A Measurement-based Analysis of the Interaction among MAC, Network and Application Layers in Wireless Sensor Networks

by

Umberto Malesci

Submitted to the Department of Electrical Engineering
and Computer Science
in partial fulfillment of the requirements to the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2005

Author .....
Department of Electrical Engineering and Computer Science
May 6, 2005

Certified by .....
Samuel Madden
Assistant Professor
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

A Measurement-based Analysis of the Interaction among MAC, Network and Application Layers in Wireless Sensor Networks

by

Umberto Malesci

Submitted to the Department of Electrical Engineering and Computer Science
on May 6, 2005 in partial fulfillment of the
requirements to the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

There have been a number of recent proposals for link and network-layer protocols in the sensor networking literature, each of which claims to be superior to other approaches. However, a proposal for a networking protocol at a given layer in the stack is typically evaluated in the context of a single set of carefully selected protocols at other layers, as well as a particular network topology and application workload. Because of the limited data available about interactions between different protocols at various layers of the stack, it is difficult for developers of sensor network applications to select from amongst the range of alternative sensor networking protocols. This thesis attempts to remedy this situation by evaluating the interaction between several protocols at the MAC and network layers and measuring their performance in terms of end-to-end throughput and loss on a large, real-world TinyOS and Mica2 mote-based testbed. We report on different combinations of protocols using different application workloads and power-management schemes. This thesis analyzes the effects of various services provided by the different protocols, such as link-level retransmission, neighborhood management, and link-quality estimation. Our analysis suggests some common sources of poor performance that developers may experience during real-life deployments; based on this experience, we propose a set of design principles and lessons for the designers of future interfaces and services in TinyOS.

Thesis Supervisor: Samuel Madden
Title: Assistant Professor

Acknowledgments

This thesis is the result of joint work with Sam Madden. I have to thank Sam for trusting me and allowing me to be part of his research group. Sam has been a fantastic source of help and encouragement in the development of the work behind this thesis. A great teacher, advisor and friend, Sam steered my research efforts with his clear and pragmatic vision. Moreover, this thesis is written in correct English only thanks to Sam's careful and continuous revision of my work.

My thanks go also to the MIST group at CSAIL, in particular to Bret Hull, Kyle Jamieson, Stan Rost and Thomer Gil, who developed, maintained, constantly improved and shared with me the mote testbed at CSAIL.

In the last year I have also been bothering many people in the sensor network community across the country. I am particular indebted to Mark Yarvis at Intel, Wei Ye and John Heidemann at USC, Joe Polastre at UC Berkeley. Thanks for your patience, help and for answering so promptly to my e-mails.

Dennis McLaughlin and Harold Hemond deserve special thanks for their support and for the opportunity they gave me to work for the Civil and Environmental Engineering Department.

I also would like to thank everybody who helped me through the course of my five years in the US; in particular my thanks go to Paolo Fresco for trusting me in the first place, Gene Bickers for his help and support in transferring to MIT, Anne Hunter for guiding me through the Institute's bureaucracy and regulations, Simon Chiavarini for advising me and introducing me to career paths beyond engineering. Last but not least,

my parents deserve my thanks for their support and help, and for accepting my desire to go abroad for such a long time.

My final thanks go to all my friends who made my "American" years in college and graduate school so much fun. In particular, Christian, Lorenzo, and Stefan in LA and Luigi, Marco and Roberto in Boston made many nights in town really memorable! Special thanks go also to Niccolò who kept me updated every week with the news from Florence. Finally, Marta and Cosimo deserve a special place in here for listening and advising me on all sorts of issues. Thanks for bearing with me.

This thesis is dedicated to anybody who deserved to be mentioned in here but I inevitably forgot.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Sensor networks consist of collections of tiny sensing devices, commonly referred to as "motes" (i.e., a speck of dust) – Figure 1-1 shows an example mote. Motes collect information using application-specific sensors and transmit data by means of a low-power radio. These motes are built to be deployed in both indoor and outdoor environments where sensing at a high level of granularity is necessary. Motes are usually battery-powered devices with very limited hardware resources to process or store data. Data is gathered by using sensors that attach via a daughter card and then transmitted to the base station. With the help of a multi-hop routing protocol, data packets are forwarded from one node in the network to the next until they reach their final destination. In order to save energy and be able to work on a single set of batteries for several months at a time, motes must remain in a low-power state, turning off their radios and processors when there is no other processing to be done.

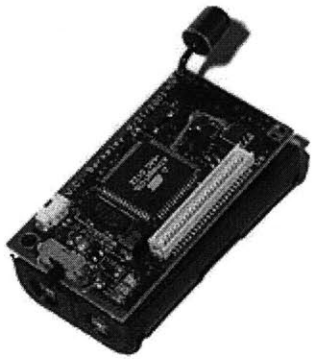

Figure 1-1: Crossbow Mica2 Mote

Sensor networks have already been deployed in several real-life scenarios. For example, hundreds of wireless sensing devices were deployed on Great Duck Island, off the coast of Maine, during the summer of 2004. The objective was to study the behavior of the Leach's Storm Petrel (Figure 1-2), a rare bird that nests on the island every year [12]. Other environmental monitoring projects involve monitoring volcanic eruptions in central Ecuador [2], or measuring microclimatic variations in humidity, temperature and pressure in botanical gardens [1] and vineyards [24].

Figure 1-2: Leach's Storm Petrel

Other applications of sensor networks involve industrial monitoring and supply chain management. Tens or hundreds of sensors can be deployed in harsh industrial environments to continuously monitor equipment and issue alerts in response to malfunctions. For example, Intel is currently working with BP (British Petroleum) in a pilot project to monitor vibrations on BP's oil tankers [25].

However, many of these recent field experiments have demonstrated that current generations of sensor networks are still too immature to be easily deployed in many

long-term real-world scenarios. In particular the deployment of sensor networks in field studies requires the integration of several networking technologies. Current sensor networking technology is such that this integration is very difficult, because protocols have subtle incompatibilities with each other. To-date, many real-life deployments have taken a monolithic approach to communication stack design, with all communications protocols being developed by the same research team or organization. These monolithic radio stacks typically lack features that can be obtained from elements of competing stacks. In this thesis, I study the interaction between several communication protocols developed by different research groups and the subtle and undocumented incompatibility issues that make sensor network programming very difficult.

1.1 Motivation

The sensor network community has proposed a large number of different networking protocols for routing [14, 21, 22], media access [4, 5, 8, 16, 17, 23], and power management [6, 8, 16, 17]. Despite significant innovation in each of these areas, there has been little work addressing the integration of protocols across areas into a single application. Published papers typically propose changes in one abstraction (e.g., a new MAC layer) while using some "default" implementation of the other abstractions (e.g., using the "standard" multi-hop routing protocol in TinyOS) and evaluating on a particular topology and a particular workload. This approach has led to a large number of competing protocol proposals which are difficult to compare with one another due to varying choices made by authors about appropriate defaults, application workloads, and network topologies. This makes it hard for an application designer to select the best set

of protocols for his or her application and impossible for other researchers to understand whether claimed differences between protocols are simply due to artifacts in one experimenter's setup or are true differences between protocols.

This thesis focuses on the TinyOS [7] operating system, because source code for many different protocol implementations is widely available and because it appears to be the currently platform of choice for sensor network research. In TinyOS, problems surrounding the interactions between protocols are aggravated by significant disagreement in the community about how functionality should be spread across different network 'layers'. Lines between layers are blurred, making innovation difficult and mixing and matching of implementations tricky as interfaces are poorly specified. As an example, consider per-link acknowledgements and retransmissions, which are widely regarded as an important feature for reducing loss in sensor networks. Different designers implement these features in different parts of the network stack. In some cases (e.g., B-MAC [17]), acknowledgements are synchronous, meaning they are transmitted by the receiver immediately after a packet is received, without re-acquiring the channel; in other cases, they are asynchronous (e.g., S-MAC [8, 16]) and treated just like regular network packets. In the synchronous case, acknowledgements must be issued by the link/MAC (media access and contention) layer [8, 16, 17] which is responsible for negotiating low-level access to the network channel. In the latter case, they may be issued by the link layer or by a high-level network layer that is also responsible for forwarding packets to some end-to-end destination [13, 14]. Similarly, retransmissions due to negative or failed acknowledgements can be implemented at either of these layers, as can duplicate suppression. Placement of these operations is not

consistent across implementations; for example, the B-MAC [17] protocol implements acknowledgements but assume that retransmissions and not duplicate suppression should be implemented in the layers above, whereas the S-MAC protocol implements all three. If a researcher wants to design a new 'network' layer, he must choose which of ACKs, retransmissions, and duplicate suppression to implement. These choices will invariably tie his implementation to a particular MAC layer and limit the generality and impact of his work; worse yet, due to unstated assumptions in various implementations, she may believe her protocol will work with a particular MAC layer only to find it does not. If the abstraction boundaries between layers were more cleanly implemented and specified, these issues would not arise.

This thesis presents a systematic study of the performance (in terms of network loss rate) of different combinations of MAC, routing, forwarding, and power management protocols that have been previously proposed in the literature. The aim of this work is to provide measurements that will enable a first step towards fixing these problems with the network protocols in TinyOS by:

1. Benchmarking several widely published protocols with the same application workload and on the same network topology.
2. Highlighting the significant differences between different implementations protocols that are ostensibly at the same 'layer'.
3. Illustrating a number of examples where interactions between different layers lead to significant performance degradation.

4. Recommending particular combinations of protocols for particular application workloads.
5. Illustrating that no one protocol at any of these layers strictly dominates any other protocol (despite claims to the contrary in the literature.).

The purpose of this thesis is not to proscribe a specific layering or to suggest that one implementation is better than another. Rather, this thesis aims illustrate some of the limitations of the current state of software in TinyOS so that the community can move towards a cleaner set of interfaces that support greater protocol diversity and allow application developers to make more informed choices about the appropriate selection of networking protocols.

Chapter 2

Sensor Network Architecture

A common architecture and set of abstractions has emerged for the network stack for wireless sensor networks. As in most network architectures, the basic abstraction is the layer. However, the TinyOS network stack differs from the traditional Internet stack in several ways:

- Layers make abundant use of cross-layering in order to increase throughput and decrease power consumption [10].
- Power management is present in many different forms in several layers.

The network stack in TinyOS can be broken into four major layers: the physical layer, the link/MAC layer (to keep consistent with the naming using in many publications, we refer to this simply as the MAC layer in the remainder of this thesis), the forwarding/routing layer (we refer to this as the routing layer in the remainder of this text), and the application layer. Moreover, the network stack also performs other two major services that are not present in Internet routing: power management and link-quality estimation.

This thesis mainly focuses on the analysis of the interaction between MAC layer, routing layer, and power management taking into consideration the application-specific requirements.

2.1 MAC Layer

As far as the Medium Access Control (MAC) protocol is concerned, the major distinction is between the use of TDMA or CSMA to negotiate channel access. This thesis focuses on CSMA-based implementations, because, although several TDMA protocols have been proposed [4], they are not available in implementations that are easily integrated with existing multi-hop forwarding and routing protocols.

The different MAC protocols developed for sensor network also differ in the additional services that they provide. For example, some MAC protocols, such as S-MAC [8, 16], also perform link level acknowledgment and retransmission and hidden-terminal avoidance via RTS/CTS. Moreover many MAC protocols also take care of power management for the entire network stack [8, 16, 17]. However, the needs of the application layer are not always considered when the MAC protocol takes care of power management. Often the application requires the mote to wake up and sense the environment at specific times and rates that are not the same as the forwarding needs of the communication stack.

2.1.1 Contention-based approaches

Contention-based approaches rely on protocols that detect and avoid packet-collisions. Packet-collisions are detected by means of acknowledgment packets and can be partially avoided using RTS/CTS (Request-To-Send/Clear-To-Send) packets as well as

exponential back-off on resend when a collision occurs. The major advantages of contention-based MAC protocols are:

- No need for a very precise time-synchronization between the devices [8].
- Flexibility in terms of both changes in the network topology and changes in the data-rates.

On the other hand, the major disadvantages of contention-based MAC protocols are related to the possible contention for the medium and to the lack of innate energy management mechanisms:

- In collision-detection based CSMA protocols (used in sensor networks), collisions increase the number of retransmissions and decrease channel capacity.
- Use of RTS/CTS packets to avoid collisions or per-link acknowledgment packets to determine if a collision occurred.
- Difficult power management. Inserting a duty cycle in CSMA-based protocols requires both time-scheduling and time-synchronization. Without such mechanisms, motes always listen to the channel, which tends to waste power receiving messages directed to other motes [5].

Power management is often performed in the MAC layer and there are several contention-based protocols designed for sensor networks that introduce duty-cycles in order to save energy [5, 7, 8].

2.1.2 TDMA-based approaches

Though this thesis does not experimentally evaluate TDMA, we briefly surveys its merits because it has some attractive properties that potentially make it useful in sensor networks. In TDMA, a single radio-frequency channel is divided into multiple time slots. Each time slot is assigned to a different device (e.g. mote) that requires access to the medium. The two main strengths of the TDMA-based approach are that it eliminates packet collisions since every device transmits during a different time slot and that it provides a built-in power management scheme since devices only need to be "on" in slots when they are sending or receiving data [4].

The major disadvantages of TDMA-approaches in sensor networks are that it requires precise time synchronization and provides a fixed, inflexible bandwidth allocation to every node.

