An Extended Internet Architecture for Low-Power Wireless Networks - Design and Implementation

Jonathan W. Hui

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2008-116
http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-116.html

September 12, 2008

An Extended Internet Architecture for Low-Power Wireless Networks - Design and Implementation

by

Jonathan Wing-Yan Hui

B.S. (Carnegie Mellon University) 2003
M.S. (University of California, Berkeley) 2005

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Computer Science

in the

GRADUATE DIVISION
of the
UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:
Professor David E. Culler, Chair
Professor Eric Brewer
Professor Paul Wright

Fall 2008

The dissertation of Jonathan Wing-Yan Hui is approved:

Chair Date

Date

Date

University of California, Berkeley

Fall 2008

An Extended Internet Architecture for Low-Power Wireless Networks - Design and Implementation

by

Jonathan Wing-Yan Hui

Abstract

An Extended Internet Architecture for Low-Power Wireless Networks -

Design and Implementation

by

Jonathan Wing-Yan Hui

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor David E. Culler, Chair

A decade ago as wireless sensor network (sensornet) research took off many researchers in the field eschewed the use of IP as inadequate and in contradiction to the needs of sensornets. Since those beginnings, the field has matured substantially, a huge collection of protocols have been invented and evaluated, and we have gained experience in how sensornets are used in practice. Over this same period, the Internet has also evolved with the introduction of IPv6 that addresses scalability and easy configuration and management.

In this dissertation, we present the design of a complete IPv6-based network architecture for sensornets that preserves the protocol layering but increases cross-layer visibility. We equate the IP link to radio neighbors, giving network- and higher-layer protocols necessary visibility into radio connectivity. We show how the link can appear always-on while operating a low duty-cycles. An adaptation layer between the link and network layers applies cross-layer optimizations to reduce packet headers and per-node state. At the network layer, we address issues of effective configuration and management over ad-hoc multihop networks as well as efficient forwarding and routing through cooperation with the link layer. By providing effective end-to-end datagram delivery, transport- and application-layer protocols remain largely unchanged.

We validate the architecture with a complete implementation that is the first to incorporate many sensornet techniques into an IP architecture, including duty-cycled links, header compression, adaptive hop-by-hop forwarding, and efficient routing. In addition to providing end-to-end interoperability with existing non-sensornet IP devices, our implementation outperforms existing sensornet approaches that do not adhere to any particular architecture or standard. In a real-world application, the implementation achieved an average duty-cycle of 0.65%, per-hop latency of 62ms, and delivery rate of 99.98% over four weeks with each node generating 1 pkt/min. In light of this demonstration of full IPv6 capability, we review the central arguments that led the field away from IP. We believe that the presence of an architecture, specifically an IPv6-based one, provides a strong foundation for sensornets going forward.

Professor David E. Culler
Dissertation Committee Chair

To my parents, Frank and Yvonne Hui.

Contents

List of Figures

List of Tables

Acknowledgments

A number of people have led me to where I am today. Their tremendous support and guidance have been critical to my success, and I am grateful for everything they have done for me.

David Culler has been an amazing advisor - more than I could have ever hoped for. He always had the uncanny ability to identify the golden nuggets out of a pile of seemingly uninteresting ideas and always pushed to "find the truth." David thinks and communicates unlike any other. While his "Cullerisms" first challenged me to understand what he was trying to say, I eventually learned that they were meant to convey new ways of thinking about the problem.

Special thanks to Eric Brewer and Paul Wright, who were on my dissertation committee. Their comments, suggestions, and feedback brought the necessary outsider's perspective that helped me to better articulate and position my work.

This dissertation would not have been possible without my fellow colleagues at Arch Rock. Their genuine interest in making sensornet technology real motivated me to always push forward with my efforts.

Brent Chun was a co-conspirator in the development of a proof-of-concept implementation that ultimately paved the way for this dissertation. His hacking abilities are mind boggling - I will never understand his ability to manipulate emacs faster than frame buffers can fill. But his continuous enthusiasm and energy helped pull me through when times were tough.

Alec Woo contributed countless thoughts and feedback on both networking and systems aspects. While he may deny it, Alec was a pioneer in the wireless sensor networking space and his deep understanding of the field demonstrates that.

Gilman Tolle's insights and fresh perspectives, especially from a user and application-layer view, have been critical to making this work both production quality and usable in the real world. Gil always kept my thoughts in check. Out of his own interest, he consistently reviewed high-level designs as well as low-level implementation details.

Wei Hong has always been supportive of my efforts. His willingness to let me explore and deviate from the plan made this dissertation possible.

The TinyOS/NEST/Soda 410 team at Berkeley was a world-class group that played a huge part in helping me develop both technically and as a person.

Prabal Dutta was always filled with ideas. He always had novel ways of tackling problems, sometimes using mundane and well-understood techniques but uniquely applied to a different domain. His diverse thoughts made me think of problems in new ways.

Joe Polastre was never shy to push me in my technical abilities, always quick to challenge my thoughts as well as identifying bugs in my code. Joe was also central to keeping our Berkeley life sane, organizing Monkey Head Thursdays and instigating the purchase of an Xbox for our research group.

Cory Sharp has taught me countless hacking tricks. He was never shy in demonstrating his crazy hacking abilities, always cheerfully showing how to better manipulate a shell and perform operations in a one-line Perl script. Cory's joyful attitude always brought light into our windowless Soda 410.

Bruce Krogh introduced me to the world of wireless sensor networks. As my undergraduate advisor, Bruce taught me what research was all about, gave me a glimpse into what graduate school would be like, and helped me publish my first peer-reviewed paper. Working under the NEST program led me to Berkeley and David Culler.

Crystal Fong has been incredibly patient with me for nearly a decade and has always been supportive of my work. She provided the necessary counterbalance to my technical and academic work. I'm so glad we have made it so far together.

My parents, Frank and Yvonne Hui are responsible for providing me with a world of opportunities, always emphasizing education and pushing me to reach for higher goals. They taught me independence while guiding me through countless trials. I would not be where I am today without their love and support.

This material is based upon work supported by the National Science Foundation under grants #0435454 ("NeTS-NR") and #0454432 ("CNS-CRI").

Chapter 1

Introduction

Over the past several decades, we have seen the formation of a ubiquitous communications infrastructure - the Internet. Much of the Internet's success can be attributed to the end-to-end [149] and layered design [12] principles set forth early as the design tenets of the IP architecture. The end-to-end principle pushed functionality to the end-points, keeping the core simple with good scaling characteristics. IPs layered architecture meant that peers communicated in terms of the capabilities provided by the layer below. The architecture decomposed networking problems into manageable components, fostering both innovation and rapid evolution. The ability for the architecture to scale with so many new technology developments over time is a testament to the success of the architecture.

Wireless sensor networks (hereafter sensornets) have the potential to finally bridge the physical and digital worlds. But their vastly different characteristics from traditional IP devices have pushed the networking problem to a new extremum. Their embedding in physical space often brings challenging operational factors, including constrained memory, computation, and communication capability; ad-hoc communication with low-power wireless links; and limited energy sources. Sensornets are also expected to have very different usage scenarios compared to traditional IP networks, such as in-network processing, structured traffic flows, and data-oriented naming and communication.

As sensornet research took shape, many researchers in the field argued forcefully that "while many of the lessons learned from Internet and mobile network design will be applicable to designing wireless sensor network applications ... sensor networks have different enough requirements to warrant reconsidering the overall structure of applications and services" [52]. The Internet architecture was eschewed for several reasons including the following [52]:

- The severe "resource constraints may cause us to give up the layered architecture".
- "The sheer numbers of these devices, and their unattended deployment, will preclude reliance on a broadcast communication or the configuration currently needed to deploy and operate networked devices."
- Localized algorithms and in-network processing will be required to achieve robustness and scalability.
- "Unlike traditional networks, a sensor node may not need an identity (e.g., an address)." Naming will be data-centric.
- "Traditional networks are designed to accommodate a wide range of applications." Sensornets will be special-purpose networks tailored to the sensing task at hand.

In addition, it was argued that to tackle the challenges of sensornets the traditional interfaces and layers of system abstraction should not be assumed [72]. By providing a framework for defining abstractions and allowing the community to progress, new network abstractions were expected to emerge [102]. Indeed, by introducing the Active Message Dispatch ID at the head of each message, rather than a conventional header format, TinyOS [126] lead away from IP. The vast array of protocols developed by the community operate at the link layer, rather than the network layer. The serial interface to a "basestation mote" favored the use of application level gateways at the root of the sensornet, so sensornets were organized in a manner similar to IrDA and USB, rather than as an IP subnet similar to Ethernet or WiFi.

Since those beginnings, the field has matured substantially, a huge collection of protocols have been invented and evaluated, and we have gained experience in how sensornets are used in practice. We have seen

significant advances in many areas, including: low-power link protocols based on low-power listening or time-synchronized communication; networking protocols providing many-to-one, one-to-many, and any-to-any communication; hop-by-hop congestion and flow control designed for these communication paradigms; transport protocols involving reliable transfer of both small and bulk data; and application-layer mechanisms to provide access to named data.

In general, sensornet research has focused much more on network protocol algorithms and mechanisms, rather than on networking in the broader sense. Much of this work did not constrain the solution to any particular architecture, leading to the development of numerous application-specific networking protocols that were hard to compose. Some work sought to provide a communication architecture that could combine these disjoint networking protocols by placing the narrow-waist of the protocol stack at the link layer [41, 46, 138]. But by being agnostic to the network protocol in use, these architectures do not define broader networking issues of discovery, naming, addressing, configuration, and management.

Proposed networking architectures sensornets that do address broader networking issues can be considered Layer 1, 2, 7 solutions because all protocols (including application layer protocols) operate directly above the link layer. But by not supporting IP and the Internet architecture, connecting these networks to conventional hosts or the Internet require stateful and complex application gateways. These application gateways must understand any application profiles that may be used in the sensornet, and any changes to the application protocols used in the sensornet require changes on the gateway. This is true not only in research, but also for existing commercial solutions, including ZigBee [195], Z-Wave [192], and WirelessHART [66].

In this dissertation, we break the trend by proposing an IPv6-based network architecture for sensornets. Based on our work in this dissertation, we believe that a communication architecture for sensornets should keep the narrow-waist at the network layer. IPv6 provides such an architecture: the IPv6 forms of layering, addressing, header formats, configuration, management, forwarding, and routing provide the necessary structure for designing and implementing mechanisms at all layers of the stack. The presence of an architecture allows designers and implementors to focus and improve the implementation in the process. We have seen such benefits with other widely-adopted architectures such as RISC and UNIX. In this dissertation,

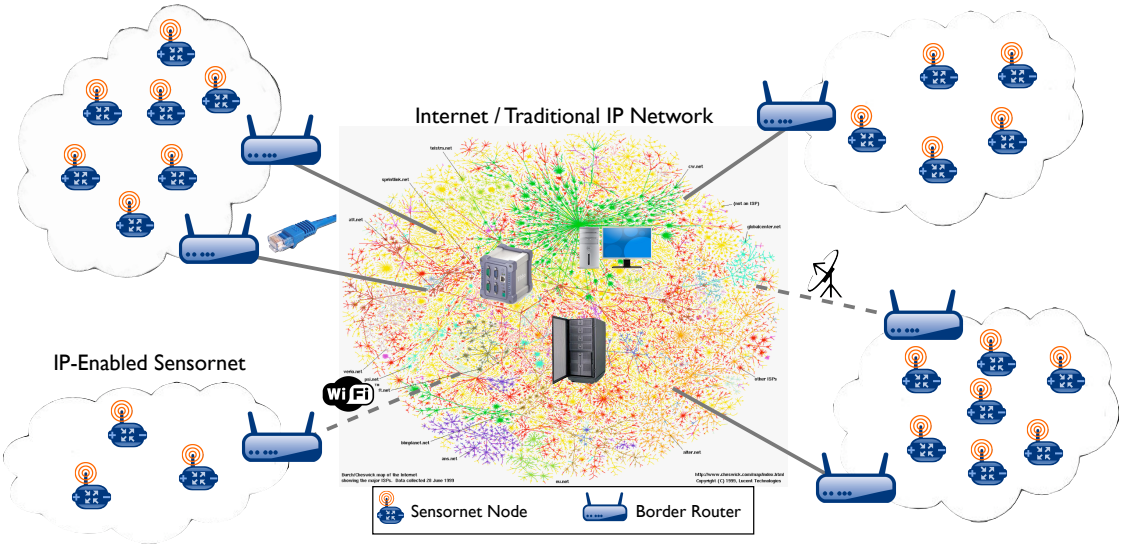


Figure 1.1: **The Next Tier of the Internet.** Using IPv6, we can extend the Internet to include sensornets, providing connectivity to the physical world. Interoperability at the network layer allows nodes to communicate end-to-end with any arbitrary IP device without requiring stateful application gateways in between. Border routers are traditional IP routers with a sensornet interface, connecting sensornets to other IP networks at the network layer using existing link technologies (e.g., Ethernet, WiFi, and satellite).

we show how to adapt the IPv6 network architecture to sensornets and demonstrate the performance, energy efficiency, and reliability that a high quality implementation can achieve with this architecture.

1.1 An IPv6-Based Network Architecture for Sensornets

Using IPv6, we can realize the full potential of the Internet, extending it with direct connectivity to the physical world through sensornets, as shown in Figure 1.1. The ubiquity of IP brings with it many advantages. It brings unprecedented interoperability at the network layer between sensornets and any other IP device. This is in contrast to Layer 1, 2, 7 solutions that require stateful gateways for application-layer protocol translation directly connected to the sensornet. Supporting IP also provides invaluable interoperability with existing hosts and services as well as the ability to utilize the broad body of existing IP tools when connecting sensornets to other IP networks (i.e., firewalls, proxies, caches, etc.).

IPv6 is the designated successor of IPv4 as the network protocol for the Internet [31]. Expecting to continue the exponential increase of IP hosts, scalability is a primary goal of IPv6. The IPv6 address space is

much larger at 128-bits to alleviate the IPv4 address shortage. Autoconfiguration (autoconf) and management is central to the IPv6 architecture, allowing hosts to autoconf IP addresses and other arbitrary parameters without a human in the loop [21, 121, 168]. Link-specific protocols (e.g., ARP and DHCP) have been pulled into the IPv6 framework [168]. IPv6 also supports a richer set of communication paradigms, including a scoped addressing architecture and multicast into the core design [30]. IPv6 also reflects the advancement of link technologies, by increasing the minimum link MTU requirement to 1280 bytes. Unquestionably, IPv6 defines a lot more functionality than IPv4. The obvious question then is: why IPv6 for sensornets?

We claim that *IPv6 is better suited to the needs of sensornets than IPv4 in every dimension*. The generality and extensibility of the IPv6 network architecture allows us to utilize mechanisms that have become so pervasive in the sensornet community. Sampled-listening, Trickle-based dissemination, hop-by-hop feedback, and collection routing are only a few of the mechanisms that allow us to implement an IPv6-based network architecture that is more efficient than other existing solutions. Furthermore, IPv6 allows for more efficient implementations than IPv4. Inclusion of necessary functionality (e.g., ARP and DHCP) that were previously outside the IPv4 framework eliminates what is essentially a second stack of communication functionality. The structure of IPv6 addresses are more amenable to cross-layer compaction. Autoconf and ICMPv6 were designed to address scalability, visibility, and unattended operation, all features necessary in production sensornets. IPv6 does require some adaptation to effectively support sensornets, but fortunately, IPv6 was designed with extensibility in mind, supporting a simple header options framework in almost every core protocol.

1.2 Contributions

Prior work showed the feasibility of an RFC-compliant IP stack for small embedded devices [40]. While they have been adapted for use in sensornets, those proof-of-concept implementations did not take advantage of the numerous mechanisms developed within the sensornet community. But seeing that it was possible to implement IP in sensornets, the IETF formed the 6LoWPAN working group to map IPv6 and

supporting protocols to support an IEEE 802.15.4 link. The working group has since produced RFC 4944 that specifies how IPv6 datagrams are carried in 802.15.4 frames [79, 117]. Our initial work led to the design specified in RFC 4944.

In this dissertation, we continue our effort to complete an IPv6-based network architecture for sensornets. This dissertation provides three main contributions:

1. We develop a complete IPv6-based network architecture for sensornets that allows end-to-end communication between low-power embedded wireless devices and any IP device at the network layer.

2. We develop a software architecture that preserves IP's layered protocol model, defining the services, interfaces, and their interactions that can incorporate many of the techniques developed within the sensornet community.

3. We present the implementation of a complete, efficient, and production-quality IPv6 solution for sensornets and show that this general network architecture outperforms existing sensornet systems, even though they do not adhere to any particular architecture or standard. This implementation incorporates many of the mechanisms developed within the sensornet community.

1.3 Roadmap

We begin our journey in Chapter 2 by describing a number of typical sensornet application classes, deriving a common set of application requirements from those classes, and identify networking challenges that set sensornets apart from more traditional networks. We then give an overview of the developments in sensornet research, IPv6, and IEEE 802.15.4 and argue how these developments enable an IPv6-based network architecture for sensornets today.

In Chapter 3, we present an overview of our IPv6-based network architecture for sensornets, introducing the network components as well as a software organization that preserves the layered model but increases cross-layer visibility. We then discuss our IP link model, which we equate with radio neighbors in order to reduce complexity at the link layer while giving layers above necessary visibility into the radio

connectivity for building effective topologies. We also present our IPv6 addressing and prefix model and show how simple constraints can reduce both memory and communication costs.

In Chapter 4, we present the design of a power-efficient link layer that supports effective datagram delivery by the IP network layer. Using sensornet techniques, we show how to achieve duty-cycles less than 1% while appearing always-on, low-latency communication, and high "best-effort" datagram delivery at the network layer. We also define a software abstraction that enables cross-layer coordination while preserving the layering, allowing the link layer to optimize transmissions based on network-layer state and the network layer to efficiently deliver datagrams.

In Chapter 5, we present the use an adaptation layer between the link and network layers that adapts IPv6 datagrams for efficient transmission within IEEE 802.15.4 frames. We show how to support the IPv6 1280-byte minimum MTU requirement over a 127-byte MTU with IEEE 802.15.4 and reduce header overhead by using shared-context compression that does not maintain per-flow state. We also describe how to express this functionality using simple and compact encodings.

The network layer has three basic components: configuration and management, forwarding, and routing. In Chapter 6, we presenting configuration and management mechanisms that are paramount to sensornets in production but not well addressed within the sensornet community. While IPv6 had the right concepts, the solutions were not quite right because they are only defined to operate over a single IP link that provides a single broadcast domain. We show how to adapt IPv6 Neighbor Discovery, Stateless Address Autoconfiguration (SLAAC), and DHCPv6 to support multihop networks with overlapping link-local scopes and incorporate sensornet techniques to lower communication overhead.

Our design maintains the separation between forwarding and routing. In Chapter 7, we present a forwarder. Through cooperation with the link layer, we show how the forwarder can be energy efficient while achieving high "best-effort" datagram delivery in a low-power wireless mulithop network. The forwarder incorporates many sensornet techniques to achieve our goals, including hop-by-hop retransmission and re-routing, hop-by-hop congestion control, streaming, and QoS.

In Chapter 8 we present a baseline routing protocol for typical sensornet constraints and workloads. We show how the routing protocol can effectively infer a connectivity graph and construct a routing topology with $O(1)$ per-node state for sensornet nodes and $O(N)$ network-wide communication costs. We show how the router avoids generating control traffic by piggybacking on ambient application traffic to communicate routing information, generate link quality estimates, enable loop and suboptimal route detection, and provide border routers with routing information.

In Chapter 9 we present a complete production-quality implementation that includes all of the functionality described in this dissertation as well as RFC-compliant UDP and TCP at the transport layer. We show what is possible to achieve with our design by building a power model using empirical data and validating the power model using a real-world home monitoring application deployment. Comparing the results to existing systems that do not adhere to any particular architecture or standard, we show that this implementation achieves lower energy cost, lower per-hop latency, and higher data reception rate while communicating more data.

Finally, we conclude with Chapter 10. In light of this demonstration of full IPv6 capability, we review the central arguments that led the sensornet community away from IP. We briefly mention the impact our work has already made on standards efforts within the IETF and other organizations as well as in academia. We believe that the presence of an architecture, specifically an IPv6-based one, provides a strong foundation for sensornets going forward.

Chapter 2

Background

Sensornet research began to take shape nearly a decade ago. Since then, the sensornet field has matured significantly with better understandings of the inherent requirements of low-power wireless networks and their applications. During that same time period, both the IETF and IEEE standards organizations have made significant advances. We have seen the introduction of IEEE 802.15.4, specifically designed for the low power, low cost, and large scale demands of sensornet applications. We have also seen the introduction of IPv6 and the necessary protocols and mechanisms needed to support the rapid growth of IP networks. Significant advances in both link and IP networking technologies, combined with research advances, are what makes it feasible to truly integrate sensornets into the Internet architecture today. In this chapter, we provide background on each of the advancements that serve as a foundation for this dissertation. But first, we start with describing a number of applications that motivate the use of sensornets.

2.1 A New Class of Computing

Gordon Bell observed that a new class of computing emerges about every decade and coined Bell's Law of Computer Classes [9]. Each new class of computing often represents a disruptive technology enabled by technical advances that allow lower cost and smaller computing devices. So far, we have seen the progres-

sion through mainframes, minicomputers, personal computers, handheld devices, and embedded computing. Sensornets represent the next class of computing. With breakthroughs in low-cost and low-power microcontrollers and radios, sensornets enable new applications that require large numbers of nodes embedded in the physical environment.

2.1.1 Typical Applications

Over the past decade, a number of applications have been proposed, deployed, and evaluated. This experience gives us a better understanding of the application requirements, which translate directly into networking challenges. These application requirements differ from more traditional IP networks. In this section, we present a number of sensornet applications to provide a better understanding of the inherent networking challenges.

Environmental Monitoring

Environmental monitoring applications include the use of sensornets in agriculture to provide greater information about soil conditions; biological studies to achieve a better understanding of the behavior of elusive birds [109, 161] or provide greater understanding of microclimates within forests [163, 170]; and to perform geological studies of volcanos [181]. These applications typically require slow periodic reporting of environmental data (e.g., temperature, occupancy) to a central server. Asynchronous events may also be reported by individual sensors for interesting events (e.g., vibrations that appear abnormal). Occasional communication from the server to the nodes is required for configuration or time synchronization. These applications are typically latency tolerant and occasional loss in delivered data is often acceptable.

Structural Monitoring and Condition-Based Maintenance

Structural monitoring and condition-based maintenance applications involve high frequency sampling, reporting, and processing of vibration data to help predict failures of factory machines, bridges, or buildings [97, 99, 187]. High frequency components in the vibration signal require sensor nodes to sam-

ple at relatively high frequencies and communicate more information to a central server than environmental monitoring applications. Like environmental monitoring, reporting latency is generally lenient. But delivery reliability is often higher due to the requirements of signal processing algorithms. Some applications require tight time synchronization to correlate time series data across nodes placed at different sense points on the same structure. Occasional communication from the server to the nodes is required for configuration and often for reliable transport protocols.

Home and Building Automation

Home and building automation applications seek to eliminate the wiring involved in controlling the building environment [155]. Examples include wireless lighting switches to control individual or large groups of lighting fixtures, wireless thermostats to control HVAC systems, security systems that monitor door positions, occupancy, smoke alarms, and even energy monitoring devices coupled with power control to reduce overall energy consumption. Like environmental monitoring, home and building automation often involves the slow periodic reporting of environmental data with asynchronous events. However, home and building automation differ in several ways. First, communication tends to be bidirectional in nature, as automation applications often involve both sensing and actuation. Second, communication latencies must occur at human time scales (e.g., control of a light from a light switch). Third, nodes may communicate with multiple nodes simultaneously, to support a single input device that controls multiple actuators (e.g., an array of light ballasts in a commercial building). Finally, devices may communicate directly with other nearby devices, in addition to a central controller. Direct communication reduces latency and communication costs, as controllers and actuators tend to have spatial locality (e.g., a light switch and a light in the same room).

Industrial Automation

Industrial automation applications involve the monitoring and control of plant machinery or the processes used to manufacture items. Industrial monitoring applications have varying degrees of network requirements. Most applications require slow periodic reporting of sensor values that is latency tolerant

in addition to occasional low-latency communication of alarms. Some applications may require periodic offloading of historical logs. Often, control is served by a human in the loop, which inspects data retrieved by a central server and issues commands to individual actuators as needed. Closed-loop control may be implemented by programming logic into the central collection point. Some envision the use of sensornets for closed-loop control that require quick response times, and this may involve the direct communication of nearby nodes to satisfy the latency requirements. Like all of the applications we've discussed so far, the physical placement of nodes are static. However, industrial automation applications generally require greater reliability for delivered data and may involve both latency-tolerant and low-latency communication.

Asset Tracking

Sensornets have been proposed to assist in providing continuous information about the location of mobile assets and sensed information about the asset itself. The concept of asset tracking spans a number of industries, including hospitals and health care to monitor the location of various medical machines, shipping and logistics to locate packages or containers and provide sensed information about their contents, or security applications for verifying the location of high value items. Unlike other applications we have discussed so far, asset tracking involves a large number of mobile nodes. The mobile nodes typically infer location based on radio connectivity with stationary nodes that have well-known physical locations. These stationary nodes also form the communication infrastructure for mobile nodes and generally have constant network connectivity with other stationary nodes, but mobile nodes may only have intermittent connectivity depending on whether or not they're in range of the infrastructure. Using these stationary nodes, the mobile nodes may offload historical logs of sensor values (e.g. temperature, humidity, and vibration).

Urban Networks

Urban networks are much like asset tracking in that a set of stationary infrastructure nodes are deployed and mobile nodes communicate through the infrastructure when they have connectivity [47, 80]. Urban networks can host a number of applications, including failure detection for street lights or underground

water pipes, monitor air quality, assist in street parking applications, and provide information and increase efficiency in traffic applications. The vision is for urban networks to be deployed across entire municipalities and their infrastructure.

2.1.2 Application Requirements

From the applications we have identified in Section 2.1.1, there are three fundamental requirements that set sensornets apart from other more traditional networks: (i) *embedding in physical space*, (ii) *large numbers of nodes*, and (iii) *low total cost of ownership*.

Embedding in Physical Space

The applications that sensornets are intended to address involve specific placement of nodes near the sense or actuation point. Unlike nodes in more traditional networks, sensornet nodes often cannot be placed where power or network connectivity is plentiful or convenient. Their embedding in physical space places strict requirements on the sensornet. First, sensornets must operate wirelessly, being powered by autonomous sources and communicate using radios. Wireless operation gives the flexibility needed to embed sensornet nodes in appropriate locations and support mobile applications. Second, the physical location often limits the size of sensornet nodes. The autonomous power source (e.g., batteries) is often the largest component of the system, and size constraints translate directly to limited power sources. Physical constraints may limit the size of batteries or the ability to harvest energy through solar panels, vibration, or other methods.

Large Numbers of Nodes

The promise of sensornets is that they can provide a high density of sense and actuation points. A single application deployment may have hundreds or thousands of nodes, and typically operate under a single administrative domain. This is in contrast to most other wireless networks. Today, WiFi networks are dominated by laptops and handheld devices where the ratio of nodes to users remains near one to one.

Lower power radio networks, such as Bluetooth, target a ratio near seven to one. However, the IEEE 802.15.4 standard developed for sensornets targeted a completely different operating point of thousands to one.

Low Total Cost of Ownership

With large numbers of nodes, the per-node cost of implementing, deploying, and maintaining a sensornet must be reasonable. The requirement for wireless operation is reinforced by the need to minimize installation costs. Constrained costs limit the resources at each node (e.g., energy capacity and memory resources). To limit maintenance costs, nodes must operate unattended for multi-year lifetimes on modest power sources. Simple configuration and management capabilities are also needed to reduce installation and management costs.

2.2 Networking Challenges

The three basic application requirements of embedding in physical space, large numbers of nodes, and low total cost of ownership impose a large set of networking challenges. Low-power wireless radios have shorter range, higher loss rates, and more dynamic link qualities than high-power radios such as WiFi. Limited memory constrains the amount of routing state, forwarding buffers, or neighbor information each node can maintain. Unpredictable deployment properties require agile protocols that scale with a wide range of node densities. In this section, we discuss the inherent networking challenges for sensornets.

At its core, a sensornet node is composed of a microcontroller (MCU), a low-power radio transceiver, and a power supply, as shown in Figure 2.1. Due to power and cost constraints, the MCUs and radios used in most sensornet applications are relatively simple and provide fewer resources than those in other computing classes. By implementing fewer capabilities, the devices draw less power in active and sleep modes and are cheaper to produce. The simplicity also allows the MCU and radio to quickly transition between sleep and active modes, a critical property for achieving low-power operation of the complete system. Most sensornet applications require less than 1% utilization of both the MCU and radio to meet their energy budget. To

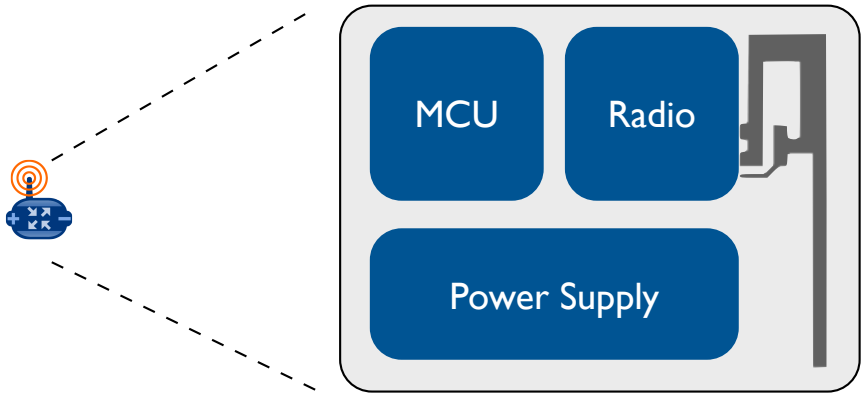


Figure 2.1: **Sensornet Node Architecture.** A typical sensornet node is composed of a microcontroller, low-power radio transceiver, and a power supply.

achieve lowest system power, most of the hardware modules must remain in the sleep state as much as possible. When the hardware module must be used, it should wake up, perform the required work, and return to sleep as quickly as possible.

In the following sections, we present representative MCUs and radios used in sensornets to provide a better understanding of the hardware constraints. We also discuss specific networking challenges that arise from such constraints.

2.2.1 Low-Power Radio Constraints

There are a number of low-power wireless radios currently available. Many have been designed for simple wireless applications, such as wireless game controllers, keyless door entries, and garage door openers. Like sensornets, these applications are also constrained in both energy capacity and cost. Many low-power radios are narrow-band radios, as they are generally lower cost, draw less power, and have faster wakeup times. The introduction of IEEE 802.15.4 in 2003 has made wide-band radios feasible for sensornet applications. Wide-band radios provide greater robustness to narrow-band interference, but comes at the cost of greater energy consumption and complexity. Typical low-power radios used in sensornet applications are shown in Table 2.1.

Make	Model	Year	802.15.4	Band	Data Rate (kbps)	Vcc (V)	TX (mA/dBm)	RX (mA)
Atmel	ATA5429	2007	No	916 MHz	20	2.4-3.6	19.3/10	27
Freescale	MC13191	2004	Yes	2.4 GHz	250	2.0-3.4	30/0	37
Nordic	nRF2401	2002	No	2.4 GHz	1000	1.9-3.6	13/0	18
RFM	TR1000	1998	No	916 MHz	115.2	2.2-3.7	12/1.5	3.8
TI	CC1000	2002	No	916 MHz	76.8	2.1-3.6	16.5/10	9.6
TI	CC1101	2007	No	916 MHz	500	1.8-3.6	15/10	14
TI	CC2420	2003	Yes	2.4 GHz	250	2.1-3.6	17.4/0	19.7
TI	CC2500	2007	No	2.4 GHz	500	1.8-3.6	21.6/1	12.8

Table 2.1: **Properties of Representative Radios Used in Sensornets.** Properties common to all low-power radio transceivers are the relatively low data rate and near-equal current consumption for receiving when compared to transmitting. IEEE 802.15.4 radios brought wide-band radios into the mainstream for sensornet applications and provide greater robustness to narrow-band interference, but come at the cost of higher complexity and additional power requirements.

All low-power radios share three important characteristics: low throughput, limited transmission output power, and significant receive power requirements. We discuss each in detail below:

Low Throughput

Low-power radios have relatively low data rates, most operating below 500 kbps, some operating down to 20 kbps. The power constraints and relative simplicity of low-power radios result in communication throughput that is generally less than what is capable over higher power wireless links. As we discuss in Section 2.2.2, attached MCUs also have limited buffering and computation capacity. So even if the radios could support a higher data rate, the attached MCUs will likely not be able to consume it.

Limited throughput constrains the amount of information nodes can communicate. This is especially true in a wireless environment, where the wireless medium is shared between all neighboring nodes. Limited throughput and channel capacity can also significantly increase communication latency between nodes.

Limited Transmit Power

The transmit current is dependent on the transmission output power or the amount of radio frequency energy emitted by the radio. The transmission output power on current radios is often limited to

around 0 dBm (1 mW). Furthermore, energy requirements increase polynomially with range, and communicating over multiple short hops with lower transmit power allows lower aggregate energy consumption than one long hop with increased transmit power. However, limiting the transmission output power raises a number of networking challenges.

- **Multihop Communication** Lower transmission power necessitates the use of multihop communication over a wireless network to support applications that utilize nodes distributed over a large geographic area. Furthermore, the limited transmission power can make it less likely for transmissions to penetrate physical obstacles. Allowing communication through other neighboring nodes makes it more likely for a node to route around the obstacle. As a result, network protocols for sensornets must configure routes and forward datagrams over a multihop network.

- **Higher Loss Rates** Lower transmission power reduces the signal-to-noise ratio (SNR) and makes the communication more susceptible to interference from surrounding electronic devices and the environment. Lower transmission power also makes it less likely for other CSMA-based wireless devices (e.g. IEEE 802.11) to identify transmissions as a busy channel. The result is a link that is likely to achieve a lower packet reception rate (PRR) than other higher-power radios such as IEEE 802.11. More generally, low-power links are likely to have highly variable link qualities compared to higher-power radios. Network protocols must provide mechanisms to ensure reliable operation and react well to link qualities that vary over time.

- **Small MTUs** Link MTUs for low-power radios are relatively small for three reasons. First, the frequency tolerance of the crystals used with low-power radios should be relatively loose, to support cheap hardware implementations. The transmission length must be short enough so that the clock drift does not affect reception. Second, packet error rates grow polynomially with larger packet sizes and lower throughput, and low-power links generally experience higher error rates. Third, the small MTUs reflect the buffering constraints due to small memory on microcontrollers that are typically coupled with low-power radios. Small MTUs require compact protocol headers to minimize header overhead. It also

requires payloads to remain minimal to avoid datagram segmentation, which can significantly increase energy consumption.

Significant Receive Current

Low-power CMOS radios operate near the 3V range and consume around 20 mA when transmitting or receiving. For many radios, the receive current is similar to the transmit current, with some consuming more current while receiving. While the receive current is relatively low compared to higher power radios, the receiver must be duty cycled to meet the power constraints of many sensornet applications. On a pair of AA batteries providing 1800 mAh, the receiver must operate at less than 1% duty cycle to operate for 1 year. Lower duty-cycles may be necessary after factoring in the cost of transmitting, the MCU, or other components.

The challenge is to allow a low receiver duty cycle and support effective communication with low latency, all while staying within the energy and memory constraints of sensornet applications. As we discuss in Chapter 4, an effective solution for duty cycling the receiver has many tradeoffs and requires some cross-layer coordination to allow upper layers to make these tradeoffs.

2.2.2 MCU Constraints

Low-power microcontrollers are typically used to satisfy the power and cost constraints of sensornet applications. Integrating required components (e.g. program memory, RAM, analog-to-digital converters, and the CPU) together into a single chip significantly reduces manufacturing costs. Microcontrollers that also integrate low-power radios into the chip are beginning to appear, achieving significantly lower cost points than the separate chips they replace. Representative MCUs typically used in sensornet applications are shown in Table 2.2.

RAM is often the dominating factor of sleep power, and most MCUs today are limited to 10 KB or less. Some MCUs allow selective RAM retention to further reduce sleep power. While one can save the RAM contents to non-volatile storage before entering the sleep state, doing so may not provide an overall benefit.

