THESIS
1
2004
59711289

This is to certify that the
thesis entitled

Distributed Sleep-Scheduling Protocols for Energy
Conservation in Wireless Networks

presented by

Rohit R. Naik

has been accepted towards fulfillment
of the requirements for the

M.S. degree in Computer Science

[illegible]

Major Professor's Signature

07/08/2004

Date

MSU is an Affirmative Action/Equal Opportunity Institution

LIBRARY
Michigan State
University

PLACE IN RETURN BOX to remove this checkout from your record.
TO AVOID FINES return on or before date due.
MAY BE RECALLED with earlier due date if requested.

DATE DUE	DATE DUE	DATE DUE

6/01 c:/CIRC/DateDue.p65-p.15

DISTRIBUTED SLEEP-SCHEDULING PROTOCOLS FOR ENERGY CONSERVATION IN WIRELESS NETWORKS

By

Rohit R. Naik

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Department of Computer Science

2004

ABSTRACT

DISTRIBUTED SLEEP-SCHEDULING PROTOCOLS FOR ENERGY CONSERVATION IN WIRELESS NETWORKS

By

Rohit R. Naik

Wireless ad hoc networks are constrained by their battery power. An important design issue, thus, is to increase the network longevity. In this thesis, we present a distributed sleep scheduling protocol that can be used for implementing synchronous interface sleep for energy conservation in wireless ad hoc networks. The central idea of this protocol is to distribute a common sleep-awake cycle schedule among all nodes within a connected partition so that the nodes can turn their interface off during the sleep section of the agreed upon schedule, and they can communicate during the wake section of the schedule. By turning the interface off, the nodes can avoid idle listening consumption, which is a known reason for non-essential energy drainage in random-access network interfaces such as those running IEEE 802.11. Since the energy saving is achieved by synchronous interface sleeping, which reduces the active communication time, this protocol is suited for low to moderate network loading conditions. This distributed sleep-awake cycle scheduling and its associated long-term (100s of msec) synchronous sleep can be used as a complementary mechanism to those employed for short term (few msec) energy-saving asynchronous sleep for 802.11 MAC. In the latter, a node's interface is forced to sleep when ongoing transmissions, not involving this node, are detected in the neighborhood. While the short-term sleep is less effective at low loads, our mechanism works particularly well for low to moderate loading conditions. We present an ns2 based simulation model of the proposed distributed scheduling and sleep protocols for evaluating and comparing their energy performance with those of plain 802.11 MAC and a centralized scheduling mechanism.

Copyright by
Rohit R. Naik
2004

To mom and dad. Thanks for being there always.

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Subir Biswas for his guidance and support throughout this work. He infused in me an enthusiasm for research. Our long discussions helped me to to formulate the problem in this thesis and to come up with efficient solutions.

I would also like to thank Dr. Jonathan Shapiro and Dr. Matt Mutka for giving valuable comments and serving on my committee. I express my thanks to Dr. Shapiro for pointing out various problems in my algorithm and suggesting solutions.

Next, I would like to thank Samir Datta for helping me understand the internals of the Network Simulator. I also express my thanks to him for giving valuable suggestions during the implementation of the algorithm. I would like to thank my lab mates Jayanthi Rao and Nagendra Singh (Nagu!) for their valuable help during the course of my thesis. It was always a pleasure to brainstorm with them.

Last but not least, I would like to thank my parents, brother Rahul, and best friend Arthi (Raku!) for their unconditional love and support.

Table of Contents

List of Tables

List of Algorithms

List of Figures

Chapter 1

Introduction

Mobile ad hoc networks are an emerging technology with a wide range of potential applications such as battle-site networks, medical systems, robotic exploration, internetworking of participants in a meeting. A Mobile Ad-hoc NETwork (MANET) is a group of mobile wireless nodes which, upon deployment, cooperatively form an infrastructure-less network without any centralized control and service infrastructure. Since MANET nodes typically run from limited energy portable batteries, a critical design issue for future wireless Ad-Hoc networks is the development of suitable communication architectures, protocols and services that reduce power consumption, thereby increasing the operational lifespan of network enabled wireless devices. Energy conservation in a MANET node not only maximizes its own operational lifespan but it can also help maximizing the network lifespan and deferring network partitioning.

One of the main constraints on ad hoc networks is limited power. In most cases, it is not feasible to change or recharge the batteries for these nodes. In such a scenario, network longevity becomes an important design issue. In addition to essential energy consumption due to transmissions and receptions, there are three main sources of non-essential energy consumption in an 802.11 network. The first source is overhearing where a node receives traffic not destined to it. The second source is collision. Collisions result in retransmissions and hence an increase in energy expenditure. The third source, which we target in this paper, is idle listening, which corresponds to energy consumption when a wireless interface is in an idle state that is neither transmitting nor receiving. In 802.11, even in idle state an interface must be up and ready to receive possible traffic. In typical 802.11 interface hardware, the power consumption during idle listening is quite significant and is often comparable to the consumption rate during reception. Lucent's 915 MHz WaveLAN card, for instance, consumes 1.15W when idling, 1.2W while receiving and 1.6W while transmitting [1]. Measurements have shown that depending on the network loading situations, idle listening can consume up to 50–100% of the energy required for receiving [1, 2]. The effect of idle listening is especially evident in low loading conditions when nodes are in the idle state most of the time.

1.1 Design Goals for Mobile Ad Hoc Protocols

The various design goals that have to be kept in mind while designing protocols for wireless ad hoc networks are given below.

- **Self-configuration.** The wireless nodes can be deployed in remote or dangerous environments. This requires that the nodes be able to communicate with each other even in the absence of an established network infrastructure. Also, there are no guarantees about the placement of the nodes. In such a scenario, the nodes have to be self-configuring, that is they should not require any global control for the set up or maintenance of the network.

- **System lifetime.** This is one of the most important metrics in an ad hoc network and the focus of our interest. The network lifetime should be prolonged as much as possible. The system lifetime can be measured as the times at which nodes die. It can also be measured as the time required for the first node to die. Even the death of a single node can reduce the network performance by causing network partitioning.

- **Latency** It is defined as the average time taken to transmit a data packet. Latency can be an important issue depending on the application. Some application are not critically time sensitive and allow sub-second latency.

- **Throughput.** This has been a traditional metric for networks in general. This can be measured as the fraction of the data packets which are received successfully. Even though throughput is generally traded off for energy efficiency in ad hoc networks, we have to ensure that it degrade beyond an application-dependent limit.

With these design goals in mind, we propose a protocol that reduces energy consumption due to idle listening by using a distributed sleep-synchronization algorithm for maintaining a common network-wide sleep-schedule. Once synchronized, nodes can turn their interface off during the sleep section of the synchronized schedule, and they can communicate during the wake section of the schedule. By turning the interface off, the nodes can avoid energy consumption due to idle listening.

Much of previous research has been focussed on designing asynchronous sleep protocols. These protocols provide short term (few msec) energy savings by avoiding overhearing. Here, a nodes interface is forced to sleep when ongoing transmissions, not involving this node, are detected in the neighborhood. While the short-term sleep is less effective at low loads, our mechanism works particularly well for low to moderate loading conditions. In this thesis, we exploit the possibility of using synchronized sleep for achieving long term (100s of msec) energy savings.

1.2 Challenges

The problem of designing a distributed synchronous sleep protocol for achieving energy savings is a nontrivial issue. Some of the challenges in the design of such a protocol are given below.

- **Convergence time.** Because of the distributed nature of the algorithm, the nodes will take time to synchronize. An important goal is to ensure that a new node entering a system starts following the existing schedule as soon as possible. Also, when a node leaves a partition and joins a new partition, it should adapt to the new schedule quickly.

- **Drift.** There will be a drift between the schedules of neighboring nodes due to the various factors such as propagation delay, queueing delays, retransmissions, local clock drifts etc. Some of these delays such as processing delays, transmission delays are fixed while others such as propagation delay, queueing delays are variable and depend on the load.

- **Latency.** The traffic generation at the application layer is independent of the sleep i.e. a packet can be generated even when a node is asleep. When the nodes are in the idle state, their radio interface is shut off. During this time, they can not transmit or receive any messages. Packets can only be transmitted when the node is awake. This increases the end-to-end delay. One of the goals of our protocol is to lower the latency.

- **Packet scheduling.** As mentioned before, the application layer generates data independent of the sleep schedule of a node i.e. a packet can be generated even when a node is asleep. In such a case, packets are stored in a buffer and transmitted when the node wakes up. The effective load that a node has to undertake in the wake period becomes more. If all nodes start transmitting packets immediately after waking up, there will be a lot of collisions at the start of the wake cycle. In order to prevent this, we have to schedule the packets in order to minimize collisions. Also the rate at which applications can generate data is limited by the size of the data queue mentioned above.

1.3 Thesis Contributions

In this thesis, we come up with a distributed synchronization algorithm which will help reduce the energy wasted by the nodes in idle listening. The energy savings are due to long-term synchronous sleep (100s of msec). The main contributions of this thesis are as follows:

- We present a novel design of a distributed schedule synchronization algorithm that can be implemented on top of 802.11 MAC layer without requiring any change to the standard protocol.
- We apply this schedule synchronization mechanism for implementing an interface sleep-wake technique for significant reduction of the energy consumption due to idle listening. We show experimentally that the algorithm works for various topologies. Experimental results also show that our protocol achieves around 50% energy savings for a duty cycle of 50% as compared to standard IEEE 802.11.

1.4 Organization of the Thesis

The thesis is organized as follows. In Chapter 2, we review the literature on related topics of efficient protocol design in wireless ad hoc networks. In Chapter 3, we describe our distributed sleep-synchronization protocol in detail. We describe the experimental setup in Chapter 4. In Chapter 5, we evaluate and compare the performance of our protocol with plain 802.11 and a centralized scheduling algorithm. Chapter 6 concludes the thesis with a brief summary and a discussion on the scope for future work.

Chapter 2

Related Work

In this Chapter, we start by describing the native power management scheme in IEEE 802.11. We then review several power management protocols.

2.1 Power-Saving Modes in IEEE 802.11

IEEE 802.11 supports two power modes: active and power-saving (PS). The protocols for infrastructure networks and ad hoc networks are different. Under an infrastructure network, there is an access point (AP) to monitor the mode of each mobile host. A host in the active mode is fully powered and thus may transmit and receive at any time. On the contrary, a host in the PS mode only wakes up periodically to check for possible incoming packets from the AP. A host always notifies it's AP when changing modes. Periodically, the AP transmits beacon frames spaced by a fixed beacon interval. A PS host should monitor these frames. In each beacon frame, a traffic indication map (TIM) will be delivered, which contains IDs of those PS hosts with buffered unicast packets in the AP. A PS host, on hearing its ID, should stay awake for the remaining beacon interval. Under the contention period (i.e., DCF), a awake PS host can issue a PS-POLL to the AP to retrieve the buffered packets. While under the contention-free period (i.e., PCF), a PS host will wait for the AP to poll it. Spaced by a fixed number of beacon intervals, the AP will send delivery TIMs (DTIMs) within beacon frames to indicate that there are buffered broadcast packets. Immediately after DTIMs, the buffered broadcast packets will be sent.

Under an ad hoc network, PS hosts also wake up periodically. The short interval that PS hosts wake up is called the ATIM window. It is assumed that hosts are fully connected and all synchronized, so the ATIM windows of all PS hosts will start at about the same time. In the beginning of each ATIM window, each mobile host will contend to send a beacon frame. Any successful beacon serves as the purpose of synchronizing mobile hosts clocks. This beacon also inhibits other hosts from sending their beacons. To avoid collisions among beacons, a host should wait a random number of slots between 0 and 2 $\times CW_{min}$ - 1 before sending out its beacon.

After the beacon, a host with buffered unicast packets can send a direct ATIM frame to each of its intended receivers in PS mode. ATIM frames are also transmitted by contention based on the DCF access procedure. After transmitting an ATIM frame, the mobile host shall remain awake for the entire remaining period. On reception of the ATIM frame, the PS host should reply with an ACK and remains active for the remaining period. The buffered unicast packets should be sent based on the normal DCF access procedure after the ATIM window finishes. If the sender does not receive an ACK, it should retry in the next ATIM window. As for buffered broadcast packets, the ATIM frames need not be acknowledged. Broadcast packets then can be sent based on contention after the ATIM window finishes. If a mobile host is unable to transmit its ATIM frame in the current ATIM window or has extra buffered packets, it should retransmit ATIMs in the next ATIM window. To protect PS hosts, only RTS, CTS, ACK, Beacon, and ATIM frames can be transmitted during the ATIM window.

The PS mode of IEEE 802.11 works well with a infrastructure network where there are access points (APs) which have a constant supply of energy. In infrastructureless networks, the PS mode is designed for a single-hop network. For multi-hop MANET networks, several problems such as clock synchronization, neighbor discovery and network partitioning can occur.

2.2 Review of Power Management Protocols

A significant amount of research has been done in the area of energy conservation both at the routing layer [3, 4, 5, 6, 7, 8] as well as at the MAC and physical layers [9, 10, 11, 12, 13, 14]. Since the protocol proposed in this paper does not involve any end-to-end or routing layer syntaxes, we will present a summary of existing research only at the MAC and physical layers.

Existing MAC and physical layer research on energy conservation in 802.11 networks can be categorized in to three broad areas. Protocols in the first area deal with transmission power control for reducing the transmission energy by choosing optimal transmission power level on a per-transmission basis [15, 16]. The key idea is that under certain radio propagation model, it is sometimes more energy-efficient (in terms of the total power consumed) to use smaller transmission power and relay messages via intermediate nodes as compared to using the maximum transmission power and reaching all nodes directly. The primary limitation of the transmission power control protocols is that they only reduce the transmission energy, which is often only a tiny fraction of the total consumed energy at low loads. As a result, the overall energy savings can be insignificant with transmission power control. While power control has been recognized to be an effective means to increase network capacity [17], its usefulness in reducing energy consumption is heavily dependent upon the radio propagation model and hardware specifications [18].

Reducing the energy consumption due to overhearing is the second area of existing research. In the protocol PAMAS [10], overhearing is controlled using a forced interface sleep when there are ongoing packet transactions in a node's neighborhood. While PAMAS relies on an out-of-band busy-tone signaling channel to exchange information about packet transmission schedules, a single channel solution to reduce overhearing is proposed in [14]. In [14], the protocol infers the neighborhood transmission state from the 802.11 Network Allocation Vectors (NAV), avoiding the need for two radio transceivers. Also, as done in [10], turning an interface off completely has the problem of unacceptable transition latency as described in [2]. To avoid this problem, the protocol in [14] forces the interface to a low energy idling state but not to a completely off state such as sleeping. While the energy savings are relatively lesser in this approach, it is more feasible from an implementation standpoint. Protocols in both [10] and [14] are referred to as short term (packet duration) asynchronous interface sleep mechanisms.

While reducing consumptions due to overhearing, the asynchronous sleep mechanisms do not address the issue of idle listening. Reducing energy consumption due to idle listening is the third area of existing research. The protocol SMAC in [9] introduces a synchronous sleep mechanism for reducing idle listening. The protocol maintains long term (100s of ms) synchronized sleep and wake cycles so that the nodes can turn their interfaces off during the sleep duration to save idling energy and they can communicate during the wake periods. Although our proposed protocol and SMAC share the basic concept of long term synchronized sleep, the protocols differ in the following significant ways.

First, the synchronized sleep functions in SMAC are embedded within a specific MAC layer protocol that is designed for sensor networks. On the other hand, we have taken a layered approach in which the synchronized sleep functions are designed within an independent Synchronization Agent layer on top of MAC. The Synchronization Agents communicate in a peer-to-peer fashion using the services of the underlying MAC protocol. Compared to SMAC, this approach has the advantage that the synchronized sleep protocol is MAC layer independent and does not require any modification to the standard MAC layer syntaxes. In this paper we demonstrate its performance using unmodified 802.11 MAC protocol. Second, unlike in SMAC, in our protocol a node maintains and follows only one schedule at any time. Maintaining multiple schedules by a node requires it to remain awake during the wake period for all its maintained schedules, and therefore nodes with multiple schedules can have significantly large energy drainage compared to the nodes with only one schedule. It is important to note that the long-term (100s of ms) synchronized sleep for reducing idling consumption in our proposal can be used simultaneously with the short term (less than 10 ms) asynchronous sleep for reducing overhearing in 802.11 networks. This is possible due to the fact that no MAC layer modification is required and therefore, the synchronized sleep protocol can operate independent of the MAC layer asynchronous sleep protocols, as reported in [10] and [14].

