POWER MODE SCHEDULING FOR AD HOC NETWORK ROUTING

Santashil PalChaudhuri
RICE UNIVERSITY

Power Mode Scheduling for Ad Hoc Network Routing

by

Santashil PalChaudhuri

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE

Master of Science

APPROVED, THESIS COMMITTEE:

__________________________
Dr. David B. Johnson, Chair
Associate Professor of Computer Science

__________________________
Dr. Willy Zwaenepoel
Karl F. Hasselman Professor of Computer Science

__________________________
Dr. Edward W. Knightly
Associate Professor of Electrical and Computer Engineering

HOUSTON, TEXAS

MAY, 2002
Power Mode Scheduling for Ad Hoc Network Routing

Santashil PalChaudhuri

Abstract

An ad hoc network is a group of mobile wireless nodes that cooperatively form a network among themselves without any fixed infrastructure. Each node in the ad hoc network forwards packets for other nodes, to allow nodes not within direct wireless transmission range to communicate, using a routing protocol. Increasingly, power consumption within ad hoc networks is becoming a core issue for these low-power mobile devices.

This thesis focuses on a novel approach for energy conservation within the ad hoc network routing protocol. I develop and evaluate two types of mechanisms for reducing this power consumption by the routing protocol in the network. The routing protocol uses information from packets received promiscuously to improve routing performance. The first part of this work is a strategy for reducing this use of promiscuous mode intelligently, thereby saving energy but retaining all the benefits of being in promiscuous mode. In addition, a wireless network interface in sleep mode expends an order of magnitude less power than in idle mode, but no packets can be sent or received while in sleep mode. In the second part of this work, I propose two probabilistic algorithms for scheduling transition from idle mode to sleep mode. Performance evaluation of these strategies shows significant reduction in power usage, with only a slight decrease in performance.
I thank my advisor Dr. David Johnson for his guidance and constant encouragement throughout the duration of my thesis. The brain-storming sessions we had for hours together was most useful for crystallizing the nascent ideas and developing new ones. During the writing of my thesis, he became my English professor, spending enormous effort for editing the document to perfection.

I am grateful to the members of my research group: Amit, Yih-Chun and Jorjeta was always providing support, suggestions and acting as sounding board for all my crazy ideas. I am specially grateful to Yih-Chun and Amit for helping me in a number of ways in this project. Special thanks goes to Rajnish, Anirban, Anupam and Rajarshi for contributing to my thesis in unique ways.

My parents have always been a constant source of inspiration and encouragement, pushing me towards bigger goals. Finally, I thank myself for putting in Herculean effort in the last few months for completing this work.
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>ii</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>iii</td>
</tr>
<tr>
<td>List of Illustrations</td>
<td>vi</td>
</tr>
<tr>
<td>List of Tables</td>
<td>viii</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Motivation</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Thesis Contributions</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Organization</td>
<td>3</td>
</tr>
<tr>
<td>2 Background</td>
<td>4</td>
</tr>
<tr>
<td>2.1 Energy Consumption in a Wireless Network Interface</td>
<td>4</td>
</tr>
<tr>
<td>2.2 Power Mode Transition</td>
<td>7</td>
</tr>
<tr>
<td>3 Related Work</td>
<td>10</td>
</tr>
<tr>
<td>4 Algorithm for Non-Promiscuous Mode scheduling</td>
<td>12</td>
</tr>
<tr>
<td>4.1 Background</td>
<td>12</td>
</tr>
<tr>
<td>4.2 Algorithm</td>
<td>13</td>
</tr>
<tr>
<td>4.3 Simulation Experiments</td>
<td>16</td>
</tr>
<tr>
<td>4.3.1 Simulation Parameters</td>
<td>16</td>
</tr>
<tr>
<td>4.3.2 Simulation Results</td>
<td>17</td>
</tr>
</tbody>
</table>
## 5 Algorithms for Sleep-Mode Scheduling  

5.1 Extension to the IEEE 802.11 Power Saving Protocol  

5.1.1 Clock Synchronization Algorithm  

5.2 The Adaptive Sleep Algorithm  

5.3 Birthday Sleep Algorithm  

5.3.1 Probability Basis  

5.3.2 The Birthday Algorithm  

## 6 Simulation Results  

6.1 Results for the Adaptive Sleep Algorithm Results  

6.2 Results for the Birthday Sleep Algorithm Results  

## 7 Conclusion
Illustrations

2.1 State Transition Diagram from Idle to Sleep 8

4.1 State Diagram of Wireless LAN Card 14
4.2 Modified State Diagram of Wireless Card 15
4.3 An example of the P2NP policy 16
4.4 Percent Energy Saving 18
4.5 Packet Delivery Ratio 19
4.6 Percent Time spent in Promiscuous Mode 19

5.1 Adaptive Sleep Mode Strategy 25
5.2 Probability of a Matching Slot 29
5.3 At Least 1 Common Slot within First $k$ Slots, $n=50$ 29
5.4 At Least 1 Common Slot within First $k$ Slots, $n=10$ 30
5.5 The Birthday Algorithm Strategy 32
5.6 Example of a Sleep Indication Table (SIT) 33

6.1 Percent Energy Saving for the Adaptive Sleep Algorithm 35
6.2 Packet Delivery Ratio for the Adaptive Sleep Algorithm 35
6.3 Latency for the Adaptive Sleep Algorithm 36
6.4 Percent of Time in Sleep Mode for the Adaptive Sleep Algorithm 36
6.5 Percent Energy Saving for the Birthday Sleep Algorithm 37
6.6 Packet Delivery Ratio for the Birthday Sleep Algorithm 38
6.7 Latency for the Birthday Sleep Algorithm . . . . . . . . . . . . . . . . . . . 38
6.8 Percent of Time in Sleep Mode for the Birthday Sleep Algorithm . . . . . . 39
Tables

2.2 IEEE 802.11 2 Mbps WAVELAN PC card Linear Model Power Consumption Measurements [6] .......................... 8
Chapter 1

Introduction

1.1 Motivation

Energy consumption in mobile computing device is universally recognized as an important issue in the low-power mobile computing scenario. The sources of power consumption are communication and computation, with communication often being the chief power consumer.