Make	**Model**	**Year**	**Arch**	**Vcc** (V)	**ROM** (KB)	**RAM** (KB)	**Active** (mA)	**Sleep** (μA)	**Wake** (μs)
Atmel	ATmega128L	2002	8-bit	2.7-5.5	128	4	5.5	10	1
Atmel	ATmega1281	2005	8-bit	1.8-5.5	128	8	3.2	1	8.4
Atmel	ATmega2561	2005	8-bit	1.8-5.5	256	8	3.2	1	8.4
Ember	EM250	2006	16-bit	2.1-3.6	128	5	7	1.5	1000
Freescale	HCS08	2003	8-bit	2.7-5.5	60	4	7.4	1	10
Freescale	MC13213	2007	8-bit	2.0-3.4	60	4	6.5	35	10
Jennic	JN5121	2005	32-bit	2.2-3.6	128	96	4.2	28.5	2500
Jennic	JN5139	2007	32-bit	2.2-3.6	128	96	2.7	3.3	2500
TI	CC2430	2007	8-bit	2.0-3.6	128	8	5.1	0.5	4*
TI	MSP430F1611	2004	16-bit	1.8-3.6	48	10	2	1	6
TI	MSP430F2618	2007	16-bit	1.8-3.6	116	8	0.5	0.5	1

Table 2.2: **Properties of Representative Microcontrollers Used in Sensornets.** Key considerations for most sensornet applications are the supply voltage (Vcc), program memory (ROM), random access memory (RAM), current consumption in active and sleep modes, as well as the wakeup time from sleep to active mode. For MCUs that offer more than one low-power mode, we chose the lowest power mode that enables wakeup from a timer and complete RAM retention. Note that the CC2430 only supports at most 4KB RAM retention in sleep mode. Many MCU families present a range of ROM and RAM configurations, we only present the models with the largest ROM/RAM configurations here.

Writing to non-volatile storage has a high energy cost, flash or EEPROM storage have a limited number of write cycles, and the wakeup/sleep time is significantly increased due to RAM save/restore operations. To effectively use selective RAM retention, the system software must utilize little RAM when sleeping and discard any temporary state. While the Jennic JN5139 allows 96KB of RAM retention in sleep state, it does so at the cost of a wakeup time three orders of magnitude larger than other MCUs.

Limited memory poses significant networking challenges for sensornets. First, limited memory constrains the amount of routing state that can be maintained. Link-state routing protocols are widely used today due to their low convergence times and robustness. However, link-state protocols are generally infeasible in sensornets, as there is not enough memory to maintain state about all links within the network. In some cases, a node may not even be able to maintain state about each link to its neighbors within radio range, much less than the whole network. Second, memory constrains the buffer size available for forwarding. Limited queuing requires greater attention to limit the number of queue overflows, which is especially critical in sensornets due to resource constraints. Every dropped packet represents a waste in energy and channel capacity already used to partially deliver the message. Third, memory constrains the amount of state the link layer can

maintain for neighboring nodes. The link may need to maintain scheduling information to optimize transmissions to duty cycled receivers or to maintain link quality statics for use by the network layer in computing routes.

Most other hardware components are essentially stateless. As a result, the choice of including other hardware components is dominated by cost, rather than power consumption. The power draw for program memory is characterized by the number of reads and writes, rather than its size. For most sensornet applications so far, 128KB has been more than sufficient. The energy cost of other hardware peripherals, such as bus controllers, DMA controllers, and ADCs depend on how often they are used. Many of these additional hardware components are typically used only for a small fraction of time.

2.2.3 Deployment Constraints

The required location of sensing often dictates the physical placement of nodes. This is in contrast to more traditional networks where physical placement is based on where power or network connectivity is good. As a result, it becomes more difficult to engineer network connectivity for typical sensornet applications. The node density can be highly variable, depending on the application's sensing locations or node mobility. Nodes may need to communicate near or through physical objects that have different effects on radio communication, such as absorption or multipath fading. Electromagnetic interference may be caused by other sensornet nodes or nearby electronic devices that may or may not be related to the sensornet deployment. The interference may either be long-term or intermittent. For these reasons, communication loss rates are often substantial, highly dynamic, and unpredictable even when nodes are stationary. While nearby nodes generally provide higher quality links due to a higher signal-to-noise ratio, the environmental and deployment factors may render higher link qualities for nodes further away.

With unpredictable deployment properties, protocols must infer connectivity with neighboring nodes based on observed properties in communicating with them. The networking should take into account both short-term and long-term changes in connectivity characteristics when configuring routes and forwarding datagrams. This is especially challenging when limited energy constrains the number of transmissions

you can use to infer radio connectivity with neighbors and limited memory constrains the amount of state used to evaluate a link.

With large numbers of nodes, autoconfiguration and management are paramount, especially in unpredictable environments. Nodes should be able to learn network parameters from neighboring nodes and configure themselves appropriately. Nodes must also provide effective management capability, so a network administrator can effectively diagnose and debug a networking problem. To be most effective, management operations must be robust and response times must be within human time scales. As a result, while some sensornet applications may only require fairly infrequent data reporting, the network should remain manageable and responsive.

2.2.4 Looking Forward

Given the cost and power constraints of sensornet applications, we should not expect to see program space, memory, and wireless communication capabilities to increase dramatically in the near future. With more traditional computing devices, we've come to expect that improvements following Moore's law will result in increased capacity with faster clock speeds at the same cost. In sensornets, however, any advances for the foreseeable future will be used to lower cost and power, rather than increase resources. New versions of MCUs and radios introduced within the past several years are following a trend of smaller die sizes, that ultimately translate into lower power and cost. Vendors are just beginning to integrate the MCU and radio, which will further reduce cost.

Historically, the capabilities of autonomous power sources have been increasing slowly at best. Energy density of battery technologies have only improved three fold between 1990 and 2004 [159]. Energy harvesting techniques, such as solar and vibration, are also improving at a similar rate, far slower than most of the computing industry.

As a result, sensornet application requirements will continue to constrain node capabilities for the foreseeable future. To effectively support the sensornet application requirements, we must tackle the

networking challenges described in this section. Fortunately, there has already been significant work in this area over the past decade.

2.3 A Decade of Advance

The significant networking challenges imposed by the sensornet application requirements led many to believe that the Internet architecture and Internet Protocol in particular was ill-suited for sensornet applications. A decade has passed since those initial claims and significant advances have been made in sensornet research, link technologies, and the Internet architecture. As we show in this dissertation, the advances in these three areas make it possible to consider extending the Internet architecture to support sensornets. In this section, we briefly describe those important advances.

2.3.1 Sensornet Research

Without being constrained to any architecture, researchers were free to tackle any portion of the networking problem in an unconstrained assault. There has been significant work that fall into particular layers of the traditional networking stack. However, many efforts span multiple layers and only portions of some layers. For example, Fusion is a congestion control protocol that implements a prioritized media-access link along with hop-by-hop and source-based flow control [81]. MultihopLQI is a routing protocol that makes direct use of IEEE 802.15.4 physical-layer information to compute routes [140]. Many applications even took on the entire stack on their own, implementing and tuning every component to meet application requirements [99, 161, 170, 181].

A number of common techniques have emerged from these research efforts. In this dissertation, we build on many of these techniques and we briefly summarize them here. In the following chapters, we also provide background on individual sensornet techniques when using them to solve specific problems.

- **Increased Cross-layer Cooperation**: In order to meet the strict resource requirements of sensornets, increased cooperation between protocol layers is necessary. This is readily apparent by the number of

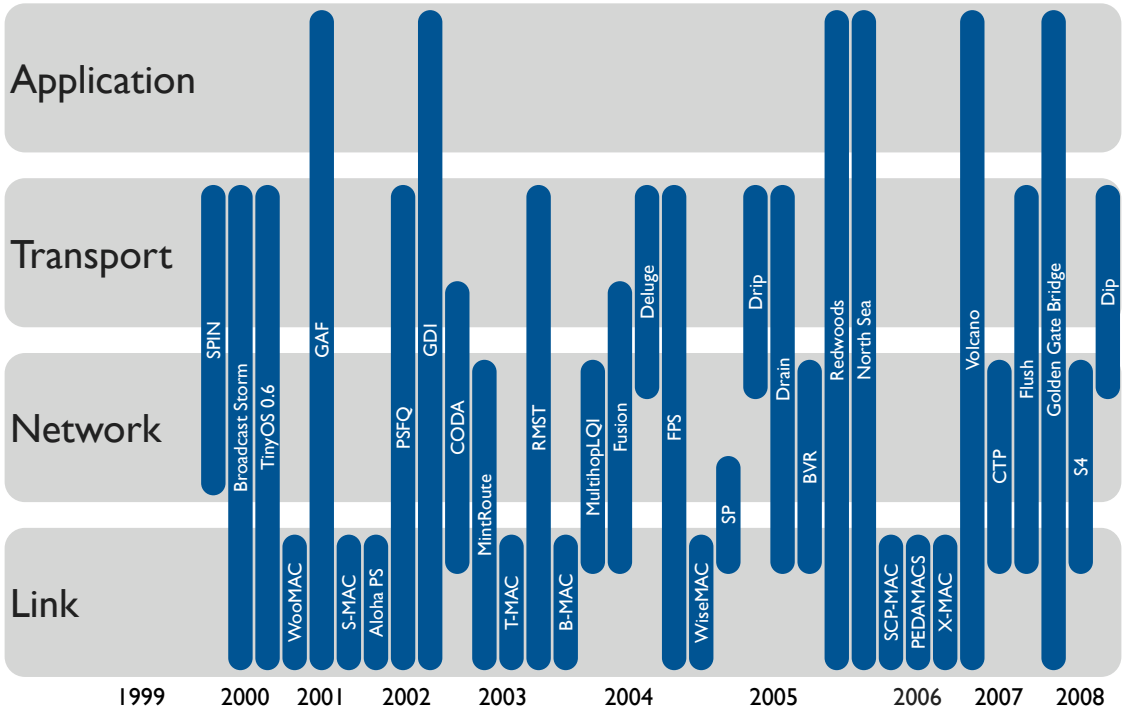


Figure 2.2: **A Decade of Sensornet Research.** This figure shows a subset of the sensornet research since it began to take shape nearly a decade ago. The lack of architecture allowed researches to address arbitrary subsets of the networking problem, from the link to the application. While a large amount of knowledge has been gained, the lack of architecture has made it difficult to build on the work of others.

research efforts that span multiple layers of the traditional networking stack, the most extreme of which are applications with the entire stack tuned to the specific application.

- **Structured Communication Flows**: Unlike traditional IP networking, most typical sensornet applications communicate in a one-to-many and many-to-one fashion. Nodes that do communicate directly with other nodes often do so with other nodes that are in physical proximity. As a result, many sensornet networking protocols take advantage of the structured flows to make the routing cost more manageable.

- **Duty-Cycled Link Protocols**: In active mode, the radio typically consumes significantly more than any other component on a sensornet node. Low-power wireless radios consume as much power when receiving, or even just listening, when compared to transmitting. As a result, there has been significant effort to reduce the listening and communication cost by using sampled listening and scheduling techniques to duty cycle the radio.

- **Link Estimation for Routing**: Because links can have widely varying link qualities, protocols must learn and estimate the quality of links and factor those estimates into the routing cost. Many sensornet routing protocols directly incorporate link quality estimation, even utilizing information from the physical layer such as received signal strength or the IEEE 802.15.4 link quality indicator. Utilizing link information in routing is a good example of increased cross-layer cooperation.

- **Controlled Floods**: The advantage of broadcast communication is that it can be efficient - a node can communicate a single message to all of its neighbors with the cost of a single transmission. However, because the broadcast medium is shared, supporting one-to-many delivery can result in the *broadcast storm* problem [123]. The Trickle algorithm was developed to tackle this problem and operates as a completely distributed protocol with almost no state [103]. A number of protocols have employed the use of Trickle to reduce power and adapt to node density.

- **Hop-by-hop Feedback**: Hop-by-hop mechanisms are used extensively for reliability and flow control in sensornets. Compared to end-to-end mechanisms (e.g. TCP), hop-by-hop mechanisms have greater visibility of conditions in the network, more capable of addressing high variance in link qualities across the network, and does not require costly end-to-end communication. However, hop-by-hop mechanisms do require bidirectional links.

- **Piggybacking**: Each frame transmission carries significant overhead. Link and network headers are often relatively large compared to the payload. Media access may require channel assessments and backoffs if a CSMA-based MAC is used. When the radio is duty cycled, the per-frame transmission overhead can be even more significant. Each frame can incur significant transmission delays while waiting for the receiver to turn on, wakeup costs for radios can be significant, and preamble sampling techniques may increase transmission overhead. Using piggybacking to coalesce communicated information into a smaller number of frames can significantly reduce energy consumption, increase throughput, and reduce channel utilization.

In general, sensornet research has focused much more on network protocol algorithms and mechanisms, rather than on networking in the broader sense. Much of this work did not constrain themselves to an architecture, leading to the development of numerous application-specific networking protocols that were hard to compose. Existing work sought to provide a communication architecture that could combine these disjoint networking protocols by placing the narrow-waist of the protocol stack at the link layer [41, 46, 138]. But by being agnostic to the network protocol in use, these architectures do not define broader networking issues of discovery, naming, addressing, configuration, and management.

Instead, based on our work, we believe that a communication architecture for sensornets should keep the narrow-waist at the network layer. IPv6 provides such an architecture: the IPv6 forms of layering, addressing, header formats, configuration, management, routing, and forwarding provide the necessary structure for designing and implementing mechanisms at all layers of the stack. The presence of an architecture allows designers and implementors to focus and improve the implementation in the process. We have seen such benefits with other widely-adopted architectures such as RISC and UNIX. In this dissertation, we show how to adapt the IPv6 network architecture to sensornets and incorporate many sensornet techniques that have emerged over the past decade. This dissertation demonstrates what is possible to achieve with an IPv6-based network architecture.

2.3.2 IP for Embedded Devices

The trend towards connecting embedded devices to the Internet have also given rise to numerous IPv4 and IPv6 stacks designed for limited memory and computation capabilities. However, many of these stacks are concerned with host-only operation, initially designed to support operation on wired networks. Furthermore, they often achieved small footprint through application-specific optimizations, sacrificing generality and RFC-compliance [10].

In 2003, uIP changed the perception that an RFC-compliant IP stack was too heavyweight for small embedded devices and demonstrated the feasibility of such a stack stack for 8-bit microcontrollers [40]. uIP has evolved to include a low-power link built on IEEE 802.15.4, showing that IP was feasible for sensor-

nets. However, the proof-of-concept implementation did not take advantage of the numerous mechanisms developed within the sensornet community [42].

Seeing that it was possible to implement IP on small devices, the IETF formed the 6LoWPAN working group in 2004 to map IPv6 and supporting protocols to low-power wireless nodes using an IEEE 802.15.4 interface. The working group has since produced RFC 4944 that specifies how IPv6 datagrams are carried in 802.15.4 frames, supporting fragmentation and header compression [79, 117]. Our initial work significantly influenced the design of RFC 4944. In this dissertation, we continue our effort to complete an IPv6-based network architecture for sensornets.

Most recently, MSRLab6 [82] and NanoStack [154] validated the feasibility of RFC 4944 in sensornets, but do not completely address broader issues of the IPv6 network architecture (e.g., configuration and management, forwarding, and routing). Mayer et. al. outlined possible ways to support these broader issues, but do not validate their thoughts using an actual implementation [112]. None of these address operation over duty-cycled links nor do they incorporate existing sensornet techniques. All of them also emulated a single IP link that spans the entire sensornet, which requires additional configuration, forwarding, and routing mechanisms at the link layer. In this dissertation, we develop a complete IPv6-based network architecture, that maintains configuration, forwarding, and routing within the network layer. Using many existing sensornet techniques, we show that our implementation can outperform existing systems that do not adhere to any particular architecture or standard.

2.3.3 IEEE 802.15.4

The IEEE works on a number of wireless technologies, in addition to wired ones, but each targets different market segments. The IEEE 802.15 working group focuses on developing standards for Personal Area Networks (PANs) or short distance wireless networks. In 2000, the 802.15.4 task group was chartered to "investigate a low data rate solution with multi-month to multi-year battery life and very low complexity." In addition to 802.15.4, the 802.15 working group works on other wireless PAN technologies that primarily

	802.15.4	**802.15.1**	**802.15.3**	**802.11**	**802.3**
Classification	WPAN	WPAN	WPAN	WLAN	LAN
Battery Life (days)	100-1000+	1-7	Powered	0.1-5	Powered
Nodes	65,535	7	243	30	1024
Bandwidth (kbps)	20-250	720	11,000+	11,000+	100,000+
Range (meters)	1-75+	1-10+	10	1-100	185 (wired)
Design Goals	Low Power, Low Cost, Large Scale	Cable Replacement	Cable Replacement	Throughput	Throughput

Table 2.3: **Comparison of Link Technology Design Goals.** This table shows the original design goals in developing the respective standards. IEEE 802.15.4 is the only link technology that attempts to achieve large numbers of nodes operating for multi-year lifetimes with simple, low cost implementations.

target cable replacement for modest sized networks and with relatively short communication range. The design goals for various task groups within the IEEE are shown in Table 2.3.

In 2003, the IEEE 802.15.4 standard was published and radio vendors began shipping 802.15.4-compliant radios shortly after. The 802.15.4 physical layer operates in three different bands (868 MHz, 915 MHz, and 2.4G Hz) and three different bit rates (20, 40, and 250 kbps). The 2.4 GHz band specifies O-QPSK modulation, while lower bands may also use BPSK and ASK modulation. All bands employ either direct sequence spread spectrum (DSSS) or parallel sequence spread spectrum (PSSS). Today, the vast majority of chips shipped support only the 2.4 GHz option with O-QPSK and DSSS. While DSSS with O-QPSK modulation makes the links more robust to narrow-band interference, the 2.4 GHz option shares the same band as 802.11 radios.

The IEEE 802.15.4 MAC layer specifies two modes of addressing: (i) an extended 64-bit address using IEEE EUI-64 MAC addresses which are globally unique and (ii) short 16-bit addresses which must only be unique within a PAN. All nodes within a PAN share a 16-bit PAN ID. The 802.15.4 MAC also specifies AES-128 authentication and encryption to support link-layer security. The amount of security information carried within each frame can be varied and represents a tradeoff between security strength and header overhead.

The IEEE 802.15.4 MAC layer specifies a maximum transmission unit (MTU) of 127 bytes, not including the frame's length byte. This MTU is relatively small compared to 802.3 and 802.11 which support

a link MTU of 1500 bytes. In addition to the payload, the 127-byte MTU must also carry any link headers, including frame type, addressing, and security information. In the worst case (with 64-bit extended addressing and 21 bytes of security information), only 81 bytes remain for the link payload. The small link MTU represents the higher error rates experienced by low-power wireless and buffering constraints on the MCUs.

Because relatively high error rates are expected, the IEEE 802.15.4 MAC also supports synchronous acknowledgments that can provide quick feedback of whether or not the frame was successfully delivered. As specified, the acknowledgment frame only reflects the sequence number of the data frame being acknowledged. No addressing information is included, expecting that tight timing is used to identify acknowledgments. In production settings, IEEE 802.15.4 acknowledgment frames are insufficient. Simultaneous transmissions and the hidden terminal problem can result in false acknowledgments. The lack of information carried in acknowledgment frames also represent a significant security threat [152]. Finally, the acknowledgment frame cannot carry any payload, removing any possibility of piggy-backing additional information within acknowledgment frames. In Chapter 4, we solve this problem by defining a new acknowledgment frame that is similar to a data frame.

There are a number of IEEE 802.15.4-compliant radios on the market today. Important properties of representative radios are shown in Table 2.4. Compared to MCUs typically coupled with IEEE 802.15.4 radios, the radio consumes more power in active mode. To achieve multi-year lifetimes, the radio must be duty-cycled. On a pair of AA batteries, the duty-cycle must be less than 1%. However, IEEE 802.15.4 only defines a limited set of power management mechanisms for edge-devices and none for devices that forward packets on behalf of others. As a result, most commercial implementations and industrial standards built on IEEE 802.15.4 forego the defined power management mechanisms when forwarding messages over multihop networks. To conserve energy at forwarding-devices, many utilize sampled listening or network-wide scheduling techniques. So far, the industry has not reached consensus on what protocol to use for duty-cycling radios. The IEEE 802.15.4e task group was recently chartered to address duty cycled operation of forwarding nodes. In Chapter 4, we seek to develop a flexible protocol that combines both sampled listening and scheduling techniques to achieve low duty-cycles while maintaining the illusion of an “always-on” link.

Make	Model	Year	VCC	Transmit		Receive		Sleep	Wake
				(mA)	(dBm)	(mA)	(dBm)	μA	(ms)
Atmel	RF230	2006	1.8-3.6	16.5	+3	15.5	-101	0.02	1.1
Freescale	MC13192	2004	2.0-3.6	30	+4	37	-92	1.0	7-20
Jennic	JN5121	2005	2.2-3.6	50	+1	45	-90	5.0	2.5
Jennic	JN5139	2007	2.2-3.6	34	+0.5	34	-97	2.8	2.5
TI	CC2420	2003	2.1-3.6	17.4	0	18.8	-95	1.0	1.0
TI	CC2430	2007	2.0-3.6	17.4	0	17.2	-92	0.5	1.0
TI	CC2520	2008	1.8-3.8	25.8	+5	18.5	-98	0.03	0.30

Table 2.4: **Properties of representative IEEE 802.15.4 radios used in sensornet applications.** The receive sensitivity specifies the minimal signal strength required for the radio to discern the signal. Transmit power specifies the maximum signal strength the radio can generate. The wakeup time specifies how long it takes for the radio to transition from sleep to receive.

While some simple non-802.15.4 radios provide a simple bit or byte-based interface, all IEEE 802.15.4 radios provide a packet-based interface. The MCUs coupled with IEEE 802.15.4 radios are expected to have limited capabilities and the 250 kbps throughput may be too high to process at a bit or byte level. Many radios implement synchronous acknowledgments and most also implement in-line AES-128 security. However, none of the radios implement any of the power management techniques, PAN formation, or PAN forwarding defined by IEEE 802.15.4. Such functionality is expected to be implemented on the MCU. This functionality separation has allowed both researchers and industry to develop alternative protocols for duty cycled operation. In Chapter 4, we take advantage of this functionality separation to develop a duty-cycled link protocol over the base functionality defined by IEEE 802.15.4.

2.3.4 IPv6

The Internet Protocol (IP) is the network layer for packet-switched internetworking across the Internet. Today, IP version 4 (IPv4) is the version most widely used across the Internet and has survived for over 30 years. IPv4 is a wildly successful protocol and has done its part to allow a global internetwork to scale to billions of users with millions of nodes. However, the success of IPv4 is catching up with its limitations. At the time, 32 bits of addressing was thought to be enough. No one imagined that the Internet would ever come close to exhausting the 32 bit address space. Experts predict IPv4 address space exhaustion somewhere between 2010 and 2012 [83].

In response, the Internet Engineering Task Force, responsible for the evolution of the Internet architecture, began work on IP version 6 (IPv6), the designated successor to IPv4. The first version of the specification was published in 1998 as RFC 2460 [31]. IPv6 is designed to handle the continued explosive growth of the Internet. Additionally, IPv6 incorporates many of the learnings gained over the 30-year tenure of IPv4. In the remainder of this section, we identify and describe key contributions of IPv6. This dissertation assumes some basic familiarity with IPv6 and the RFCs. For more background on IPv6, see [64].

Large Address Space

To address the continued growth of IP hosts, IPv6 expands the address space from 32 to 128 bits. This massive expansion is enough for 3,911,873,538,269,506,102 addresses per square meter of the Earth's surface [75]. While this may seem excessive, the IPv6 addressing architecture takes advantage of this expanded address space to support additional functionality that was not possible in IPv4.

An IPv6 address can be classified as a *unicast*, *multicast*, or *anycast* address [74]. A unicast address uniquely identifies an interface on an IPv6 node. A multicast address identifies a group of IPv6 interfaces. An anycast address is assigned to multiple IPv6 interfaces and packets sent to the anycast address are delivered to only one of the interfaces. IPv6 addresses also carry a notion of *scope*: a topological span within which the address may be used as a unique identifier for an interface or set of interfaces [30]. One important scope is the *link-local* scope, used to identify interfaces on a single link. The link-local scope is used to discover neighboring nodes and build higher-level structures (e.g., routing). Global scope is used to identify interfaces across the Internet.

Link-local unicast addresses are used when communicating with other IP hosts attached to the same link and should never be routed. Link-local addresses are composed of the *Link-Local Prefix* and the *Interface Identifier (IID)*. A link-local address does not need to be configured with a global routing prefix. As a result, IP hosts can self-assign (or autoconfigure) a link-local unicast address to an interface by deriving the IID from its link-layer address. Link-local addresses are always present and should never change and are widely used

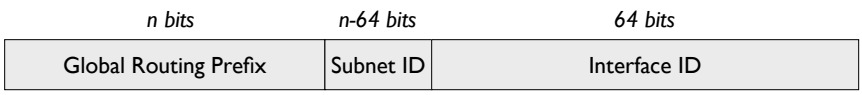


Figure 2.3: **Global Unicast Address Structure.** A global unicast address is composed of a Global Routing Prefix (uniquely identifying a site), a Subnet ID (uniquely identifying a link with the site), and an Interface Identifier (uniquely identifying an interface within a link).

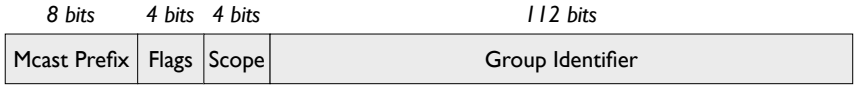


Figure 2.4: **Multicast Address Structure.** A multicast address is composed of the IPv6 Multicast Prefix (ff00::/8), a Flags field (which indicate a well-known or dynamically assigned address), a Scope field (to limit the scope of a multicast address), and a Group Identifier (which identifies the multicast group within the given scope).

in the IPv6 architecture for discovery and autoconfiguration mechanisms. Link-local addresses also allow nodes to communicate directly even when no router is present.

The global unicast prefix is `2000::/3` and the link-local unicast prefix is `FE80::/10`. Global unicast addresses are used when communicating with devices off-link. Global unicast addresses are composed of a *Global Routing Prefix*, a *Subnet ID*, and an Interface Identifier (IID). The Global Routing Prefix uniquely identifies a site, the Subnet ID uniquely identifies a link within the site, and the Interface ID uniquely identifies an interface within the link. The structure of a global unicast address is shown in Figure 2.3.

Multicast addresses are composed of the *IPv6 Multicast Prefix*, *Flags*, *Scope*, and a *Group Identifier*, as shown in Figure 2.4. The multicast scope is used to identify subsets of a Group Identifier based on the IP topology. Link-local multicast addresses are typically used to support address-free communication, useful for discovery and autoconfiguration. There are a number of well-known Group Identifiers. All IPv6 nodes are required to subscribe to the all-nodes multicast group, while all IPv6 routers are also required to subscribe to the all-routers multicast group.

Each interface is assigned multiple IPv6 addresses to support the various communication paradigms of the IPv6 addressing architecture. A properly configured interface is always assigned both a link-local unicast address and the all-nodes multicast address. Additional IPv6 addresses may be assigned to support global communication, other multicast groups, or anycast addresses.

Link-Layer Bootstrap and Discovery

IPv4 relies on a number of link-specific protocols for configuration and discovery. Address Resolution Protocol (ARP) was defined at the link layer to map link-layer addresses to IP addresses. Bootstrap Protocol (BOOTP) and Dynamic Host Configuration Protocol (DHCP) were also defined at the link layer to support IP address configuration. IPv6 folds all of the necessary bootstrap and discovery mechanisms into the IPv6 framework. Much of this functionality has been folded into the Internet Control Message Protocol (ICMPv6) [21]. DHCPv6 was defined as a UDP protocol operating on top of IPv6. ICMP was a part of IPv4, providing important health information about the network through ICMP errors and diagnostic functions such as ICMP Echo Request and Echo Reply, but did not provide a means to configure network parameters, IP addresses, or resolve link-layer addresses.

ICMPv6 introduces Neighbor Discovery (ND), which incorporates address resolution along with the ability to keep track of neighboring nodes, find routers, and propagate network and configuration parameters to other nodes on the link. Neighbor Discovery relies on the link-local addressing scope integrated into the IPv6 addressing architecture. ICMPv6 Router Advertisements allow other IP hosts on the link to configure default routes and learn configuration parameters.

Autoconfiguration

To help network administrators and users cope with the growing numbers of IP hosts, IPv6 includes extensive autoconfiguration capabilities as part of the architecture. The goal of autoconfiguration is to help ensure that no manual configuration of IP hosts is required before they can communicate on the network. This is not only important for dealing with large numbers of devices, but was also intended for the growing number of “headless” or embedded devices, such as home appliances.

IPv6 autoconfiguration supports both *stateless* [168] and *stateful* [38] modes. The latter is provided by DHCPv6, which is a new version of DHCP used extensively in IPv4 networks. But while DHCP is layer-two protocol in IPv4, DHCPv6 operates as a UDP protocol in IPv6 and utilizes link-local addressing to discover DHCP servers and multicast addressing for address-free communication. Like DHCP, DHCPv6

provides centralized management and control of network devices. Devices request configuration information for IPv6 addresses an other parameters from a DHCPv6 server and the server replies with appropriate information to each requesting node.

Stateless autoconfiguration allows IP hosts to configure their interfaces with global unicast addresses without the need of a DHCPv6 server. This functionality is not supported in IPv4 and removing the need for a DHCPv6 server can provide increased flexibility and robustness for network operation. To statelessly configure an IPv6 address, IP hosts simply learn the Global Unicast Prefix from neighboring nodes and derive an Interface Identifier from the link-layer address. This simple mechanism is made possible by the large 128-bit IPv6 address space. IPv4 also supports a limited form of stateless autoconfiguration but only for local addresses [16].

Protocol Options Framework

A protocol options framework is pervasive throughout the IPv6 architecture, and is supported by the IPv6 header as well as other IPv6 protocols such as Neighbor Discovery and DHCPv6. Allowing options support two important capabilities. First, options allows protocol extensibility. While we strive to design the perfect protocol, we often gain additional learnings after broad deployment. Furthermore, new technologies may arrive or operating environments may change that violate assumptions made in the initial protocol design. Protocol options provide a convenient path for extending functionality while providing backwards compatibility with existing devices. Second, protocol options provide a convenient mechanism for reducing the overhead of protocol fields carried in a datagram. Only those options required for communicating the information are included in the datagram, leaving a small base header common to all messages in the protocol.

Protocol options are often appended to the base protocol header using a Type-Length-Value (TLV) format. ICMPv6 Router Advertisements, for example, carry Prefix Information in an option field and is only included when stateless autoconfiguration is used to configure the network. When stateful autoconfiguration is used, Router Advertisements do not need to include Prefix Information, resulting in a smaller message.

2.4 Why Sensornets and IP Today?

The landscape for sensornet networking has changed significantly over the past decade. Significant advances in sensornet research has given us a better understanding of the inherent networking challenges for sensornet applications. The introduction of IEEE 802.15.4 gives us the first standard link designed specifically for the low power, low cost, and large numbers required by many sensornet applications. IPv6 has also seen a significant advancement, addressing scalability issues through a larger address space and autoconfiguration, greater flexibility in communication paradigms through the IPv6 addressing architecture, and the ability to extend its capabilities through protocol options.

Given the significant advances in all three areas, it is time to revisit the assumptions that formed the basis of so much sensornet research. In this dissertation, we present the design and implementation of an IPv6-based network architecture for sensornets. From this work, we find that IPv6 and sensornets are, in fact, highly complimentary. Most of the unique networking requirements of sensornets are well-served by the IPv6 architecture. Mechanisms developed in the sensornet space provide elegant solutions for problems that have not been well addressed by conventional IETF solutions. Sensornet researchers have tended to focus more on network protocol algorithms and mechanisms, rather than networking in a broader sense. The lack of a clear networking architecture for sensornets has made it difficult to create composable results and leads to ill-defined research problems. The IPv6 forms of layering, separation of routing from forwarding, configuration and management, and naming provide the missing structure and can contribute a strong foundation for sensornet research going forward.

It may seem unintuitive that IPv6 makes an IP-based architecture more attractive on sensornets, when compared to IPv4. IPv6 utilizes a much larger address space and results in significantly larger headers. IPv6 includes additional functionality not previously considered a core part of IPv4, with multicast, Neighbor Discovery, autoconfiguration, and protocols that were previously link layer specific. The protocol options framework can place additional overhead in parsing protocol headers. But it is precisely these differences that make IPv6 attractive in sensornets and make a resource constrained IP-based solution easier than in

the IPv4 setting. The large, simple address space allows link-layer addressing information to be included, removing the strict requirement for address resolution. The scoped multicast architecture naturally provides a way to implement discovery protocols or localized algorithms. Functionality designed to address the sheer numbers, unattended use, and need for ease of configuration and management are well aligned with the needs of sensornet applications. Systematic inclusion of previously link-specific services such as DHCP into the IPv6 framework increases commonality between protocols and reduces implementation overhead. Finally, the pervasive protocol options architecture provides a natural path for adapting existing IPv6 protocols to the needs of sensornets, rather than developing entirely new ones.

But even with the added benefits of IPv6, significant issues remain in order to support a complete IPv6-based network architecture in sensornets. The network layer requires a robust and effective link layer that is both low power yet IP friendly. Many IP protocols assume that IP links provide a single broadcast domain, especially configuration protocols such as IPv6 Neighbor Discovery. That link must allow the network layer to achieve high best-effort datagram delivery, while respecting the energy and memory constraints of the platform. Routing protocols must provide reachability while taking into account link dynamic qualities. Furthermore, we must find a way to apply mechanisms that have become pervasive in sensornet research to an IPv6 architecture. Of course, all of this needs to occur while maintaining the layered and structured decomposition of IP protocol stacks to provide a strong foundation moving forward. In this dissertation, we show that all of these items can be achieved while meeting the strict resource constraints and application requirements of sensornets.

2.5 Summary

In this chapter, we presented a set of typical sensornet applications from which we derived a common set of application requirements, including embedding in physical space, large numbers of nodes, and low total cost of ownership. We discussed how these few application requirements impose a wide range of networking challenges for sensornets. Low-power radios, MCU limitations, and typical deployment character-

istics require networking protocols to support multihop communication, variable loss rates, and small MTUs while fitting within strict resource limitations. These challenges are what led the research community away from an IP-based networking architecture a decade ago. But since those initial claims, sensornet research has matured significantly, a new link technology, IEEE 802.15.4, has emerged, and the Internet architecture has evolved significantly with the introduction of IPv6. The time is ripe to revisit the initial claims of whether an IP-based architecture can serve the unique needs of sensornets. In the remainder of this dissertation, we demonstrate that an IP-based architecture is, in fact, well aligned with the needs of sensornets.

Chapter 3

An IPv6-Based Network Architecture for Sensornets

In this chapter, we present an overview of our IPv6-based network architecture, outlining the network components and organization as well as the software architecture used to implement the network architecture. We then present the foundations for our network architecture for sensornets. Our IP link model equates link-local scope with radio transmission range, providing IP protocols necessary visibility into the underlying radio connectivity. We also discuss the IPv6 addressing model, defining a new sensornet scope that provides a flat namespace to the sensornet to support better static addresses even when the topology changes. Using a common prefix across the sensornet allows for efficient IPv6 header compression and smaller forwarding and routing state. In the following chapters, we walk through each layer of the protocol layering stack, starting with the link layer.

3.1 Extending the Internet Architecture

Following Bell's Law, we've seen a new class of computing emerge about every ten years. Over the past thirty years, IP and the Internet have successfully supported the emergence of each new class and

the growing numbers of IP hosts, accepting minor adaptations as necessary. IP has successfully been used to internetwork minicomputers, personal computers, laptops, cellular phones and hand held devices. Newer classes are progressively getting smaller in size, larger in numbers, and increasing mobility through wireless connectivity. Sensornets is the latest computing class to emerge, but IP has not yet been extended to support them effectively.

The goal of this dissertation is to enable this next tier of the Internet, by extending the Internet architecture to support efficient operations in a sensornet setting. In this extended Internet, nodes within the new tier should appear simply as another IP host, communicating end-to-end with any arbitrary IP host using IP. As shown in Figure 3.1, the extended Internet connects sensornet networks like any other IP network, using IP routers that route IP datagrams between different media. In this dissertation, we focus on providing IP over IEEE 802.15.4, utilizing the only industry standard link designed for the sensornet applications. However, the design of our IP architecture makes few assumptions on the specifics of IEEE 802.15.4, and should be generally applicable to other low-power radio technologies.

3.1.1 Network Components

An IEEE 802.15.4 network (commonly referred to as a Personal Area Network (PAN) within the IEEE) is defined by the set of nodes within physical proximity that communicate using a common set of link properties. Minimally, all interfaces in a given network communicate using the same PAN ID and compatible link-layer protocols. The latter includes the use of medium access mechanisms such as CSMA or TDMA, whether or not frequency hopping is in use, how the radios are duty cycled to reduce average power draw, and the specific security mechanisms used to authenticate or encrypt IEEE 802.15.4 frames. A goal of our architecture is to preserve the layered model and remain flexible to the specifics of the link-layer mechanisms.

