a single hop, single flow setting, our protocol performs the same as the static optimal in terms of energy and latency. The energy consumption is always significantly reduced when compared to STEM and $T = \infty$. The rate estimation protocol also works well when the rate of traffic is time-variant. Even when the rate is changing frequently, our protocol's energy consumption performs only slightly worse than the optimal and much better than STEM and $T = \infty$. As the rate change becomes less frequent, the rate estimation protocol converges toward the static optimal in energy consumption.

We then tested the protocols in a multihop scenario with a single flow. Here, we found that the trends for energy consumption and latency are about the same as the single hop case. In particular,

the rate estimation protocol closely mirrors the optimal. However, at increased rates, the number of packet drops became more significant in response to increased contention for all protocols. STEM, in particular, showed a large number of packet drops because of more frequent full wakeups for reasons elaborated on in Section 4.4.

We next tested all the protocols, except the static optimal, in single hop, multiple flow scenarios. The static optimal was not considered since it only has meaning in the single flow case. We tested three scenarios involving: multiple senders and one receiver, one sender and multiple receivers, and multiple sender-receiver pairs (i.e., multiple connections). For the multiple sender and multiple connection scenarios, we found that the rate estimation protocol performs about the same relative to STEM and $T = \infty$ as the single flow case. In the multiple receiver case, the rate estimation protocol has lower energy consumption than STEM and $T = \infty$, but the relative difference depends on the number of receivers. When the number of receivers is large, there is little difference since STEM and $T = \infty$ are constantly doing full wakeups and the rate estimation protocol is doing periodic wakeups very frequently. The major difference in this scenario is that the rate estimation protocol is able to keep the per-packet latency relatively constant and much lower than STEM and $T = \infty$, which show a significant increase as the number of receivers grows. This is caused by the longer service time in STEM and $T = \infty$ since packets must wait for a full wakeup, whereas, with rate estimation, packets only have to wait until the next periodic wakeup. When the number of receivers is large, the periodic wakeups are frequent. Therefore, packets are serviced faster than if a full wakeup was done and do not spend as much time in the queue.

Chapter 5

Conclusions and Future Work

We have analyzed a protocol for sensor networks that increases energy efficiency by allowing packet buffering, thereby amortizing the energy cost of communication over multiple packets. Because storage space may be a scarce resource in sensors, we propose adding a second, low-power radio to allow senders to force receivers to wakeup when a specified number of packets are being buffered. Our analysis reveals an optimal timeout value for periodically waking up to send and receive packets which minimizes energy consumption. Our protocol uses rate estimation to achieve results comparable to the optimal. In addition, we show significant energy savings over other, similar protocols. The protocol seems to behave well when multiple hop and multiple flow scenarios are introduced as well. In such situations, it almost always outperforms other protocols in energy and latency.

For future work, we outline a few directions that could be pursued:

More Realistic Environment We would like to adapt our protocol into a more realistic environment. For example, every sensor node could report periodically at a low rate in steady state. Then, when an event occurs, the affected sensors begin transmitting at a much higher rate for the duration of the event.

Staggering Schedules We would also like to investigate how to efficiently stagger multiple schedules such that interfering nodes will minimize the time they are both on.

Multiple Wakeup Channels Currently, we assume all nodes must share the same wakeup channel. However, interesting problems arise if we consider the case in which a few bits can be encoded in the wakeup signal. Then, nodes only wakeup if their pre-assigned bit pattern is

in the wakeup signal. For example, if we know the rates at which nodes are sending data and have k wakeup channels to use, we would like to assign channels to the nodes in such a way that minimizes energy consumption caused by full wakeups. There are also other ways to partition the wakeup channel, other than encoding bits in the wakeup signal, which can be explored. For example, a different frequency band could be used for each wakeup signal and nodes could be assigned a frequency on which to listen for wakeups.