The design of routing protocols is challenging in wireless ad hoc networks due to the mobility of nodes. Routing protocols that are purely on-demand have shown the best performance overall [2]. There has been some research on conserving power in these routing protocols. Most of this research on power conservation has focused on controlling the transmission power of the network interface. While significant in terms of reducing the power consumption in the wireless transmitter of a sender, it does little to conserve power among the other nodes — receivers, forwarders, and nodes not involved in this communication.

A network receiver interface has four different modes:

1. **Receiving in promiscuous mode:** On-demand ad hoc network routing protocols benefit from keeping the receivers in this mode at all times. In this mode, any node can overhear a packet sent between other nodes within its transmission range. By snooping on these packets promiscuously, a node gains information on the topology around it without needing to send any packets. It also caches the route of any overheard packet from its header for future use. Although this use of promiscu-
ous mode is useful for routing purposes, it has significantly higher consumption of power. The radio of a wireless network interface can detect after receiving the packet header, whether the destination of a packet is itself or a broadcast address. Hence, the node can discard packets not for itself even without receiving the packet fully, thereby typically using much less power in non-promiscuous mode than in promiscuous mode.

2. **Receiving in non-promiscuous mode:** As mentioned in the previous item, if a receiver is in non-promiscuous mode, it can filter out the packets not destined for itself in the wireless network interface hardware. This saves the energy required to receive and process the whole packet.

3. **Idle mode:** A receiver in idle mode can either receive or transmit. It burns power because it has to listen to the wireless medium all the time to determine whether or not there is a packet transmission going on.

4. **Sleep mode:** Sleep mode has very low power consumption. The network interface cannot receive or transmit in this mode until it is woken up into idle mode by an explicit instruction from the node. It takes a finite time to transition from sleep mode to idle mode and vice versa.

There is currently little work in the area of energy management in ad hoc networks that provides for transition between these different modes based on need. In this thesis, I propose several mechanisms to intelligently control these transitions.

### 1.2 Thesis Contributions

In this thesis, I introduce three algorithms for transitioning from one power mode to another in the wireless network interface of a node in an ad hoc network.
First, I propose an algorithm for a wireless card to transition from promiscuous mode to non-promiscuous mode. This algorithm is based on a property of some reactive ad hoc network routing protocols that have been proposed, that there is a *route discovery phase* that starts with a global broadcast to discover a new route to some destination.

Second, I propose two algorithms for transition of a wireless network interface from idle mode to sleep mode. In order for these algorithms to work, I show how time synchronization might be done in a multi-hop ad hoc network. The first of those two algorithms makes a node transition to sleep mode when it is not actively involved in sending, receiving, or forwarding data. The node periodically goes to idle mode and to sleep mode, with a duty cycle depending on the number of neighbors it has. The second algorithm is motivated by the *birthday paradox*. Each node randomly wakes up and goes to sleep. Neighboring nodes can communicate only when both are awake. A *Sleep Indication Map (SIM)* is sent by every node to all of its neighbors after every interval. Each node utilizes this SIM to schedule traffic to its neighbours. Finally, I model these algorithms in the *ns-2* network simulator and study their performance.

### 1.3 Organization

The remainder of this thesis is organized as follows. In Chapter 2, I describe the energy usage characteristics on which my design and evaluation is based. In Chapter 3, I discuss related work. Chapter 4 presents an algorithm for intelligently switching from promiscuous to non-promiscuous mode, as well as the performance evaluation of that algorithm. Chapter 5 explains my extensions to the 802.11 Power Saving (PS) protocol, and proposes two algorithms for scheduling transitions to different power modes, which are evaluated via simulations in Chapter 6. Finally, I conclude in Chapter 7 and discuss potential new research directions.
Chapter 2

Background

2.1 Energy Consumption in a Wireless Network Interface

In the evaluation of any algorithm for energy conservation, an estimate of energy consumption is necessary. In particular, the more closely a simulation reflects specific hardware, the more accurate the estimate of energy consumed in the simulation experiments is.

The wireless network interface from Feeney [6] has possible energy consumption modes:

- **Transmit**: While transmitting data
- **Receive**: While receiving data
- **Idle**: Can either transmit or receive but is doing neither; the node is monitoring the channel for incoming packets
- **Sleep**: Can neither transmit nor receive until explicitly woken up

Table 2.1, from Feeney [6], shows the actual current drawn by a wireless network interface card in the four modes above. This table shows that receive and idle mode require similar power, while transmit mode requires slightly greater power. Sleep mode requires more than an order of magnitude less power than idle mode. This implies that the network interface expends similar energy, whether it is just listening or actually receiving data. Hence, intelligently switching to sleep mode whenever possible will generally create significant savings.
Table 2.1: IEEE 802.11 2 Mbps Wavelan PC Card Characteristics [6]

<table>
<thead>
<tr>
<th>Mode</th>
<th>Specification</th>
<th>Measured</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>9mA</td>
<td>14mA</td>
</tr>
<tr>
<td>Idle</td>
<td>n/a</td>
<td>178mA</td>
</tr>
<tr>
<td>Receive</td>
<td>280mA</td>
<td>204mA</td>
</tr>
<tr>
<td>Transmit</td>
<td>330mA</td>
<td>280mA</td>
</tr>
<tr>
<td>Power Supply</td>
<td>5V</td>
<td>4.74V</td>
</tr>
</tbody>
</table>

The energy consumed by a wireless network interface when a node transmits, receives, or discards a packet can be described using a linear equation

\[ Energy = m \times size + b \]  

There is a fixed cost \( b \), associated with device mode changes and channel acquisition overhead. The incremental component \( m \), is proportional to the size of the packet. The total cost of a packet is the sum of the costs incurred by the sending node and all receivers.

Experimental results [6] confirm the accuracy of the linear model. Experiments were performed to determine values for the linear coefficients \( m \) and \( b \) for various operations of the network interface.