Chapter 3

Distributed Sleep-synchronization Protocol

The aim of this protocol is to reduce energy consumption due to idle listening. This is achieved by making use of a concept of periodic sleep and wake cycles that are synchronized across each neighbor pairs in the network. To explain it further, any pair of neighbor nodes will maintain a synchronized sleep-wake schedule so that their interfaces can sleep during the sleep period and they can execute 802.11 MAC protocol for data transaction during the wake period. With this mechanism, it is expected that during low loading situations the idle listening consumption will be significantly slashed by putting interfaces to sleep states that typically has much lower power ratings. For example, if a node sleeps for 500 msec and stays awake for 500 msec alternately, then the duty cycle is 50%. The energy savings of this protocol depends on the wake-sleep duty cycle, which can be defined as the wake duration normalized by the total cycle duration. The wake-sleep duty cycle will also determine the sustainable loading range, and it should be appropriately dimensioned based on the applications and their loading requirements. Higher duty cycles can handle higher loads at the cost of lower idle energy savings.

The proposed distributed algorithm is implemented within a *SyncAgent* just above the MAC layer as shown in Figure 3.1. A *SyncAgent* runs in each node and it handles all sleep-synchronization related functions including distributed schedule maintenance and instructing the wireless interface hardware to sleep and wake up according to the converged schedule. The *SyncAgent* uses standard MAC layer broadcast services and it does not rely on any specific underlying MAC layer protocol.

An implicit assumption of our protocol is that all nodes within a network are required to use the same wake-sleep cycle duration and duty cycle. The duty cycle should be chosen depending on the network loading conditions and the cycle duration should be dictated by the sleep-to-wakeup latency overhead of the wireless interface card. We assume that the cycle duration and duty cycle parameters are either manually configured or dynamically determined across the network using an offline mechanism.

3.1 Sleep-Synchronization Mechanism

The basic mechanism for synchronization is to use a SYNC packet that a node broadcasts periodically with its own record of a sleep-schedule. A sleep-schedule is indicated by the next instance of the beginning of a sleep period, expressed as a relative time. Upon entering a network, a node attempts to discover any existing sleep schedule from the received SYNC packets. If it receives a SYNC packet within a pre-defined timeout period, the node adopts the received schedule and starts following it. If it does not receive one and the timeout expires then the node generates its own schedule and eventually broadcasts it using a SYNC packet. After a node adopts a schedule, it starts following a sleep/wake cycle as shown in Figure 3.2.

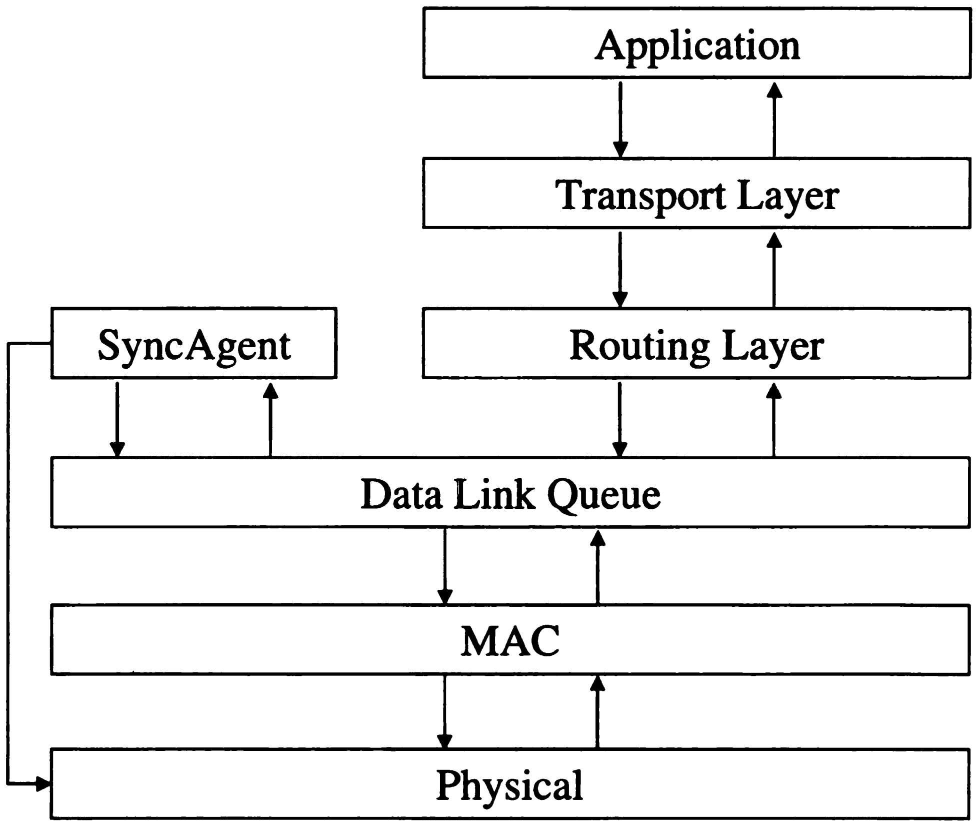


Figure 3.1: Layered Implementation of the Synchronization Protocol

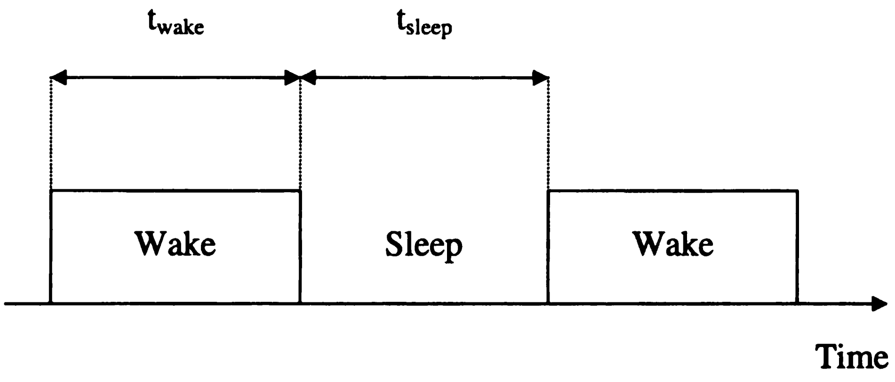


Figure 3.2: Periodic Wake and Sleep Cycle

3.2 Format of the SYNC Packet

The adopted SYNC packet format is shown in Figure 3.3. The SYNC packet is less than 30 bytes in size. Various fields in the SYNC packet are explained below.

- **{Owner IP Address, Sequence Number}.** A schedule in the network is uniquely identified by this pair. Each schedule is said to belong to a 'owner' node which had first created the schedule. The Owner IP Address is the IP address of that node of origin. The Owner IP Address field is used to break a tie when a node has to decide between multiple schedules received from different neighbors. As described in Section 3.3, the Sequence Number field is utilized for preventing the use of stale schedules.

- **Sender IP Address.** Every time a node sends a SYNC packet it populates this field with its own IP address. In other words, this field carries the IP address of the current sender of a SYNC packet.

- **Time of Next Sleep (t_k).** This indicates to the relative time after which the sending node is scheduled to sleep. It is important to note that t_k is a relative time. Using relative timing relieves our protocol from requiring global

Owner IP Address	Sequence Number	Sender IP Address	Time of next sleep (t_k)	Delay Offset (t_q)

Figure 3.3: SYNC Packet Format

time-synchronization using Network Time Protocol [19] or Global Positioning System (GPS).

- **Delay Offset (t_q).** After the SyncAgent (see Figure 3.1) generates a SYNC packet with all the described fields, it is handed to the MAC layer for a broadcast delivery. In higher loading situations it is possible for the SYNC packet to experience variable and unpredictable delivery delay that is contributed by the MAC buffer queueing, MAC back-off and retransmissions of unicast packets ahead of the SYNC packet. As a result, when a receiver uses this SYNC packet to adopt a sleep-schedule based on the Time of Next Sleep field, significant drift in synchronization may result. We use the Delay Offset field to alleviate this problem. Initially, the SyncAgent puts a time stamp in this field based on the time of the packet generation. Later, just before broadcasting the packet, MAC layer calculates the amount of time the SYNC packet has spent in the MAC buffer and MAC engine, and puts that difference in this field. When a node receives the SYNC packet, it computes the difference between Time of Next Sleep and Delay Offset to identify the actual time of the sleep. With this mechanism, the only remaining source of drift is the propagation delay.

3.3 Sleep-synchronization Algorithm

When a node enters a network, it waits for 'N' wake-sleep cycles to learn about any existing schedule through one or multiple SYNC packets. If it does not hear any schedule, the node creates its own schedule and broadcasts it using SYNC packets. If during these 'N' cycles, it receives one schedule then it adopts the received schedule and starts following it. But it is also possible for the node to receive more than one schedule during that 'N' cycle period and in that case a voting based resolution is adopted as follows.

Each received schedule is uniquely identified by its {*Owner IP Address, Sequence Number*} fields. The receiving node first attempts to find if there is any schedule that has been received from more than one node. If one is found, it is considered as a schedule of majority. The receiving node computes the average of the schedules of all received SYNC packets having this majority schedule and then it adopts the resulting average schedule. Averaging is done to reduce the drift caused by variation in propagation delays and the drift in system clocks. If the difference between the computed average schedule and the node's current existing schedule is less than a GUARD_TIME (a user chosen parameter), the node keeps following its current schedule. Else it starts following the computed averaged schedule. The use of GUARD_TIME allows an acceptable synchronization drift while preventing change of a node's sleep-schedule too often.

If all received schedules are distinct and no majority schedule is found then the *Owner IP Address* is used to break the tie among all received schedules. The receiving node chooses the schedule with lowest *Owner IP Address*. The pseudo code for the described algorithm is illustrated in Algorithm 1.

Unlike in SMAC [9], in our protocol a node maintains and follows only one schedule at any time. This means that within a network partition there is only one converged schedule (with unique Owner IP Address, Sequence Number combination)

```
Wait for N cycles
if no SYNC pkt was recvd then
    Make its own schedule
else
    if one SYNC pkt was recvd then
        mySchedule = recvdSchedule
    else
        Pick the schedule which is in majority. You can identify a schedule
        from its owner IP field
        if there is no schedule in majority then
            Pick all the SYNC pkts having the lowest owner IP
            From these, pick the ones having the highest sequence #
        end
        if there are still multiple schedules then
            avgSchedule = Average(schedules from all the chosen pkts)
            if $(avgSchedule - mySchedule > GUARD\_TIME)$ then
                mySchedule = avgSchedule
            else
                Keep following current schedule
            end
        else
            mySchedule = SYNC pkt schedule
        end
    end
end
```

Algorithm 1: Pseudo-code for SyncAgent of a node that enters a network

that is followed by all the nodes in that partition. A small amount of drift in the schedule across neighbor nodes is expected due to errors in relative sleep times caused by propagation delay and clock drifts. This drift may accumulate across nodes which are multiple hops apart. But since the drift in synchronization affects the MAC layer communication abilities only across the neighbors, large network wide cumulative drift does not have any adverse effects. Therefore, the goal of the protocol is to minimize drift only across the neighbor nodes.

In SMAC, maintaining multiple schedules by a node requires it to remain awake during the wake period for all its maintained schedules. As a result, the nodes with multiple schedules suffer from higher idling energy drainage than the nodes that maintain only one schedule. In our approach, by forcing each node to maintain only one schedule this energy concern has been addressed. With our one schedule per node approach, however, the following special mechanism is needed to handle situations when a node comes in contact with multiple neighbors that belong to more than one partition with multiple different schedules.

A node periodically (once in a pre-defined number of cycles) stays awake during an entire cycle. This is required for a node to detect if any of its neighbors is following a schedule that is different from its own. By staying awake for the entire cycle, a node is able to receive SYNC packets from its neighbors and can detect any schedule mismatch. The node will also send SYNC packets twice (once during the wake period and once during the sleep period) to make sure that its schedule reaches to all neighbors irrespective of their cycle phase differences.

SYNC packet broadcast by a node is a quasi-periodic process. After a broadcast of its own schedule, a node chooses a random number of sleep-wake cycles between 1 and 'N' and transmits the SYNC packet after that many cycles. Here, 'N' is a user defined parameter. Higher values of 'N' give rise to higher convergence time while reducing the overhead of control packets. Quasi-periodicity is useful for avoiding SYNC packet collisions due to simultaneous SYNC packet transmissions by neighbor nodes. SYNC packets are transmitted like other regular broadcast packets without any specific time interval reserved within a node's wake duration.

Note that the Sequence Number field of a SYNC packet is incremented only by the originator of a schedule. It is not incremented by a node during a rebroadcast. The pseudo code for Sequence Number update logic is shown in Algorithm 2.

```
while 1 do
    Wait for rand(N) cycles
    if (myIP == ownerIP of mySchedule) then
        Increment Sequence#
    end
    Broadcast SYNC packet containing your current schedule
end
```

Algorithm 2: Pseudo-code for Sequence Number Update

3.4 Data Transmission in the Presence of Periodic Sleep

Data packets from applications in an ad hoc node are generated independent of the sleep or wake state of the wireless interface. When the MAC layer receives a packet for transmission from the upper layer and the wireless interface is in sleep mode then the packet is temporarily buffered till the interface becomes operational during the next wake period according to the local sleep-wake schedule. Upon the interface wake up, buffered packets are transmitted based on the usual MAC procedures.

At steady state, since all neighbor nodes wake up roughly at the same time, attempts by all nodes to clear up their buffer in a burst may cause severe temporary MAC layer local congestions and subsequence packet drops. In order to prevent this, we introduce a random jitter between consecutive transmissions of the packets that were buffered during the interface sleep duration. By random staggering with this artificially introduced jitter, MAC layer collisions at the beginning of a wake period is minimized across the neighbor nodes.

The amount of jitter is dimensioned as follows. Once the interface wakes up (see Figure 3.4) after a sleep period, the node measures the accumulated buffer length to estimate:

- The number of packets that will need to be flushed during the wake period that

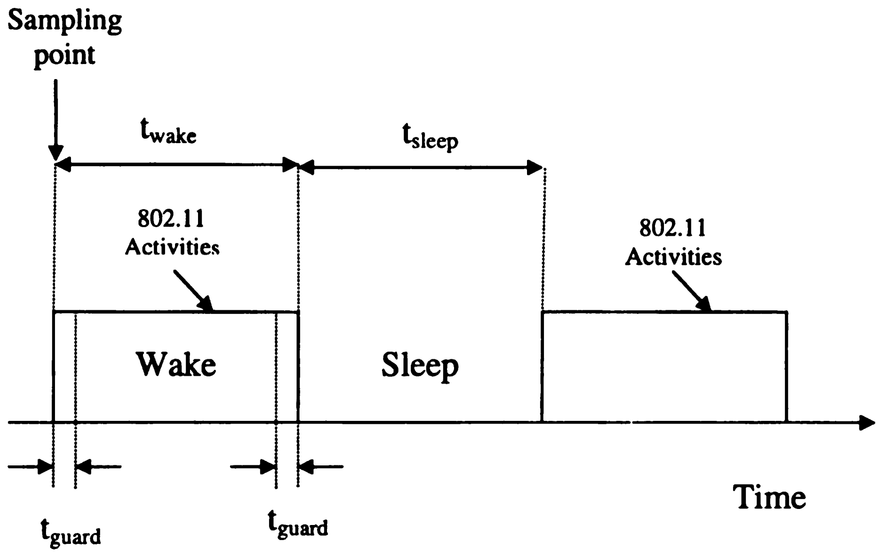


Figure 3.4: Calculating Jitter for Data Packets

has just started.

- The approximate packet arrival rate (from upper layers) by dividing the accumulated buffer length by the sleep duration, which is when the buffered packets have actually arrived.

The arrival rate information is useful to estimate the number of packets that are expected to be generated by the local applications during the current wake period. Effectively, the node has to transmit all the existing buffered packets and the packets that are expected to arrive during the rest of the wake period. In order to accommodate for the drift in schedules among two neighbors we introduce a guard time at the beginning and end of each wake period (see Figure 3.4). Although its interface is in the wake state, during this guard time, the node does not attempt to transmit any packet.