The topology of an IEEE 802.15.4 subnet can take on many forms. In the simplest case, all nodes are within direct communication range of each other and form a single hop network. However, the short transmission range of low-power radios often requires nodes to communicate over multiple hops to provide

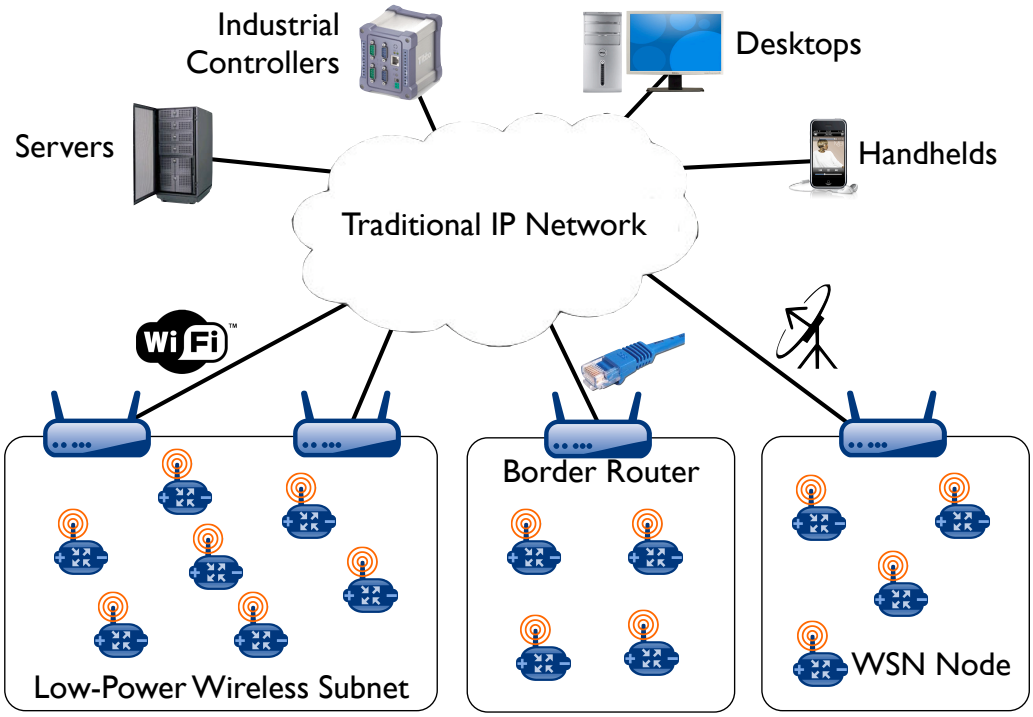


Figure 3.1: **Extending the Internet Architecture.** Communicating natively with IPv6, nodes can communicate end-to-end with each other an any arbitrary IP device over the wide-area at the network layer. Border routers connect sensornets to other IP networks using traditional wired or wireless (e.g., WiFi and satellite) links and do not require any application-specific state.

greater spatial coverage. Note that multihop subnetworks are not unique to IEEE 802.15.4 and is widely used in traditional IP links such as switched Ethernet.

An IEEE 802.15.4 subnet is composed of *edge* and/or *forwarding* devices. Multihop networks require a subset of nodes to operate as forwarders to deliver packets between nodes over multiple hops. It is valid to have a network that is composed exclusively of edge devices (in a single hop network) or forwarding devices (in both single and multihop networks). A network composed only of forwarding devices can provide greater robustness, as it gives greater choice in selecting routes to a destination. In general, most devices communicate using a single IEEE 802.15.4 interface. Forwarding devices use the same interface to both receive and transmit packets when forwarding. In typical sensornet applications, it is common to have a forwarding node also act as both an IP host and router, hosting applications while providing reachability for others in the network.

A forwarding device that also routes between two or more physically different interfaces is called a *border router*. The sensornet may be connected to other IP networks through one or more border routers that forward IP datagrams between different media, one of which provides connectivity to the sensornet. Border routers may even implement IPv4-to-IPv6 translation to support interoperability with IPv4 networks [124]. Because border routers forward datagrams at the network layer, they do not maintain any application-layer state. Other ad-hoc network architectures (e.g., ZigBee) require stateful and complex application gateways to connect sensornets to other networks. Any changes to the application protocols used in the sensornet require changes on the gateway. In contrast, border routers remain agnostic to the set of applications deployed in the network.

Supporting IP communication on all sensornet devices allows end-to-end IP communication with any arbitrary IP device, including nodes both inside and outside the same sensornet. Communication with IP devices outside the sensornet occurs through one or more sensornet border routers that provide connectivity to other IP subnetworks. IP support also allows a sensornet to take advantage of the expansive mechanisms and infrastructure that exists for IP-based networks. This includes the use of other IP links (e.g., Ethernet, WiFi, or WiMax), firewalls to define security domains, proxies to increase network efficiency, among others.

3.1.2 Layered Architecture

IP's layered protocol model means that peers communicate in terms of the mechanisms provided by the layers below. The link must allow the network to achieve high "best-effort" datagram delivery, enabling end-to-end mechanisms to achieve reliable transport, all to provide applications with an effective communication channel. The narrow waist of this architecture is the IPv6 network layer. The network layer is composed of three components: (i) configuration and discovery, (ii) forwarding, and (iii) routing.

However, IP does not specify the mechanisms used to implement those capabilities. This flexibility allows us to select the appropriate mechanisms that allow us to implement an IPv6-based network architecture in an efficient way and while the mechanisms themselves are not application-specific, their use may be. We utilize many techniques pioneered in the sensornet community and cast them in a software architecture that

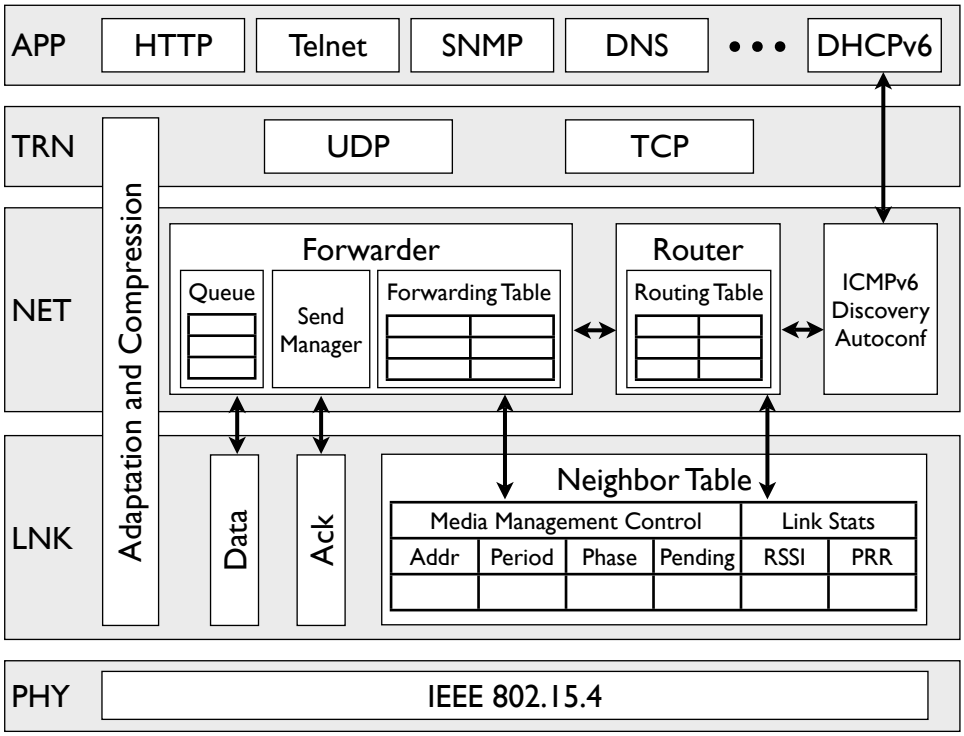


Figure 3.2: **IPv6 Software Architecture for Sensornets.** The architecture for sensornets preserves the protocol layering and structured decomposition of the traditional IPv6 architecture. The network layer represents the "narrow-waist" of the architecture.

preserves the layering and functionality separation, as shown in Figure 3.2. The following chapters in this dissertation presents the software architecture in detail, including the services, interfaces, protocols, and their interactions. We work from the bottom up, starting with the link.

3.1.3 Cross-Layer Cooperation

To meet the constraints and challenges of sensornets, the components in our architecture work together in a cooperative manner, utilizing basic mechanisms provided by other components. For example, the IP forwarder notifies the link layer when it has multiple packets queued for the same next-hop destination. Doing so allows the link to apply optimizations that increase throughput and reduce the energy cost of delivering those packets. The IP router cooperates with the link layer by indicating which neighbors will most likely be selected as next-hop destinations. Often, link layers maintain neighbor state, such as time synchro-

nization to reduce the transmission cost to those neighbors or link estimates to assist with computing route costs.

More interestingly, in Chapter 8, we show that it is possible to develop a routing protocol that does not generate any additional messages beyond those used for configuration and management as well as the application. The IP router utilizes link quality statistics computed by the link as part of the route metric and those link estimates are updated whenever the forwarder delivers a message. By intelligently updating the forwarding table, the IP router can update link estimates for neighbors without requiring additional control traffic. The IP router discovers potential default routes by snooping on ICMPv6 Router Advertisements, already required for propagating network configuration information.

Respecting the IP architecture, the interfaces used in our architecture to enable cross-layer cooperation only expose certain capabilities but do not indicate the specific mechanisms used to implement those capabilities. While forwarder indicates that it has multiple datagrams for the same destination, it does not specify how the link should deliver those datagrams. While the router indicates what neighbors it is most likely to communicate with, it does not specify how the link optimizes for those neighbors. By not specifying specific mechanisms, the architecture facilitates new developments moving forward.

3.1.4 Applying Sensornet Concepts

This dissertation represents the first effort in applying sensornet concepts to an IPv6-based network architecture. In doing so, we show that our IPv6-based network architecture can outperform any existing system that does not adhere to any architecture or standard. We see that, in many cases, the sensornet mechanisms provide elegant solutions to problems that have not been well addressed by conventional IETF approaches for low-power wireless networks. For example, we apply the Trickle algorithm to implement controlled floods for disseminating network information for autoconfiguration and multicast communication that exceeds link-local scope. We use hop-by-hop feedback to increase reliability and flow control mechanisms when forwarding datagrams. The architecture uses piggybacking of information to vastly reduce message overhead and carry network-layer information on link acknowledgments and enforce routing con-

sistency when forwarding datagrams. The architecture also relies on compaction mechanisms that rely on shared context to significantly reduce header overhead and memory requirements for forwarding and routing state.

3.2 Avoiding IP Link Emulation

The IP protocol layering architecture separates the network layer from the link layer with the expectation that protocols at or above the network layer are agnostic to the specific link layer in use below. However, the vast majority of IP links in use today provide a number of similar capabilities, many of which have been taken for granted to make simplifying assumptions when designing IP-based protocols. While the introduction of radio links (e.g. WiFi) challenged some of these assumptions, many have been adapted to support those capabilities through emulation. However, the challenges in supporting IP link emulation become more acute with low-power radio links. In this section, we characterize basic assumptions about the IP link made by IP and core protocols built above IP.

3.2.1 Traditional Assumptions

While IP has successfully adapted to include a number of link technologies as part of the networking architecture, IP and protocols built above often assume that the link layer provides three basic properties.

- **Always-On**: The IP link provides a *connectionless* communication service. Any node can communicate with any other node attached to the same IP link at any time without the need to establish a connection. The always-on property allows nodes to communicate without establishing or maintaining any link-layer state for each other and supports the ability to discover neighboring nodes with low latency, probe neighbor reachability at any time, and allow highly dynamic route updates. In general, these capabilities enable a more robust network.

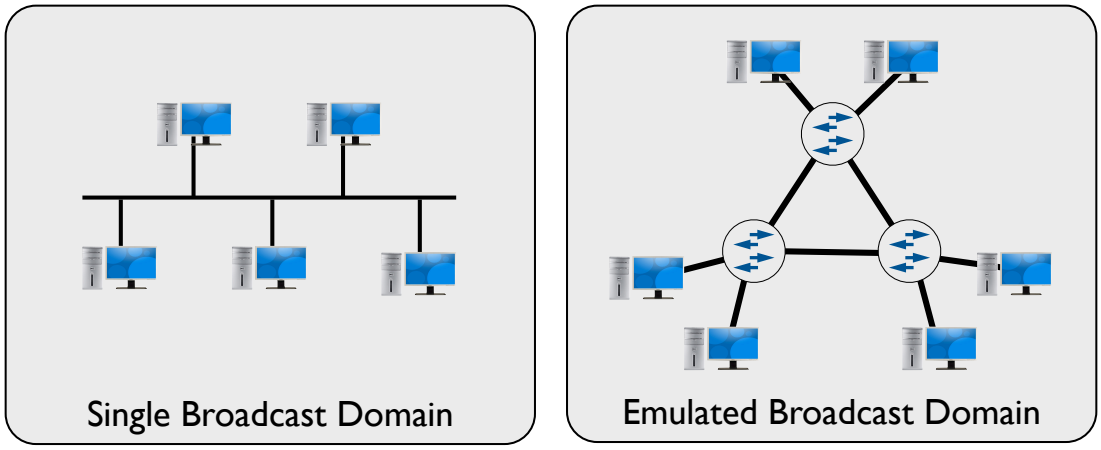
- **Best-Effort Reliability**: The link must allow the network layer to achieve high "best-effort" datagram delivery and enable end-to-end mechanisms to achieve reliable transport. Often, datagram delivery

rates must be higher than 99%. Some transport protocols (e.g. TCP) assume that any loss is due to network congestion rather than losses at the link. Furthermore, some existing protocols only validate reachability of a neighbor rather than characterizing the packet error rate on that link, including Neighbor Discovery and traditional IP routing protocols for wired networks. A modest packet error rate can cause such protocols to over-react, based on premature determination that a link is down.

- **Single Broadcast Domain**: The IP link provides *transitive* reachability for all nodes on the link. Transitive reachability implies that if node A can send packets to node B and node B can send packets to node C, then node A can send packets to node C. In effect, having transitive reachability gives every node the same view of the IP link. The single broadcast domain is commonly assumed by discovery and configuration protocols to reach all nodes on an IP link, as well as ICMPv6 redirect.

In general, these three basic assumptions are supported by the vast majority of IP links in use today. Point-to-point links trivially provide these assumptions, with only two nodes attached to a single link. Ethernet, the most wide-spread multi-point link technology in use today, also supports these fundamental assumptions. Ethernet was initially designed to provide a single broadcast medium that all attached nodes communicate over. Because Ethernet communicated over a broadcast medium, the always-on property was naturally supported. The Carrier Sense Multiple Access with Collision Detect (CSMA/CD) media-access control provided high best-effort reliability. But scalability issues forced the adoption of Ethernet switches that divided the medium into separate physical domains.

Ethernet switches expanded the physical reach of Ethernet links and increased scalability by breaking up the collision domain into multiple physical domains, as shown in Figure 3.3. However, the design of switched Ethernet was careful to maintain the original capability of a single broadcast domain. All communication begins with a flood throughout the switched Ethernet, which directly supports an always-on network and a single broadcast domain. Ethernet switches optimize non-broadcast transmissions by snooping on delivered packets and pruning branches over a spanning tree. This simple mechanism allows switched Ethernet to emulate a single broadcast domain without any perceivable difference to the layers above. While the opti-



(a) Basic Ethernet (b) Switched Ethernet

Figure 3.3: **Basic vs. Switched Ethernet.** Ethernet has always provided a single broadcast domain, even with the development of switched Ethernet. Figure (a) shows the physical topology of a basic Ethernet link, with all nodes connected to the same physical media. Figure (b) shows the physical topology of a switched Ethernet link, with nodes separated by two or more Ethernet switches. In both cases, a broadcast transmission reaches all nodes.

mizations support significant scalability, they are merely optimizations and no connection must be configured before communicating with a neighbor.

The introduction of WiFi as a LAN technology brought similar challenges. While wireless is inherently a broadcast domain, it does not naturally support a single broadcast domain. With a limited transmission area centered on each node, radio communication does not guarantee transitive reachability. Security mechanisms also make WiFi links to the access point look more like point-to-point links, providing no ability to directly communicate with other WiFi clients. WiFi solved this problem in a similar manner as switched Ethernet. A number of WiFi clients are assumed to be in direct connection with WiFi access points, and these access points are often connected through a wired backbone, as shown in Figure 3.4. The access points and clients form a spanning tree, with each node pair connected by an edge. Multihop delivery occurs over the spanning tree and broadcast communication is emulated by retransmitting the packet on each edge.

Another challenge is that WiFi's wireless communication provides inherently less reliability than switched Ethernet. Unlike Ethernet, collision detect cannot be used to detect failed transmissions. Instead, WiFi utilizes synchronous acknowledgments to ensure successful delivery to the destination, retransmitting

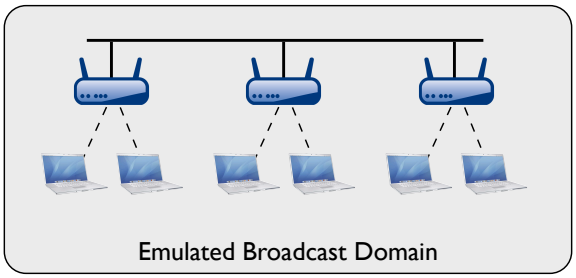


Figure 3.4: **WiFi Physical Topology.** In a typical WiFi deployment, WiFi clients are in direct radio communication with WiFi access points, and these access points are connected over a wired backbone. A single broadcast domain is emulated over the entire topology and link layer retransmissions between the clients and access points are implemented to maintain relatively high reliability.

the packet if no acknowledgment is received. As a result, while WiFi does not naturally support the common assumptions of IP links, WiFi has been generally successful in emulating those properties just switched Ethernet has.

LAN emulation is attractive when it is transparent to the upper layers - it allows the use of existing protocols that have taken the traditional IP link assumptions for granted. Switched Ethernet and WiFi are the most widely deployed LAN technologies today, and while their physical topologies do not natively support common assumptions about the IP link, emulation mechanisms have been largely successful in supporting those assumptions. This is, in large part, possible due to their significant capabilities, providing low latency, high throughput, and high capacity. Such attempts, however, have not always been met with such success, even on purely wired networks.

Asynchronous Transfer Mode (ATM) is a network technology that provides a connection-oriented, packet-switched communication service [1]. The primary goal of ATM is to provide Quality of Service (QoS) guarantees to meet the needs of real-time audio and video streaming. By being connection-oriented, ATM does not natively support the traditional IP link assumptions. First, ATM does not provide an always-on abstraction. Before two nodes can communicate, a virtual circuit between them had to be configured. Second, ATM does not support a single broadcast domain. In fact, ATM does not support any form of broadcast, and provides only point-to-point links between pairs of nodes with a virtual circuit. ATM LAN Emulation (LANE) was developed in attempt to emulate the always-on capability and a single broadcast domain by

introducing a Broadcast and Unknown Server (BUS) [5]. The BUS connected all ATM nodes by establishing a virtual circuit to each node. However, LANE configuration was cumbersome and the architecture limited scalability with a single-point of failure.

Compared to Ethernet and WiFi, ATM utilized complex routing mechanisms to establish virtual connections that meet QoS guarantees. IP over ATM operated with completely independent routing mechanisms at the link and network layers which proved to be problematic. While ATM routing could optimize paths across the ATM network, they did so without any knowledge from the networking layer above. Similarly, IP routing was unable to take direct advantage of information below, and could only infer connectivity characteristics provided by ATM. IP routing protocols could not properly optimize IP routes end-to-end, leading to proposals for exposing abstracted ATM topology information [59]. However, the larger issue is that the independent routing protocol at different layers lead to more frequent conflicts, routing loops, and unwanted interactions between different routing layers. For example, when IP routers connected over ATM are running OSPF, a single link failure can generate $O(N^4)$ traffic in response to a single link failure [1].

ATM placed too much policy and complexity at the link layer and LAN emulation was not able to make them transparent to the network layer. The introduction of Multiprotocol Label Switching (MPLS), however, made IP over ATM much more attractive by removing all of the policy from ATM and reducing it to a simple switching network [148]. MPLS computes routes at the network layer among ATM switches using standard IP routing protocols such as OSPF or IS-IS, with each ATM switch acting as an IP router. MPLS then applies those routes to ATM switches at the link layer. The result is a layer three routed ATM network that forwards datagrams at layer two, which gives IP the necessary visibility into the ATM topology that LAN emulation couldn't provide.

[1] A link failure can affect $O(N^2)$ virtual circuits and if ATM does not react quickly enough, OSPF will assume the links are down and generate $O(N^2)$ link-state updates http://www.ciscosystems.com/univercd/cc/td/doc/product/wanbu/bpx8600/mpls/9_3_1/mpls01.htm

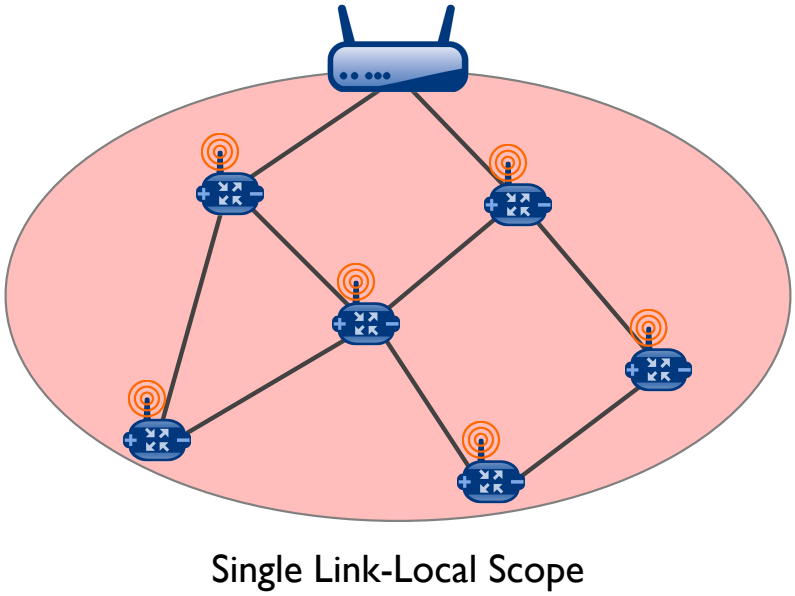


Figure 3.5: **LAN Emulation for Low-Power Wireless Networks.** LAN emulation abstracts a multihop wireless network as a single IPv6 link-local scope. Link-local IP communication requires link layer routing and forwarding. Link-local IP multicast requires the network to flood the message across the network.

3.2.2 Challenges of LAN Emulation in Sensornets

LAN emulation for sensornets would abstract the entire multihop wireless network as a single IPv6 link-local scope, as shown in Figure 3.5. At first glance, sensornets share a few properties with switched Ethernet and WiFi networks that seem attractive for LAN emulation in sensornets. First, sensornets are often multihop networks. Second, sensornets don't necessarily require connection-based communication with other sensornet nodes. Third, the wireless communication medium is naturally a broadcast medium. These properties make it seem reasonable that LAN emulation can be done effectively in sensornets.

However, sensornets differ from traditional IP links in significant ways. Switched Ethernet and WiFi transparently provide network-wide broadcast in a resource intensive way, by retransmitting messages on every link. Sensornets, however, operate with severe resource constraints, including buffer sizes, channel capacity, and energy. Furthermore, channel capacity is a shared resource between neighboring nodes, due to overlapping radio transmission regions inherent to multihop wireless networks. Naive flooding methods can lead to the *broadcast-storm* problem, where wireless retransmissions saturate the channel and reduce

the reliability of the flood itself [123]. Controlled flooding mechanisms avoid channel saturation at the cost of propagation delay or topology maintenance [103]. Furthermore, energy-conserving link layers that duty cycle the radio often optimize for unicast transmissions, resulting in relatively expensive broadcast operations when compared to unicast, as we see in Chapter 4.

Resource constraints involve more than just broadcast emulation - providing independent mechanisms at both the link and network layers potentially duplicate similar functionality. For example, LAN emulation requires configuration, discovery, and management mechanisms at the link layer, while the IP layer provides its own mechanisms through ICMPv6. With LAN emulation, the routing must occur at the link layer, as the entire sensornet appears as a single broadcast domain to the network layer above. Border routers must implement different forwarding and routing protocols for the link and network layers. Due to the challenges of low-power wireless, routes between nodes may have interesting and complex dynamics. If we learned anything from ATM LANE, transparent LAN emulation over routing protocols with complex dynamic properties will be no simple task to solve.

LAN emulation also constrains the communication paradigms available to IP protocols. The smallest addressable scope that extends beyond the local node in IPv6 is link-local. With LAN emulation, link-local scope covers the entire sensornet. IP protocols have no way of addressing or discovering neighbors within direct radio transmission range. As a result, IP protocols have no visibility into the radio connectivity and are incapable of creating higher-level structures based on that radio connectivity (e.g., routing topologies). In fact, when communicating with any node, the link layer must first join the link network and establish routes to the destination before IP can begin communicating. Being able to directly communicate with a neighboring node is useful for implementing application protocols that utilize in-network processing, as well as debugging and management.

3.2.3 Equating the IP Link with Radio Range

To effectively support IP in low-power wireless networks, we take a similar approach to MPLS, by simplifying the link and exposing only the most basic capabilities required for the network layer to operate

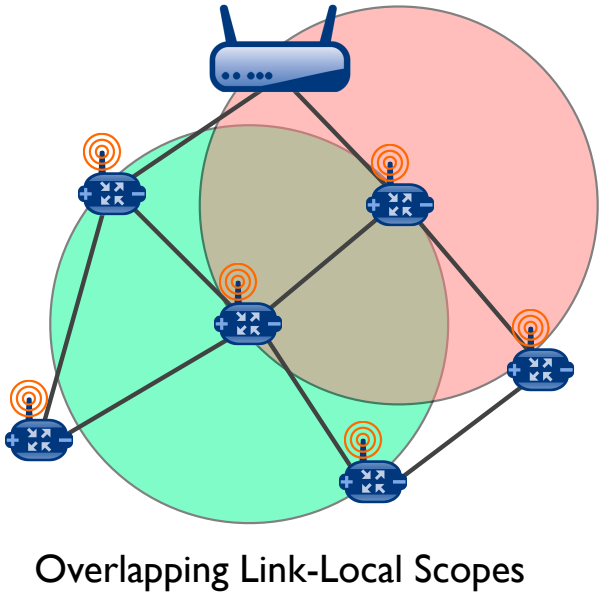


Figure 3.6: **Radio Range ⇔ Link-Local Scope.** Without LAN emulation, a sensornet network is composed of overlapping IPv6 link-local scopes, each defined by the radio transmission range of a node. With overlapping link-local scopes, the link does provide a single broadcast domain. However, the link remains simple and does not enforce any policies, exposing only the most basic capabilities to the network layer above.

effectively. Our IP link model equates link-local scope with the radio communication domain, resulting in a multihop wireless network connected by overlapping link-local scopes, as shown in Figure 3.6. Note, however, that this model does not support a single broadcast domain with transitive reachability. Core IP protocols such as Neighbor Discovery explicitly assume a single broadcast domain. In Chapter 6, we present necessary modifications to support a network composed of overlapping link-local scopes.

Interestingly, our link model does not constrain what layer implements forwarding within a sensornet. The forwarding may occur at the link layer as in MPLS, or at the network layer with traditional IP forwarding. Link-layer forwarding provides the ability to implement different service models and perform traffic engineering. However, link-layer forwarding requires additional addressing information in the packet to identify the end-points of an IP hop. In contrast, network-layer forwarding operates directly on the IP addressing information. IP traffic class and flow labels may also be used to implement different service models and perform traffic engineering. Both link- and network-layer forwarding have their merits, and our IP link model does not dictate which to use.

To provide high best-effort datagram delivery, forwarding mechanisms within the sensornet require additional capabilities from the link. Specifically, link acknowledgments should be exposed, allowing forwarders to implement hop-by-hop retransmissions. If forwarding occurs at the network layer, the link layer should not implement its own retransmission policy, and leave it up to the network layer. Allowing the network layer to specify the retransmission policy allows the forwarding to select alternate next-hop destinations when transmissions fail. So while the link layer may not explicitly provide high best-effort packet delivery, it provides the mechanisms for the network layer to do so.

3.3 IPv6 Addressing & Prefix Model

IPv6 interfaces are configured with one or more IPv6 addresses and one or more IPv6 prefixes. Because our IP link model does not provide transitive reachability, no prefixes may be used to determine whether or not the destination is on-link, except for the link-local prefix. IP places few restrictions on how IPv6 addresses and prefixes are assigned to interfaces within a network. However, we often place additional restrictions to assist with routing and address manageability. The most common example is assigning prefixes that are topologically consistent to allow prefix aggregation, reduce routing state, and support hierarchical routing. In a similar manner, careful IPv6 address assignment can allow significant optimizations within a sensornet.

We constrain the IPv6 addressing model in three ways.

1. We define a new IPv6 addressing scope called the *sensornet scope*. We require that all IPv6 address, including link-local addresses, assigned to nodes in the sensornet are unique with the sensornet scope. Doing so removes the need for nodes to continuously evaluate uniqueness of their address among neighboring nodes. In Chapter 6, we discuss autoconfiguration methods for assigning IPv6 addresses and ensuring their uniqueness within the sensornet.

2. Our IPv6 architecture requires a direct mapping between the IID and link addresses. Doing so allows nodes to resolve network and link layer addresses without any communication or address resolution caches.

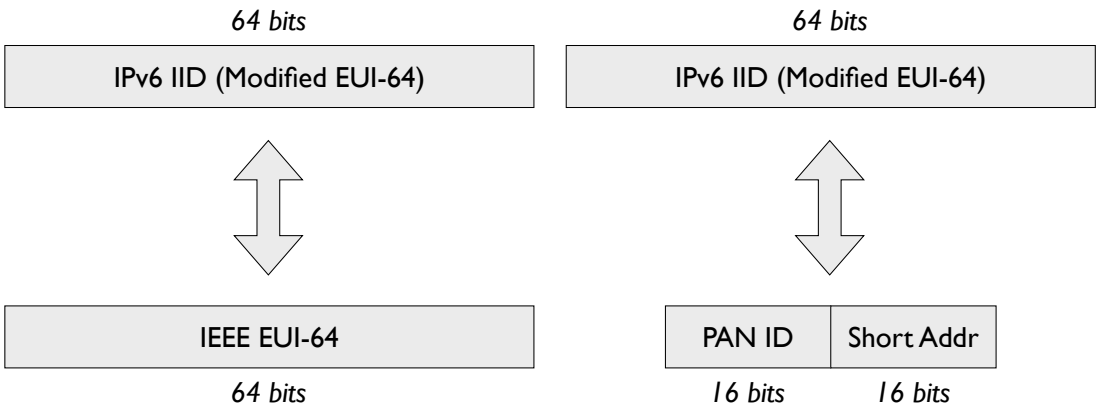
3. The architecture also assumes that IPv6 addresses are configured using global prefixes common to the sensornet, supporting compaction mechanisms to significantly reduce header overhead and memory requirements for forwarding and routing.

We discuss further the benefits of the latter two constraints in the following sections.

3.3.1 Interface Identifier

When assigning unicast IPv6 addresses to an IEEE 802.15.4 interface, our architecture requires that the Interface Identifier (IID) provides a direct mapping to an 802.15.4 link address, as shown in Figure 3.7. The IID may be derived from either the extended 64-bit IEEE EUI-64 or the 16-bit short link address. The former is assumed to be globally unique, while the latter must only be unique within a sensornet. The IPv6 architecture requires the IID to take on a Modified EUI-64 format, which differs from an IEEE EUI-64 by inverting the meaning of the universal/local bit [74]. As a result, deriving an IID from the IEEE EUI-64 link address only requires inverting the universal/local bit. Deriving an IID from a 16-bit short address involves setting the bottom 16 bits to the 16-bit short address, the upper 16-bits to the PAN ID or any other identifier that uniquely identifies the PAN, and ensuring that the universal/local bit indicates local scope. The derivation from EUI-64 or short address is distinguished by the universal/local bit. Furthermore, the IID derivation from short address naturally provides privacy extensions [120].

Deriving IIDs from link addresses provides significant advantages especially in resource constrained environments. First, because the IIDs and link addresses map one-to-one, there is already an implicit mapping between link and network layer addresses. Address resolution protocols are not required, reducing communication latency and energy consumption, as well as memory utilization due to associated caches. Second, the binding implies that if one address is unique within the sensornet, the other is as well. As a result,



(a) IEEE EUI-64 (b) 16-bit Short Address

Figure 3.7: **Direct Mapping Between Interface Identifiers and Link Addresses.** An IPv6 IID derived from an IEEE EUI-64 link address differs only universal/local bit. IPv6 inverted the the meaning of the universal/local bit for deployment convenience [74]. An IPv6 IID derived from a 16-bit short address also includes the PAN ID or any other identifier that uniquely identifies the PAN. The two derivations are distinguished by the universal/local bit.

a single Duplicate Address Detection (DAD) mechanism may be used to ensure uniqueness of both link- and network-layer addresses. Third, the binding allows greater opportunities for packet header compaction and reduced forwarding and routing state, which we take advantage of in Chapter 5.

3.3.2 Global Routing Prefix

In our sensornet IPv6 architecture, global routing prefixes are assigned to an entire sensornet scope, rather than allowing IPv6 addresses to carry arbitrary global routing prefixes. Using common global routing prefixes forms a flat namespace for the entire sensornet. The flat namespace ensures that there is no coupling between the underlying connectivity and the addressing architecture. Nodes are free to move within the network or change the routing topology without worrying about IPv6 address changes. The flat namespace also provides greater robustness to changes in wireless connectivity, due to environmental effects on wireless communication. The flat namespace also allows for standard solutions (e.g., Neighbor Discovery Proxy [167]) to address node mobility across different border routers.

Having prefixes common to all nodes in the sensornet also provides shared context that sensornet nodes can optimize for in significant ways. Common prefixes allows nodes to elide the global prefix in the packet header. Within the sensornet, source and destination addresses never need to carry the common prefix, because neighboring nodes are assumed to agree on what the common prefix is. The result is an IPv6 address containing a common prefix is compressible to 8 bytes when the IID is derived from an IEEE EUI-64 address and 2 bytes when the IID is derived from the short link address. Header compaction opportunities also apply to source route information in an IPv6 Routing Header and even ICMPv6 error messages that encapsulate the error-causing datagram.

Common prefixes also significantly reduce memory requirements for forwarding tables and routing. Because all nodes within a sensornet are assigned common prefixes, forwarding and routing protocols must only be concerned with operating over the IIDs rather than complete IPv6 addresses. Interestingly, operating with IIDs is tantamount to operating directly with link addresses.

3.4 Summary

In this chapter, we briefly described the physical network components of edge and routing devices, as well as border routers, in a typical IP-connected sensornet. We presented an overview of our IPv6-based network architecture, which preserves the protocol layering and functional decomposition of the Internet architecture. We then discussed the foundations of our IPv6-based network architecture, starting with the our IP link model that equates an IP link with radio transmission range, outlining reasons why LAN emulation is poorly suited to the constraints and challenges of sensornets. We also describe the IPv6 addressing and prefix architecture, taking advantage of the large IPv6 address space to reduce communication and memory requirements with minor constraints on assigning IPv6 addresses.

The following chapters present the first complete IPv6 implementation for sensornets. For the first time, we bring together IPv6, IEEE 802.15.4, and results from the sensornet space, showing that all three areas are, in fact, highly complimentary. This dissertation shows that it is possible preserve the layering

while fitting within the strict resource constraints of sensornet, setting the baseline for future development and research. In fact, we show that our IPv6-based network architecture outperforms existing systems that do not adhere to any architecture.

Chapter 4

Link Layer - Supporting the Always-On Illusion on Duty-Cycled Links

In this chapter, we present the design of Emnet, a duty-cycled link-layer for multihop sensornets. The link-layer builds on a large body of existing work, pulling in sampled-listening, scheduling, and streaming techniques to efficiently provide an always-on illusion even while the link is operating at duty-cycles less than 1%. Unlike prior link protocols for sensornets, Emnet only provides the most basic capabilities through a simple link abstraction and leaves all of the policy up to upper layers. In the following chapters, we show how the layers above make effective use of the basic link mechanisms to achieve efficient operation for the entire system.

4.1 Minimizing Idle-Listening While Appearing Always-On

While IEEE 802.15.4 specifies a low-power wireless link standard, the industry has not yet come to agreement on a link protocol for duty-cycled operation in a multihop network. To address this, we develop a duty-cycled link protocol that provides the illusion that it is always-on. As discussed in Chapter 3, an

always-on link provides a connectionless communication service that is friendly to IP and enables more robust networks.