I use this energy model and the associated values [6,7] in the simulations in this thesis. The following discussion is about the assumptions of this model. The model does not consider link layer fragmentation. It is expected that the linear model would continue to apply, with each instance of fragmentation introducing a small step discontinuity, reflecting the fixed fragmentation overhead. This model also does not consider energy consumed in unsuccessful attempts to acquire the channel (media contention), or in messages lost due to collision, bit error, or loss of wireless connectivity. Such experiments were deemed difficult
to obtain in controlled experimental measurements [6].

Although bandwidth metrics count the number of packets sent over the wireless media, energy consumption metrics must take into account the reaction of all the network interfaces within wireless transmission range of the transmitters.

The operations of the wireless network interface can be divided into these three operations:

- **Broadcast Traffic**: The IEEE 802.11 broadcast traffic, the sender listens briefly to the channel and sends message if the channel is clear. Defining the fixed channel access cost \( b_{b\text{-send}} \) and \( b_{b\text{-recv}} \), and the incremental payload costs \( m_{b\text{-send}} \) and \( m_{b\text{-recv}} \), Equation 2.1 gives:

  \[
  \text{Transmit Cost} = m_{b\text{-send}} \times \text{size} + b_{b\text{-send}} \\
  \text{Receive Cost} = m_{b\text{-recv}} \times \text{size} + b_{b\text{-recv}}
  \]

- **Point-to-point Traffic**: For point-to-point traffic, the fixed cost includes both the channel access cost and the MAC negotiation cost, where the channel access cost is the same as that in the broadcast case. In the IEEE 802.11 MAC protocol, the source sends an RTS (Request-to-Send) control message, identifying the destination. The destination responds with a CTS (Clear-to-Send) message. Upon receiving the CTS, the source sends the data and awaits an ACK from the destination. This control overhead is taken into account in the fixed overhead to send a point-to-point packet. Defining the fixed channel access cost for sending and receiving a packet as \( b_{p\text{-send}} \) and \( b_{p\text{-recv}} \) respectively, and the incremental payload costs as \( m_{p\text{-send}} \) and \( m_{p\text{-recv}} \) respectively, Equation 2.1 gives:

  \[
  \text{Transmit Cost} = m_{p\text{-send}} \times \text{size} + b_{p\text{-send}} \\
  \text{Receive Cost} = m_{p\text{-recv}} \times \text{size} + b_{p\text{-recv}}
  \]
The incremental payload costs, $m_{p2p-send}$ and $m_{p2p-recv}$ are the same as those for broadcast traffic.

- **Non-destination Traffic**: Non-destination nodes in the range of either the sender or the receiver overhear some or all of the traffic. Depending on whether the node is in promiscuous mode, the node receives or discards the overheard packet.

The energy cost of discarding a packet is highly dependent on the MAC implementation. For the IEEE 802.11 MAC protocol, non-destination nodes in non-promiscuous mode can enter into a reduced energy consumption mode while data is being transmitted in the vicinity. For non-promiscuous nodes discarding traffic, Equation 2.1 gives:

$$Discard\ Cost = m_{discard} \times size + b_{discard}$$

For promiscuous nodes overhearing traffic, Equation 2.1 gives:

$$Discard\ Cost = m_{prom-recv} \times size + b_{prom-recv}$$

Table 2.2 gives the experimental data of all the cases detailed earlier [6]. The promiscuous discard mode has a negative value, because the power usage is calculated relative to the idle mode power consumption.

### 2.2 Power Mode Transition

From the above discussion, the energy expended by a wireless network interface in each mode is known. What is also important is the amount of energy expended and latency required to switch from one mode to another. While switching between active (transmit or receive), promiscuously idle, and non-promiscuously idle mode does not take any significant time or energy, transition to sleep mode is of particular interest.
Table 2.2: IEEE 802.11 2 Mbps WAVELAN PC card Linear Model Power Consumption Measurements [6]

<table>
<thead>
<tr>
<th>Operation</th>
<th>$m$ ($\mu$J/byte)</th>
<th>$b$ ($\mu$J)</th>
</tr>
</thead>
<tbody>
<tr>
<td>broadcast send</td>
<td>$1.9 \times size$</td>
<td>266</td>
</tr>
<tr>
<td>broadcast recv</td>
<td>$0.50 \times size$</td>
<td>56</td>
</tr>
<tr>
<td>p2p send</td>
<td>$1.9 \times size$</td>
<td>454</td>
</tr>
<tr>
<td>p2p send</td>
<td>$0.50 \times size$</td>
<td>356</td>
</tr>
<tr>
<td>prom recv</td>
<td>$0.50 \times size$</td>
<td>66</td>
</tr>
<tr>
<td>prom discard</td>
<td>$-0.58 \times size$</td>
<td>24</td>
</tr>
</tbody>
</table>

Figure 2.1: State Transition Diagram from Idle to Sleep
Figure 2.1 shows that there is a minimum critical time, only over which a state transition to sleep mode is useful. $T_{\text{trans}}$ amount of time is used by a network interface to go from idle to sleep mode and vice versa. During this time interval, no packet can be transmitted or received by the network interface, rendering it useless during this time. There is a need to minimize this useless amount of time as much as possible.

For example, in the ORiNOCO World PC Card from Agere Systems [19], the $T_{\text{trans}}$ time is specified to be less than 75 $\mu$sec. This implies that if 1% of the time is allowed for this transition, the sleep period has to be at least equal to 10 ms. No extra energy is required to make this transition, according to the specification of the ORiNOCO card.
Chapter 3

Related Work

A critical issue for mobile devices is power. Though access to data has become ubiquitous, access to power is a major constraint. A lot of research has been done in the area of low power. Low power is particularly important in mobile wireless ad hoc networks. Various attempts at saving power have been proposed, from the application layer [11] to the physical layer [17]. While significant power can be saved in all of these layers, I tackle the problem of saving power in the routing and MAC layer.

Solutions addressing power savings in the routing and MAC layer include the following:

- **Transmission Power Control:** In wireless communication, controlling transmission power has a large impact, ranging from power savings to transmission rate, interference, and error rates.