Now consider S_{Q1} to be the length of the data buffer at the start of a wake period. Then the estimated packet arrival rate (packets per second) can be expressed as:

$$\frac{S_{Q1}}{t_{sleep} + t_{guard}}$$

Therefore, the expected number of packets that can arrive during the MAC activities within a wake period is:

$$\frac{S_{Q1}}{t_{sleep} + t_{guard}}(t_{wake} - t_{guard})$$

Adding the number of packets in the current buffer and the packets expected during the current wake period, the effective number of packets that the node will have to transmit during the wake period (excluding the guard time) can be expressed as:

$$S_{Q2} = S_{Q1} + \frac{S_{Q1}}{t_{sleep} + t_{guard}}(t_{wake} - t_{guard})$$

$$S_{Q2} = S_{Q1}(\frac{t_{wake} + t_{sleep}}{t_{sleep} + t_{guard}})$$

Now the goal is to uniformly space all these packets over the effective transmission duration $t_{wake} - 2t_{guard}$. Therefore, the inter-packet spacing with a random jitter is computed as:

$$t_{jitter} = Random :: Uniform(\frac{t_{wake} - 2t_{guard})}{S_{Q2}}$$

After the t_{jitter} is computed at the beginning of a wake period, the node sends one packet in every t_{jitter} duration to the MAC layer for transmitting it to a neighbor.

Chapter 4

Simulation Models and Experimental Setup

The proposed sleep-synchronization protocol and its associate sleep mechanism for idling energy saving has been implemented using the ns-2 network simulator version 2.26 with the CMU wireless extension [20]. Ns is an event-driven simulator with extensive support for simulation of wireless network protocols. In order to implement the sleep-synchronization protocol, we have added several features to ns. The extensions include protocol architectures and energy dissipation models. Detailed discussion of these extension can be found in the Appendix.

4.1 ns-2 Simulation Models

In order to compare different protocols, it is important to have good models for different aspects of communication. This section describes the models that are used for channel propagation and radio energy dissipation.

4.1.1 Channel Propagation Model

In a wireless channel, the electromagnetic wave propagation can be modelled as a power law function of the distance between the transmitter and receiver. In addition, if there is no direct, line-of-sight path between the transmitter and receiver, the electromagnetic wave will bounce off objects in the environment and arrive at the receiver from different paths at different times. This causes multipath fading, which again can be modelled as a power law function of the distance between the transmitter and receiver. No matter which model is used, the received power decreases as the distance between the transmitter and receiver increases [21].

For the experiments described in this thesis, both the free space model and the multipath fading model are used depending on the distance between the transmitter and receiver, as defined by the channel propagation model in ns. If the distance between the transmitter and receiver is less than a certain cross-over distance 'd_c', the Friis free space model [22] is used (d^2 attenuation). Otherwise, the two-ray ground propagation model is used (d^4 attenuation). The cross-over distance is defined as follows:

$$d_c = \frac{4\pi\sqrt{L}h_r h_t}{\lambda} \tag{4.1}$$

where

L $\geq$ 1 is the system loss factor not related to propagation,

h_r is the height of the receiving antenna above ground,

h_t is he height of the transmitting antenna above ground, and

λ is the wavelength of the carrier signal.

If the distance is less than the d_c, then the received signal power, according to the Friis free space equation, is calculated as follows.

$$P_r(d) = \frac{P_t G_t G_r \lambda^2}{(4\pi d)^2 L} \tag{4.2}$$

where

$P_r(d)$ is the receive power given a transmitter-receiver separation of d,

P_t is the transmit power,

G_t is the gain of the transmitting antenna,

G_r is the gain of the receiving antenna,

λ is the wavelength of the carrier signal,

d is the distance between the transmitter and the receiver, and

$L \geq 1$ is the system loss factor not related to propagation.

This equation models the attenuation when the transmitter and receiver have a direct, line of sight communication, which will only occur if the transmitter and receiver are close to each other. If the distance is greater than d_c, then then the received signal power is calculated using the two-ray ground propagation equation as follows:

$$P_r(d) = \frac{P_t G_t G_r h_t^2 h_r^2}{d^4} \tag{4.3}$$

where

$P_r(d)$ is the receive power given a transmitter-receiver separation of d,

P_t is the transmit power,

G_t is the gain of the transmitting antenna,

G_t is the gain of the receiving antenna,

h_r is the height of the receiving antenna above ground,

h_t is the height of the transmitting antenna above ground, and

d is the distance between the transmitter and the receiver.

In this case, the received signal comes from both the direct path and a ground-reflection path [21] . Due to destructive interference when there is more than one path through which the signal arrives, the signal is attenuated as d^4. At the cross-over point, both equations give the same result.

In the experiments described in this thesis, an omnidirectional antenna was used with the following parameters: $G_t = G_t = 1$, $h_r = h_t = 1$m, no system loss ($L = 1$), 914 MHz radios, and $\lambda = \frac{3\times10^8}{914\times10^6} = 0.328$ m. Using these values, $d_c = 86.14$ m. Using these values, Equations 4.2 and 4.3 simplify to:

$$P_r(d) = \begin{cases} 6.82 \times 10^{-4}\frac{P_t}{d^2} & : d < 86.14m \\ 2.25\frac{P_t}{d^4} & : d \geq 86.14m \end{cases} \tag{4.4}$$

4.1.2 Radio Energy Model

A lot of research has been done in the area of low-energy radios. In this thesis, we assume a simple radio model where the transmitter dissipates energy to run the radio electronics and the power amplifier and the receiver dissipates energy to run the radio electronics. As discussed in the previous section, the power attenuation is dependent on the distance between the transmitter and receiver. For relatively short distances, the propagation loss can be modelled as inversely proportional to d^2, whereas for longer distances, the propagation loss can be modelled as inversely proportional to d^4. Power control can be used to invert this loss by setting the power amplifier to ensure a certain power at the receiver.

The radio characteristics used this thesis are given in Table 4.1. All the nodes communicate using half-duplex wireless radios that conform to 802.11-based 914 MHz Wave-LAN wireless radios with a bandwidth of 2Mbps and a nominal transmission radius of 250m. We use the same energy model as [23]. The ratio of the energy consumption in the Idle:Rx:Tx modes is about 1:1.2:1.7. The energy consumption for switching between awake and sleeping modes is negligible and thus not considered. We also assume that the energy consumed in the sleep mode is very small and therefore that is not considered in our model.

Radio Bandwidth	2 Mbps
Radio Transmission Range	250 m
Radio Interference Range	550 m
Transmit Power	1400 mW
Receive Power	1000 mW
Idle Power	830 mW

Table 4.1: Radio Parameters

4.1.3 MAC Layer Specifications

We use the IEEE 802.11 standard at the MAC layer. A brief explanation of the operation of IEEE 802.11 Distributed Coordinated Function (DCF) protocol is as follows. In this protocol, an exchange of RTS and CTS precedes data communication. Both RTS and CTS packets contain the proposed duration of data transmission. Nodes located in the vicinity of communicating nodes, that overhear either (or both) of these control packets, must themselves defer transmission for this proposed duration. This is called virtual carrier sensing and is implemented by using a variable called the network allocation vector (NAV). A node updates the value of the NAV with the duration field specified in the RTS or CTS. Thus the area covered by the transmission range of the sender and receiver is reserved for data transfer. Nodes located in this region do not initiate any transmission while communication is in progress. This RTS/CTS mechanism is used to overcome the hidden terminal problem [24].

The IEEE 802.11 MAC protocol uses a backoff interval to resolve channel contention. Before initiating transmission, a node S chooses a random backoff interval from a range [0, CW], where CW is called the contention window. Node S then decrements the backoff counter by 1 after every idle slot. When the backoff counter reaches 0, node S transmits the packet. If the transmission from S collides with some other transmission (collision is detected by the absence of a CTS), S doubles its CW, chooses a new backoff interval and attempts retransmission. The carrier sensing mechanism in IEEE 802.11 includes physical carrier sensing and virtual carrier sensing. IEEE 802.11 invokes the backoff procedure only after a channel has been sensed idle for DIFS duration. A shorter interframe space (SIFS), is used to separate transmissions belonging to a single dialog (i.e., a node performs physical carrier-sense for SIFS duration before transmitting CTS, DATA and ACK frames).

We make the following assumption about the MAC layer. The MAC layer gives priority to SYNC packets. Hence the SYNC packets are always queued at the beginning of the data link queue. Also, in order to take care of the delay caused by the time a SYNC packet stays in the queue, we assume that the MAC layer puts a time offset in the SYNC packet which takes care of the above mentioned delay.

4.2 Experimental Setup

4.2.1 Simulation Parameters

The sleep-synchronization protocol related parameters used in the simulation are mentioned in Table 4.2. A 28–byte SYNC packet size has been used with all the parameters specified in Figure 3.3. The cycle time is chosen to be 1 second, and the wake and sleep durations are 0.5 sec each, resulting in a duty cycle of 50%. Each node transmits a SYNC packet once every 5 cycles on an average. The size of the data packet is chosen to be 512 bytes.

At the link layer interface queue, the SYNC packets are given priority over the data packets by ensuring that a SYNC packet is always inserted at the head of the interface queue. This priority treatment, together with the delay offset mechanism described in Section 3.2 ensures controlled drift in synchronization across a pair of

Size of SYNC pkt	20 bytes
Size of data pkt	512 bytes
Cycle time	1 sec
Wake duration	0.5 sec
Sleep duration	0.5 sec
SYNC pkt broadcast	Once every 5 cycles

Table 4.2: Simulation Parameters

neighbor nodes.

4.2.2 Traffic Models

We use constant bit rate (CBR) traffic. For the static network, we use only local 1–hop traffic. This allows us to check the performance of the MAC layer without incurring additional route discovery costs. For the mobile scenario, we use multi-hop traffic. Ad hoc On-Demand Distance Vector Routing AODV [25] is used as the routing protocol. In the mobile environment, there is higher drift due to the overhead of AODV control packets. In order to account for the delay caused by this overhead, we assume that the MAC layer puts a time stamp on the SYNC packet. This time stamp is used to correct the offset caused by the time spent by the SYNC packet in the data link queue and retransmissions. This time is then subtracted at the receiver to get a better estimate of time of next sleep.

4.2.3 Network Topology and Mobility Model

We check the performance of our protocol in both static and mobile scenarios. For the static scenario, we test linear, circular, grid and random topologies. For the mobile scenario, we use the Random Way Point model.

Random Waypoint Model

In this mobility model, a node begins by staying in one location for a certain period of time (i.e., a pause time). Once this time expires, the node chooses a random destination in the simulation area and a speed that is uniformly distributed between [*speedmin; speedmax*]. The node then travels toward the newly chosen destination at the selected speed. After reaching the destination, it pauses for a random time p ($0 < p \leq max\_pause\_time$) before starting the process again.

4.2.4 Sleep Mechanisms

Centralized Sleep

In order to compare our algorithm with the best case scenario, we develop a centralized-sleep model. In the centralized version it is assumed that a global clock and a predefined sleep-wake schedule is centrally available to all the nodes so that they can control their sleep-wake operation without having to deal with the synchronization drift that is present in the distributed case. There is no overhead of SYNC packets.

Distributed Sleep

This is our implementation of distributed synchronous sleep. Here a node uses SYNC packets to let neighboring nodes know its schedule.

Chapter 5

Experiments and Results

We carry out experiments for plain 802.11, centralized sleep and distributed sleep. We briefly summarize the operation of each of these protocols.

In plain 802.11, nodes are awake all the time. The IEEE Distributed Coordinated Function (DCF) protocol is used for data communication. This represents the baseline scenario, where there are no savings in the idle energy. In the centralized sleep mechanism, all nodes follow a periodic sleep-wake schedule. It is assumed that a global clock and a predefined sleep-wake schedule is centrally available to all the nodes. This protocol produces better result as it has knowledge about the schedules of all the nodes and does not have to deal with synchronization drifts. However, it requires the nodes to be equipped with GPS or NTP. This represents the best case scenario. In the distributed sleep mechanism, all the nodes follow a periodic sleep-wake schedule. But they have to synchronize themselves on the fly. This protocol has the advantage of being distributed, self configuring and not requiring location information for synchronization purposes.

The performance of the distributed sleep mechanism depends on the topology. For example, in a dense topology such as a grid, there are more chances for collision for a SYNC packet. The performance of our protocol also depends on the traffic

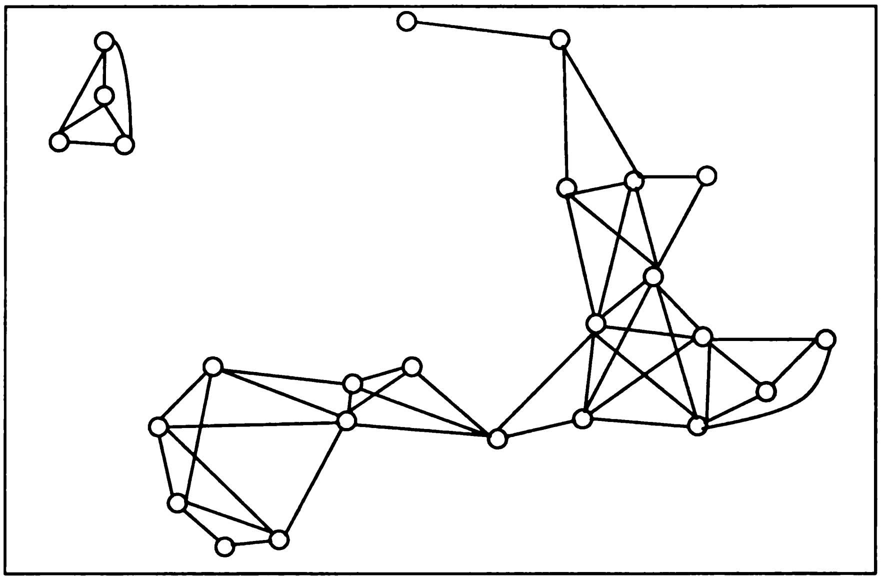


Figure 5.1: 25-node Random Test Network

model used. For example, multi-hop traffic may suffer extended delays due to the periodic sleep-wake cycles. To understand these dependencies, we have considered several simulation scenarios. Experiments were carried out for random, linear, circular and grid topologies. For each topology, we evaluated the performance of the three protocols for single-hop and multi-hop traffic. For the multi-hop scenario AODV was used as the routing protocol. Each simulation was run for 500 seconds.

5.1 Random Topology

For the random topology, we used 25 nodes randomly distributed in a 1300mx800m region as shown in Figure 5.1. In this section, we evaluate the performance of the three protocols for the above random topology for both one-hop and multi-hop traffic.

5.1.1 One-hop Traffic

In this scenario, each node establishes a long-lived CBR flow (400 sec) with one of its randomly chosen neighbors. The mean rate at which nodes generate packets is varied to characterize the protocol performance at different loading conditions.

The idle energy consumption per packet as a function of load is presented in Figure 5.2. Idle energy consumption refers to the average idle energy consumed for packets that are successfully delivered. Due to lower packet transmission activities, at low loads, the idle energy consumed by 802.11 is very high. As a result, the room for savings is significant. At a load of 5 Kbps per flow, with the proposed distributed sleep mechanism, the reduction of idle energy from the plain 802.11 is nearly 49%, which is the percentage of the time a node sleeps. At the same load, the savings with the centralized case is 54%. Energy consumption is generally higher for the distributed case as compared to the centralized case primarily due to the overhead of SYNC packets and drift between schedules across neighbor nodes.

At lower loads (up to 25 Kbps per flow), we found that idle energy accounted for 60-95% of the total consumed energy. Hence, savings of 49% in idle energy can potentially lead to significant savings in the total energy. At higher loads, transmission and reception energies become significant.

Note that in few instances the synchronized sleep protocols are able to achieve a percentage of idle energy savings that is more than the percentage of the time a node sleeps. The reason for this is the following. Since the data packets from applications are continuously generated during both the wake and sleep periods, with 50% duty cycle, the effective data load during the wake period is twice that of the plain 802.11 case. This higher effective load reduces the idle energy expenditure in the wake period, thus accounting for the increased savings.