In addition to pure TDMA, Hohlt et al [6] propose a hybrid solution, where transmission is scheduled using time slots, but medium contention and packet collisions are still possible. This hybrid approach reduces medium contention by assigning different time slots to different clusters of motes, while simultaneously avoiding complexities in terms of time-synchronization that a full TDMA solution would require. However, the current implementation only allows a one-way communication toward the root of the routing tree and it therefore cannot be deployed with many data-collection applications, such as TinyDB [9], that need broadcast and multicast communication as well.

2.2 Routing Layer

Multi-hop protocols are widely used in sensor networks because, by multi-hopping, nodes are able to send data-packets beyond the transmission range of a single low-power radio.

In tree-based multi-hop routing (Figure 2-1), every node in the network keeps a neighborhood table containing the best neighbor nodes that it can use to route packets toward the root of the tree. A node has a parent that it uses to route packets toward the root; the routing protocol decides which neighbor node should play this role and when it should switch parents. The most recent multi-hop protocols base routing decisions on the number of hops and on the estimated link quality [14, 15, 21, 22].

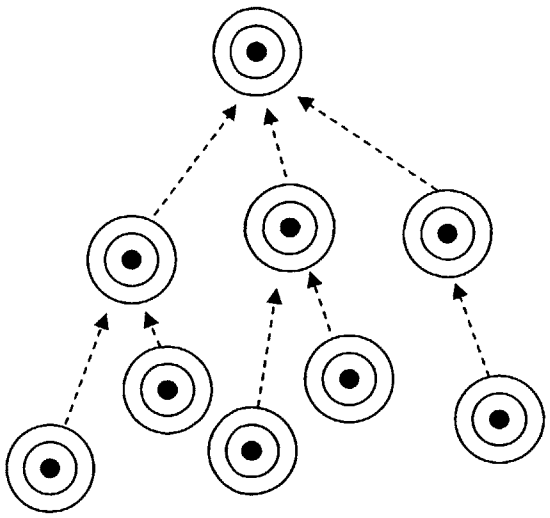

Figure 2-1: Multi-hop tree-based routing topology

Some services such as link-level retransmission and link-quality estimation are performed in the routing layer in some implementations and in the MAC layers in other implementations. This irregularity regarding the placement of key services inside the

network stack makes designing a new, general-purpose implementation of a specific network layer difficult, as the layer will have to make assumptions about the services provided by other layers that will invariably tie it to one set of accompanying implementations. For example, some routing protocols rely heavily on snooping packets addressed to other nodes [14]. Therefore, power management schemes implemented in the MAC layer that save energy by decreasing idle listening will break when used with such routing protocols.

This tight and non-standardized correlation between the services required by the routing protocol and the services provided by the underlying MAC layer make many implementations difficult to reuse across multiple implementations; chapter 4 and chapter 5 illustrate a number of such problems that exist in practice today. Chapter 6 discusses some design options for remedying these problems.

2.3 Power Management

Many sensor network applications require the network to survive several months or years in the field without changing batteries. To achieve such lifetimes, it is necessary to keep devices in a low-power ("sleeping") state most of the time. The devices wake up only when they need to perform a particular task, such as sensing or receiving and transmitting data. This duty cycling needs to adapt to different application-specific data rates and to variable network sizes.

Power management is often performed in the MAC layer and there are several contention-based protocols designed for sensor networks that introduce duty-cycles in order to save energy [5, 8, 16, 17]. Some protocols, such as B-MAC, implement a power

management scheme (called "low-power listening" by the B-MAC designers) that increases packet preamble length (and hence transmission costs) but decreases idle-listening costs by requiring nodes to listen for packet transmissions only once period preamble period. Note, however, that this approach does not solve the "problem" of nodes consuming power listening to packets addressed to other nodes.

Other protocols, such as S-MAC [8, 16] or T-MAC [5], employ a listen/sleep duty-cycle, creating a cluster of neighboring nodes that share the same schedule and therefore wake up and go to sleep all at the same time.

Another commonly used power management scheme for sensor nets, used in the TASK/TinyDB system [1], works at the application layer and divides the time into two frames; an active frame and a sleeping frame. In the active frame, motes transmit and route the data-packets that were queued during the sleeping frame. Due to the large size of the two frames, this approach does not need very precise time-synchronization. However, it is not very flexible because the size of the frames is fixed and does not adapt to either the network size or to the traffic characteristics.

Chapter 3

Studied Implementations

This thesis focuses on routing, MAC and power management protocols that have been implemented using the TinyOS platform. Unfortunately only a small percentage of all the protocols that have been proposed in the literature have a mature and robust enough implementation that can be integrated and used on a real test bed. Therefore, in this thesis we had to limit the research to two different MAC protocols (B-MAC [17] and S-MAC [8, 16]) and three different routing protocols (/lib/Route, MINTRoute and ReliableRoute [14]). Both B-MAC and S-MAC implement a power management scheme integrated with the MAC layer, therefore in all the experiments changing MAC protocol meant also changing power management scheme. Within each protocol variant, this thesis looks at performance differences with and without power management.

3.1 MAC Protocols

B-MAC and S-MAC are the most mature and widely used MAC protocols within the TinyOS project. They are both based on packet-collision avoidance schemes and they both integrate a power management scheme within the MAC protocol. However, the

greatly different in their architecture, in the additional services they provide, and in the techniques they use to achieve energy-efficient operations.

3.1.1 B-MAC

The standard TinyOS MAC protocol is a contention-based protocol called B-MAC [17]. As discussed above, B-MAC provides power management via low-power listening; the "recommended" preamble length in B-MAC is 100ms [17], which is the value used here. B-MAC has been shown to outperform other MAC protocols in previous studies [17], and has been carefully tuned for the radio used in Mica2 motes like those in our deployment.

On Mica2s, B-MAC supports synchronous acknowledgments that require only a few extra bit-times on the end of each packet to transmit. This depends on the ability of the sender and receiver to quickly switch roles at the end of a packet transmission and remain synchronized before any additional sender can sense an idle channel and begin transmitting.

B-MAC does not perform link-level retransmission or hidden terminal avoidance using RTS/CTS schemes. The designers assume that such protocols will be implemented at higher layers if necessary.

In B-MAC, every mote overhears every packet transmitted; this allows high-layer network protocols to employ snooping for link-quality estimation [14] and in-network processing [9].

The designers of B-MAC do not assume that a single set of parameters will work with every possible application and they do not try to make B-MAC agnostic of the protocols that run above it. Instead, B-MAC offers control to the protocols that sit on top

of it, allowing to the routing and application layers to change parameters like the low-power listening duration or the number and type of retransmissions used. The idea is to enable cross-layer optimization without imposing a particular API on end users. Unfortunately, as far as we know, most implemented routing layers do not make use of this cross-layer tuning API.

3.1.2 S-MAC

S-MAC [8, 16] is a contention based MAC protocol that adds into the MAC layer power management, link-level retransmission, duplicate packet suppression, hidden terminal avoidance using RTS/CTS, and link-quality estimation.

The power management scheme is based on shared schedules between "neighborhoods" of small groups of motes. Some motes follow more than one schedule simultaneously and therefore are able to forward packets from one neighborhood to another.

The S-MAC power management scheme tries to minimize energy consumption by decreasing the overhearing of other motes' transmissions. Motes sleep when they detect a transmission not addressed to them, and neighborhoods deactivate during times when they are not actively scheduled.

Unlike B-MAC, S-MAC does not provide any way for the higher layers to change its MAC parameters but assumes, as it happen for Internet routing, that every layer can be completely separated and independent from the others.

3.2 Routing Protocols

The three different routing protocols tested differ in terms of routing algorithm, metrics and services provided. /lib/Route seeks to minimize the number of hops that each packet traverses. On the other hand, MINTRoute and ReliableRoute route packets based on link-quality estimates that seek to maximize the probability of a packet being delivered.

3.2.1/ lib/Route

Route was the standard routing protocol in TinyOS 1.1.0; it has been supplanted by MINTRoute. Route performs link-quality estimates but bases routing decisions mainly on hop count, using link-quality estimates simply as a threshold to prune very low quality links.

3.2.2 MINTRoute

MINTRoute is the new standard routing protocol for TinyOS. Unlike /lib/Route, MINTRoute bases its routing decisions mainly on link-quality estimates rather than minimum hop count. The quality estimates for sending and receiving are used to select a parent that will minimize the expected number of transmissions to reach the root of the network. The literature [1, 14, 15, 21] reports better performance using link quality estimates rather than minimum hop count. Moreover, MINTRoute add to /lib/Route a topology stabilization mechanism in order to avoid frequent parent switching.

MINTRoute's design and implementation involves several hidden assumption that makes it inappropriate under certain conditions. For example, MINTRoute assumes the capability to snoop every neighbor's packets. This makes using MINTRoute with a

MAC that doesn't conform to this specification (such as S-MAC) problematic; we discuss this issue further in chapter 5.

3.2.3 ReliableRoute

ReliableRoute uses the same routing algorithm as MINTRoute but implements link-level retransmissions. However, it does not implement duplicate suppression; to provide valid end-to-end throughput results, I implemented a duplicate suppression algorithm for purposes of our experiments. By default ReliableRoute performs up to 5 link-level retransmissions. In the experiments we decreased the maximum number of retransmissions to 3 in order to achieve consistency with S-MAC which also performs up to 3 retransmissions by default.

ReliableRoute performs packet retransmissions based on acknowledgement information that it expects the MAC layer to provide and therefore does not work with any MAC protocol, such as S-MAC, that does not conform to this specification.

3.3 Application Workload

All the experiments use a workload similar to that of many environmental monitoring applications like TinyDB [9] or Surge (a simple TinyOS application designed to collect readings from all motes in a network at a fixed rate).

Environmental monitoring applications, broadly speaking, have the following characteristics and requirements:

- Data rate: Low and Fixed
- Longevity of the networks: Long

- Data-delivery reliability: Low
- Topology: Mostly static and many-to-one (Tree)
- Acceptable latency: high

As this thesis is mainly focused on sensor networks for data collection, we use simply a workload of this type. In particular, we use different routing and MAC layers under the Surge application.

Chapter 4

Evaluations and Experiments

In this chapter, we study the performance of the different MAC and routing protocols under different application workloads. All our results are based on experiments run on the 61-node Mica2 (Figure 1-1) [18] and Cricket [19] test bed at MIT CSAIL. The test bed (Figure 4-1) is installed on the 9th floor of the Gates Building in the MIT Stata Center and spans an area of 16,076 square feet. Although both types of motes are able to run the same executables, in order to ensure consistent results, we run only on the Mica2 motes, therefore using only 46 out of 61 motes. Each Mica2 mote has an Atmel ATmega128L microcontroller with 4 KB of RAM, 128 KB of flash, and a CC1000 radio running at 433 MHz that modulates at 38.4 symbols per second using Manchester encoding. Moreover, each mote was attached to a Crossbow MIB6000 Ethernet interface board. The board provides power to the mote and allows remote reprogramming and data collection via Ethernet. We managed and controlled the entire test-bed though a Motelab [3]-based web interface.

In all experiments, each mote was running the same version of Surge (TinyOS 1.1.0 release [7]). Surge transmits sensor readings at a fixed rate. We run the experiments at several different data rates; this thesis reports here results where we send one packet

every 10 seconds and one packet every 60 seconds, representing a high-load and a low-load scenario.

One packet every 10 seconds creates a high-contention situation even with no power-management scheme enabled. Polastre et al. [17] show that a 20-node Mica2 network can deliver (in aggregate) about 16 packets per second when running B-MAC. With 46 nodes each sending 1 packet every 10 seconds, with an average of node-depth of 3, the entire network sends 13.5 packets per second, which is close B-MAC maximum throughput capabilities. Following the trends shown by Polastre et al. [17], we expect our 46 node network to have somewhat less throughput than a 20 node network.

At a rate of 60 seconds per packet, we generate 2.25 packets per seconds on the network, clearly below the B-MAC throughput limits.