  Different work in this area has focused on topology control, determination of optimum transmission power, and increasing network throughput [3,8,13,16,18,21,22].

- **Power-Aware Routing:** In this area, the chief concern is extending the lifetime of a network. This work takes into account the remaining battery level of a node in determining optimum routes, such that none of the nodes are over utilized. The paper by Li et al [13] is a recent example of work in this area. This research area does not explicitly try to reduce power usage, but tries instead to maintain a uniformity in the power usage across all nodes, so that the network lifetime is extended.
• **Low Power Mode:** In this area of research, nodes are kept in one of the available power modes. Each node tries to be in the lowest power mode possible and still maintain connectivity in the network. Solving this problem of which mode to be in, is a hard problem for ad hoc networks. Little work has been done in this area. Work by Tseng et al [20] is the closest to my work. It proposes certain randomized algorithms for changing to sleep mode. These algorithms do not assume the existence of any time synchronization between the nodes. The work by Xu et al [23, 24] proposes a few algorithms to go into sleep mode, but it uses GPS for location information, to find node density. Chen et al [4] proposed an algorithm where some hosts serve as coordinators, which are chosen according to their remaining battery power, and the number of neighbors they can connect to.

McGlyan et al [14] use certain probabilistic methods similar to mine, for neighbor discovery. Time synchronization among nodes in a network is a problem several papers have addressed [5, 12, 15].
Chapter 4

Algorithm for Non-Promiscuous Mode scheduling

4.1 Background

In some on-demand routing protocols, caching overheard information is used to provide performance optimization. In order to explain how my strategy works, I will now introduce the mechanism of routing in reactive ad hoc routing protocols [10]. These protocols discover routes only when necessary for routing a packet to a destination. The protocol does not try to maintain routes between all the nodes for all the time. It uses two basic mechanisms, though there are a lot of optimizations possible of these:

- **Route Discovery**: New routes are discovered by a sender by broadcasting a ROUTE REQUEST packet. A node on hearing this ROUTE REQUEST packet, stamps its node id in the header of the packet, and rebroadcasts the packet to its neighbors. By continuing this process, the packet finally reaches the intended destination. This destination unicasts a ROUTE REPLY packet back to the source. Thereby a new route is discovered and can be used by any node in the path of the route. All the nodes within wireless transmission range of the path of the packet can also overhear the route and cache it for future use.

- **Route Maintenance**: If a node detects that a link in the source route breaks while the route is being used, that node returns a ROUTE ERROR packet to the source. If the source has another route to the destination in its cache, it uses that route. If no
other route to the destination is present at the source, the source starts another Route
Discovery.

4.2 Algorithm

In analyzing the mechanism of routing by the reactive ad hoc network routing protocol, I reach the following conclusions:

- A ROUTE REQUEST packet has a packet header that gives information on the path of traversal of that particular packet. Any node receiving this packet can determine a path back to the source, using the header information from this ROUTE REQUEST packet. Caching this information is usually very useful, as the ROUTE REQUEST packet provides a significant amount of information about the local topology. As a ROUTE REQUEST packet is broadcasted, it can be heard by all nodes in the vicinity, whether or not those nodes are in promiscuous mode.

- There is a fixed time-out interval after a source node sends a ROUTE REQUEST packet, within which time it expects to receive back a ROUTE REPLY packet. If the source does not receive any REPLY, it resends the ROUTE REQUEST packet with a higher time-out interval. If the source node does receive a ROUTE REPLY packet, it can use the route specified in the ROUTE REPLY to start sending DATA packets. Any neighboring node promiscuously overhearing packets will gather no more information from these DATA packets after the first DATA packet. Hence, I conclude that there is not much gain for a node to remain in promiscuous mode, after a data flow has started.

As described in Chapter 2, work on measuring energy consumption of a wireless network interface [6] has shown that the network interface of a node in promiscuous mode
expends more energy than being in non-promiscuous mode. Usually a node is always kept in promiscuous mode to overhear packets, and to take advantage of routes in those packets.

In Figure 4.1, I show the state diagram of a wireless LAN card. It has 3 modes: active, idle, and sleep.

- In **Active mode**, a packet is either being transmitted or being received.

- In **Idle mode**, the interface is just listening for packets — this is the default mode in typical networks.

- In **Sleep mode**, the interface cannot receive or transmit unless woken up. It has extremely low power consumption.

When a packet starts to be received by an interface, the interface switches to active mode. After receiving the packet, the interface changes its mode back to idle. In base station based wireless network, it is easy to schedule transition from idle to sleep mode, as the base station is the only node that can communicate directly with any other node. In an infrastructure-less ad hoc network, it is not possible to predict or schedule communication. Hence, each node always stays in active mode, listening for packets directed towards itself. For this reason the transition from idle to sleep mode is shown as “unknown” in Figure 4.1.

Using the analysis of Route Discovery and Route Maintainance in a reactive ad hoc network routing protocol, I propose an algorithm for saving power. Figure 4.2 shows the
mode transitions in this algorithm, called P2NP, separating the idle mode of Figure 4.1 into promiscuous idle and non-promiscuous idle modes. The transitions between promiscuous and non-promiscuous modes are explained below.

In this strategy, a node is kept in non-promiscuous mode. In this mode, the node can receive only packets addressed directly to itself or broadcast packets. This will enable the node to perform both as a forwarder, if it is explicitly mentioned in the source route of the packet, and as a receiver. Figure 4.3 shows the logic of the P2NP policy.

When a node receives a broadcast ROUTE REQUEST packet, it transitions to promiscuous mode. It remains in promiscuous mode for a certain Promiscuous Interval (PI) period. The value of PI is set such that within this timeout, the node should be able to capture most of the resultant ROUTE REPLY packets and also some of the initial data packets. It does not need to overhear any more data packets, as those packets contain no extra routing information. After the PI period, the node transitions back to non-promiscuous routing mode. The energy savings in this algorithm comes from the ability to discard rather than receive packets while in the periods of non-promiscuous mode.
The PI value is chosen experimentally based on the following factors: A node should be able to listen to the majority of the resultant ROUTE REPLY packets. This is done by statistically finding out a period for which a certain percentage of ROUTE REPLY packets have been received. As the Route Request Timeout period is an estimate of the maximum Round Trip Time (RTT) in a network, I use this estimate to guess the length of PI. In my simulations evaluating this algorithm, PI is varied in multiples of this RTT.

