An Energy-Efficient Communication System

for

Ad hoc Wireless and Sensor Networks

by

William Nii Adjetey Tetteh

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

Dec 2007

[February 2007]

Author_____
Department of Electrical Engineering and Computer Science
Dec 19, 2007

Certified by____
r. Jason Redi
Division Scientist, BBN Technologies
Thesis Supervisor

Certified by_____
.er Lampson
, MIT CSAIL
Supervisor

Accepted by_____
ur C. Smith
Professor of Electrical Engineering
Chairman, Department Committee on Graduate Theses

An Energy-Efficient Communication System

for

Ad hoc Wireless and Sensor Networks

by

William Nii Adjetey Tetteh

Submitted to the
Department of Electrical Engineering and Computer Science

Dec 19, 2006

In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

Existing ad hoc wireless and sensor network systems often trade off energy against performance. As such, it is hard to find a single deployable system that supports high data rates while maintaining energy-efficient operation. This research addresses the problem by designing a communication system that achieves high performance and reduces the energy needed for delivering data in multi-hop networks by a factor of 100 or more over IEEE 802.11. The system is composed of a duty cycling and pseudo-random Medium Access Control (MAC) that provides deterministic access to the shared medium. Furthermore, the MAC provides link level *QOS* to support high data rates required for real-time traffic as well as delay-bounded services such as voice and multi-media streaming.

Thesis Supervisor: Dr. Jason Redi
Title: Division Scientist, BBN Technologies

Thesis Supervisor: Prof. Butler Lampson
Title: Adjunct Professor, MIT Computer Science and Artificial Intelligence Laboratory

Contents

1 List of Figures 4

2 List of Tables 5

3 Acknowledgements 6

4 Introduction 7

4.1 Problem Statement and Motivation 8

4.2 Summary of Main Contributions 9

4.3 Organization 10

5 Background and Related Work 11

5.1 Contention-based MAC Protocols 11

5.1.1 IEEE 802.11 11

5.2 TDMA based MAC Protocols 12

5.2.1 Directory Approaches 13

5.2.2 Grouped-TDMA Approaches 13

5.2.3 Pseudo-Random Approaches 14

5.3 Related Work 14

5.3.1 S-MAC 14

5.3.2 T-MAC 15

5.3.3 DMAC 15

5.3.4 SEEDEX 16

5.3.5 LMAC 17

5.3.6 TRAMA 17

5.4 Summary 18

6 System Architecture 20

7 PHY Component 22

7.1 Hail Radio 22

7.2 Data Radio 25

8 MAC Component 26

8.1 Mac Protocol 27

8.1.1 Generating Schedules 28

8.1.2 Unicast-Receive Schedule 31

8.1.3 Broadcast-Transmit Schedule 33
8.1.4 Unicast-Transmit Schedule 34
8.1.5 Scheduling and Slot Rules 40
8.2 Node Throughput 41
8.3 Synchronization 43
8.3.1 Time Synchronization 44
8.3.2 Slot Synchronization 45
9 System Design Details 53
9.1 Outgoing Over the Air (OTA) Data 53
9.2 Incoming OTA Data 54
9.3 Data Link Lower Layer 55
9.3.1 Detailed Description 55
9.3.2 Interfaces 56
9.4 Mac State 57
9.4.1 Detailed Description 57
9.4.2 Interfaces 60
9.5 Neighbor Discovery 60
9.5.1 Detailed Description 61
9.5.2 Interfaces 63
9.6 Link Characterization 64
9.6.1 Detailed Description 64
9.6.2 Interfaces 65
10 Discussion of Results 66
10.1 Synchronization Accuracy 67
10.2 Latency 69
10.3 Energy 72
11 Conclusions and Future Work 73
12 References 76
13 Appendix A Hail Transceiver Options Spec. Sheet 79
14 Appendix B Packet Formats 80

1 List of Figures

Figure 1: Network deployment scenario – sensors and soldier units. 8
Figure 2: System Architecture 21
Figure 3: Chipcom SmartRF® CC1010 transceiver. 23
Figure 4: Hail and Data radio frequency hopping within a slot. 25
Figure 5: CSMA/CA vs. a duty cycling MAC. 26
Figure 6: When to turn the Hail Radio "on". 32
Figure 7: Dynamic slot allocation using an adaptive controller 34
Figure 8: Dynamic slot allocation algorithm for w slots. Each slot, x, is assigned to at most one node within a 2-hop neighborhood. 36
Figure 9: Analysis of dynamic slot allocation algorithm. 37
Figure 10: The controller adapts PUT values towards the Efficiency Line when there is overload. 40
Figure 11: Node B aligns its slot boundary to Node A upon reception of a message from Node A. ξ is the slot boundary skew between A and B. 46
Figure 12: (a) Tree-based slot synchronization - the root node disseminates slot boundary information throughout the entire network (b) peer-to-peer slot synchronization- neighboring nodes apply a common filter function to converge at a common slot boundary. 47
Figure 13: p_2 aligns its slot to p_1 without coordinating with p_3 thus increasing the slot boundary skew between p_2 and p_3. 52
Figure 14: Low-level MAC protocol and slot timing (50 ms slot size) 59
Figure 15: (a) Experimental setup and network topology (b) Histogram of the absolute slot boundary skew between neighboring nodes before synchronization. 66
Figure 16: Histograms of absolute slot boundary skew between neighboring nodes in tree-based and peer-to-peer slot synchronization. 67
Figure 17: (a) Average end-to-end latency experienced by application packets within the network for increasing hop distance and increasing unicast receive probability threshold (b) Histogram of end-to-end latency over a single hop. The majority of application packets arrive in less than 500ms if the Hail radio listens 60% of the time 69
Figure 18: Adapting PUR values along a multi-hop path from sender to receiver. 70

Figure 19: Adapting PUR improves latency – only the first few packets experience significant delay. 70
Figure 20: Histogram of packet latencies 70
Figure 21: Energy and time for each radio state ($2x10^{-3}$ duty cycle). Data Idle includes energy consumed by the transceiver clock 71
Figure 22: Energy-efficiency per node 72
Figure 23: Daily energy usage per node 72
Figure 24: Overall specification comparison of Micrel and Chipcon single-chip transceivers 79
Figure 25: Hail Packet Format. 80
Figure 26: Hail Ack Packet Format 80
Figure 27: Data Packet Header Format. 81
Figure 28: Data Ack Packet Format. 81

2 List of Tables

Table 1: Tradeoff Matrix for Slot Synchronization Schemes 50

3 Acknowledgements

I would like thank my advisors, Dr. Jason Redi and Prof. Butler Lampson, for their guidance and support throughout this research. I am truly honored to be mentored by two of the brightest minds alive.

I would also like to express my gratitude to Jim Bertone, my manager at BBN Technologies, for giving me the opportunity to pursue my educational goals while remaining gainfully employed.

Special thanks to all members of the MNS JAVeLEN Phase I and Phase II teams. Your hard work formed the foundation of this research and I am highly indebted to every one of you for your contributions.

Lastly, I would like to dedicate this thesis to my loving parents and my sisters, Adjeley and Adjorkor. Thank you for your patience and support during this period.

4 Introduction

An ad hoc network is a collection of devices equipped with a radio and some networking capability that enables communication between these devices. Ad hoc network systems exist in several forms varying from size, computational power, and battery capacity. The key to ad hoc networks is that devices that are otherwise out of radio range can still communicate via relaying data through nearby devices. In addition, ad hoc networks can support node mobility and a wide range of application traffic. The military relies on ad hoc networks for some of its communication needs because ad hoc network systems are quickly deployable without any dependence on an existing infrastructure.

One common application of ad hoc networks is wireless sensor networking. Networked sensors often collaborate to perform specific tasks such as battlefield surveillance and environmental monitoring. In some cases, each sensor independently collects data that it sends to a central location for post processing. Most military applications have a fundamental reliance on the delivery of audio and video feeds across the network. Thus, in addition to data fusion and post processing, sensor systems that are both energy-efficient and support high data rates can play active roles on the battlefield by delivering low latency application data. Existing sensor systems, however, are less than ideal for military applications because they waste too much energy that limits their lifetime or they offer very low data rates barely usable for the delivery of audio and video.

Since ad hoc network systems typically rely on batteries as their primary energy source, it is important that the entire system operate efficiently to maximize battery lifetime. In addition, the heterogeneous nature of these ad hoc systems affects the performance and design of Medium Access Control (MAC) protocols that allows these nodes to communicate efficiently. A MAC protocol essentially attempts to mediate access to the shared wireless medium by preventing interfering nodes from transmitting at the same time. Multiple nodes transmitting at the same time result in collisions at the receiver that wastes energy.

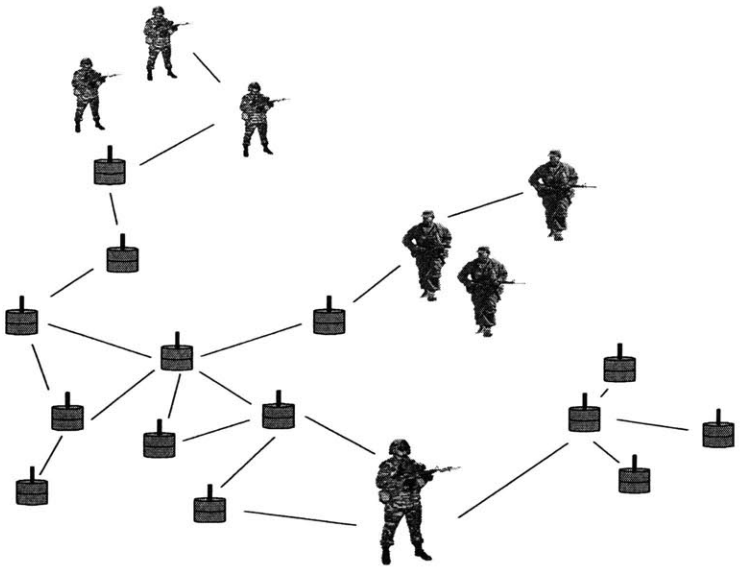


Figure 1: Network deployment scenario – sensors and soldier units.

Besides collisions, a major source of inefficiency a MAC protocol has to contend with is idle listening. Wireless LAN (WLAN) radio receivers usually use around 860mW when idling in receive mode. Idle listening consumes 50-100% of the energy required for receiving in WLAN radios [29] and it occurs because nodes listen for traffic all of the time even when there is no data to be received. Assuming a 1Mbps wireless channel with an average network load of 5 messages an hour and an average message size of 5000 bits, a duty cycling MAC protocol that is only 25% accurate (i.e. keeps the receiver powered on 4 times more often than it needs to) will consume 36000X ($10^{4.5}$) less energy over a MAC protocol that keeps the receiver powered on all of the time. Clearly, it is possible to achieve significant energy savings by intelligently powering down the receiver when there is no data to be received.

4.1 Problem Statement and Motivation

Besides performance, soldiers on mission have two main concerns about their radios, namely, how much the units weigh and how long the batteries will last. These concerns apply to all portable radio and battlefield communications devices ranging from data-collecting sensors in remote locations to individual soldier units (Figure 1). In military surveillance where the data rates required are much higher, a network of video enabled

sensors should stream video as quickly as possible when they detect changes in the environment so that the soldier can react. Similarly, when there is relative inactivity, the sensors should conserve as much energy as possible. Unfortunately, existing ad hoc network systems often trade off energy against performance. As such, it is hard to find a single deployable system that achieves high data rates while maintaining energy-efficient operation. The goal of this research is to address this problem by designing an ad hoc network system that achieves both high performance and low energy usage. Critical to the successful operation of such a system is a Medium Access Control (MAC). This thesis describes a MAC protocol that is energy-efficient under most traffic loads and is able to adapt as battlefield conditions change. An energy-efficient MAC protocol increases device lifetime and it enables devices to transmit important information for longer periods.

4.2 Summary of Main Contributions

To make the entire system energy-efficient, this research focuses on total system wide energy gains in hardware and software rather than trying to optimize specific metrics. For example, in low duty cycle environments, focusing on a low power transmitter will not yield the maximum energy savings. A better solution is to turn the receiver "off" when it is not in use. The summary of contributions for each of the two main components of the system is as follows:

PHY Component:

- Split the PHY component into two radios - a Hail radio and a Data radio. The Hail radio is a low power and low data rate radio while the Data radio is a high power and high data rate radio. Both radios are always "off" by default and the MAC determines when and whether a particular radio should be "on" or not.

MAC Component:

- Disseminate pseudo-random seeds and probability thresholds to generate uncorrelated wakeup schedules.

- Instead of using preset parameters, it is better to learn and adapt. The MAC adapts to traffic patterns and other network events by dynamically adjusting transmission and reception schedules.
- Maintain accurate synchronization between nodes using novel energy-efficient slot synchronization techniques with minimal message overhead.
- Support collision free access and high channel utilization in high duty cycle environments.
- Increase information sharing between subsystems. Protocols can make intelligent inferences from reported PHY component statistics such as Received Signal Strength Indication (RSSI).

4.3 Organization

This thesis presents an energy-efficient system for wireless ad hoc and sensor networks. The system achieves energy-efficiency by interoperating algorithms and protocols between the PHY and MAC components. The rest of the thesis is organized as follows. Chapter 5 presents some background information and related work relevant to the design of energy-efficient MAC protocols in general. Chapter 6 presents the system architecture and highlights key subsystems that interact closely to achieve energy-efficient operation of the entire system. Chapter 7 describes the PHY component with emphasis on the functioning of the Hail radio. Chapter 8 introduces the MAC component and discusses the operation of an energy-efficient pseudo-random MAC protocol that adapts to application throughput requirements. Closely tied to the successful operation of the MAC protocol is synchronization. Chapter 8 also presents new energy-efficient techniques used by the MAC component to achieve accurate synchronization between nodes. Chapter 9 discusses system design, implementation details, and interfaces between MAC subsystems. Chapter 10 discusses and analyzes the performance results obtained from a 20-node multi-hop network that implements the MAC and PHY components discussed in this report. Chapter 11 concludes this report by summarizing the work done in this research and additional topics for future work.

5 Background and Related Work

The goal of every MAC protocol is to ensure that nodes sharing a common physical channel do not interfere with each other's transmissions. Interfering transmissions result in a collision at the receiver; thus wasting energy and reducing channel utilization. In general, MAC protocols are classified as either contention-based or TDMA based. The next section briefly describes contention-based and TDMA based MAC protocols.

5.1 Contention-based MAC Protocols

In contention-based protocols, a node wishing to transmit at any time must contend for the channel prior to transmission. Contention-based protocols are therefore inefficient due to the following reasons:

- Idle-listening – Since nodes do not know in advance whom to expect a packet from, they have no choice but to keep their receivers "on" all of the time. In low duty cycle networks, idle-listening dominates the total energy consumed.
- Overhearing – This occurs when a node receives a packet intended for another node. The energy used by the receiver to decode the packet is wasted.
- Collisions – Collisions occur when the transmission of two nodes interfere with each other. Retransmission of collided packets requires additional energy.
- Control message overhead – To reduce collisions, contention-based protocols use control message exchanges to reserve the channel for transmission. These control messages increases the amount of energy used per application bit.

The next section briefly describes the IEEE 802.11 protocol [30]. IEEE 802.11 is a contention-based MAC protocol that is often the baseline for measuring energy-efficiency.

5.1.1 IEEE 802.11

The IEEE 802.11 protocol uses Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) with positive acknowledgement. A node wishing to transmit first senses the

channel. If the channel is busy, the node defers transmission to a later time. On the other hand, if the channel is free for a specified period called the Distributed Inter Frame Space (DIFS), the node can transmit. The receiving node on receiving a valid packet sends an acknowledgement (Ack) to the sender to indicate that the transmission was successful. If the sender does not receive an Ack, it retries a specified number of times until it receives an Ack. CSMA/CA/Ack does not eliminate all collisions and it is possible for two nodes to collide because they cannot hear each other (hidden terminal problem). To avoid this problem, a node wishing to transmit a packet first sends a Request to Send (RTS) control packet that includes the source, destination, and duration of the transmission. The destination responds, if the channel is free, with a Clear to Send (CTS) control packet that repeats the duration specified in the RTS. All nodes overhearing the RTS/CTS or both will defer from accessing the channel for the specified duration. The RTS/CTS scheme only applies to unicast packets so broadcast packets are susceptible to collisions.

5.2 TDMA based MAC Protocols

Time Division Multiple Access (TDMA) divides the channel into slots. A node only transmits in a reserved slot. Consequently, TDMA is the natural choice for energy conserving MAC protocols because there is no idle listening, no collisions, no overhearing, and it does not require additional message overhead to access the shared medium. To reap these benefits of TDMA in duty cycling networks requires synchronization of schedules between nodes. Synchronized schedules ensure that nodes do not waste energy transmitting to other nodes that do not have their receivers "on". A number of TDMA based MAC protocols have been developed and evaluated for reducing the amount time a receiver must be "on" [8]. These protocols allow a dramatic reduction of energy without substantially increasing network latency for a variety of traffic loads. However, all of these MAC protocols assume an infrastructure-based wireless network framework where nodes can easily synchronize to a central base station that has infinite energy. Even though these protocols and their underlining assumptions are not applicable to wireless ad hoc and sensor networks, they nonetheless provide a hint as to what performance improvements are available. The next section describes the three main

categories of energy-conserving TDMA based MAC protocols defined in [6] that seek to trade-off energy consumption with latency.

5.2.1 Directory Approaches

The first category of energy conserving TDMA schemes utilizes a periodically transmitted broadcast directory. Most commercial wide-area wireless services use a direct variation of this scheme. The general idea with this approach is that a base station node periodically broadcasts a directory listing of the packets currently waiting in its queue. The transmission of the actual packets immediately follows the transmission of the directory listing. Nodes periodically turn on their receivers to listen to the directory and to figure out when the next set of packets will arrive from the base station. On receiving the directory, a node schedules itself to listen for the packets according to the times stated in the directory. This method is extremely robust in the presence of highly clustered traffic since the directory listing states explicit times for packet transmission. However, due to the periodic broadcast of the directory, this approach can result in either large expenditures of energy when there is low traffic volume, or a high average delay for a reduction in the frequency of directory broadcasts.

5.2.2 Grouped-TDMA Approaches

The second category of energy conservation schemes is a variation of the classic TDMA approach. Classic TDMA provides delay guarantees by assigning each node a unique transmission slot. If the energy consumption of a node is the average number of slots in which a node gets to transmit, it is easy to see that each node has an energy consumption of $1/N$, where N is the number of nodes. In large networks, classic TDMA has unacceptable high delay bounds since each node is able to transmit once in every N slots. Classic TDMA can be adapted for energy conservation by partitioning the nodes into N/x disjoint groups, where x is the group size. Grouping the nodes increases the average energy used per node to $1/(N/x)$, but the delay bound improves because the number of groups bounds how often a node gets to transmit.

5.2.3 Pseudo-Random Approaches

The pseudo-random TDMA scheme is a new class of randomized protocols based on deterministic schedules [25]. This approach exploits the power of randomization for fairness, while providing the advantages of determinism that allows a node to predict every other node's state in each slot. In this class of protocols, all nodes run the same pseudo-random number generator and determine the state of their neighbors in each slot, based on a probability parameter p_i, which could be different for each node i depending upon the expected traffic or energy conservation requirements for that node. To avoid a complete overlap of node schedules, each node initializes its pseudo-random generator using a unique seed. A node knows the seed of every other node in the network, thus enabling it to determine the schedule of its neighbors. The pseudo-random approach achieves the best balance between energy conservation and delay while allowing dynamic slot allocation based on application demands. The MAC protocol proposed in this research is pseudo-random based.