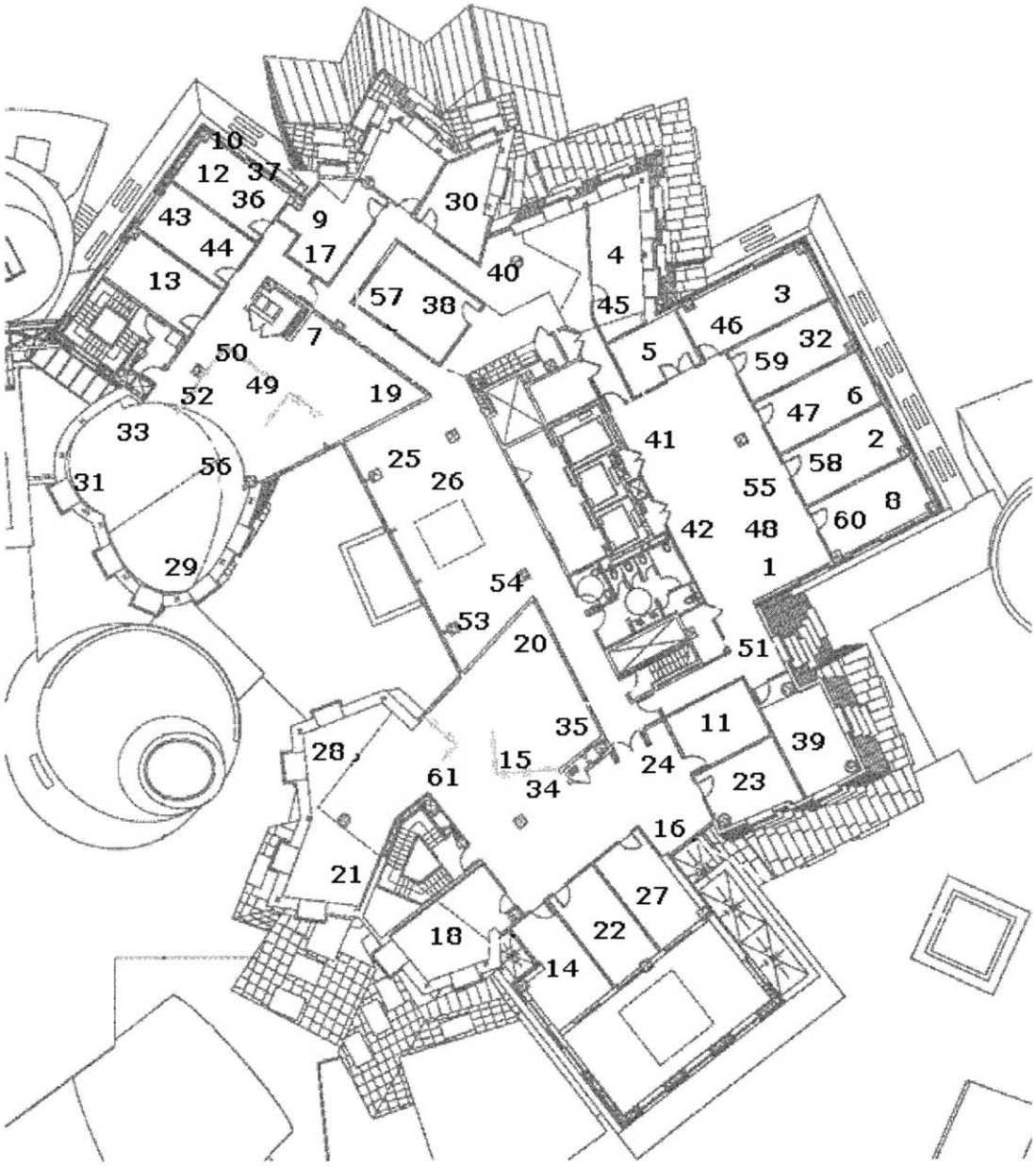

Figure 4-1: Floor plan of the 9th floor of the Gates Building in the MIT Stata Center. The numbers refer to the position of the motes. Node 8 is the root node.

4.1 B-MAC

In our experiments with B-MAC we varied three different parameters: the routing protocol, the transmission rate and the preamble duration. We tested the latest version of B-MAC [17] using three different routing protocols: /lib/Route, MINTRoute, and ReliableRoute. We performed experiments at both high and low data rate. We used two variants of power management: in "always-on" mode and using low-power listening with the default preamble length of 100 ms. Each experiment ran for 60 minutes and was repeated 5-8 times.

We measured throughput by calculating the percentage of messages sent by the motes that were actually collected by the root node. Table 4-1 shows the results using different power-management schemes and traffic conditions. In the overall average throughput measurements, we omit any nodes that were not able to reach the root node with any packet; these are the "dead" nodes listed in Table 4-1. Here, avg. and σ represent the average throughput and the standard deviation of all trials; min and max represent the best and worst trial in the set.

From the results reported in Table 1-1 and Figures 4-4 and 4-5, we observe:

- In always-on mode and under high data rate conditions, there is no significant difference in performance among the different routing protocols. We believe this is due to frequent packets collisions that void the benefits that any particular routing metric provides.

B-MAC Always On					
High Data Rate (10 seconds period)					
Routing Protocol	Throughput				Dead Nodes
	Avg.	σ	Max.	Min.	
MINT Route	56%	12%	72%	45%	0
/lib/Route	48%	2%	51%	46%	0
Reliable Route	47%	6%	54%	41%	0
Low Data Rate (60 seconds period)					
Routing Protocol	Throughput				Dead Nodes
	Avg.	σ	Max.	Min.	
MINT Route	49%	5%	54%	43%	0
/lib/Route	61%	14%	82%	47%	0
Reliable Route	55%	10%	71%	45%	0
B-MAC LPL 100 ms					
High Data Rate (10 seconds period)					
Routing Protocol	Throughput				Dead Nodes
	Avg.	σ	Max.	Min.	
MINT Route	37%	3%	41%	33%	1
/lib/Route	24%	7%	33%	17%	1
Low Data Rate (60 seconds period)					
Routing Protocol	Throughput				Dead Nodes
	Avg.	σ	Max.	Min.	
MINT Route	40%	2%	43%	39%	1
/lib/Route	24%	11%	34%	8%	2

Table 4-1: Performance using B-MAC and various routing protocols. "Dead nodes" refers to the number of nodes that reported 0% throughput.

- In always-on mode and under low-traffic conditions, /lib/Route performs better than MINTRoute and ReliableRoute. However, in always-on mode the throughput of the various protocols is not dramatically different; our results in Section 5.2 suggest that B-MAC may not be well suited to the particular application workload generated by Surge.
- Link-level retransmissions slightly improve throughput when medium contention is low, but decrease throughput when medium contention is high. Link-level retransmissions create a trade-off: on one hand they increase the probability that a particular packet is successfully received; on the other hand, they increase medium contention by increasing the average number of packets that need to be transmitted.
- With low-power listening, MINTRoute consistently performs better than /lib/Route. Figures 4-2 and 4-3 show the distribution of parents for each node during a single run with MINTRoute and Route (respectively). Notice that MINTRoute and Route are both fairly stable in their choice of parent, but that they differ in their selection of the best parent.

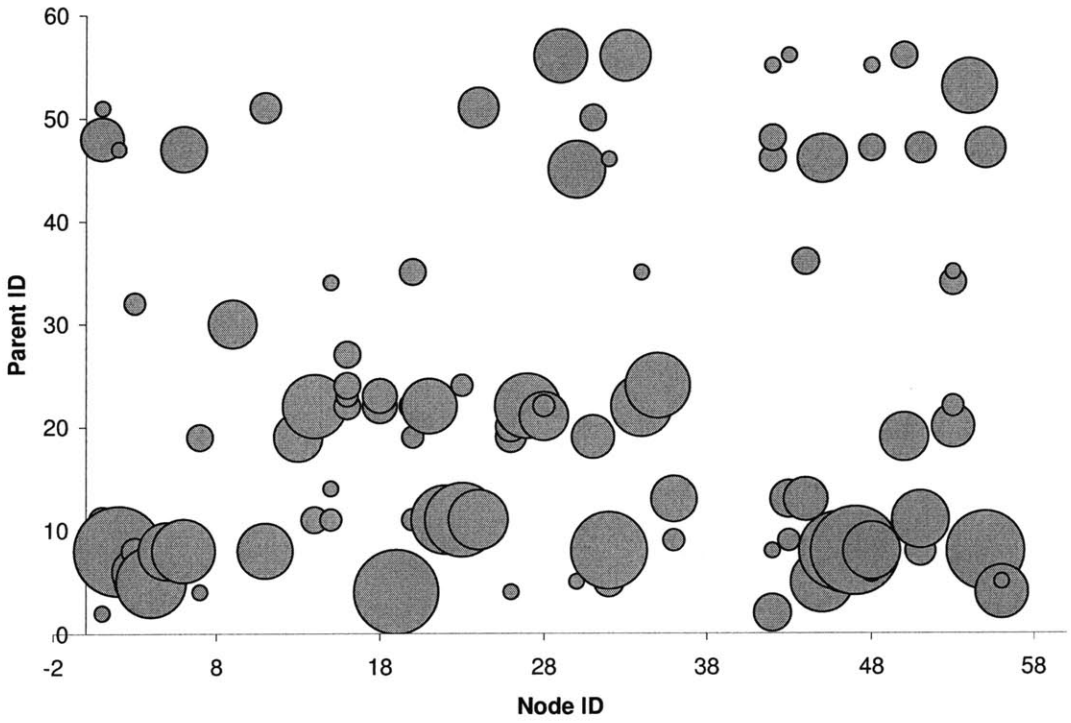

Figure 4-2: Parent distribution based on child's node ID using B-MAC with Low Power Listening, MINTRoute and 60 sec. data rate. Circle size is proportional to the number of times a given node routed data via a given parent.

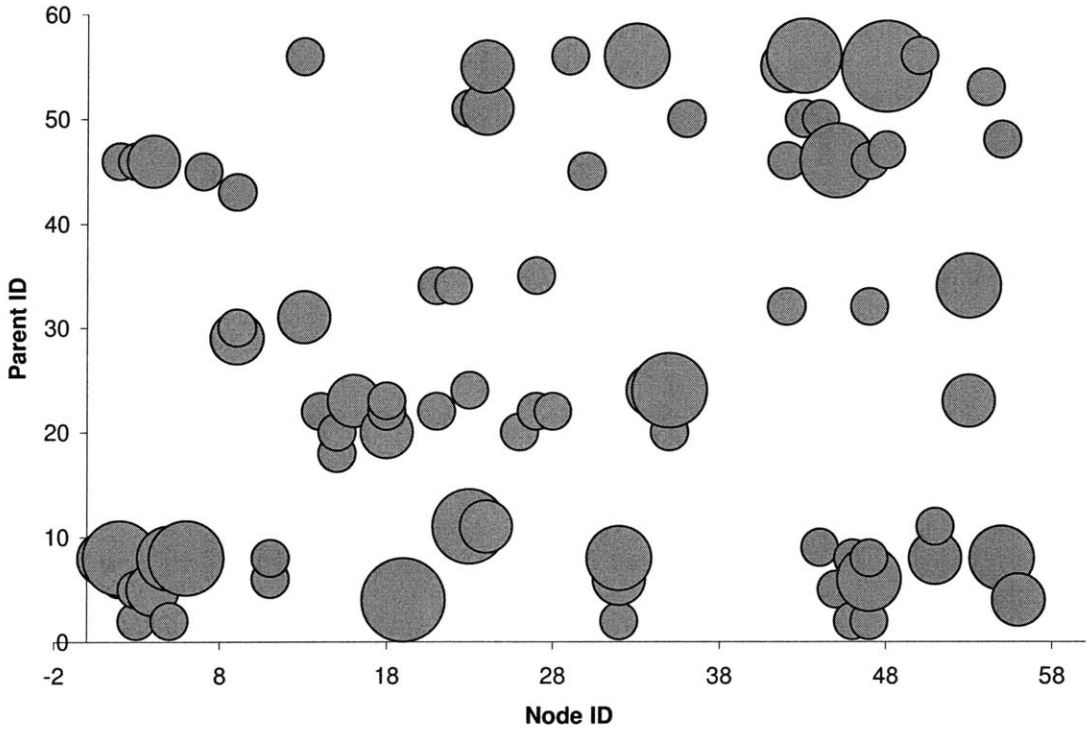

Figure 4-3: Parent distribution based on child's node ID using B-MAC with Low Power Listening, Route and 60 sec. data rate.

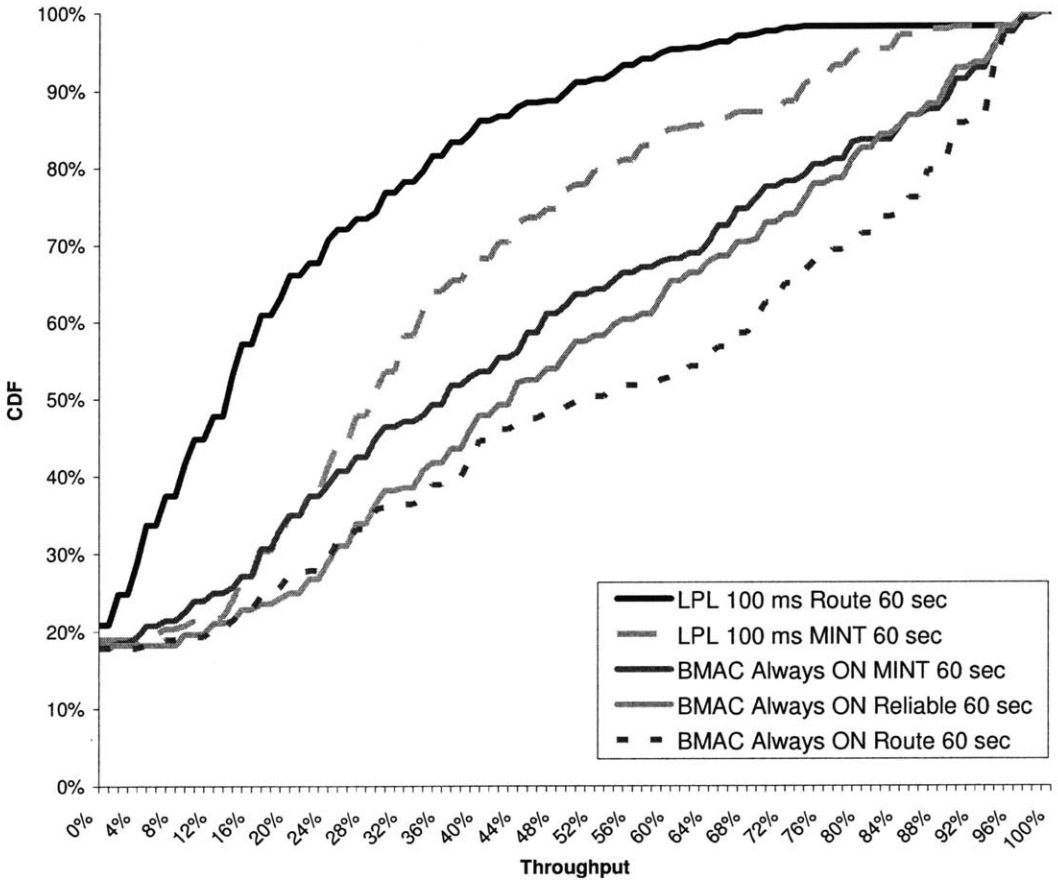

Figure 4-4: CDF of the average throughput using various configurations of B-MAC and 60 sec data rate. Here, steeper, more convex curves indicate worse performance (as fewer nodes achieve high throughput), whereas more concave curves indicate better performance.