4.3 Simulation Experiments

To evaluate the protocol, I modeled it in ns-2 network simulator [1], which is a popular simulator for use in the networking community. The energy model described in Chapter 2 was also incorporated into the simulator for modeling the energy usage of the nodes.

4.3.1 Simulation Parameters

The simulation parameters were chosen to be similar to those used in the ad hoc networking research community, so that comparison of the simulation results is easier. The specific parameters were:
• A fixed area of $1500 \times 300$ meters was used in the simulation.

• The experiments were run for 900 simulated seconds each.

• The mobility pattern consisted of 50 nodes placed randomly in the fixed area. The nodes moved about using the random waypoint mobility model [10], with speeds ranging from 0 to 20 meters/second. Each node had a 30-second pause time.

• The communication used was 10 Constant Bit Rate (CBR) flows between 10 randomly chosen pairs of nodes. Each flow consisted of 4 packets per second, each packet having 512 bytes. The bandwidth of the wireless medium was set to be 2 Mbps.

• Each data point in the graphs are taken from an average of 3 runs, each run having a different communication and mobility pattern. The different movement and communication scenarios for each version of the protocol compared were identical.

### 4.3.2 Simulation Results

The simulations are done in the simulation environment described in the previous section. The Promiscuous Interval (PI) value was varied. A maximum Route Request Timeout (RRT) of 1 second was chosen for this experiment. The value was found by experimentation in this scenario for different RTT periods.

PI is varied from 0.01 times maximum RRT to 10 times maximum RRT. As maximum RRT is equal to 1 second, the $x$-axis actually gives the period of PI in seconds. Figure 4.4 shows the energy usage with changing values of PI. The graph shows an exponential decrease in energy usage for decreasing values of PI. This is because smaller PI intervals mean less time spent in promiscuous mode and hence less energy spent. The percentage energy savings is maximum for $PI = 0$, which implies that the node is never in promiscuous
mode. The results show about 5.5% energy saving for PI = 1 second. Although the energy savings are not remarkable in their values, it is significant that energy savings is achieved using a novel technique not proposed before.

Figure 4.5, shows a small drop in Packet Delivery Ratio (PDR) using this technique. The reason for the decrease in PDR is because the nodes are not always in promiscuous mode, learning new routes and topology information. For PI = 8 seconds, the PDR became constant at 98.7%. Decreasing PI to 1 showed only a slight decrease of PDR, which was still above 98%. Decreasing PI below 0.5 seconds has a sharp decrease in the PDR, as the node came close to being in non-promiscuous mode always. It is significant to note that, with PI = 0.01 seconds, a PDR of 95.5% is achieved, compared to a PDR of 90% when completely non-promiscuous. PI = 0.01 seconds yielded an energy saving of 6.9%.

Figure 4.6 shows the percentage time spent in promiscuous mode as a function of PI.
Figure 4.5: Packet Delivery Ratio

Figure 4.6: Percent Time spent in Promiscuous Mode
Chapter 5

Algorithms for Sleep-Mode Scheduling

All the nodes in an ad hoc network do not always participate in receiving, sending, and forwarding of data packets. Rather, many nodes do not need to be in the topology to maintain connectivity. If these nodes can be identified, then they can be put into sleep mode, rather than them staying in idle mode. Sleep mode uses an order of magnitude less energy than idle mode, so this would be very useful in terms of saving energy. The nodes that are probable candidates for putting into sleep mode are those that have not originated, forwarded, or received data for a certain fixed interval. These nodes are then sent to the sleep mode. However, these nodes are present in the network and can work as forwarders if the need arises. Being in sleep mode prevents a node from being able to receive any packets. So, there is a need to use some technique to periodically wake these nodes to idle mode.

5.1 Extension to the IEEE 802.11 Power Saving Protocol

The IEEE 802.11 MAC protocol [9] has support for a power-saving (PS) mode. In an infrastructure-based network, there is an access point (AP) that communicates data packets with each node. Periodically, the AP transmits a beacon frame spaced by a fixed beacon interval. Each beacon frame contains a Traffic Indication Map (TIM). The TIM frame contains identities of PS hosts for which the AP is holding buffered packets. A PS node monitors the TIM frames and remains awake for the rest of the interval if it is listed in the
TIM; otherwise the node goes to sleep.

In IEEE 802.11 ad hoc mode, PS nodes behave similar to infrastructure mode, except that all the nodes contend for sending a beacon, compared to only the AP in infrastructure mode. The *Ad hoc Traffic Indication Map (ATIM)* frame is sent by each node having buffered packets. The ATIM frame contains information about the destination for the buffered packets. The destination node receiving this ATIM frame stays awake for the rest of the period, and otherwise goes into sleep mode.

The problem with the PS mode of IEEE 802.11 is that it was designed for a single-hop (fully connected) network. When applied to ad hoc networks, certain problems arise. Tseng et al [20] have recently analyzed the problems:

- **Clock Synchronization:** IEEE 802.11 PS mode assumes a completely connected network, and hence a beacon frame transmission can synchronize all the nodes. In a multi-hop ad hoc network though, this simple clock synchronization is not possible because of communication delays and mobility. In Section 5.1.1, I suggest a clock synchronization algorithm for use in an ad hoc network.

- **Neighbor Discovery:** A wireless node is aware of the existence of neighbors only if there is an ongoing transmission while the node is awake. As PS mode reduces the period of being awake, it diminishes its chances of having accurate neighbour information as well as the chance of itself being discovered by other nodes. In Section 5.2, I describe an algorithm in which accurate prediction of the number of neighbors is not necessary.
5.1.1 Clock Synchronization Algorithm

In this section, I present a solution for synchronization of the clocks of all nodes in a multi-hop ad hoc network. Many network synchronization algorithms have been proposed over the years: a server periodically sends a message containing its current clock value to the clients. This work explores a form of time synchronization that is similar to that proposed by Elson et al [5]. They characterize four distinct components:

1. **Packet Build Time**: The amount of time spent at the sender to construct the message. This includes kernel protocol processing and variable delays introduced by the operating system. This time also accounts for the time required to transfer the message from the host to its network interface. This time is typically small and can be measured.