5.3 Related Work

TDMA based MAC protocols have several advantages over contention-based protocols especially in infrastructure networks where dynamic allocation of slots and synchronization of node schedules can be achieved using a common controller or base station. It remains a challenge to dynamically allocate slots and synchronize schedules efficiently between nodes in ad hoc wireless and sensor networks. The next section briefly describes existing MAC protocols for wireless ad hoc and sensor networks.

5.3.1 S-MAC

S-MAC is a contention-based MAC protocol designed for wireless sensor networks in [2]. To reduce idle-listening, nodes periodically sleep and form virtual clusters to auto-synchronize on sleep schedules. A node turns its receiver "off" when it goes to sleep to conserve energy. The basic protocol is as follows; each node goes to sleep for some time and then wakes up to listen and see if any of its neighbors has data to send to it.

The S-MAC protocol requires periodic synchronization between nodes to remedy clock drift. In addition, because the listen period of 500ms is much longer than typical clock drift rates, nodes can easily exchange schedules by broadcasting to their immediate neighbors. If multiple neighbors want to access the channel, they need to contend for the channel as is done in IEEE 802.11 using RTS/CTS control packets. S-MAC divides the listen period into two parts, the first part for SYNC messages and the second for RTS messages. One key feature of S-MAC is synchronized sleep wake state, where all nodes listen at the same time and go to sleep at the same time. S-MAC avoids overhearing by requiring that all immediate neighbors of both the sender and receiver sleep for the specified duration in RTS/CTS message exchanges.

5.3.2 T-MAC

T-MAC is a contention-based MAC protocol developed in [5]. Unlike S-MAC that uses a fixed sleep duty cycle, T-MAC uses an adaptive cycle to handle load variations over time. T-MAC also uses a virtual clustering synchronization scheme inspired by S-MAC to synchronize nodes. Nodes contend for active periods using RTS/CTS message exchanges. T-MAC reduces idle listening time by transmitting all messages in bursts of variable length, while sleeping and buffering messages between active periods. T-MAC uses a time-out mechanism to determine the end of the active period. A node will keep listening and potentially transmit as long as it is an active period. An active period ends when no activation event has occurred for a time T_A that determines the minimal amount per active period. Overhearing avoidance is optional in T-MAC and a node should not go to sleep while its neighbors are communicating, since it may be a receiver of a subsequent message. T-MAC suffers from an early sleeping problem where a node goes to sleep when a neighbor still has messages for it.

5.3.3 DMAC

DMAC [3] is an energy-efficient low latency MAC protocol for tree-based data gathering in wireless sensor networks. The key feature in DMAC is that the active period of nodes on the multi-hop path is staggered. DMAC divides each slot into a receiving, sending,

and sleep period. In the receiving period, a node expects to receive a data packet and it sends an Ack packet back to the sender. In the sending period, a node sends a data packet to its next-hop and it receives an Ack packet. In the sleep period, nodes turn their receivers "off" to save energy. The receiving and sending periods have the same length u that is enough for one data packet transmission and reception. Depending on its depth d in the data-gathering tree, a node skews its wakeup-scheme du ahead from the schedule of the sink. Assuming that data delivery only moves in the direction from the sink towards the root, intermediate nodes will have a sending slot immediately after the receiving slot. To reduce message overhead, DMAC does not employ the use of RTS/CTS messages, instead, it uses CSMA to contend for the channel. DMAC uses a slot length of 10ms so fine-grained synchronization is required. The authors mention RBS [11] as a possible synchronization protocol.

5.3.4 SEEDEX

SEEDEX [12] is a TDMA based MAC protocol motivated by the poor capacity scaling of ad hoc networks. SEEDEX tries to avoid collisions without making reservations for every packet by allowing nodes to choose transmission slots opportunistically. The key idea is to generate a schedule driven by a pseudo-random number generator. By exchanging the seeds of the pseudo-random number within a 2-hop neighborhood, nodes effectively publish their schedules to all hidden as well as exposed nodes. SEEDEX defines a set of possible states for each slot – Silent, Listening for packets (L) and Possibly Transit (PT). A node wishing to transmit a packet to a neighbor could choose a slot that is state PT for itself and is state L for the neighbor. SEEDEX has two preset parameters, p and x. Parameter p specifies the probability that a node declares a slot as PT and $min\ (1,\ x/n)$ is the probability that it actually transmits in a PT slot where n is the number of nodes within a 2-hop neighborhood. The authors found a value of $x = 1.5$ and $x = 2.5$ to be optimal when offered load is low and high respectively. They also found out the maximizing value of p for a given network size appears to be insensitive to the offered load and the topology of the network.

5.3.5 LMAC

LMAC [4] is a TDMA based protocol that divides time into fixed-length frames. A frame consists of a preset number of time slots. Nodes reserve time slots for transmission so communication is collision-free. During its time slot, a node transmits a message that consists of a control message and a data unit. The control message addresses the intended receiver, reports the length of the data unit, and some synchronization information. As a result, all nodes must listen for control messages from neighboring nodes in every time slot. When a node receives a control message addressed to another node, it sleeps for the remainder of the slot to avoid overhearing. Similarly, a time-out interval ensures that nodes do not waste energy idle-listening in time slots that are not reserved. LMAC encodes time slot reservation using a number of bits equal to the number of time slots in a frame. A node can reserve a time slot that is currently not reserved by any of its neighbors and reserved slots are reusable after 2-hops. It is possible during network setup that multiple nodes will reserve the same time slot. If this happens, other nodes will detect and mark the time slot as a colliding slot in their control messages. The colliding nodes on learning that their chosen slot is colliding will attempt to reserve another time slot after a random back off time. LMAC requires that the number of time slots in a frame is larger than the maximum node-degree in the network. This ensures that every node in the network can find an empty slot in the network in finite time. The authors used a frame length of 32 time slots in their design.

5.3.6 TRAMA

TRAMA [7] reduces energy consumption by ensuring that transmissions are collision-free, and by allowing nodes to switch to a low power idle state whenever they are neither transmitting nor receiving. TRAMA consists of three components: the Neighbor Protocol (NP) and the Schedule Exchange Protocol (SEP), which allow nodes to exchange 2-hop neighbor information and their schedules; and the Adaptive Election Algorithm (AEA), which uses neighborhood and schedule information to select the transmitters and receivers for the current time slot. TRAMA uses a distributed election scheme based on information about the traffic at each node to determine which node can transmit at a

particular time slot. TRAMA assumes a single time-slotted channel for both data and signaling transmissions. Time is divided into random and scheduled access periods. Signaling occurs during random-access periods while data transmission occurs during scheduled access periods. NP propagates 1-hop neighbor information among neighboring nodes during the random access period to obtain consistent 2-hop topology information across all nodes. During the random access period, nodes perform contention-based channel acquisition. Thus, signaling packets are prone to collisions. TRAMA uses scheduled access for collision-free data exchange and for schedule propagation. Schedules essentially contain current information on traffic originating from a node and the corresponding set of receivers for the traffic. A node has to announce its schedule using SEP before starting actual transmission and nodes synchronize using GPS or other time synchronization techniques.

5.4 Summary

The three contention-based MAC protocols (S-MAC, T-MAC, and DMAC) improve upon CSMA/CA to make it energy-efficient. They address the idle-listening problem by synchronizing nodes and implementing a sleep duty cycle. Nodes in S-MAC and T-MAC maintain very loose synchronization that introduces an idle listening period at the beginning of each active slot. Although this idle listening period at the beginning of the slot may be short, its effect on energy consumption becomes significant when the offered load is low. Besides idle-listening at the beginning of each active slot, S-MAC idle listens for the maximum listen duration of 0.5s when the channel is actually idle while T-MAC uses a 15ms timeout to avoid idle-listening for an entire active period.

S-MAC and T-MAC require nodes wishing to transmit a message to contend for the channel using RTS/CTS messages that consume additional energy. Consequently, all nodes have to wakeup at the beginning of every active slot to receive these RTS/CTS messages. Waking up in every slot consumes energy and can be very wasteful when the offered load is low. S-MAC and T-MAC address the overhearing problem by ensuring that nodes go to sleep when they overhear RTS/CTS message exchanges between other

nodes. While the synchronized sleep wake behavior in S-MAC and T-MAC makes scheduling easy, it also increases the probability of collision as offered load increases. DMAC addresses some of the issues found in S-MAC and T-MAC by using fine-grained synchronization, avoiding the use of RTS/CTS messages, and ensuring that nodes have staggered active periods. Unfortunately, DMAC only works well for the data gathering tree scenario and is not a general enough to handle multiple flows between arbitrary sources and destinations. T-MAC and S-MAC achieve energy consumption reduction up to 98% compared to CSMA under homogeneous load. In a sample scenario with variable load, T-MAC uses 5 times less energy than S-MAC since T-MAC uses an adaptive sleep cycle [5]. However, the early sleeping problem in T-MAC can degrade performance and is reason for concern. DMAC achieves better energy-efficiency than S-MAC and T-MAC with significantly lower delay for tree-based applications.

As offered load nears channel capacity, the number of collisions increases significantly in contention-based MAC protocols. T-MAC and S-MAC have been shown to collapse under high offered load [18,19] due to contention overhead and lack of adaptation. The TDMA based protocols (L-MAC, SEEDEX, and TRAMA) attempt to eliminate collisions by ensuring that nodes transmit in reserved slots. Although the slot reservation scheme in LMAC ensures that there are no collisions, LMAC incurs high latencies and reduced throughput since nodes can reserve at most one time slot within a frame. In addition, LMAC requires nodes to wakeup at the beginning of every slot to listen for control and synchronization messages. Waking up in every slot winds up dominating the total energy consumed when the offered load is low. TRAMA achieves a maximum throughput of about 40% over S-MAC and CSMA and 20% over 802.11 at the expense of considerable high latency and high algorithmic complexity [18]. TRAMA also requires nodes to exchange traffic schedules; the message overhead for exchanging these schedules can very high when the offered load is high. SEEDEX, on the other hand, uses a pseudo-random scheme to determine transmit slots without explicitly exchanging traffic schedules. SEEDEX achieves a maximum throughput of about 10% greater than 802.11 but it does not guarantee collision free transmission [12]. However, choosing an appropriate set of preset parameters depending on the offered load minimizes the

probability of collision. All three TDMA schemes assume the use of some time synchronization protocol without factoring in the energy implications of the chosen synchronization protocol.

The contention-based protocols (S-MAC, T-MAC, and DMAC) perform well when the offered load is low and collapse under high offered load. The TDMA-based protocols (LMAC, SEEDEX, and TRAMA), on the contrary, perform well when the offered load is high but suffer from reduced throughput, inefficient synchronization techniques, high scheduling messages overheard, and high latencies when the offered load is low. Clearly, an adaptive MAC protocol is needed to address these issues.

6 System Architecture

Figure 2 shows the system architecture and highlights key subsystems that interact closely to achieve energy-efficient operation of the entire system. The MAC component is composed of four main subsystems- Data Link Lower Layer (DLLL), Mac State (MS), Neighbor Discovery (ND) and Link Characterization (LC). The PHY component is composed of the Hail and Data radios. The bold arrows in Figure 2 indicate Over the Air (OTA) data flow within the system. Besides the MAC and PHY components, other components necessary for multi-hop data delivery are Routing, Forwarding, and a transport layer interface [1]. This report will not present a detailed discussion of these additional components since they are not critical to the energy savings of the entire system and are shown here only for completeness.

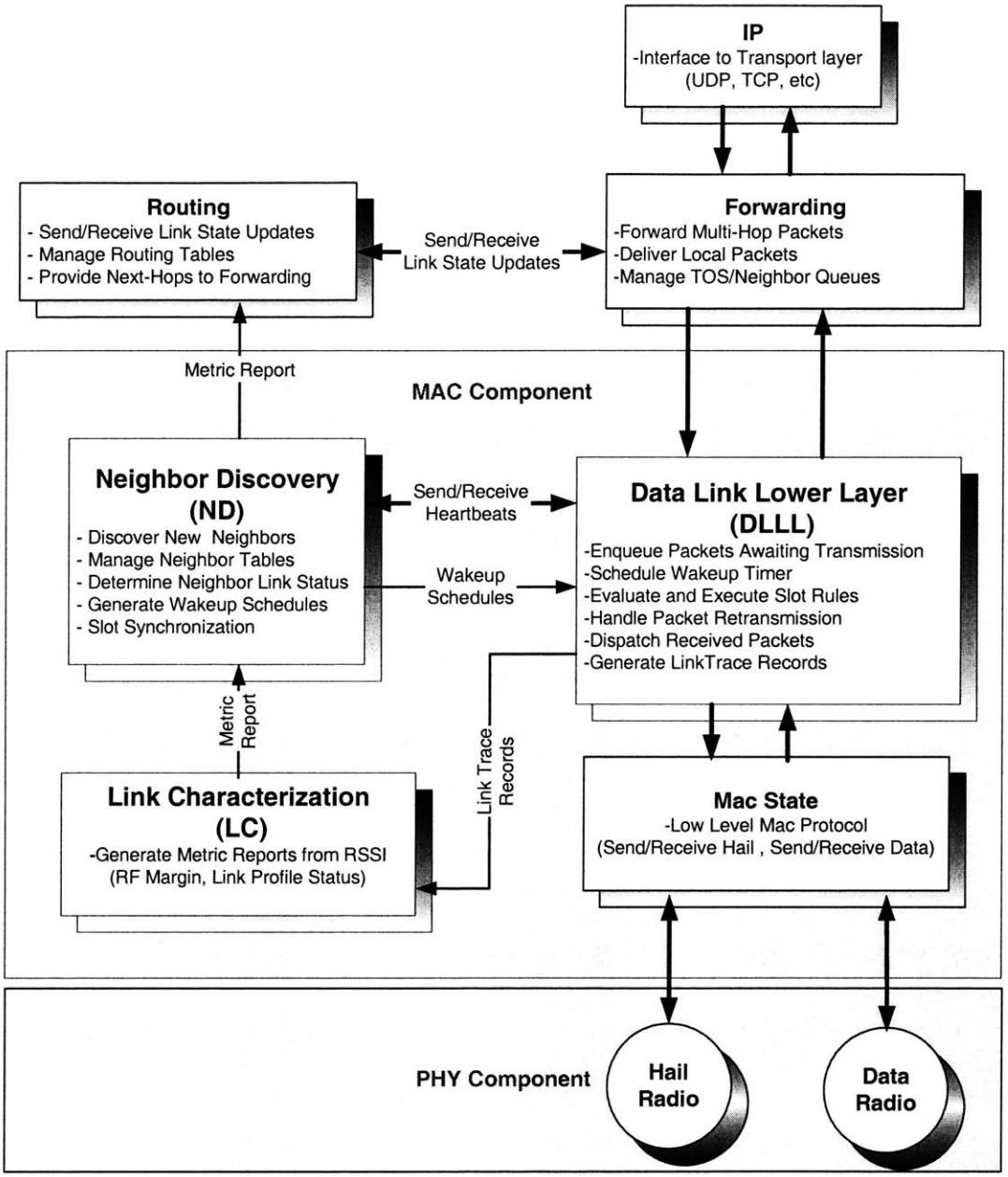


Figure 2: System Architecture.

7 PHY Component

The PHY component has two portions – a low power, low data rate Hail radio, and a high power, high data rate Data radio [1]. The Hail radio transmits data at a rate in the order of tens of kilobits, while the Data radio transmits at megabit or higher data rates. While the total energy per bit is theoretically the same regardless of data rate, a low data rate radio actually has the advantage of needing less accuracy and reliability in terms of oscillators and mixers (e.g. more "slop" is allowable when the data rates are lower). Therefore, it is actually possible to design a low data rate radio that uses les power than a high data rate radio scaled by a reduced time per bit. The low data rate Hail radio occasionally wakes up to receive Hail signaling between nodes. The signaling indicates whether to turn on the high power Data radio to receive Data packets. The Data radio stays in the "off" state until the Hail radio detects the correct signaling. The Hail and Data radios need to use approximately the same frequency band so that:

- Fading and path loss in the Hail packets assists in power control for the Data packets.
- One can expect a similar energy-per-bit needed for the Data packet as well as the Hail packet that precedes it.

Of course, the Hail and Data radios could actually be the same radio with modes that have significant differences in data rate and power requirements. This research assumes a two-radio architecture similar to [24,26] without loss of generality.

7.1 Hail Radio

The core idea behind Hail packets is that there is a low power Hail radio that the MAC deterministically wakes up to see if anyone wishes to send data to the higher power Data radio. A Hail packet is essentially a means of signaling the MAC to wake up the Data radio. Each Hail packet contains very little data— 32 to 48 bits. When the network duty cycle is very low, the likelihood of Hail packet collision is negligible. However, as the duty cycle increases, occasional Hail packet collisions can occur. In this case, the MAC adapts its transmission schedules to provide collision free access. The Hail Radio design provides robust frequency-hopped FSK modulation and packet repetition over several

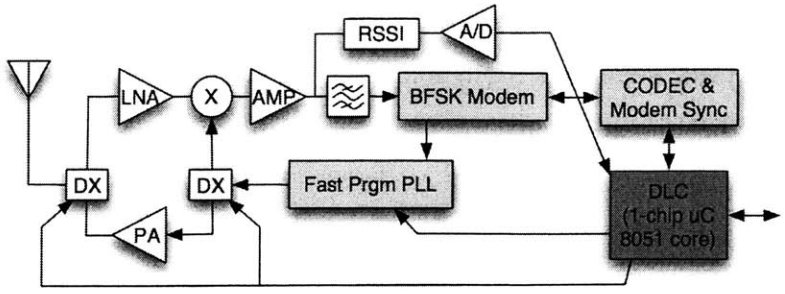


Figure 3: Chipcom SmartRF® CC1010 transceiver.

hops to provide diversity gain against narrowband fades and interference without a need for either channel estimation or complex signal processing. Furthermore, the Hail radio uses a Reed Solomon code to improve random hop selection.

The Hail radio function requirements match up rather well to several very low power all-CMOS transceivers and so-called commercially available "systems on a chip". The key design element is an extremely simple receiver circuit operating with the least energy practical. These devices usually power up from very low power standby or idle modes very quickly (within a few milliseconds) to provide up to 100 Kbps transport. The more capable of these devices support frequency hopping. The robustness against interference and multipath fading that frequency hopping offers is critical to reducing network control traffic. The benefits include fewer message retransmissions, fewer fallbacks to search modes, and the ability to sort out simultaneous channel access attempts by multiple nodes without resorting to a contention protocol and further packet transmissions.

The Hail transceiver chip considered for this research is the Chipcom SmartRF® CC1010 shown in Figure 3. This low power transceiver design expends minimum energy on network control and maintenance. Employing some form of multipath fade mitigation noticeably improves the reliability of the Hail transceiver. One multipath fade mitigation technique is repeating the Hail over a small selection of frequencies within the wide band. For each frequency, the Hail transceiver listens for a reply before moving to the next frequency, thereby minimizing the number of Hail attempts and energy used in transmitting Hail messages. For consistent and predictable frame timing across the

network, the Hail transmitter transmits the full number of repetitions. However, the MAC begins to power up the Data radio and begin frequency settling it as soon as the MAC receives a Hail packet indicating an impending Data packet of interest as shown in Figure 4.

The Hail transceiver centerpiece is a fast settling programmable phase locked loop (PPLL) frequency synthesizer. This fast PPLL design permits frequency hopping at rates up to 1000 hops per second. At half this maximum hop rate and at the highest data rate supported, 76.8 kbps, this allows for five independent frequency Hails with 32 bits of data and 60 bits preamble each, with a total power-on time of just 10 ms predicated on a 14.7456 MHz PPLL oscillator. Synchronizing to a transmitted Hail at least once every 100 seconds supports a Hail time-of-arrival uncertainty of 1 ms; dropping the hop rate to 500 hops per second therefore assures that this 1 ms time uncertainty is covered. One feature of the CC1010 that is particularly useful is the received signal strength indicator (RSSI) circuit analog output that can be configured for sampling by an on-chip A/D converter and presentation on a DLC register. This information, along with transmitted power level information encoded in the Hail message, is useful for adaptation within the MAC.