In active mode, the radio typically consumes significantly more than any other component on a sensornet node. A simple power model for radio communication takes into account the cost of listening, transmitting, receiving, and overhearing.

$$P_{total} = P_{listen} + P_{tx} + P_{rx} + P_{overhear}$$

Nearly a decade ago, Xu, Heidemann, and Estrin. observed that *idle-listening* completely dominates system energy consumption when the radio is not duty-cycled [188]. Low-power wireless radios consume as much power when receiving, or even just listening, when compared to transmitting [165].

To reduce the idle-listening cost, the radio must be duty-cycled and hence a transmitter can only send packets to a receiver at specific times. The coordination of receiver-transmitter schedules, we term *Media Management Control (MMC)*, and is orthogonal to *Medium Access Control (MAC)*, which defines how to *arbitrate* access to the media between simultaneous transmitters. When links operate at less than 1% duty-cycle, channel contention is rare.

Three basic mechanisms have emerged to address the idle-listening problem: *sampled listening* [13, 49, 73, 92, 137], *scheduling* [50, 66, 77, 175, 189], and *listen-after-send* [127, 128]. Sampled listening supports the always-on abstraction, trading reduced listening cost for increased transmission cost. This is the proper tradeoff, as transmissions are typically rare. Sampled listening monitors the channel periodically using short receive checks (often implemented by measuring the RSSI) to determine if a frame is being transmitted by a neighboring node. A node transmits frames by lengthening its transmissions to be as long as the receiver's sample period. In contrast, scheduling involves synchronizing time and schedules across nodes so that nodes know a priori when the receiver's media is enabled. Scheduling removes the need to lengthen transmissions, but comes at the cost of needing to establish and maintain state for each neighbor. Finally, listen-after-send allows ultra-low duty cycles, but at the cost of increased latency.

Existing link layer proposals implement too much policy in the link layer and make most solutions unnecessarily application specific. For example, most employ only one of the three techniques. While some proposals have combined sampled listening and scheduling, they still remain specific to certain communication scenarios [48, 191]. SCP-MAC, for example, requires all nodes to synchronize channel samples, which support efficient broadcast communication but effectively reduces the channel capacity and increases contention. Basic link layer mechanisms proposed within the sensornet domain are not specific to any application, rather the specific use of those mechanisms are.

Our goal in designing a duty-cycled link is to consume minimal power while providing the following IP-friendly properties:

- **Always-On**: Nodes should be able to communicate without establishing a connection or requiring any existing state.
- **Low Latency**: Transmission delays to any neighboring node should be low.
- **Broadcast Capable**: Nodes should be able to broadcast frames to all neighboring nodes, regardless of node density.
- **Synchronous Acks**: The link should allow IP to achieve high "best-effort" datagram delivery.

In this chapter, we develop Emnet, a duty-cycled link layer for sensornets that provides IP-friendly capabilities. We develop Emnet on IEEE 802.15.4 radios, using the standard physical layer and frame formats defined by the specification. Emnet uses *sampled listening* techniques as the baseline communication mechanism, to support an always-on property and low latency communication. Emnet also uses a number of mechanisms to allow application-specific optimizations. For example, Emnet uses *scheduling* techniques minimize energy costs when frequently communicating to specific neighbors. Emnet employs *packet streaming* to dynamically increase throughput and reduce per-packet energy costs. Finally, Emnet also defines its own acknowledgment frames. Emnet acknowledgments allow piggybacking of network layer information to support hop-by-hop feedback. In the process, we also repair the security flaws and false-positive acknowledgment problems of IEEE 802.15.4 [152].

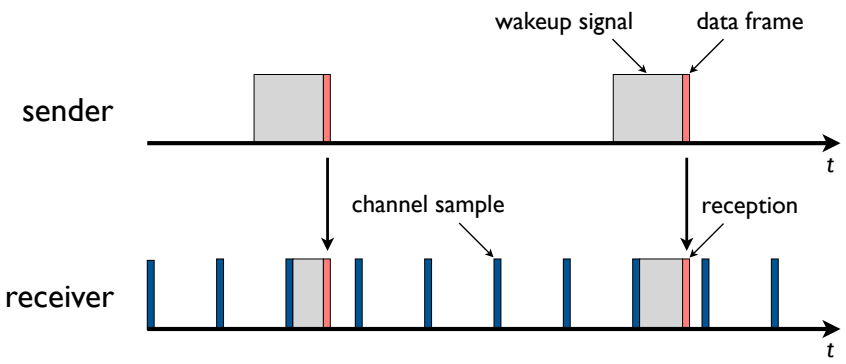


Figure 4.1: **Sampled Listening.** Nodes use short channel samples to periodically listen for transmissions from neighbors. A node can transmits packets by lengthening transmissions to meet or exceed the receiver's channel sample period. Because no connection state must be established or maintained, sampled listening supports the always-on property.

4.2 Background

Three basic mechanisms have been proposed to reduce idle-listening and provide effective media management: *sampled listening*, *scheduling*, and *listen-after-send*. In this section, we provide a brief background of each.

4.2.1 Sampled Listening

Sampled listening samples the channel periodically using short receive checks to determine if a frame is being transmitted by a neighboring nodes. Channel samples are often implemented by turning on the receiver just long enough to measure radio frequency energy on the channel, provided by many radios through a received signal strength indicator (RSSI). Typically, measuring RSSI requires much less time than it takes to decode a radio signal and much less than receiving a frame, minimizing the amount of time the receiver is enabled. Lower energy cost per channel sample allows the link to listen more frequently.

A short channel sample period provides a number of useful properties. First, a node can transmit packets by lengthening the transmissions to match or exceed the receiver's channel sample period. The node need not establish a connection with the neighboring node to synchronize time, establish schedules, or synchronize any state. As a result, sampled listening directly supports the always-on property. Second, more

frequent channel samples directly translate into reduced communication latency, since the earliest a node can receive a packet is at the next listening period. An always-on network with low latency communication is needed for mobile applications and effective network management. Third, more frequent channel samples also allow increased channel capacity, as nodes are given a communication opportunity with each channel sample.

Sampled listening with lengthened transmissions pushes the energy cost from the receiver to the transmitter. This energy tradeoff is the right one for most sensornet applications workloads, as transmissions are relatively infrequent and listening must happen frequently to support a responsive network. By not requiring any state or time synchronization, the link does not require any communication between neighboring nodes simply to maintain network connectivity. However, the increased transmission cost becomes less favorable in application workloads that require more frequent transmissions.

4.2.2 Scheduling

Scheduling involves time synchronization between neighboring nodes and establishing communication schedules that indicate when nodes should listen or transmit. Simple scheduling involves network-wide time synchronization, with all nodes listening at the same time [170, 191]. Network-wide listening does not require any per-neighbor state, but reduces the effective channel capacity and increases contention because all network communication is constrained to small time windows. Alternatively, a different schedule can be assigned to each pair of neighboring nodes. Per-neighbor schedules allow more effective use of the channel because the network's transmissions are spread over time, but requires per-neighbor state that must be established and maintained.

The challenge with scheduling is how nodes establish a connection with the network or neighboring nodes. Without knowing the schedule, a joining node does not know when neighboring nodes will listen to the channel. One naive approach has all connected nodes periodically beacon scheduling information and joining nodes leave their receiver enabled until receiving a beacon. In general, the join process can have significant energy costs and latency, limiting the cost effectiveness of scheduling especially in dynamic environments.

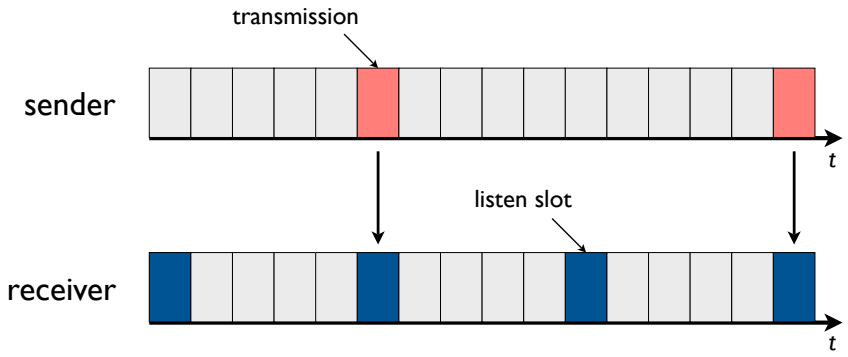


Figure 4.2: **Scheduling.** Nodes are time synchronized and communication schedules are configured. Synchronization does not require increased transmission costs, but does require a baseline cost for establishing and maintaining time synchronization and communication schedules.

Mobility or environmental effects can change the physical connectivity, requiring nodes to frequently re-join or establish new connections.

Compared to basic sampled listening, scheduling does not require added transmission costs, but does require nodes to establish connections before they can communicate with neighboring nodes. As a result, basic scheduling does not support the always-on abstraction. The lengthened transmissions in sampled listening essentially perform per-packet receiver synchronization while scheduling relies on a background process to constantly maintain receiver synchronization. The former does not require the node to maintain any per-neighbor synchronization state, better supporting more dynamic topologies. The latter, however, dictates a baseline for energy consumption simply for maintaining synchronization.

4.2.3 Listen-After-Send

Listen-after-send involves nodes that periodically probe forwarding nodes for pending messages [127, 128]. Forwarding nodes that receive messages destined for a neighbor that is operating with listen-after-send must buffer messages until the next time the neighbor probes for pending messages. Periodic probes involve a normal transmission to the forwarding node, using synchronous acknowledgments to indicate whether or not packets are pending. If so, the probing node continues listening to the channel to receive packets from the forwarder, which immediately being transmitting buffered packets after the acknowledgment.

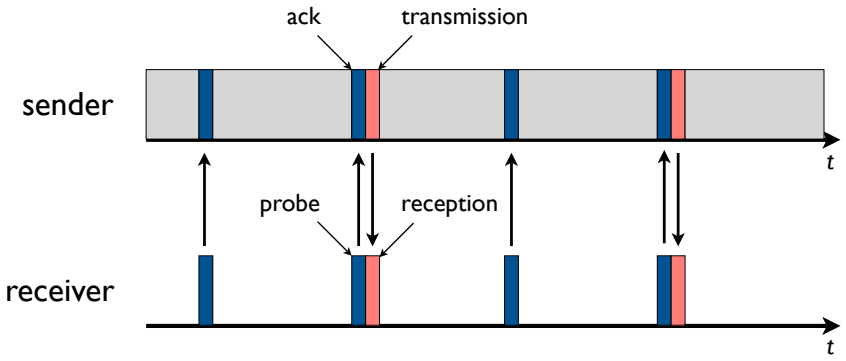


Figure 4.3: **Listen-After-Send.** Nodes periodically probe forwarding nodes for any pending messages. Forwarding nodes buffer messages until the destination node sends a probe message. Listen-after-send allows nodes to operate at low duty cycles, without requiring any synchronization costs or increased transmission costs at the forwarder.

Listen-after-send allows nodes to operate at low duty cycles, does not incur any transmission overhead at the forwarding node, and does not need to establish or maintain synchronization. However, these benefits come at the expense of communication latency and buffering at the forwarding nodes.

4.3 Media Management Control

Our MMC builds on B-MAC [137] and WiseMAC [48]. Sampled-listening provides the baseline communication mechanism, providing a robust always-on abstraction and broadcast communication without any existing state. We employ scheduling techniques to reduce transmission costs, channel utilization, and overhearing. However, we improve on WiseMAC by embedding addressing and timing information into the wakeup signal, allowing us to significantly reduce the cost of channel samples, receiving frames, and overhearing messages. We also add streaming capabilities to increase throughput and lower average transmission costs. Finally, we expose the mechanisms through a simple, but expressive link abstraction derived from SP [138]. Note that while we specifically address 802.15.4 radios, the protocol is not fundamentally constrained to 802.15.4.

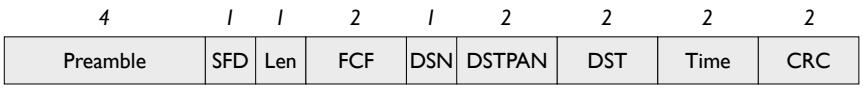


Figure 4.4: **Chirp Frame.** Emnet implements the wakeup signal using short *chirp frames* sent back-to-back. Chirp frames include *addressing information* to minimize overhearing costs and a *rendezvous time* to indicate when the sender will transmit the data frame and reduces energy cost at the receiver. Short chirp frames sent back-to-back are fundamental in minimizing the energy cost of channel samples.

4.3.1 Sampled Listening

Sampled listening requires two primitives: (i) a wakeup signal when transmitting and (ii) channel sampling. Initial work implemented the wakeup signal by extending the preamble, repeatedly transmitting a short bit-pattern that receivers can recognize when listening. Preamble extension was possible to do on primitive, low-level radios that provided a bit- or byte-based interface. However, radio advancements over the past ten years have resulted in increased throughput, automatic CRC generation, inline AES-128 authentication and encryption, among others. These advancements were necessarily accompanied by the move to packet-based interfaces that generally do not allow arbitrary control over the preamble length. Furthermore, IEEE 802.15.4 defines a fixed length and bit-pattern for the preamble and extending it would break compliance with the standard.

To generate a wakeup signal on packet-based, IEEE 802.15.4 radios, Emnet implements the chirp signal using short *chirp frames* [13]. Emnet allocates an IEEE 802.15.4 reserved frame type to identify the chirp frame from other frames, such as beacon, MAC command, and data. As shown in Figure 4.4, the chirp frame is an IEEE 802.15.4-compliant frame and contains a *destination address* and a *rendezvous time* that indicates the time remaining until the actual data frame is transmitted. Addressing information significantly reduces overhearing costs, as unintended destinations can abort reception early. The rendezvous time allows the intended destination to power down until the data frame begins transmission. This not only reduces receive cost, but makes the cost independent of the chirp signal length. Sending short chirp frames back-to-back as close as possible is fundamental in reducing the cost of channel samples. Sending short chirp frames back-to-back reduces the period between chirp transmissions. Minimizing channel sample cost allows listening to occur more frequently to support a lower communication latency at the same duty-cycle.

Channel samples follow a *staged-wakeup* approach, implementing a form of *passive vigilance* [44, 51]. Like most existing approaches, we use RSSI provided by the radio. But only relying on RSSI can result in many false positives, especially when there is significant environmental noise. False positives increase energy cost as the link layer keeps the receiver on in attempt to receive a data frame. So rather than going directly into receive mode and waiting for the data frame, we insert additional stages that continue to process the signal and decides whether or not to move to the next stage or abort. The staged-wakeup process is as follows: The link enables the receiver and reads the RSSI. If the RSSI is above threshold, the link searches for a Start of Frame Delimiter (SFD) that indicates the start of a packet reception. If an SFD is detected, the link searches for addressing information to determine if the frame is destined for it. On an address matches, the link attempts to receive the remainder of the chirp frame. After receiving a chirp frame, the link reads the rendezvous time and disables the receiver until that time. Each search in this staged-wakeup has an associated timeout. If any of the timeouts expire during the process, the link immediately ends the staged-wakeup process and disables the receiver.

The staged-wakeup allows Emnet to operate in new domains not possible with existing sampled listening implementations. By including addressing information, overhearing costs are reduced to the cost of receiving a chirp frame. By including a rendezvous time, the cost of receiving is reduced to the cost of receiving a single chirp frame and the data frame itself. An edge device communicating with a powered router can take full advantage of the new capabilities, by shifting the entire energy burden to the powered forwarding device. Existing channel sampling implementations require the receiver to remain on until the data frame arrives, making both transmission and reception costs proportional to the channel sample period.

4.3.2 Synchronous Acknowledgments

To allow the network-layer to achieve high “best-effort” datagram delivery, synchronous acknowledgments are required to support hop-by-hop retransmissions. IEEE 802.15.4 specifies a simple ack frame, including no addressing information and only echos the sequence number of the received data frame. Match-

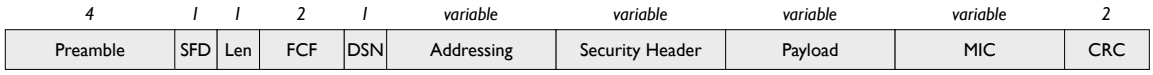


Figure 4.5: **Revised Ack Frame.** Emnet re-defines the IEEE 802.15.4 ack frame by allowing addressing and security headers, as well as a payload. The revised ack frame solves the current problems with IEEE 802.15.4, including false positive acks and inability to authenticate and encrypt ack frames. The payload allows piggybacking of link or network layer information to support hop-by-hop feedback.

ing an ack frame to a data frame relies on both timing and the sequence number. Unfortunately, the ack frame defined in IEEE 802.15.4 is insufficient for use in production networks for the following reasons:

1. **No Addressing Information**: Lack of addressing information results in acknowledgments that are always sent to the broadcast address. Doing so leads to false positives, which reduce the reliability of the link. The hidden terminal problem in multihop networks or high channel contention increase the likelihood of false positives. For example, if two different nodes simultaneously unicast messages with the same data sequence number, one may receive an acknowledgment that was intended for the other transmission.

2. **No Security Mechanisms**: Unlike IEEE 802.15.4 data frames, ack frames do not support AES-128 authentication or encryption [152]. An attacker can easily inject acks to perform a denial-of-service attack or affect any protocol that relies on those synchronous acknowledgments, including hop-by-hop flow control for forwarding or link estimation protocols for routing.

3. **No Payloads**: An IEEE 802.15.4 ack frame cannot carry a payload. Allowing a payload in ack frames enables the use of hop-by-hop mechanisms that have become some pervasive in the sensornet area. As we discuss in the following section, we piggyback scheduling information on ack frames to optimize link transmissions by neighboring nodes. We also piggyback network layer information, to implement network layer hop-by-hop reliability and flow control.

To address the limitations of IEEE 802.15.4, we define a new ack frame. For the most part, the ack frame is indistinguishable from a data frame, except that the IEEE 802.15.4 frame type indicates an ack frame. The new ack frame can carry both source and destination addresses along with a sequence number to properly

match an ack frame with data frames, security headers for authentication and encryption, and a payload for piggybacking additional information. The ack frames share the same addressing and security mechanisms that data frames do and does not require any added implementation overhead. Addressing information not only removes the ambiguity, but also relaxes the timing requirements for sending ack frames.

4.3.3 Scheduling

Basic sampled listening trades listening energy for increased transmission costs. However, if the receiver's listening schedule is known, the transmitter does not need to send a long wakeup signal [48, 191]. Instead, the transmitter could wait until the appropriate time, send a short wakeup signal and then the data message. Knowing the receiver's schedule can significantly reduce the transmission energy cost and make the energy cost independent of the receiver's channel sample period. However, the tradeoff is that the transmitter must maintain state about the receiver's schedule.

Emnet uses node-local synchronization to learn an individual receiver's listening schedule, by piggybacking timing information in sent frames. Ack frames are extended to include information about the sender's listening period and phase. The sample phase is specified as the time between a global time point and the next channel sample. If global time synchronization is not available, the sample phase may be specified from the ack's transmission time. Piggybacking scheduling information in ack messages can allow the transmitter to learn the receiver's schedule after a single acked transmission. When destination's schedule is known, shortened wakeup signals are used when transmitting. The wakeup signal length accounts for the synchronization guard time, and the synchronization error determines the length. More frequent communication reduces the average wakeup signal length, as synchronization information exchanges more frequently. The synchronization error is bounded by the receiver's channel sample period.

Unlike existing scheduled approaches, Emnet only uses scheduling information as a *hint* to optimize performance rather than relying on it for correctness. In other words, the hint is used to optimize transmissions and does not depend on the information to communicate with a neighboring node. If transmissions with short wakeup signals are not acknowledged, Emnet increases the wakeup signal duration. In the

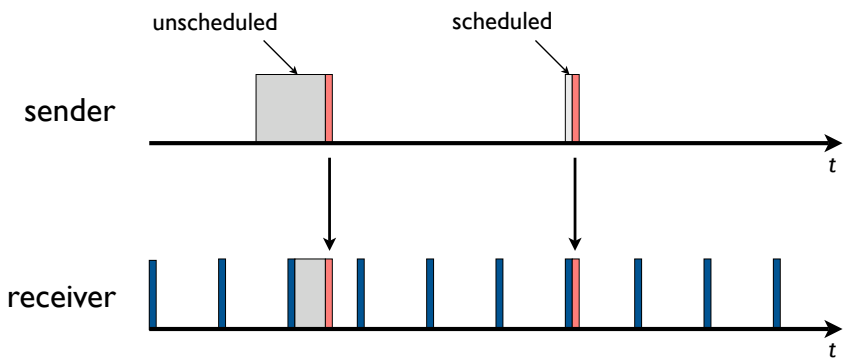


Figure 4.6: **Scheduling Optimization.** Emnet uses scheduling to optimize transmissions. Channel sample schedule information is piggybacked on ack transmissions. Emnet only uses scheduling information as a hint and does not require it to communicate with neighboring nodes.

worst case, Emnet can fall back to the receiver's channel sample period. Piggybacking scheduling information on ack frames binds the synchronization mechanism with traffic generated by upper layers and makes the scheduling mechanism adaptive upper-layer traffic. Existing approaches often trade continuous synchronization costs with synchronization error and make decisions that are independent of the traffic characteristics.

Node-local scheduling allows nodes to determine their own schedules. In general, scheduling reduces average channel utilization because wakeup signals are reduced and contention decreases because schedules are usually non-overlapping. Localized scheduling allows greater flexibility and robustness because node only need to establish and maintain synchronization state with their neighbors. Two nodes can operate in isolation without a global manager. However, Emnet does not preclude the use of global coordination. A central manager can still be used to push schedule information into nodes. As with much of our architecture, Emnet only provides the basic mechanisms, allowing the specific use of those mechanisms to be application-specific.

4.3.4 Streaming

Emnet allows packet streaming to increase throughput and efficiency when communicating multiple frames back-to-back to the same destination [137, 138]. Emnet uses the Frame Pending bit in the IEEE 802.15.4 header to indicate to the receiver that another data frame will be following immediately. By setting

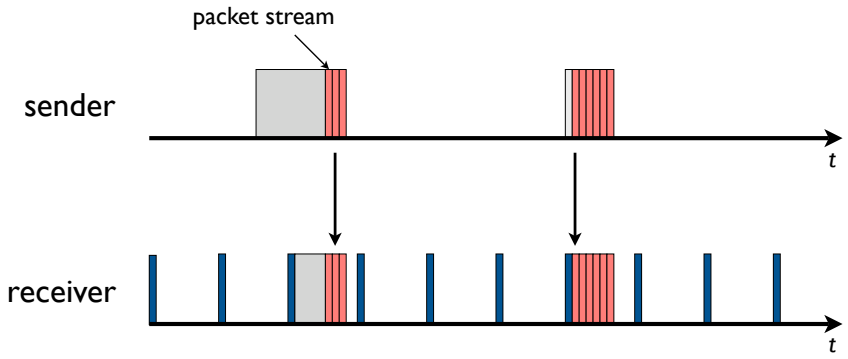


Figure 4.7: **Streaming Optimization.** Emnet uses streaming to quickly send data frames back-to-back, increasing throughput and efficiency.

the Frame Pending bit, a node can send data frames one-by-one without delay. Each transmitted frame, however, still generates a synchronous acknowledgment to ensure that the receiver is still receiving frames and is used to communicate hop-by-hop flow control information that we present in Chapter 7.

4.4 Media Access Control

By default, the link protocol uses CSMA/CA for its MAC because there is no coordination between the transmission schedules of nodes. When no scheduling information is known, the MAC simply delays the start of the wakeup signal by a random amount, performs a channel assessment to check for activity, and begins the wakeup signal transmission if the channel is clear. If the channel is busy, the transmission is delayed by another random amount.

When scheduling information is known, Emnet increases the wakeup signal duration by a random amount. The wakeup signal cannot be shortened because it represents the synchronization error and the length must be randomized or collisions are likely to occur when other nodes are synchronized with the same receiver. After determining the randomized wakeup signal duration, the MAC waits until the appropriate time, performs a channel assessment, and begins transmitting the wakeup signal if the channel is clear.

CSMA/CA is not used for all transmissions. Acknowledgment frames are synchronous, and Emnet sends them immediately after receiving a data frame without any added delay or channel assessment. All

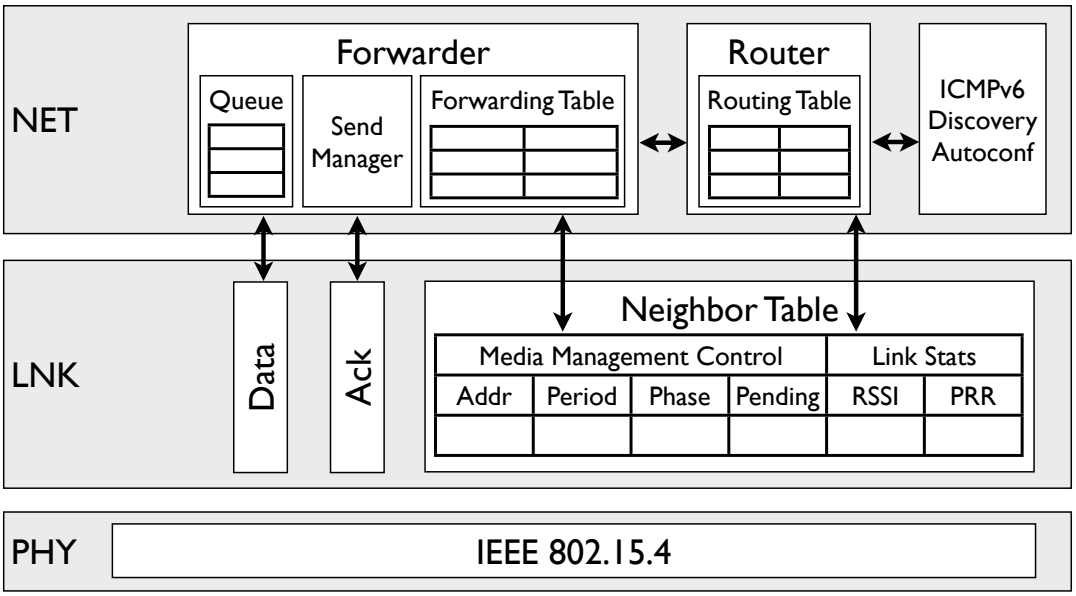


Figure 4.8: **Link Abstraction.** Emnet shares information with upper layers through a simple abstraction. The information sharing allows Emnet and upper layers to optimize their operation based on the information. A neighbor table allows the network layer to indicate which neighbors to optimize for and Emnet uses the table to store synchronization information and link estimation state that the network layer can use for routing decisions. A frame pending indicator allows the network layer to utilize the streaming feature.

frames in a stream, except the first frame, are sent immediately after receiving the ack frame for the previous data frame. While CSMA/CA is the default mechanism, Emnet does not preclude the use of TDMA when transmission schedules are coordinated. When transmission schedules are non-overlapping, the wakeup signals no longer require random increases and can remain equal to the synchronization error.

4.5 Link Software Abstraction

From an IP perspective, none of the link mechanisms described here are specific to IP, although their selection was guided by IP. The purpose of the software abstraction is to expose necessary mechanisms that allow the network layer above to perform efficient forwarding and allow the link to take guidance from the network layer to optimize performance. Note that while SP proposes to lower the narrow-waist to the link layer by providing an effective abstraction, we propose that the narrow-waist remains at the network layer.

An effective link software abstraction makes it easy to take advantage of all of the link's functionality while preserving the layering as much as possible. We build on our work with SP [138], but simplify the abstraction because we assume a single network layer above. Our link abstraction exposes a set of generic attributes that the link and network layers use to share information between them, as shown in Figure 4.8. The abstraction exposes a *neighbor table* used to store any information about neighboring nodes. Each entry in the table stores the following attributes:

- **Link Addresses**: Storing both short 16-bit and extended IEEE EUI-64 addresses. Both addresses are used to allow the link and network layers to lookup information for a specific neighbor.
- **Scheduling Information**: Stores the channel sample phase and period for the neighboring node. The link layer establishes and maintains scheduling information for any node in the neighbor table to optimize transmissions to those neighbors. Alternatively, the network layer can override these attributes to establish more custom schedules.
- **Link Quality Information**: The link layer maintains link quality estimates for any node in the neighbor table. The link maintains the packet error rate for the most recent transmissions. The link also maintains a windowed average of the received signal strength indicator (RSSI) for received messages. The network layer may use the link quality information to assist in computing routing costs and selecting routes. We discuss the use of the link quality information for IP routing in Chapter 8.
- **Frame Pending Indicator**: The network layer uses the frame pending indicator to notify the link layer that frames are pending for that neighbor. The frame pending indicator enables the use of Listen-After-Send techniques, where the link sets the Frame Pending bit in the IEEE 802.15.4 ack header to notify the probing node that frames are pending for it.

In general, memory constraints will limit the number of entries in the neighbor table. The MMC uses a least-recently used (LRU) policy for evicting neighbors to insert new neighbors into the table. However, the link allows the network layer to pin specific entries in the table, preventing the link from evicting them. The network layer uses the neighbor table to notify the link layer which neighbors are important to optimize

transmissions and maintain link quality estimates for. Allowing the network layer to choose the neighbor set is important because routes are ultimately chosen at the network layer based on estimated route costs.

The link abstraction also augments the data path by including attributes with each frame transmission and reception. When transmitting a frame, the network layer can set the *frame pending* attribute to indicate that additional transmissions to the same frame are pending and allows the link layer to take advantage of streaming optimizations. The result of each transmission indicates whether or not the transmission was acked and, if so, returns the RSSI of the received acknowledgment. Each received frame is also tagged with the RSSI. Augmenting the data path with RSSI information provides upper layers with a link quality estimate without requiring an entry in the neighbor table.

The link abstraction also exposes the local channel sample phase and period attributes to upper layers. Exposing attributes for the MMC of each node allows upper layers to adjust the channel sample period to trade communication latency for listening cost or coordinate schedules between nodes to ensure that they don't overlap. For example, the link abstraction allows a TDMA-based network architecture that centrally assigns listening and communication schedules.

4.6 Related Work

The link layer has received much attention from the sensornet community, especially since the idle-listening problem was widely recognized as the most significant factor in energy consumption [188]. Most existing work has focused on developing sampled listening and scheduling. More recent efforts have begun to acknowledge that sampled listening and scheduling are, in fact, orthogonal mechanisms that can be used together. Other work has focused on exposing the mechanisms through a minimal, yet flexible abstraction. Emnet builds heavily on existing work, bringing them together into a single architecture.

4.6.1 Sampled Listening

Initial work with sampled listening began in 1987, with the DARPA packet radio [92] and was rediscovered in the sensornet space in 2002 [49, 73]. Sampled listening supports the always-on abstraction by allowing nodes to communicate without first establishing time synchronization or scheduling state. These initial protocols periodically woke up to check the channel for activity, using received signal strength or searching for a preamble. The wakeup signal was a lengthened preamble. But the lack of addressing or timing information made overhearing and receiving just as costly as transmitting. Nodes had to keep receiving until the data packet was transmitted. As a result, the effectiveness of initial sampled listening protocols were highly sensitive to the channel sample period and traffic characteristics of the application.

B-MAC recognized the sensitivity to link settings and proposed that the link should focus on providing the mechanism to upper layers and leaving the policy up to those upper layers [137]. B-MAC exposed control of the channel sample period, preamble length, and CSMA/CA backoff. B-MAC showed that allowing upper layers to specify link settings can result in greater performance.

The introduction of packet-based radios spurred work in finding alternatives to generate a wakeup signal. X-MAC proposed sending short wakeup packets [13]. Overhearing costs were minimized by including a destination address in the wakeup frame. Receivers could cut the wakeup signal short by acknowledging receipt of a wakeup frame. However, doing so requires inserting gaps between wakeup frames to allow the receiver to acknowledge the wakeup frame. By inserting gaps, the channel sample duration and associated listening cost increases significantly. Alternatively, increased listening costs significantly increases the communication latency at the same listening duty cycle. For comparison, X-MAC reported a channel-sample duration of 15 ms, while Emnet only requires 640 us.

4.6.2 Scheduling

Link protocols that rely only on scheduling do not support the always-on abstraction, as nodes must first establish a connection with neighboring nodes. Time synchronization and scheduling techniques may either be network-wide or pair-wise. S-MAC and T-MAC took a network-wide approach for both time

synchronization and scheduling, with all nodes waking up simultaneously [175, 189]. During each listen period, nodes exchange synchronization information and any pending data frames. S-MAC specified a fixed period of 115 ms for the listen period, while the sleep period determined the duty cycle. T-MAC improved on S-MAC by shortening the listen period when no activity is detected.

Most other work focused on enabling pair-wise scheduling. PEDAMACS supported pair-wise scheduling, but over a single-hop domain [50]. PEDAMACS relied on a powered, central coordinator to broadcast timing and scheduling information to all nodes in the network. The central controller had to be within direct radio range of all the network nodes. PEDAMACS was an effective, simple mechanism for infrastructure-based deployments. FPS supported multihop networks and used network-wide time synchronization, but scheduling was done in a distributed manner by the network layer [77]. FPS used a tree routing structure to synchronize nodes, assign communication slots, and route packets. FPS demonstrated that network layer knowledge can significantly increase the effectiveness of the link protocol.

WirelessHART supports pair-wise scheduling over a multihop network but through the use of a central coordinator [66]. Nodes joining the network must discover neighbors, synchronize with the network, send a message to the coordinator, and wait for a communication schedule from the coordinator. According to the datasheet, the join period can take up to 1 hour and consume up to 20 mA while doing so [43]. On a pair of AA batteries, 100 node join operations will completely exhaust the energy capacity. Once joined, schedules can be made to ensure there is no self-interference by assigning non-overlapping slots through central coordination. However, the use of a central coordinator can significantly increase the cost of establishing connections with neighboring nodes and presents a single point of failure.

4.6.3 Hybrids

Aloha-PS first proposed the use of scheduling to optimize transmissions to neighboring nodes. Aloha-PS suggested the use of long preambles when a node is joining a network, then uses scheduling to eliminate the wakeup signal [49]. Follow on work in WiseMAC completed the story, by placing synchronization information in acknowledgments [48]. Emnet is most closely related to WiseMAC, using sampled-

listening as the baseline communication mechanism and scheduling techniques to reduce transmission costs, channel utilization, and overhearing. However, we improve on WiseMAC by embedding addressing and timing information into the wakeup signal, allowing us to significantly reduce the cost of channel samples, receiving frames, and overhearing messages. We also add streaming capabilities to increase throughput and lower average transmission costs. Finally, we expose the mechanisms through a simple, but expressive link abstraction.

A number of hybrid proposals combined sampled listening or scheduling techniques with other prior protocols. Channel-sampling techniques were brought to improve listening costs of S-MAC and T-MAC [65]. Listen periods were shortened dramatically by using short channel-samples. While energy consumption was reduced, synchronization costs still remained. Uncertainty-driven B-MAC (UBMAC) combined scheduling techniques with B-MAC [60]. UBMAC accounted for synchronization errors by appropriately sizing the wakeup signal length. However, unlike Emnet, UBMAC does not support pair-wise synchronization mechanisms.

SCP-MAC used channel sampling techniques, but in combination with scheduling to reduce the wakeup signal length [191]. SCP-MAC used network-wide synchronization, requiring all nodes to sample the channel simultaneously. Doing so does not require any per-neighbor state and supports efficient broadcast, but reduces effective channel capacity and increases contention. Emnet does not require network-wide synchronization, allowing more effective use of the channel. Note, however, Emnet does not preclude network-wide synchronization.

4.6.4 Link Abstractions

In our previous work, we proposed SP as a unifying link abstraction for sensornets through a shim between the link and network layers [138]. SP argued that the link should operate as the narrow-waist of the protocol stack, providing a common abstraction for a number of link layers below and supporting a wide range of network layers above. A goal of SP was to allow multiple network protocols above to operate simultaneously, which led to placing transmission scheduling functions within the SP shim to account for

pending transmissions from different network protocols. SP proposed a message pool interface to allow network protocols to inform SP of pending transmissions. SP proposed the use of neighbor tables to support greater information sharing between the link layer, SP, and the network layer. SP also proposed message futures, allowing the network layer to indicate how many packets are pending for a particular destination.

Our IPv6-based network architecture differs in that we now believe that IP should remain as the narrow waist of the protocol stack. The link abstraction need only support a single network protocol above, namely IP. As a result, the link abstraction does not provide a message pool interface. All transmission scheduling occurs within the network layer's forwarder, rather than implementing them in a shim below the network layer. Another difference is that the message futures interface is simplified down to a single Packet Pending bit. Finally, unlike SP, our link abstraction exposes link quality estimates for use by upper layers.