2. **Access + Transmission Time**: The access time is the amount of delay incurred waiting for access to the wireless channel. This is specific to the MAC protocol being used. The transmission time depends on the bandwidth and size of the packet.

3. **Propagation Time**: The amount of time needed for a message to transit from a sender to receivers, once it has left the sender. In a wireless network, this time is very small as it is simply the physical propagation time of the message through the medium at the speed of light.

4. **Receive Time**: The amount of time required for processing at the receiver’s network interface, to receive the message from the channel and notify the host of its arrival. This time is very small.

In this scheme, nodes occasionally send a *time stamp* message via a local non-propagating broadcast. The average period period between sending these *time stamp* messages depends
on the accuracy of the clock in the nodes, and the granularity of time synchronization desired. Using the above analysis, the actual clock time of the sender is calculated at the present time, by adjusting the time stamp time.

1. The *Packet Build time* is typically bounded by a small constant. The time required for it is added to the time stamp time.

2. The *Access time* varies depending on the congestion and number of neighbours in the local area. I assume that the congestion faced by the time stamp sender is the same over an interval of time. Hence, the access time is calculated for sending a packet, and that value is added to the time stamp time.

3. The *Receive time* and the *Propagation time* are small enough and can be neglected, or a fixed very small constant can be added to the received time stamp time, depending on the granularity of clock accuracy desired. *Transmission time* is constant (for a fixed-size time stamp message) and is also added.

This adjusted time stamp time will be very close to the actual current time of the sender, when the receiver receives the packet. Any node in the network will receive such time-stamps from all its neighbours periodically. The receiver averages each of those time stamps into its own clock value.

**Proof of convergence of the node clocks over time**

1. **Fastest clock:** For the node having the fastest clock, the clock of all its neighbours will be slower than this node’s clock. So, every time period, the averaging done will reduce its clock time and bring it closer to average clock time for the network.

2. **Slowest clock:** For the node having the slowest clock, the clock of all its neighbours will be faster than this node’s clock. So, every time period the averaging done will
increase its clock time and bring it closer to average clock time for the network.

This proves that the above clock synchronization strategy is bound to have convergence of the clock time for all the nodes in the network. Typically, crystal oscillators are accurate on the order of one part in \(10^{-6}\). This is equal to 1 ms drift in 1000 seconds. So, an accuracy of 1 ms can be achieved, by making the time interval between consecutive time-stamp broadcasts to be at most 1000 seconds.

5.2 The Adaptive Sleep Algorithm

The Adaptive Sleep Algorithm is the first of my two proposed protocols to attempt to schedule transition to sleep mode for saving energy without losing much in terms of performance.

The algorithm is explained in Figure 5.1. A node transitions from promiscuous mode to non-promiscuous mode in the same way as detailed in Section 4.2. The Promiscuous Interval (PI) is chosen from the experiments done in Section 4.3 by varying the PI length. Another interval called Sleep Interval (SI) is the period of time from the start of the most recent promiscuous mode, or a period when the transceiver of the node is idle, whichever is larger. After SI time interval, the periodic sleep cycle starts.

The periodic sleep cycle repeats at a period called the Sleep Period (SP). It also has a variable Duty Cycle (D), which is the fraction of the SP period that the node is in idle mode. For \((1 - D)\) fraction of SP, the node remains in sleep mode. Anytime a sender in sleep mode needs to send data, it transitions to active mode and starts sending. When a node is in sleep mode, if a ROUTE REQUEST is transmitted by a neighbor node, it will not be able to receive it. Depending on the length of time that the node is in sleep mode, it might wake up in time to receive the subsequent re-broadcasts of the ROUTE REQUEST messages by the node’s neighbours. Hence, the SP needs to be small enough so that, although a node may
have missed the initial ROUTE REQUEST, it wakes up in time to receive the subsequent re-broadcasts. In the worst case, the node in sleep mode might miss all transmissions of the initial ROUTE REQUEST. If the source node did not find any route to the destination, the node that was previously in sleep mode will have another chance to hear the re-broadcast of ROUTE REQUEST by the source node.

In densely populated ad hoc networks, many nodes can be interchangeably used for routing purposes. In a dense network, any node might sleep for a longer duration, as the relative utility (for forwarding purposes) of all nodes decrease. I use an approach similar to that used by Xu et al [23].

1. Estimate Neighbour Count: There are a number of published approaches for estimating the neighbour count of a node in a wireless network. In this work, a simple approach is used, where each node keeps a list of all the nodes from which it has received a packet whenever it is listening. Each entry in the list is deleted after a period of time $T_{update}$, unless the node receives another packet from that node. In this way any extra overhead of neighbour discovery messages is avoided.
2. **Estimate of Duty Cycle:** The idea is to keep the total idle time for all nodes in a neighbourhood constant. To that effect, the optimal solution would be

\[ \Sigma(N_i \times D_i) = k \]  

(5.1)

where \( N_i \) is a particular node in the neighbourhood, \( D_i \) is the duty cycle of that node, and \( k \) is a constant. To achieve an approximation of this optimal solution, an estimate of the neighbour count is used by each node. Let node \( N_i \) estimate its neighbour count to be \( nc_i \). Then this node will calculate its \( D_i \) from the following equation

\[ (nc_i \times D_i) = k/nc_i \]  

(5.2)

Solving for \( D_i \),

\[ D_i = k/(nc_i^2) \]  

(5.3)

This re-calculation of \( D_i \) is done as frequently as desired.

There is no need for accurate count of the number of neighbours, since the algorithm provides a natural feedback mechanism. If the neighbor count is underestimated by a node, that node stays awake for a longer time, during which it hears from more neighbours, and subsequently reduces its duty cycle.