The fundamental idea—trading modulation bandwidth efficiency (receiver sensitivity) for very loose oscillator accuracy requirements resulting in very fast effective oscillator settling time for nearly instant circuit start up—can be incarnated in a number of alternative ways. In receive mode, the CC1010 consumes only 30 mW near 450 MHz (9.1mA @3.3V) and 39.3 mW near 850 MHz (11.9mA @3.3V). Besides the CC1010, other low power transceiver chips commercially available are the Micrel Semiconductor MICRF500 (operation near 850 MHz) and the MICRF501 (operation near 450 MHz).

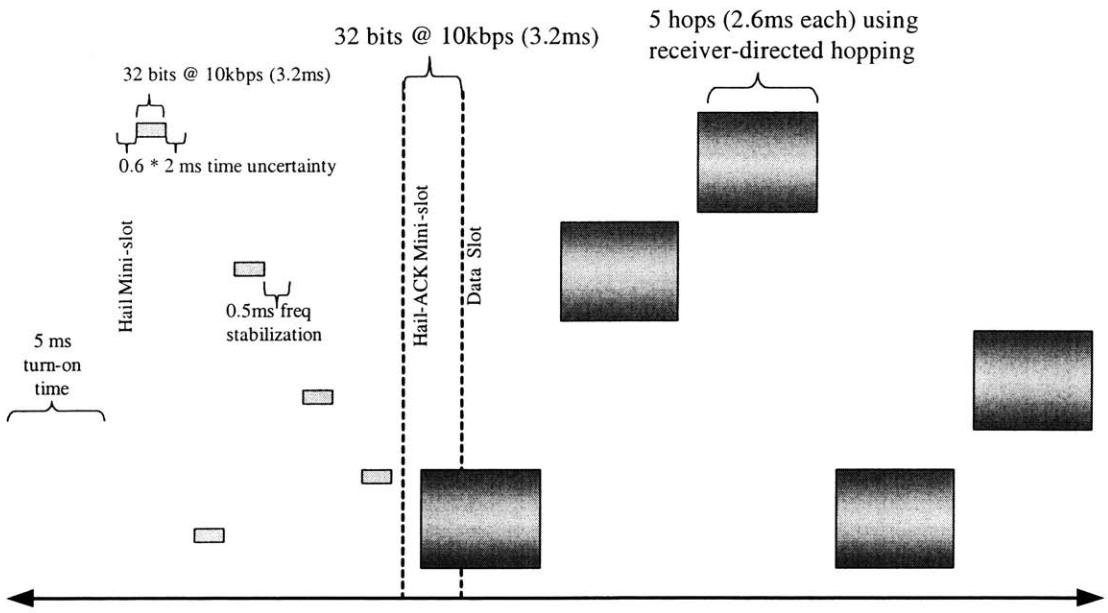


Figure 4: Hail and Data radio frequency hopping within a slot.

Appendix A shows a detailed comparison of the Micrel and Chipcon devices. The CC1010 transceiver has a small disadvantage in raw power consumption and Adjacent Channel Interference (ACI) reject performance but supports an order of magnitude higher hop and data rate (commensurate with the ACI performance factor). The faster hop and data rate, in addition to a faster oscillator turn on time, more than compensates the difference in power use.

7.2 Data Radio

The high data rate Data radio meets the need for timeliness and energy-efficiency via power control in transporting large application data across the network. This research assumes a Data Radio transceiver that is replaceable with nearly any other high power and high data rate transceiver that provides similar capacity and performance. The Data Radio uses a SDPSK/DOQPSK modulation that supports very power-efficient power amplifiers and uses a 5/7 Reed-Solomon codec. The modulation scheme ensures a tight spectrum that limits unintentional interference to adjacent channel receivers. In addition,

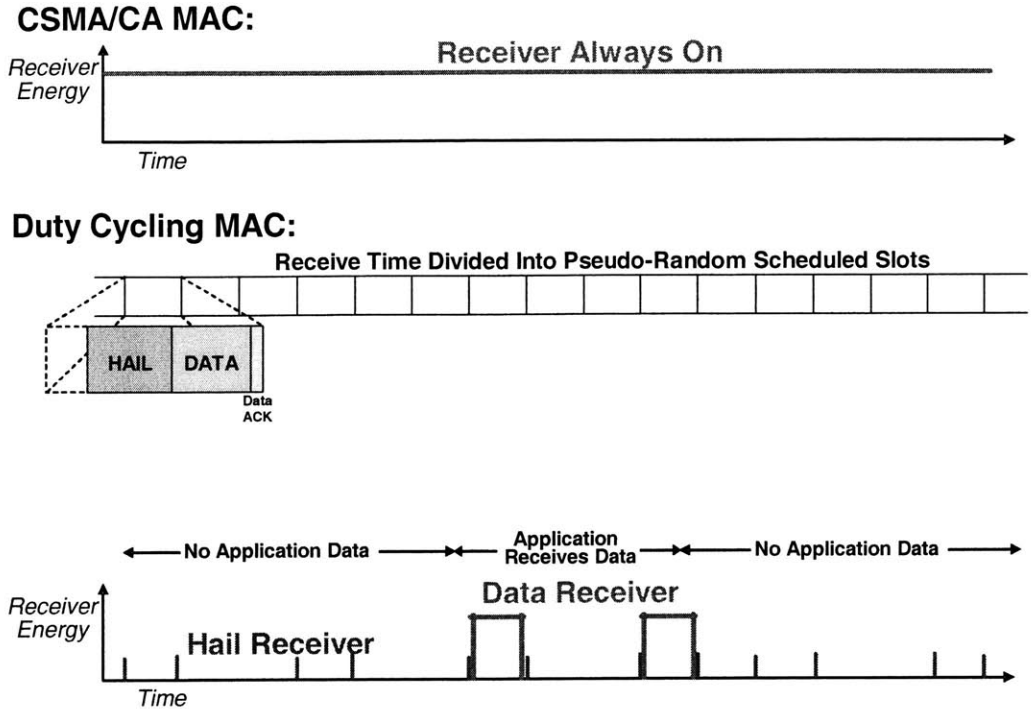


Figure 5: CSMA/CA vs. a duty cycling MAC.

the Data radio transceiver spans 90+ dB received power range with excellent error rate performance. Since the Data radio is not critical to the energy savings of the entire system, a detailed treatment of its functioning will not be presented.

8 MAC Component

The MAC Component manages the 1-hop exchange of data between two nodes. This is achieved by waking up the Hail radio at predetermined times, and only waking up the Data radio when the MAC determines that data packets are ready to be sent or received (Figure 5). The MAC component also manages 1-hop neighbors in a way that dynamically adapts to network changes, traffic rates, and application requirements. In addition, the MAC component comprises multiple subsystems that perform the following key functions:

1. Slot synchronization – Determining and maintaining accurate synchronization between neighboring nodes. Note that nodes do not need to have an agreement on

an epoch or other global slot-numbering scheme; they just need to agree on the periodicity of slots with some precision.

2. Neighbor Discovery – Managing neighbors and generating schedules for a node and all of its neighbors.
3. MAC Protocol - Managing data transmission, retransmission, and reception based on the schedules.
4. Link Characterization – Determining an appropriate set of metrics based on PHY layer information to describe the link status for each neighbor.

8.1 Mac Protocol

The MAC divides time into fixed sized slots. Each slot contains up to three "sub slots" – a Hail mini-slot, an optional Hail-Ack mini-slot, and a Data slot. A node will listen with its Hail radio in a subset of the Hail mini-slots. When a node *A* has a Data packet to send to node *B*, *A* first sends a Hail packet to node *B* in *B's* Hail mini-slot using its low data rate Hail radio. The Hail packet indicates *B* as the desired recipient of the subsequent Data packet transmission and it indicates whether *A* requests a Hail-Ack in response to the Hail. The Hail provides other information such as the code-rate for the upcoming Data packet transmission. If *A* requests a Hail-Ack, then *B* responds to the Hail with a Hail-Ack. The MAC sends the Hail-Ack in the Data slot using the Hail radio. After the Hail and the optional Hail-Ack, the MAC turns on the Data radio to receive the actual Data packet. If a Data-Ack is requested by node *A*, the Data packet is immediately followed by either a Data-Ack or Data-Nack from node *B* to node *A* using the Data radio.

Note that a node sends a Data-Nack if it receives only the Hail packet and not the subsequent Data packet. A node sends a Data-Nack for other reasons such as cross-layer indications that the destination is not reachable via this node as a next-hop. The Data-Nack therefore includes an error code describing the reason for the Nack. Since the Hail does not indicate the length of the packet, the receiver has no choice but to wait until the end of the Data slot to send the Data-Nack since it does not know when the transmission

of the Data packet ended. Note that in some cases, putting the Data-Nack right after a data packet and right at the end of the slot is the same thing if the packet is of maximal length. In all cases, however, there is enough room at the end of a slot to send either an Ack or a Data-Nack.

8.1.1 Generating Schedules

Most radios spend a significant amount of time in receive mode. As a result, most MAC protocols assume that neighboring nodes are always awake and ready to receive data. In low offered load environments, keeping the receiver "on" all the time causes the amount of energy consumed by the receiver to dominate the total energy consumed by the node. To address this problem, the MAC in this research employs the use of a very low power and low data rate Hail radio that listens at particular times to determine whether to turn on the Data radio or not. While it is acceptable for the Hail radio to be always "on" and ready to receive data in every Hail mini-slot, it is not advantageous to do so from an energy standpoint. Keeping the Hail radio "on" in every single Hail mini-slot fixes a high lower bound on the total amount of energy used by the node as the Hail radio winds up dominating the total amount of energy used over time. In low offered load environments, it makes sense to put the Hail radio to sleep in some slots. Consequently, the MAC uses a pseudo-random sequence to define when a node's Hail receiver should be awake and when it should be asleep.

The MAC uses a linear congruential algorithm to produce pseudo-random numbers that it uses together with a supplied threshold to generate wakeup schedules. The algorithm takes the form of $I_{j+1} = (aI_j + c) \ mod \ m$. Using values of $a = 7^5$, $c = 0$, and $m = 2^{31}-1$ yields a reasonably good generator. In an actual implementation, however, it may be more efficient to preload each node with a very large table of pseudo-random numbers. A node can easily figure out exactly which pseudo-random numbers its neighbor picks for a particular slot using the neighbor's advertised seed and cycle position.

The pseudo-random number generator (PRNG) allows the MAC to generate a schedule that allows the Hail radio to listen only in particular slots. For a given seed, a PRNG generates a long list of pseudo-random numbers distributed between 0.0 and 1.0. The list of pseudo-random numbers eventually repeats over a very long period or cycle. The cycle position is the current position within the period. When a node generates a pseudo-random number, it compares the number against a probability threshold that determines whether the node should be awake or not. For example, if the PRNG generates 0.31 and the probability threshold is 0.40, then the node will be awake since 0.31 is less than 0.40. If the PRNG generates a 0.73, the node will sleep instead. The PRNG uses the unique node-id assigned to each node as a seed. The advantages of using a PRNG are:

- Schedules are essentially uncorrelated across nodes.
- Each node compactly represents its wakeup schedule using probability thresholds, cycle positions, and a seed.
- A node can dynamically decrease or increase the amount of time it stays awake by adjusting its probability threshold.

Each node knows the pseudo-random number generator (PRNG) used by every other node in the network. Thus, given a cycle position, a probability threshold, and a seed, a node can easily generate every other node's schedule. Pseudo-random number generators also have the advantage that they are essentially uncorrelated between two nodes (given different seeds) and the number of wakeup slots they define can be dynamically determined by setting the appropriate probability threshold. The MAC uses pseudo-random sequences to determine when the Hail receiver should be "on" and listening for transmissions from neighboring nodes. The use of pseudo-random sequences allows nodes to have uncorrelated, but completely predictable wake-up times. Furthermore, the frequency of wakeup is adapted on a per node basis given particular throughput, latency, and energy requirements. Each node maintains three types of pseudo-random sequences – one each for unicast-transmit, unicast-receive, and broadcast-transmit. A node maintains its own unicast-transmit schedule in addition to the unicast-receive and broadcast-transmit schedules of itself and all its neighbors.

Each node *i*, maintains and reports the following items to its neighbors through periodic heartbeats:

- Node *i*'s unicast-receive probability threshold (PUR_i) and unicast-receive cycle position (CUR_i).
- Node *i*'s broadcast-transmit probability threshold (PBT_i), and broadcast-transmit cycle position (CBT_i).
- Node *i's* unicast-transmit probability threshold (PUT_i) and unicast-transmit cycle position (CUT_i) in addition to the unicast-transmit probability and unicast-transmit cycle position of *i's* neighbors. (*PUT* and *CUT* values are distributed up to 2-hops).

PUR_i and CUR_i define slots that node *i* will be awake listening for unicast traffic. PBT_i and CBT_i define slots that node *i* can use to send broadcast traffic. PUT_i and CUT_i define slots reserved for node *i* to send unicast traffic. A node is required to be awake and ready to receive data in the broadcast-transmit slots of its neighbors and in the unicast-receive slots of itself. Note that the unicast-transmit and unicast-receive uniquely define when nodes can send and receive unicast traffic. A node can send unicast traffic to another node in a particular slot if the slot is both a unicast-transmit of the sender and a unicast-receive of the recipient.

In each slot, a node *i* generates a new pseudo-random number for itself and determines the pseudo-random number that any its neighbors will draw in the same slot using the neighbor's advertised cycle position. The number returned by a PRNG is compared against a given probability threshold. If the number is less than or equal to the probability threshold, the slot is marked "1" and if the number is greater than the threshold, the slot is marked "0". The Hail radio stays awake in slots marked "1" and sleeps otherwise. Cycle positions are incremented modulo the known cycle period, so that a node can accurately predict when a neighbor will be awake in the future by just incrementing the cycle position.

The choice of probability thresholds is critical for the energy-efficient operation of the entire MAC. Probability thresholds determine how often a node will be awake for unicast-receive, as well as how often a node requires that its neighbors listen for its broadcast-transmit. Unicast-receive slots can also be used for discovering new neighbors, so in a wide variety of scenarios, a node switches to "always listening" mode (unicast-receive probability threshold of 1.0) when it believes that the network is in a state of flux. All probability thresholds are dynamically determined based on the perceived need for the Hail receiver to be "on" to minimize delay while balancing the desire to keep it "off" as much as possible to conserve energy.

8.1.2 Unicast-Receive Schedule

The MAC determines unicast-receive slots by the unicast-receive probability threshold *PUR*, seed *SUR*, and cycle position *CUR*. If the PRNG for node *i* generates a number less than or equal to *i's PUR* then node *i* sets its Hail receiver to be "on" in the Hail mini-slot. Neighbors of node *i* know the same parameters (i.e. *PUR*, *SUR*, and *CUR*) used by *i* therefore they know when node *i* is awake and ready to receive data.

A node always starts up with a maximum *PUR* of 1.0 (i.e. Hail radio is awake in every Hail mini-slot), and then it gradually reduces this probability threshold to a configurable minimum depending on network stability and traffic requirements. Anytime a node determines network changes, it sets *PUR* to a maximum of 1.0. Increasing *PUR* to a maximum of 1.0 allows a node to receive heartbeats and then learn the *PUR*, *SUR*, and *CUR* values of other nodes in the network. A node usually increases it *PUR* to a maximum of 1.0 when:

- It is first powered up
- It detects a new neighbor
- It learns of a new 2-hop neighbor
- It learns of a link state change within *k* hops
- It becomes isolated (i.e. losses all of its neighbors)

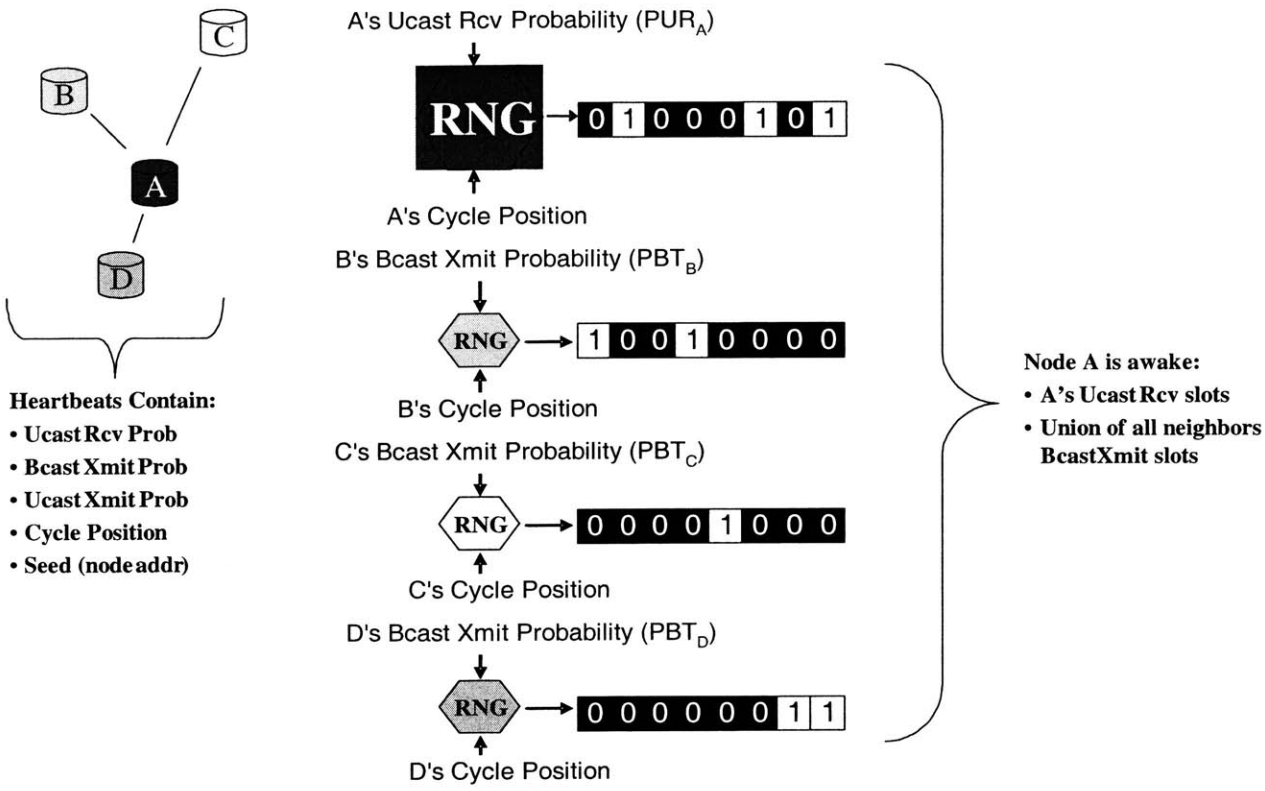


Figure 6: When to turn the Hail Radio "on".

All of the reasons above indicate changes within the local neighborhood of a node so it imperative that the node keep its Hail receiver "on" as often as possible. Keeping the Hail "on" in every Hail mini-slot allows a node to effectively capture and react to the on-going network changes. A node keeps the Hail "on" in every Hail mini-slot until the network stabilizes (e.g. no new heartbeats or link state updates are received). The *PUR* decreases appropriately once the network stabilizes.