The tradeoffs due to the periodic sleep are network throughput, MAC layer packet drops and delivery delay. Figure 5.3 shows the network throughput for the three pro-

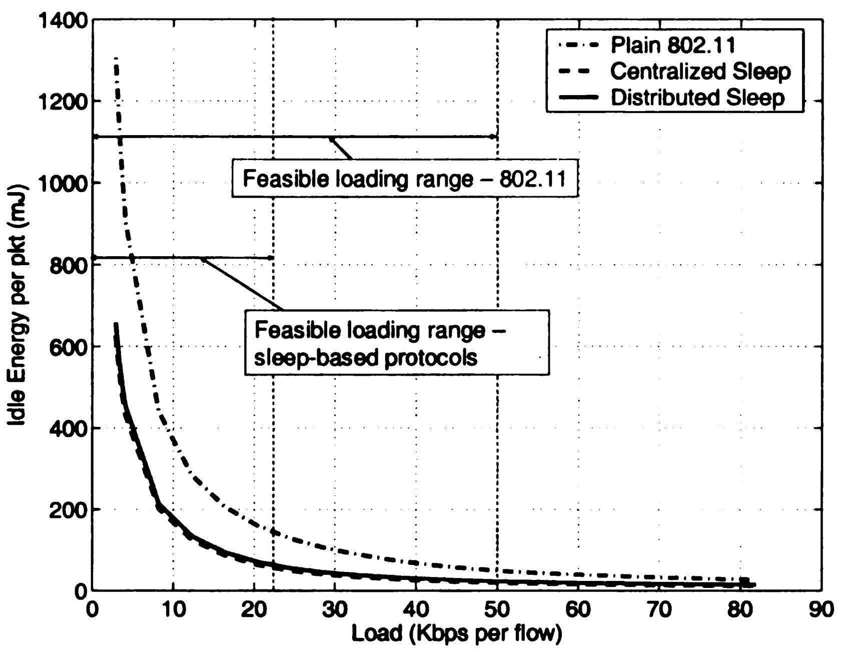


Figure 5.2: Idle Energy Consumption

tocols under consideration. While for the plain 802.11 the network throughput starts saturating at loads around 50 Kbps per flow, the throughputs for both centralized and distributed sleep mechanisms starts saturating at loads around 22 Kbps. The point of saturation indicates the sustainable throughput and feasible loading range of the network.

At similar MAC layer packet drops, with 50% wake-sleep duty cycle, the expected throughput saturation for the distributed and centralized sleep cases would be around 25 Kbps per flow which is half of the 802.11 case. But the MAC layer drop rates are higher for both the sleep protocols because of the higher effective loads during the wake periods. As explained before, the effective load during the wake period is higher because a node does not transmit packets that are generated by its applications during the sleep period and it transmits all the packets during the wake period. This higher MAC layer packet drops explain why the sleep based protocols saturate at a load of 22 Kbps as opposed to the expected 25 Kbps per flow.

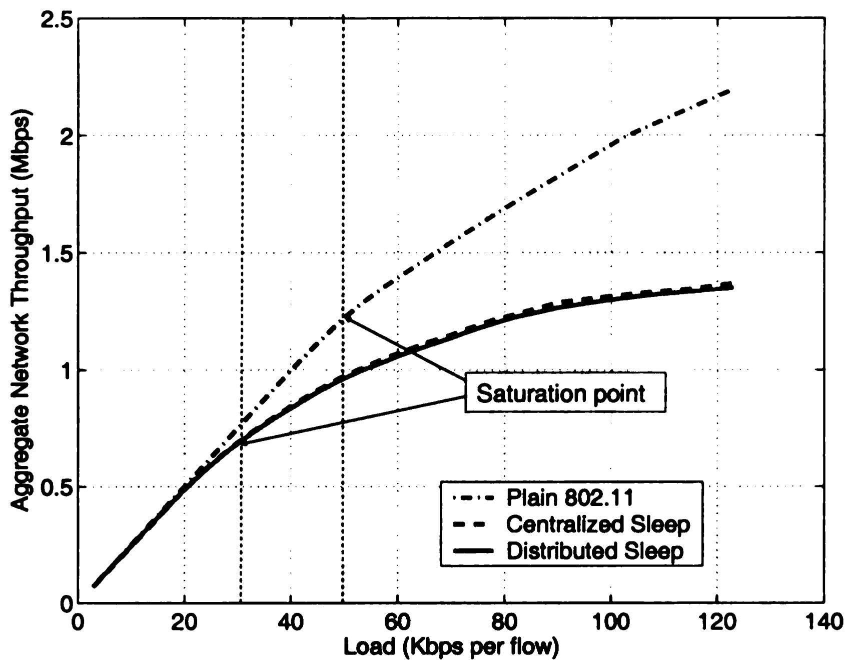


Figure 5.3: Sustainable Network Throughput

Packet drop rates for all three protocols are shown in Figure 5.4. The maximum drop rates for both the sleep based protocols are 5% within the feasible loading region, which is up to 22 Kbps per flow. And the maximum packet drop rate for the plain 802.11 protocol is approximately 2.5% within its feasible loading region, which is up to 50 Kbps per flow. Additional packet drops in the sleep based protocols are mainly due to the higher effective loads during the wake periods as explained before. To summarize, within the feasible loading regions the proposed sleep mechanism does not increase any significant packet drops compared the plain 802.11. The maximum drop rate of 5% is considered well within the tolerable limits of most of the data oriented applications.

Within the feasible loading zone, the MAC layer packet delivery delay is directly proportional to the wake-sleep cycle duration. In other words, for a given duty cycle (50% for our experiments), with larger wake-sleep cycle duration, the delay is higher.

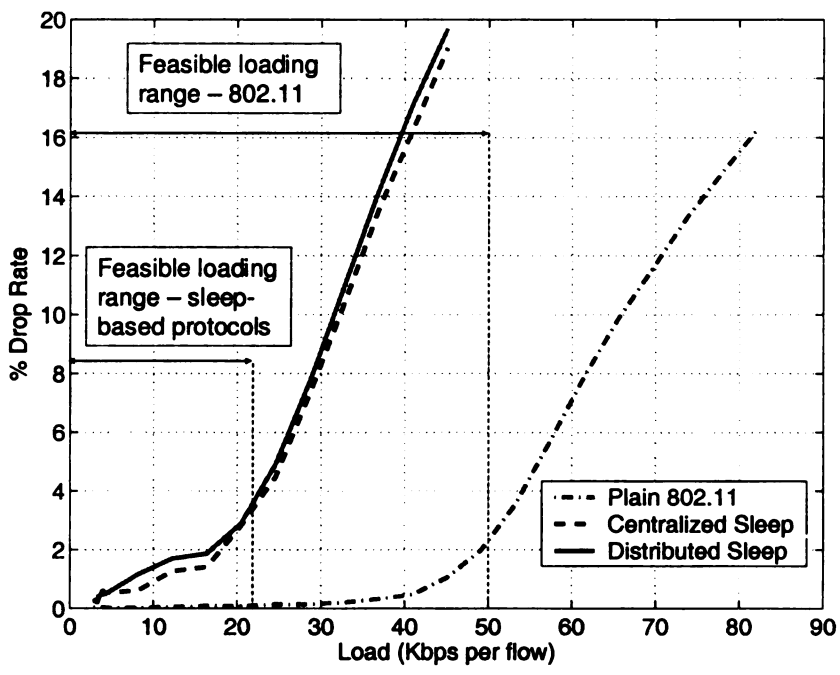


Figure 5.4: Packet Drop Rate

The reason for this is that packets coming from the application layer during a sleep period remain queued up in the data buffer for the entire sleep duration. With larger cycle periods, the sleep durations are also larger, and as a result, the experienced delays are more.

The effects of varying sleep-wake cycle duration with changing loading conditions are shown in Figure 5.5. All the graphs in this figure represent our proposed distributed sleep synchronization protocol running at 50% wake-sleep duty cycle. As expected, the delay increases with increasing cycle duration. At the maximum feasible loading condition (22 Kbps per flow), the delay range is from 58ms (for 200ms cycle duration) to 250ms (with 1sec cycle duration). For the plain 802.11 protocol, with 1 sec cycle duration, the delay at the maximum feasible loading (50 Kbps per flow) was found to be around 12ms. With the distributed sleep synchronization protocol, the saving in idling energy and the maximum feasible network load depend on the

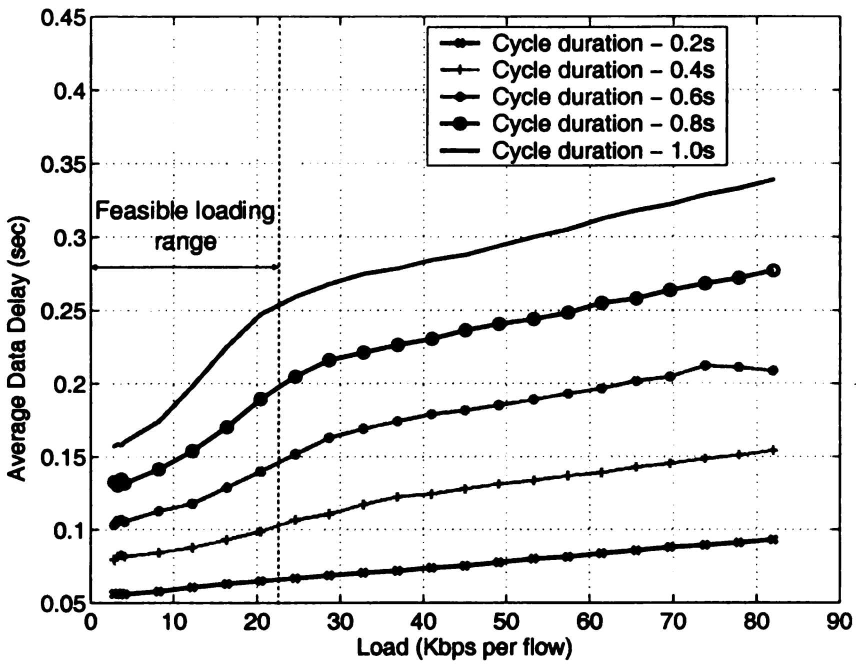


Figure 5.5: Delay for Varying Sleep-wake Cycle Durations

wake-sleep duty cycle but not on the wake-sleep cycle duration. However, as evident from Figure 5.5, from the packet delivery delay standpoint, it is desirable to use a cycle duration as small as feasible. The lower bound for the sleep-wake cycle duration will be determined by the wake-up latencies of the wireless interface hardware. Most of today's wireless interfaces typically take 100s of milliseconds to transition from sleep to any other state. Lucent's WaveLan card, for instance, takes more than 100ms [1] for this transition. This transition latency puts a limit on the minimum cycle duration that can be used.

Figures 5.6 and 5.7 show the average energy spent on transmission and reception of a packet, respectively. The transmission and reception energies with sleep is higher than that without sleep. This is because of the overhead of SYNC packets and the increased number of retransmissions. The sleep mechanism increases the effective load during the wake period. Due to this, there are more number of retransmissions per

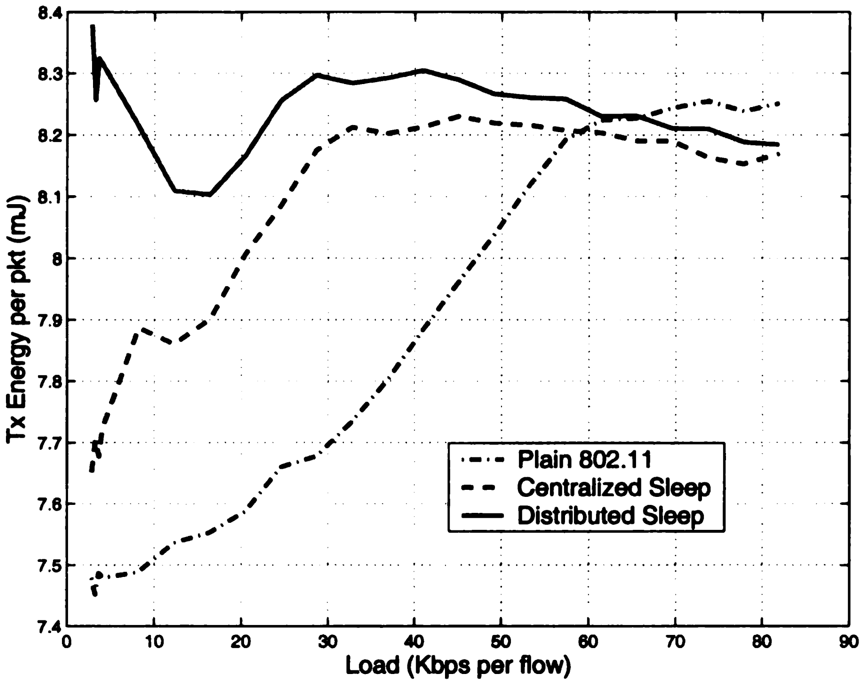


Figure 5.6: Tx Energy Consumption

successful delivery. We can see that at lower loads (up to 22 Kbps), they contribute very little (around 3%) to the total energy spent. This justifies our approach of using periodic sleep to reduce idle energy expenditure, which is a significant cause of energy consumption at lower loads.

5.1.2 Multi-hop Traffic

In this scenario, each node establishes a long lived CBR connection (400 sec) with one other randomly chosen node. The Ad hoc On-Demand Distance Vector (AODV) protocol [25] is used for route establishment and maintenance. A brief overview of the AODV protocol operation is given below.

The AODV algorithm enables dynamic, self-starting, multihop routing between participating mobile nodes wishing to establish and maintain an ad hoc network. AODV allows mobile nodes to obtain routes quickly for new destinations, and does not

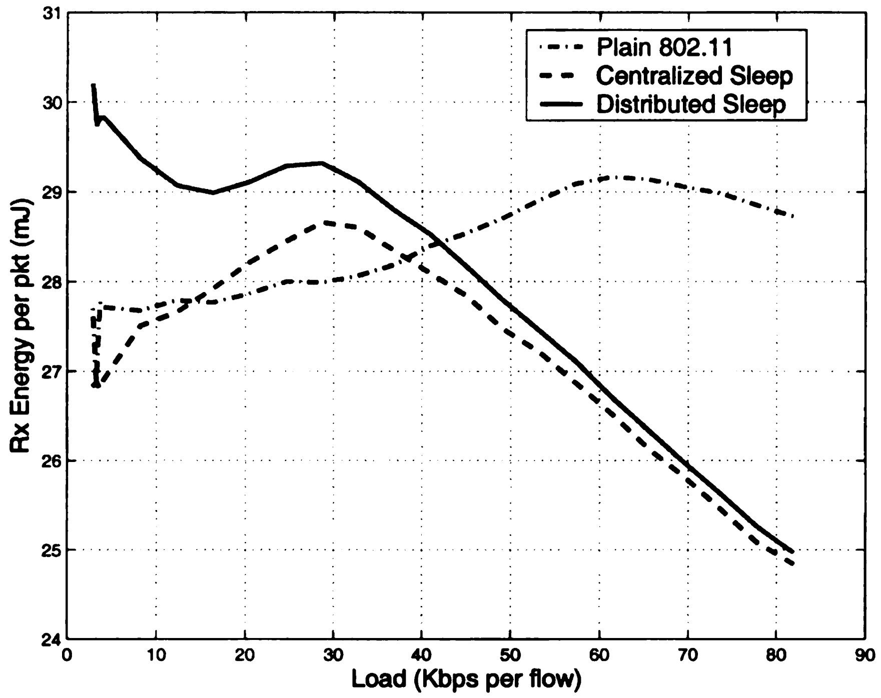


Figure 5.7: Rx Energy Consumption

require nodes to maintain routes to destinations that are not in active communication. AODV builds routes using a route request / route reply query cycle. When a source node desires a route to a destination for which it does not already have a route, it broadcasts a route request (RREQ) packet across the network. Nodes receiving this packet update their information for the source node and set up backwards pointers to the source node in the route tables. In addition to the source node's IP address, current sequence number, and broadcast ID, the RREQ also contains the most recent sequence number for the destination of which the source node is aware. A node receiving the RREQ may send a route reply (RREP) if it is either the destination or if it has a route to the destination with corresponding sequence number greater than or equal to that contained in the RREQ. If this is the case, it unicasts a RREP back to the source. Otherwise, it rebroadcasts the RREQ. Nodes keep track of the RREQ's source IP address and broadcast ID. If they receive a RREQ which they have already processed, they discard the RREQ and do not forward it.