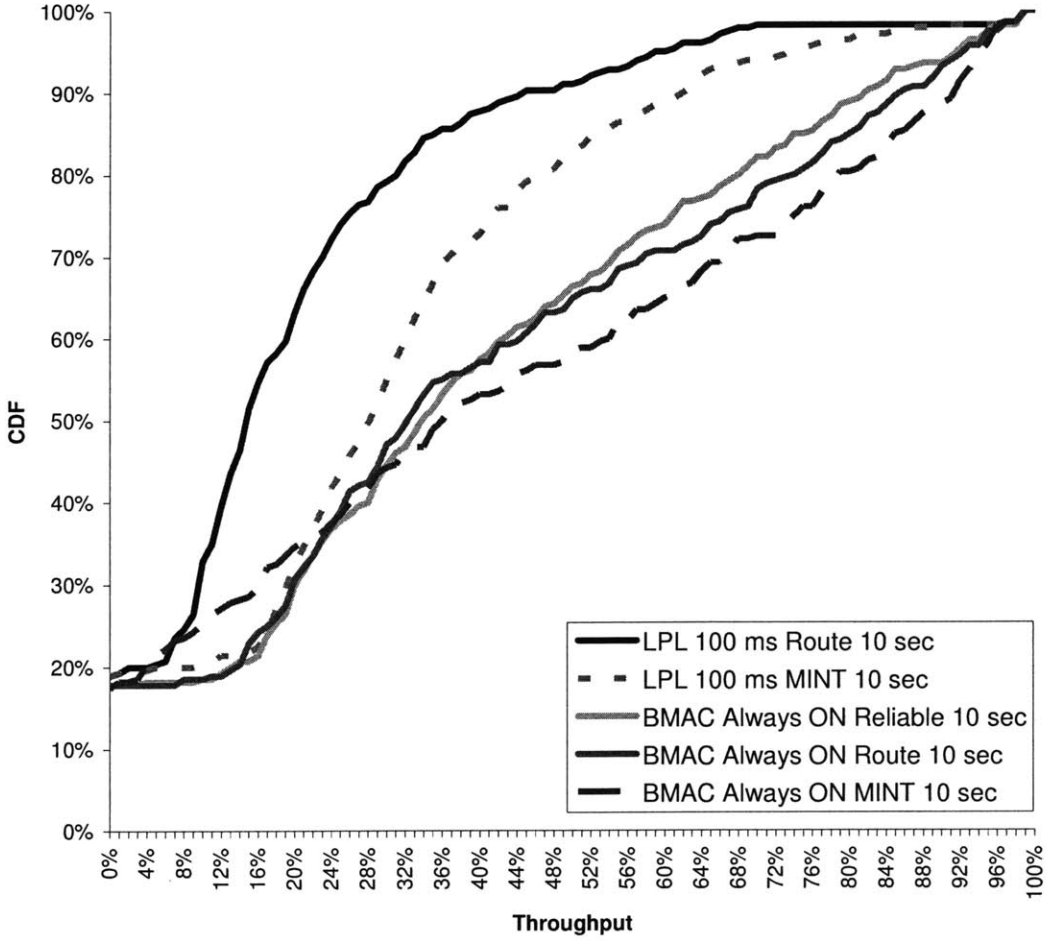

Figure 4-5: CDF of the average throughput using various configurations of B-MAC and 10 sec data rate.

4.2 S-MAC

As with B-MAC, in our experiments with S-MAC we varied the routing protocol, the transmission rate and the duty cycle. we tested the latest version of S-MAC (version 1.2) using: /lib/Route and MINTRoute (we did not try ReliableRoute because S-MAC implements retransmissions at the link layer).

We use the same high-rate and low-rate workloads as in the previous section. We tested S-MAC using two duty cycles: 90% and 10%. The latter is the default value for S-MAC and we picked 90% because is close to always-on mode but it still involves schedules and neighbors management. We ran 60 minute experiments and every experiment was run between 5 and 8 times.

Tables 4-2 and 4-3 show the results under different duty cycles and traffic conditions, with S-MAC with and without link-level retransmissions.

From the results in Table 4-2 and 4-3 and from figure 4-6, we observe:

- As in the case of B-MAC, link-level retransmissions do not always improve end-to-end throughput. At 90% duty cycle retransmissions tend to slightly improve end-to-end throughput at both rates. However, at 10% duty cycle, when medium contention is higher, retransmissions consistently harm end-to-end throughput.
- At 90% duty cycle, MINTRoute performs slightly better than /lib/Route. However, at 10% duty cycle /lib/Route substantially outperforms MINTRoute.

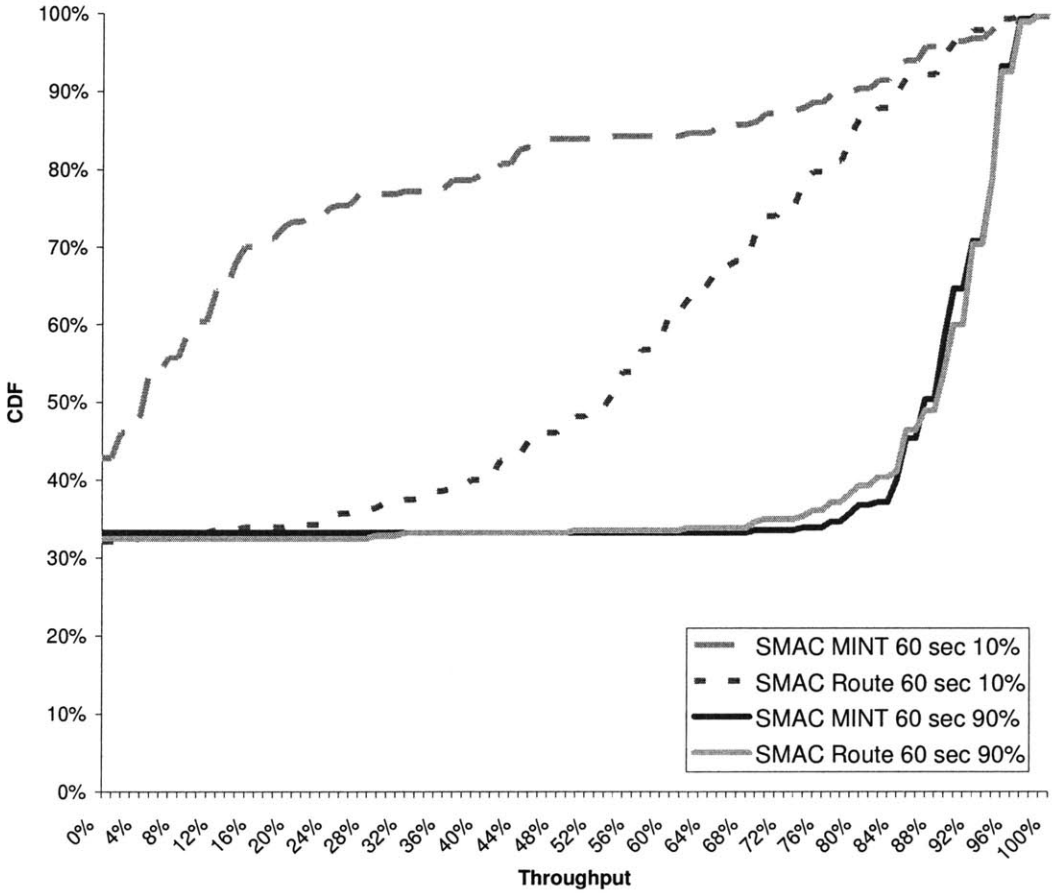

Figure 4-6: Average throughput CDF using various configurations of S-MAC with 60 sec data rate.

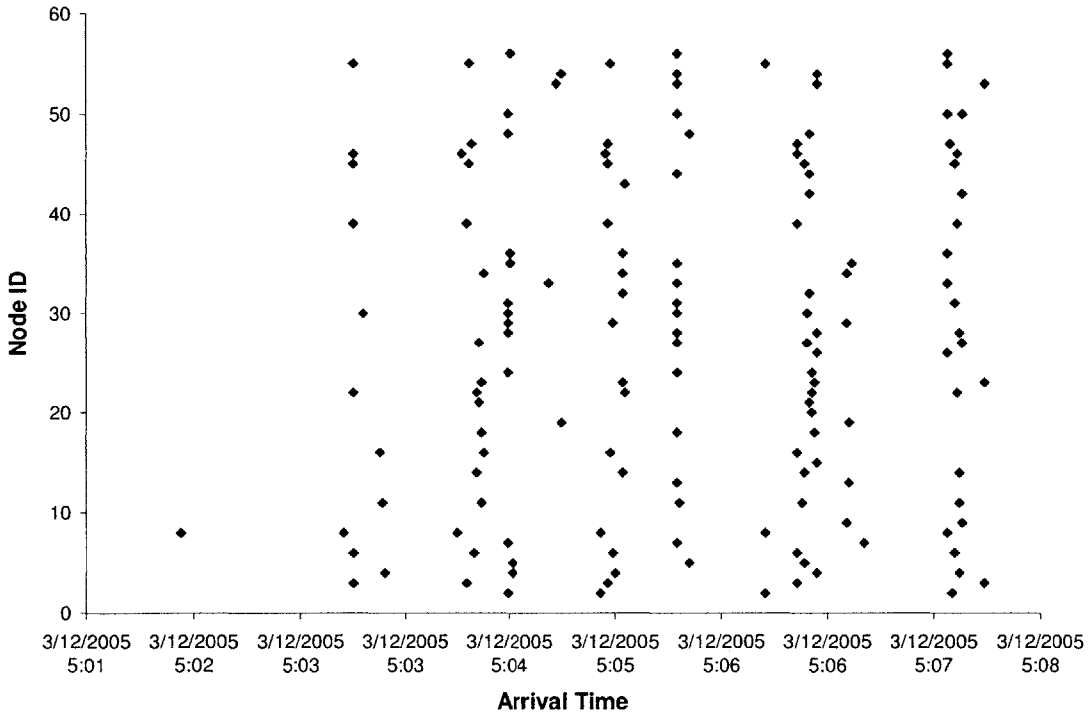

Figure 4-7: Arrival pattern of packets at the root node using S-MAC/Route with 10% duty cycle at 60 sec data rate.

S-MAC 90% duty cycle					
High Data Rate (10 seconds period)					
Routing Protocol	Throughput				Dead Nodes
	Avg.	σ	Max.	Min	
MINT Route	44%	13%	61%	26%	8
/lib/Route	31%	10%	40%	20%	6
Low Data Rate (60 seconds period)					
Routing Protocol	Throughput				Dead Nodes
	Avg.	σ	Max.	Min	
MINT Route	90%	3%	92%	85%	9
/lib/Route	89%	2%	91%	87%	8
S-MAC 90% duty cycle with no retransmissions					
High Data Rate (10 seconds period)					
Routing Protocol	Throughput				Dead Nodes
	Avg.	σ	Max.	Min	
MINT Route	32%	15%	45%	6%	2
/lib/Route	26%	7%	36%	19%	2
Low Data Rate (60 seconds period)					
Routing Protocol	Throughput				Dead Nodes
	Avg.	σ	Max.	Min	
MINT Route	90%	1%	91%	88%	2
/lib/Route	85%	3%	87%	80%	2

Table 4-2: Performance using S-MAC at 90% duty cycle and various routing protocols. "Dead nodes" refer to the number of nodes that reported 0% throughput.

- By studying the nodes routing tables, we determined that nodes using MINTRoute had difficulty finding a parent and tended to lose their parent once they found one. We believe this is because MINTRoute takes advantage of snooping to perform link-quality estimation and it is unable to cope with a MAC layer that saves energy by turning off the radio to reduce idle-listening. On the other hand, the /lib/Route algorithm is based mainly on minimum hop count and is much more robust under low duty cycle operations.
- From Figures 4-8 and 4-9 we can clearly see how increasing the S-MAC duty cycle creates instability in the parent selection algorithm. Figure 8 shows stable parent selection when a 90% duty cycle is used. On the other hand, Figure 9 shows how parent selection becomes unstable and parent switching more frequent when a 10% duty cycle is employed.
- By looking at Figure 4-7, we deduce that S-MAC duty cycling is able to spread in time packets that are simultaneously transmitted by all nodes and therefore take advantage of additional medium capacity.