4.7 Summary

In this chapter, we presented Emnet, a duty-cycled link layer for sensornets. Using of sampled listening techniques, Emnet supports the always-on property that is traditionally assumed of IP links. Emnet allows the network-layer to achieve high "best-effort" datagram delivery through the use of synchronous acks, but does not implement any retransmission policy itself. The Emnet ack frames solve existing issues with IEEE 802.15.4, including security and false positive acks. Emnet implements scheduling optimizations to reduce energy consumption, increase channel usage efficiency, and reduce channel contention. Emnet also implements streaming optimizations to increase throughput while reducing overall energy cost. A simple link abstraction allows the link and upper layers to share information for use with optimizing overall performance. From an IP perspective, none of the link mechanisms described here are specific to IP, although their selection was guided by IP.

In later chapters, we show how the network layer makes effective use of the link abstraction to cooperate with the link layer and improve performance. We demonstrate in Chapter 7 how the forwarder utilizes the streaming mechanisms to improve throughput and efficiency. In Chapter 8, we demonstrate

how the router inserts entries into the neighbor table and makes use of link quality estimates from the link. The increased cross-layer cooperation is what allows us to implement an IPv6-based network architecture. Furthermore, the simple link abstraction allows us to do so without violating the layered protocol model.

Chapter 5

Adaptation Layer - Transmission of IPv6 Datagrams over IEEE 802.15.4

In this chapter, we present an adaptation layer for communicating IPv6 datagrams within IEEE 802.15.4 frames. The adaptation layer supports three functions: (i) IPv6 header compression (LOWPAN_HC) to reduce header overhead, (ii) datagram fragmentation to support the IPv6 minimum MTU, and (iii) support for layer-two forwarding. Inspired by IPv6's extension headers, the adaptation format uses a header stacking approach that is simple to parse and allows for compact forms. LOWPAN_HC compresses headers using shared-context techniques at the network layer and stateful techniques at the transport layer, reducing 48-byte UDP/IPv6 headers down to 8 bytes in the best case. Many ideas in this chapter have been adopted by the IETF and published as a proposed standard in RFC 4944 [117].

5.1 Coping with Large IPv6 Datagrams

IPv6 supports a number of mechanisms that make an IP-based architecture more attractive on sensornets, including a large IPv6 address space, autoconfiguration of large numbers of nodes, and a protocol options framework pervasive throughout the IPv6 architecture. However, as specified, IPv6 packet formats

are too large to be feasible for the low-power radios typically used in sensornets. IPv6 even specifies a 1280 byte minimum MTU requirement for the link, reflecting advancements in more traditional link technologies that are expected to be most commonly used.

To support the IPv6 minimum MTU requirement, an *adaptation layer* must be introduced to *fragment* large IPv6 datagrams [62, 90, 143]. The adaptation layer logically resides between the link and network layers and is responsible for fragmenting, delivering, and reassembling IPv6 datagrams that do not fit within a single IEEE 802.15.4 frame. Given that an adaptation layer is required to deliver IPv6 datagrams, we take the opportunity to include other functionality that makes delivery of IPv6 datagrams more cost effective.

The adaptation layer could support *header compression* mechanisms to reduce header overhead and the frequency of fragmenting IPv6 datagrams [183, 184]. Compared to the 127-byte IEEE 802.15.4 MTU, network and transport layers are relatively large. The IPv6 header is 40 bytes, not including any IPv6 extension headers. UDP and TCP, both typical transport layer protocols, have header sizes of 8 and 20 bytes, respectively. In contrast, the maximum available payload for an IEEE 802.15.4 frame may be as small as 81 bytes, when carrying extended addresses and security headers within the link header. As a result, as little as 22 bytes of application payload can make a single IEEE 802.15.4 frame insufficient for transporting an IPv6 datagram. Furthermore, many sensornet applications typically communicate with relatively small payloads, and leaving network and transport layers uncompressed can result in significant header overhead.

The adaptation could also support multihop forwarding of IPv6 datagrams at layer two in a similar manner to Multiprotocol Label Switching (MPLS) [148]. While the initial motivation for MPLS was to simplify forwarding hardware by using simple labels to identify the destination, MPLS also naturally provides mechanisms for supporting multiple service models and perform traffic engineering. Similar functionality may be useful in the sensornet space, where different routes may be used to trade different levels of communication latency with buffering and energy cost. Layer-two forwarding may also be used to support multipath forwarding of fragments for a given IPv6 datagram. By forwarding IPv6 datagrams at layer three, fragmentation and reassembly must logically occur at every hop. However, constraining fragments to a single

path naturally enforces in-order delivery and allows better use of link optimizations such as scheduling and streaming.

In 2005, the Internet Engineering Task Force (IETF) formed the 6LoWPAN working group to define an adaptation layer that enables IPv6 over IEEE 802.15.4 links. Initial work focused on defining a header format for mechanisms defined in the adaptation layer. The initial mechanisms included fragmentation, layer-two forwarding, and IPv6 and UDP header compression. We took the initial developed by the working group and improved upon them, by creating a clean yet concise header format and generalizing header compression to support unicast and multicast communication over link-local and non-link-local scopes. The current proposed standards from 6LoWPAN as well as improvements to the standards are a result of this work [117].

In this chapter, we develop an adaptation layer to transport IPv6 datagrams over IEEE 802.15.4. The adaptation layer supports the following mechanisms:

- **Fragmentation**: To divide IPv6 datagrams across multiple IEEE 802.15.4 frames and support the IPv6 1280-byte minimum MTU requirement.
- **Layer-Two Forwarding**: To support MPLS-like organizations where routing occurs at layer three and forwarding occurs at layer two.
- **Header Compression**: To reduce header overhead and the need for fragmentation by applying cross-layer optimizations and compressing commonly used values. The header compression supports both flow-independent and flow-based mechanisms. The former minimizes network state and gives the network layer maximum flexibility to dynamically change next hop routes. Flow-dependent compression, however, takes advantage of redundancies within a flow.

5.2 Delivering IPv6 Datagrams

To deliver IPv6 datagrams using IEEE 802.15.4 frames, the adaptation layer's header format must be expressive enough to support all of the adaptation layer's functionality. To be effective, especially in the sensornet domain, the header format must also be simple to parse, allow compact forms when only some

mechanisms are in use, and leave room for future extensibility. In this section, we present a header format that meets these goals.

5.2.1 Background

The initial 6LoWPAN header format was derived from an adaptation layer proposed within the IETF for IPv6 over IEEE 1394 (FireWire) [90]. The IEEE 1394 adaptation header only supports fragmentation of IPv6 datagrams. The first two bits in the header format distinguish different headers for an unfragmented datagram or the first, last, and interior fragments of a fragmented datagram. A full fragment header includes a size, tag, and offset and specifying different fragment headers allow compact forms when not all fields are needed. The IEEE 1394 adaptation header is simple and easy to parse, in large part because it only provides fragmentation.

The initial 6LoWPAN header also used the first two bits to indicate which fragmentation fields appeared in the header. But other than the first two bits, the two header formats differ greatly. Unlike IEEE 1394 adaptation, 6LoWPAN seeks to also support layer-two forwarding, as well as IPv6 and upper-layer header compression. Addressing, sequence number, and hop limit fields were added, as well as a protocol type field to indicate whether IPv6 header compression was in use. In developing the 6LoWPAN header format, fields were added and moved around as implementors discovered deficiencies in the format. With each tweak, the resulting header format grew increasingly complex and hard to parse.

The format that was subjected to working group last call is shown in Figure 5.1 and had a number of deficiencies. First, the header format was fairly difficult to parse. Orthogonal mechanisms such as fragmentation, layer-two forwarding, and multicast support were distributed across the entire header format. The location of specific header fields depended on the inclusion of other header fields. Furthermore, some header fields were not byte-aligned. The lack of a clean structure translates directly into increased code and memory requirements to process the header. Second, the header format was not compact, always including fields for fragmentation and layer-two forwarding even when neither are required. Finally, the header leaves no room to define additional functionality, which is especially important due to relative inexperience of using low-power

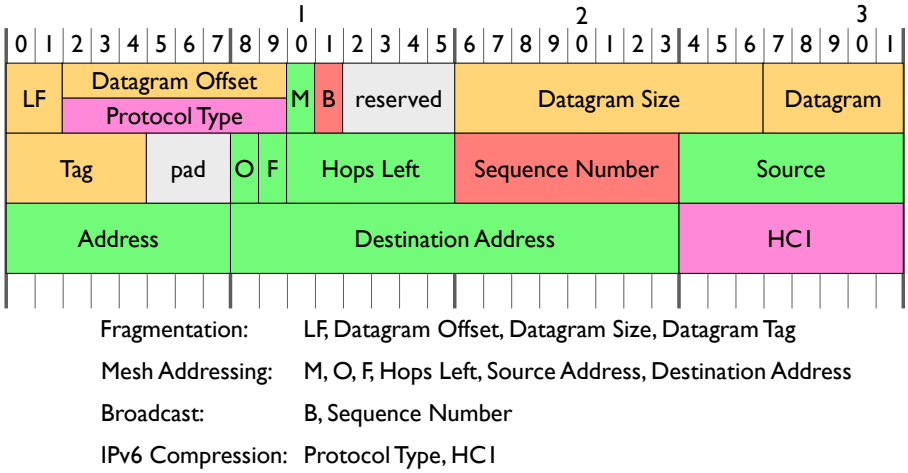


Figure 5.1: **Initial 6LoWPAN Header Format.** The initial 6LoWPAN header format was based on the IEEE 1394 adaptation format, but extended to include layer-two forwarding and upper-layer header compression [90]. The result of many minor tweaks, the header format intertwined orthogonal concepts and was not the most compact in simple cases.

wireless links in IP networks. For example, the header left no provisions to support additional forwarding (e.g. source-based forwarding and QoS) or configuration and management (e.g. ICMP) mechanisms.

Given the significant issues with the proposed 6LoWPAN header format, we decided to do a clean-slate design. In defining a new format, we were inspired by the clean structure of the IPv6 header. IPv6 defines a base header, which includes fields that must be included with every IPv6 datagram such as source and destination IPv6 addresses. IPv6 also defines extension headers that may be appended to the base header when additional functionality must be expressed. Defined extension headers include hop-by-hop options, routing, fragment, and destination options. IPv6 allows multiple extension headers to be appended, one after the other, as shown in Figure 5.2. Each header includes a *Next Header* field used to identify the following header. We call this approach *header stacking*.

Header stacking has two advantageous properties. First, header stacking keeps orthogonal concepts separate. Addressing fields are contained in the base header, while routing and fragment fields are contained within their respective extension headers. Second, header stacking naturally elides headers that are not required simply by not including them, resulting in compact forms when not all of the functionality is

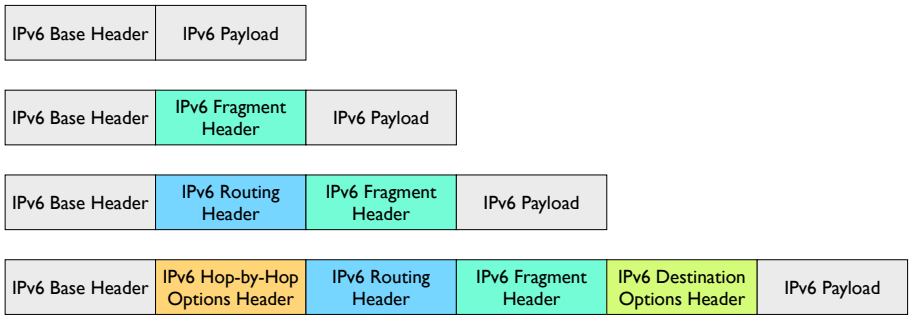


Figure 5.2: **IPv6 Header Stack.** IPv6 defines a base header, as well as a number of extension headers that may be "stacked" on to the base header. *Header stacking* separates orthogonal functionality, naturally provides compact forms when not all functionality is used, and leaves room for future extensibility.

used. Third, the stacked header leaves room for defining new functionality in the future by defining new Next Header values.

In the following sections, we present a new header format that supports all functionality provided by the initial format but provides a clean, easy-to-parse structure, more compact form, and room for extensions. This new format was adopted as a proposed standard by the IETF with RFC 4944 [117].

5.2.2 Header Stacking Format

Our 6LoWPAN header format uses a header stacking format inspired by IPv6. We initially define three headers, one to express each of the mechanisms we intend to support in the adaptation layer. The *Fragment* header is used whenever the payload cannot be carried in a single IEEE 802.15.4 frame. The *Mesh Addressing* header is used whenever IEEE 802.15.4 frames are delivered over multiple radio hops to support layer-two forwarding. Finally, the *IPv6 Header Compression* header is used to compress the IPv6 header of the encapsulated datagram.

The 6LoWPAN header stack may include some, all, or none of the 6LoWPAN headers. When multiple 6LoWPAN headers are included, they must appear in the following order: Mesh Addressing, Fragment, and IPv6 Header Compression. The ordering constraint seeks to simplify header processing implementations. Typical header stacks are shown in Figure 5.3. Each header is identified by a variable length *header type*, carried at the start of each header. Headers we expect will be used most often are assigned shorter

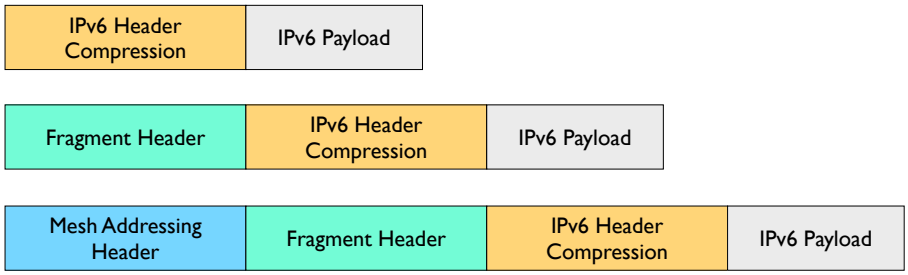


Figure 5.3: **6LoWPAN Header Stack.** The stacked 6LoWPAN header format keeps orthogonal concepts separate using different headers. A stacked header format naturally provides a compact form in the simplest case by eliding headers for those mechanisms not in use.

header types, trading reduced header overhead for increased overhead in less frequently used headers. New headers may be defined in the future by allocating a new header type. One header type is reserved to allow non-6LoWPAN protocols operating alongside with 6LoWPAN, similar to the relationship between IPv4 and ARP. Unlike Ethernet and WiFi, the IEEE 802.15.4 header does not carry a protocol type field.

Fragment

The Fragment header is used when the payload is too large to fit in a single IEEE 802.15.4 frame. The Fragment header is analogous to the IEEE 1394 Fragment header and includes three fields: *Datagram Size*, *Datagram Tag*, and *Datagram Offset*. Datagram Size identifies the total size of the unfragmented payload and is included with every fragment to simplify buffer allocation at the receiver when fragments arrive out-of-order. Datagram Tag identifies the set of fragments that correspond to a given payload and is used to match up fragments of the same payload. Datagram Offset identifies the fragment's offset within the unfragmented payload and is in units of 8-byte chunks. To allow arbitrary byte offsets would require 11 bits to support the 1280 byte minimum MTU requirement, but requiring 8-byte alignment requires only 8 bits for the offset.

The Fragment header format is shown in Figure 5.4. The header type is only two bits. The third bit is used to compress the datagram offset field when it is zero, as it always is zero in the first fragment. All other fragments cannot elide the datagram offset. The Fragment header is 4 bytes for the first fragment and 5 bytes for the remaining fragments.

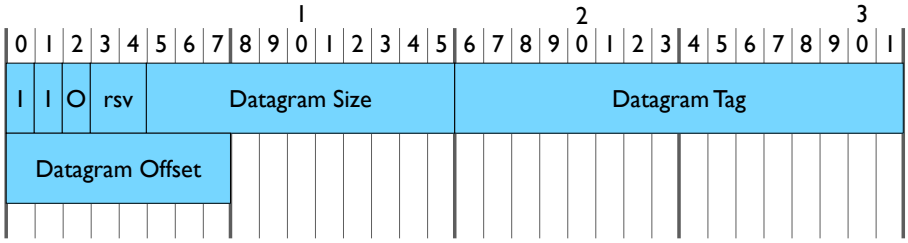


Figure 5.4: **Fragment Header.** The Fragment header support IPv6 datagram fragmentation and includes the size and tag for the fragmented IPv6 datagram and an offset within that datagram.

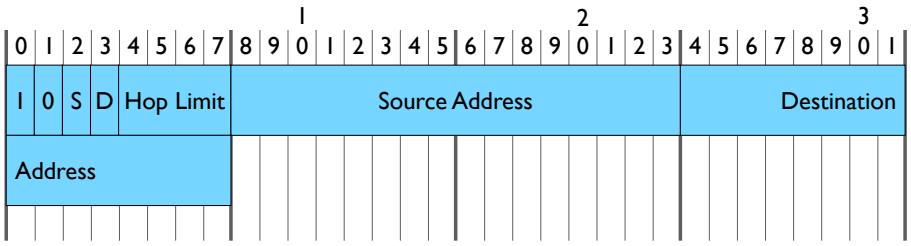


Figure 5.5: **Mesh Addressing Header.** The Mesh Addressing header supports layer-two forwarding of IPv6 datagrams and includes the source and destination addresses of the IP hop and a hop limit.

Mesh Addressing

The Mesh Addressing header is used to forward 6LoWPAN payloads over multiple radio hops and support layer-two forwarding. The mesh addressing header includes three fields: *Hop Limit*, *Source Address*, and *Destination Address*. The Hop Limit field is analogous to the IPv6 Hop Limit and limits the number of hops for forwarding. The Hop Limit field is decremented by each forwarding node, and if decremented to zero the frame is dropped and not forwarded any further. The source and destination addresses indicate the end-points of an IP hop. Both addresses are IEEE 802.15.4 link addresses and may carry either a short or extended address.

The Mesh Addressing format is shown in Figure 5.5. The header type is only two bits. The third and fourth bits indicate which addressing mode to use for the source and destination addresses. The following bits carry the hop limit and addressing fields. The mesh addressing header ranges between 5 and 17 bytes depending on the addressing modes in use.

IPv6 Header Compression

The IPv6 Header Compression header is used to compress an IPv6 header carried in the 6LoWPAN payload and is identified by an 8-bit header type. A shorter header type could be used, but we left flexibility in defining other header compression formats. IPv6 Header Compression marks the end of a 6LoWPAN header stack, indicating a transition from layer-two to layer-three in the packet processing. No 6LoWPAN headers may be included after the IPv6 Header Compression header. We present the header compression format in Section 5.3, as header compression deserves its own section.

5.2.3 Layer-Two vs. Layer-Three Forwarding

The 6LoWPAN adaptation layer provides a layer-two forwarding mechanism but does not mandate that it should be used. The sensornet and IETF communities, have not yet reached consensus on whether forwarding is best provided by layer two or layer three. Both have their benefits.

With forwarding at layer two, the network operates much like MPLS. But rather than arbitrary labels, forwarding operates directly on IEEE 802.15.4 link addresses. One potential advantage of layer two forwarding is that 6LoWPAN fragments can be delivered over multiple hops without requiring fragmentation or reassembly at each hop. In addition to keeping forwarding functions simple, layer-two forwarding allows the use of multiple paths to deliver fragments for a given datagram. Multipath routing has shown promise in being able to increase aggregate throughput and reduce average energy and memory requirements among forwarding nodes. However, multipath forwarding presents its own challenges, with inter-path interference, out-of-order delivery, and flow control over multiple paths.

Another potential advantage of layer-two forwarding is to enable different service classes and traffic engineering, as is done with MPLS. While the Mesh Addressing header does not carry a simple label in the way that MPLS does, it can be used to implement such a feature by assigning nodes multiple link addresses. In effect, the different link addresses specify different ways of forwarding a datagram to the same destination. Alternatively, we could define a new 6LoWPAN header to carry labels as MPLS headers do. The challenge

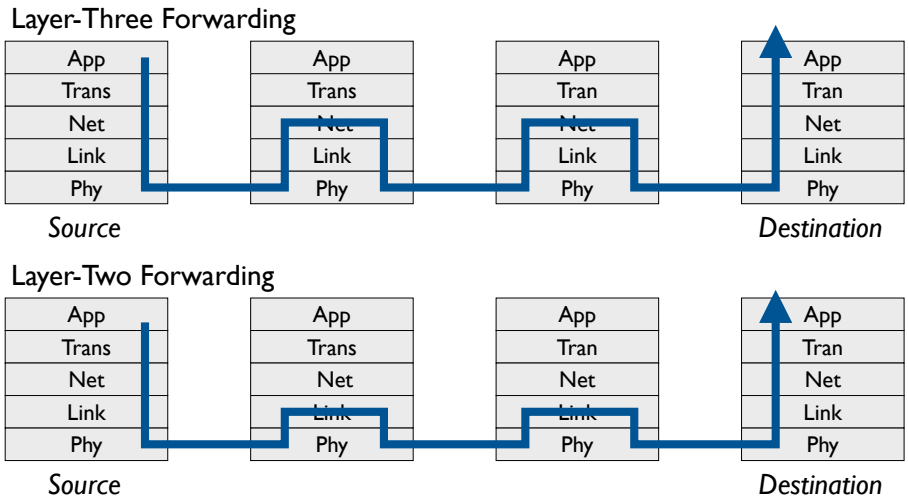


Figure 5.6: **Layer-Two vs. Layer-Three Forwarding.** Layer-two forwarding delivers IEEE 802.15.4 frames over multiple radio hops, effectively abstracting multiple radio hops as a single IP hop. Layer-three forwarding only delivers IEEE 802.15.4 frame over a single radio hop and IP is responsible for forwarding datagrams over multiple radio hops.

of supporting multiple services classes and traffic engineering is to distribute labels to forwarding nodes in a cost-effective manner.

Layer-three forwarding operates with each radio hop also serving as an IP hop. Layer-three forwarding may seem more resource intensive than layer-two forwarding, simply because it must operate on IPv6 information that is generally larger than layer-two information (e.g. addresses). However, as discussed in Chapter 3, we've specified an addressing architecture that makes very little difference between them. With common global routing prefixes assigned to all nodes and Interface Identifiers derived from link addresses, layer-three forwarding actually operates over link addresses rather than arbitrary IPv6 addresses. Other fields include Version, Traffic Class, Flow Label, and Hop Limit. As we will see in the following section, the former three fields may be compressed in most cases, while Hop Limit exists in both IPv6 and Mesh Addressing headers. Layer-three forwarding can actually be more header efficient, as the Mesh Addressing header is not included.

The only potential drawback of layer-three forwarding is that it requires 6LoWPAN fragmentation and reassembly at every radio hop. A naive approach performs complete reassembly at every hop along

the path, but doing so places significant buffering requirements at forwarding nodes. An alternate approach allows a forwarding node to forward a fragment to the next hop, without requiring complete reassembly. The initial fragment contains the IPv6 header, which the forwarder can use to determine a next-hop node to forward the datagram. To forward the first fragment, the forwarder decrements the IPv6 Hop Limit, updates the Datagram Tag in the Fragment header, and submits the frame for transmission to the next hop. Because only the first fragment contains addressing information, the forwarder must also establish a mapping between a datagram tag and source address pair to a different datagram tag and destination address pair. Some additional state is required to implement a timeout. In effect, the initial fragment establishes a virtual circuit for the remaining fragments.

By not reassembling the datagram, the forwarder trades at least 1280 bytes to buffer a complete IPv6 datagram for little over 20 bytes to maintain the tag and address mapping. Furthermore, the reduced state is short-lived, being reclaimed after forwarding the last fragment or the timeout expires. While layer-three forwarding constrains fragment delivery to a single path, there are a number of benefits of utilizing only a single path. First, fragments are delivered in order, simplifying reassembly at the destination. Second, forwarding along a single path allows the use of link-layer optimizations to reduce transmission overhead and increase throughput. With Emnet, for example, forwarders can maintain synchronization state for the next hop to reduce transmission overhead, utilize streaming to increase throughput, and reduce the channel sample period to reduce latency. Applying such optimizations in a multipath settings wouldn't yield as much of a benefit, especially when the number of fragments is small - only one or two fragments will be delivered over each path. Finally, transmitting all fragments along a path takes advantage of the temporal properties of wireless links. Links that are performing well are likely to continue performing well in the near future.

5.2.4 Related Work

Many link technologies do not require much of an adaptation layer to support IPv6. The only real requirement is an intermediate encapsulation header that distinguishes IPv6 datagrams from other protocols that operate directly on the link (e.g. ARP, BOOTP, and DHCP). Ethernet uses EtherType, a two-byte

identifier. InfiniBand uses a four-byte header, two of which carry the EtherType field [17]. Many other link technologies use the IEEE 802.2 LLC and SNAP headers to encapsulate IP datagrams, including WiFi (802.11), Token Ring (802.5) [25], FDDI [23, 95], Fibre Channel [34], IPX [113], and SMDS [135].

For links with smaller MTUs, adaptation layers that sit above them often implement fragmentation. While IPv4 only has a 576 byte minimum MTU requirement, supporting larger MTUs at the link-layer minimizes costly fragmentation across a network built with different but more capable links. IPv6 has a 1280-byte minimum MTU requirement and makes adaptation layer fragmentation mandatory for links that have smaller MTUs. Existing fragmentation methods differ in how much information they put in each frame.

ARCnet uses a single *split* field to support fragmentation [143]. For unfragmented datagrams, the split field is set to zero. On the first fragment, the split field is set to the number of fragments to expect. On subsequent fragments, the split field is set to the fragment number in the sequence. The sequence number field carries the same value in all fragments. IEEE 1394 (FireWire) is similar to the monolithic format initially proposed in 6LoWPAN [90]. The first two bits in the adaptation header indicate whether the frame contains an unfragmented datagram, first fragment, or subsequent fragment. All fragments carry a datagram size and label fields, while subsequent fragments carry an offset field. Asynchronous Transfer Mode (ATM) uses small, fixed-sized cells to communicate (48-byte payload MTU), and requires fragmentation to communicate IP datagrams [148]. AAL5 used only use a bit in ATM's cell header to indicate a last fragment that contains a payload length and CRC to ensure complete and in-order fragment delivery. AAL5 eliminates per-cell overhead by assuming mostly reliable and in-order fragment delivery.

5.3 Compressing IPv6 Datagrams

While the adaptation layer allows communication of IPv6 datagrams using IEEE 802.15.4 frames, IPv6 header compression is a necessary to make IPv6 communication feasible in sensornets. Network- and transport-layer headers are large. A standard UDP/IPv6 header is 48 bytes and is shown in Figure 5.7. In this section, we present a method for compressing network and transport-layer headers.

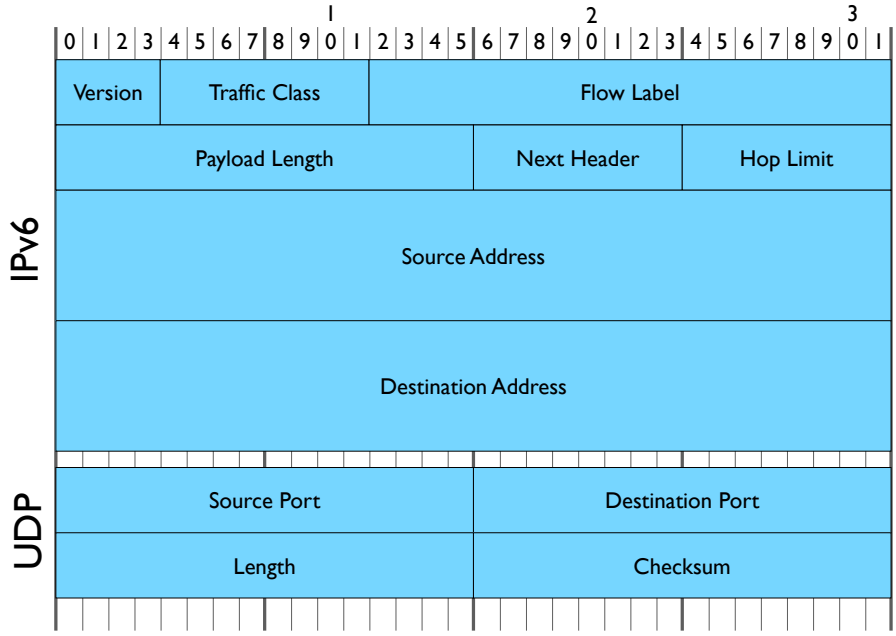


Figure 5.7: **UDP/IPv6 Header.** The IPv6 and UDP headers are relatively large at 40 and 8 bytes, respectively. The IPv6 addresses alone consume 32 bytes. Network- and transport-layer header compression is necessary for efficient operation.

5.3.1 Background

Flow-Based Compression

Traditional IP header compression techniques are *flow-based*, learning compressing redundancies within a flow [11, 32, 86]. Flow-based compression techniques are optimized for long-lived flows and assume that the compressor and decompressor are in direct and exclusive communication over a highly constrained link (e.g. modem over phone line). Flow-based compression operate across layers, compressing both network and transport headers. By learning redundancies within a flow, compression ratios increase over time and can achieve a single byte for both headers in the best case.

Flow-based compression has been successful and widely used, but are ill-suited for sensornets, where low-rate multihop communication involving many nodes is common. Flow-based compression requires both compressor and decompressor to establish and maintain per-flow state. In a multihop deployment, compression and decompression must happen hop-by-hop, implying that forwarders must maintain per-flow

state at every hop. State requirements at forwarding nodes can become prohibitive in large networks, even if each node was an end-point for a single flow. Compression state for each flow must be consistent between the compressor and decompressor, requiring a link with high best-effort packet delivery for efficient operation. Significant loss rates can greatly reduce compression benefits, as additional effort is spent to re-synchronize compression state between the end-points. Nodes must also reestablish per-flow state when changing routes, which limits compression in mobile network and constrains the frequency at which routes can change.

Originally meant to optimize long-lived and relatively high-rate flows, existing flow-based compression mechanisms are not well suited for the low-rate flows typical in sensornets. Many sensornet applications only report data in a single direction, relying on high best-effort packet delivery rather than end-to-end reliability mechanisms. The latter generally incurs significant overhead. Due to low data rates in the forward direction, each message requiring an explicit acknowledgment. Existing flow-based compression techniques compress values by observing commonalities within a flow over time. Low data rates also imply that flow-based compression can take significantly longer to converge on a high compression ratio.

Flow Independent, Stateless Compression

The 6LoWPAN working group initially defined a *stateless* header compression mechanism for IPv6 and UDP headers in RFC 4944 [117]. By definition, stateless header compression does not maintain any per-flow state and hence is flow-independent. RFC 4944 compresses datagrams by exploiting redundancies across layers, including the link, network, and transport layers. Length fields are always elided, assuming that they could be derived from lower-layer headers, the IPv6 Interface Identifier could be elided when they are derived from addresses carried in the link layer. Other fields are compressed by assuming common values and compressing the common case. For example, IPv6 Version is assumed to be 6, Traffic Class and Flow Label are assumed to be zero, Next Header is assumed to be UDP, TCP or ICMPv6, and the prefix for both source and destination addresses are assumed to be the link-local prefix. In the best case, RFC 4944 can compress a UDP/IPv6 header down to 7 bytes, and the encoding byte is shown in Figure 5.8.

Figure 5.8: **Stateless Header Compression.** RFC 4944 compresses IPv6 and UDP headers by exploiting redundancies across layers and assuming commonly-used values for other header fields. However, RFC 4944 only allows compression of link-local IPv6 addresses, meaning that RFC 4944 can not compress addresses carrying a global routing prefix.

RFC 4944 provides effective compression for link-local communication, but has little effectiveness when communicating beyond link-local scope. RFC 4944 carries the IPv6 address in full, when the IPv6 address carries a global routing prefix. Furthermore, RFC 4944 does not compress IPv6 multicast addresses. IPv6 protocols often use link-local multicast for discovery and address-free communication. IPv6 addresses account for 80% of the IPv6 header, making their compression essential. To realize the full benefit of an IP-based network, sensornet nodes must be able to communicate effectively when communicating with arbitrary IP devices connected to other IP networks. This implies that the compression mechanism must be effective for global and multicast addresses as well.

Shared-Context Compression

Like stateless compression, shared-context compression is also flow-independent but exploits commonalities in headers across all flows within a network [184]. As a result, shared-context compression requires all nodes to establish some shared-context. This is in contrast to flow-based compression where only the compressor/decompressor of a given flow establish and maintain state. For example, all interfaces in a network are generally assigned with IP addresses that share a common global routing prefix. As a result, nodes within the sensornet can exploit this shared context to compress common prefixes that appear in the header. For stub networks, all communication into and out of the network will carry at least one common prefix. For communication within the network, both source and destination addresses will carry the common prefix.

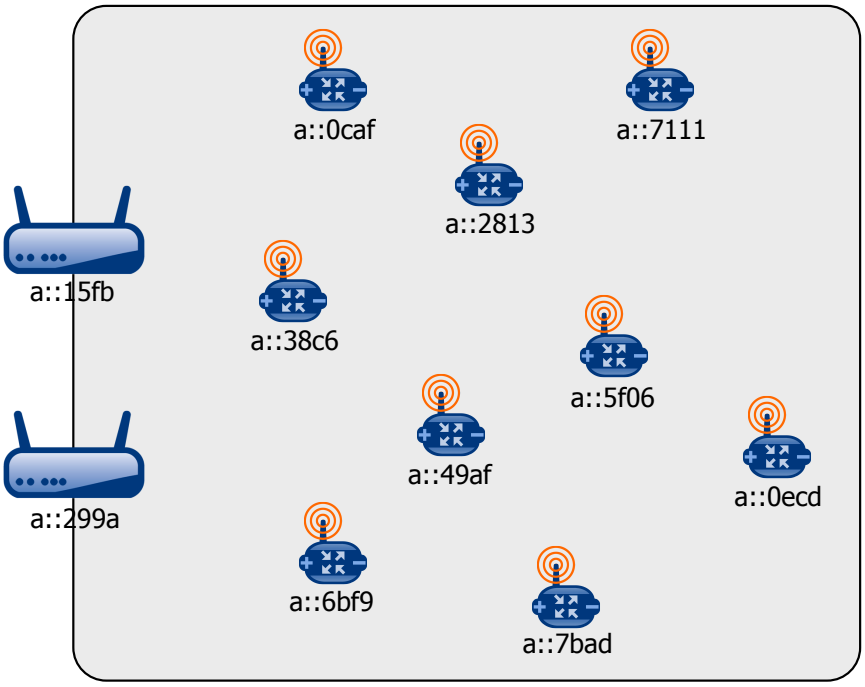


Figure 5.9: **Shared-Context Compression.** Network-layer headers for datagrams transmitted in the same network have high correlation regardless of flow, such as global routing prefixes carried in IPv6 addresses, as shown in the figure. Shared-context compression is flow-independent and takes advantage of the correlations to compress headers without requiring per-flow state.

Hybrid Compression

Stateless and shared-context compression works well at the network layer since network-layer headers often have significant commonalities across all communication flows within a network. However, transport-layer headers have much more commonality within a particular flow than across flows. Examples include the source and destination ports, as well as sequence numbers. As a result, it may be desirable to support stateful compression for transport-layer headers in combination with stateless and shared-context compression at the network layer [183]. One nice property of stateful compression at the transport layer is that any compression state is only maintained at the end-points. Furthermore, reliable transport layers require bidirectional communication, which can be relied upon to establish and maintain compression state.

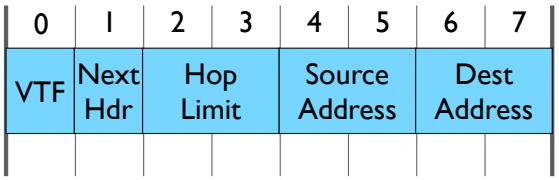


Figure 5.10: **IPv6 Header Compression.** The IPv6 compression encoding uses a single byte to indicate the compression of Version, Traffic Class, Flow Label, Hop Limit, Next Header, and Source and Destination address fields. Payload Length is always elided and a 40-byte IPv6 header can be compressed down to a single byte, including the encoding byte itself.

5.3.2 IPv6 Header Compression

We present a header compression scheme, LOWPAN_HC, for an IPv6 network over IEEE 802.15.4 radios. LOWPAN_HC builds on RFC 4944 and generalizes it to support both global and multicast communication in addition to link-local communication. We incorporate existing work in hybrid compression, but extend it to support both stateless and stateful techniques at both the network and transport layers.

Like RFC 4944, LOWPAN_HC exploits redundancies between the link and network headers. Specifically, *Payload Length* may be derived from the link-layer header. When datagrams fit within a single IEEE 802.15.4 frame, the Payload Length can be derived from the IEEE 802.15.4 header. When fragmented, the Payload Length may be derived from the Datagram Size field of the 6LoWPAN Fragment header. The *Interface Identifier (IID)* in unicast addresses may also be derived from the link-layer header. The direct mapping between IIDs and link addresses is described in Chapter 3.