As the RREP propagates back to the source, nodes set up forward pointers to the destination. Once the source node receives the RREP, it may begin to forward data packets to the destination. If the source later receives a RREP containing a greater sequence number or contains the same sequence number with a smaller hopcount, it may update its routing information for that destination and begin using the better route.

As long as the route remains active, it will continue to be maintained. A route is considered active as long as there are data packets periodically travelling from the source to the destination along that path. Once the source stops sending data packets, the links will time out and eventually be deleted from the intermediate node routing tables. If a link break occurs while the route is active, the node upstream of the break propagates a route error (RERR) message to the source node to inform it of the now unreachable destination(s). After receiving the RERR, if the source node still desires the route, it can reinitiate route discovery.

We see that the energy consumption for multihop traffic is much more due to the overhead of AODV control packets. Total energy consumption is around 5000 mJoules/pkt as opposed to 1300 mJoules/pkt for one-hop traffic. We still achieve 50% energy savings as can be seen from Figure 5.8.

The network throughput for the three protocols is shown in Figure 5.9. While the network throughput for the plain 802.11 case starts saturating at around 10 Kbps, the throughput for both centralized and distributed sleep mechanisms start saturating at 4 Kbps. The drop rate with sleep is only 2.5% more than that without sleep within the feasible loading region as can be seen from Figure 5.10.

The delay experienced by the data packets in the feasible loading region (up to 4 Kbps) is less than 0.5 sec. The increased delays are due to additional overhead of route discovery. Due to RREQ and RREP packets, the effective load that a node

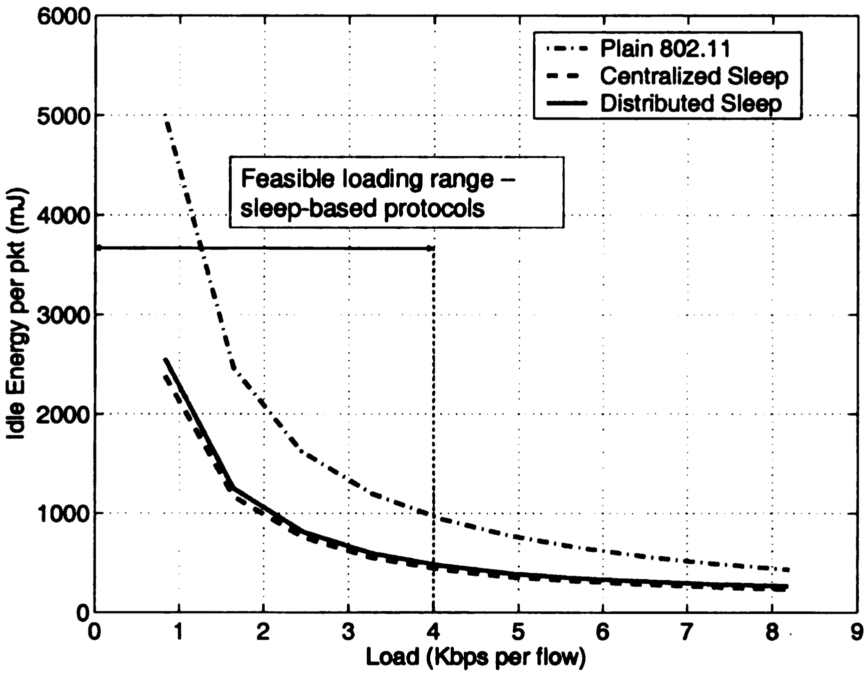


Figure 5.8: Idle Energy Consumption

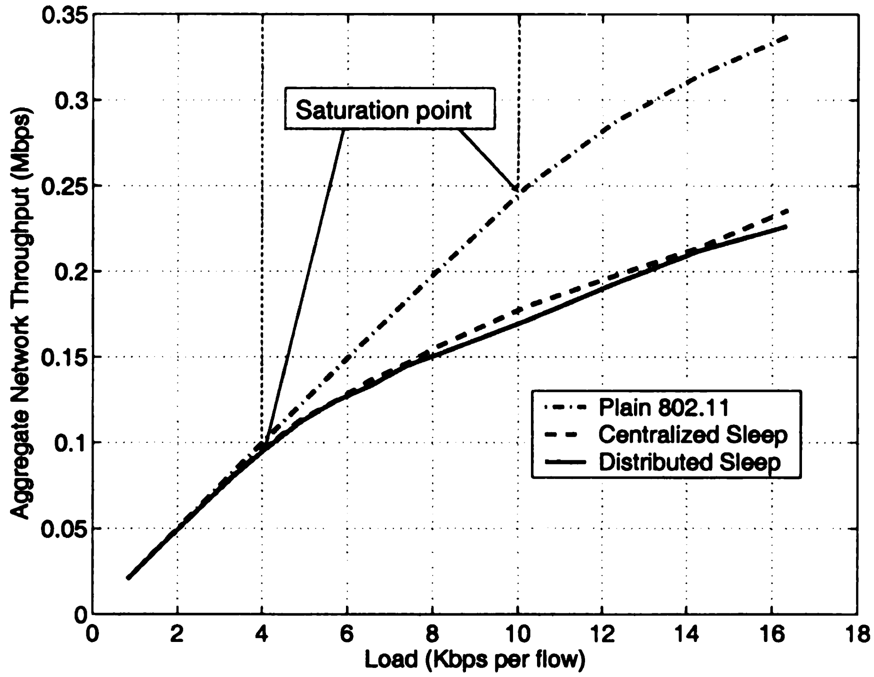


Figure 5.9: Sustainable Network Throughput

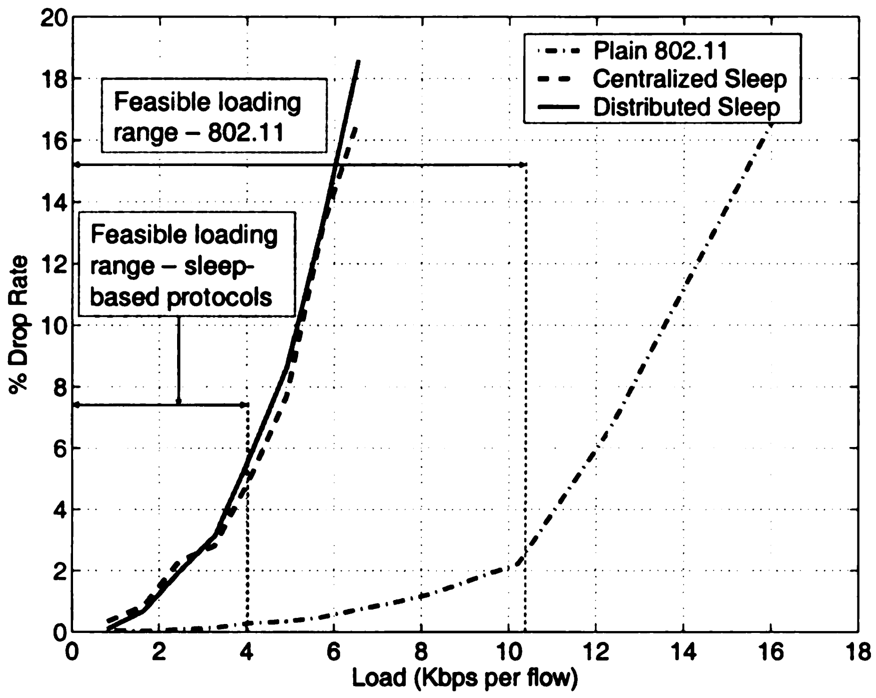


Figure 5.10: Packet Drop Rate

experiences in the wake cycle becomes high. Also, route discovery takes longer with the sleep mechanism. This increases the initial delay for packets when a node does not know of any routes to the destination. The delay can be reduced by using lower cycle times as discussed in the previous section. The average data delay is shown in Figure 5.11.

The measure of the control overhead of AODV packets can be seen from the increase in the transmission and reception energies as compared to the one-hop case. We see a considerable increase from 8 mJoules/pkt in the one-hop scenario to 25 mJoules/pkt in the multi-hop scenario. Even the reception energies in the multi-hop scenario are higher. The transmission and reception energies are capture in Figures 5.12 and 5.13, respectively.

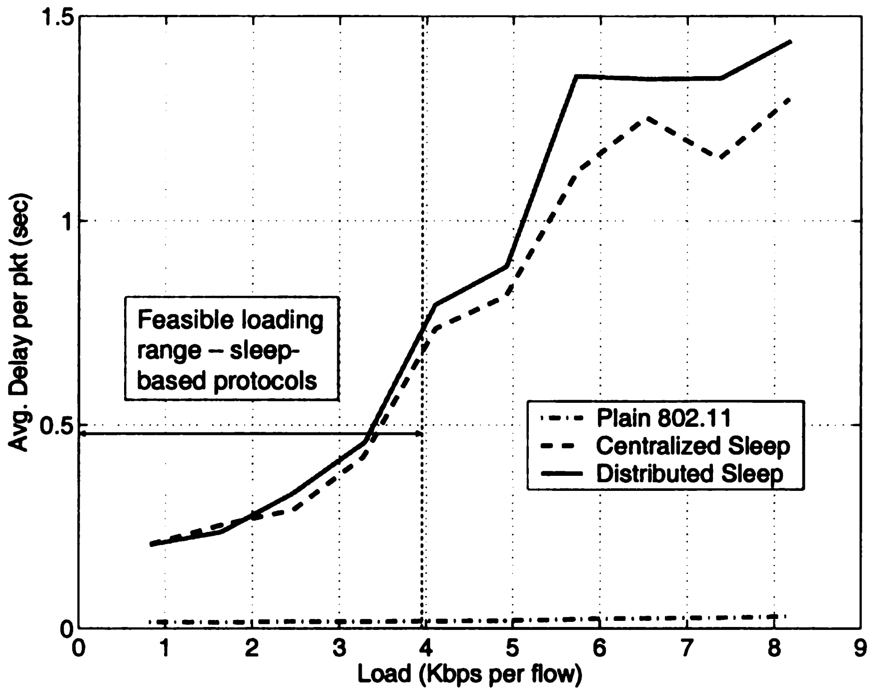


Figure 5.11: Average Data Delay

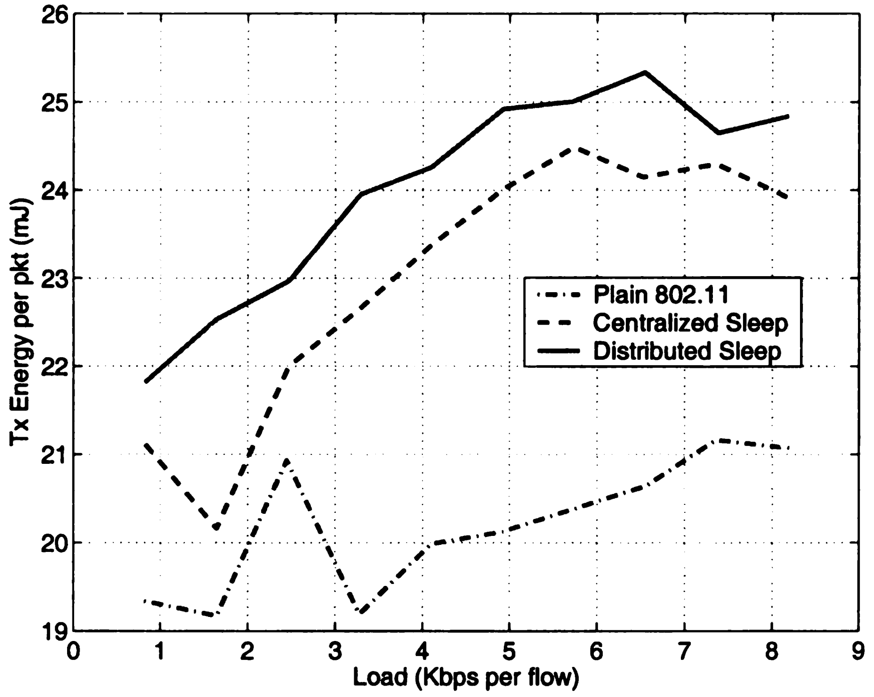


Figure 5.12: Tx Energy Consumption

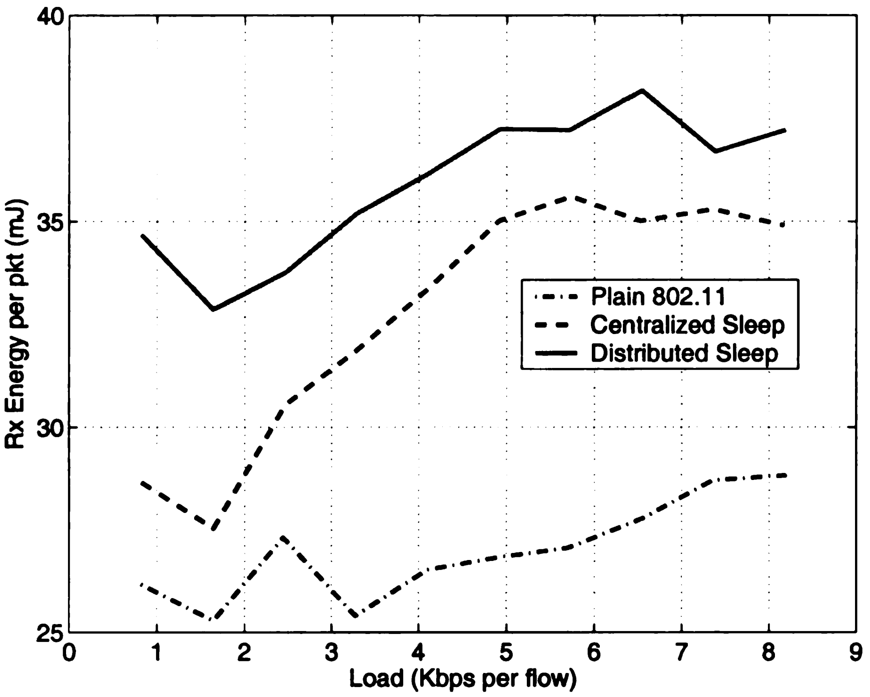


Figure 5.13: Rx Energy Consumption

5.2 Other Topologies

Experiments were carried out for the linear, circular and grid topologies. The results for these experiments can be found in Appendix B.

The linear topology consisted of 10 nodes at a distance of 200 m from each other. This scenario tests the performance of the protocol for systems which are not very dense. The results are shown in Figure B.1. The results show that we achieve around 50% energy gains. The feasible loading region with sleep is around 60 Kbps while it is 120 Kbps without sleep. The average delay delay with sleep is around 250 ms for a load of 60 Kbps. This delay is for a cycle period of 1 sec with 50% duty cycle. The delay can be reduced by reducing the the cycle period.

The circular topology consisted of 10 nodes each at a distance of 200 m from each other. The results are shown in Figure B.2. he feasible loading region with sleep is around 60 Kbps while it is 120 Kbps without sleep. The various metrics such as delay, throughput and energy savings are comparable to the linear topology.

For the grid topology, we construct a 5x5 grid consisting of 25 nodes. This topology is used to evaluate the performance of our protocol in a dense network. The results are shown in Figure B.3. Due to the dense topology, the feasible loading region for the sleep based protocols reduced to 18 Kbps while it was around 46 Kbps without sleep. Energy savings were still around 48%.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis, we have presented a distributed sleep scheduling protocol that can be used for implementing synchronous interface sleep for idling energy conservation in wireless ad hoc networks. A ns-2 based simulation model has been developed to evaluate and compare the protocol's performance with plain 802.11 and a centralized version of the sleep-wake protocol. Results from simulation experiments have indicated that the protocol is well suited for low to moderate network loading conditions under which the idling consumption is very large in 802.11 networks. Main conclusions from the thesis are four fold.

First, the maximum feasible loading point for sleep synchronization protocol can be determined by properly dimensioning the wake-sleep duty cycle. For a chosen duty cycle, the protocol is capable of achieving consistent savings in idling energy consumption within the feasible loading zone. Results in this paper shown that with 50% duty cycle, the savings in idle energy is around 49% throughout the entire feasible loading region.

The second conclusion is that although the packet drop rate worsens with synchronized sleep, the proposed protocol manages to limit the increase within tolerable limits for most data applications. For the presented results, the maximum packet drop rate increases from 2.5% for the plain 802.11 protocol to only 5% for the synchronized sleep mechanisms.

The third conclusion is that the packet delivery delay increases with higher wake-sleep cycle duration and to limit that, the cycle duration should be chosen at the smallest value that is allowed by the physical properties of the wireless interface cards.