S-MAC 10% duty cycle					
High Data Rate (10 seconds period)					
Routing Protocol	Throughput				Dead Nodes
	Avg.	σ	Max.	Min	
MINT Route	13%	7%	19%	2%	19
/lib/Route	20%	9%	36%	14%	3
Low Data Rate (60 seconds period)					
Routing Protocol	Throughput				Dead Nodes
	Avg.	σ	Max.	Min	
MINT Route	36%	7%	45%	28%	14
/lib/Route	63%	10%	74%	50%	8
S-MAC 10% duty cycle with no retransmissions					
High Data Rate (10 seconds period)					
Routing Protocol	Throughput				Dead Nodes
	Avg.	σ	Max.	Min	
MINT Route	16%	4%	21%	10%	2
/lib/Route	26%	7%	36%	19%	2
Low Data Rate (60 seconds period)					
Routing Protocol	Throughput				Dead Nodes
	Avg.	σ	Max.	Min	
MINT Route	37%	4%	42%	31%	24
/lib/Route	84%	3%	88%	81%	2

Table 4-3: Performance using S-MAC at 10% duty cycle and various routing protocols. "Dead nodes" refer to the number of nodes that reported

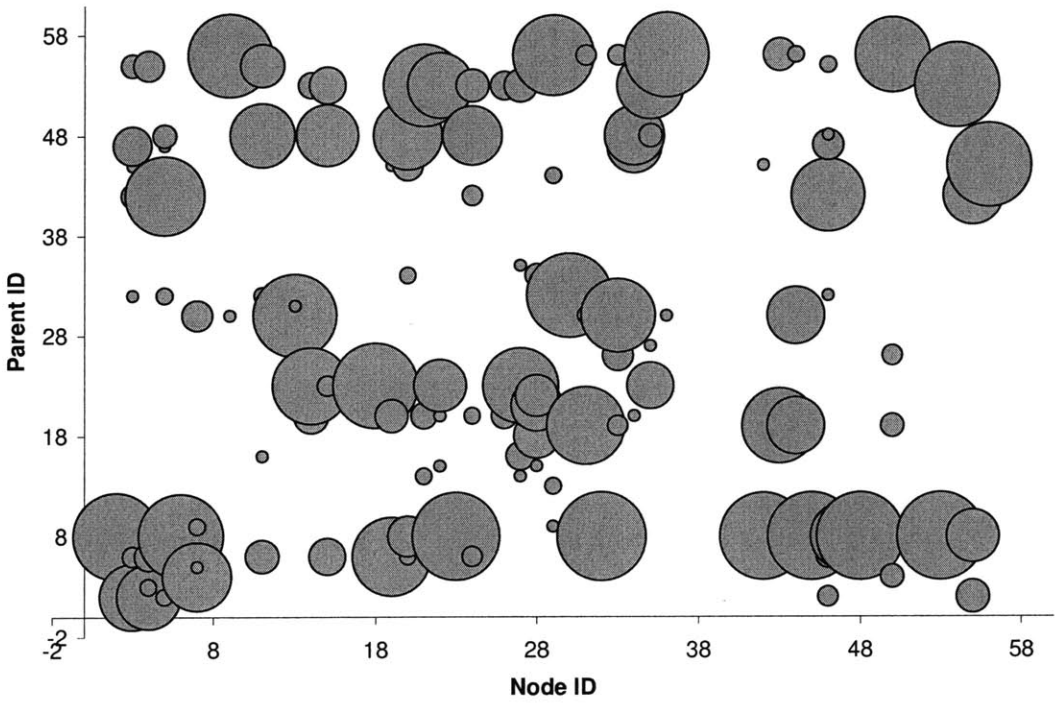

Figure 4-8: Parent distribution based on child's node ID using S-MAC with 90% duty cycle, Route and 60 sec. data rate. Circle size is proportional to the number of times a given node routed data via a given parent.

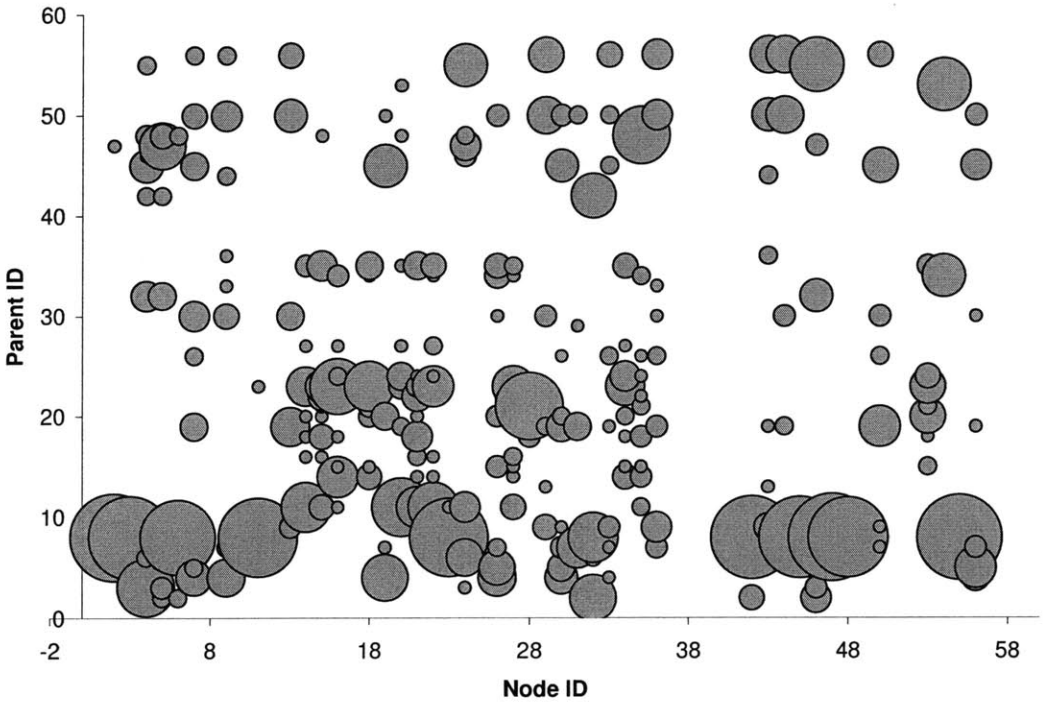

Figure 4-9: Parent distribution based on child's node ID using S-MAC with 10% duty cycle, Route and 60 sec. data rate.

Chapter 5

Overall Performance

In this chapter, this thesis compares the performance results from S-MAC and B-MAC to further study the source of many of the differences that we observed. Table 5-1 and Figures 5-1 and 5-2 summarize our results.

In the low-contention scenario with a data rate of 1 packet per minute (Figure 5-2), S-MAC with Route clearly outperforms all the other combinations of MAC and routing protocols. S-MAC at 10% duty cycle with Route reaches a throughput nearly equivalent to S-MAC at 90% duty cycle and much higher than B-MAC in any configuration.

In the high data rate scenario (Figure 5-1), the best configuration was B-MAC in always-on mode with MINTRoute.

DATA RATE	POWER MANAGEMENT: Enabled	POWER MANAGEMENT: Disabled
High	MINTRoute B-MAC 37%	MINTRoute B-MAC 56%
Low	Route S-MAC 84%	MINTRoute S-MAC 90%

Table 5-1: Best protocol combinations matrix with respective average throughput.

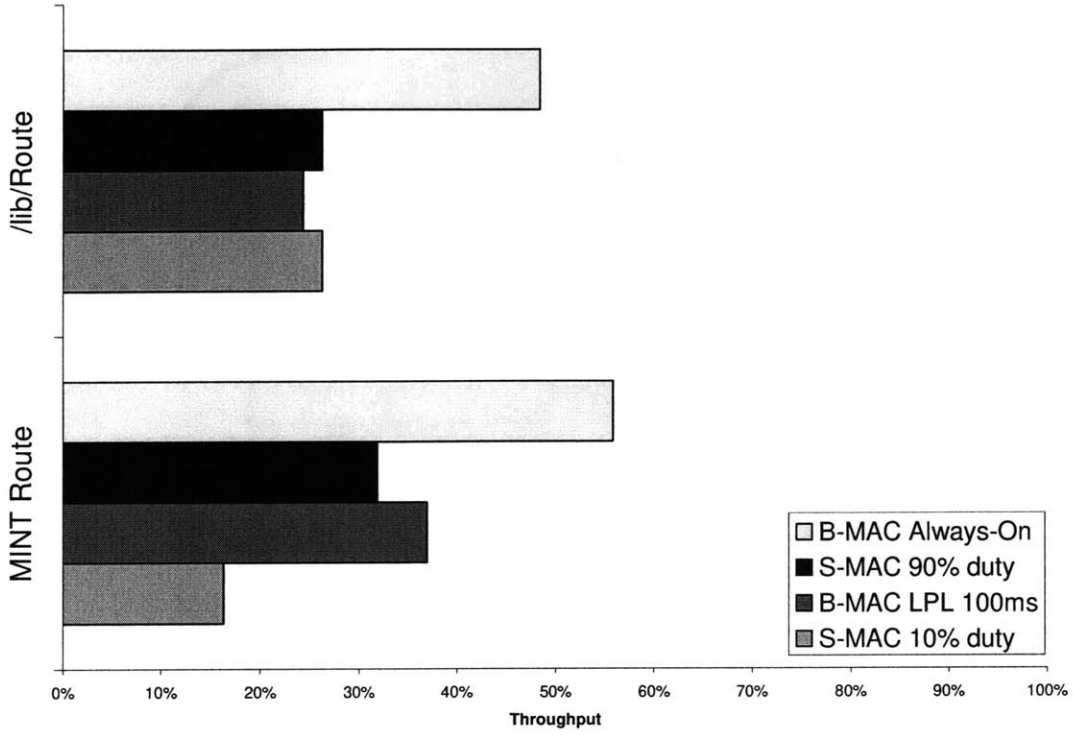

Figure 5-1: Average throughput comparison between S-MAC and B-MAC using different routing protocols and energy-management scheme and high data rate (10 sec).

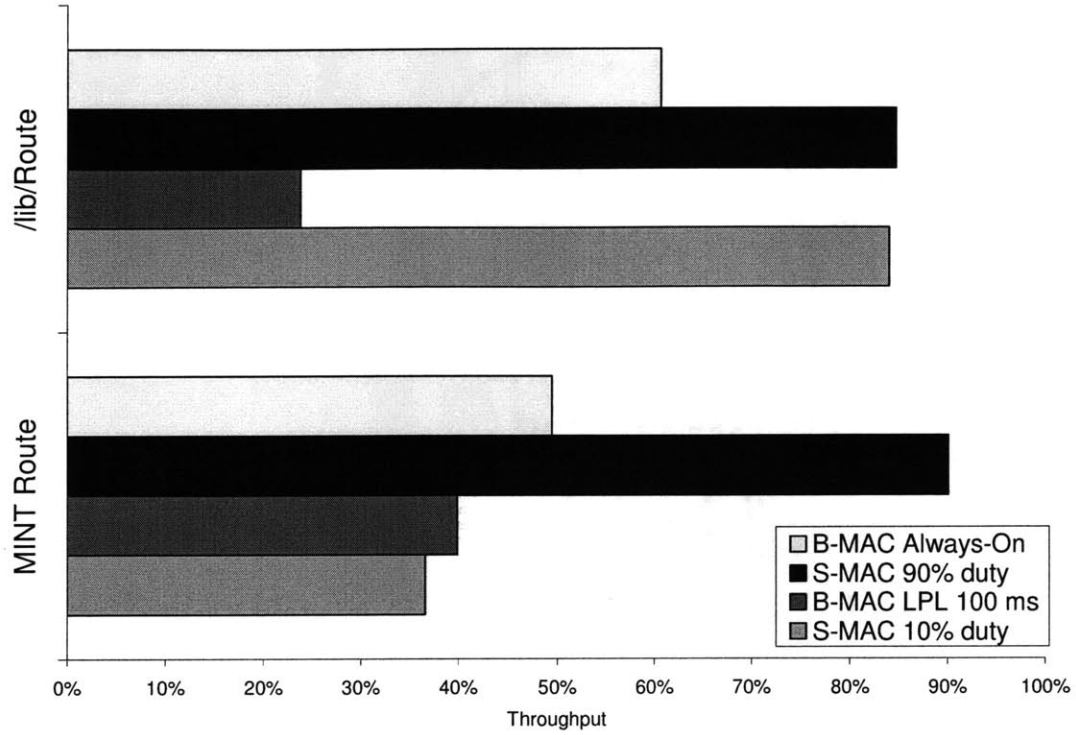

Figure 5-2: Average throughput comparison between S-MAC and B-MAC using different routing protocols and energy-management scheme and low data rate (60 sec).

5.1 Power Management

Looking at the results for low-power configurations at low data rate in Figure 5-2 and Figure 5-4, S-MAC with an appropriately tuned network stack clearly outperforms B-MAC. However, at high-rate, no configuration using a power management scheme was able to exceed a 50% throughput (even with retransmissions), making all the studied protocols inappropriate for the majority of real-life applications.

Figure 5-4 shows the CDF of the throughput for the experiments at 60 sec. data rate with power management enabled. Notice that the curve corresponding to S-MAC with Route is much more convex than the others, suggesting that a large fraction of nodes is able to achieve relatively high throughputs, whereas other approaches performed quite poorly in this setting.