Besides network changes, nodes can dynamically adjust their *PUR* to match traffic requirements. Each node maintains a running average percentage of unicast-receive slots actually used to receive data. This average runs over a configurable window size. If the percentage falls below a configurable threshold, the *PUR* reduces by a configurable factor since the node is under-utilizing its unicast-receive slots. Similarly, if the slot usage exceeds a certain configurable threshold, the *PUR* increases by a configurable factor to allow the node to receive more data. When a node changes its *PUR*, it notifies

its neighbors with the updated value. Note that it is advantageous to ramp up the *PUR* very quickly when needed and to decrease it very slowly to find the right equilibrium level without unduly affecting end-to-end delay. In addition, the *PUR* should not fall below a configurable minimum so that during extended periods of network inactivity, the whole network does not drift into a state where all nodes are asleep for exceedingly long times and communication is virtually impossible.

8.1.3 Broadcast-Transmit Schedule

The MAC uses broadcast-transmit slots to send control data such as heartbeat beacons and routing link state updates. A node is required to turn its Hail receiver "on" in the broadcast-transmit slots of all its neighbors so the choice of *PBT* has serious implications on the amount of energy consumed by each node. Setting the appropriate initial *PBT* requires modules that generate broadcast traffic to predict the amount of data that they are likely to send. The Routing component and Neighbor Discovery (ND), for instance, can accurately predict the average amount of broadcast traffic they are likely to send over a period. ND sends heartbeat beacons at a known rate. Similarly, Routing sends periodic link state updates at a known rate. Routing also sends event driven link state updates at a rate proportional to the max rate of occurrence of the specified event. The number of link state updates rebroadcast within the network is proportional to the size of the network. Given all this information, predicting the amount of control broadcast traffic is straightforward.

Even though ND initializes each node with an initial *PBT* value that essentially maps to the average number of broadcast slots needed to transmit heartbeat beacons and routing link state updates, it is possible for nodes to dynamically increase or decrease this threshold to support application broadcast data. The scheme for dynamically adjusting the *PBT* value is similar to the one used for adjusting the unicast-receive threshold *PUR*. In this case, however, each node maintains a running average percentage of broadcast-transmit slots that are actually used to send broadcast data. This average runs over a configurable window size.

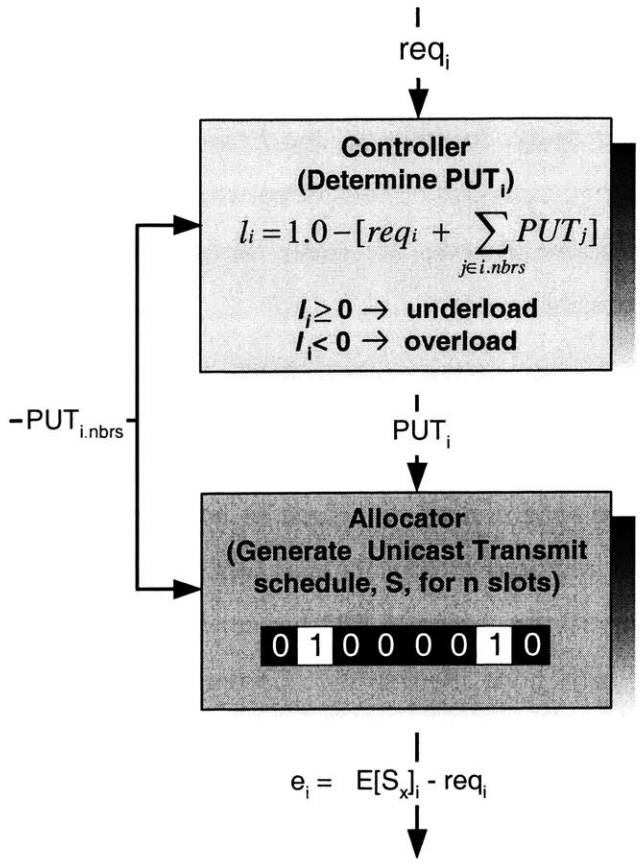


Figure 7: Dynamic slot allocation using an adaptive controller.

If the broadcast-transmit slot usage exceeds a certain configurable threshold, the *PBT* increases by a configurable factor to allow the node to send more broadcast data. Similarly, the *PBT* decreases by a configurable factor when the usage falls below a configured threshold. Like *PUR*, when a node changes its *PBT*, it notifies its neighbors with the updated value.

8.1.4 Unicast-Transmit Schedule

A node's *PUT*, which is a function of its throughput requirements, determines how often the node gets the chance to transmit unicast data. *PUT* values range from a minimum value of 0.0 to a maximum of 1.0. Nodes only have to exchange *PUT* values within a 2-hop neighborhood. Assuming a Unit Disk Graph model for ad hoc networks, exchanging

PUT values up to 2-hops avoids collisions due to "hidden terminals" and it allows slot reuse after 2-hops. In addition, exchanging *PUT* values allows the MAC protocol to adapt effectively to the varying throughput requirements of each node.

8.1.4.1 Dynamic Slot Allocation

This section describes a simple distributed algorithm that dynamically allocates collision-free unicast-transmit slots within a 2-hop neighborhood. Besides its simplicity, a key feature of the algorithm is that it is energy-efficient and it does not require nodes to exchange traffic schedules. In addition, the algorithm attempts to achieve max-min fairness for an arbitrary number of node requirements. An algorithm is max-min fair if it maximizes the minimum allocation to a node whose requirement is not fully satisfied.

The algorithm is composed of two cooperating parts – a controller and an allocator (Figure 7). The controller adapts to the throughput requirements of a node and allocator is responsible for the actual allocation of slots to match the requirement. A node's throughput requirement, *req*, is conveniently expressible as a request ratio. For example, if a node sets its *req* to 0.2, it is essentially requesting 2 slots in every 10 slots for unicast-transmit. The actual ratio of slots received by the node depends on the *req* values of other nodes in its 2-hop neighborhood. Although the optimal allocation of slots such that throughput is always maximized in TDMA networks is *NP*-hard [22,23,27], the controller attempts to maximize network throughput by allowing nodes to transfer unused slots to other nodes that need extra slots. Transferred slots can later be requested for by simply manipulating the *req* input to the controller.

```
Slot Allocation Algorithm:
N ← nodeid of all nodes within 2-hops
PUT ← PUT of all nodes within 2-hops
my_nodeid ←N0

for ( x=0;  x < w;  x++ )
    R← ∀ i, set of random numbers chosen by Ni for slot x
    S ← ∀ i, set of i's s.t Ri ≤ PUTi

   // Case 1
     if ( |S| == 1 )
        y = S0
        winner ← Ny

  // Case 2
    else if(|S| > 1 )
        R ← ∀ i, set of random number chosen by Si for slot x
        y ← Si s.t Ri = max(R)
       winner ← Ny

  // Case 3
    else
       R ← ∀ i, set of random numbers chosen by Ni for slot x
       winner ← Ni s.t Ri = max(R)

// Assign Slot
   if ( winner == my_nodeid )
      slotx ←1
  else
     slotx ← 0
end
```

Figure 8: Dynamic slot allocation algorithm for w slots. Each slot, x, is assigned to at most one node within a 2-hop neighborhood.

The algorithm shown in Figure 8 generates a collision-free unicast transmission schedule S at each node for a specified number of slots, w. For all slots x, at most one node assigns x to itself if it is deemed the winner of that slot. The probability of winning a slot x, $E[S_x]$, is shown in Figure 9.

Let S be the transmission schedule generated by the slot allocator for node i, and for slots x = 1, 2 ...w define

$S_x = 1$ *if slot x is allocated to node i*
$= 0$ *otherwise*

$$E[S_x] = 1 * P(S_x = 1) + 0 * P(S_x=0)$$
$$= P(S_x=1)$$

The event $S_x = 1$ *occurs under the following cases:*

Case 1: Single potential winner
Node i wins the slot and all n-1 neighbors (up to 2-hops) of i lose the slot.
This happens with probability $PUT_i * \prod_{j \in i.nbrs} (1\text{-}PUT_j)$

$$\therefore P_1 = PUT_i * \prod_{j \in i.nbrs} (1\text{-}PUT_j)$$

Case 2: Multiple potential winners
Node i wins the slot together with m = 1 to n-1 neighbors (up to 2-hops).
This happens with probability

$$PUT_i * \sum_{m=1}^{n-1} \binom{n-1}{m} * \prod_{j \in m} (PUT_j) * \prod_{j \notin m} (1\text{-}PUT_j)$$

The slot is assigned randomly from the set of m potential winners including i

$$\therefore P_2 = PUT_i * \sum_{m=1}^{n-1} \binom{n-1}{m} * \frac{1}{m+1} * \prod_{j \in m} (PUT_j) * \prod_{j \notin m} (1\text{-}PUT_j)$$

Case 3: No potential winners
Neither node i nor its neighbors (up to 2-hops) win the slot.
This happens with probability $(1\text{-}PUT_i) * \prod_{j \in i.nbrs} (1\text{-}PUT_j)$

The slot is assigned randomly from the set n of nodes in 2-hop neighborhood

$$\therefore P_3 = \frac{1}{n} * (1\text{-}PUT_i) * \prod_{j \in i.nbrs} (1\text{-}PUT_j)$$

$$\Rightarrow E[S_x] = P(S_x = 1) = P_1 + P_2 + P_3$$

Figure 9: Analysis of dynamic slot allocation algorithm.

The controller uses a max-min fair algorithm to determine the appropriate *PUT* value for a particular *req* input at each node. For every 2-hop neighborhood N, if $C_N = 1.0$ is the capacity of N, the following constraints exist on the vector $PUT = \{PUT_i \mid i \in N\}$ at each node.

$$PUT_i \geq 0 \quad \forall\ i \in N$$

$$\sum_{i \in N} PUT_i \leq C_N \quad \forall\ i \in N \tag{1}$$

A vector of *PUT* values satisfying these constraints is said to be efficient. In addition, a vector of *PUT* values is said to be max-min fair if it is efficient, and for each node i in N, PUT_i cannot be increased while maintaining efficiency without decreasing PUT_j for some other node $j \neq i$ for which $PUT_j \leq PUT_i$ [9,10].

The input to the allocator at each node i is a vector of *PUT* values of all the nodes within *i's* 2-hop neighborhood including i and the output is a transmission schedule S. The algorithm for generating S is shown in Figure 8. The unicast transmission schedule S can be visualized as a sequence of independent and identically distributed (i.i.d) Bernoulli random variables where each slot a node, i, can transmit in has a value of "1" with probability $E[S_x]_i$ and "0" otherwise (Figure 9). Since every slot is assigned to i with probability $E[S_x]_i$, $E[S_x]_i$ also represents the ratio of slots assigned to i.

The allocation error $e_i = E[S_x]_i - req_i$. The value of e at each node can either be positive, negative, or zero depending on the node's *req* and the *reqs* of other nodes within a 2-hop neighborhood. The value of e may be useful to applications since it affects throughput and latency. A negative value of e means that the controller cannot meet the node's current *req* for slots while a non-negative value of e means *req* has been met. The total sum of *reqs* within a 2-hop neighborhood is unknown to the controller, however, the controller can determine whether the sum exceeds 1.0 or not by evaluating:

$$l_i = 1.0 - [req_i + \sum_{j \in i.nbrs} PUT_j] \qquad (2)$$

The value of l represents either an overload, or underload of *reqs* within the neighborhood. There is a close relation between the values of e and l. e is a local error measure while l is a property of the neighborhood. The controller at each node uses l to determine the state of the neighborhood.

Overload State:

A controller detects overload when the value of l within its 2-hop neighborhood of size n is negative. Overload means there is more demand for transmission slots than is physically available. This also implies that the current request for the allocation of slots is inefficient. To correct this, the controller attempts to ensure efficiency by setting PUT_i to the output of the max-min fair algorithm for node i. When a controller detects a change in its *PUT*, it immediately updates its neighbors (up to 2-hops) with the current value so that other controllers can adapt their *PUT* values. Figure 10 shows in 2D how a controller adapts to overload by moving a point P_2 that is otherwise inefficient and unfair to P_2^1 that is both efficient and fair. When l is negative, controllers make the necessary adjustments to ensure that *PUT* values approach the Efficiency Line.

Underload State:

A controller detects underload when the value of l within a 2-hop neighborhood of size n is greater than or equal to 0. A positive l value implies that there are un-requested slots available for use. In this case, $PUT_i = req_j$, and this is also the initial state of the controller. The max-min fair algorithm used by the controller ensures that a node's *req* is always met when there is underload. After meeting the node's *req*, the allocator assigns an additional $k=l/n$ unicast-transmit slots within the neighborhood to each node. There is no energy cost associated with assigning these extra transmission slots since unicast-transmit slots do not determine when the Hail radio is turned "on" as shown earlier in Figure 6. In addition, allocating these additional slots is an optimization that helps to reduce packet latency as they serve as a buffer of slots for handling temporary spikes in network traffic.

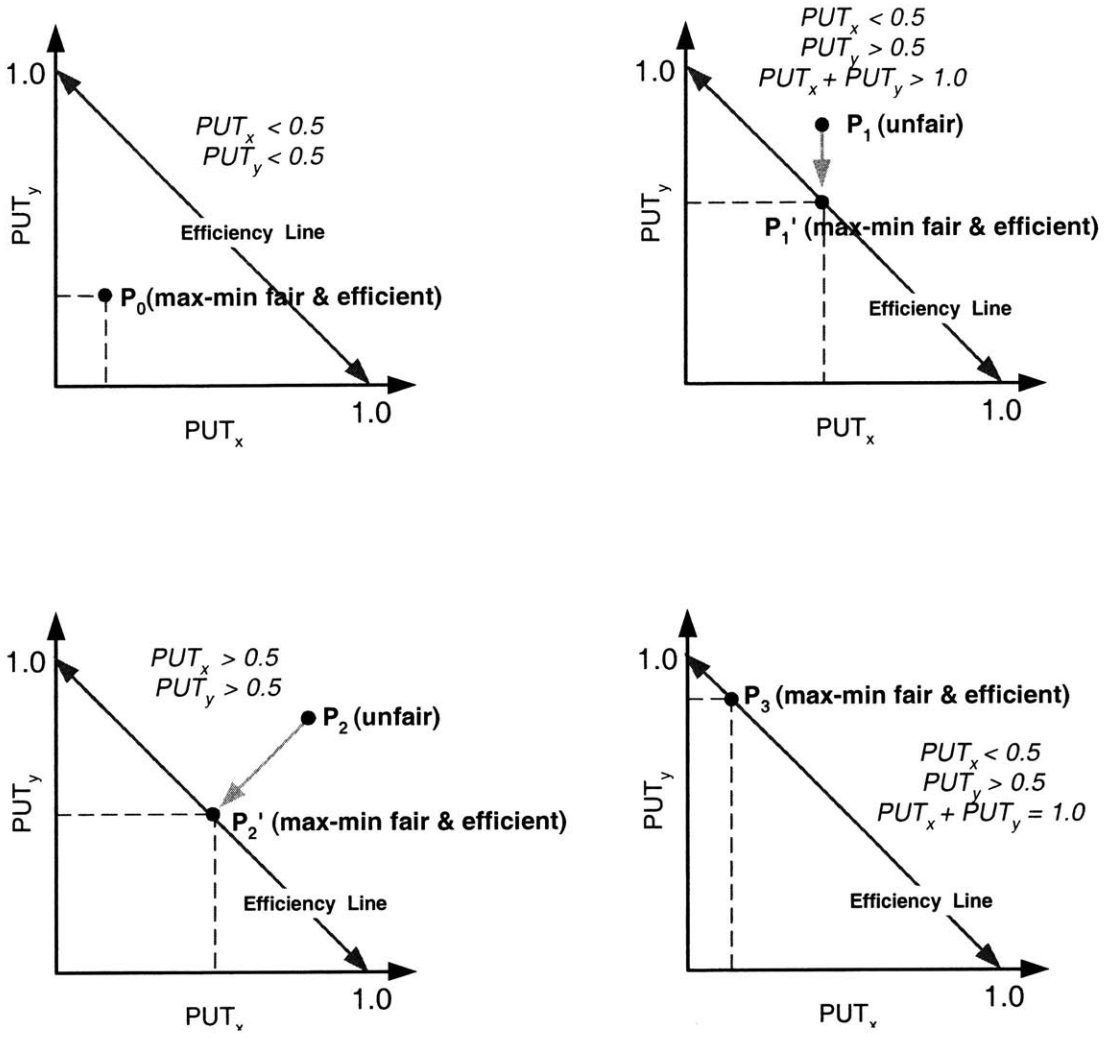


Figure 10: The controller adapts PUT values towards the Efficiency Line when there is overload.

8.1.5 Scheduling and Slot Rules

After generating the prescribed set of schedules, a node must follow these scheduling rules to determine when it can either transmit or receive data.

1. A node is expected to have its Hail receiver "on" and ready to receive data in:
   - Slots marked as "1" for its unicast-receive schedule.
   - Slots marked as "1" for any of its neighbor's broadcast-transmit schedule.

2. A node can send broadcast traffic in:
   - Slots marked as "1" for its broadcast-transmit schedule
3. A node can send unicast traffic to any neighbor in:
   - Slots marked as "1" for its unicast-transmit schedule and also marked as "1" for any of its neighbor's unicast-receive schedule

Since unicast-receive, broadcast-transmit, and unicast-transmit slots are assigned independently using the PRNG, it is possible for a single slot to be "1" for both unicast-receive and broadcast-transmit. To resolve conflicts of this nature, all nodes in the network must follow an additional set of rules.

1. If a node i has a broadcast-transmit slot, all neighbors of i have to be in receive mode ready to accept data from node i (i.e. broadcast-transmit slots always preempt unicast-transmit and unicast-receive slots)
2. If node i has a broadcast-transmit slot, but it does not have any broadcast data to send, then it is permissible for node i to send unicast data since all of node i's neighbors are awake and ready to receive a packet from i.
3. If a node i has a unicast-transmit slot and it has a packet to send to node j which indicates a unicast-receive slot, i will send the packet to j even if the slot is a unicast-receive slot for node i as well. (i.e. unicast-transmit has precedence over unicast-receive)

8.2 Node Throughput

Node throughput is essentially a measure of successful unicast transmissions achievable by a node in the network. In each assigned unicast-transmit slot, a node can only transmit data to a neighbor that is awake within the same slot. Hence, a node's throughput depends on how often it gets to transmit data, $E[S_x]$, and how often its neighbors are awake to receive data, PUR. Given that transmit and reception schedules are independent, the throughput of a directed link l_{ij} between a node pair (i, j) is $(E[S_x]_i * PUR_j)$. The link throughput is equivalent to the probability of a successful transmission from i to j.

Therefore, the minimum-throughput link along the path from a source S to a destination D determines the throughput of a multi-hop path within the network. A node's throughput is often less than the sum of its outgoing link throughput; this is because in any given slot, a node uses at most one outgoing link to transmit data. Thus, the node throughput C_i is the probability of a successful transmission on any outgoing link from i.

If PUR_j is the probability that a neighbor j of i is awake in any slot, then the event A that at least one neighbor of i is awake in any slot has probability

$$P(A) = 1 - \prod_{j \in i.nbrs} (1 - PUR_j) \qquad (4)$$

Since every slot is assigned to i with probability $E[S_x]_i$,

$$C_i = E[S_x]_i * P(A) \qquad (5)$$

When offered load is high, the MAC maximizes throughput at the expense of increased energy consumption. On the other hand, when offered load is low, the MAC minimizes energy consumption at the expense of decreased throughput. These trade offs are necessary to achieve both high performance and low energy usage as network conditions change. The MAC determines how often nodes get to transmit and receive data by adjusting probability thresholds *PUT* and *PUR* respectively. The MAC adjusts a node's throughput by either increasing or decreasing the capacity on any one of the node's outgoing links subject to certain constraints. The *PUT* is efficiency and fairness constrained while the *PUR* is energy constrained. The capacity of a directed link between a node pair i and j can be increased by unilaterally increasing PUR_j. Higher *PUR* values increase the energy consumed by a node because the Hail radio listens for data in more slots. Unlike *PUR*, changing *PUT* values involves coordination with other nodes to ensure collision free transmission.