For other fields, LOWPAN_HC assumes common values and either elides them completely or specifies compressed forms of those common values. For IPv6, *Version* is always 6 and LOWPAN_HC elides it completely. LOWPAN_HC assumes that *Traffic Class* and *Flow Label* are normally zero. LOWPAN_HC also assumes that the global routing prefix for both *Source* and *Destination* addresses normally match a common prefix assigned to the sensornet. Finally, LOWPAN_HC supports compression of arbitrary next headers, such as UDP or IPv6 extension headers. When the next header is compressed, *Next Header* is elided and carried in compressed form using an encoding we describe in the following section.

The resulting encoding for IPv6 compression is shown in Figure 5.10. The first bit indicates if Traffic Class and Flow Label are 0 and elided or carried inline. When Traffic Class and Flow Label are carried inline, Version is also carried inline to maintain byte-alignment, a property not maintained in RFC 4944. The second bit indicates whether next header compression is used, and if so, Next Header is elided. The third and fourth bits indicate whether or not the Next Header is elided or carried inline. When elided, the compressed form indicates whether the Next Header is 1, 64, or 255, the most common initial values for Next Header used by IP-based protocols. Finally, two bits are used for each address and indicates whether the compressed header carries the full 128-bit address inline or is compressed to a shorter form.

Unicast Address Compression

LOWPAN_HC uses shared-context compression to reduce the size of IPv6 addresses when the prefix is either link-local or the common prefix. LOWPAN_HC elides the 64-bit prefix and leaves the full 64-bit IID, lower 16 bits of the IID, or no bits at all. The 64-bit prefix is identified by using two different 6LoWPAN header types, which indicates link-local or global scope. The common prefix may be distributed with autoconfiguration, as we discuss in Chapter 6. When configuring addresses, it is possible for neighboring nodes to disagree on the common prefix. While the decompressor may reconstruct the IPv6 address using the wrong global prefix, end-to-end integrity checks at the transport-layer (e.g. UDP checksum) will likely catch such errors. LOWPAN_HC compresses IPv6 addresses to 64 bits when the IID is derived from an extended link address and 16 bits when the IID is derived from a short link address. Finally, the IID may be compressed to zero bits if it can be derived directly from the IEEE 802.15.4 header or the 6LoWPAN Mesh Addressing header.

To compress IPv6 addresses external to the sensornet, LOWPAN_HC establishes mappings between a 16-bit short address identifier with arbitrary IPv6 addresses. To communicate the mapping to all nodes within the sensornet, we use Neighbor Discovery's Router Advertisement messages. The 16-bit short address identifier is carried in the compressed IPv6 header using the 16-bit compressed form of the IPv6 address. To distinguish between internal and external addresses, we divide the 16-bit name space into multiple ranges.

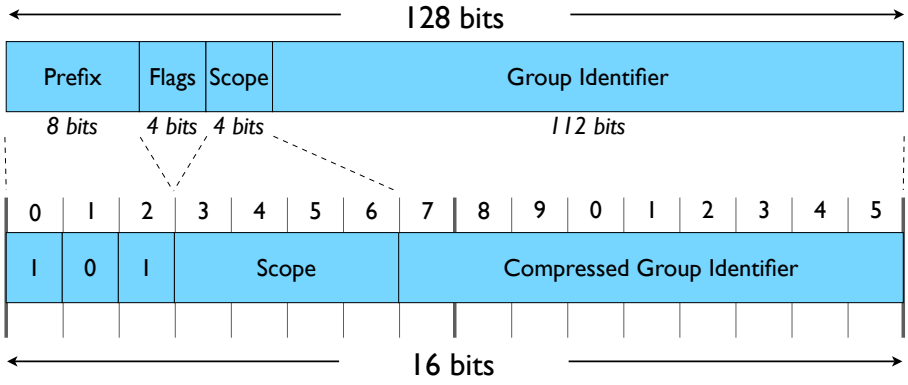


Figure 5.11: **Multicast Address Compression.** LOWPAN_HC compresses 128-bit well-known multicast addresses down to 16 bits. Discovery and communication protocols often use well-known multicast addresses. LOWPAN_HC assumes that the 4-bit Flags field is zero, carries the 4-bit Scope field inline, and maps the 112-bit Group Identifier down to 9 bits.

LOWPAN_HC identifies internal addresses with a '0' as the first bit and external addresses with '100' as the first three bits. Dividing the address space in this way requires short link addresses to carry a '0' in the first bit.

Multicast Address Compression

LOWPAN_HC compresses well-known IPv6 multicast addresses down to 16 bits, using the 16-bit compressed form. LOWPAN_HC identifies compressed multicast addresses by setting the first three bits to '101'. Of the remaining bits, LOWPAN_HC uses four bits to carry the full 4-bit multicast scope inline and uses the last 9 bits to indicate the specific multicast group identifier. The multicast flags are completely elided and assumed to be all zeros, indicating a permanently assigned multicast address, that the multicast address is not assigned based on the network prefix, and that it doesn't embed the address of a Rendezvous Point. The mapping between 112-bit and compressed 9-bit group IDs must be well-known. Finally, when subscribing to a well-known multicast address, the node must understand both compressed and uncompressed forms. Without doing so, it is impossible to tell if the node has a 9-bit mapping for the subscribed address.

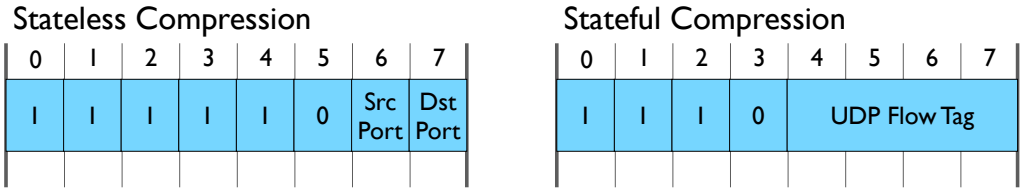


Figure 5.12: **UDP Header Compression.** An encoding for compressing next headers must carry an identifier in the initial bits. We define both a stateless and stateful compression for UDP headers. The stateless version compresses ports in a subset of the ephemeral port range. The stateful version compresses both ports down to a single label. Both versions always compress UDP Length, but never compress UDP Checksum. Future specifications may define encodings for other Next Header values.

5.3.3 Next Header & Transport-Layer Compression

LOWPAN_HC introduces a framework for allowing compression of arbitrary next headers, whereas RFC 4944 only allows compression of UDP, TCP, and ICMPv6. LOWPAN_HC allows next header compression by utilizing a bit in the IPv6 encoding to indicate whether the next header is compressed. If so, the bits immediately following the compressed IPv6 header form a variable-length next header identifier. The identifier specifies the next header being compressed and the compression method. A variable-length identifier allows LOWPAN_HC to optimize for frequency of using a particular next header and the number of bits required for compression.

We initially define UDP header compression, as its connection-less property is well-suited for many sensornet applications. Like the network layer, UDP compression can either be stateless or stateful. The UDP header is 8 bytes and contains a Source Port, Destination Port, Length, and Checksum. Both stateless and stateful compression always elide *Length* and derives it from lower-layer headers. *Checksum*, however, is always carried inline, as it is required for IPv6 and provides protection against LOWPAN_HC decompression errors. Stateless compression assumes a common value for the upper 8 bits of the *Source Port* or *Destination Port*, which is in the ephemeral range. When both ports carry are within the common range, LOWPAN_HC compresses the UDP header to three bytes. The encoding for stateless UDP compression is shown in Figure 5.12.

Stateful UDP compression allows LOWPAN_HC to compress the UDP header down to two bytes, regardless of what ports are used. Because both source and destination ports are static within a flow, LOWPAN_HC compresses both down to a single tag. Nodes initially communicate using the stateless form. Nodes can negotiate a tag by sending an ICMPv6 messages. When communicating with IP devices outside the sensornet, LOWPAN_HC relies on border routers for stateful compression. Putting the translation at the border router rather than at the end-point maintains compatibility with existing IP devices. The encoding for stateful UDP compression is shown in Figure 5.12.

In general, stateful compression works well at the transport-layer and above, where headers in different packets among a flow have very high correlation. We envision similar approaches to compressing transport-layer (e.g. TCP) headers and more generally any next header, including IPv6 extension headers. A significant body of work has focused on stateful compression for transport-layer headers, and future definitions for other headers can draw on that work [11, 32].

5.3.4 Related Work

Efficient IP datagram communication over highly constrained links has been an area of concern for quite some time. Some of the earliest work in reducing header overhead began with Thinwire when readily available remote links were around 1200 - 9600 baud connections [53]. Thinwire compressed packets by transmitting only fields that have changed. Jacobson improved Thinwire's compression scheme by incorporating difference encoding, reducing header overhead from 13 to 3 bytes [86]. IP Header Compression (IPHC) further extended Jacobson's initial work to include support for IPv6, UDP, and arbitrary headers identified by the Protocol Type [32]. To increase TCP performance, IPHC introduced TWICE and the ability to request full header frames. For non-TCP traffic, headers are periodically sent uncompressed to limit errors caused by packet drops. RObust Header Compression (ROHC) observed that vulnerability to packet loss is primarily due to difference encoding, and IPHC's loss-recovery optimizations require duplex links [11]. Instead, ROHC uses Least Significant Bit (LSB) encoding to be robust to short bursts in packet loss. ROHC also provides a framework using different loss-recovery techniques depending on link-specific characteristics.

Traditional IP header compression techniques all operate on individual flows by observing common values within a given flow and eliding them when possible. Flow-based compression is also sensitive to packet losses, as each frame carries important information to keep both compressor and decompresser in sync. Three error recovery techniques may be used to mitigate packet loss: (i) frequently transmitting uncompressed headers, (ii) rely on feedback from the decompresser, (iii) and some hybrid of the two. Choosing between these approaches depends on link-specific characteristics. Because transport-layer headers have high correlation in the time domain, LOWPAN_HC allows more traditional compression techniques at the transport-layer. Like IPHC and ROHC, LOWPAN_HC uses different compression schemes for UDP and TCP.

Largely due to HTTP, traffic characteristics have shifted more to short-lived flows. As a result, Frequency-Based Header Compression (FBHC) was introduced to keep state across flows and minimize uncompressed headers at the beginning of each flow [182]. Rather than maintaining per-flow state, FBHC advocated per-user state, as flows from the same user have greater commonality. But even per-user state was problematic for access routers that must manage thousands or millions of clients, as is the case in cellular networks. To minimize state further, Stateless Header Compression (STHC) uses existing route table and ARP cache information to compress packets without requiring per-flow or per-user state [184]. The main observation is that there are many similarities in the network-layer header independent of flow, making effective compression independent of flow possible. But STHC does not address transport-layer compression, as they tend to experience greater commonality within the same flow. Like STHC, LOWPAN_HC compresses the network-layer header independent of flows, but STHC assumes that compression only occurs on a single link. Instead, LOWPAN_HC includes necessary fields such as Source Address and Hop Limit to support multihop communication.

Most recently, a hybrid approach (labeled HHC) that incorporates both flow-independent and flow-based compression has been proposed for wireless sensor networks [183]. Flow-independent compression is used for the network-layer, while per-flow compression is used at the transport layer. However, HHC relies on a flow register to aggregate network-layer information across all flows in the PAN. The flow register may

Functionality	**Header Overhead** (bytes)	
	minimum	maximum
Fragmentation	4	5
IPv6	2	40
Layer-Two Forwarding	5	17
Layer-Three Forwarding (Intra-PAN)	4	16
Layer-Three Forwarding (Internetwork)	4	32
UDP (Stateless)	5	7
UDP (Stateful)	3	7

Table 5.1: **Header Overhead for Specific Functionality.** This table shows the header overhead for specific functionality. The header stack format allows easy inclusion and exclusion of specific functionality provide compact headers. IPv6 represents the header-type and LOWPAN_HC encoding fields. Layer-Two forwarding represents the Mesh Addressing header. Layer-three forwarding represents the addressing overhead. UDP represents the next-header identifier, encoding, uncompressed portions of the UDP port, and the UDP checksum.

be centralized or distributed, but nodes must query information from the flow register before headers may be compressed. Reliance on the flow register increases state and communication overhead. Inspired by HHC, LOWPAN_HC also uses a hybrid compression scheme. But rather than using a flow register, LOWPAN_HC relies on configuration of common prefixes to allow nearly stateless compression. Link-local and network prefixes may be compressed without any state or communication because of a common understanding of how prefixes are assigned. LOWPAN_HC also realizes greater compression ratios by heavily exploiting redundancies in the link-layer header. Minimal state may be added to LOWPAN_HC to optimize compression of arbitrary addresses.

5.4 Header Overhead Analysis

The adaptation layer provides support for network- and transport-layer header compression, datagram fragmentation, and layer-two forwarding. The header stack format provides a well-defined mechanism for including extra functionality when needed and leaving out functionality that is not needed. We show the header overhead for various functionality in Table 5.1.

The fragmentation header consumes 4 to 5 bytes, depending on whether it includes the offset field. The adaptation layer supports layer-two forwarding through the Mesh Addressing header, which requires 5 to

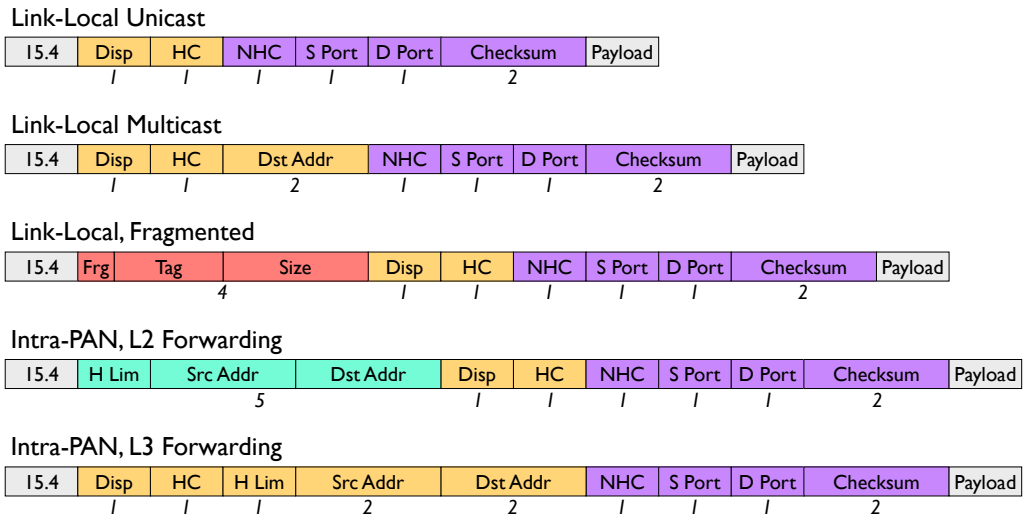


Figure 5.13: **Headers for Common Communication Patterns.** Link-local unicast allows for the most compact forms, compressing a UDP/IPv6 header down to 7 bytes. Link-local multicast requires an additional 2 bytes to carry a compressed multicast address. Fragmented datagrams require another 4 bytes in the first fragment. Subsequent fragments carry a 5-byte fragment header, but do not compressed IPv6 or UDP headers. Communication over multiple hops requires a hop limit as well as addressing information.

17 bytes, depending on what link address types the header includes. Layer-three forwarding occurs directly on IP, and with LOWPAN_HC only consumes 4-16 bytes when forwarding within the PAN and 4-32 bytes when forwarding outside the PAN. Intra-PAN communication includes either the 16-bit or 64-bit address forms, while internetwork communication may include full 16-byte IPv6 addresses. In general, one address will include the common prefix and LOWPAN_HC will compress it. Furthermore, if context is established for destinations external to the sensornet, LOWPAN_HC can compress both addresses down to 2 bytes each.

The total header overhead involves the sum of one or more headers in the header stack. Header stacks used with common communication patterns are shown in Figure 5.13. We show the total header overhead for representative header stacks in Table 5.2. The analysis assumes the use of short address in the Mesh Addressing header or IPv6 addresses, allowing the most compact forms. Link-local UDP/IPv6 communication only requires 7 bytes of total overhead, 1 byte for the LOWPAN_HC header type, 1 byte for the LOWPAN_HC encoding, 1 byte for the UDP next header identifier, 2 bytes for the UDP ports, and another 2 bytes for the UDP checksum. Global communication consumes 12 to 28. The added overhead includes the

Scenario	**Typical Header Overhead** (bytes)
Link-Local Unicast	7
Link-Local Multicast	9
Link-Local, Fragmented	11
Intra-PAN, L2 Forwarding	12
Intra-PAN, L3 Forwarding	12
Internetwork, L2 Forwarding	28
Internetwork, L3 Forwarding	26

Table 5.2: **Typical Header Overhead.** This table shows the header overhead for typical header stacks. The analysis assumes IPv6 addresses that include IIDs derived from short 16-bit addresses. In general, overhead will range between 7 bytes for link-local communication and 12 to 28 bytes for global communication.

Mesh Addressing header when forwarding at layer two, a Hop Limit field when forwarding at layer three, and a full IPv6 address when communicating with devices outside the sensornet.

5.5 Summary

In this chapter, we presented an adaptation layer for communicating IPv6 datagrams using IEEE 802.15.4 frames. The adaptation layer supports three functions: (i) IPv6 header compression (LOWPAN_HC) to reduce header overhead, (ii) datagram fragmentation to support the IPv6 minimum MTU, and (iii) support for layer-two forwarding. The mechanisms are expressed using a header stack format. LOWPAN_HC compresses headers using both shared-context and stateful compression mechanisms. The former does not require any per-flow state and exploits redundant information across layers and common values to achieve compact headers. The latter provides additional compression, especially at the transport layer where correlation is much higher with a flow. The result is a compact header that can express a wide range of communication patterns and functionality. The header stack format was adopted by the IETF and published as a proposed standard in RFC 4944 [117]. LOWPAN_HC builds on the compression mechanism defined in RFC 4944 and currently forms the basis of improved header compression efforts within the 6LoWPAN working group.

Header compression applies to more than just packet headers - compression is also useful in reducing memory requirements for routing and forwarding, among others. Assuming a common global routing prefixes blurs the difference between the link and IPv6 addresses. Routing and forwarding need only operate

on the Interface Identifiers, which are derived directly from link addresses. However, we must first define a mechanism for autoconfiguring IPv6 addresses using common global routing prefixes and Interface IDs derived from link addresses. We tackle the autoconfiguration problem in the following chapter.

Chapter 6

Network Layer - Configuration and Management

The IPv6 network layer is composed of three components: (i) configuration and management, (ii) forwarding, and (iii) routing. In this chapter, we start with the first of three components as it is necessary for forming and maintaining an IPv6 network. IPv6's goals of supporting large numbers of nodes, unattended operation, and easy configuration and management are paramount to sensornets. But while IPv6 had the right concepts, the solutions defined in RFCs are not quite sufficient for sensornets. The solutions for IPv6 Neighbor Discovery and Autoconfiguration assume a single broadcast domain with few resource limitations. In this chapter, we show how to map these mechanisms to sensornets.

6.1 Configuring Large Numbers of Nodes

For nodes to join a network and communicate with other IP devices, they must be able to configure and maintain a number of parameters. Nodes must configure a link-local IPv6 address to communicate with neighboring nodes on the same IP link and a global unicast address with a global routing prefix to communicate with off-link nodes within the sensornet or other connected IP networks. All IP addresses must

be unique within their respective scopes. To communicate to arbitrary IP destinations, nodes must discover routers and configure a default route. An effective network must also be manageable, providing mechanisms that give information about the health of the network, as well as error messages when errors occur.

The IPv6 designers expect continued explosive growth in the number of IP hosts, both on private IP networks and on the public Internet. They envision that IPv6 will soon reach down into everyday things, such as common household appliances, many which do not have traditional computer displays or are necessarily operated by a person knowledgeable in IP networking. As a result, one of the primary goals was to support easy configuration and management of large numbers of devices. Along with expanding the IP address space from 32 to 128 bits, the Internet Control Message Protocol (ICMPv6) [21] and Dynamic Host Configuration Protocol (DHCPv6) [38] were developed to support effective configuration and management of large networks.

ICMPv6 provides three basic configuration and management mechanisms: *Neighbor Discovery* [121], *Stateless Address Autoconfiguration* [168], and *informational and error messages* [21]. Neighbor Discovery gives nodes the ability to discover other nodes attached to the same link, map addresses between the network and link layers, discover routers and configure default routes, as well as provide parameters for configuring hosts attached to the link. DHCPv6 supports centralized configuration and management for distributing configuration parameters to individual nodes. IPv6 addresses represents only one class of configuration parameters supported by Neighbor Discovery and DHCPv6. ICMPv6 informational message provide important information about the health of the network, while error messages provide necessary information about datagram delivery errors. Learning from IPv4, IPv6 folds many of the essential link-specific mechanisms into the IP framework (e.g. Address Resolution Protocol (ARP) [136] and Dynamic Host Configuration Protocol (DHCP) [36]), improves router discovery mechanisms, and adds new functionality such as neighbor discovery and address autoconfiguration.

The IPv6 design goals of supporting large numbers of nodes, unattended operation, and easy configuration and management are paramount to sensornets. The mechanisms developed within the IPv6 architecture far exceed anything that has been proposed specifically for sensornets. As a result, it seems natural

to utilize the mechanisms already defined for IPv6. However, in the context of sensornets, the solutions currently defined in RFCs are not quite sufficient for the following reasons:

1. **Single IP Link**: Neighbor Discovery assumes operation over a single IP link and is only defined for routers configuring hosts. In a multihop sensornet topology, effective network configuration requires network parameters must propagate over multiple hops.

2. **Single Broadcast Domain**: The solutions defined for IPv6 assume that the IP link provides a single broadcast domain, one of the traditional assumptions discussed in Chapter 3. Our sensornet architecture, however, equates link-local scope with radio range and does not support a single broadcast domain. Duplicate Address Detection, on-link prefixes, and ICMPv6 Redirect are a few mechanisms that assume a single broadcast domain.

3. **IPv6 Address Uniqueness**: IPv6 requires addresses to be unique within their respective scopes. However, this requirement is not sufficient to support a network composed of overlapping link-local scopes. For link-local addresses, a sensornet node's IPv6 address must be unique not only among its neighbors, but also among its two-hop neighbors. Furthermore, sensornets often have more dynamic topologies caused by the inherent variability in connectivity or node mobility. For this reason, it is desirable to maintain address uniqueness across the entire sensornet so that nodes do not need to continuously detect and repair address conflicts.

In this chapter, we show how to apply IPv6 Neighbor Discovery (IP6-ND) mechanisms in sensornets. We define a new neighbor discovery protocol (LP6-ND) designed specifically for ad-hoc networks with strict resource constraints. LP6-ND carries forward much of IP6-ND, but makes simple assumptions to reduce the generality and associated protocol overhead. For example, by assuming that Interface Identifiers are always derived from link-layer addresses, Neighbor Discovery is not needed for address resolution. LP6-ND also supports efficient dissemination of network parameters to support multihop sensornets.

This chapter also presents Address Autoconfiguration mechanisms designed for sensornets to maintain uniqueness of all IPv6 addresses (including link-local) across the entire sensornet. Our network architec-

ture for sensornets applies cross-layer optimizations, supporting SLAAC when link-layer mechanisms (e.g., PAN coordinator) maintain uniqueness of link addresses and DHCPv6 when these link-layer mechanisms cannot be assumed. SLAAC is attractive because it minimizes network-layer overhead when other link-layer address management mechanisms already exist. DHCPv6 assumes no such link-layer functionality and combines both link- and network-layer address management into a single mechanism, reducing overhead. Both SLAAC and DHCPv6 rely on centralized mechanisms, the only difference is that the former applies centralized techniques at the link layer while the latter applies them at the network layer. The advantage of a centralized approach is that it avoids the use of multicast communication and the expensive floods required to implement them. Both also support management of configuration parameters other than IPv6 addresses.

6.2 Neighbor Discovery

6.2.1 Background

IP6-ND provides necessary configuration mechanisms for nodes to communicate with neighboring nodes and, more generally, arbitrary IPv6 devices. As defined, Neighbor Discovery supports a wide range of functions - it allows nodes to discover routers attached to the link, discover on-link prefixes to help determine if a destination is a neighboring node, discover network parameters when configuring an IPv6 interface, statelessly autoconfigure addresses when attaching to a link, determine uniqueness of an address through Duplicate Address Detection, resolve address mappings between the link and network layers, maintain reachability information to neighboring nodes, and redirect traffic by informing nodes of a better first-hop route. IP6-ND folds in many of the mechanisms that were necessary yet link-specific with IPv4, including Address Resolution Protocol (ARP) [136] and Dynamic Host Configuration Protocol [36].

All communication for IP6-ND occurs within link-local scope. All of the services are supported through a five message types: *Router Advertisements*, *Router Solicitations*, *Neighbor Advertisements*, *Neighbor Solicitations*, and *Redirect*. Routers multicast Router Advertisements to announce their presence, on-link prefixes, and network configuration parameters. Router Advertisements include prefix information and

indicate whether or not to use stateless or DHCP address autoconfiguration. Hosts may multicast Router Solicitations to request Router Advertisements from neighboring routers. Hosts use Neighbor Solicitations to resolve mappings between link- and network-layer addresses, determine if an IPv6 address is already in use by another node on the link, or to determine reachability to a neighboring node. Neighbor Advertisements are used to respond to Neighbor Solicitations or announce a link-layer address change. Finally, nodes can communicate Redirect messages to inform a node of a better next-hop route. Neighbor Discovery often requires address-free communication and relies extensively on link-local multicast addresses when sending messages.

IP6-ND is only defined for host configuration and is not intended for router configuration. Neighbor Discovery assumes that network parameters and on-link prefixes apply only to a given link and that routers are configured with the necessary parameters either manually or through other mechanisms, such as IPv6 Prefix Delegation [171]. However, unlike traditional IP links, ad-hoc wireless networks may be connected over multiple hops with overlapping link-local scopes. The embedded nature of sensornet nodes mean that sensornet border routers are the only feasible entry point for configuration of sensornets. As a result, network configuration parameters must propagate over multiple hops or overlapping link-local scopes. This is a significant departure from IP6-ND, as we wish to configure both hosts *and* routers.

sensornets add to the challenge, by requiring operation under strict resource constraints. As defined, IP6-ND can have significant overhead, requiring continuous transmissions to propagate network parameters, resolve address mappings, and maintain reachability information to neighboring nodes. ARP caches can help reduce the cost of address resolution, but do so at the cost of increased memory. However, simple assumptions can significantly reduce transmission overhead. By always deriving Interface Identifiers from link-layer addresses, Neighbor Solicitation and Advertisements are no longer needed to support address resolution.

In the following section, we show how to map IP6-ND to low-power ad-hoc networks and create LP6-ND, a neighbor discovery protocol designed for sensornets. With LP6-ND, the only messages that nodes must transmit periodically are Router Advertisement messages. Nodes may transmit Router Solicitations to request immediate transmission of Router Advertisements from neighboring routers. LP6-ND also

provides Neighbor Solicitations and Advertisements to support neighbor reachability and neighbor discovery. Compared to IP6-ND, LP6-ND only provides the basic mechanisms for use by other components. With simplifying assumptions, we can reduce IP6-ND to a minimal form, supporting the strict resource constraints of sensornets.

6.2.2 Discovering Routers

In this section, we show how LP6-ND discovers routers and propagates network parameters over multiple hops in an efficient manner. Like IP6-ND, LP6-ND utilizes Router Advertisement messages to announce the presence of routers and network configuration information. Neighboring nodes listen for Router Advertisements to discover routers, configure default routes, and retrieve any configuration parameters. We draw on the existing message formats defined by IP6-ND but compact them to better support small link MTUs typical to sensornets.

Router Advertisements

The Router Advertisement message in LP6-ND is almost identical to that defined in IPv6-ND, as shown in Figure 6.1. Router Advertisements carry *Current Hop Limit* to indicate the default hop limit value to use for outgoing messages, a *Managed Address Configuration* flag to indicate if DHCPv6 should be used for address autoconfiguration, a *Other Configuration* flag to indicate if DHCPv6 should be used to obtain other parameters, and *Router Lifetime* field that indicates the lifetime of the advertising router. LP6-ND differs in that Router Advertisements do not carry the Reachable Time and Retrans Timer fields used for Neighbor Unreachability Detection (NUD), where ND is responsible for maintaining reachability information for neighboring nodes.

The exclusion of maintaining neighbor reachability information within LP6-ND represents our desire to only provide the minimal capabilities required for upper layers to communicate effectively. In a sensornet setting, we argue that it is better for protocols to maintain their own reachability information if they need it. A general component that maintains reachability information for all neighbors is infeasible for

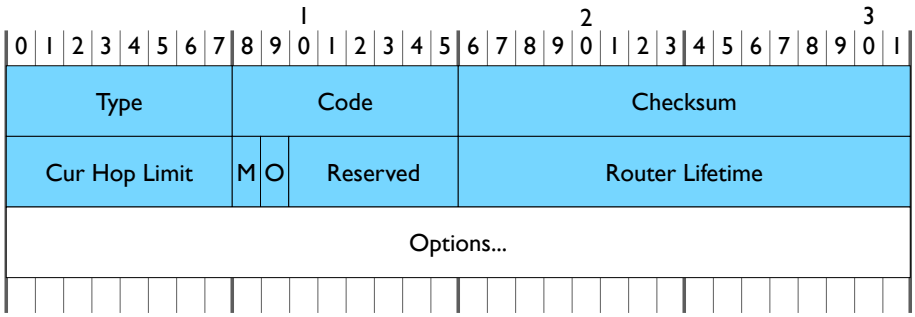


Figure 6.1: **Router Advertisement Format.** IPv6 routers transmit Router Advertisement messages to announce their presence and communicate configuration parameters. The LP6-ND format differs from IP6-ND only in removing fields related to Neighbor Unreachability Detection (NUD). LP6-ND does not provide a general mechanism for maintaining reachability information to all neighboring nodes, as the communication and memory cost make it infeasible in sensornets.

sensornets because of the associated communication and memory costs. Protocols designed for sensornets often maintain reachability for a small subset of neighbors. For example, routing protocols often utilize a subset of links to form a routing topology. Application-level bindings with neighboring nodes are often small and static. Furthermore, the concept of reachability may differ between protocols especially since sensornet links can have a wide range of link qualities. We show a concrete example of how a routing protocol maintains its own reachability information in Chapter 8.

In a single-hop deployment, all nodes are within range of a sensornet border router and LP6-ND's Router Advertisement message is sufficient to configure all hosts within range. However, for multihop deployments, information contained in the Router Advertisement as well as any options must be communicated over multiple hops to all nodes in the sensornet. In the following section, we discuss how to LP6-ND extends Router Advertisements to support multihop deployments.

Multihop Information Option

Using Router Advertisements, LP6-ND implements a dissemination mechanism to communicate information from sensornet border routers to all nodes in the sensornet. Nodes periodically transmit Router Advertisements that include an option with a sequence number to indicate the freshness of the information. When propagating new network parameters, the border router increments the sequence number. Nodes always

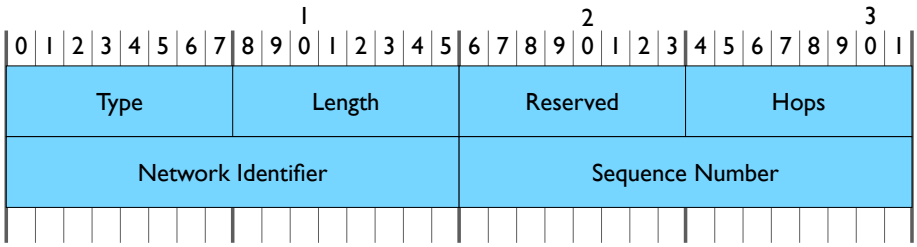


Figure 6.2: **Multihop Information Option Format.** Routers include this option in Router Advertisement messages to support parameter dissemination over multiple IP hops. While traditional Neighbor Discovery only operates over a single IP link, sensornets operate as a collection of overlapping link-local domains. Router Advertisements are sent using the Trickle algorithm [103] to minimize transmission overhead.

accept the latest information for configuring their interfaces. Routers then rebroadcast the information to propagate it further. LP6-ND uses the Trickle protocol to manage Router Advertisement transmissions [103]. Trickle has emerged as the de-facto mechanism for disseminating information, as it is extremely simple and operates with purely local rules. Trickle's communication overhead scales well with network density, supports quick propagation when new values are introduced, low overhead in the steady state, and requires small and constant state.

Trickle manages the transmission period using an adaptive timer, and the transmission rate represents a tradeoff. Sending at a faster rate allows quicker router discovery and parameter propagation, but consumes more energy and channel utilization. Sending at a slower rate uses less resources, but suffers from slower propagation times. Trickle bounds the transmission period τ_c by $[\tau_{min}, \tau_{max}]$. With each transmission, τ_c doubles up to τ_{max}. A sequence number is used to indicate an information change. When a received sequence number differs, τ_c is reset back to τ_{min}. Nodes always record and advertise the highest Sequence Number value. Signals from other components (e.g., router or upper layers) may also reset τ_c to τ_{min}, and we utilize this feature in Chapter 8. To reduce transmission overhead, Trickle randomly selects a transmission time in the range $[\frac{1}{2}\tau_c, \tau_c]$ and suppresses transmissions if k or more messages are received with the same sequence number before transmission. Suppression allows the protocol to scale well with density by reducing the number of redundant transmissions in dense networks.

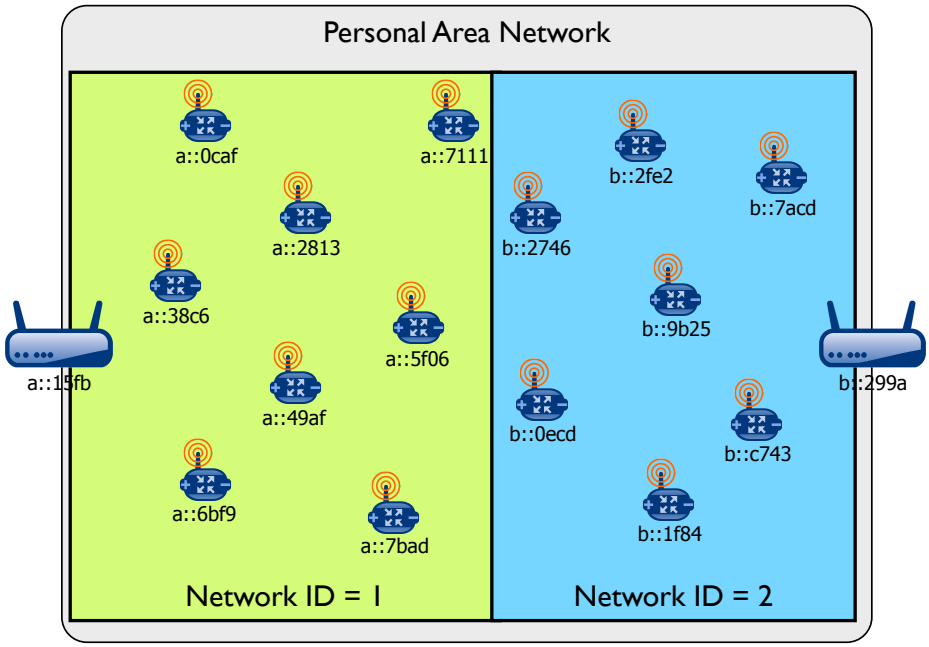


Figure 6.3: **Different Global Routing Prefixes in the Same PAN.** Border routers advertising different Network Identifiers can be used to split the network using different global routing prefixes. The Network Identifier represents a connected set of sensornet nodes and one or more border routers, allowing the adaptation layer to exploit shared context when communicating among them.

To support Trickle, we define a Router Advertisement option: the *Multihop Information Option*. As shown in Figure 6.2, the Multihop Information Option includes a *Sequence Number* for use with Trickle. A *Network Identifier* is used to identify the source of the information. The Network Identifier ensures that a node only accepts information from one source and does not propagate that information to nodes configured from a different source. Not only is the Network Identifier required for correctness in the presence of multiple networks, but it also allows a network administrator to segment a sensornet using more than one border router with different network parameters, as shown in Figure 6.3. The Multihop Information Option also includes a *Hops* field to indicate the number of hops from the nearest border router that is seeding the network parameters.

Figure 6.4: **Time Information Option Format.** Routers include this option in Router Advertisements to establish a network-wide time base. In many sensornet applications, time synchronization is a required service. Piggybacking other configuration parameters on Router Advertisements, such as time synchronization, significantly reduces transmission overhead for communicating the information.