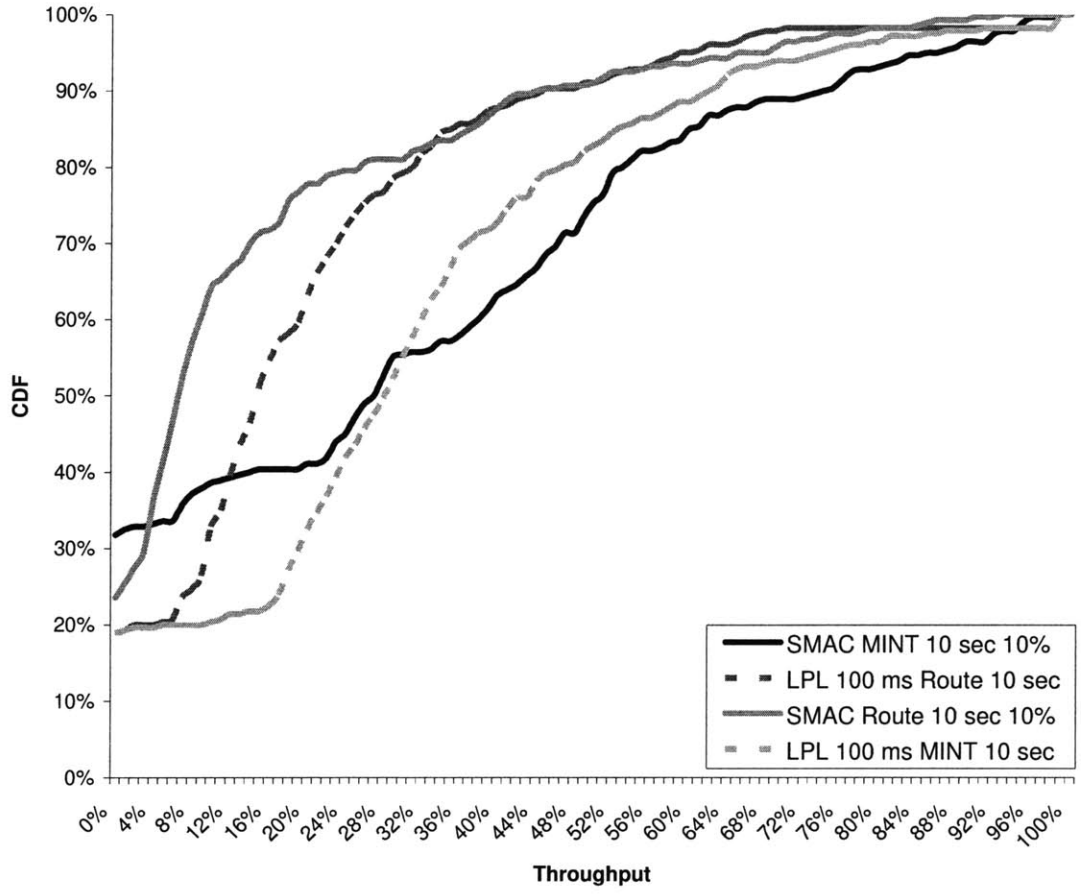

Figure 5-3: CDF of throughput at high data rate (10 sec) using Low Power Listening with B-MAC or 10% duty cycle with S-MAC.

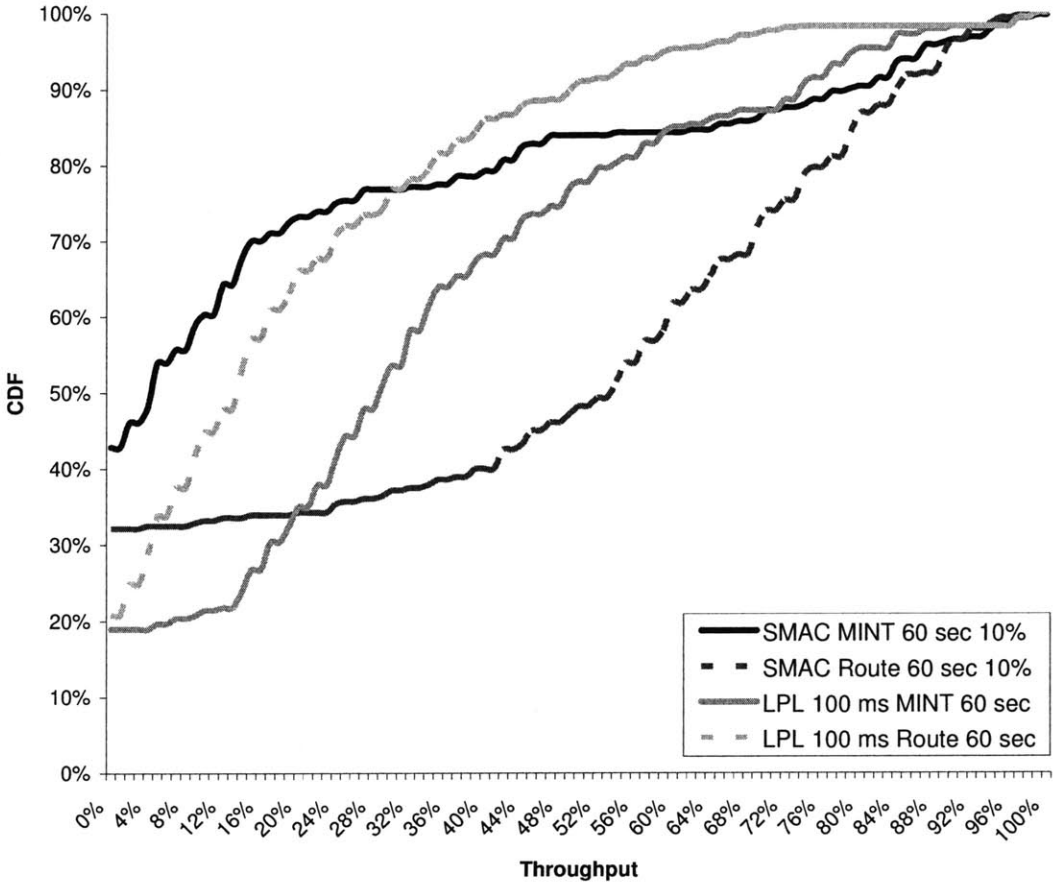

Figure 5-4: CDF of throughput using Low Power Listening with B-MAC or 10% duty cycle with S-MAC at low data rate (60 sec).

5.2 Inter-arrival time of send requests

The application layer strongly influences the performance of the entire network. We varied the pattern of send requests across different nodes and studied how this variation impacts the performance of both B-MAC and S-MAC. In the standard case, nodes in Surge transmit data at the beginning of every time period with no randomization; since nodes begin running at about the same time, this leads many send requests occurring at the same time across many nodes (Figure 5-6). To study the effect of eliminating this bursty behavior, we forced nodes to be out of phase by having them delay by a time proportional to their node ID (Figure 5-7). Using B-MAC, the average throughput increased from 49% to 93% transmitting at a rate of 60 sec. per packet and from 56% to 91% transmitting at a rate of 10 sec. per packet (Figure 5-5). With S-MAC, we see no benefits from using this technique, since S-MAC already spreads results in time (as shown in Figure 4-7).

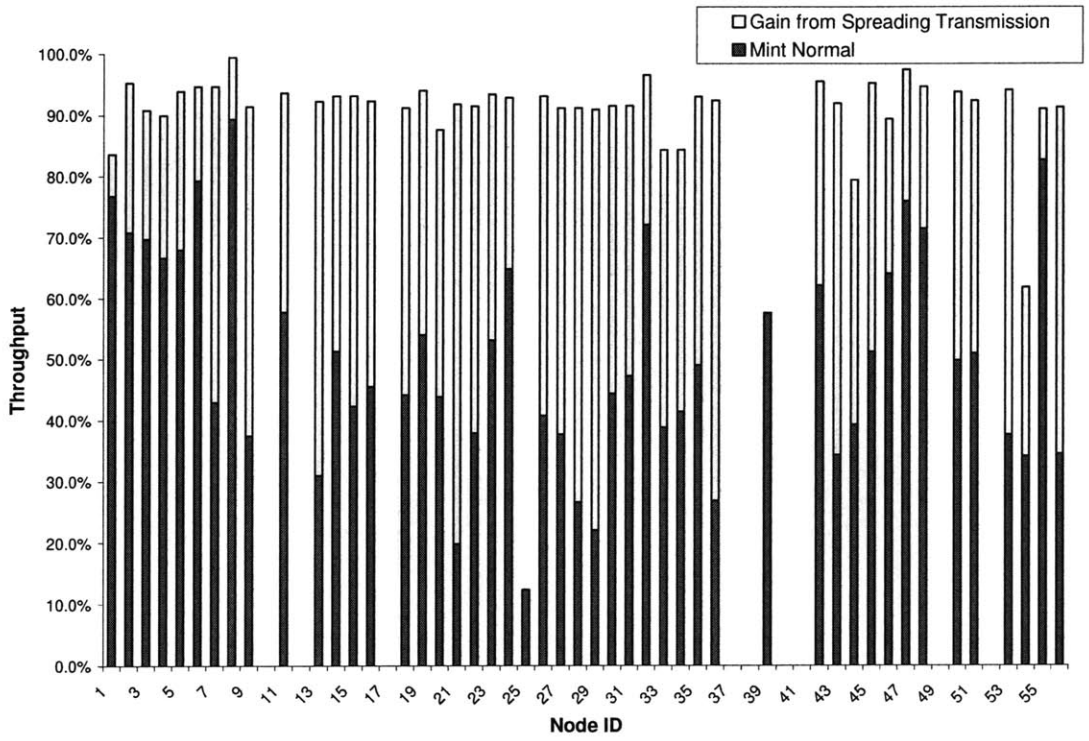

Figure 5-5: Average throughput per node. B-MAC/MINTRoute combination at data rate of 10 sec data rate with and without application-level delays between sending at different nodes ("spreading").

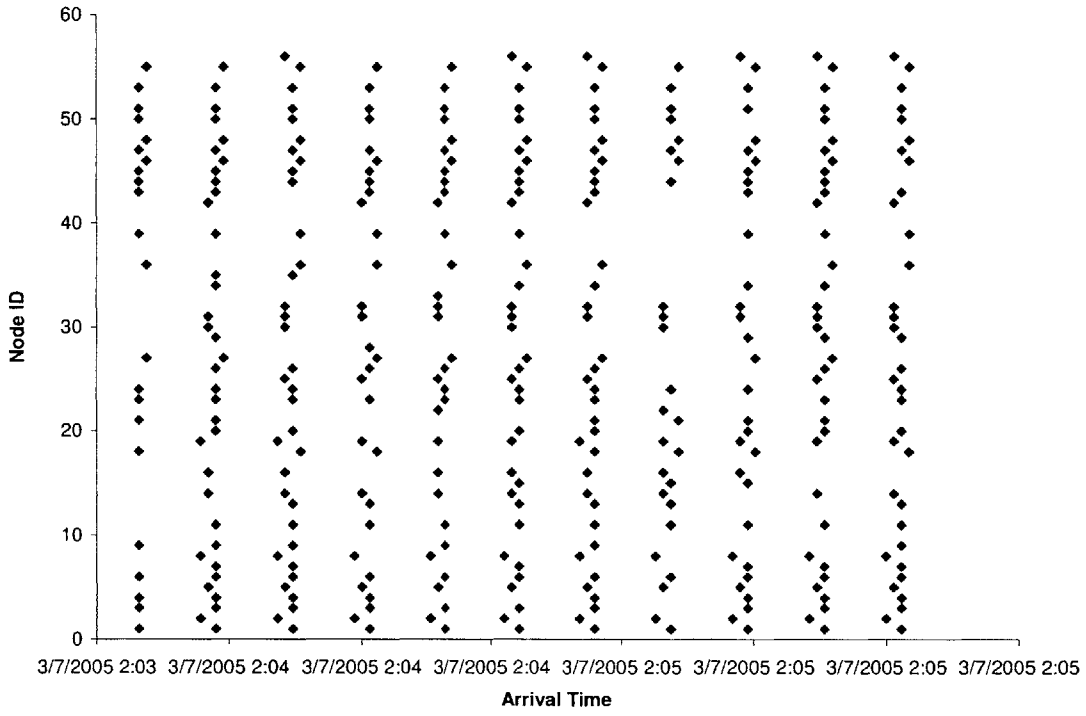

Figure 5-6: Arrival pattern of packets at the root node using B-MAC/MINT at 10 sec data rate.

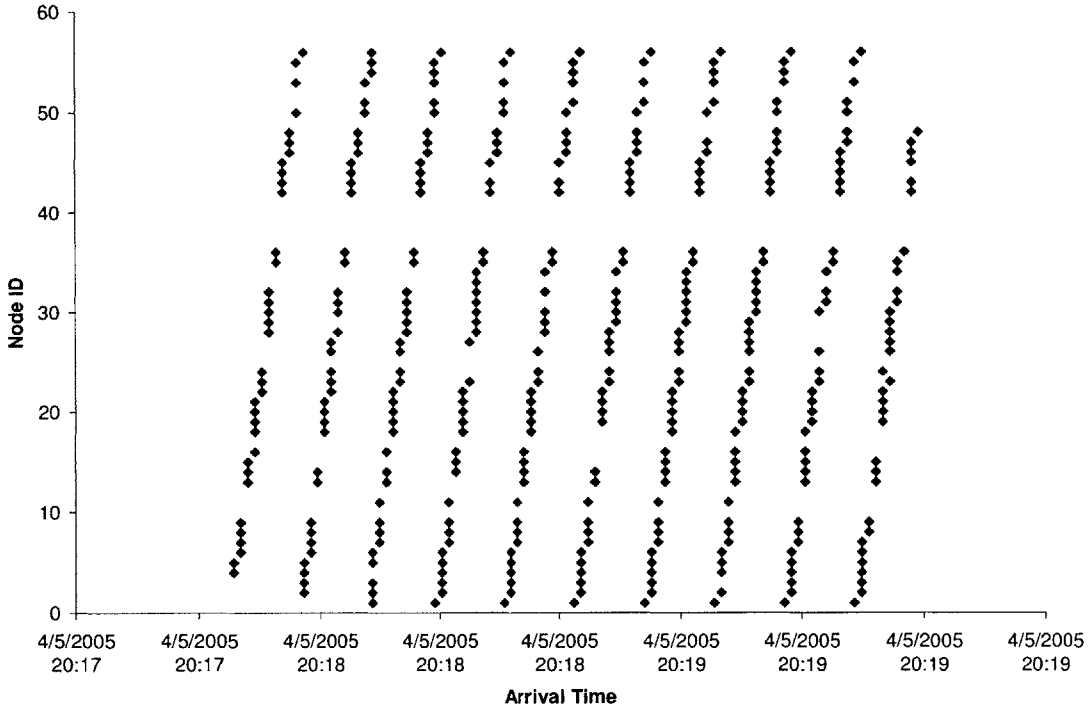

Figure 5-7: Arrival pattern of packets at the root node using B-MAC/MINT at 10 sec data rate having the application performing shifted transmissions based on the node ID.