When offered load is high and every node has lots of data to send to all of its neighbors, the MAC adapts the *PUR* at each node to a maximum value of 1.0. Similarly, the distributed algorithm for allocating slots ensures that each node in a 2-hop neighborhood of size n gets $E[S_x] = 1/n$ of the slots available for unicast transmission. Given that each node has values of 1.0 and $1/n$ for *PUR* and $E[S_x]$ respectively, the resulting throughput at each node is $1/n$, which is asymptotically optimal. The throughput vs. energy tradeoff for low offered load is slightly complex because the goal is to minimize energy while maintaining reasonable throughput. When offered load is low, the MAC adapts *PUR* to very small values to conserve energy. Smaller *PUR* values decrease the energy consumed by a node because the Hail radio listens for data in fewer slots. In addition, smaller *PUR* values reduce node throughput as shown in (5). The reduction in node throughput is compounded by the restriction that nodes can only transmit data in assigned slots to ensure collision free access. One optimization is to relax the efficiency and fairness constraint when offered load is low and to allow nodes to send data in any slot to a neighbor that is awake in the same slot. In this case, $E[S_x] = 1.0$ and a node's throughput is only bounded by how often its neighbors are awake.

$$C_i = O\left(1 - \prod_{j \in i.nbrs} (1 - PUR_j)\right) \qquad (6)$$

Note that this is an upper bound on the achievable throughput since packet transmissions are no longer collision free. However, when offered load low, allowing nodes to send data in any slot to a neighbor that is awake is a reasonable optimization since collisions are less likely when nodes have very little traffic to send.

8.3 Synchronization

Energy is a premium resource in wireless ad hoc and sensor networks. Consequently, most energy-efficient MAC protocols use TDMA and attempt to conserve energy by duty cycling the receiver since the "on" time of the receiver usually dominates the total energy consumed by the node. However, the success of any energy conserving TDMA based MAC protocol is highly dependent on maintaining accurate synchronization between

neighboring nodes in the network. To achieve synchronization, some TDMA based MAC protocols in wireless networks either use time synchronization protocols with high synchronization message overhead or assume that nodes have access to an external time source such as Global Positioning Satellite (GPS). While GPS is very accurate and provides close to perfect timing, it is not an energy-efficient solution and it requires that nodes have access to the clear skies. Since one cannot always guarantee access to the clear skies for all network deployments, dependence on GPS is by no means desirable.

Synchronization is a continuous and not a one-time process because clocks have different oscillator frequencies that cause relative drift over time. Relative drift causes slot misalignment; resulting in collisions and significant loss of data. The other result of inaccurate synchronization in duty cycling networks is that nodes will wakeup to transmit and receive at the wrong time. This may continue indefinitely even under very low traffic load until the nodes run out of power. The following sections discuss some synchronization techniques applicable to wireless ad hoc and sensor networks.

8.3.1 Time Synchronization

Assuming that each node in the ad hoc network has a hardware clock that runs at the rate of real time with some bounded drift and a logical clock that can be updated based on its hardware clock and a set of external messages, time synchronization seeks to synchronize the logical clocks of all nodes in the ad hoc network. Several schemes exist for achieving time synchronization in wireless networks [11,14,17,21]. One example, RBS, is a scheme in which nodes send reference beacons to their neighbors using physical layer broadcasts. The key to using reference beacons is the assumption that the physical channel has a propagation delay close to 0. The RBS algorithm has two phases. In the first phase, a transmitter broadcasts a reference beacon to a set of receivers that record the arrival time of the reference beacon according to their local clocks. In the second phase, the receivers exchange their observations of the reference beacon to determine the relative phase offset between their clocks. Even though RBS achieves accurate synchronization to within a few microseconds, the additional message overhead for the second phase is considerably

high since it requires $O(n^2)$ messages to synchronize n nodes. Besides consuming energy, these synchronization messages may actually affect performance by reducing the throughput available to applications.

8.3.2 Slot Synchronization

This section discusses slot synchronization that is an alternative to time synchronization schemes such as RBS. In a TDMA network with fixed slot sizes, requiring time synchronization between all nodes may not be necessary. Instead, a weaker requirement, slot synchronization, should be sufficient. Slot synchronization has significantly less message overhead and is better suited for multi-hop network environments. The slot synchronization problem is different from the traditional time synchronization problem. While the latter aims at synchronizing logical time between all nodes in the network within a certain degree of accuracy, the former aims at synchronizing the slot boundaries of every node with its neighbors within some accuracy [13]. Thus, slot synchronization is mainly useful for MAC protocols and less suitable for application specific requirements such as distributed detection of timed of events. Instead of executing and then deriving slot synchronization from a time synchronization protocol, the MAC component in this research adopts an alternative approach that attempts to achieve slot synchronization directly. That is, nodes may have different hardware and logical clock readings or even slot numbers, but they all share a consistent view of when slots begin. The slot alignment property allows slot synchronization protocols to have simpler design, less synchronization message overhead, and faster synchronization convergence time.

8.3.2.1 Slot Alignment

Figure 11 shows how a node B aligns its slots to another node A upon reception of a packet from A. First, assume that each slot has a fixed size of d real time units. Also, assume that node A sends a packet to node B in real time T_A from MAC_A and node B receives the packet in real time T_B in MAC_B. When node B receives the packet in MAC_B, it

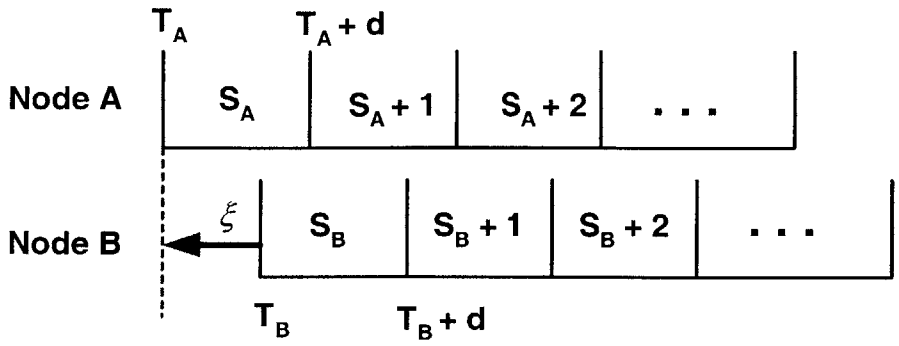


Figure 11: Node B aligns its slot boundary to Node A upon reception of a message from Node A. ξ is the slot boundary skew between A and B.

updates its logical clock L_B to its current hardware clock value H_B. Assume that $T_B - T_A = delay_{AB}$ is the delay from when the packet was sent from MAC_A to when it was received in MAC_B. Since T_A and T_B are unknown, MAC_B can alternatively calculate $delay_{AB}$ using delay timestamps. Assuming a 0 propagation delay, $delay_{AB} = delay_A + delay_B$ where $delay_A$ is the time taken for the packet to travel from MAC_A to PHY_A and $delay_B$ is the time taken for the packet to travel from PHY_B to MAC_B. $delay_A$ and $delay_B$ can be calculated using A and B's hardware clocks respectively. If PHY_A sends $delay_A$ to node B together with the packet from node A, MAC_B can easily calculate $delay_{AB}$. If A and B believe that the current slot number in real time T_A and T_B is S_A and S_B respectively, the goal is to align the beginning of slots S_A and S_B with respect to real time T_A. To achieve this, node B on receiving in slot S_B a packet sent from node A in slot S_A adjusts L_B such that $L_B - delay_{AB}$ corresponds to the beginning of slot S_B. If slots S_A and S_B are aligned in real time T_A, then $S_A + 1$ and $S_B + 1$ will also be aligned in real time $T_A + d$.

Assuming a 0 clock drift between nodes A and B, a one-time alignment of B's slot boundary to A's boundary would guarantee that $\forall\, i>0$, slots $S_A + i$ and $S_B + i$ will also be aligned in real time $T_A + i*d$. In reality, A and B's clocks are likely to drift further apart as i increases. Relative clock drift causes slot boundaries to misalign, thus requiring A and B to realign their slot boundaries periodically. The key to slot alignment is that nodes align their slots in real time using delay timestamps without synchronizing their clocks.

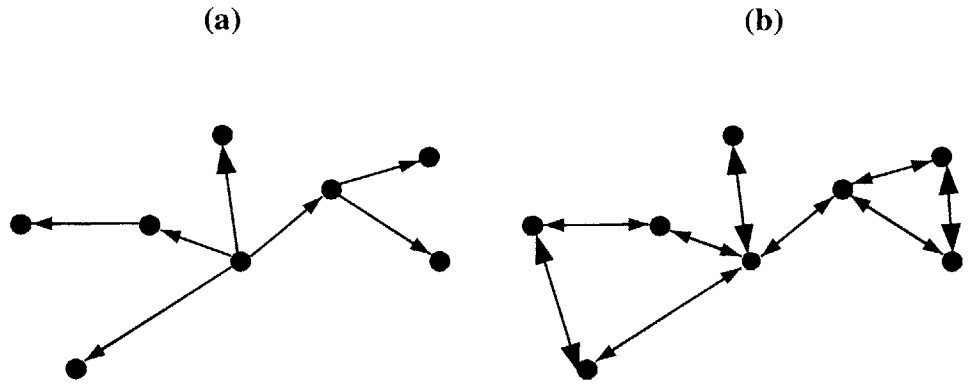


Figure 12: (a) Tree-based slot synchronization - the root node disseminates slot boundary information throughout the entire network (b) peer-to-peer slot synchronization- neighboring nodes apply a common filter function to converge at a common slot boundary.

8.3.2.2 Problem and Design Parameters

A definition of the slot synchronization problem is as follows: Assuming that nodes have bounded communication frequency and are equipped with hardware clocks that have bounded drift, slot synchronization requires that the communication frequency and other network parameters bound the worst-case slot boundary skew between two neighboring nodes. This section lists some design parameters for slot synchronization in duty cycling wireless ad hoc and sensor networks. It also describes a family of slot synchronization schemes for achieving the desired balance between accuracy, energy-efficiency, and fault tolerance. The family of slot synchronization schemes comprises tree-based and peer-to-peer schemes. In general, tree-based schemes converge fast with a high probability to within a certain accuracy bound while peer-to-peer schemes have nice fault tolerant properties desirable in tactical military networks. The slot synchronization schemes described here are energy-efficient because they do not generate additional messages and they depend solely on required message exchanges (heartbeats and or routing updates) between nodes to achieve and maintain synchronization.

Important design criteria for slot synchronization schemes include:

- Synchronization accuracy: The worst-case skew between the slot boundaries of any two neighboring nodes. Slot boundary skew directly affects the choice of slot

guard times. Smaller guard times reduce energy consumption because nodes turn their receivers "on" for a shorter duration.

- Synchronization convergence time: The time taken for all nodes in the network to achieve slot synchronization.
- Energy-efficiency: The number and frequency of broadcast messages needed to achieve and maintain slot synchronization.
- Fault-tolerance: The ability of the slot synchronization algorithm to handle node failures and the addition of new nodes to the network.

8.3.2.3 Tree-based Slot Synchronization

Tree-based slot synchronization schemes attempt to create and maintain a synchronization tree that is essentially a topological sub graph of the underlining network graph. This topological sub graph is often a rooted tree constructed dynamically by the slot synchronization algorithm. Once the synchronization tree is determined, the root node distributes slot boundary information along the edges of the tree via intermediate nodes to the leaf nodes. A node aligns its slot boundary to that of its predecessor node in the synchronization tree upon receiving a message from the latter. By near-optimally choosing the synchronization tree (e.g. Minimum Hop Level Spanning tree or its approximation, a Breadth First Search (BFS) tree) the worst-case slot boundary skew between neighbors can provably be reduced [13,16]. Synchronization trees can be constructed using Link State Updates or Heartbeat beacons.

1. Synchronization Tree using Link State Updates

This scheme uses cross-layer information between the Routing and MAC components to achieve slot synchronization between nodes in an ad hoc network. This scheme is not only simple and easy to implement, but it also energy-efficient because it achieves slot synchronization without introducing any new messages or additional overhead in addition to the underlining routing protocol overhead. A suitable routing protocol for this cross-layering scheme is a link state routing protocol. In link state routing protocols, nodes

flood information about the state of the links to their neighbors throughout the network. Every node receives a copy of the link state updates and independently creates a topology of the entire network. Since link state protocols send link state updates only when the state of a link changes, they tend to conserve network bandwidth and energy during periods of network stability. Besides energy and bandwidth conservation, some other nice properties of link state routing protocols useful for slot synchronization include avoidance of loop-forming routes, and minimal convergence time. Once the routing protocol has converged and each node has a consistent view of the network topology, the next step is to assign parents to each node in the network by constructing a shallow spanning tree of the network rooted at all sources. The shallowest such spanning tree can be used to optimally assign parents that minimize slot boundary skew from the root to any node. A child node aligns its slot upon reception of a message from its parent. The root node by definition has no parent and so it defines the slot boundary for the entire network. Computing multiple spanning trees may be computationally intensive so for average case topologies, constructing one BFS tree rooted at the most connected node should yield similar results.

2. Synchronization Tree using Heartbeat Beacons

In ad hoc networks that use routing protocols other than link state, it is possible to construct a synchronization tree using heartbeat beacons only. The algorithm described here uses heartbeat beacons to achieve slot synchronization and is similar to FTSP [28]. Here, a designated or elected root node drives the process of achieving slot synchronization within the network by means of periodic heartbeats that contain a sequence number. Only the root node increments this sequence number before sending a heartbeat. Each non-root node keeps track of the largest sequence number it has heard about and rebroadcasts this sequence number in its heartbeat. On receiving a heartbeat from a neighbor, a node compares its sequence number to that reported by the neighbor. If the neighbor reports a larger sequence number, the receiving node adopts the larger sequence number and aligns its slot boundary to that neighbor. Aligning slots to neighbors that report larger sequence numbers achieves slot synchronization along the edges of a minimum-delay spanning tree from the root to all leaf nodes. In this case, the

Table 1: Tradeoff Matrix for Slot Synchronization Schemes

Scheme	Advantages	Disadvantages
Tree-based	+ High synchronization accuracy for near-optimal sync-trees; especially if the root is chosen to be the graph center. + Energy efficient for near-optimal sync-trees because fewer nodes perform local broadcast. + Fast convergence. Once an optimal sync-tree has been constructed, slot alignment can begin immediately upon reception of a message from a predecessor node. + Reusability. In networks where node faults are rare, using an optimal sync-tree for maintenance reduces message overhead and saves energy. + Mathematical analysis is relatively easy.	- Generally not fault tolerant. Issues such as node failure, mobility, or even new nodes joining the network can perturb the existing tree and induce sync-errors. - Protocol overhead of maintaining a sync-tree can increase energy consumption. - Generally does not scale well with extremely large networks. - Distributed sync-tree construction and root election in large networks introduces additional protocol complexity.
Peer-to-Peer	+ Fault tolerant. Therefore, mobility or nodes joining and leaving the network has little effect on the synchronized state of the network. + No root election, tree computation or maintenance. Scalable to larger networks. + Relatively easier to implement. + Energy efficient because protocol overhead is kept to an absolute minimum.	- Average sync-error may be high depending on the network topology. - Longer convergence time. -Less synchronization accuracy. High synchronization accuracy can be achieved by increasing the frequency of local broadcasts; but this increases energy consumption. - Mathematical analysis and proving properties of optimal filter functions is difficult.

synchronization tree formed is extremely dynamic since is a modulated by the natural heartbeat rate of the nodes in the network.

8.3.2.4 Peer-to-Peer Slot Synchronization

This class of slot synchronization schemes explores the entire topological graph and do not impose a sub graph as a basis for slot synchronization. The goal remains the same for every node- to achieve consensus on slot boundaries with its neighbors. To achieve slot synchronization, a node maintains the current pair wise slot boundary skew between itself and its neighbors. It then applies a filter function (such as mean, median, max etc.) to this vector of values and then aligns its slot boundary according to the output of the filter. The peer-to-peer scheme described above may take a while to converge depending on the choice of filter function and network size. Unlike tree-based schemes, peer-to-peer schemes are robust and fault tolerant since there is no notion of a root node within the network. In addition, slot alignment can begin immediately upon reception of a message without waiting for the formation of a tree structure. Peer-to-peer schemes, on the contrary, have longer synchronization convergence time and they may require a higher broadcast frequency to synchronize every node in the network accurately (Table 1). In addition, it is likely that a particular filter function has good convergence properties in dense networks but poor convergence properties in other networks. The challenge with peer-to-peer schemes is figuring out filter functions that are appropriate for arbitrary network topologies and what the tradeoffs are.

8.3.2.5 Analysis

This section analyzes the worst-case slot boundary skew between neighboring nodes in the tree-based and peer-to-peer slot synchronization schemes described earlier. The worst-case slot boundary skew between neighboring nodes is particularly important because it determines the size of the chosen guard time between slots for the entire network. The synchronization model used for the analysis is similar to that proposed in [20]. The main characteristics of the model are that clocks have bounded drift and message exchanges are reliable. In addition, it assumes that uncertainty due to propagation and other operational delays are negligible compared to clock drift. In duty cycling networks where nodes rarely transmit data, this is a reasonable assumption since relative clock drift between nodes dominates clock skew. The model further assumes that

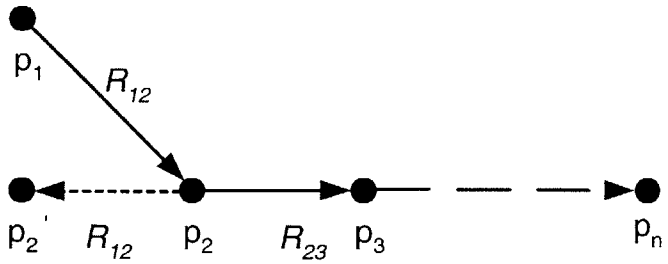


Figure 13: p_2 aligns its slot to p_1 without coordinating with p_3 thus increasing the slot boundary skew between p_2 and p_3.

nodes have bounded communication frequency. This means that a node communicates with each neighbor at least once every T time units. Lastly, the hop distance, d, is a distance measure between nodes and the network diameter, D, is the maximal hop distance within the network.

[15,20] prove that for any synchronization algorithm with a lower bound on the rate of increase of every node's logical clock, the clock skew between neighboring nodes is $\Omega\left(T \frac{\log D}{\log \log D}\right)$ where D is the maximal hop distance and T is a bound on the communication frequency between nodes. This fundamental result means that the clock skew between neighboring nodes is not a local property since it depends on the size of the network. This lower bound applies to slot synchronization since slot boundaries that are analogous to logical clocks increase at the rate of the hardware clock after a slot alignment. The tree-based and peer-to-peer slot synchronization schemes described in this report achieve a slot boundary skew between neighboring nodes that can be significantly larger than the proven lower bound. Both schemes presented achieve a worst-case slot boundary skew of $O(TD)$ between neighboring nodes. The actual slot boundary skew is larger because nodes align their slots boundaries without coordinating with all of their neighbors. The peer-to-peer scheme is likely to have better slot boundary skew on average since it involves some coordination although not explicit. Explicit coordination requires additional messages that minimize the slot boundary skew between neighboring nodes at the cost of energy-efficiency.