Adjust T_{thresh} Dynamically Our protocol may be improved if we integrate the technique shown in Figure 3.2 with our current protocol. Thus, T_{thresh} would be adjusted dynamically rather than having a static value. The advantage to this is, when traffic is heavy, T_{thresh} could be made larger since it may take more time for the sender to access the medium to transmit a packet. The disadvantage of this approach is that it requires more synchronization to make sure neighboring nodes agree on a T_{thresh} value.

Eliminating Filter Packets The overhead of filter packets could be reduced if the information they contain was piggybacked onto RTS and/or DATA packets.

Appendix A

Matlab Code for Analysis

```
function e = e_pkt(T,R,L,N)
global P_sleep
global E_trans_on E_filter_RX E_filter_TX E_data_RX E_data_TX E_thresh
global E_trans_off E_MAC_RX E_MAC_TX E_wake_TX E_wake_RX E_prop E_DIFS
global Data_Bytes

global infty
infty = 10000;

% All values in W,s,J
stem_T = 299e-3;
stem_TRX = 1e-3;

P_TX = 81e-3;
P_RX = 30e-3;
P_idle = 30e-3;
P_TX_wake = 81e-3;
P_RX_wake = 30e-3;
P_idle_wake = 30e-3;
P_sleep = 3e-6 * (stem_T/(stem_T + stem_TRX)) + ...
   P_idle * (stem_TRX/(stem_T + stem_TRX)) + 3e-6;
```

```
P_trans_on = 30e-3;
P_trans_off = 30e-3;

T_trans_on = 0;
T_trans_off = 0;
T_thresh = 20e-3;
T_DIFS = 50e-6;
T_SIFS = 10e-6;
T_prop = 2e-6;
T_wake = stem_T + 2*stem_TRX;

PLCP_bytes = 4;
IP_bytes = 20;
Data_MAC_bytes = 32;

Filter_size = 33 + PLCP_bytes;
Data_Bytes = 30;
Pkt_size = Data_Bytes + Data_MAC_bytes + PLCP_bytes + IP_bytes;
RTS_size = 20 + PLCP_bytes;
CTS_size = 14 + PLCP_bytes;
ACK_size = 18 + PLCP_bytes;
BW = 40e3; %bps

T_data = ((Pkt_size*8) / BW);
T_filter = ((Filter_size*8) / BW);
T_RTS = ((RTS_size*8) / BW);
T_CTS = ((CTS_size*8) / BW);
T_ACK = ((ACK_size*8) / BW);
```

```
E_trans_on = P_trans_on * T_trans_on;
E_trans_off = P_trans_off * T_trans_off;
E_wake_RX = P_idle_wake * (stem_T/2);
E_wake_TX = P_TX_wake * T_wake;
E_filter_RX = P_RX * T_filter;
E_filter_TX = P_TX * T_filter;
E_data_RX = P_RX * T_data;
E_data_TX = P_TX * T_data;
E_RTS_RX = P_RX * T_RTS;
E_RTS_TX = P_TX * T_RTS;
E_CTS_RX = P_RX * T_CTS;
E_CTS_TX = P_TX * T_CTS;
E_ACK_RX = P_RX * T_ACK;
E_ACK_TX = P_TX * T_ACK;
E_DIFS = P_idle * T_DIFS;