The fourth and final conclusion is that the distributed version of the sleep mechanism performs nearly as well as the centralized version in which a centralized clock distribution is assumed for eliminating synchronization drifts due to propagation delay and clock-drifts.

6.2 Future Work

Future work on this topic includes providing protocol extensions for supporting node mobility and network partitioning in the presence of distributed wake-sleep activities. For handling network partitioning, a node can transmit a SYNC packet twice - once during the wake period and once during the sleep period. This will ensure that all schedules will get the SYNC packet. Also, each node can keep a neighbor table which contains the number of nodes that are following its schedule. It can send this information in the SYNC packet. Nodes can use this information to decide on which schedule to adopt.In mobile scenarios, we could transmit more SYNC packets at the expense of energy savings. This will increase the chances that SYNC packets are transmitted successfully.

We are also going to evaluate the effects of different MANET routing protocols on the performance of the proposed distributed sleep protocol. We have already seen the effect of AODV on our protocol. We see higher delays due to overhead of AODV control packets. We plan to evaluate the performance with other routing protocols such as DSR, DSDV and TORA.

APPENDICES

Appendix A

ns Extensions

To implement the distributed sleep-synchronization protocol and the centralized protocols, we added several features to ns [20], an event-driven network simulator with extensive support for simulation of wireless network protocols. Developed at the University of California at Berkeley and the Lawrence-Berkeley National Laboratories in collaboration with the VINT (Virtual InterNetwork Testbed) project, ns has a simulation engine written in C++ with a command and configuration interface using OTcl. Network topologies can be easily described using the primitives Nodes, Links, Agents, and Applications, where Nodes represent end-hosts in the network, Links are the connectors through which Nodes communicate, Agents are used to implement different network protocols and are the points where packets are created and consumed, and Applications are used to generate data and perform different application-specific functions. Once the topology has been created, simulations can be run by starting the Applications on different nodes at various points in time.

While ns was developed as a simulator for wired networks, researchers at Carnegie Mellon University added extensive support for wireless networks. The CMU additions include mobile nodes, MAC protocols, and channel propagation models (that were described in Section 4.1. Figure A.1 shows the implementation of a mobile node. The

Application class is written using the Tcl front-end, while the other functions that make up the node are written using the C++ engine. The Application creates "data packets" that are sent to the Agent. The Agent performs the transport and network-layer functions of the protocol stack. The Agent sends packets of data to CMUTrace, which writes statistics about the packets to trace files. The packets are then sent to a Connector which passes them to the Link-Layer for data-link processing. After a small delay, the packets are sent from the Link-Layer to the Queue, where they are queued if there are packets ahead that have not yet been transmitted. Once a packet is removed from the Queue, it is sent to the MAC, where media access protocols are run. Finally, the packet is sent to the Network Interface, where the correct transmit power is added to the packet and it is sent through the Channel. The Channel sends a copy of the packet to each node connected to the channel. The packets are received by each node's Network Interface and then passed up through the MAC, Link-Layer, Connector, CMUTrace, and Agent functions. The Agent de-packetizes the data and sends notification of packet arrival to the Application.

A.1 SyncAgent

The *SyncAgent* in each node manages the distributed synchronization with its neighbors. It is implemented as a subclass of the Agent class. We have implemented two timers, a *SyncTimer* and a *SleepTimer*. The *SyncTimer* fires once every 'N' cycles to check if the node has received any SYNC packets. Depending on this it adjusts its schedule. It then broadcasts its schedule to the neighboring nodes. The *SleepTimer* fires at the beginning of the sleep and wake periods. The *SyncAgent* has a pointer to the network interface. During the sleep period, its turns OFF the network interface so that no packets are received. At the beginning of the wake period, it turns the network interface ON When applications generate data, it is stored in a

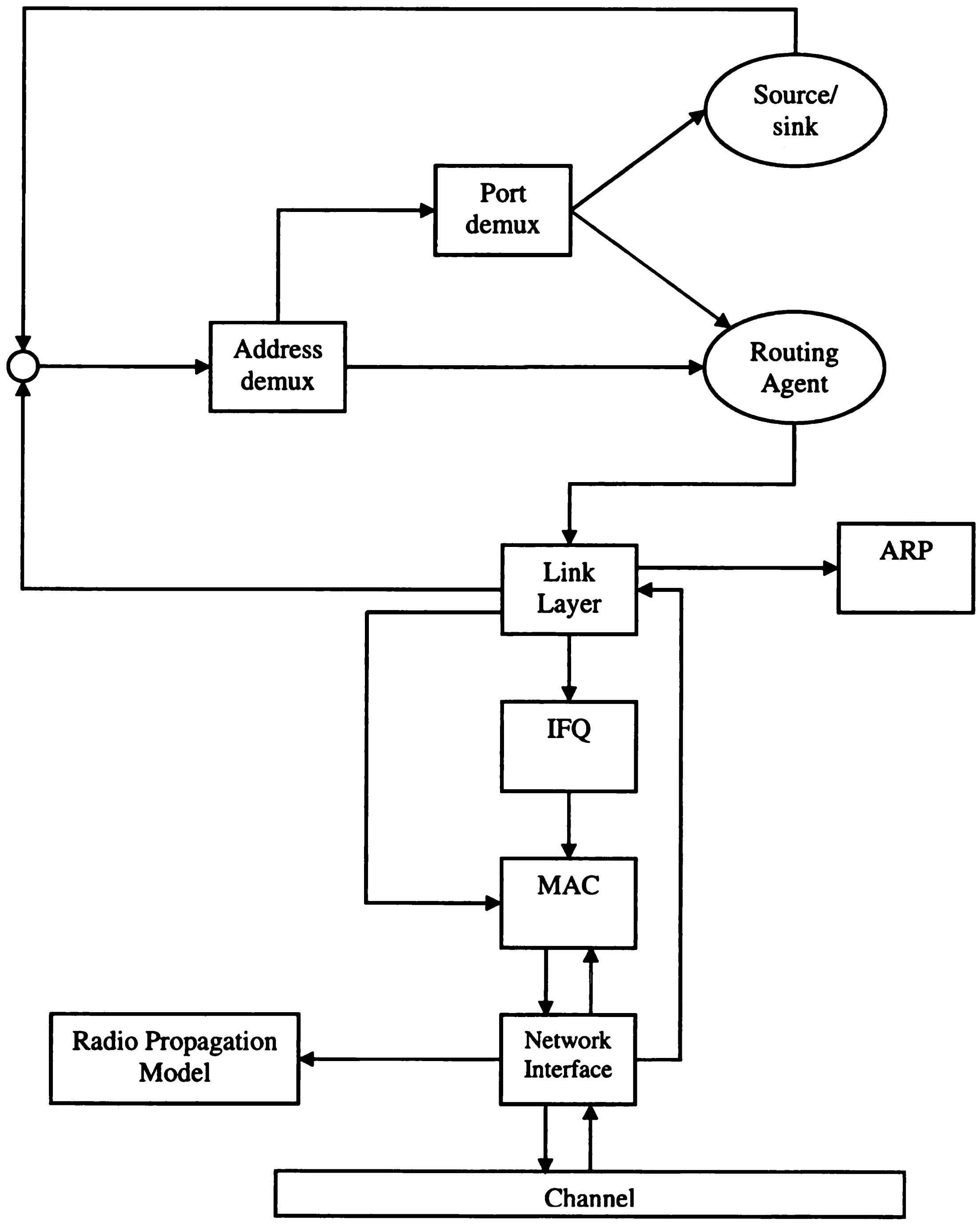


Figure A.1: Block Diagram of a ns2 Mobile Node

data buffer. During the wake period, the *SyncAgent* uses a novel algorithm to space the data packets and transmit them.

A.2 IF Queue

The IF Queue is an interface queue which is located between the link layer and the MAC. All packets coming down from the agent are stored temporarily in this queue. The MAC layer then takes the packets from this queue and sends them to the network interface. In order to reduce drift, we have given priority to SYNC packets. Hence the SYNC packets are always placed at the head of the IF queue.

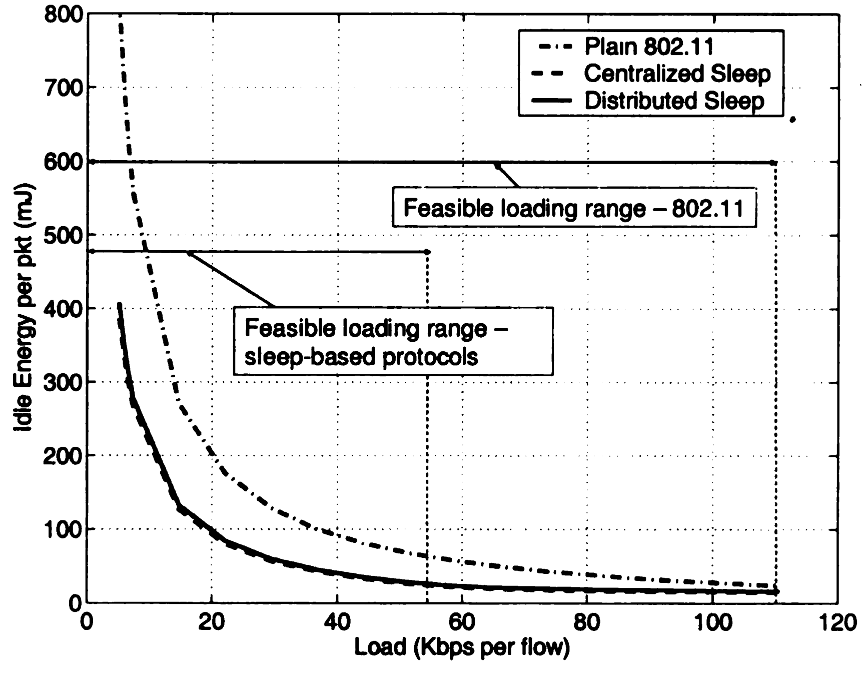
A.3 Network Interface

The Network Interphase layer serves as a hardware interface which is used by mobilenode to access the channel. This interface, subject to collisions and the radio propagation model, receives packets transmitted by other node interfaces to the channel. The interface stamps each transmitted packet with the meta-data related to the transmitting interface like the transmission power, wavelength etc. This meta-data, present in packet header, is used by the propagation model in the receiving network interface to determine if the packet has minimum power to be received and/or captured and/or detected (carrier sense) by the receiving node. We have implemented a sleep and wake state in the network interface. If an interface is in the sleep state, it drops all packets in either direction. The *SyncAgent* controls controls the sleep/wake state of the network interface.

Appendix B

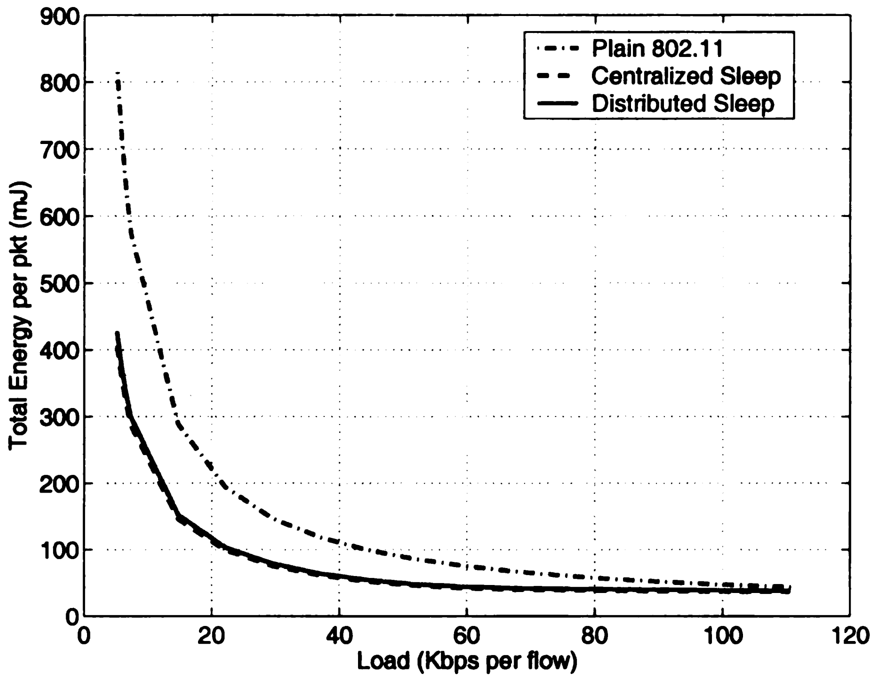
Results for Various Topologies

B.1 Linear Topology

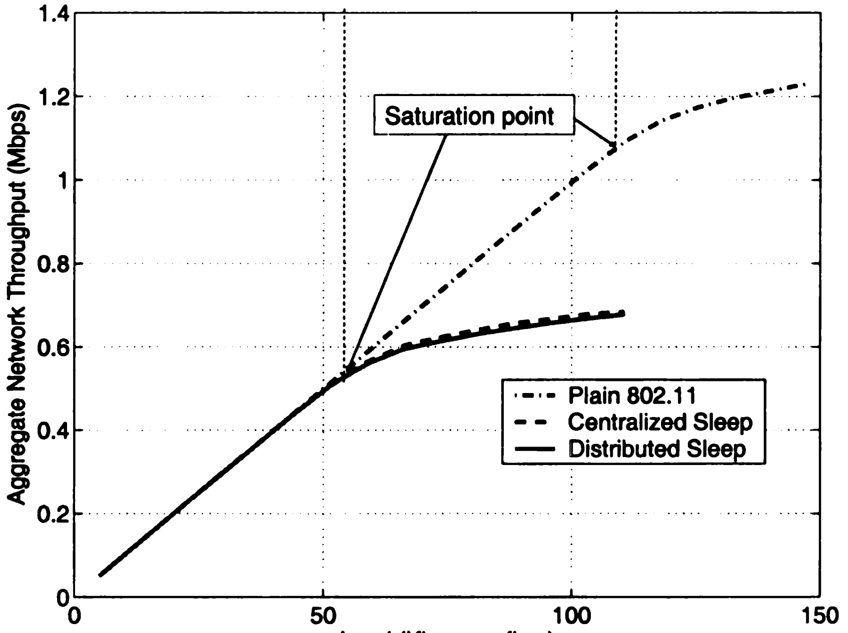


(a) Idle Energy Consumption

Figure B.1: Results for Linear Topology

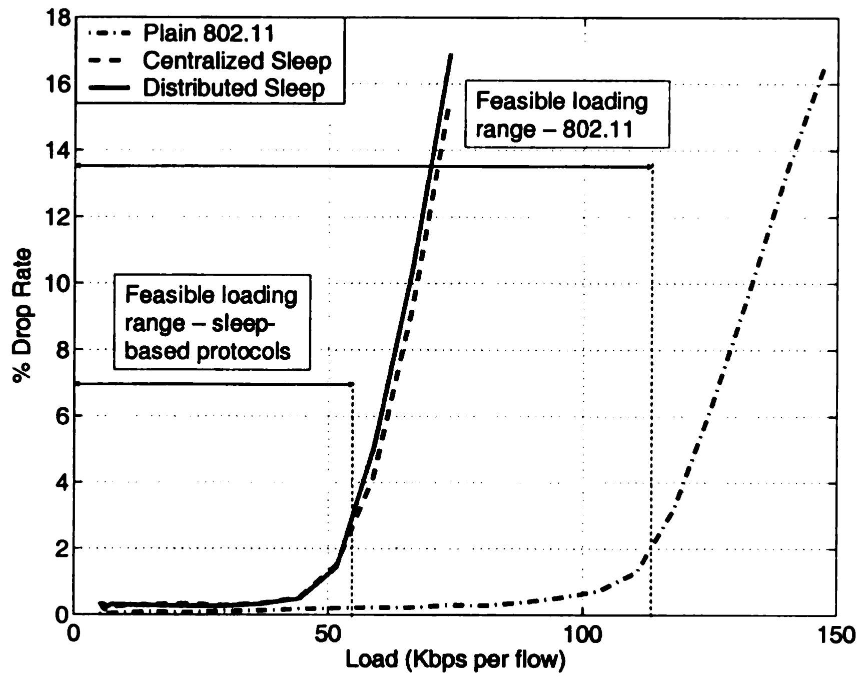


(b) Total Energy Consumption

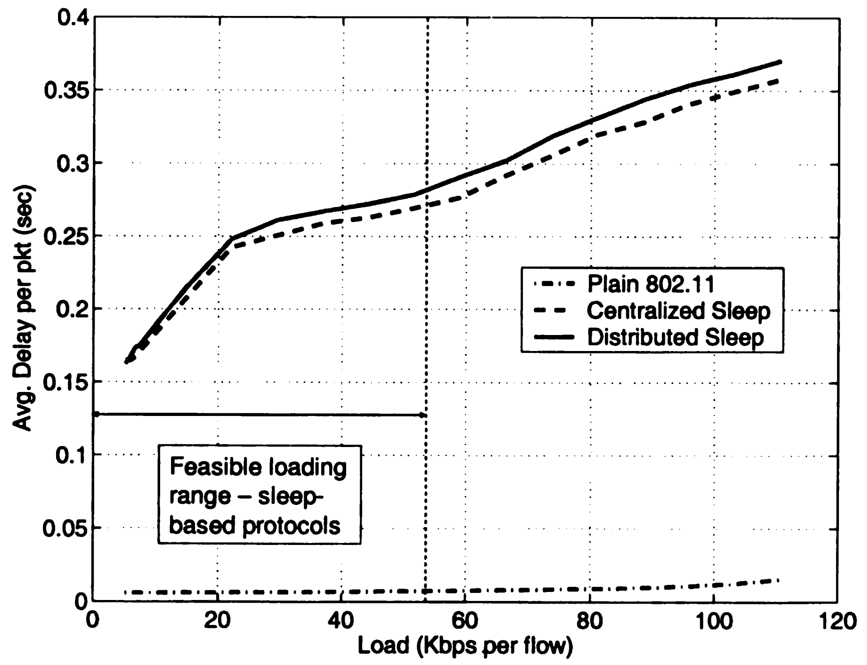


(c) Sustainable Network Throughput

Figure B.1: Results for Linear Topology (con't)