To prove the worst case slot boundary skew between neighboring nodes in the slot synchronization schemes described above, assume a line of n nodes $p_1 \ldots p_n$ where there is an undirected edge p_{ij} between nodes p_i and p_j $\forall\, i, j$ and $j\text{-}i = 1$ and p_j aligns its slot to p_i on receiving a message from p_i. Assume that R_{ij}, which is $O(T)$, represents the real time units that have elapsed since p_j last aligned its slot to p_i. Since p_j aligns its slot to p_i without coordinating with p_{j+1}, it is possible to create an execution where $R_{ij} > R_{jk}$ for k and $i=1...n$ *and* $j=2...n\text{-}1$. As a result, neighboring nodes p_i and p_j can have a slot boundary skew of $O(TD)$ as shown in Figure 13.

The worst-case slot boundary skew lasts for only $O(T)$ real time units and it implies that slot guard times need to be at most $O(TD)$ to guarantee collision free transmissions between neighboring nodes. However, larger guard times result in unnecessary idle listening at the beginning of each slot thus increasing the total energy consumed by a node. In reality, the expected slot boundary skew between neighboring nodes is smaller than the proven upper bound since the pseudo-random MAC protocol randomizes packet transmissions. Randomized packet transmission minimizes the probability of occurrence of a decreasing sequence of R_{ij}*'s* that result in the worst-case slot boundary skew.

9 System Design Details

This section describes detailed operation of each MAC subsystem and the interactions and interfaces between MAC subsystems as shown in Figure 2. In the actual implementation, each MAC subsystem runs as a separate thread that communicates with other threads via an Interprocess Communications (IPC) mechanism.

9.1 Outgoing Over the Air (OTA) Data

Outgoing OTA data originates from two places:

- Applications.
- Neighbor Discovery (ND) subsystem within the MAC component.

The following steps describe how the MAC component handles outgoing OTA data.

1. The Data Link Lower Layer (DLLL) looks at the next-hop and type of service (TOS) of the outgoing OTA data buffer to determine the appropriate queue to enqueue the buffer.
2. When a transmission slot comes up, DLLL peeks (obtains a reference without dequeuing) the appropriate queue for a packet to a next-hop that is listening within the same slot.
3. If a packet is found, DLLL updates the transmit status of that buffer to "transmitting", and sends the OTA data buffer to Mac State.
4. Mac State implements the appropriate MAC protocol (e.g. Hail-Hack-Data-Ack). The MAC protocol includes all or a subset of the following: waking up the Hail radio, sending a Hail, waiting for a Hail-Ack, waking up the Data Radio, sending the OTA data buffer, waiting for a Data-Ack, and finally putting the Hail and Data radios to sleep. Mac State sends all messages, including the Hail and the OTA data buffer to the appropriate PHY.
5. Mac State waits for the Data radio to respond with an "transmit status". The "transmit status" indicates whether the Data radio transmitted the OTA data successfully or not. Mac State forwards the "transmit status" to DLLL.
6. Depending on the "transmit status", DLLL will either release the OTA data buffer or leave it in the queue for re-transmission in the next available slot.

9.2 Incoming OTA Data

Incoming OTA data originates from only one place - the Data radio. Incoming OTA data traces the following steps within the MAC component.

1. DLLL wakes up at the beginning of the slot to listen for a packet from a particular neighbor. It informs Mac State that the slot is a receive slot.
2. Mac State then sends a command to wakeup the Hail radio. The command also directs the Hail radio to listen on a particular frequency during the Hail mini-slot.
3. If the Hail radio receives a Hail message, it sends it to Mac State.

4. If the received Hail indicates a broadcast packet or a unicast packet to the receiving node, Mac State enables the Data radio to receive the subsequent Data in the Data slot. If the sender requests a Hail-Ack, Mac State sends the Hail-Ack before enabling the Data radio to receive.
5. If the Data radio receives Data in the Data slot, it sends the received OTA data buffer to Mac State. Mac State sends the received OTA data buffer to DLLL and Acks the packet if the sender requests one.
6. DLLL looks at the header information in the received OTA data buffer to decide whether the data is either a heartbeat beacon for ND or a data packet for an application.

9.3 Data Link Lower Layer

The Data Link Lower Layer (DLLL) subsystem acts as the scheduler and OTA data buffer dispatcher within the MAC component. DLLL also runs a set of predefined rules on the wakeup schedules generated by Neighbor Discovery (ND) to determine the appropriate action for each slot. For example, if a slot is a unicast-receive slot of a neighbor and a node can transmit within the same slot, DLLL will retrieve the highest priority unicast packet to that neighbor and sends it down to Mac State for transmission. If a neighbor's requirements for slot allocation changes (by adjusting probability thresholds), ND provides DLLL with an updated wakeup schedule for that neighbor.

9.3.1 Detailed Description

More specifically, DLLL does the following:

- Receives schedules from the ND subsystem.
- Receives OTA data buffers from ND or applications and enqueues them on the appropriate next-hop queue.
- For each slot, decides whether to sleep, transmit, or receive a packet based on the wakeup schedules and slot rules.
- Accurately schedules the wakeup timer to wakeup on slot boundaries.

- Handles transmission of all OTA data buffers and if a transmission fails for some reason, it retries a configurable number of attempts.
- Analyzes received OTA data from Mac State and dispatches heartbeat beacons and other data to ND and applications respectively. For each received OTA data buffer, DLLL generates a link trace record using the in-band RSSI information provided by the PHY. DLLL sends the link trace record to LC.

9.3.2 Interfaces

The following interfaces exist between DLLL and other MAC components:

- *Receive an IPC containing wakeup schedules from ND.* This IPC contains the wakeup schedules for a node and all of its neighbors. Upon reception of this IPC, DLLL reschedules the wakeup timer to match the new schedules.
- *Receive an IPC containing an outgoing OTA data buffer.* This IPC contains the outgoing OTA data buffer that DLLL enqueues in the appropriate next-hop queue for later transmission.
- *Receive an IPC containing the transmit status of an outgoing OTA data buffer from Mac State.* This IPC contains the OTA data buffer with an updated "transmit status" field. Upon reception on this IPC, DLLL either retries or releases the buffer depending on the "transmit status".
- *Receive an IPC containing an incoming OTA data buffer from Mac State.* This IPC contains the incoming OTA data buffer.
- *Send an IPC containing a received OTA data buffer to either ND or an application.* This IPC contains the incoming OTA data buffer. DLLL sends ND heartbeat beacons to ND and all other OTA data buffers to applications.
- *Send an IPC containing a link trace record to LC.* This IPC contains the in-band information (e.g. RSSI) indicated by the PHY for all incoming OTA data buffers.
- *Send an IPC containing an outgoing OTA data buffer to Mac State.* This IPC contains the OTA data buffer data received from either ND or an application.
- *Send an IPC containing a receive command to Mac State.* DLLL sends this IPC to Mac State when it expects to receive data in a particular slot.

9.4 Mac State

The Mac State subsystem executes the low-level MAC protocol for transmitting and receiving OTA data. The low level MAC protocol typically involves both Hail and Data radios. When Mac State receives an OTA data buffer from DLLL, it implements the MAC protocol specified by DLLL for that OTA data buffer. The MAC protocol indicates whether to use a Hail, Hail-Ack, and or Data-Ack for a particular packet transmission. Mac State is also responsible for transitioning the radios to the appropriate state before and after each packet transmission or reception. Mac State sends all received OTA data buffers to DLLL. Note that data received from the Hail radio is processed and handled within Mac State, and is not sent up to DLLL.

More specifically, Mac State performs the following tasks:

- Implements the low level MAC protocol for transmission and reception of OTA data (i.e. Hail, Hail-Ack, Data, and Data-Ack) where Hail-Ack and Data-Ack are optional.
- Interfaces with the PHY component by implementing the PHY Interface Control Document (ICD) for communicating with both Hail and Data Radios.
- Controls and manages the state of the PHY component.
- Accepts outgoing OTA data buffers from DLLL.
- Sends incoming OTA data buffers received from the Data radio to DLLL.

9.4.1 Detailed Description

Figure 14 shows the low-level protocol for sending and receiving OTA data. Consider a node *A*, which has a packet for a next-hop node *B*. Node *A* first calculates the next wakeup slot, *x*, for node *B* (i.e. the next time node *B's* Hail receiver will be "on"). Node *A* then sleeps until slot *x* arrives (perhaps waking up in between to send packets to other nodes). In slot *x*, when *B* turns on its Hail receiver during the Hail mini-slot, *A* sends a Hail packet to *B*. Appendix B shows the packet format for the Hail data. The Hail radio

repeats the Hail data across a set of H tones, where H is typically a small value (e.g. 5). Repeating the Hail data across a set of tones increases the likelihood of delivering the Hail data even in the presence of frequency selective fading.

The source address field in the Hail packet is useful for situations where multiple Hails arrive at a destination node. In this case, the node sends a Hail-Ack to a particular node indicating which transmitter gets to send. The optional CRC field is useful for verifying the integrity of the Hail data. A node sets the "Want Hail-Ack bit" to verify that the receiving node is actually awake before it sends the OTA data buffer via the Data radio. The downside with requesting the Hail-Ack is that it consumes additional energy. The Hail-Ack also eats into the Data slot thus reducing the size of the available Data slot.

A node usually requests a Hail-Ack if:

- There has been a recent loss of Data or Data-Ack (i.e. the current OTA data buffer is a retransmission) to the destination node.
- There is mobility or other instability within the network.
- It has not heard from the destination node for over a long period.

If the sender requests a Hail-Ack and it fails to receive a Hail-Ack from destination node, the sender does not send the Data packet. Instead, the sender calculates the next time the destination node will be listening and it waits until that time to resend a Hail.

Appendix B shows the packet format for the Hail-Ack. The power level field in the Hail-Ack describes the amount of excess power received by the destination and therefore the sender decreases the transmit power of the OTA data packet by this amount. The excess power is also used to determine the transmit power for the Hail-Ack. The code rate describes the rate that the destination wants the sender to use for the following OTA data packet. If a Hail does not request a Hail-Ack, the OTA data packet immediately follows the Hail. Mac State sends the OTA data packet using the code rate and power specified in

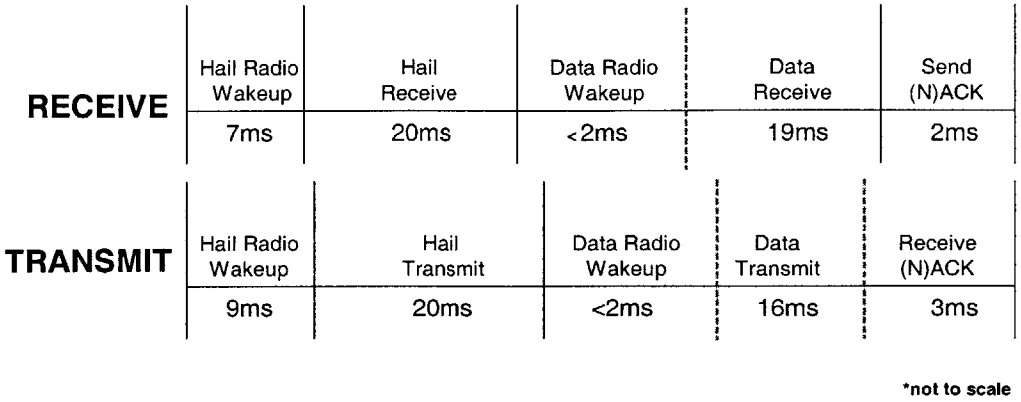


Figure 14: Low-level MAC protocol and slot timing (50 ms slot size)

either the Hail or the Hail-Ack. If the Hail does not request a Hail-Ack, Mac State uses the code rate and power specified in the Hail otherwise it uses the values in the Hail-Ack instead.

Appendix B shows the header format of the OTA data packet. Some fields of interest in the OTA Data packet header include:

- *Want-Data-Ack.* This bit indicates whether the sender wants the destination to acknowledge reception of the OTA data packet. If the Want-Data-Ack bit is set, the destination node immediately sends a Data-Ack upon reception of the Data packet.
- *DD Seq #, TOS, and the DD Reset.* DLLL uses these bits for duplicate detection of unicast packets.
- *Slot number.* As nodes do not have an agreement on an epoch or other global slot-numbering scheme, the slot number is useful for calculating slot offsets between nodes. A node can accurately predict the slot number of any node using its own slot number and the slot offset between them.

9.4.2 Interfaces

The Mac State subsystem interfaces to DLLL and the PHY as follows.

- *Receive an IPC containing an outgoing OTA data buffer from DLLL.* This IPC contains an OTA data buffer to the Data radio. The IPC also specifies the desired MAC protocol for the packet transmission.
- *Receive an IPC containing data from the PHY.* This catchall IPC handles all messages from the PHY component. The messages include responses to radio specific commands defined in the PHY ICD (e.g. Radio State Transition Acks, Transmit Acks, and Reset Acks etc.), Data and Data-Acks received by the Data radio, and also, Hail and Hail-Acks received by the Hail radio.
- *Send an IPC containing an incoming OTA data buffer to DLLL.* This IPC contains an OTA data buffer received by the Data radio.
- *Send an IPC containing the OTA data buffer transmit status to DLLL.* This IPC contains the OTA data buffer with an updated "transmit status" field.
- *Send an IPC containing data to the PHY.* This catchall IPC handles all messages to the PHY component. The messages include radio specific commands defined in the PHY ICD (e.g. Radio State Transition, Reset, Transmit, and Reboot etc.), Data and Data-Acks to the Data radio, and also, Hail and Hail-Acks to the Hail radio.

9.5 Neighbor Discovery

The ND subsystem performs the following general functions:

- Neighbor Discovery – Determines 1-hop neighbors and monitors the link-state to each neighbor.
- Wakeup Scheduling – Generates transmission and reception schedules.
- Slot Synchronization – Synchronizes slots between neighbors using a defined slot synchronization scheme.

The ND subsystem performs these specific tasks:

- Periodically broadcasts heartbeat beacons and scores heartbeat beacons received from each neighbor. The arrival time of received heartbeats is also used for slot synchronization.
- Manages the probability threshold, seed, and cycle position for unicast-receive *(PUR, SUR, CUR),* unicast-transmit *(PUT, SUT, CUT)* and broadcast-transmit *(PBT, SBT, CBT)* for a node and all its neighbors.
- Uses the Pseudo-Random Number Generator (PRNG) to generate the appropriate schedules for a node and all of its neighbors. The inputs to the PRNG are threshold, seed, and cycle position.
- Manages the desired frequency hop-sets for both Hail and Data radios.
- Uses the neighbor's score and the energy metric value reported by LC to determine the link status for each neighbor. ND reports the link status to the Routing component.

9.5.1 Detailed Description

Neighbor discovery is the process by which nodes in the network discover each other using periodic heartbeat beacons. A node maintains a neighbor table that has a separate entry for each node from which it has received a heartbeat. The node entry contains information about a particular neighbor such as probability thresholds, cycle position, and link status. ND uses a scoring scheme to determine the link status for each neighbor. The link status for a neighbor can have one of three values – "up", "down", and "attempting".

When ND receives a heartbeat from a node that is not already in its neighbor table, it creates an entry for the newly discovered neighbor. Subsequently, ND uses the neighbor's advertised probability thresholds and cycle position to generate a wakeup schedule for that neighbor. ND sends the wakeup schedule to DLLL so that DLLL knows when to wakeup to receive future heartbeats from this new neighbor. After updating DLLL with the neighbor's wakeup schedule, ND sets the neighbor's link status to "attempting". A link status of "attempting" means that ND is yet to determine whether

the link is usable for data transmission. ND continues to score the new neighbor while the link status remains in "attempting" using a simple scoring scheme. If the neighbor's score reaches a configurable "up" threshold, ND updates the link status from "attempting" to "up" and notifies the Routing component that the neighbor is a valid next-hop for data transmission. On the other hand, if the link status remains in "attempting" for a configurable and extended period, ND transitions the link status to "down" before removing the neighbor from its neighbor table. It is important that ND notify DLLL that the link status to the neighbor is "down" so that energy is not wasted waking up to receive heartbeats from that neighbor.

9.5.1.1 Sending Heartbeats

Each node in the network periodically sends heartbeats to its neighbors at a pre-configured interval. Besides sending periodic heartbeats, a node sends a heartbeat when it changes any of its probability thresholds. The contents of the heartbeat message are as follows:

- The unique node-id of the sender.
- Periodic heartbeat interval of the sender.
- Probability thresholds *(PUR, PUT, PBT)* and cycle position of the sender.
- *PUT* values are distributed up to 2-hops, so the heartbeat must contain the sender's neighbor list in addition to the *PUT* and cycle positions of each neighbor.

9.5.1.1.1 Scoring and Determining Link Status

ND maintains a configurable number of bins for scoring the link to each neighbor. The width of each bin is equivalent to the periodic heartbeat interval of that neighbor. Initially, ND initializes the neighbor's bins to zero points. When ND receives a heartbeat from a neighbor, the current bin for the neighbor accumulates a point. After each periodic heartbeat interval, ND assigns the neighbor a score that is equal to the number of non-zero bins before advancing to the next bin (possibly wrapping around) and zeroing its contents. If the score is greater than or equal to a configurable up score, the link status is

set to "up". Otherwise, if the score is less than or equal to a configurable down score, the link status is set to "attempting". A neighbor's link status transitions from "attempting" to "down" only if it remains in "attempting" for an extended period.

A link status cannot transition directly from "up" to "down" and vice-versa without going through "attempting". Note that ND relays all link status transitions to the Routing component. The Routing component only considers neighbors with link status "up" as valid next-hops. To ensure symmetry of links used for data transmission, it is important that each node verify the existence of its node-id in the neighbor's heartbeat before declaring the neighbor's link status to be "up" (in addition to the neighbor meeting the up score requirement). Without symmetric links, a transmitting node would not receive a Data-Ack or Data-Nack sent at the end of the slot.

In addition to scoring neighbors, ND uses the filtered Radio Frequency (RF) margin for each neighbor to determine the appropriate link status. Link Characterization (LC) uses the raw RSSI values provided by the PHY component to calculate the filtered RF margin. ND compares the measured margin against a configurable threshold and sets a neighbor's link status to "attempting" if the margin is greater than the threshold. In this case, the RF margin prevents ND from declaring links that are actually quite poor to be "up", even though enough heartbeats were lucky enough to get through. Similarly, if the margin is less than the threshold, ND sets the link status to "up". In this case, ND keeps links "up" even if heartbeats were lost due to collisions. ND passes the measured RF margin for each neighbor to the Routing component as a metric report. The Routing component uses this metric report for power control and for the selection of energy-efficient routes.

9.5.2 Interfaces

ND interfaces to the rest of the MAC component as follows:

- *Send an IPC containing wakeup schedules to DLLL.* This IPC contains the wakeup schedule for a node and all of its neighbors.

- *Receive an IPC containing a received heartbeat buffer from DLLL.* This IPC contains a heartbeat buffer received OTA.
- *Receive an IPC containing metric reports from LC.* This IPC contains a metric report (e.g. RF margin) for each neighbor.
- *Send an IPC containing neighbor down update to DLLL.* This IPC informs DLLL that a neighbor is down and that DLLL should no longer wakeup to either receive from or transmit to this neighbor.
- *Send an IPC containing neighbor reports to Routing.* This IPC contains the metric report for each neighbor in addition to the link status determined by ND.

9.6 Link Characterization

The Link Characterization (LC) component uses raw RSSI or other similar in-band information provided by the PHY component to generate a useful set of metrics for each neighbor. ND and Routing are the main consumers of LC metrics. ND uses the set of metrics generated by LC to determine the link status for a particular neighbor while the Routing component uses the metrics to choose energy-efficient routes for delivering packets.