A detailed simulation analysis of the performance of this protocol is done in the next chapter.

### 5.3 Birthday Sleep Algorithm

The Birthday Sleep Algorithm is the second of the two algorithms proposed in this thesis for scheduling transition to sleep mode for saving energy without losing much performance.
5.3.1 Probability Basis

The inspiration behind this protocol is the Birthday Paradox. The Birthday Paradox is the probability that at least two people in a room have their birthday on the same day. The paradox is that with as few as 23 people in the room, the probability exceeds one half. This is called a paradox, as it is counter-intuitive that with as few as 23 people and 365 possible days, the probability that any pair of people have the same birthday exceeds that of the reverse. This idea has been applied for neighbour discovery by McGlynn et al [14]. I use a similar idea for convergence between a receiver and a transmitter.

Over a period of $n$ fixed length slot periods, two wireless nodes independently and randomly select $m$ of these slots. In each of these $m$ slots chosen by a node, the node remains in idle mode listening for packets. In the remaining $n - m$ slots, it goes to sleep mode, thereby saving energy. The energy saved is approximately $(n - m)/n \times 100$ percent of the original energy consumed by the wireless network interface.

I calculate the probability that both the nodes are in idle mode in the same slot, for at least one slot out of the $n$ possible slots. Let the two nodes be A and B.

The number of ways A can choose $m$ out of $n$ things is

$$C(n, m)$$ (5.4)

The number of ways B can choose $m$ out of $n$ things is

$$C(n, m)$$ (5.5)

The total number of ways A and B can choose $m$ things each is

$$C(n, m)^2$$ (5.6)

The number of ways $k$ out of $m$ slots chosen by B are same as A is

$$C(m, k) \times C(n - m, m - k)$$ (5.7)
The number of ways at least 1 out of \( m \) slots chosen by B are same as A is
\[
\sum_{k=1}^{m} (C(m,k) \times C(n-m,m-k))
\tag{5.8}
\]

The probability that A and B have at least one idle mode slot in common is
\[
(C(n,m) \times \sum_{k=1}^{m} (C(m,k) \times C(n-m,m-k)))/(C(n,m) \times C(n,m))
\tag{5.9}
\]
\[
= (\sum_{k=1}^{m} (C(m,k) \times C(n-m,m-k)))/(C(n,m))
\tag{5.10}
\]

Figure 5.2 shows the plot of this probability function for increasing \( m \), for various values of \( n \). These results show the following properties:

- The probability that the two nodes will have a common slot increases with the increase in the ratio \( m/n \).
- The increase in probability is more rapid, the larger the \( n \).
- Energy is saved during the \( n-m \) slots where the receiver is kept in sleep mode.

For \( n = 10 \), the probability is 99.9% for \( m = 5 \). So, the amount of saved energy is \( 5/10 \times 100 = 50\% \). If \( n = 50 \), the probability is 99.9% for \( m = 14 \). Hence, the amount of saved energy is \( 35/50 \times 100 = 70\% \). This suggests that making \( n \) larger is going to be beneficial for saving energy.

If the total probability of there being a common slot is calculated, it might be the case that the common slot is the last of the \( m \) slots. In that case, there would be a significant number of retransmissions or a significant amount of latency involved in trying to send a packet to a neighbor. So, the probability of the common slot being within the first \( k \) of \( m \) positions should be analyzed.

The earlier analysis can be extended. Node A chooses \( m \) out of \( n \) slots. Node B chooses \( k \) slots, where \( k \leq n \), and each of the \( k \) slots is different than the \( m \) slots chosen by A.
Figure 5.2: Probability of a Matching Slot

Figure 5.3: At Least 1 Common Slot within First $k$ Slots, $n=50$
The number of ways of B choosing $k$ slots out of $n$ is

$$C(n, k)$$ (5.11)

The number of ways of B choosing $k$ slots out of $n - m$ is

$$C(n - m, k)$$ (5.12)

The probability of choosing $k$ slots all different than those chosen by A is

$$C(n - m, k)/C(n, k)$$ (5.13)

The probability of B choosing at least 1 slot the same as those chosen by A, within $k$ slots is

$$1 - C(n - m, k)/C(n, k)$$ (5.14)

This function is plotted for $n = 50$ in Figure 5.3 and $n = 10$ in Figure 5.4. Although the total probability might have been higher for larger $n$, the probability of a matched slot
within \( k \) slots is not helped by larger \( n \). In fact, smaller \( n \) give a larger probability that that there is a matching slot within lower values of \( k \).

I chose the value of \( n \) to be 10 instead of 50 for this reason. For \( n = 10, m = 7 \), there is a 95\% probability that the slots of the two nodes are together alive within first 2 alive slots.

### 5.3.2 The Birthday Algorithm

In this section, I take advantage of the conclusion of Section 5.3.1, where I showed the effect of choosing the probability with which a node stays in sleep mode. Using the strategy of time synchronization in Section 5.1.1, the close synchronization of the clocks of all nodes in an ad hoc network can be achieved.

For a particular value of \( n \), where \( n \) is the number of slots in a time interval \( T \), a value of \( m \) is chosen. \( m \) is the number of slots in which a node is in idle mode listening for packets. The performance in terms of latency and Packet Delivery Ratio (PDR) will be lower, compared to the case in which the node is always in idle mode, but the gain is in terms of energy saved by remaining in sleep mode. There is a trade-off involved.

The lower the ratio of \( m/n \), the larger the energy saving with a consequent lower performance. Hence, this ratio is a tunable parameter depending on the need for saving energy versus better performance. Let \( p_i = m_i/n \) denote the probability of staying awake for \( node_i \), where \( m_i \) is the number of slots \( node_i \) is awake and \( n \) is the total number of slots.

If a node is a sender or receiver of data or if it is an active forwarder of data, then the node always remains awake. Hence,

\[
p_i = 1, \text{ for } i \in S, D, F
\]

where \( S = \) Set is the set of all sources at the current time, \( D = \) the set of all destinations at the current time, and \( F = \) the set of all forwarders at the current time.
For all other nodes, $p_i$ depends on the policy.