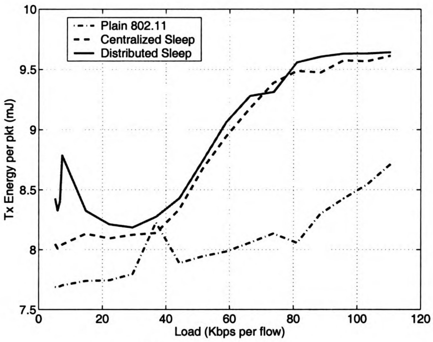


(d) Drop Rate

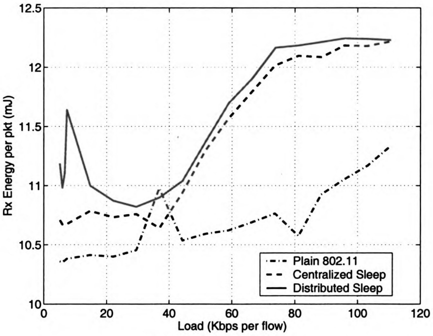


(e) Average Data Delay

Figure B.1: Results for Linear Topology (con't)



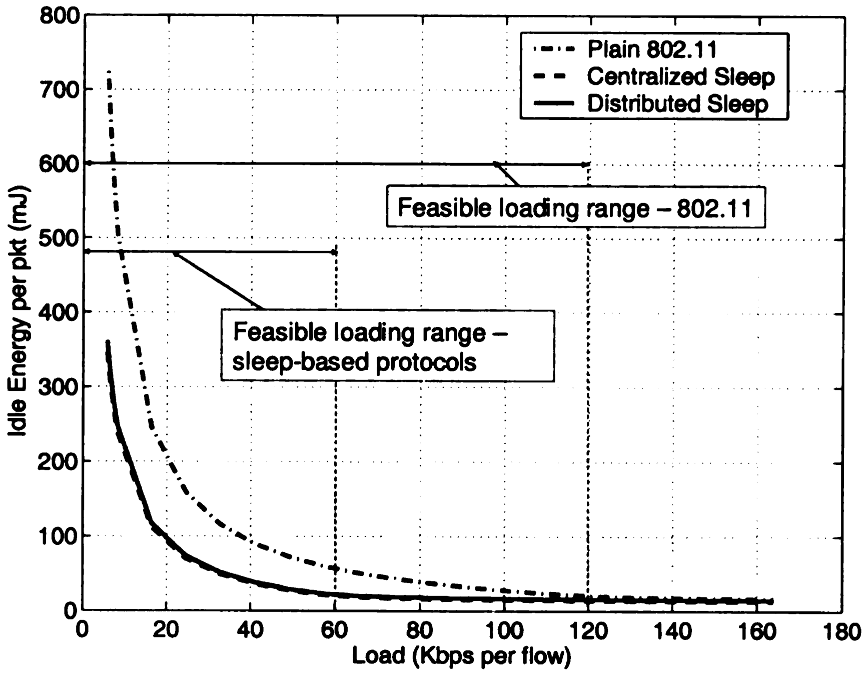
(f) Tx Energy Consumption



(g) Rx Energy Consumption

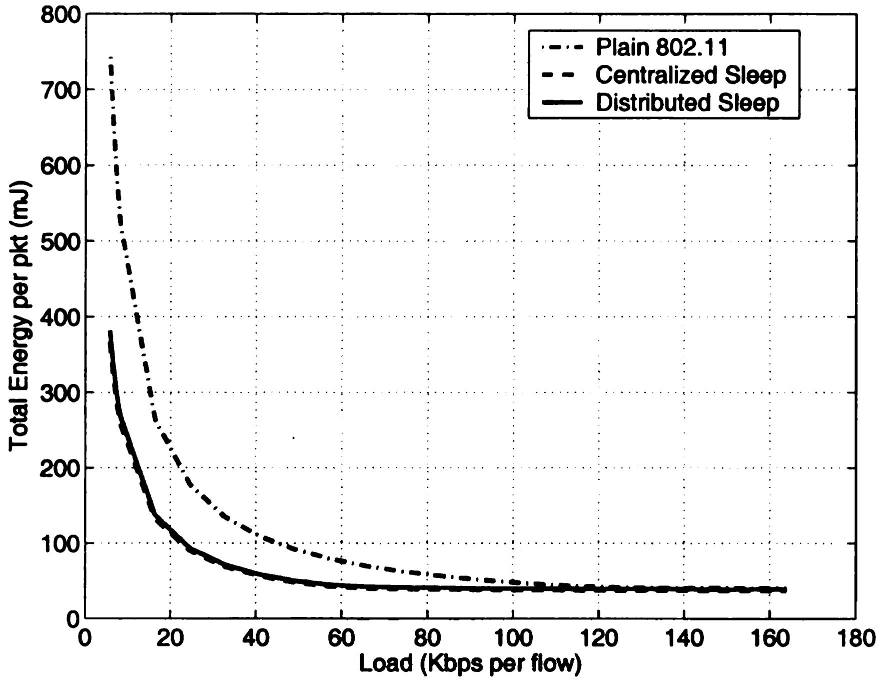
Figure B.1: Results for Linear Topology (con't)

B.2 Circular Topology

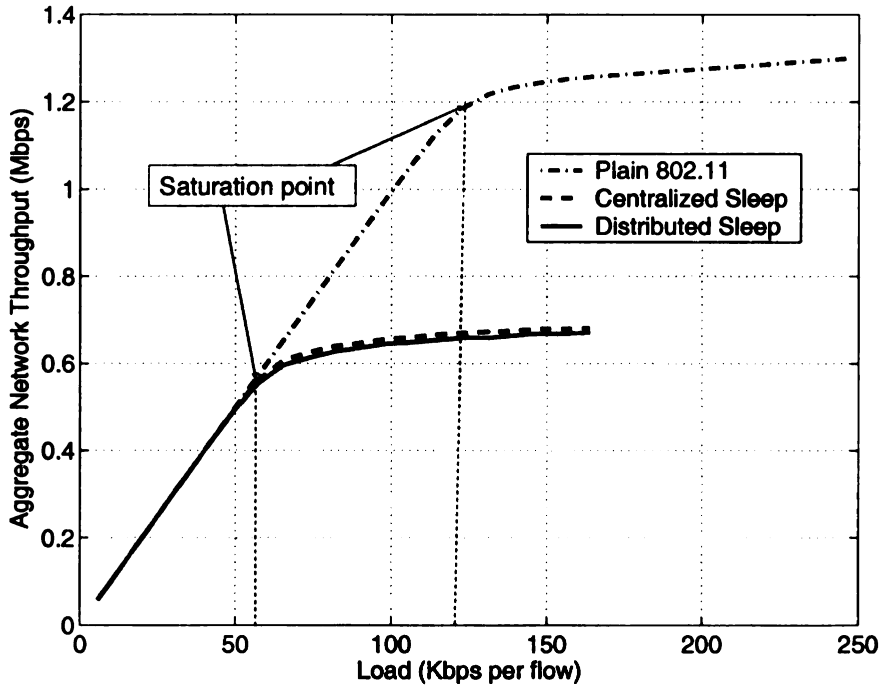


(a) Idle Energy Consumption

Figure B.2: Results for Circular Topology

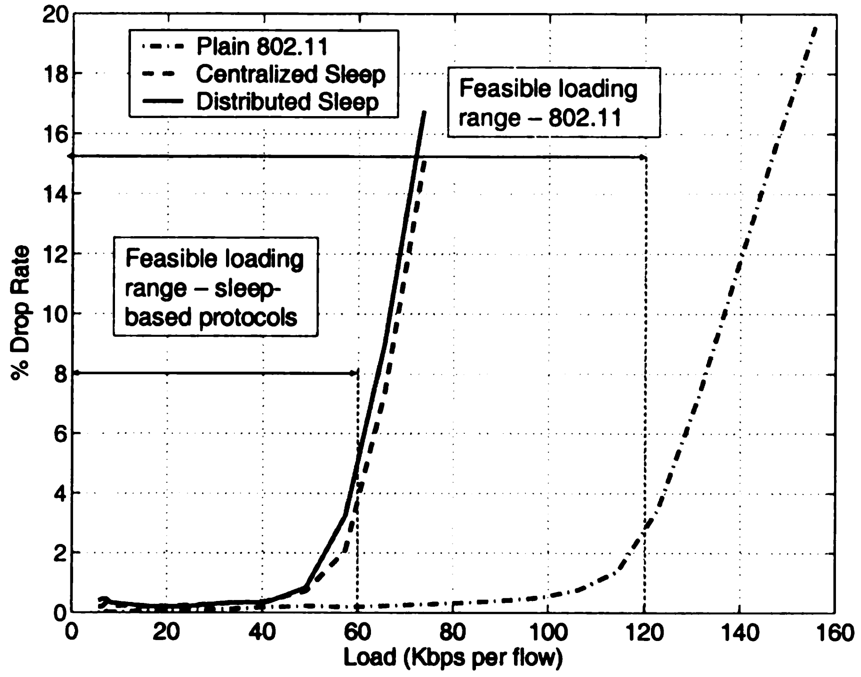


(b) Total Energy Consumption

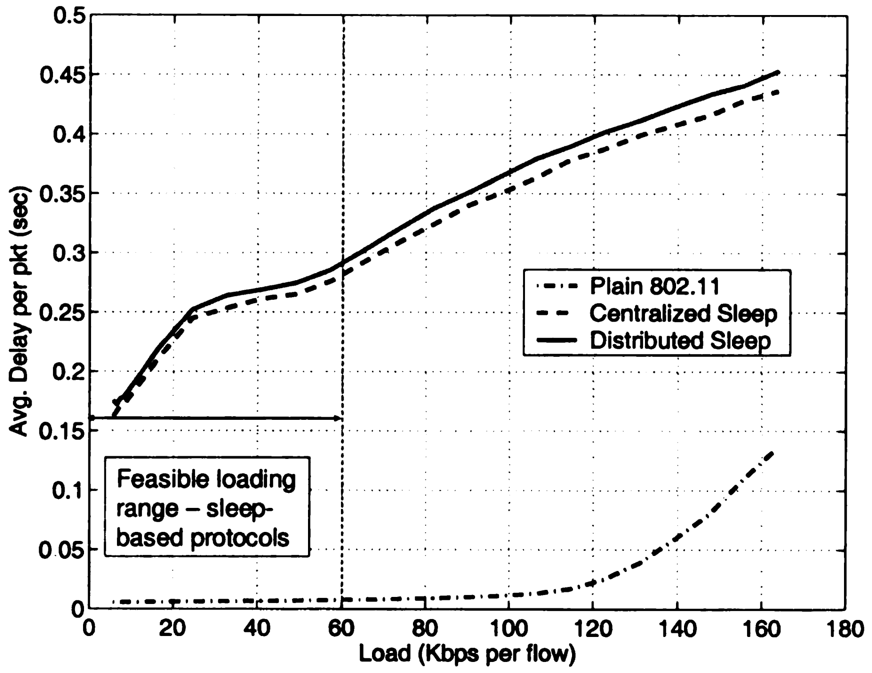


(c) Sustainable Network Throughput

Figure B.2: Results for Circular Topology (con't)

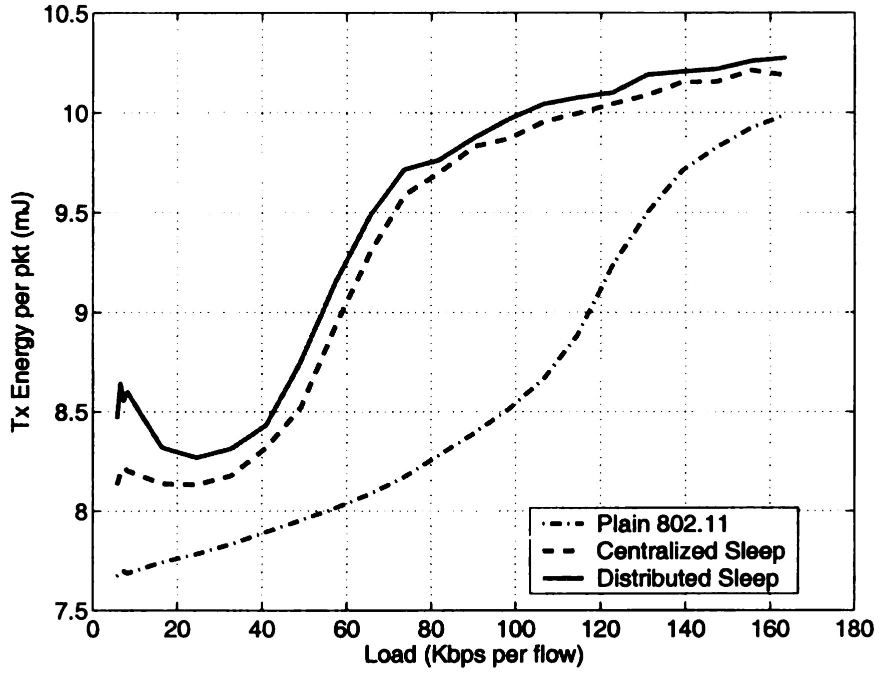


(d) Drop Rate

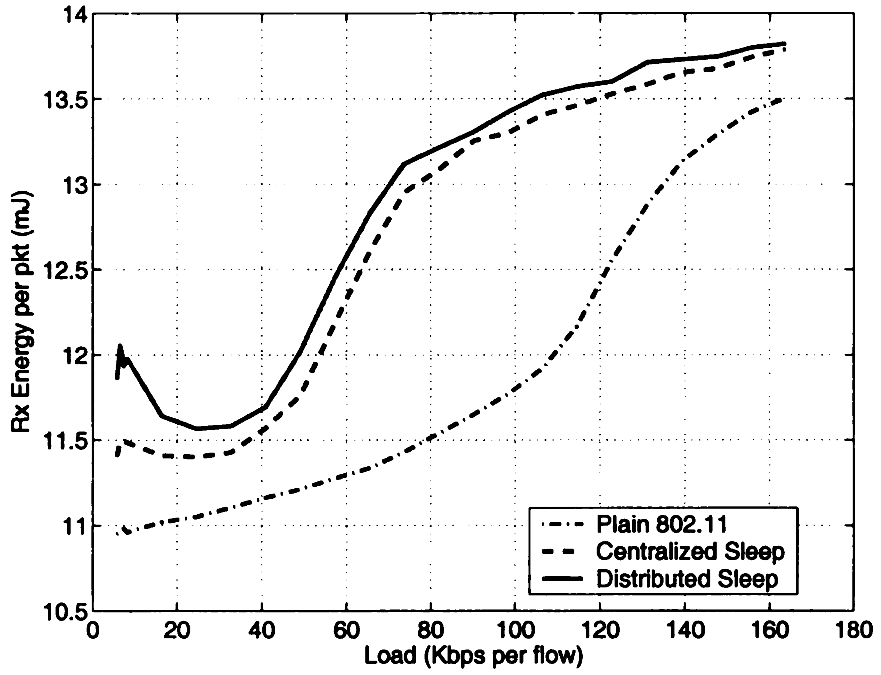


(e) Average Data Delay

Figure B.2: Results for Circular Topology (con't)