9.6.1 Detailed Description

LC uses information provided by both Hail and Data radios to generate a summarized set of metrics for each neighbor. In addition to using LC metrics to choose energy-efficient routes, the Routing component uses LC metrics to determine the appropriate transmit power needed to close the link to a particular neighbor. ND also uses LC metrics in the case where ND has not received a heartbeat from a neighbor in a while and is about to update the neighbor's link status to "down". Prior to setting the link status to "down", ND requests a link profile status from LC. A link profile status indicates whether LC supports ND's decision to bring the link "down". LC sends updated metrics to ND and the Routing component on a configurable periodic basis except in the case where ND makes an explicit request for a particular metric. The metrics reported by LC include:

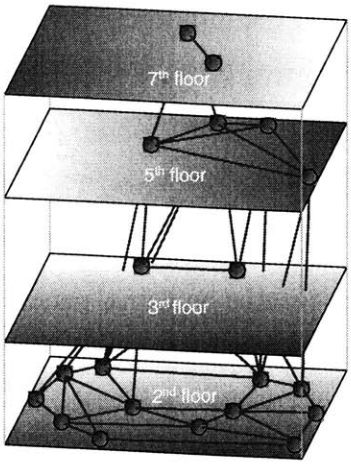
- RF Margin (dB) – This an estimate of the difference between the minimum required transmit power and the maximum amount of transmit power available, for the most robust data rate available from the radios. Routing uses this value to determine the minimum transmit power required to close the link to a particular neighbor. The RF Margin value is determined by considering packets which have been received from that neighbor, along with the transmit antenna pattern, receive antenna pattern, transmit power, receive sensitivity and other physical layer parameters.
- Link Profile Status – LC use raw RSSI values as well as the second order statistics of the RF Margin values to generate the link profile status. The result is integer value that indicates whether this link profile is "excellent", "good", "marginal", or "bad". Thus, the link profile status provides a subjective description of the link to a particular neighbor.

9.6.2 Interfaces

LC interfaces to the rest of the MAC component as follows:

- *Receive an IPC containing a link trace record from DLLL.* This IPC contains the raw information (e.g. RSSI) received in-band together with the OTA data buffer from the PHY.
- *Send an IPC containing metric reports to ND.* This IPC contains a metric report for each neighbor.

(a)



(b)

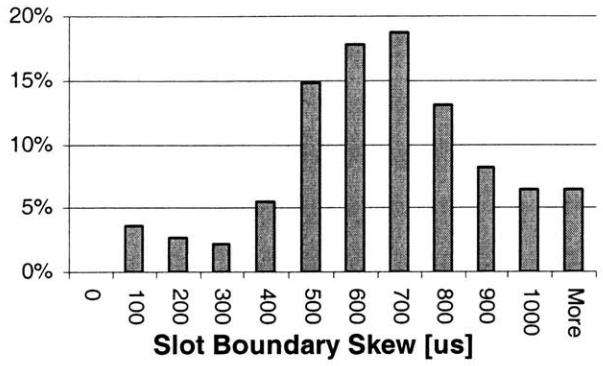


Figure 15: (a) Experimental setup and network topology (b) Histogram of the absolute slot boundary skew between neighboring nodes before synchronization.

10 Discussion of Results

This section discusses the performance results of an ad hoc network that implements that MAC and PHY components presented in this report. The experimental setup, as shown in Figure 15(a), is a 20-node multi-hop network with nodes located on the 2^{nd}, 3^{rd}, 5^{th} and 7^{th} floors of a multi-storey building. The network has a diameter of 5 with node degree ranging from 1 to 6. The performance metrics measured include synchronization accuracy, latency, and energy. The baseline MAC protocol for comparing the energy consumed by a node in the network is IEEE 802.11. MGEN is used to generate application traffic at rates of 0.001pps, 0.01pps, 0.1pps, and 1.0pps that correspond to network-wide duty cycles of $2x10^{-5}$, $2x10^{-4}$, $2x10^{-3}$, and $2x10^{-2}$ respectively. For a given duty cycle, each node generates MGEN application traffic with a Poisson distribution to a random destination within the network.

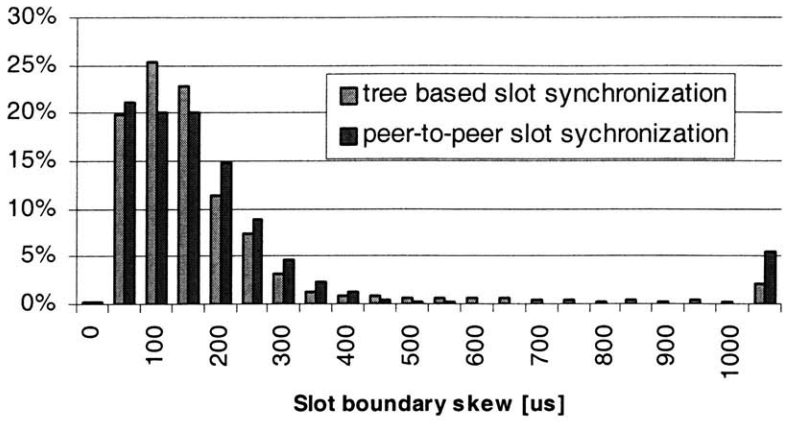


Figure 16: Histograms of absolute slot boundary skew between neighboring nodes in tree-based and peer-to-peer slot synchronization.

10.1 Synchronization Accuracy

Figure 15(b) shows a histogram of the slot boundary skew between neighboring nodes in the network before synchronization. When nodes are first powered on, they operate in asynchronous receive mode by keeping the Hail receiver "on" all of the time until they synchronize their slots to their neighbors. To measure the initial slot boundary skew, each node in the network broadcasts a heartbeat once every 30s and the slot boundary skew between nodes is recorded each time a node receives a heartbeat from a neighbor. As shown in the Figure 15(b), without any synchronization, the slot boundary skew between neighbors can be arbitrarily large- up to the slot size of 50ms.

To reduce the slot boundary skew between neighbors, the performance of two energy-efficient slot synchronization schemes presented earlier are examined. The schemes are tree-based slot synchronization using heartbeat beacons and peer-to-peer slot synchronization using a median filter. The network parameters remain the same as before. That is, each node generates periodic heartbeats at a rate of once every 30s and the slot size is 50ms with a guard time of 2ms. In addition, each node is equipped with a hardware clock that has a worst-case drift of 10ppm.The slot boundary skew is recorded for all received packets but nodes only synchronize on heartbeat packets. Figure 16 shows histograms of the slot boundary skew between neighboring nodes for both

synchronization schemes. After convergence, 95% of all received messages recorded a slot boundary skew of 0-300us in both schemes and 98% of all received messages recorded a slot boundary skew of less than 1ms which is well within the 2ms guard time. Given a slot guard time of 2ms, a node can begin duty cycling the Hail radio when the slot boundary skew between the node and all of its neighbors is very small (~ 500 us).

As shown in Figure 16, the peer-to-peer scheme has less variance than the tree-based scheme but it also recorded more samples that are multiple standard deviations away from the mean. These extreme samples show that it took a longer time for the peer-to-peer scheme to converge as compared to the tree-based slot synchronization scheme. In addition, the choice of a median filter means that it is likely that a small fraction of nodes have difficulty achieving very accurate synchronization because of their location in the network topology. The tree-based slot synchronization scheme, on the other hand, converges quickly and achieves very good accuracy even though it has a slightly higher variance than peer-to-peer synchronization. The higher variance is because it takes a variable amount of time for slot boundary updates from the root to propagate throughout the entire network. Even though the peer-to-peer slot synchronization scheme has a relatively higher synchronization convergence time, the performance results indicate that peer-to-peer slot synchronization compares well accuracy wise with tree-based slot synchronization once it has converged. Moreover, it is possible that there exist filter functions besides the median filter that converge faster and improve upon the performance of peer-to-peer slot synchronization.

It is important that when nodes are initially powered on or when nodes try to join established networks, they listen asynchronously for at least the maximum heartbeat interval of the network to discover other nodes and to synchronize their slots before going into duty cycling mode. If a tree-based slot synchronization scheme is in use, nodes joining and leaving the network may temporarily disrupt the synchronized state of the network due to the election of a new root or the assignment of new parents. On the other hand, if a peer-to-peer scheme is in use, nodes can join and leave the network with minimal impact on the synchronized state of the network.

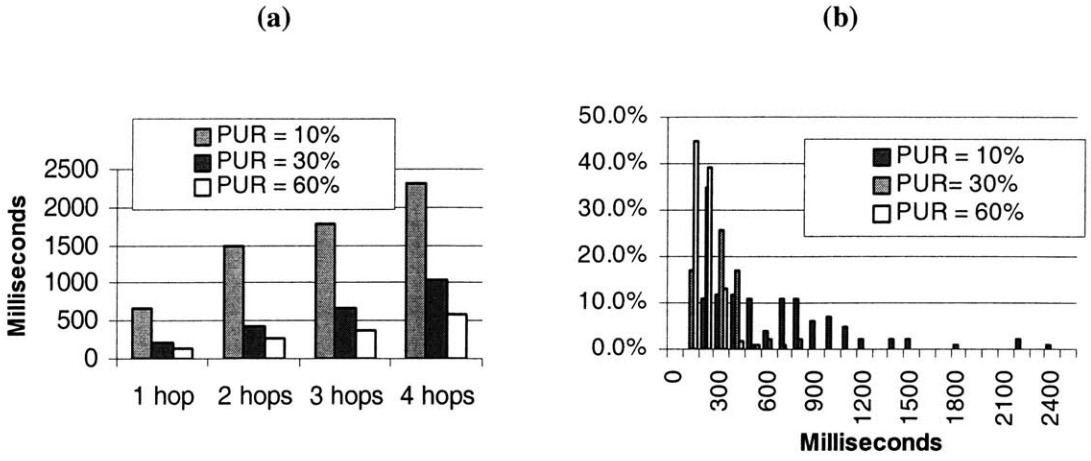


Figure 17: (a) Average end-to-end latency experienced by application packets within the network for increasing hop distance and increasing unicast receive probability threshold (b) Histogram of end-to-end latency over a single hop. The majority of application packets arrive in less than 500ms if the Hail radio listens 60% of the time.

Peer-to-peer slot synchronization is more robust and fault tolerant because there is no single node that determines the slot boundary for the entire network. On the battlefield, this is a very important feature because "blowing" up the root node in a tree-based slot synchronization scheme requires the election of a new root. The root election process may involve sending additional messages that increases the energy consumed by nodes.

10.2 Latency

This section discusses the end-to-end latency experienced by application packets within the network. For a given duty cycle, each node in the network generates application traffic to a random destination node within the network. The end-to-end latency is measured from the time an application packet arrives at the source MAC to the time when it is successfully delivered to the destination MAC. Figure 17(a) shows the relation between average end-to-end latency and hop distance. It is not surprising that the average end-to-end latency increases with hop distance. Figure 17(a) also shows the trade off between *PUR* and latency. Increasing *PUR* increases the energy consumed but it also reduces the average end-to-end latency (Figure 17(b)) because nodes listen for data in more slots. The MAC adapts *PUR* values to achieve a good balance between latency and energy consumption.

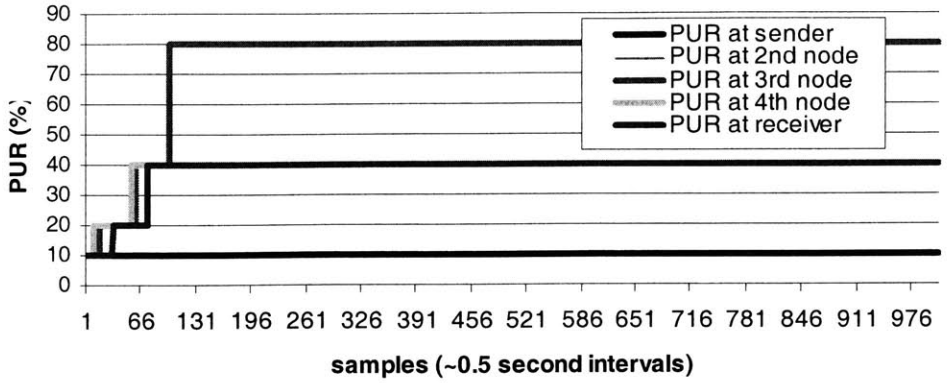


Figure 18: Adapting PUR values along a multi-hop path from sender to receiver.

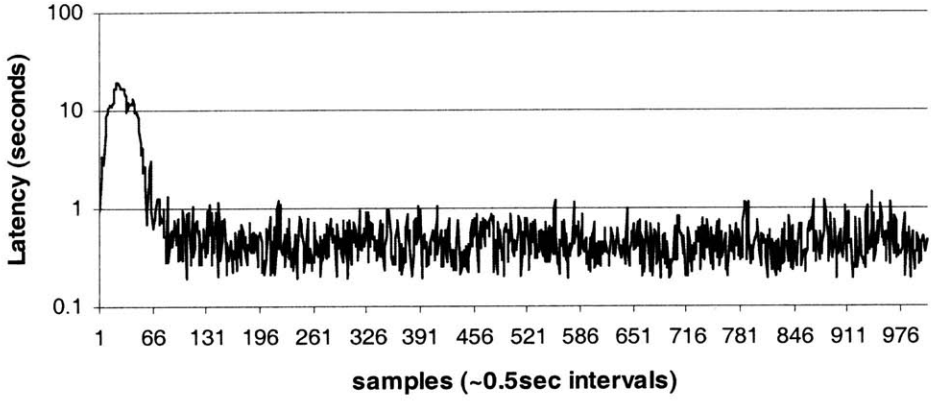


Figure 19: Adapting PUR improves latency – only the first few packets experience significant delay.

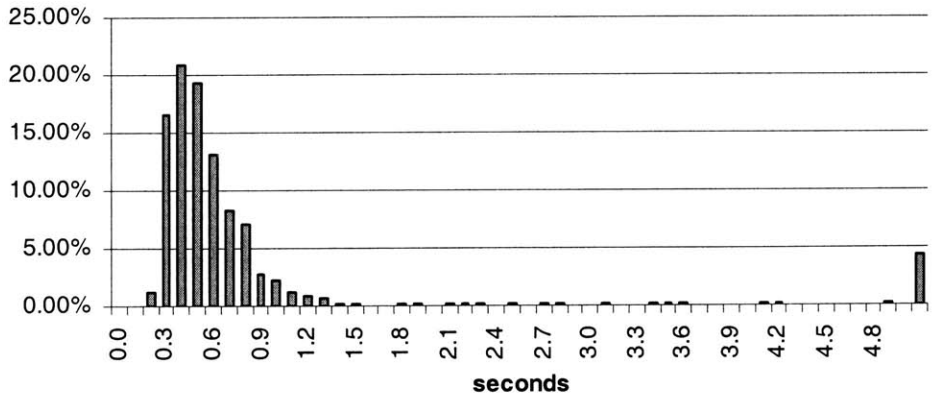


Figure 20: Histogram of packet latencies.

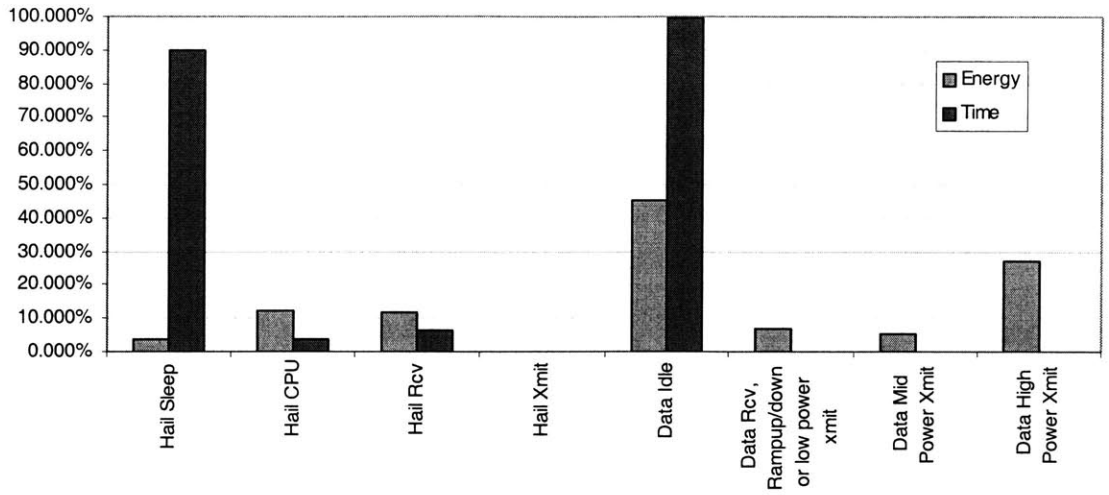


Figure 21: Energy and time for each radio state ($2x10^{-3}$ duty cycle). Data Idle includes energy consumed by the transceiver clock.

Figure 18 shows how the MAC adapts *PUR* values in an idle network that suddenly begins to generate traffic. Nodes along a multi-hop path from the source to destination adapt their *PUR* values but the source need not adapt its *PUR* value. As shown in Figures 19 and 20, the initial packets from the source experience a longer delay because the next-hops are asleep most of the time. As more packets arrive at the next-hops, the MAC adapts by increasing *PUR* so that Hail radio listens in more slots. Increasing a node's *PUR* increases the capacity of the link to the node and later traffic to the node experience very little delay.

Sometimes, it possible for a node to have data to a next-hop that is awake to receive data in very few slots. In order to reduce the latency of these data packets, the OTA data header includes a *MoreData* bit that allows a node to request that the next-hop increase its *PUR* to a maximum (i.e. 1.0) in order to receive more data. This allows nodes to make quick transitions from sleeping in almost every slot to listening in every slot. A node sets the *MoreData* bit in the header of a packet to a next-hop when the queue depth to that next-hop exceeds a configurable threshold. This feature is also useful for an application data stream that has high priority and cannot wait for MAC to adapt.

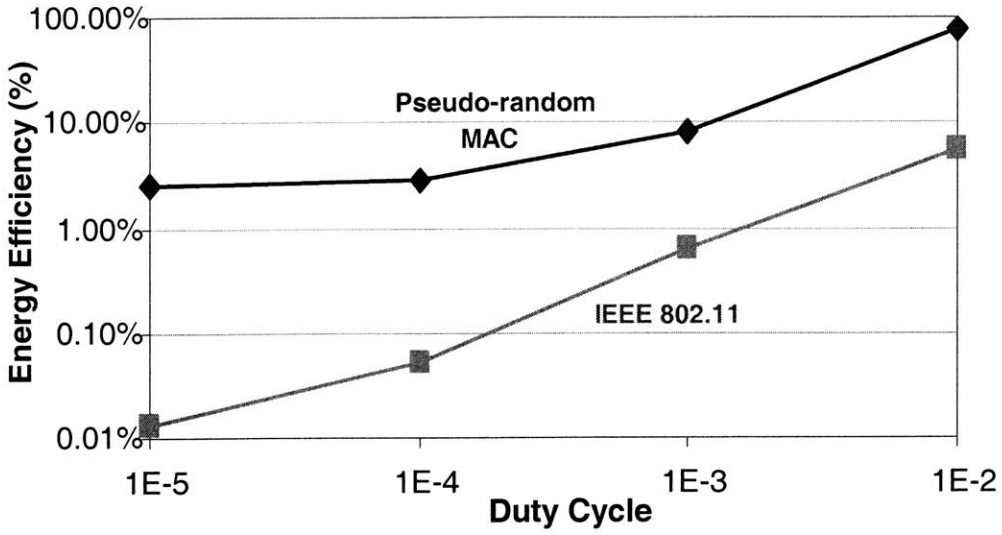


Figure 22: Energy-efficiency per node.