- If the policy is to save energy on a per node basis, then $p_i$ is set to a constant value, for $i \notin S, D, F$.

- If the policy is to maximize the lifetime of a network, then $p_i$ may be set to a function of the remaining energy of node $i$, for $i \notin S, D, F$.

All nodes are awake for the first slot out of $n$ slots. Depending on $p_i$, each node randomly chooses $m_i$ out of the $n$ slots to be idle. Each node builds a *Sleep Indication Map (SIM)* containing $n$ bits, with the $r^{th}$ bit set to 1 or 0 depending on whether the node will be idle or asleep in the $r^{th}$ slot of that $T$ period. At the beginning of every $n$ slots taking $T$ time in total, the SIM is sent out by each node in a local broadcast. Hence, every node has knowledge of the sleep and idle slots for all of its neighbors. Figure 5.5 shows how this algorithm works.
A node uses the knowledge gained from the SIM packets from its neighbors to build a Sleep Indication Table (SIT). An example SIT is shown in Figure 5.6, where each row of the SIT is the SIM received from one of that node’s neighbors.

A node uses its SIT to schedule packets to any neighbor. The specific policies it uses for scheduling are as follows:

1. A node schedules a broadcast (primarily of ROUTE REQUEST packets) by finding the earliest slot with maximum number of neighbours in idle mode. Then that broadcast will have the maximum intended impact.

2. A node schedules the delivery of a unicast packet by looking for the nearest idle slot of the intended next hop neighbor.

3. If the information for a next hop node is not present in the SIT, then the source node sends the packet immediately.
Chapter 6

Simulation Results

This chapter presents the results of the simulation of the Adaptive Sleep Algorithm and the Birthday Sleep Algorithm. The simulation environment is same as that detailed in Section 4.3.

6.1 Results for the Adaptive Sleep Algorithm Results

To evaluate the Adaptive Sleep Algorithm, the duty cycle was varied from 10% of the Sleep Period (SP) to 90% of the SP, at 10% intervals.

Figure 6.1 shows the energy usage for different values of the duty cycle. There is a linear decrease in energy usage for decreasing duty cycle. This is because smaller duty cycles mean more time spent in sleep mode and therefore more energy savings. The percent energy savings is maximum when the duty cycle is as low as possible. A 50% duty cycle resulted in energy savings of 30%, while a 10% duty cycle resulted in energy savings of over 50%. This is a significant improvement in terms of energy savings.

Figure 6.2 shows the corresponding drop in Packet Delivery Ratio (PDR). A fall in PDR was expected because the nodes were in sleep mode, which resulted in some routes not being discovered or packets being dropped from full buffers. For a 50% duty cycle, the PDR dropped to 97.7% from 98.7% in always idle mode. For a duty cycle of 10%, the PDR is 96.8%. Hence the change in PDR is 1% and 2% for an energy saving of 30% and 50%, respectively.
Figure 6.1: Percent Energy Saving for the Adaptive Sleep Algorithm

Figure 6.2: Packet Delivery Ratio for the Adaptive Sleep Algorithm
Figure 6.3: Latency for the Adaptive Sleep Algorithm

Figure 6.4: Percent of Time in Sleep Mode for the Adaptive Sleep Algorithm
Figure 6.3 shows the latency as a function of duty cycle. The latency has a linear increase with decreasing duty cycle. The latency doubles for a duty cycle of 10% when compared to the node being idle always. The latency increases from 12 ms to about 25 ms for a duty cycle of 10%.

Figure 6.4 shows a linear relation between duty cycle and percent sleep time. This relationship is linear because the duty cycle is directly proportional to the percent sleep time. The slope of the line is not exactly -45 degree, because energy is also expended in transmission and receiving packets rather than only being in idle mode.

6.2 Results for the Birthday Sleep Algorithm Results

In this experiment, the total number of slots was 10. The number of slots awake randomly, \( m \), was varied from 5 to 10. This implies an idle percentage of 50\% to 100\% of total time.
Figure 6.6: Packet Delivery Ratio for the Birthday Sleep Algorithm

Figure 6.7: Latency for the Birthday Sleep Algorithm
Figure 6.5 shows the decrease in energy usage with decreasing values of $m$. This decrease is because smaller $m$ mean a larger time spent in sleep mode, thereby saving more energy. For 50% of the slots being idle, an energy saving of 46% is obtained, while 70% of the slots being idle shows a energy saving of 28%. This is a substantial improvement in terms of energy savings.

Figure 6.6 shows the drop in PDR. For 50% awake slots, the PDR dropped to 97.7% from 98.7% in always idle mode. For 70% awake slots, the PDR is almost 98%. Hence the change in PDR is 1% and 0.6% for an energy saving of 46% and 28% respectively. Figure 6.7 shows the latency as a function of $m$. The latency has a exponential increase with decrease in the value of $m$. Figure 6.8 shows a linear relation between duty cycle and percent sleep time, for the same reasons as discussed in the previous section.
Chapter 7

Conclusion

This thesis makes the following contributions in the area of power mode scheduling in ad hoc networks:

- Modeled a realistic energy model into the ns-2 network simulator.
- Identified energy saving by a wireless network interface staying in non-promiscuous mode. Proposed and modeled an algorithm for scheduling transition from promiscuous mode to non-promiscuous mode.
- Proposed an extension to IEEE 802.11 Power Saving (PS) mode for multi-hop wireless networks, for time synchronization among the nodes.
- Proposed and modeled two algorithms for scheduling transitions between idle and sleep modes.
- Simulated each of these algorithms to show energy savings.

Future work that can be done to extend this work further include the following:

- Transmission power control, which will be complementary to this work for saving energy in ad hoc networks.
- Using multi-rate data transmission to save energy.
- Energy conservation for multicast and broadcast protocols.
- Applying these ideas to sensor networks, where low-power is even more critical.
Bibliography


[6] Laura Feeney and Martin Nilsson. Investigating the Energy Consumption of a Wire-