E_SIFS = P_idle * T_SIFS;
E_prop = P_idle * T_prop;
E_MAC_TX = E_DIFS + 3*E_SIFS + 4*E_prop + E_RTS_TX + E_CTS_RX + E_ACK_RX;
E_MAC_RX = E_DIFS + 3*E_SIFS + 4*E_prop + E_RTS_RX + E_CTS_TX + E_ACK_TX;
E_thresh = P_idle * T_thresh;

e_fh = str2func('e_pkt_complex_sleep');
e = feval(e_fh,T,R,L,N);

function e = e_pkt_complex_sleep(x,R,L,N)
global P_sleep E_trans_on E_filter_RX E_filter_TX E_data_RX ...
   E_data_TX E_thresh E_trans_off E_MAC_RX E_MAC_TX E_wake_TX ...
   E_wake_RX E_prop E_DIFS Data_Bytes
T = x;
```

```
p_f = p_full(R,T,L);
p_nfe = p_notfullempty(R,T,L);
p_e = exp(-R*T);
p_nf = p_notfull(R,T,L);
q_nf = q_notfull(R,T,L);
t_s = t_sleep(R,T,L);
E_iter = ...
   p_f * ...
      (E_wake_TX + (N-1)*E_wake_RX + N*E_trans_on + E_DIFS*N + ...
         E_filter_TX + (N-1)*E_filter_RX + 2*N*E_prop + ...
         L*(E_MAC_RX + E_data_RX + E_MAC_TX + E_data_TX) + 2*E_thresh + ...
         N*E_trans_off + t_s*P_sleep*N) + ...
   p_nfe * ...
      (2*E_trans_on + ...
         q_nf*(E_MAC_RX + E_data_RX + E_MAC_TX + E_data_TX) + ...
         2*E_thresh + 2*E_trans_off) + ...
   p_e * (2*(E_trans_on + E_thresh + E_trans_off)) + ...
   p_nf * (P_sleep*T*N);
e = (E_iter ./ (Data_Bytes*8*(p_f.*L + p_nfe.*q_nf)));

function p = p_full(R,T,L)
p = 1 - (p_notfull(R,T,L));

function p = p_notfull(R,T,L)
global infty
if T == infty
   p = 0;
else
   i = [0:(L-1)];
```

```
   p = sum(exp(-R.*T).*(((R.*T).^i)./vec_factorial(i)));
end

function p = p_notfullempty(R,T,L)
global infty
if T == infty
   p = 0;
else
   i = [1:(L-1)];
   p = sum(exp(-R.*T).*(((R.*T).^i)./vec_factorial(i)));
end

function q = q_notfull(R,T,L)
global infty
if T == infty
   q = (L-1);
else
   i = [1:(L-1)];
   num = sum(((i.*(R.*T).^i)./vec_factorial(i)));
   den = sum((((R.*T).^i)./vec_factorial(i)));
   q = (num ./ den);
end

function t = t_sleep(R,T,L)
global infty
if T == infty
   t = (L/R);
else
   t = (quad(@t_ex_fn,0,T,[],[],R,L) ./ ...
```

```
      quad(@t_dist_fn,0,T,[],[],R,L));
end

% x * f(x) for sleep time
function t = t_ex_fn(x,R,L)
t = (x .* t_dist_fn(x,R,L));

% f(x) for sleep time
function t = t_dist_fn(x,R,L)
t = ((R.^L)./(gamma(L)).*x.^(L-1).*exp(-R.*x));

function f = vec_factorial(V)
for i=1:length(V),
   f(i) = factorial(V(i));
end;
```