5.3 Scaling issues

Figure 5-8 shows the performance of S-MAC and B-MAC with networks of different sizes. From these experiments, it is clear that B-MAC does not scale as well as S-MAC since the end-to-end average throughput decreases as we increase the number of transmitting nodes in the network. Using B-MAC Always-on and MINTRoute and a data rate of 60 seconds per packet, we measured an average throughput of 78% when we had only 15 nodes operational and of 49% when we had 46 nodes operational. On the other hand, because S-MAC partitions the network into different schedules and spreads sending over time, it is able to scale much better – in fact, we did not see any performance degradation as we increased the network size using 90% duty cycle. With S-MAC we noticed degradation of performance only when using 10% duty cycle and more than 15 nodes in the network. We believe this effect is due to the fact that the offered load is approaching the channel capacity.

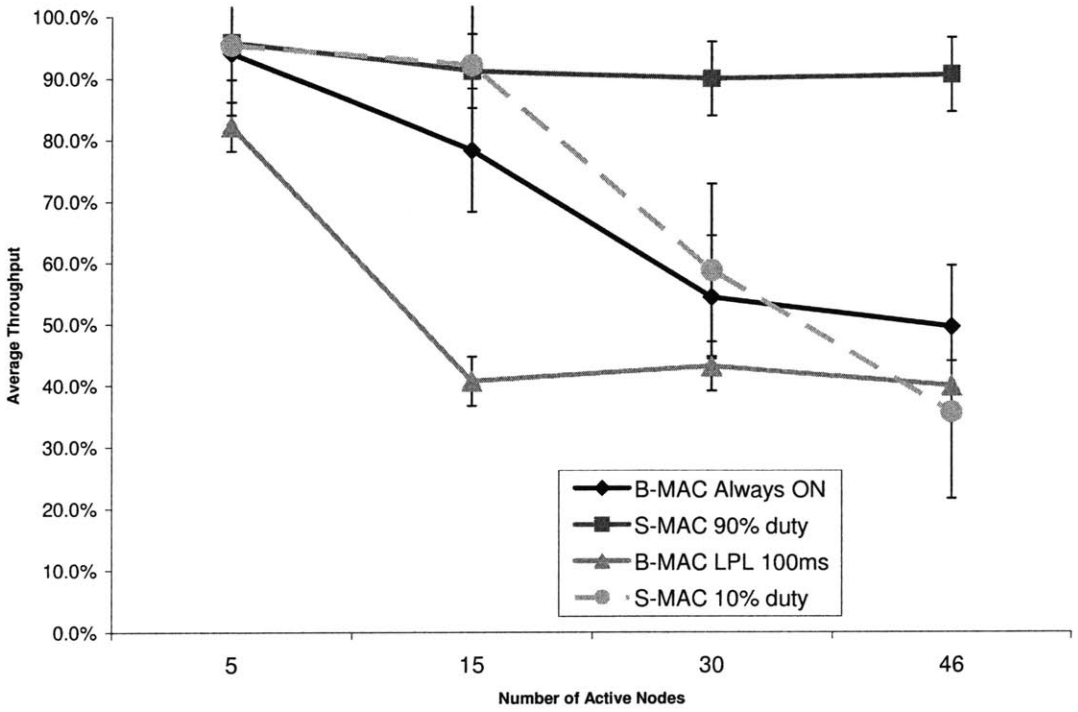

Figure 5-8: Average throughput of B-MAC and S-MAC with different network size. B-MAC is in always-on mode and with LPL 100ms at 60 sec data rate, S-MAC is at 90% duty cycle and 10% duty cycle and 60 sec data rate. Error bars are ±2σ.

Chapter 6

Discussion

Here this thesis briefly relates some observations that are clear from our performance analysis of the state of networking in TinyOS:

- Some power management schemes (and MAC layers) prohibit snooping on non-local radio traffic while some applications and routing layers rely on snooping for proper functioning. This is a fundamental issue that limits the ability to intermix different layer implementations. We discuss this issue in Section 6.1 in more details.
- Tuning power management settings (e.g., LPL preamble length and S-MAC sleep percentage) as well as other constants (e.g., link quality thresholds) is very hard for application designers, and making appropriate choices can dramatically affect application performance. For example, we saw that large, high-rate, power-managed networks with default power management settings in TinyOS perform poorly with any combination of MAC/routing layer, while small networks can perform quite well with default power management settings. Even when interfaces for tuning

parameters are provided (as in B-MAC), it is often unclear how adjusting these settings will affect network performance.

- Application workload and type of traffic dramatically affect network throughput. For example, we saw how introducing delays at the application can increase network average throughput in B-MAC from 50% to 90%, but that such changes have little effect on S-MAC. Such hidden dependencies make it very difficult for application designers to switch from one network stack to another and can be quite frustrating when deploying an application.

- No MAC/routing combination wins in every possible situation. The choice of MAC layer, in particular, can dramatically affect the effective channel utilization of applications in unexpected ways.

- Aside from issues where MINTRoute and S-MAC's power management scheme interacted very badly, we observed surprisingly little sensitivity to routing protocol in any of our experiments. Switching from Route to MINTRoute increases overall throughput by about 5% (σ=3.1%) (excluding the S-MAC at 10% duty-cycle case). In contrast, choosing the appropriate MAC for a given workload affects the overall throughput by about 16% on average.

- Link-level retransmissions do not always help and sometimes they hurt end-to-end throughput by increasing overall network contention. We discuss this issue in more detail in Section 6.2.

- Some protocols do not scale properly when run on larger networks. For example, B-MAC's performance with 46 transmitting nodes is 30% lower than with only 15 transmitting nodes. This effect is much less pronounced with S-MAC.
- When an energy-saving scheme is used, the literature often assumes that the root node will not follow the energy-saving scheme and will remain always on [17]. Although this assumption is probably valid in many real-life scenarios [20], from our experiments we observed that leaving the root node always on does not make any statistically-significant difference as far as the overall throughput of the network is concerned.

6.1 The idle listening vs. snooping trade-off

Our experiments suggest that there is a fundamental tradeoff between reducing idle-listening and utilizing overhearing in sensor networks.

There appear to be two primary uses for overhearing: in in-network processing, with applications such as TinyDB [9] or monitoring diffusion phenomena [11], and in network protocols, such as MINTRoute, that use it to collect statistics about network performance. In the former case, overheard messages are used to improve performance but are not necessary for correctness; in the latter case, as in MINTRoute, overhearing is necessary for acceptable network performance. In cases when the ability to overhear is impaired (as when S-MAC shuts off the radio channel), very bad behavior can result.

In the literature there are several examples of successful link-quality estimation techniques that do not involve overhearing of unicast packets addressed to other nodes [15, 21, 22]. Therefore, we believe that link-quality estimation can be performed without

the need for overhearing, which suggests that building routing protocols that depend on it is probably a bad idea (since any power conscious application is likely to want to turn off the radio at least some of the time).

6.2 Link-level retransmissions

Several proposals have claimed that link-level retransmissions substantially increase reliability of wireless sensor networks [13, 14, 17].

Our results show that this is not always the case. Link-level retransmissions present a trade-off between increasing the probability of end-to-end transmission success and decreasing overall medium-contention. From our results, we noticed that if the network is not congested, link-level retransmissions tend to benefit the overall end-to-end throughput. However, in cases where the network is already congested, link-level retransmissions actually decrease overall throughput, as illustrated in Figure 6-1.

This figure illustrates that at high duty cycles when medium-contention is low, link-level retransmissions improve throughput; on the other hand, at low duty cycle when medium-contention is high, link-level retransmissions decrease performance.

These observations suggest that network protocols need some facility to determine whether losses are due to contention or simply transient external interference.

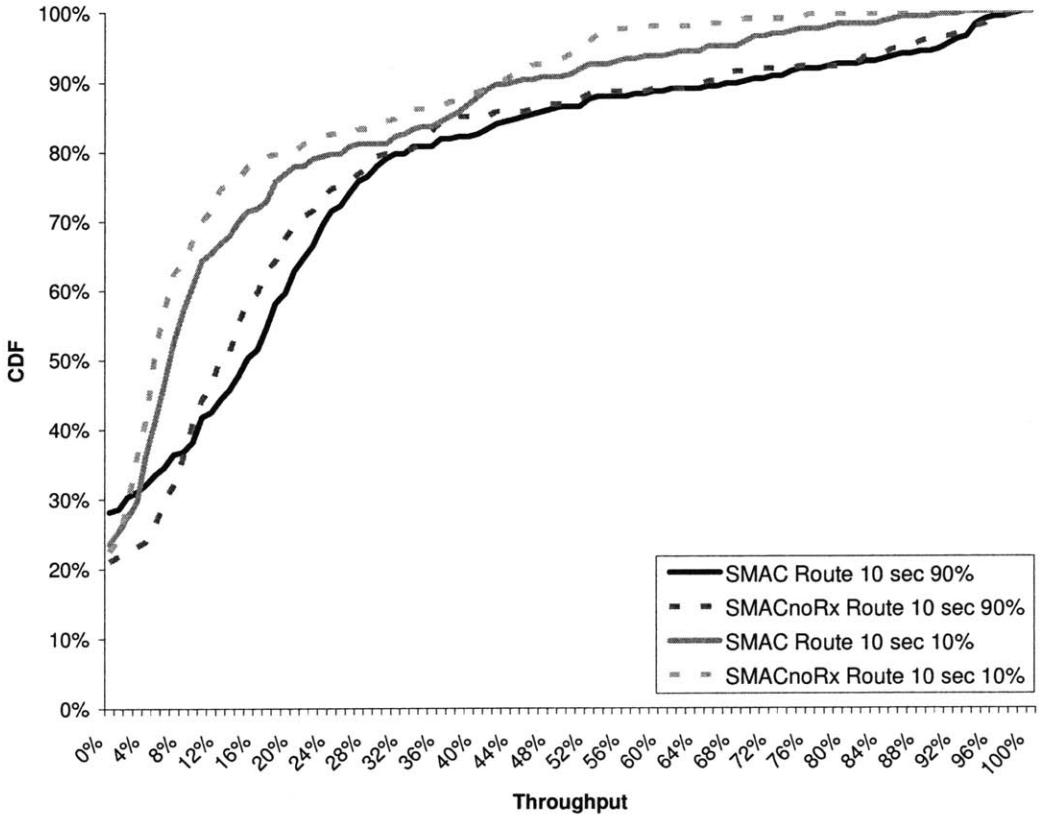

Figure 6-1: Throughput CDF using S-MAC and varying the maximum number of link-level retransmissions and the data rate.

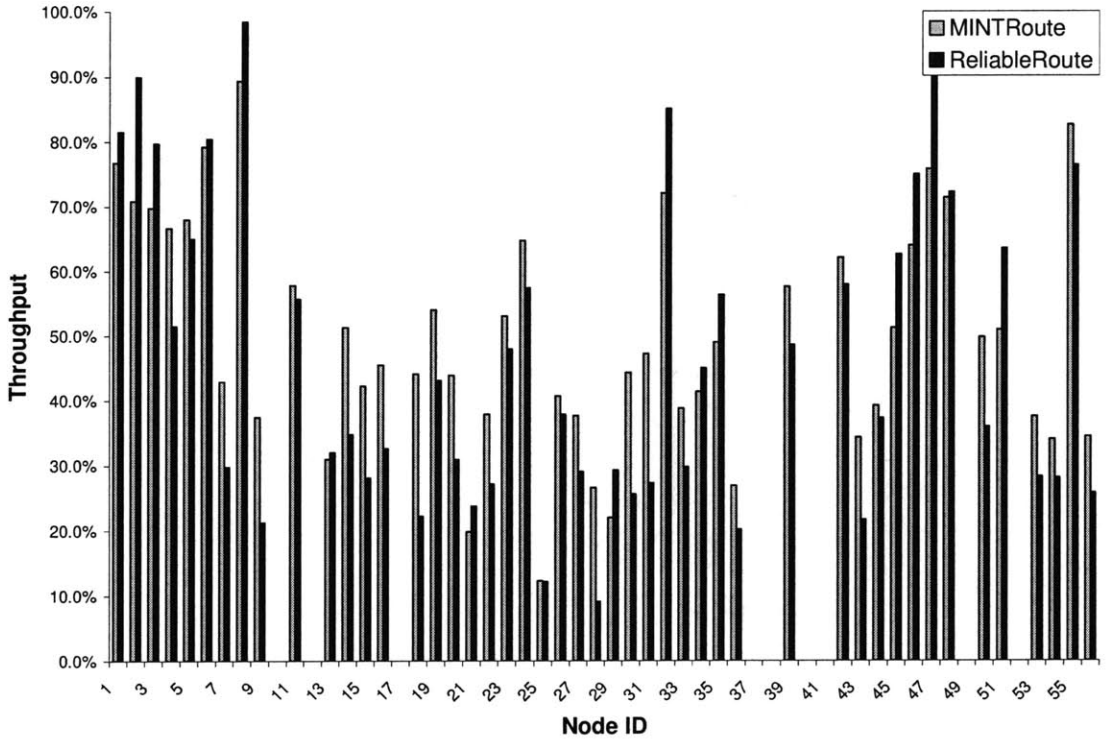

Figure 6-2: Average throughput for each node in the network running B-MAC at 10 sec period with MINTRoute and ReliableRoute.