Time Information Option

Because Router Advertisements are designed to disseminate configuration information, we use them to communicate any additional information useful to sensornets. *Time Synchronization* is one such service that can piggyback information on existing Router Advertisements. A network-wide time base may be used to coordinate sensor readings or actions between nodes in a network. Coordination may be required due to application requirements, such as the environmental phenomena being sensed, or to reduce energy consumption of system-level operations, such as radio communication. Global time is also useful for recording the time of occurrence for various actions. Furthermore, sensornet hardware platforms typically do not include a real-time clock and must obtain global time information from the network.

To support time synchronization, we define a Time Information Option for Router Advertisements. The option represents time using a representation similar to POSIX time and the option format is shown in Figure 6.4. The option represents seconds using a 4-octet field and milliseconds using a 2-octet field. The option fields must be populated as close to the time of transmission at the physical layer, to minimize any uncertainty due to resource arbitration. Refinements may be done over time to correct for clock skew and drift.

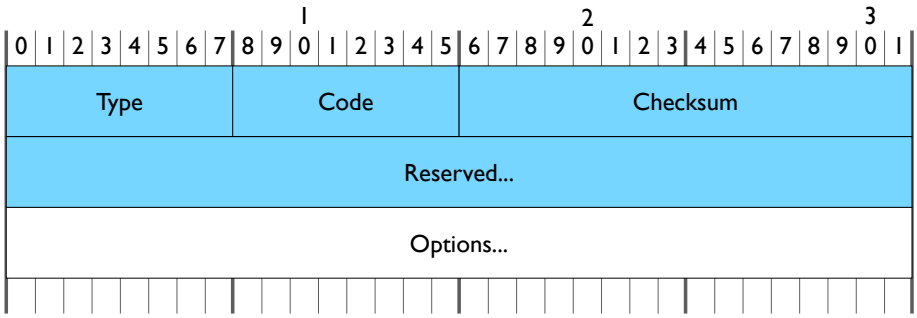


Figure 6.5: **Router Solicitation Format.** Hosts transmit this message to request Router Advertisement transmissions from neighboring routers. Routers receiving a solicitation react by resetting the Trickle timer to the minimum period.

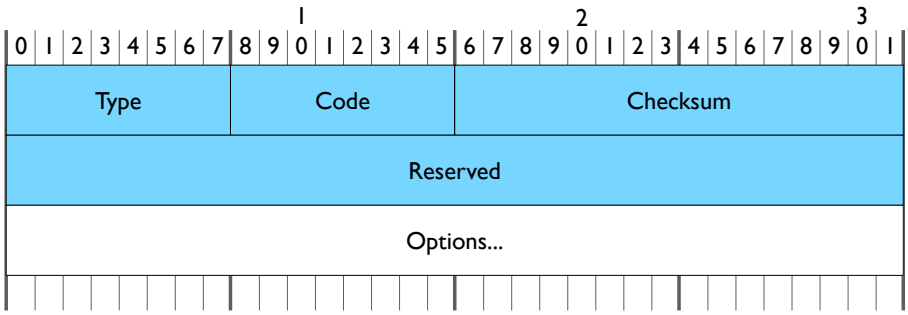
Router Solicitation

Nodes may send Router Solicitations to request neighboring routers to immediately generate and transmit Router Advertisements. The message format in LP6-ND is identical to IP6-ND, as shown in Figure 6.5. Routers receiving a Router Solicitation simply reset the transmission period τ_c to τ_{min}. However, Router Solicitations do not cause immediate generation of Router Advertisements, as there is little opportunity for the density-aware suppression mechanisms to succeed.

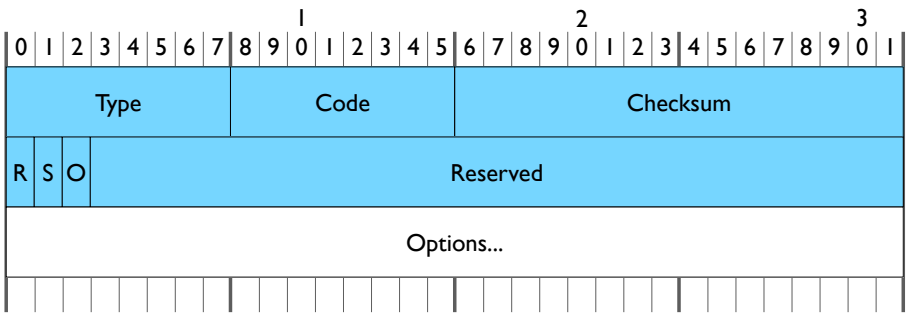
6.2.3 Discovering Neighbors

LP6-ND uses Neighbor Solicitation and Advertisement messages to support Neighbor Unreachability Detection and neighbor discovery. Unlike IP6-ND, we do not need to use LP6-ND for address resolution because we derive Interface Identifiers from IEEE 802.15.4 addresses. LP6-ND also does not support Duplicate Address Detection, expecting that they will be implemented by other components (e.g., DHCPv6). As a result, the message formats for LP6-ND only differ in that they do not need to carry a Target Address, as shown in Figure 6.6.

Nodes may use Neighbor Solicitation and Advertisement messages to determine reachability of neighboring nodes. However, unlike IP6-ND, LP6-ND does not maintain its own reachability information for neighboring nodes and is reflected by the removal of Neighbor Unreachability information in the Router Advertisement message. Due to strict resource constraints, sensornet protocols typically maintain reachability



(a) Neighbor Solicitation



(b) Neighbor Advertisement

Figure 6.6: **Neighbor Solicitation and Advertisement Format.** Nodes transmit Neighbor Solicitation and Advertisement messages to discover neighbors and verify network-layer reachability. Nodes do not need to determine link-layer addresses because our architecture establishes a direct mapping between link addresses and Interface Identifiers.

information for a small subset of neighbors. As a result, LP6-ND only provides the mechanisms for nodes to verify reachability, allowing components to issue Neighbor Solicitations as needed. This is an example where we have chosen to remove policy from within the network layer and only provide the basic capabilities for use by other components.

LP6-ND also extends Neighbor Solicitation and Advertisements to allow hosts to discover other neighboring hosts. The lack of Redirect and on-link prefix determination makes it difficult for nodes to optimize communication to nodes that are only one hop away. As a result, nodes may optionally send Neighbor Advertisements to announce their presence. Nodes may also multicast Neighbor Solicitations to request Neighbor Advertisements from neighboring hosts. Optimizing routes between nodes separated by a single hop can reduce communication costs across the network. Single hop communication is an important common

case, especially due to the physical nature of sensornet applications, where nearby nodes are more likely to communicate with each other. We discuss this routing optimization further in Chapter 8.

6.2.4 Discussion

LP6-ND carries forward necessary functionality defined by IP6-ND, including router discovery and network parameter discovery [121]. While the IP6-ND specification [121] is not quite right for the needs of sensornets, we showed how to extend it to support the ad-hoc wireless networks with strict resource constraints. Specifically, LP6-ND implements a dissemination protocol using Router Advertisements and the Trickle protocol to support parameter dissemination over multiple hops. Trickle is simple and scalable, and allows us to avoid many of the magic constants specified in existing standards [121]. We envision that Trickle can be applied to other IP protocols with similar benefits.

Periodic Router Advertisements by routers represent the only continuous transmission overhead caused by Neighbor Discovery. Router Advertisements are necessary to support router discovery and disseminate configuration parameters across the network. LP6-ND reduces protocol overhead by reducing the generality of functions provided by IP6-ND. Assuming that Interface Identifiers are always derived from link-layer addresses removes the need for explicit communication to determine link-layer addresses. LP6-ND also removes support for maintaining reachability to all neighboring nodes, as doing so can be infeasible under strict resource constraints. LP6-ND only provides the basic mechanisms to support Neighbor Unreachability Detection and discovery of neighboring nodes, expecting other components to implement their own usage and definition of reachability. Other features, such as on-link prefix discovery and Redirect, are not applicable to an ad-hoc wireless link that does not support a single broadcast domain.

6.3 Address Autoconfiguration

6.3.1 Background

To communicate with neighboring nodes on the same link, a node must first configure a link-local address. To communicate with nodes multiple hops away, the node must also configure a global address. With the goal of supporting easy configuration and management for large numbers of nodes, address autoconfiguration mechanisms were developed within the IPv6 framework. Stateless Address Autoconfiguration (SLAAC) allows nodes to form an IPv6 address by taking an IPv6 prefix and appending an Interface Identifier derived from the interface's link-layer address [168]. Routers announce IPv6 prefixes in Router Advertisements for use with SLAAC using a Prefix Information Option. SLAAC is simple and allows operation of both small and large networks without the need for a central server, but lacks any centralized visibility or management.

In addition to SLAAC, DHCPv6 was also developed to support centralized configuration and management of IPv6 addresses [38]. In IPv4, DHCP was designed to operate directly above the link, outside the IP framework. However, DHCPv6 was developed within the IPv6 architecture as a UDP protocol. All DHCPv6 communication occurs within link-local scope, and either a DHCPv6 server or relay agent must be resident on that link. Communication over link-local scope is necessary because nodes do not yet have global addresses. Relay agents form an application-level overlay, allowing nodes to utilize link-local communication to communicate with a server that may not be attached to the same link. Compared to SLAAC, DHCPv6 provides centralized visibility and management at the cost of additional communication overhead.

IPv6 nodes must verify uniqueness of the autoconfigured IPv6 address before permanently assigning it to the link. An IPv6 address remains *tentative* until its uniqueness has been verified. When the uniqueness check succeeds, the address is declared as *preferred* and assigned to the interface. If the uniqueness check fails, the IPv6 address cannot be assigned to the interface. In cases where the IPv6 address will be unique with high probability, nodes may declare an address as *optimistic* while its uniqueness is being

verified. Nodes may assign optimistic address to an interface, but its use is restricted as it may cause conflicts with other devices.

The existing procedure utilizes the Duplicate Address Detection (DAD) mechanism provided by IP6-ND to verify uniqueness of an address [121]. DAD assumes that the scope of uniqueness is limited to a single link that supports a single broadcast domain. With these assumptions, DAD is relatively simple, relying on link-local multicast to query if an address is already in use. However, different mechanisms must be used when the IP link does not provide a single broadcast domain. In an ad-hoc network, nodes must maintain uniqueness among its two-hop neighbors or larger scope so that conflicts do not occur when neighboring nodes attempt to use that node's address. A naive solution extends the basic DAD solution to support a multicast query to all nodes over multiple hops. However, doing so adds significantly to the communication overhead.

Adding to the challenge is the high variability in topology due to node mobility or changing connectivity characteristics. Topology changes are equivalent to nodes entering and leaving the two-hop neighbor set. If nodes only verify uniqueness among their two-hop neighbors, they must demote an address to tentative or optimistic and perform DAD any time new neighbors appear. Furthermore, nodes must also continuously monitor the topology to determine when the neighbor set changes. Instead, it is more desirable to establish uniqueness across the entire network so that nodes only perform DAD when they first join the network. Existing work on DAD for ad-hoc networks either rely on multicast communication over multiple hops or unicast communication with a central server. The multicast approach aligns well with the goals of SLAAC, as it operates without a server. However, multicast in sensornets requires expensive floods where communication costs grow polynomially with network size. Centralized techniques either use DHCPv6 directly or similar techniques to ensure IPv6 address uniqueness within an address scope and can operate using only unicast communication.

In the remainder of this section, we present address autoconfiguration mechanisms for sensornet. We define a new scope called *sensornet scope* that includes all nodes within the sensornet, as shown in Figure 6.7. We require that all IPv6 addresses must be unique within that scope. By verifying uniqueness of all

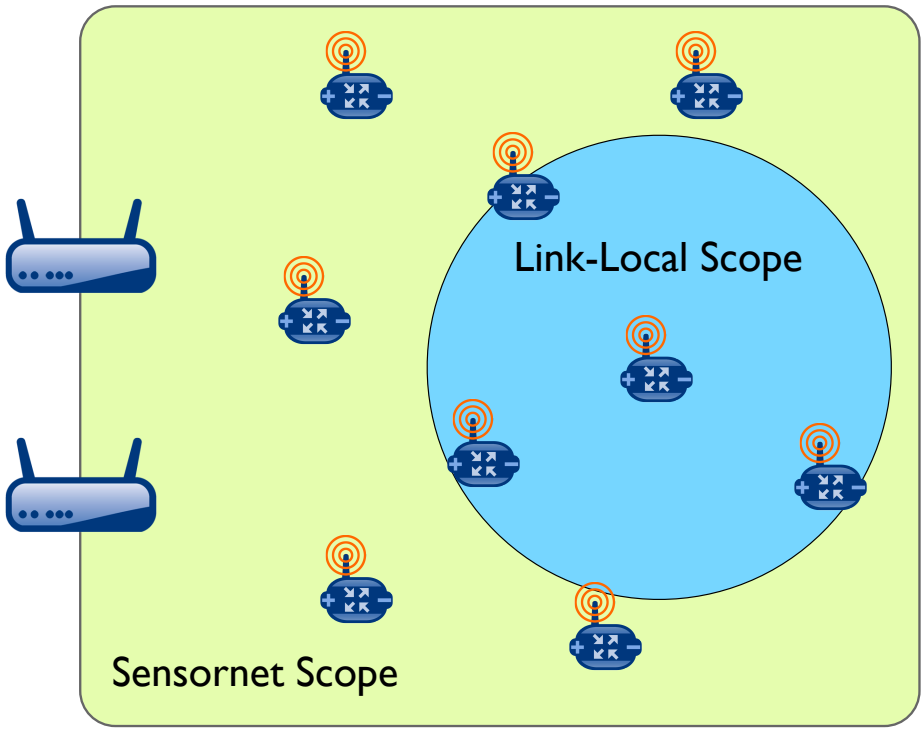


Figure 6.7: **Sensornet Scope.** Sensornet scope is well-defined and consists of all nodes within the sensornet. All IPv6 addresses, including link-local addresses, must be unique within the entire sensornet so that nodes do not need to verify uniqueness every time the topology changes.

IPv6 addresses (including link-local addresses) across the sensornet when first joining, nodes do not need to perform DAD every time other nodes join the network or the topology changes.

We show the operation of both SLAAC and DHCPv6 within sensornets, both applying cross-layer optimizations to minimize resource requirements. SLAAC relies on link-layer mechanisms (e.g., PAN coordinator) to maintain uniqueness of link addresses and avoid the use of DAD at the network layer, allowing IPv6 addresses to be immediately declared as preferred. DHCPv6 does not assume such link-layer mechanisms, maintaining uniqueness of IPv6 addresses on its own. Nodes still configure a link-local address using SLAAC, but declare the address as optimistic. Using DHCPv6, the node verifies uniqueness of the link-local address and requests other global addresses to assign to its interface. DHCPv6 may also be used to assign short link addresses, removing any need for link-layer mechanisms to manage link addresses.

Both SLAAC and DHCPv6 rely on centralized mechanisms, the only difference is that the former applies centralized techniques at the link layer (e.g., PAN coordinator) while the latter applies them at the application layer (e.g., DHCPv6 server). The advantage of a centralized approach is that it avoids the use of multicast communication and the expensive floods required to implement them.

6.3.2 Stateless

SLAAC is *stateless* because it allows nodes to configure their own addresses and there is no central entity that maintains a list of IPv6 addresses in use within the network. SLAAC allows nodes to configure IPv6 addresses by concatenating an IPv6 prefix with an Interface Identifier that follows the Modified EUI-64 format [168]. The Modified EUI-64 format only differs from the IEEE EUI-64 format by flipping the universal/local bit. The prefix is either well-known (e.g., the link-local prefix fe80::/10) or provided in Router Advertisement messages. Interface Identifiers are usually generated from link addresses. With an IEEE 802.15.4 interface, the Interface Identifier may be derived from the extended IEEE EUI-64 address by flipping the universal/local bit. An Interface Identifier may be derived a short address by setting the lower 16 bits to the short address and the universal/local bit to indicate local scope.

Nodes obtain global routing prefixes from one or more Prefix Information Options contained in Router Advertisements. The option format is unchanged from the existing specification [121] and shown in Figure 6.8. The Prefix Information Option contains the *Prefix Length*, *Valid Lifetime* of the prefix, *Preferred Lifetime* of the prefix, and the *Prefix*. Routers use the Prefix Information Option to advertise both on-link prefixes and prefixes for use with SLAAC. The *L* and *A* flags indicate whether the prefix is for on-link or SLAAC, respectively. In our sensornet architecture, however, no prefix other than the link-local prefix may be used as as on-link prefix. On-link prefixes assume a single broadcast domain with transitive reachability. To support header compression, we define a third flag *C* that indicates whether or not the prefix represents the common prefix used for shared-context compression.

According to the specification, nodes must first declare all addresses as *tentative* until uniqueness is verified within scope using Duplicate Address Detection (DAD). Traditionally, DAD makes the joining node

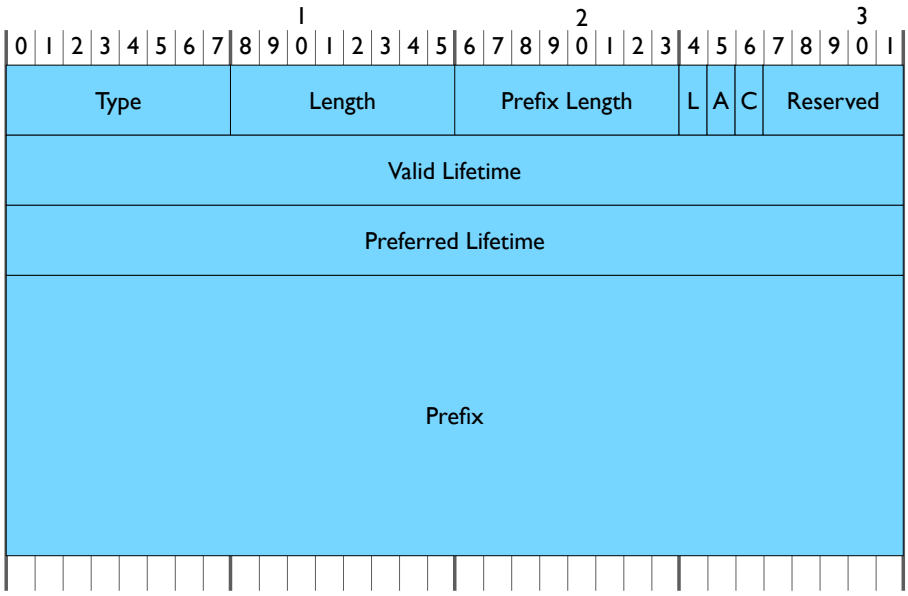


Figure 6.8: **Prefix Information Option.** Routers advertise global routing prefixes using the Prefix Information Option carried within a Router Advertisement. Unlike traditional IP networks, we do not permit this option to specify on-link prefixes because the overlapping link-local domains in a sensornet does not provide a single broadcast domain.

responsible for establishing uniqueness so that existing nodes do not have to continuously perform DAD every time a new node joins. To support this same behavior, we define a new scope called *sensornet scope* that includes all nodes within the sensornet. Establishing uniqueness of all addresses, including link-local addresses, within sensornet scope allows nodes to maintain their addresses even when new nodes join the network or the topology changes due to node mobility or changing connectivity.

DAD traditionally relies on using multicast to implement a completely distributed protocol without the need for a central server. However, multicast communication over multiple hops is costly in resource-constrained sensornets. In some network configurations, other mechanisms may be in use to maintain uniqueness of link addresses. IEEE 802.15.4, for example, specifies the use of a PAN coordinator to assign short addresses within the PAN. As a result, we allow nodes to declare IPv6 addresses as preferred if uniqueness of the Interface Identifier is already established by other mechanisms. By applying this cross-layer optimization, the network-layer overhead of configuring IPv6 addresses is minimal and only requires nodes to receive a Router Advertisement that contains prefix information to configure global addresses, as shown in Figure 6.9.

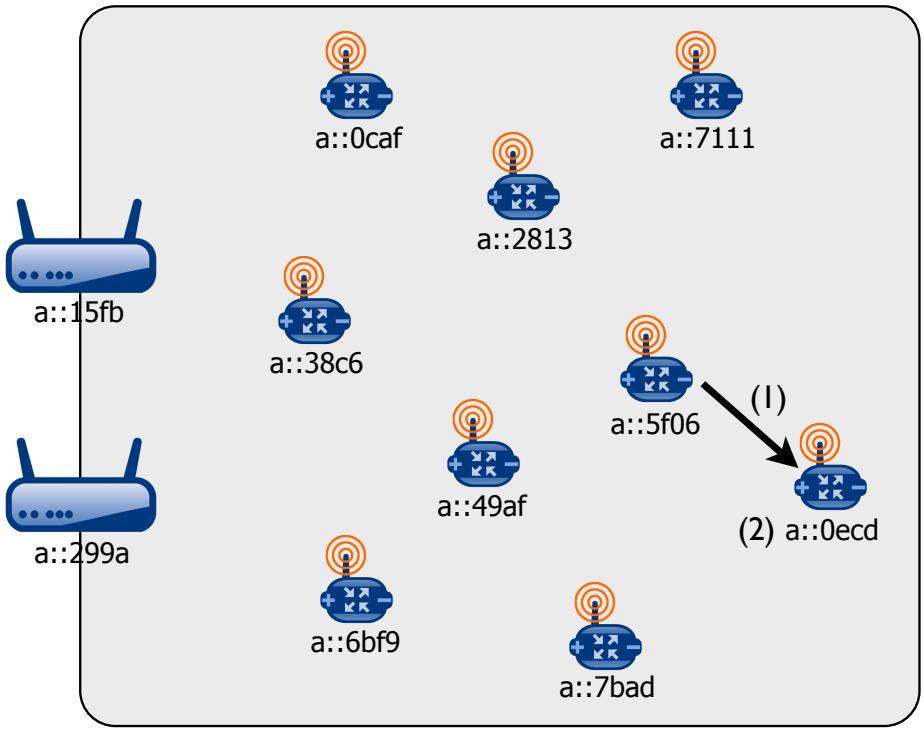


Figure 6.9: **Stateless IPv6 Address Autoconfiguration of Global Addresses.** Using SLAAC, nodes configure IPv6 addresses by taking an IPv6 prefix and appending an Interface Identifier derived from link-layer addresses. Global routing prefixes may be distributed using a Prefix Information Option carried in a Router Advertisement. SLAAC has little overhead: (1) clients receive prefix information contained in a router advertisement and (2) use the prefix to assign an IPv6 address. However, our use of SLAAC requires the link layer to maintain uniqueness of link addresses.

In cases where there are no link layer mechanisms to establish uniqueness of the Interface Identifier, we show how we can use DHCPv6 to maintain uniqueness of IPv6 addresses within a sensornet. While DHCPv6 represents a stateful approach, it allows nodes to avoid multicast communication when establishing uniqueness. We show how we can use DHCPv6 in sensornets. Note that existing link-layer mechanisms to maintain uniqueness of link addresses also represent a stateful approach similar to DHCPv6.

6.3.3 Stateful (DHCPv6)

DHCPv6 represents a stateful approach to address autoconfiguration [38]. Nodes request IPv6 addresses from a central DHCPv6 server, which then decides how to respond and with what IPv6 address. The DHCPv6 protocol can only assume communication over link-local scope, as a node may not yet be

configured with a global address. As a result, DHCPv6 assumes that there is a DHCPv6 *agent* attached to the link, which may be either a *server* or a *relay agent*. According to the existing specification, clients first send a Solicit message to the link-local, all-DHCP-agents address to discover DHCP agents. DHCP agents announce their presence using Advertisement messages. Clients then choose one agent and send a DHCP request to it and wait for a reply.

Relay agents form an application-level overlay that allows requesting nodes to communicate over link-local scope while querying a server that is multiple IP hops away. Relay agents forwards DHCPv6 messages between a client and a server on different IP networks. A relay agent forwards requests by encapsulating them in a DHCPv6 Relay-Forward message, which includes a *Link Address* used to identify the IP network and a *Peer Address* that holds the source IPv6 address of the client. The server then copies the Link Address and Peer Address to a Relay-Reply message, so that the server can send the reply to the client through the Relay Agent.

Because the DHCPv6 protocol does not assume a single broadcast domain, the protocol can run unmodified in a sensornet. In our sensornet network architecture, all routers also host DHCPv6 relay agents. Relay agents on sensornet nodes always forward DHCPv6 messages to the nearest border router using the subnet router anycast address. Border routers either host DHCPv6 servers directly or DHCPv6 relay agents. As a result, sensornet border routers serve as the only configuration points for the location of a DHCPv6 server.

When joining a network, nodes first autoconfigure a link-local address using SLAAC with the IEEE EUI-64 link address. Because nodes use DHCPv6 to also determine the uniqueness of an IPv6 address, nodes cannot use a preferred link-local address to communicate with a DHCPv6 agent. Instead, because IEEE EUI-64 addresses are intended to be globally unique, we allow them to be declared as optimistic [118]. Nodes can use optimistic addresses before their uniqueness has been established as long as they are unique with high probability. Note, however, that we do not allow global IPv6 addresses to be declared as optimistic, as they can pollute routing tables before uniqueness is established.

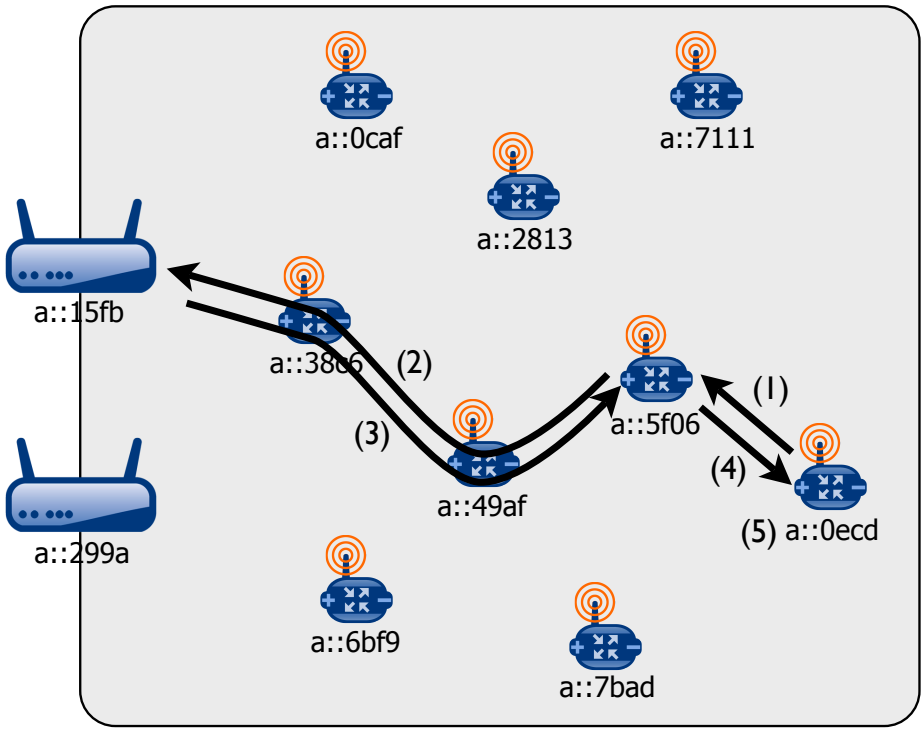


Figure 6.10: **DHCPv6 Address Assignment in Sensornets.** All sensornet routers host DHCPv6 relay agents, relaying DHCPv6 requests to a border router using the subnet-router anycast address. An example process is as follows: (1) the client sends a request to a relay agent, (2) the relay agent forwards the message to the border router, (3) the border router communicates a reply back to the relay agent, (4) the relay agent communicates the message to the requesting client, and (5) the client assigns the provided IPv6 address to its interface. The border router may host a server or a relay agent that forwards DHCPv6 messages on towards a server.

To reduce the overhead of DHCPv6, we rely on Router Advertisements to announce the presence of DHCPv6 agents. Doing so eliminates the need for a Solicit/Advertise exchange. Also, we can eliminate the overhead of a Relay-Forward header within the sensornet by realizing that the Peer Address may be derived from the DHCP Unique Identifier and the Link Address may be derived from the IPv6 header. However, sensornet border routers must decompress the Relay-Forward header before processing the message to maintain compatibility with existing DHCPv6 implementations. Figure 6.10 shows DHCPv6 operation within a sensornet.

6.3.4 Discussion

SLAAC and DHCPv6 represent two different methods for autoconfiguring interfaces within a sensornet with global unicast IPv6 addresses. Both SLAAC and DHCPv6 apply cross-layer optimizations to minimize total overhead. SLAAC relies on existing link-layer mechanisms to minimize network-layer overhead. DHCPv6, however, makes no assumptions of the mechanisms provided by the link and supports operation with networks that do not provide those optimizations. Note that in both cases, a central server is used to maintain uniqueness of addresses across the entire a sensornet.

SLAAC is attractive in sensornets because of its minimal network-layer overhead when existing link-layer mechanisms maintain uniqueness of link-layer addresses. Receiving a single Router Advertisement with a Prefix Information Option is enough to configure a global unicast address. The communication remains within link-local scope, reduces resource requirements on the rest of the network, and reduces the configuration latency. Renumbering a network also has relatively low overhead, only requiring dissemination of a new global routing prefix through Router Advertisement messages.

DHCPv6 provides centralized configuration and management of the network at the network layer. Unlike SLAAC, DHCPv6 does not assume that the uniqueness of link-layer addresses is maintained by the link layer. In addition to assigning IPv6 addresses, DHCPv6 may be used to manage assignment of short link addresses. Using DHCPv6 to manage both link- and network-layer addresses reduces resource requirements because it avoids duplicating mechanisms between layers. Effective use of a PAN coordinator assumes link-layer configuration and management, forwarding, and routing - all mechanisms provided by the IPv6 network layer. Using a DHCPv6-only solution, fewer messages must be exchanged and message parsing all occurs within the IPv6 framework.

Both stateless and stateful methods are useful for configuring parameters other than IPv6 addresses. Stateless configuration of other parameters can utilize Router Advertisements to disseminate parameters across the sensornet. For example, a DNS Configuration option may be used to configure hosts with DNS servers [89]. DHCPv6 may also be used to distribute arbitrary parameters using additional DHCPv6 options. In existing IP networks, DHCPv6 servers are used to provide hosts with DNS server addresses [37] or other

routers with network prefixes using Prefix Delegation [171]. Centralized configuration and management also makes customized replies based on IEEE EUI-64 link addresses simple.

6.4 ICMP Error & Informational Messages

In addition to providing a framework for network-layer configuration and management protocols, ICMPv6 provides general information about the health of the network through ICMP error and informational messages. For example, ICMPv6 echo messages can be used to test the availability and communication latency of communicating with an IP host and is utilized by the well-known *ping* utility. ICMPv6 error messages are sent whenever a node cannot properly deliver a datagram to the intended destination, such as an unknown routes or an unopened port. *Traceroute* exploits errors generated by IP nodes to discover the path used to reach a destination.

ICMPv6 is an invaluable resource for the network deployment and management, especially sensor-nets where low-power wireless communication can lead to unpredictable behaviors. ICMPv6 echo messages can help determine network connectivity issues. For example, having a node flash an LED whenever it receives and responds to an ICMP Echo Request can help determine if communication to the destination is working properly and, if so, whether communication back to the source is also working. Relying on Destination Unreachable errors can help determine the path towards the destination in the same manner that traceroute does. Destination Unreachable errors may also be helpful in closing transport- or application-layer sessions early. With knowledge that messages are not reaching the destination properly, end-to-end retransmissions can be stopped early to reduce energy and channel consumption.

ICMPv6 errors carry the error-causing IPv6 datagram in the payload so that the source can better determine the cause of the error. Our IPv6 header compression is easily applied to both the ICMPv6 datagram and the error-causing datagram. To generate the ICMPv6 error, the error-causing IPv6 datagram can be copied into the payload in its compressed form. One optimization stems from the fact that the source address of the error-causing datagram is the same as the destination address of the ICMPv6 error. As a result, we

can fully compress the IPv6 source address of the error-causing datagram and set the appropriate bits in the LOWPAN_HC encoding to indicate that the address is completely elided. The decompressor expands the elided address in a similar fashion, by deriving it from the encapsulating header.

6.5 Related Work

ICMPv6, Neighbor Discovery, and Address Autoconfiguration far exceed mechanisms that have been developed within the sensornet community. However, the existing mechanisms are not quite sufficient for sensornets because they were designed to operate over a single broadcast domain. In 2004, the IETF formed the Ad-Hoc Network Autoconfiguration (AUTOCONF) working group to develop mechanisms for autoconfiguring link-local and global IPv6 addresses. But much of the work to date has been focused on address autoconfiguration, rather than on broader aspects of ICMPv6 such as Neighbor Discovery.

Active autoconfiguration protocols (sometimes called Strong DAD) rely on floods to verify uniqueness of an IP address. In the most basic form, a node randomly picks an address and floods the entire network to verify uniqueness [132]. Nodes already using the address reply to indicate duplication. Flooding the network is attractive because it does not require the use of a central server. However, floods are expensive especially on resource constrained devices. Furthermore, nodes must constantly check for conflicts to support network merging and partitioning. One approach is to limit floods by creating a hierarchy, where multiple subnets are created using elected leader nodes [54, 180]. Leader nodes only need to verify uniqueness with other leader nodes, while other nodes only need to maintain uniqueness within a single leader scope.

Passive autoconfiguration protocols (sometimes called Weak DAD) detect duplicate addresses by maintaining extra state along with the IP address. One approach is to have routers maintain a Virtual IP Address, which is simply the concatenation of the IP address plus a unique key identifying the node [174]. Using routing information in combination with the key, routing nodes can determine if addresses are duplicated. A later approach defines ways to detect duplicate addresses without the need to disseminate additional keys or information [179]. By observing inconsistencies or unexpected behavior in route updates, routers can

determine duplicate addresses with high probability. Passive autoconfiguration protocols are attractive in that they require minimal message overhead, especially compared to active protocols. However passive protocols depend on routing protocols and in some cases the specific properties of a routing protocol.

Both active and passive autoconfiguration first generate addresses then detect whether a duplicate occurs. Another approach to autoconfiguration is to allocate addresses such that duplicates do not occur [115, 116, 164]. These approaches generally extend DHCP by allowing ad-hoc routers to act as distributed DHCP servers. By assigning DHCP prefixes, address pools are distributed as nodes join. Nodes respond to requests by splitting their address pool and delegating half to the requesting node. Address allocation approaches are attractive in that they naturally support recurring merge and partitions. However, where address allocation fails is when address pools must be reclaimed due to nodes leaving the network indefinitely. As a result, nodes must flood the network to reclaim assigned prefixes or detect duplicates.

The address autoconfiguration mechanisms we developed in this chapter applies cross-layer optimizations to reduce the cost of configuring both link- and network-layer addresses. When link-layer mechanisms maintain uniqueness of link addresses, SLAAC may be used to minimize network-layer overhead in configuring IPv6 addresses. However, when no underlying mechanisms can be assumed, we rely on DHCPv6 to configure IPv6 and link addresses. Using a centralized server allows nodes to utilize only unicast communication to configure IPv6 addresses. Because DHCPv6 does not assume a single broadcast domain, we can utilize the protocol as specified. However, we do make some optimizations to reduce communication overhead.

6.6 Summary

In this chapter, we focused on essential services for forming and maintaining an IPv6 network. We showed how to adapt IPv6 Neighbor Discovery (used to discover routers, neighboring nodes, and configuration parameters) to support sensornets. By defining new Neighbor Discovery options, we were able to leverage the existing standard and modify it to support a network of nodes connected by overlapping link-

local domains. We also presented address autoconfiguration methods necessary to configure both link-local and global IPv6 addresses so that nodes can communicate with neighboring IPv6 hosts and arbitrary IPv6 devices connected to different IP networks. We leverage existing standards, including SLAAC and DHCPv6 but apply cross-layer optimizations to make them feasible in sensornets. Finally, we discussed the utility of ICMP error and informational messages within sensornets, where wireless communication can lead to communication problems that are hard to diagnose.

With ICMPv6, we now have the ability to form and maintain an IPv6 network in a sensornet setting. The mechanisms are in place to discover neighbors and potential default routes, configure link-local IPv6 addresses to communicate with neighboring nodes and global IPv6 addresses to communicate with arbitrary IPv6 nodes multiple hops away. In the following chapters, we discuss remaining components of the network layer, including how nodes forward datagrams in an energy-efficient way and how nodes choose default routes as well as route to arbitrary sensornet nodes. Note that these services are required to support DHCPv6, where relay agents may need to forward requests to border routers hops away.

Chapter 7

Network Layer - Forwarding

In this chapter, we address the second component of our IPv6-based network layer: forwarding. We present the design of an energy-efficient forwarder for sensornets. The IP network layer must provide high enough "best-effort" datagram delivery to allow effective end-to-end communication and reliable transport. IP does not specify the mechanisms used for forwarding, allowing us to choose ones necessary for sensornets. Our forwarder design is primarily concerned with energy efficient forwarding and utilizes techniques developed within the sensornet community, including hop-by-hop retransmissions and congestion control, streaming, and QoS. The forwarder also supports efficient multicast forwarding using Trickle.