References

[1] P. Juang, H. Oki, Y. Wang, M. Martonosi, L.-S. Peh, and D. Rubenstein, "Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet," in *The Tenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, October 2002.

[2] W. Ye, J. Heidemann, and D. Estrin, "An Energy-Efficient MAC Protocol for Wireless Sensor Networks," in *IEEE INFOCOM 2002*, June 2002.

[3] MICA2 Mote Datasheet.
`http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/6020-0042-01_A_MICA2.pdf`.

[4] C. Raghavendra and S. Singh, "PAMAS – Power Aware Multi-Access protocol with Signalling for Ad Hoc Networks," *ACM Computer Communications Review*, July 1998.

[5] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. Srivastava, "Optimizing Sensor Networks in the Energy-Latency-Density Design Space," *IEEE Transactions on Mobile Computing*, vol. 1, pp. 70–80, January-March 2002.

[6] C. Guo, L. C. Zhong, and J. M. Rabaey, "Low Power Distributed MAC for Ad Hoc Sensor Radio Networks," in *IEEE GlobeCom 2001*, November 2001.

[7] IEEE 802.11, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, 1999.

[8] J. Elson and K. Römer, "Wireless Sensor Networks: A New Regime for Time Synchronization," in *ACM Hot Topics in Networks 2002*, October 2002.

[9] Y.-C. Tseng, C.-S. Hsu, and T.-Y. Hsieh, "Power-Saving Protocols for IEEE 802.11-Based Multi-Hop Ad Hoc Networks," in *IEEE INFOCOM 2002*, June 2002.

[10] H. Woesner, J.-P. Ebert, M. Schläger, and A. Wolisz, "Power-Saving Mechanisms in Emerging Standards for Wireless LANs: The MAC Level Perspective," *IEEE Personal Communications*, pp. 40–48, June 1998.

[11] L. M. Feeney, "A QoS Aware Power Save Protocol for Wireless Ad Hoc Networks," in *IFIP Med-hoc-Net 2002*, September 2002.

[12] M. J. McGlynn and S. A. Borbash, "Birthday Protocols for Low Energy Deployment and Flexible Neighbor Discovery in Ad Hoc Wireless Networks," in *ACM MOBIHOC 2001*, October 2001.

[13] R. Zheng, J. C. Hou, and L. Sha, "Asynchronous Wakeup for Ad Hoc Networks," in *ACM MOBIHOC 2003*, June 2003.

[14] J.-C. Cano and P. Manzoni, "Evaluating the Energy-Consumption Reduction in a MANET by Dynamically Switching-off Network Interfaces," in *Sixth IEEE Symposium on Computers and Communications (ISCC'01)*, July 2001.

[15] E.-S. Jung and N. H. Vaidya, "An Energy Efficient MAC Protocol for Wireless LANs," in *IEEE INFOCOM 2002*, June 2002.

[16] E.-S. Jung and N. H. Vaidya, "A Power Saving MAC Protocol for Wireless Networks," tech. rep., University of Illinois at Urbana-Champaign, 2002.

[17] J. Deng and Z. J. Haas, "Dual Busy Tone Multiple Access (DBTMA): A New Medium Access Control for Packet Radio Networks," in *IEEE International Conference on Universal Personal Communication (ICUPC) 1998*, October 1998.

[18] J. M. Rabaey, M. J. Ammer, J. L. da Silva Jr., D. Patel, and S. Roundy, "PicoRadio Supports Ad Hoc Ultra-Low Power Wireless Networking," *IEEE Computer*, July 2000.

[19] J. M. Rabaey, J. Ammer, T. Karalar, S. Li, B. Otis, M. Sheets, and T. Tuan, "PicoRadios for Wireless Sensor Networks — The Next Challenge in Ultra-Low Power Design," in *Proceedings of the ISSCC*, February 2002.

[20] L. C. Zhong, J. Rabaey, C. Guo, and R. Shah, "Data Link Layer Design for Wireless Sensor Networks," in *IEEE MILCOM 2001*, October 2001.

[21] E. Shih, P. Bahl, and M. J. Sinclair, "Wake on Wireless: An Event Driven Energy Saving Strategy for Battery Operated Devices," in *ACM MOBICOM 2002*, September 2002.

[22] C. F. Chiasserini and R. R. Rao, "Combining Paging with Dynamic Power Management," in *IEEE INFOCOM 2001*, April 2001.

[23] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. Srivastava, "Topology Management for Sensor Networks: Exploiting Latency and Density," in *ACM MOBIHOC 2002*, June 2002.

[24] A. Woo and D. E. Culler, "A Transmission Control Scheme for Media Access in Sensor Networks," in *ACM MOBICOM 2001*, July 2001.

[25] Y. Xu, J. Heidemann, and D. Estrin, "Adaptive Energy-Conserving Routing for Multihop Ad Hoc Networks," Tech. Rep. 527, USC/Information Sciences Institute, 2000.