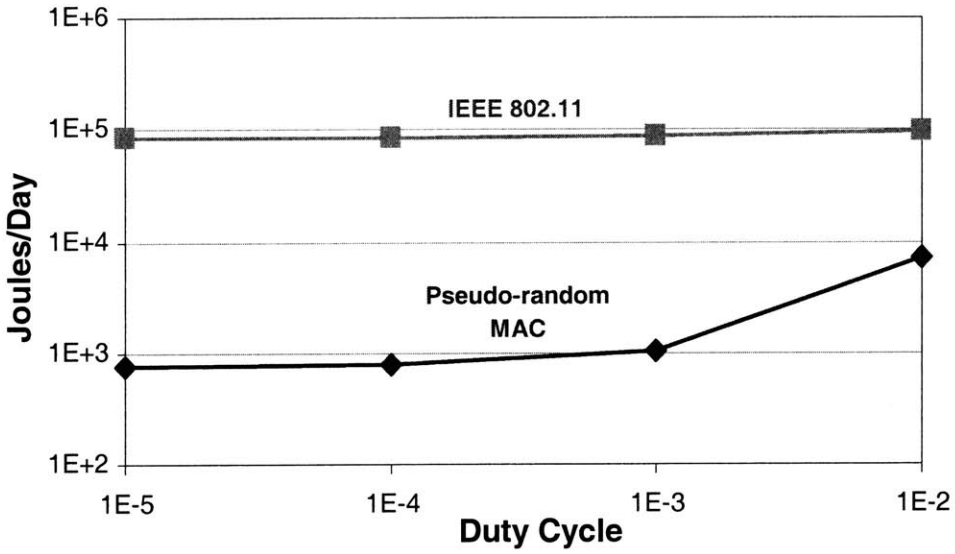


Figure 23: Daily energy usage per node.

10.3 Energy

This section discusses the energy gains of the duty cycling MAC presented in this thesis vs. 802.11. The energy metrics of interest are the total energy consumed by a node and energy-efficiency. Energy-efficiency is the ratio of energy used for application data to the

total energy consumed by the node. Each node in 802.11 uses about 100KJ daily irrespective of how much data it sends (Figure 23). This is because in 802.11 the radio receiver idle listens most of the time, which dominates the total energy consumed by the node. In the duty cycling MAC, the Hail and Data radios can only be in one of four states- sleep, idle, transmit, and receive. To measure the total energy consumed by a node in the network, the total time spent by the Hail and Data radios in each state is recorded. The product of the time spent in each state and the power consumed in each state is the total energy consumed by a node in the network. As shown in Figure 21, at low duty cycles, most of the time and energy is spent sleeping. Figures 22 and 23 summarize the energy gains of the entire system. Each node consumes a measly 1KJ per day at low duty cycles that is ideal for sensor networks.

11 Conclusions and Future Work

This thesis presents the design and implementation of an ad hoc network system that is energy-efficient under most traffic loads and is able to adapt as network conditions change. The system achieves a 100X reduction in the total energy consumed as compared to IEEE 802.11 for low offered loads. To put this energy reduction in perspective, the proposed system would last 33 days on two standard AA batteries while IEEE 802.11 would only last 7 hours. The MAC protocol used is TDMA based so it is inherently energy conserving and the lack of contention overhead guarantees that the system achieves high channel utilization and does not collapse under high loads. The MAC protocol tackles the difficult problem of dynamic slot allocation and synchronization of schedules between nodes in an energy-efficient manner. Some key ideas employed include using compactly represented pseudo-random seeds and probability thresholds to generate uncorrelated wakeup schedules between nodes and maintaining accurate synchronization between nodes using a slot synchronization scheme with minimal message overhead. In addition, the MAC protocol assigns slots using a distributed slot allocation algorithm that is fair and provides link level *QOS*. Furthermore, the MAC provides deterministic access to the shared medium while supporting high data rates required for real-time traffic as well as delay-bounded services such as voice and multi-media streaming. Experimental and simulation results show that the ad hoc network

system presented in this report achieves improved performance and dramatic reduction in energy consumed for a wide range of scenarios. Areas for future work include:

- *Dynamic Slot Allocation:* Explore the feasibility of other distributed slot allocation algorithms that are energy-efficient, guarantee collision free access, and provide additional link level *QOS* that is flexible enough to support explicit reservation of slots to particular destinations. Additional work is also needed to explore better schemes for adapting probability thresholds as network conditions change.
- *Slot Synchronization:* Explore new filter functions that improve upon the performance of peer-to-peer slot synchronization schemes by achieving faster synchronization convergence time. Alternatively, explore hybrid schemes that use a tree-based scheme to achieve initial synchronization and then use a peer-to-peer scheme to maintain accurate synchronization. Extend slot synchronization algorithms to do some regression analysis on measured slot boundary skew. Regression analysis is useful for post-facto synchronization where nodes continue to synchronize their slot boundaries for an extended period without exchanging messages. Finally, slot synchronization is a relatively new concept and additional theoretical analysis is needed in that regard.
- *Transport Protocols:* Existing transport protocols (TCP and UDP) originally designed for wired networks are not suitable for duty cycling ad hoc wireless and sensor networks where nodes are often asleep. New transport protocols designed specifically for duty cycling networks can provide additional performance improvements and further reduction in energy consumed by nodes using techniques such as in network caching, local recovery, and per flow *QOS*.
- *CPU Energy Savings:* Most of the energy savings in this system is achieved by keeping both radios off when they are not in use and by limiting the transmit power of the high power Data radio where possible. The system achieves additional energy savings by keeping the number of protocol messages to an absolute minimum. Further work in this area will involve exploring the option

of actually turning off the CPU running the MAC and going into hibernation mode during long periods of inactivity. Such a feature will allow deployment of sensor networks months in advance of when they are actually needed. Dynamic clock frequency modification or "under clocking" of the CPU is another energy conservation technique that is worth exploring.

12 References

[1] J. Redi, I. Casineyra, C. Partridge, K. Manning, R. Rosales-Hain, R. Ramanathan, and S. Kolek, "Joint Architecture Vision for Low Energy Networking (JAVeLEN) – DARPA-ATO Connectionless Networking Program, Phase 1", *BBN Technical Report 8408,* December 2004.

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

[3] G. Lu, B. Krishnamachari, and C. S. Raghavendra, "An Adaptive Energy-Efficient and Low-Latency MAC for Data Gathering in Wireless Sensor Networks," *Int. Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN),* SantaFe, NM, April 2004.

[4] L.F.W. van Hoesel, and P.J.M Havinga, "A lightweight medium access protocol (LMAC) for wireless sensor networks: Reducing Preamble Transmissions and Transceiver State Switches," *First International Conference on Networked Sensing Systems,* Tokyo, 2004.

[5] T. van Dam, and K. Langendoen, "An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks," *Proc. of the 1st international conference on Embedded networked sensor systems*, 2003.

[6] J. Redi, and D. Avresky, "Performance of Energy-Conserving Access Protocols Under Self-Similar Traffic," *Proc. of IEEE Wireless Communications and Networking Conference,* New Orleans, LA, September 1999.

[7] V. Rajendran, K. Obraczka, and J.J Garcia-Luna-Aceves, "Energy-Efficient, Collision-Free Medium Access Control for Wireless Sensor Networks," *Proc. of the 1st international conference on Embedded networked sensor systems*, 2003.

[8] J.Redi, "Energy-Conserving Protocols for Wireless Data Networks," *Boston University, Ph.D Thesis,* 1998.

[9] D. Chiu, and R. Jain, "Analysis of the Increase Decrease Algorithms for Congestion Avoidance in Computer Networks," *Journal of Computer Networks and ISDN,* Volume 17, Number 1, June 1989.

[10] D. Bertsekas, and R. Gallager, "Data Networks," *Prentice-Hall, New Jersey*, 1992.

[11] J. Elson, L. Girod, and D. Estrin, "Fine-Grained Network Time Synchronization using Reference Broadcasts," *5th Symposium on Operating Systems Design and Implementation (OSDI),* Boston, MA, December 2002.

[12] R. Rozovsky, and P.R. Kumar, "SEEDEX: A MAC protocol for ad hoc networks," *Proc. of The ACM Symposium on Mobile Ad Hoc Networking and Computing, MOBIHOC,* 2001.

[13] L. Dai, P. Basu, and J. Redi, "Energy Efficient Slot Synchronization Techniques for Wireless Sensor Networks," *BBN Technical Report 8429,* 2005.

[14] T. Locher, and R. Wattenhofer, "Oblivious Gradient Clock Synchronization," *20th International Symposium on Distributed Computing (DISC),* Stockholm, Sweden, September 2006.

[15] L. Meier, and L. Thiele, "Gradient Slot Synchronization in Sensor Networks," *Proc. of the twenty-fourth annual ACM symposium on Principles of distributed computing,* Las Vegas, NV, July 2005.

[16] H. Attiya, D. Hay, and J.L. Welch, "Optimal Clock Synchronization under Energy Constraints in Wireless Ad-hoc Networks," *9th International Conference on Principles of Distributed Systems,* Pisa, Italy, December 2005.

[17] R. Fan, I. Chakraborty, and N. Lynch, "Clock Synchronization for Wireless Networks," *Principles of Distributed Systems: 8th International Conference, OPODIS,* Grenoble, France, December 2004.

[18] G.P. Halkes, T. van Dam, and K.G. Langendoen, "Comparing Energy-Saving MAC Protocols for Wireless Sensor Networks," *Mobile Networks and Applications,* Volume 10 , Issue 5, October 2005.

[19] K. Langendoen, and G. Halkes, "Energy-Efficient Medium Access Control," *Embedded Systems Handbook,* Embedded Systems Handbook, CRC Press, 2004.

[20] R. Fan, and N. Lynch, "Gradient Clock Synchronization," *Proceedings of the Twenty-third Annual ACM Symposium on Principles of Distributed Computing,* St. Johns, Newfoundland, Canada, July 2004.

[21] T.K. Srikanth, and S. Toueg, "Optimal Clock Sychronization," *Journal of the ACM (JACM),* Volume 34 , Issue 3, July 1987.

[22] P. Varbrand, and D. Yan, "Maximal Throughput of Spatial TDMA in Ad Hoc Networks,"

[23] B. Shrader, M. Sanchez, and T.C. Giles, "Throughput-delay Analysis of Conflict-free Scheduling in Multihop Ad-hoc Networks,"

[24] M.J. Miller, and N.H. Vaidya, "Minimizing Energy Consumption in Sensor Networks Using a Wakeup Radio," *IEEE Wireless Communications and Networking Conference (WCNC),* 2004.

[25] I. Chlamtac, C. Petrioli, and J. Redi, "Energy-conserving access protocols for identification networks," *IEEE Transactions on Networking,* February, 1999.

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

[27] J. Gronkvist, "Assignment Methods for Spatial Reuse TDMA,"

[28] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, "The Flooding Time Synchronization Protocol," *Proc. of the 2nd international conference on Embedded networked sensor systems,* 2004.

[29] Mark Stemm, and Randy H Katz, "Measuring and reducing energy consumption of network interfaces in hand-held devices," *IEICE Transactions on Communications,* August 1997.

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

13 Appendix A Hail Transceiver Options Spec. Sheet

Hail Transceiver Core "System on a Chip"		MICRF501	MICRF500	CC1010
Freq. Range	(MHz)	300- 600	700- 1100	300- 1000
Tx RF Power (to 50-ohm/antenna)	Min (dBm)	-20	-20	-20
	Max (dBm)	12	10	10 to 4 (vs F)
	Power Control (dBm)	3 incr.	3 incr.	1- 3 incr. (vs F)
Rx Sensitivity (dBm @ BER exponent = -3)		-105	-104	-107(2)
RSSI Dynamic Range	(dB)	60	60	55
Tx DC Power(1) (from Battery) @ Max	(mW)	135	150	150
Rx DC Power(1)	(mW)	24	30	30
Standby (Oscillator-Only) DC Power (1)	(mW)	0.9	0.9	0.1- 0.3 (vs F)
Transceiver Off (Leakage)	(mW)	<0.006	<0.006	<0.006
Oscillator Start-Up Time	(ms)	5 (est.)	5 (est.)	2
Rx Settling Time	(ms)	1	1	0.5(4)
Rx-Tx Switch Time	(ms)	2(3)	2(3)	0.25(3)
Max. Hop Rate	(hops/s)	250	250	1000
Max. Bit Rate (BFSK Modulation w/ Manchester)				
	Chip-Internal Modulator (b/s)	2400	2400	600- 76.8K
	Chip-External Modulator (b/s)	128K	128K	--
	ACI Reject Ratio @ 200 kHz Hop Spacing (dB)	45	45	--
	@ 1 MHz Hop Spacing (dB)	57	57	43

(1) includes all chip-external components
(2) @ 2400 b/s
(3) in Standby
(4) programmable (performance trade-off)

Figure 24: Overall specification comparison of Micrel and Chipcon single-chip transceivers.

14 Appendix B Packet Formats

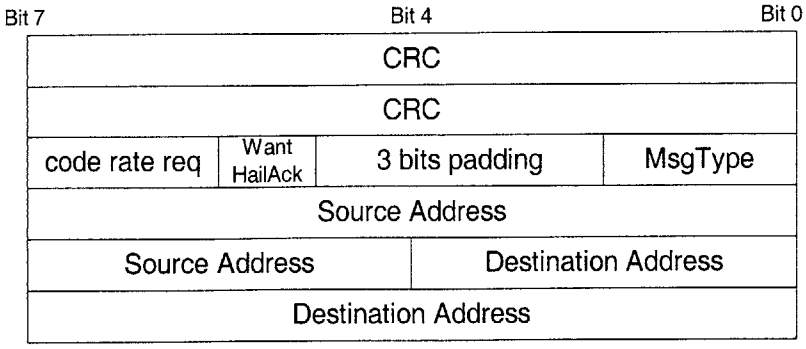


Figure 25: Hail Packet Format.

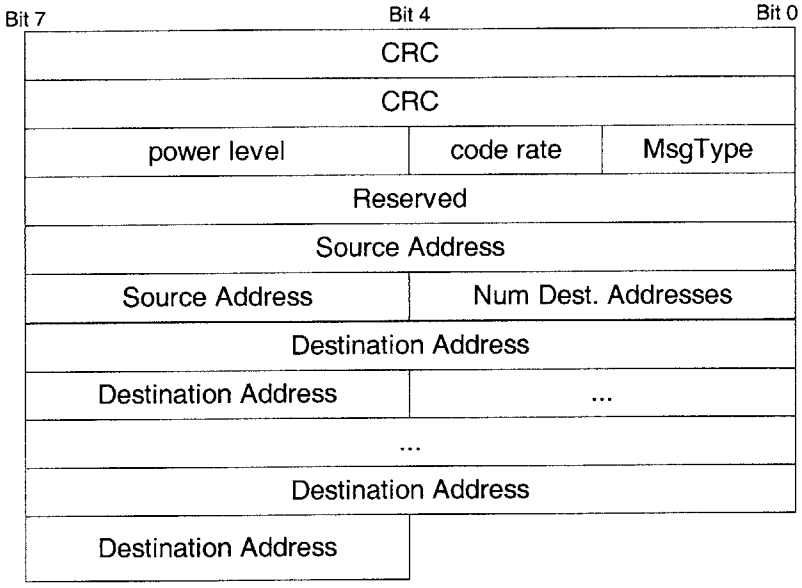


Figure 26: Hail Ack Packet Format.

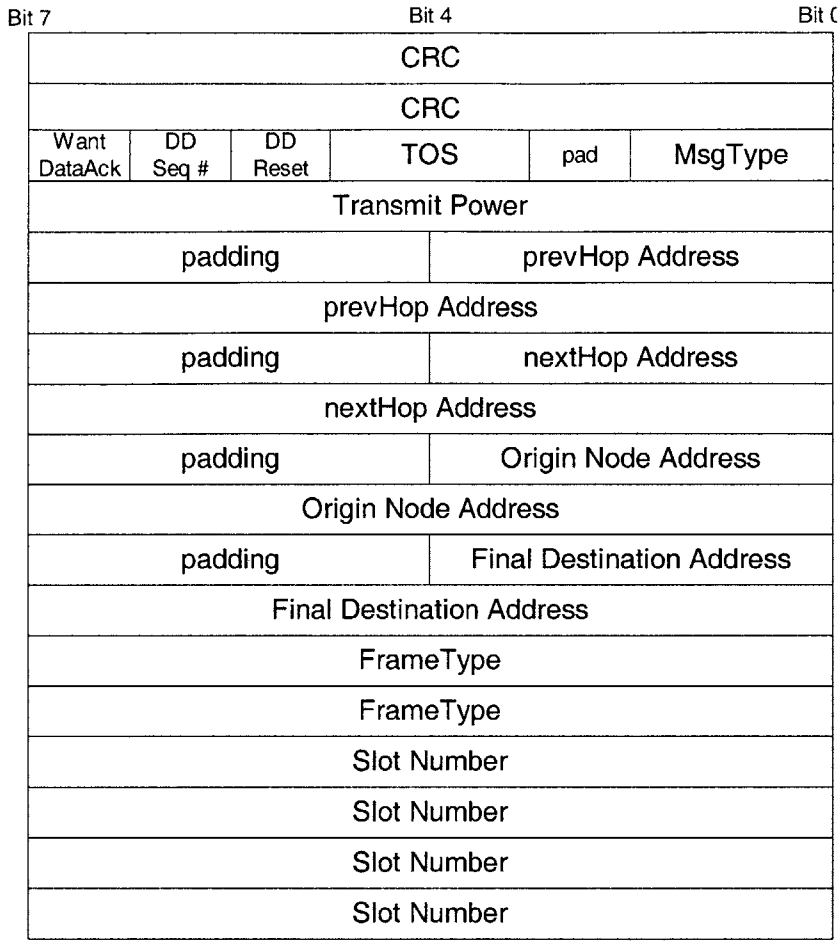


Figure 27: Data Packet Header Format.

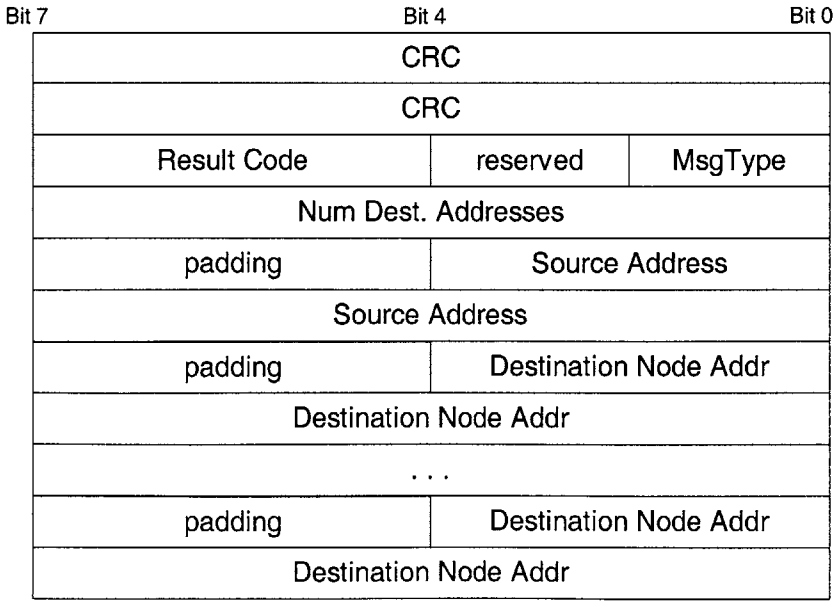


Figure 28: Data Ack Packet Format.