6.3 Cross-layering

Others have noted the porosity of network layers and the extent of cross-layering optimization in sensor networks [10]. We observed this as well; for example, different components of the routing layer choose to implement different parts of link-layer retransmissions in different layers. In the case of B-MAC, for example, the application implements duplicate suppression, the network layer implements retransmissions, and the MAC layer implements acknowledgements. This makes the job of application designers very difficult. Our experience suggests that this cross-layering is a source of incompatibility among protocols, since, as with the routing and link layers in B-MAC, it tends to create coupled sets of layers that depend on specific, non-standard cross-layer APIs. Though this coupling may help increase performance of a single application or network stack in the short-term, careless cross-layering limits the ability of protocol designers to innovate at different layers and will ultimately make developing solid implementations of sensor network protocols very hard. A pressing need in the sensor network community is to converge on standard APIs and agree to abide by them.

6.4 Towards an efficiency-oriented architecture

The quest for energy-efficiency will clearly be among the principal drivers for the success of any architecture for sensor networks. Furthermore, since energy-efficiency was never a goal or a design constraint on the Internet [26], it is not clear that the strict layering that characterizes the Internet architecture fits the needs of sensor networks. Because most sensor network deployments are homogeneous in terms of goals and objectives [10], sensor network designers are able to trade flexibility in terms of

applications and types of traffic for energy-efficiency. Sensor networks do not need to be able to carry every type of traffic over the same network stack or using the same forwarding algorithm. Sensor networks designers can restrict the possible applications to better tailor a particular implementation and achieve a greater level of optimization and energy-efficiency. For example, the nodes of a network deployed in Central Ecuador to perform volcanic eruption monitoring do not have to be necessarily interoperable with the nodes deployed on an oil tanker to measure vibrations in the engine. Therefore, no common protocol (the so-called "narrow waist" of the Internet architecture) is required. Every protocol can be changed in order to best fit the application requirements.

Our experiments clearly show that approaches like S-MAC that try to apply strict layering to sensor networks do not achieve particularly good results in terms of throughput or energy-efficiency. For example, like the IP layer in the Internet, S-MAC was designed to be agnostic to the protocols running above it. However, some routing protocols perform poorly using S-MAC because they conflict with its attempts to provide link-level retransmissions and power scheduling.

Every node in a sensor network tends to be, at the same time, a source of packets and a router for other nodes' packets. Furthermore, it is often the case that applications want all of the nodes in the network to perform a certain level of in-network processing to avoid wasting energy forwarding packets that can be combined or filtered in-network. Exposing every packet to the application layer violates some of the layering principles that are fundamental to the Internet architecture [27]. The routers in a sensor network must be application-aware in order to be more efficient in their routing decisions. However, being application-aware does not mean violating the end-to-end argument or

developing software that is not based on modular and interchangeable components. Saltzer *et al.* [26] describe how performance requirements may require moving some functions into the communication stack.

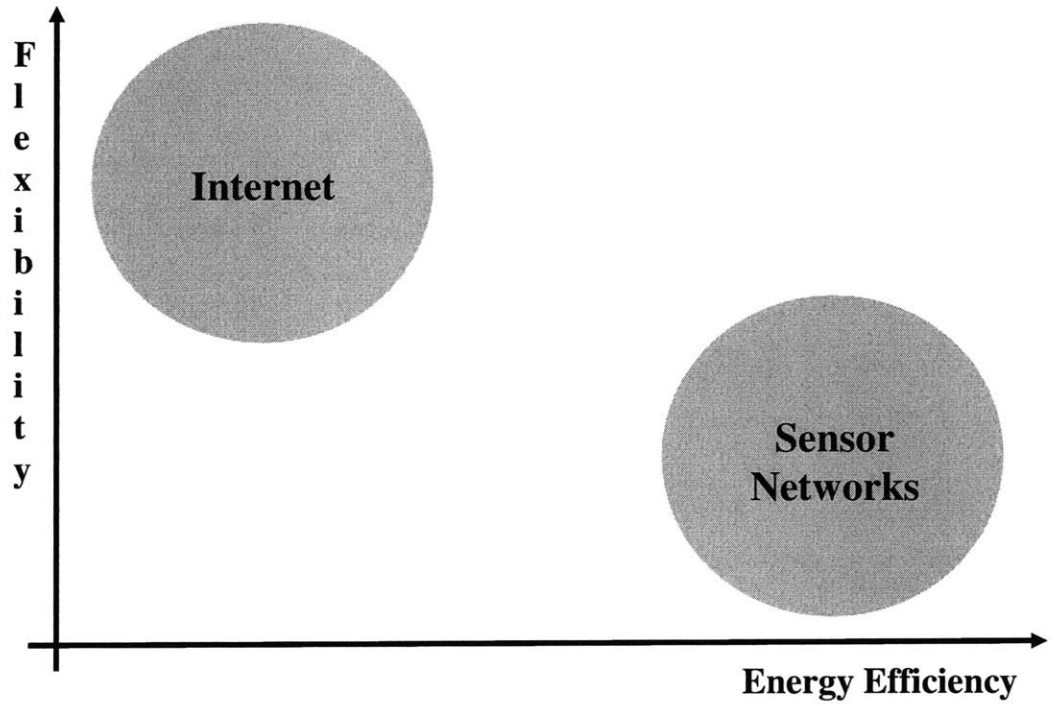

Figure 6-3: Trade-off energy-efficiency versus flexibility of running multiple applications and types of traffic on the same network implementation.

So far the approach of the research community has been to increase energy-efficiency and manage complexity by creating a layered communication stack with additional interconnections between layers. As we described in Section 6.3, this approach of using cross-layering has not been able to create the desired modular and interchangeable implementations because we lack a standardized API for the various

layers. However, some recent work has been done in this direction, describing new layered architectures with the goals of sensor networks in mind [28].

Moreover, other researchers have proposed non-layered abstractions, such as the neighborhood [29, 30] or the protocol heap [20]. The neighborhood is clearly an important concept in wireless ad-hoc networks and many layered protocols are based on it. However, no working implementation is able to prove that the neighborhood abstraction can capture all the communication tasks present in a network stack or that that neighborhoods are sufficient to supplant layering. Moreover, some of the proposed abstractions [29] are not designed with power-efficiency in mind, leaving the application running on top the neighborhood abstraction with no control over communication resources.

By proposing a non-hierarchical and highly interconnected abstraction called a "role", Braden et al. solve some of the limitations in terms of flexibility and efficiency that layering presents [20]. The limited services and the homogeneity of sensor networks might be easily represented using this highly interconnected "role" abstraction that might allow achieving the energy-efficiency required in sensor networks operations. However, it is not clear how much overhead and complexity the "role" abstraction might generate compared to a layered architecture because of the variable size of the heap-based header.

6.4.1 Recommendations

Without ignoring innovative and promising abstractions such as the neighborhood and the role, the sensor network community should improve the current cross-layered architecture as embodied by the current TinyOS radio stack [17]. We believe that this

approach has the most promise for the development of a usable and working implementation in the short term. By establishing a set of standardized interfaces between the various layers and agreeing on the functions that every layer should perform, the developers will be able to overcome the current interoperability drawbacks of the cross-layered architecture. At the same time, the architects should go beyond the standard Internet hour-glass architecture acknowledging the following peculiarities of sensor networks:

- Full interoperability across all applications is not a requirement for sensor networks. Additional efficiency might be gained by allowing the substitution of every layer in the network stack (no "narrow waist").
- The rapid changes happening in hardware development require fast software development based on code reuse and modular design. For this reason, standardized and agreed-upon interfaces between layers are required.
- In-network processing is an important means to achieve energy-efficiency; therefore nodes in a sensor network need to perform application-aware intelligent forwarding.

Hence, we advocate a layered but highly interconnected architecture for sensor networks, with no common layer across all the implementations (no "narrow waist") but with standardized interfaces that allow interchanging protocols to best suite the application-specific requirements. Moreover, a sensor network won't be a "stupid" network with only "smart" hosts. Every node will perform intelligent application-aware routing to minimize energy and bandwidth waste.

Chapter 7

Conclusion

By studying how different combinations of MAC, routing, and power management schemes interact with each other under several different application workloads, this thesis has illustrated several issues with the current state of protocol implementations in TinyOS. First, and somewhat to our surprise, we found that no combination of MAC and routing protocols dominates all others, and that some combinations of MAC and routing protocols are particularly incompatible with one another. Second, we observed that some issues that we thought would dramatically affect performance (routing protocols, retransmissions) had little effect. Third, we observed (as others have), that cross-layer optimizations tend to blur lines between layers in sensor networks, and that this blurring makes the design of modular, interchangeable software components very difficult. We believe these lessons are an important step towards understanding the source of performance problems in sensor networks and that they will prove invaluable in our own and other's future work designing next generation protocol architectures for sensor networks.

Bibliography

[1] P. Buonadonna, D. Gay, J. Hellerstein, W. Hong, and S. Madden, "TASK: Sensor Network in a Box," *European Workshop on Sensor Networks*, 2005.

[2] Geoff Werner-Allen, Jeff Johnson, Mario Ruiz, Jonathan Lees, and Matt Welsh, "Monitoring Volcanic Eruptions with a Wireless Sensor Network," *EWSN 2005*, January 2005.

[3] M. Welsh, and G. Werner-Allen, G. Motelab webpage – http://motelab.eecs.harvard.edu.

[4] V. Rajendran, K. Obraczka, and J.J. Garcia-Luna-Aceves, "Energy-Efficient, Collision-Free Medium Access Control for Wireless Sensor Networks," *ACM SenSys '03*, November 2003.

[5] T. van Dam, K. Langendoen, "An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks," *SenSys '03*, November 2003.

[6] B. Hohlt, L. Doherty and E. Brewer, "Flexible Power Scheduling for Sensor Networks," *Proceedings of the Third International Symposium on IPSN,* April 2004.

[7] TinyOS, http://webs.cs.berkeley.edu/tos/

[8] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," *in 21st Conf. of the IEEE Computer and Communications Soc. (INFOCOM)*, pages 1567-1576, June 2002.

[9] S. Madden, M. Franklin, J. Hellerstein, and W. Hong, "TAG: a tiny aggregation service for ad-hoc sensor networks," *5th Symposium on Operation System Design and Implementation*, December 2002.

[10] P. Levis, S. Madden, D. Gay, J. Polastre, R. Szewczyk, A. Woo, E. Brewer and D. Culler, "The Emergence of Networking Abstractions and Techniques in TinyOS," *Proceedings of the First USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI 2004)*, March 2004.

[11] L. Rossi, B. Krishnamachari, C. Kuo, "Distributed Parameter Estimation for Monitoring Diffusion Phenomena Using Physical Models," *First IEEE International Conference on Sensor and Ad hoc Communications and Networks (SECON)*, October 2004.

[12] R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler, "Lessons from a sensor network expedition," *in Proceedings of the First European Workshop on Wireless Sensor Networks (EWSN '04)*, January 2004.

[13] J. Zhao, and R. Govindanm "Understanding Packet Delivery Performance in Dense Wireless Sensor Networks," *ACM SenSys '03*, November 2003.

[14] A. Woo, T. Tony, and D. Culler, "Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks," *ACM SenSys 2003*, November 2003.

[15] D. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A High-Throughput Path Metric for Multi-Hop Wireless Routing," *in Proceedings of. ACM MobiCom*, September 2003.

[16] W. Ye, J. Heidemann, and D. Estrin, "Medium Access Control With Coordinated Adaptive Sleeping for Wireless Sensor Networks," *in IEEE/ACM Transactions on Networking*, pages 493-506, June 2004.

[17] J. Polastre, J. Hill, and D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks," *ACM SenSys 2004*, November 2004.

[18] Crossbow – http://www.xbow.com

[19] Crickets – http://www.nms.lcs.mit.edu/crickets

[20] R. Braden, T. Faber, and M. Handley. "From Protocol Stack to Protocol Heap - Role-Based Architecture". *First Workshop on Hot Topics in Networks (HotNets-I),* 2002.

[21] M. Yarvis, W. Conner, L. Krishnamurthy, J.Chhabra, B. Elliott, and A. Mainwaring, "Real-World Experiences with an Interactive Ad Hoc Sensor Network," *IWAHN 2002*, August 2002.

[22] M. Yarvis, N. Kushalnagar, H. Singh, A. Rangarajan, Y. Liu, and S. Singh, "Exploiting Heterogeneity in Sensor Networks," *INFOCOM 2005*, March 2005.

[23] G. Lu, B. Krishnamachari, and C. Raghavendra, "An Adaptive Energy-Efficient and Low-Latency MAC for Data Gathering in Wireless Sensor Netwoks," *IPDPS '04*, 2004.

[24] Broke, and J. Burrell, "Form ethnography to design in a vineyard," *in Proceedings of the Design User Experience (DUX) Conference*, June 2003. Case Study.

[25] Intel R&D – http://www.intel.com/research/exploratory/wireless_sensors.htm

[26] D. D. Clark, "The design philosophy of the DARPA Internet protocols," *in SIGCOMM*, pages 106-114, August 1988.

[27] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-t-End Arguments in System Design," *in ACM Transactions on Computer Systems*, pages 277-288, November 1984.

[28] D. Culler, P. Dutta, C. T. Eee, R. Fonseca, J. Hui, P. Levis, J. Polastre, S. Shenker, I. Stoica, G. Tolle, and J. Zhao, "Towards a Sensor Network Architecture: Lowering the Waistline," *to appear in Proceedings of the Tenth Workshop on Hot Topics in Operating Systems (HotOS X)*, 2005.

[29] K. Whitehouse, C. Sharp, E. Brewer, and D. Culler, "Hood: A neighborhood abstraction for sensor networks," *MOBISYS '04*, June 2004.

[30] M. Welsh, and G. Mainland, "Programming Sensor Networks using Abstract Regions," *NSDI '04*, March 2004.