[26] Y. Xu, J. Heidemann, and D. Estrin, "Geography-informed Energy Conservation for Ad Hoc Routing," in *ACM MOBICOM 2001*, July 2001.

[27] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks," in *ACM MOBICOM 2001*, July 2001.

[28] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks," in *IEEE Hawaii International Conference on System Sciences 2000*, January 2000.

[29] T. van Dam and K. Langendoen, "An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks," in *ACM SENSYS 2003*, November 2003.

[30] R. Zheng and R. Kravets, "On-demand Power Management for Ad Hoc Networks," in *IEEE INFOCOM 2003*, April 2003.

[31] P. Nuggehalli, V. Srinivasan, K. Chebrolu, and R. Rao, "Energy Aware Sampling Schemes," in *IEEE WCNC 2000*, September 2000.

[32] S. Singh, M. Woo, and C. Raghavendra, "Power-Aware Routing in Mobile Ad Hoc Networks," in *ACM MOBICOM 1998*, October 1998.

[33] S. Banerjee and A. Misra, "Minimum Energy Paths for Reliable Communication in Multi-hop Wireless Networks," in *ACM MOBIHOC 2002*, June 2002.

[34] S. Doshi and T. X. Brown, "Minimum Energy Routing Schemes for a Wireless Ad Hoc Network," in *IEEE INFOCOM 2002*, June 2002.

[35] R. Krashinsky and H. Balakrishnan, "Minimizing Energy for Wireless Web Access with Bounded Slowdown," in *ACM MOBICOM 2002*, September 2002.

[36] T. Simunic, L. Benini, P. Glynn, and G. D. Micheli, "Dynamic Power Management for Portable Systems," in *ACM MOBICOM 2000*, August 2000.

[37] R. Kravets and P. Krishnan, "Application-driven power management for mobile communication," *ACM Wireless Networks*, pp. 263–277, July 2000.

[38] M. Anand, E. Nightingale, and J. Flinn, "Self-Tuning Wireless Network Power Management," in *ACM MOBICOM 2003*, September 2003.

[39] V. Raghunathan, C. Shurgers, S. Park, and M. B. Srivastava, "Energy-Aware Wireless Microsensor Networks," *IEEE Signal Processing Magazine*, March 2002.

[40] C. E. Jones, K. M. Sivalingam, P. Agrawal, and J. C. Chen, "A Survey of Energy Efficient Network Protocols for Wireless Networks," *ACM Wireless Networks*, July 2001.

[41] M. Stemm and R. H. Katz, "Measuring and Reducing Energy Consumption of Network Interfaces in Hand-Held Devices," *IEICE Transactions on Fundamentals of Electronics, Communications, and Computer Science*, August 1997.

[42] L. M. Feeney, "An Energy Consumption Model for Performance Analysis of Routing Protocols for Mobile Ad Hoc Networks," *ACM Mobile Networks and Applications*, June 2001.

[43] L. M. Feeney and M. Nilsson, "Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Environment," in *IEEE INFOCOM 2001*, April 2001.

[44] D. P. Helmbold, D. D. E. Long, and B. Sherrod, "A Dynamic Disk Spin-down Technique for Mobile Computing," in *ACM MOBICOM 1996*, November 1996.

[45] P. Greenawalt, "Modeling Power Management for Hard Disks," in *IEEE Conference on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS) 1994*, January 1994.

[46] X. Yang and N. H. Vaidya, "A Wakeup Scheme for Sensor Networks: Achieving balance between energy saving and end-to-end delay." In submission, 2003.

[47] Poisson Distribution from MathWorld.
`http://mathworld.wolfram.com/PoissonDistribution.html`.

[48] Gamma Distribution from MathWorld.
`http://mathworld.wolfram.com/GammaDistribution.html`.

[49] Gamma Function from MathWorld.
`http://mathworld.wolfram.com/GammaFunction.html`.

[50] `ns-2` – The Network Simulator.
`http://www.isi.edu/nsnam/ns`.