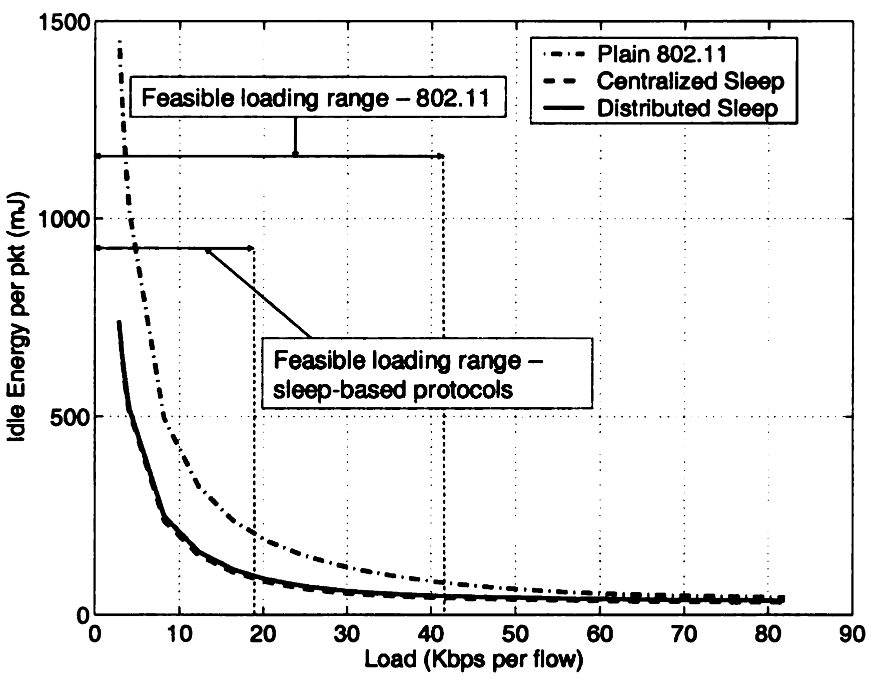
(f) Tx Energy Consumption



(g) Rx Energy Consumption

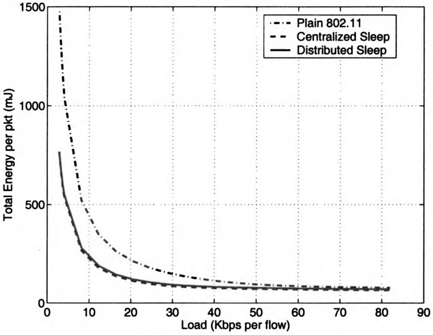
Figure B.2: Results for Circular Topology (con't)

B.3 Grid Topology

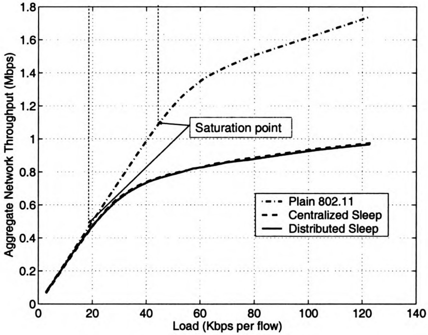


(a) Idle Energy Consumption

Figure B.3: Results for Grid Topology

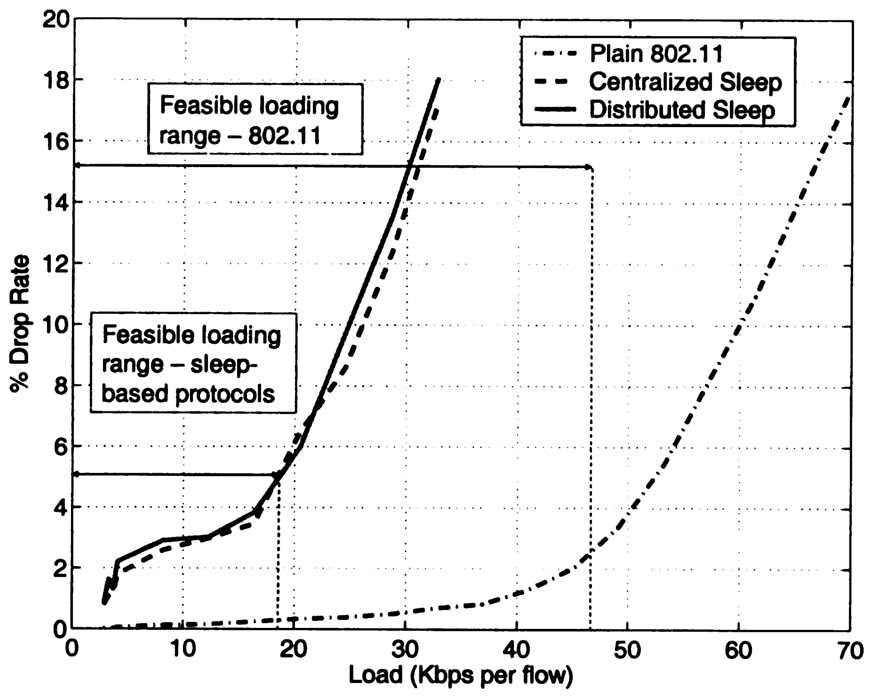


(b) Total Energy Consumption

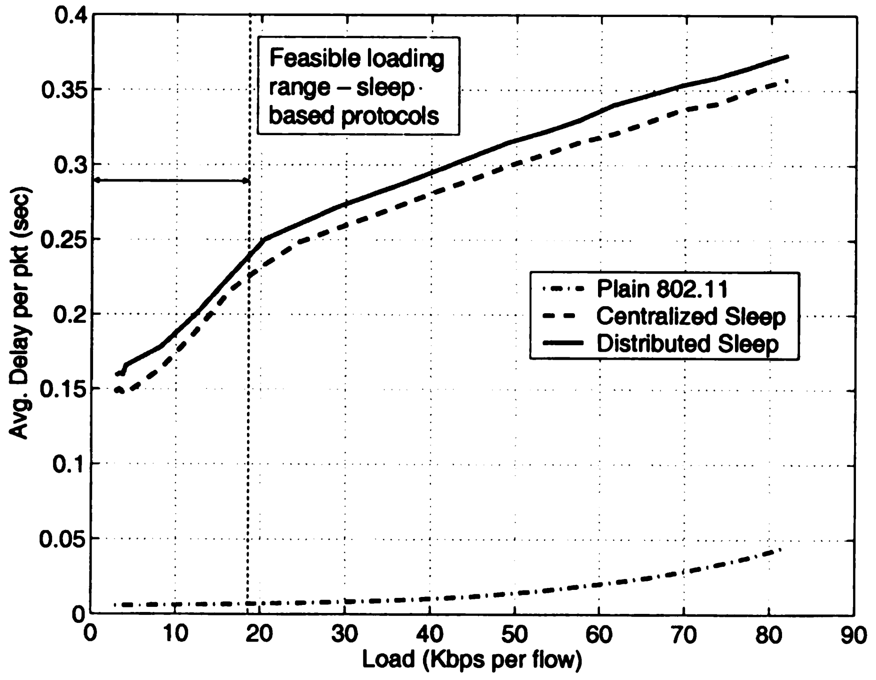


(c) Sustainable Network Throughput

Figure B.3: Results for Grid Topology (con't)

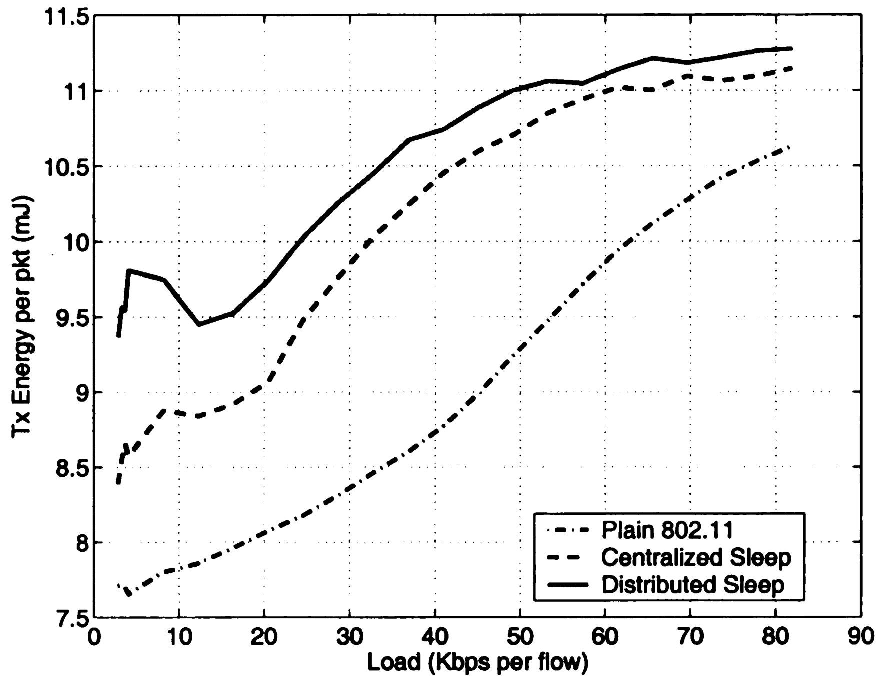


(d) Drop Rate

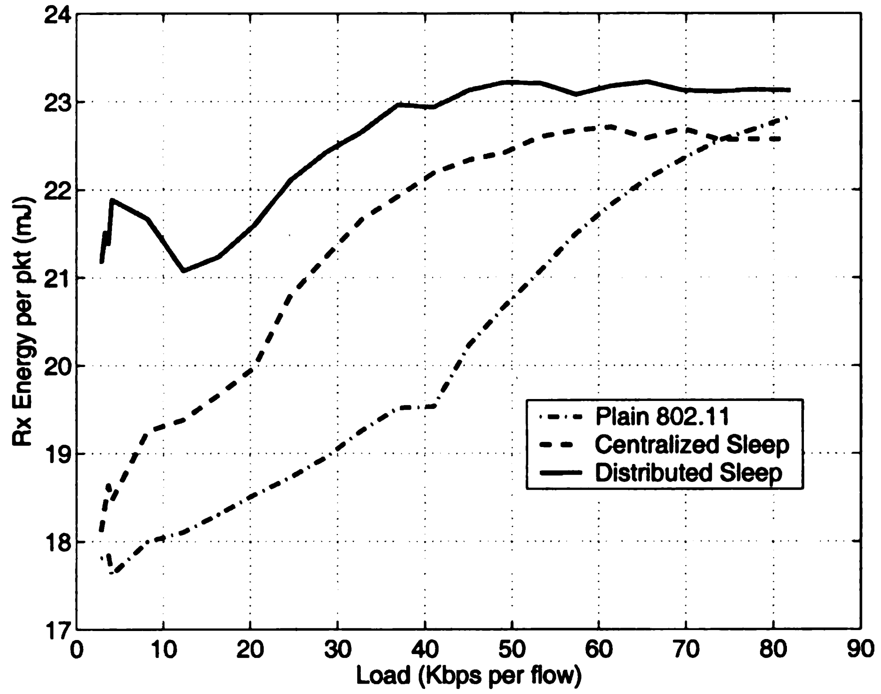


(e) Average Data Delay

Figure B.3: Results for Grid Topology (con't)



(f) Tx Energy Consumption



(g) Rx Energy Consumption

Figure B.3: Results for Grid Topology (con't)

BIBLIOGRAPHY

[1] M. Stemm and R. Katz. Measuring and reducing energy consumption of network interfaces in hand-held devices. *IEICE Transactions on Communications*, E80-B(8):1125–1131, August 1997.

[2] Oliver Kasten. *Energy Consumption.* URL: `http://www2.inf.ethz.ch/~kasten/research/bathtub/energy_consumption.html`.

[3] K. Dantu M. Maleki and M. Pedram. Power-aware source routing protocol for mobile ad hoc networks. In *Proceedings of ISLPED'02*, Monterey, California, 2002.

[4] M.Woo S. Singh and C.S. Raghavendra. Power-aware routing in mobile ad hoc networks. In *Proceedings of ACM/IEEE*, Dallas, Texas, 1998.

[5] John Heidemann Ya Xu and Deborah Estrin. Adaptive energy-conserving routing for multihop ad hoc networks. Research Report 527, USC/Information Sciences Institute, October 2000.

[6] N. Gupta and S. Das. Energy-aware on-demand routing for mobile ad-hoc networks. In *Proceedings of IWDC*, Calcutta, India, 2002.

[7] C. Toh. Maximum battery life routing to support ubiquitous mobile computing in wireless ad hoc networks. IEEE Communications Magazine, June 2001.

[8] R. S. Sreenivas S. Narayanaswamy, V. Kawadia and P. R. Kumar. Power control in ad hoc networks : Theory, architecture, algorithm and implementation of the compow protocol. In *European Wireless Conference*, 2002.

[9] J. Heidemann W. Ye and D. Estrin. An energy-efficient mac protocol for wireless sensor networks. In *Proceedings of The 21st Annual Joint Conference of the IEEE Computer and Communications Societies.* INFOCOM, June 2002.

[10] S. Singh and C.S. Raghavendra. Pamas: Power aware multi-access protocol with signaling for ad hoc networks. *ACM Computer Communication Review.*

[11] Juan Carlos Cano and Pietro Manzoni. Evaluating the energy consumption reduction in a manet by dynamically switching-off network interfaces. In *Proceedings of the 6th IEEE Symposium on Computers and Communications*, Hammamet, Tunisia, 2001.

[12] R. Kannan R. Kalidindi, L. Ray and S. S. Iyengar. Distributed energy-aware mac protocol for wireless sensor networks. In *International Conference on Wireless Networks*, Las Vegas, Nevada, 2003.

[13] E. S. Jung and N. H. Vaidya. An energy efficient mac protocol for wireless lans. In *INFOCOM*, 2002.

[14] S. Biswas and S. Datta. Reducing overhearing energy in 802.11 networks by low-power interface idling. In *IEWCN*, Phoenix, Arizona, April 2004.

[15] E. S. Jung and Nitin Vaidya. A power control mac protocol for ad hoc networks. In *Proceedings of ACM International Conference on Mobile Computing and Networking.* MobiCom, September 2002.

[16] Alaa Muqattash and Marwan Krunz. A distributed transmission power control protocol for mobile ad hoc networks. IEEE Transactions on Mobile Computing, 2003.

[17] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, IT-46(2):388404, 2000.

[18] A. Chandrakasan W. R. Heinzelman and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the Hawaii International Conference on System Sciences*, 2000.

[19] David L. Mills. *Network Time Protocol (Version 3) Specification, Implementation and Analysis.* Network Working Group Request for Comments: 1305, 1992. URL: `http://www.ntp.org/`.

[20] K. Fall and K. Varadhan. *The ns Manual.* The VINT Project. UC Berkeley, LBL, USC/ISI, and Xerox PARC, December 2003. URL: `http://www.isi.edu/nsnam/ns/ns-documentation.html`.

[21] T. Rappaport. *Wireless Communications: Principles & Practice.* Prentice-Hall, Inc., New Jersey, 1996.

[22] H. T. Friis. A note on a simple transmission formula. In *Proceedings of IRE, 34*, 1946.

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

[24] P. Karn. Maca - a new channel access method for packet radio. In *Proceedings of 9th ARRL Computer Networking Conference*, 1990.

[25] E. Belding-Royer C. Perkins and S. Das. *Ad hoc On-Demand Distance Vector (AODV) Routing.* Network Working Group Request for Comments: 3561, July 2003. URL: `http://www.faqs.org/rfcs/rfc3561.html`.

MICHIGAN STATE UNIVERSITY LIBRARIES
3 1293 02504 4003