7.1 Energy-Efficient Datagram Forwarding

The IP architecture elegantly separates *forwarding* from *routing*. The forwarder is responsible for receiving datagrams from an interface, performing next hop lookups in a forwarding table, and submitting the message for transmission on the appropriate interface. The router is responsible for managing entries in the forwarding table. In contrast to much sensornet work, which typically integrates forwarding and routing together, our IPv6-based architecture for sensornets maintains the separation, as shown in Figure 7.1.

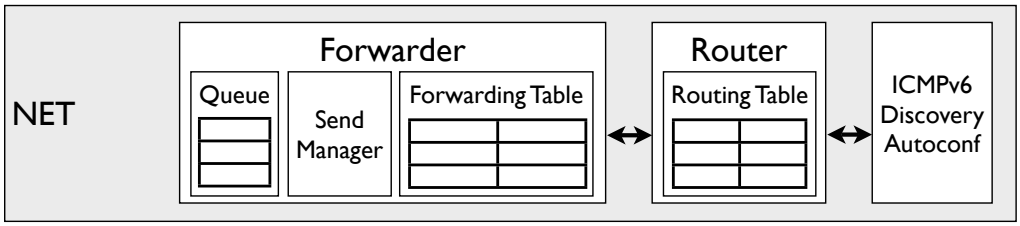


Figure 7.1: **Separation of Forwarding and Routing.** IP separates forwarding from routing. Received datagrams not destined for the node are placed in a forwarding queue. The forwarder processes messages in the forwarding queue by performing a forwarding table lookup to determine the next-hop destination and submits the datagram to the link layer for transmission. The router is only responsible for maintaining forwarding table entries. The forwarding table serves as a narrow interface between the forwarder and router.

The IP network layer must provide high enough best-effort datagram delivery to allow end-to-end mechanisms to achieve effective communication and reliable transport. But IP does not take a position on the use of specific mechanisms or the software architecture used to implement those mechanisms. Forwarding mechanisms can have a significant influence on overall performance and is well studied in both wired and wireless contexts. The rapid growth of the Internet, significant advancements in link technologies, and increase in data streaming have led past efforts to focus on maximizing overall throughput and fairness for many independent flows. Naive forwarding policies that don't respond to congestion at the next-hop destination can lead to higher queue occupancies on average and more frequent queue overflows. The higher communication latency and loss rates directly translate into lower throughput, especially when coupled with end-to-end congestion control mechanisms such as TCP. Mechanisms that randomly drop messages to keep average queue sizes small [55, 69, 193], provide feedback directly to the source [144], perform hop-by-hop flow-control mechanisms [114, 125, 130], and increase the reliability and visibility of lossy, wireless links [7] have all been proposed primarily to increase throughput and fairness.

Unlike traditional IP networks, typical sensornet applications have vastly different traffic characteristics and requirements. Typical sensornet application workloads have lower aggregate throughput and channel utilization than is often assumed in traditional IP networks. As a result, any congestion that may occur in sensornets are more likely to be temporary occurrences, rather than the common case. Many sensornet applications, for example, report one to few datagrams with periods on the order of minutes to hours. Some

sensornet applications may require infrequent transfer of bulk data, but rarely from multiple nodes simultaneously. In many cases, sensornet applications may be tolerant of communication delay, but may require low-latency communication for rare but time-sensitive events or alarms.

Rather than throughput and fairness, we are more concerned about achieving high end-to-end success rates with high *energy-efficiency*. To no surprise, energy-efficiency is a metric not typically considered for traditional IP networks where energy is plentiful. However, efforts within the sensornet community have also not addressed energy-efficient forwarding well, even when not constrained by the IP framework. While existing work has focused on transmission efficiency, this metric only addresses the number of message transmissions rather than the energy cost of both transmitting and receiving. With duty-cycled links, the energy cost of transmitting can vary by an order of magnitude or more depending on what optimizations the link-layer can apply. Forwarding multiple messages to the same next-hop destination consecutively are more likely to take advantage of scheduling and streaming optimizations within the link layer. Broadcast transmissions often have higher energy cost than unicast transmissions, because it requires all neighboring nodes to wakeup using a full wakeup signal or network-wide synchronization. Snooping on transmissions from neighboring nodes significantly increases receive costs and may be prohibited by some link protocols due to per-neighbor state requirements. Many existing efforts in sensornets rely on both broadcast and snooping primitives to communicate hop-by-hop feedback [45, 81, 96, 146, 178, 185].

The different traffic characteristics and requirements for sensornets cause us to rethink the forwarding mechanisms used to achieve energy-efficient delivery. In this chapter, we present forwarding mechanisms that have *energy-efficiency* as a primary goal. Energy-efficiency also implies high success rates in datagram delivery, as any dropped messages represent wasted resources. Energy-efficient unicast forwarding is supported by three mechanisms: (i) *hop-by-hop recovery*, (ii) *streaming*, and (iii) *congestion control*. Because these mechanisms often trade increased energy efficiency for increased latency, the forwarder also applies simple *QoS* mechanisms that allow upper layers (i.e., transport) to select the forwarding policy. Energy-efficient multicast forwarding is supported by the Trickle algorithm. All of the mechanisms we discuss in this chapter are simple and do not require any per-flow state.

The mechanisms used to implement IP forwarding are mostly independent of IP. Our architecture defines link-local scope as those nodes reachable within radio communication range, requiring routing protocols to operate at the network layer. However, forwarding can occur at the network-layer (as done with traditional IP forwarding) or at the link-layer (as done with MPLS). As discussed in Chapter 3, the link- and network-layer forwarding approaches have their respective tradeoffs. The forwarding mechanisms we develop in this chapter are not specific to layer that forwarding occurs.

7.2 Unicast Forwarding

7.2.1 Background

The IP architecture traditionally places reliable delivery and flow control in the transport layer, placing such functionality solely at the end points. The Transmission Control Protocol (TCP) was the such mechanism to become mainstream and remains widely used today [141]. TCP provides a reliable data stream by implementing end-to-end retransmissions, where end points retransmit data that is not explicitly acknowledged. Furthermore, TCP infers state about the network by observing communication throughput, delay, and dropped messages to perform flow control [85]. TCP attempts to maximize throughput by operating as close to the congestion point as possible.

TCP's end-to-end property elegantly keeps functionality at the end points and leaves the core network simple, but lacks complete visibility of the network. TCP assumes that congestion is the cause of packet drops, cannot distinguish between queuing delays and communication delays, and cannot determine the extent of the congestion. Furthermore, communication delays limit how quickly TCP can react to changing conditions. Interesting interactions from Tail Drop forwarders caused network-wide synchronization when reacting to congestion, reducing overall throughput. These challenges limited the maximum throughput TCP could achieve, as well as the fairness between flows.

Extra assistance from the network itself was necessary to provide greater visibility into the network and improve overall network performance. One class of efforts assisted end-to-end rate controllers by plac-

ing congestion detection and notification mechanisms within the network. Some provided early notification by randomly dropping packets before congestion occurs, implicitly notifying end points that congestion is near [69, 85, 193]. Others provide explicit notification by piggybacking information on data packets [94, 145] or sending explicit messages back to the source [144]. Some efforts decouple the detection from notification [55].

Another class of efforts involves *hop-by-hop control*, placing rate controllers within the network in addition to congestion detection and notification [114, 125, 130]. Hop-by-hop mechanisms are capable of detecting, providing feedback, and reacting to conditions directly where they occur. As a result, hop-by-hop mechanisms are more capable of reacting quickly and appropriately to local conditions. Hop-by-hop mechanisms naturally provide back-pressure all the way to the source, but only does so as congestion in the network builds. In contrast, end-to-end mechanisms suffer from larger communication latencies, lack of visibility of the cause or location of congestion when it occurs, and the inability to distinguish between queuing delay and propagation delay. Routing changes, for example, may cause significant changes in the communication properties along a path, and end-to-end mechanisms may require significant time to detect and react to the change. Hop-by-hop mechanisms, however, could immediately detect and react to the route change where it occurred.

In the wireless space, hop-by-hop mechanisms are widely used to forwarding messages. The advent of wireless links, such as WiFi and cellular, invalidated TCP's assumption that congestion is the primary cause for message drops. Initial wide-spread deployment of wireless links were mostly infrastructure-based, where only the first and last hops were likely to be wireless. Even so, link-layer recovery and cross-layer optimizations that expose the cause of message drops to the transport layer have been shown to significantly improve performance [7].

Hop-by-hop mechanisms have also been shown to be beneficial in multihop wireless networks. High spatial and temporal variability, a shared wireless medium, and limited communication resources are best served by hop-by-hop mechanisms that can quickly react to local conditions [45, 81, 96, 146, 178, 185]. In addition, hop-by-hop recovery is used to increase end-to-end reliability [158, 177]. Wireless links

inherently have higher loss rates, and end-to-end loss rates increase polynomially with the number of radio hops. Relying solely on end-to-end retransmission would imply that recovery costs would also increase polynomially with the number of hops. In contrast, the cost of hop-by-hop recovery only increases linearly with the number of hops.

Interestingly, few mechanisms were proposed to directly address energy efficiency. Some efforts attempt to address energy efficiency by evaluating transmission efficiency, noting the relationship between wasted transmissions and wasted energy. However, even fewer efforts so far has evaluated their design on duty-cycled link protocols. Duty-cycled links can greatly impact protocols that rely on snooping or broadcast to propagate feedback information, both of which can be expensive in duty-cycled operation. Link transmission costs also depend on what optimizations the link can apply. For example, submitting messages as a batch may allow the link layer to amortize startup costs across multiple transmissions.

In the following sections, we present a set of forwarding mechanisms that seek to minimize energy consumption of forwarding datagrams along a path, including *hop-by-hop recovery*, *streaming*, and *congestion control*. These mechanisms typically reduce energy consumption at the cost of communication delay, so we also address simple *Quality of Service (QoS)* mechanisms that allow upper layers to specify the policy.

7.2.2 Hop-by-Hop Recovery

The forwarder implements *hop-by-hop recovery* to increase end-to-end delivery rates when delivering datagrams. Improved end-to-end delivery rates increases overall energy efficiency of delivering datagrams. Messages that fail to reach the destination represent wasted energy, as some energy has been spent in unsuccessfully delivering the datagram. Improved end-to-end delivery rates also reduces the frequency of triggering recovery mechanisms used with end-to-end reliable transport. Compared to end-to-end retransmissions, hop-by-hop retransmissions only occur where they are needed. Hop-by-hop retransmissions only burden the node and neighborhood where transmission failures occur, adapting quickly to local conditions. End-to-end retransmissions, however, burden all nodes on the path from the problem area to the source, increasing recovery latency and overall energy costs.

Retransmissions

To reduce packet drops, the forwarder defaults to a custody-transfer approach, only dequeuing messages after the next-hop destination acknowledges reception by the final destination or placement in the forwarding queue. The custody-transfer approach attempts to increase energy efficiency by reducing packet drops, but also induces larger queue occupancies and communication latencies. Traditional best-effort forwarders often drop messages in attempt to keep queue occupancies small, limit the scope of traffic build-up that may affect other nearby flows, and provide implicit congestion signals to avoid congestion. Traditional forwarders also attempt to increase throughput and fairness for many independent flows. Instead, our primary goal in sensornets, however, is to maximize energy efficiency and end-to-end delivery rates, rather than high throughput and fairness. Dropping messages to keep queues flowing wastes the resources already consumed in partially delivering those messages.

The forwarder implements hop-by-hop custody-transfer by piggybacking acknowledgment information on link-layer acknowledgments, adding a single bit to indicate whether or not the packet was successfully received. While the forwarder must process the received packet before the link-layer acknowledgment can be sent, the added processing delay is negligible compared to the relatively long time-scales of IEEE 802.15.4 communication. Furthermore, we've modified the link-layer acknowledgments in Chapter 4 to include addressing information, making them less dependent on strict timing requirements. To ensure that unforeseen errors do not prevent successful packet delivery, the forwarder also dequeues messages after the number of retransmissions for a packet exceeds a threshold.

Traditional approaches that strictly adhere to the layering do not include upper-layer information in link acknowledgments. Specifically, basic link acknowledgments only indicate transmission failures, but provide no information of whether upper layers were able to accept or process the packet. Memory constraints common to sensornets limit forwarding queue sizes, increasing the likelihood of queue overflows. While we could disable link-layer acknowledgments when queues are full, the sender cannot distinguish between transmission failure or congestion at the receiver. Later, we discuss how including such upper-layer information can also be used for congestion control as well.

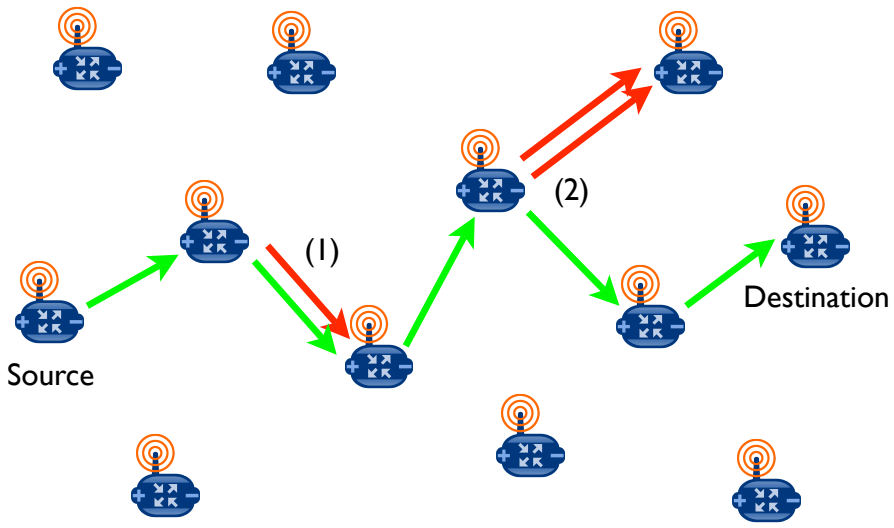


Figure 7.2: **Hop-by-Hop Recovery.** Hop-by-hop recovery seeks to increase energy-efficiency by increasing end-to-end success rates. (1) The forwarder implements hop-by-hop retransmissions using a custody-transfer approach to reduce packet drops. (2) By performing a next-hop lookup for each retransmission, the forwarder allows the router to update the forwarding table between retransmissions and take advantage of receiver diversity.

Re-Routing Support

The forwarder performs a next-hop lookup in the forwarding table on *every* retransmission. Doing so allows the network layer to utilize receiver diversity, where the router can modify forwarding table entries when transmission attempts fail. Failed transmissions reduce the link quality estimate provided by the link layer, and the router may react by updating the forwarding table. We provide a complete discussion of the router's actions in Chapter 8. More traditional approaches often perform a single next-hop lookup and submit the datagram to the appropriate interface, assuming that link qualities are relatively stable. With wireless communication, however, link qualities are highly variable and have high temporal locality [157]. A transmission failure indicates that subsequent transmission to the same destination in the near future are also likely to fail. Both environmental effects and node mobility can instantly change the link quality. When such link failures occur, re-routing increases the likelihood that the forwarder can continue to make forward progress, lowering communication latency while improving end-to-end delivery rates.

Re-routing can duplicate datagrams while forwarding, when retransmissions to different next-hop destinations are successfully received but unsuccessfully acknowledged. Duplicate suppression is simple

when retransmissions occur to the same destination, by caching link-layer sequence numbers received with each packet. However, retransmissions to different next-hop destinations can lead to duplicates that are not detected by the link. To help suppress such duplicates, the source node marks each generated packet with a unique sequence number. The sequence number may be contained within an IPv6 Hop-by-Hop Option header or in a 6LoWPAN adaptation header depending on what layer the forwarding is implemented. Using the source address and sequence number, the forwarder can detect and suppress duplicate messages already buffered in the forwarding queue. Suppression only occurs for enqueued messages - simply caching metadata for recently forwarded packets could lead to message drops due to transient routing loops.

7.2.3 Streaming

The forwarder also implements *streaming* to reduce the average energy cost of transmitting messages. When multiple enqueued packets have the same next-hop destination, the forwarder transmits them back-to-back. Doing so allows the link layer to apply streaming optimizations we presented in Chapter 4. The forwarder notifies the link layer of pending transmissions for the same destination through the Media Management Control interface. The link layer's streaming optimizations reduce transmission overhead of subsequent transmissions, increase communication throughput, and increase energy efficiency. Streaming also improves energy efficiency by exploiting temporal properties of link quality. While re-routing seeks to avoid repeated transmission failures, streaming takes advantage of the higher probability of success immediately after a successful transmission.

To increase streaming opportunities, the forwarder intentionally delays forwarding those messages that are marked as delay-tolerant. The forwarder buffers delay-tolerant messages until either queue occupancy or time spent in the forwarding queue exceeds a threshold. The source marks messages as delay-tolerant using either an IPv6 Hop-by-Hop Option header or a 6LoWPAN adaptation header, depending on which layer the forwarder operates. While holding on to delay-tolerant messages induces higher queue occupancies, the link's streaming optimizations allow the forwarder to quickly transfer multiple packets as a single unit. The

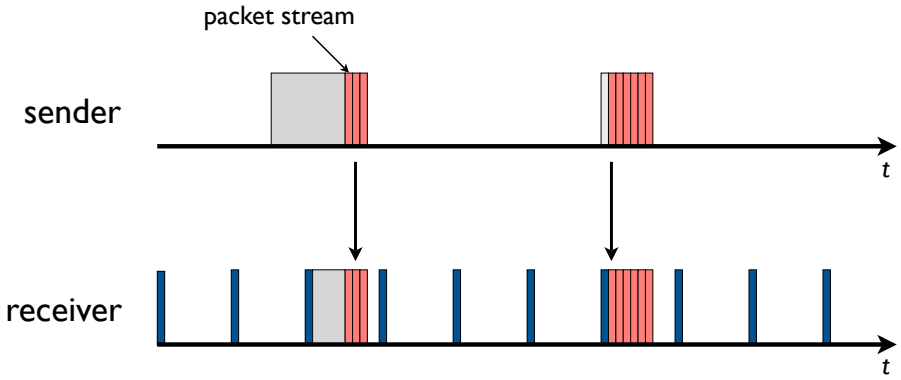


Figure 7.3: **Streaming.** Streaming packets destined for the same next-hop node increases energy efficiency by reducing transmission overhead and exploiting temporal locality typical to wireless links.

forwarder is then free to receive a streamed transfer from neighboring nodes. This bursty transfer approach allows the network to fully utilize streaming and increase energy efficiency.

7.2.4 Congestion Control

Transmission failures translate directly into inefficient resource utilization, consuming both energy and channel capacity. Transmission failures may be due to a number of reasons, such as topology changes due to node mobility, transient interference, or simply the lossy nature of the wireless link. Congestion forms another significant cause for transmission failures. One form of congestion involves high forwarding queue occupancies, causing transmission failures because the next-hop destination is unable to accept the message. Another form of congestion involves contention on the wireless medium, where competing transmissions can collide and prevent the intended receivers from decoding the signal. Unlike other causes of transmission failures, forwarding policies can significantly affect the frequency and magnitude of congestion.

The forwarder detects congestion in two ways. First, the forwarder detects congestion when it fails to receive acknowledgments from the next-hop destination. In addition to collisions caused by congestion, failure to receive acknowledgments may simply be due to the lossy nature of wireless links. In both cases, however, the forwarder should reduce the transmission rate in response. Reducing the transmission rate reduces contention and the likelihood of collisions. Slowing the transmission rate also respects the temporal

properties of wireless link qualities, where a transmission failure indicates a higher probability for transmission failures in the near future.

Second, the forwarder detects congestion by monitoring queue occupancies, indicating congestion only when the queue is full. In contrast, traditional approaches indicate congestion before the queue actually becomes full in attempt to keep queue sizes small and avoid queue overflows [55, 94, 145]. Our forwarder differs because it seeks to maximize energy efficiency, rather than overall throughput and latency. By allowing full use of the queue, the forwarder can stream more packets at a time, maximizing the benefits of streaming. Because hop-by-hop recovery already indicates when the forwarding queue is full, the forwarder does not require additional signals to indicate congestion. Furthermore, the forwarder's custody-transfer approach helps ensure fewer packet drops when queues are full.

The forwarder reacts to congestion by controlling the transmission rate to next-hop destinations, using an additive-increase, multiplicative decrease (AIMD) controller. Because energy efficiency is the primary goal, the AIMD controller remains fairly conservative at the cost of reduced throughput. The specific controller parameters depend on the kind of congestion detection. While indication of a full queue provides conclusive evidence of congestion, failure to receive link-layer acknowledgments does not. Failure to receive a single acknowledgment may be due to short-term interference. As a result, the rate decrease is smaller for lost link-layer acknowledgments compared to queue overflows. An important detail is that streaming always transmits packets back-to-back as quickly as possible. The rate controller only determines how often the node initiates streams.

Because hop-by-hop recovery implements a custody-transfer approach, hop-by-hop congestion control will appropriately apply back-pressure all the way to the application source as queues fill. One problem often associated with hop-by-hop back-pressure is that a single congestion point can influence surrounding nodes, affecting other independent flows that are nearby but do not go directly through the congestion point. The concern is reduced in sensornet settings because they often have more structured communication where nodes communicate to a few destinations rather than large numbers of independent flows. This is in contrast to more traditional IP networks where we often assume arbitrary point-to-point communication.

7.2.5 Quality of Service

In many cases, the forwarder conserves energy at the cost of communication latency. The custody-transfer approach induces larger queue occupancy by not allowing the forwarder to dequeue packets until they are accepted by a next-hop node. Streaming also induces increased queue occupancy by delaying packets in the hopes of receiving additional packets to stream. Conservative congestion control policies also reduce the transmission rate to help increase energy efficiency. Increased queue occupancies lead to larger queue servicing times. In the worst case, full queues do not allow the forwarder to receive new messages and prevent messages from making forward progress.

There are times, however, when communication latency is more important than energy efficiency. Some sensornet applications may require low-latency communication to report critical alarms or events. Network control and management traffic are often limited by the time-scales of end-to-end communication. Control protocols that provide information about the state and health of the network may need to quickly alert the traffic source about delivery errors. When increased throughput is desired, reliable transport protocols such as TCP benefit from low-latency communication. Management protocols used with interactive interfaces are generally more useful when operating a human time scales.

One of the main points of supporting IP is to provide interoperability with existing IP devices. To maintain compatibility with existing protocols, the forwarder attempts to forward all datagrams with low communication latency (at the cost of energy efficiency) unless they are explicitly marked as *latency-tolerant*. Without being flagged as latency-tolerant, the forwarder does not delay forwarding datagrams, makes congestion control mechanisms less conservative, and enqueues them ahead of any latency-tolerant messages in the queue. In general, however, we expect many sensornet applications to mark the majority of datagrams a latency-tolerant and enable better use of energy-efficiency optimizations. Either the source or border router can mark ingress datagrams as latency-tolerant using an IPv6 Hop-by-Hop Option or 6LoWPAN adaptation header.

Full queues can have the most significant impact on communication latency because they prevent messages from making forward progress. The forwarder mitigates the effects of full queues using two dif-

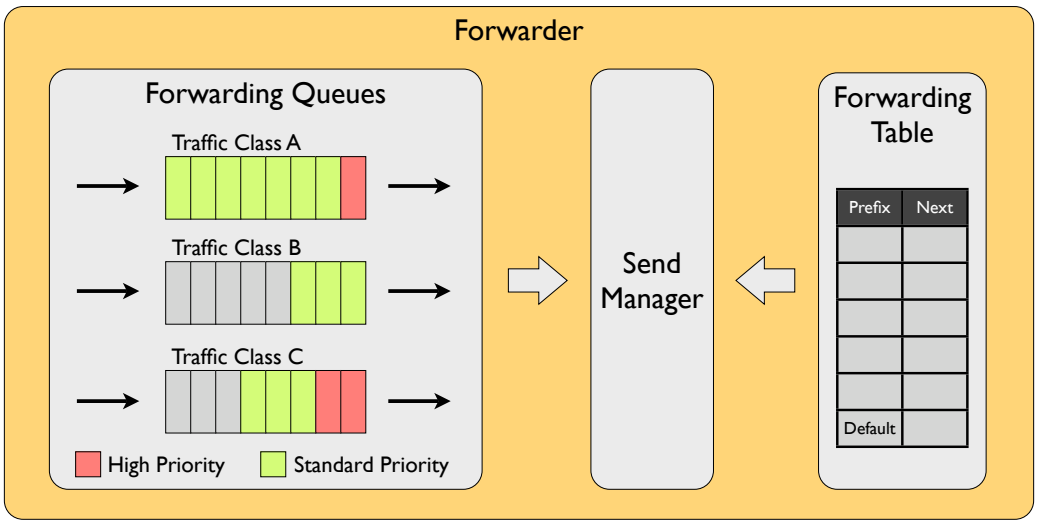


Figure 7.4: **Mitigating Full Queues.** The forwarder supports different priority levels, where high-priority traffic allows the forwarder to evict messages when the queue is full. The forwarder also supports different traffic classes, providing separate queues for each traffic class. Both cases allow messages to make forward progress even when low priority traffic or other classes are experiencing congestion.

ferent mechanisms. First, nodes may tag individual datagrams as *high priority*. When the forwarder receives a high priority message and queues are full, the forwarder evicts messages not marked as high priority from the queue. However, high priority messages never evict other high priority messages. The priority marking is completely orthogonal to the latency-tolerant marking, the former only indicates what messages may be evicted while the latter indicates if extra effort is requested to forward packets.

The second mechanism to mitigate the effects of full queues is *queue reservations*. The forwarder supports different traffic classes by allocating a separate queue for each class. With logically separate queues, the forwarder can continue to accept messages for a class even if others are congested without evicting other messages in the forwarding queue. Within each traffic class, the forwarder also reserves at least one entry for locally generated traffic, preventing starvation of local traffic when forwarding traffic keeps the forwarding queue full. Compared to the priority marking, queue reservations represent a pessimistic approach to mitigating the effects of full queues.

Different traffic classes are useful in supporting different communication patterns. For example, one traffic class may be used for egress traffic towards the border router while another traffic class may be

used for ingress traffic away from the border router. Separate queues allow communication in one direction to continue, even if the opposite direction is congested. This capability can be critical in telling the sources to reduce their traffic rates or propagating application-level alarms. Finally, another traffic class may be used for link-local traffic, allowing nodes to always communicate with neighboring nodes even when there is congestion in forwarding other traffic. We develop such a routing architecture that makes effective use of these traffic classes in Chapter 8. Limiting traffic flows to ingress, egress, and link-local supports the needs of typical sensornet application workloads and also constrains the amount of routing that the router must maintain.

While limited memory constrains queue sizes for all traffic classes, the queue sizes may be adjusted dynamically to support changes in traffic characteristics. For example, a typical sensornet application workload may require a steady stream of data traffic towards a border router, where allocating a large queue to egress traffic serves the application best. However, occasional firmware upgrades may require forwarders to temporarily allocate additional queue space to ingress traffic. Dynamic allocation may also be used to support increased fairness. For example, the number of entries reserved for locally generated traffic can be varied relative to other traffic to provide fair throughput allocation under congestion. However, we leave a more complete treatment of dynamic allocation adjustments to future work.

7.2.6 Related Work

Our work build upon a significant body of existing work that focuses on delivering messages, with high end-to-end success rates, throughput, and fairness. We bring together concepts of hop-by-hop recovery and congestion control, streaming, and QoS from both IP and sensornet communities into a single forwarding architecture. The primary difference is that we are concerned with energy-efficient forwarding, rather than maximizing throughput or fairness. Unlike most existing approaches, we explicitly addressed operation on duty-cycled links, where different kinds of transmissions can have vastly different energy costs and snooping is not generally supported.

The notion of introducing extra logic at each hop is heavily influenced by work in hop-by-hop rate control for wired networks, which showed that distributing the control logic hop-by-hop allows the protocol to react quicker and more appropriately at the site of congestion [114, 125, 130]. As a result, fewer packets are dropped and the network operates more efficiently. The primary goal of wired protocols is to maximize throughput and fairness, which often results in careful attention to direct interactions with TCP. Instead, our forwarder's congestion control seeks to maximize energy efficiency and does not assume TCP as the dominant transport protocol.

Many efforts in rate control for sensornets rely on hop-by-hop mechanisms. Hop-by-hop control is especially attractive in sensornets because it does not require end-to-end communication, which consumes precious resources even in areas that are not experiencing congestion. Woo et. al. showed that a simple hop-by-hop control is effective in improving fairness and energy efficiency in multihop wireless networks, by monitoring packet drops and using AIMD rate control [185]. CODA proposed continuous monitoring of channel activity to determine congestion [178]. However, continuous channel sampling is expensive because it requires the radio's receiver to be active. Fusion proposed the simultaneous use of mechanisms across different layers for more effective congestion control, including hop-by-hop flow control, a prioritized MAC, and rate limiting source traffic [81]. While our forwarder also combines similar mechanisms, Fusion relies on snooping of all messages to efficiently communicate the congestion bit. Our forwarder does not assume a snooping capability, and piggy-backs congestion information on link-layer acknowledgments instead. IFRC is another hop-by-hop rate control mechanism for sensornets that continuously shares congestion information based on average queue lengths with all potential interfering nodes, rather than only applying explicit back-pressure to nodes the routed path [146]. However, to efficiently share congestion information, IFRC also relies on nodes to snoop on their neighbors' transmissions.

Flush is a reliable transport protocol that utilizes hop-by-hop rate control to minimize interference and message drops [96]. Flush is unique in that it attempts to estimate forward path interference by utilizing physical properties (e.g. RSSI) when snooping on messages. But Flush, designed for bulk data collection,

assumes only a single flow at a time and attempts to maximize it's throughput. Like IFRC, Flush uses snooping, which may not be available in duty-cycled links.

While congestion control also aims to reduce dropped messages, explicit recovery mechanisms are used when messages are dropped due to transmission errors. Most recovery mechanisms in wireless networks also rely on hop-by-hop mechanisms. The Snoop protocol demonstrated how link-layer recovery for IEEE 802.11 can dramatically improve TCP performance, because local recovery often reacts faster than end-to-end mechanisms and better decouples message drops due to transmission errors from congestion [7]. Snoop also demonstrated the benefits of cross-layer optimizations, by sharing information between the link and transport layers to further improve performance. However, Snoop is only concerned with an infrastructure-based deployment, where the wireless link is a single radio hop.

Hop-by-hop reliability mechanisms are widely used in proposed forwarding mechanisms for sensornets. PSFQ is a reliable transport protocol that relies on hop-by-hop recovery [177]. PSFQ observed that end-to-end mechanisms are inappropriate for sensornets, where relatively high transmission errors can make the polynomial growth in recovery costs prohibitive. RMST makes a similar observation but argues that hop-by-hop recovery should occur at both link and transport layers [158]. While our forwarder implements hop-by-hop retransmissions, it also emphasizes the use of hop-by-hop retransmissions to utilize receiver diversity by allowing the router to modify forwarding table entries.

The use of streaming is inspired by the work with B-MAC, which showed that a low-power link can significantly reduce average delay and transmission cost when sending messages back-to-back [137]. The benefits of streaming were further demonstrated with network-level experiments when validating the SP design [138]. However, the streaming concept had yet to be integrated into a complete network architecture.

7.3 Multicast Forwarding

7.3.1 Background

IPv6 multicast provides a powerful primitive - hosts can simply send to a multicast group without maintaining state about members within the group. Instead, routers are responsible for maintaining state about what nodes are subscribed to the multicast group and delivering multicast datagrams addressed to them. Multicast addresses follow the IPv6 scoped address architecture and each address contains a scope identifier in addition to the group identifier [30]. Nodes subscribe to specific group identifiers and the scope indicates topological range from the sending node. Link-local scope only reaches those nodes directly attached to the link and is commonly used for discovery and autoconfiguration. Larger scopes reach nodes attached to other networks and their boundaries are administratively defined.

Typical sensornet applications benefit from a multicast primitive. Dissemination is useful for sensornet applications that require group operations, such as multiple light controllers in a building automation application. Often operating under a single administrative domain, dissemination protocols are often used to distribute configuration parameters to efficiently deliver configuration parameters to all nodes in a sensornet [169]. Dissemination protocols have also proved useful for address-free communication, useful for network management when nodes do not obtain proper network-layer addresses or when application-level naming is desired.

Link-local multicast is simple to implement, using broadcast or multicast capabilities at the link layer and network-layer address matching. Global multicast, however, requires forwarding messages over multiple hops to every member in the multicast group. Doing so efficiently is difficult, and scalability problems have prevented wide-spread use of IP multicast. Routers must learn and maintain routes towards hosts subscribed to multicast groups, creating a multicast distribution tree [29, 176]. Source-specific multicast builds a tree for each source, requiring forwarding based on both multicast and source address.

Ad-hoc wireless networks create additional challenges for IP multicast. Traditional methods for creating multicast distribution tables have significant state requirements, whereas sensornet nodes often have

constrained memory. Multicast distribution trees work best when the underlying topology is relatively static, a characteristic not typical to wireless links. Node mobility and time-varying link qualities often make maintaining multicast distribution trees costly and difficult.

Instead, efforts to support IP multicast in ad-hoc networks resort to simple flooding mechanisms [108]. In its basic form, all nodes within scope receive all multicast datagrams. A forwarding node that receives a multicast datagram simply rebroadcasts the message to forward it along and present the datagram to upper layers if subscribed to the multicast group. Forwarding nodes do not forward duplicate messages, made possible by a sequence number in the datagram. The beauty of flooding is that forwarders maintain little state - only a cache of sequence numbers for duplicate detection. Routers do not need to maintain multicast distribution trees or group membership.

The challenge with flooding is that they can lead to the *broadcast-storm problem*, where competing broadcast transmissions saturate the channel and reduce the reliability of the flood itself [123]. Simple flooding does not take advantage of the shared medium, resulting in many redundant transmissions by neighboring nodes that scales with increasing density. Some efforts attempt to minimize redundant transmissions by dynamically selecting a subset of nodes to act as multicast forwarders [18, 129]. However, these topology formation protocols create a binding to the underlying topology and require continuous maintenance. Alternative efforts rely on simple local rules, using probabilistic [107, 151] or adaptive [71] mechanisms to adjust transmission rates and suppress duplicates. However these efforts often trade reliability for energy-efficiency, and do not provide explicit reliability mechanisms. Such tradeoffs can make parameters deployment specific.

Within the sensornet space, Trickle has emerged as the de-facto dissemination algorithm [103]. Trickle uses a "polite gossip" policy, where nodes periodically rebroadcast the message but stay quiet if they have recently overheard the identical message. The rebroadcast period adjusts dynamically, using a short period when new messages are propagating through the network but a long period at other times to ensure reliable delivery with low overhead in the steady state. Trickle's suppression policy minimizes redundant transmissions, reducing energy and allowing it to scale well with large node densities. While Trickle was

originally developed for reliable code distribution, we carry its concepts forward to support IP multicast forwarding in sensornets.

7.3.2 Trickle-Based Multicast

IPv6 multicast operates over a specified scope, such as link-local scope or the sensornet scope we defined in Chapter 6. While link-local scope is ill-defined because it depends on the current radio connectivity, sensornet scope is well-defined and includes all nodes within the sensornet. In addition to these scopes, network administrators may choose to define other scopes. Drawing on the needs of typical sensornet applications, the multicast primitive should reliably reach all nodes within a well-defined scope. We believe multicast will be used for setting configuration parameters or issuing commands to multiple nodes, scenarios that require high end-to-end reliability. As with unicast forwarding, the primary goal of multicast forwarding is to maximize energy efficiency.

To implement multicast forwarding within the sensornet, we opt for a controlled flooding approach that minimizes forwarding state and eliminates group membership maintenance. Like other existing approaches for ad-hoc networks, all nodes within scope both receive and forward messages. Hosts subscribe to multicast groups simply by notify the network layer to pass up any multicast messages addressed to that group. Source nodes and border routers tag multicast datagrams with a sequence number using an IPv6 Hop-by-Hop Option or a 6LoWPAN adaptation header so that nodes can detect duplicate messages.

The forwarder implements the Trickle algorithm to manage forwarding transmissions [103]. Trickle provides efficient and density-aware forwarding as well as reliable delivery through its epidemic properties. Forwarding a multicast message can be viewed as a state consistency problem, where all nodes should eventually obtain the latest multicast message. The forwarder periodically advertises the current multicast sequence number using a Trickle timer. To minimize advertisement overhead, the forwarder piggybacks advertisements on Router Advertisements using a Router Advertisement option. As discussed in Chapter 6, Router Advertisement transmissions are already controlled by a Trickle timer. Receiving a different sequence number indicates inconsistency between neighbors and the forwarder responds by resetting the Trickle timer. Along

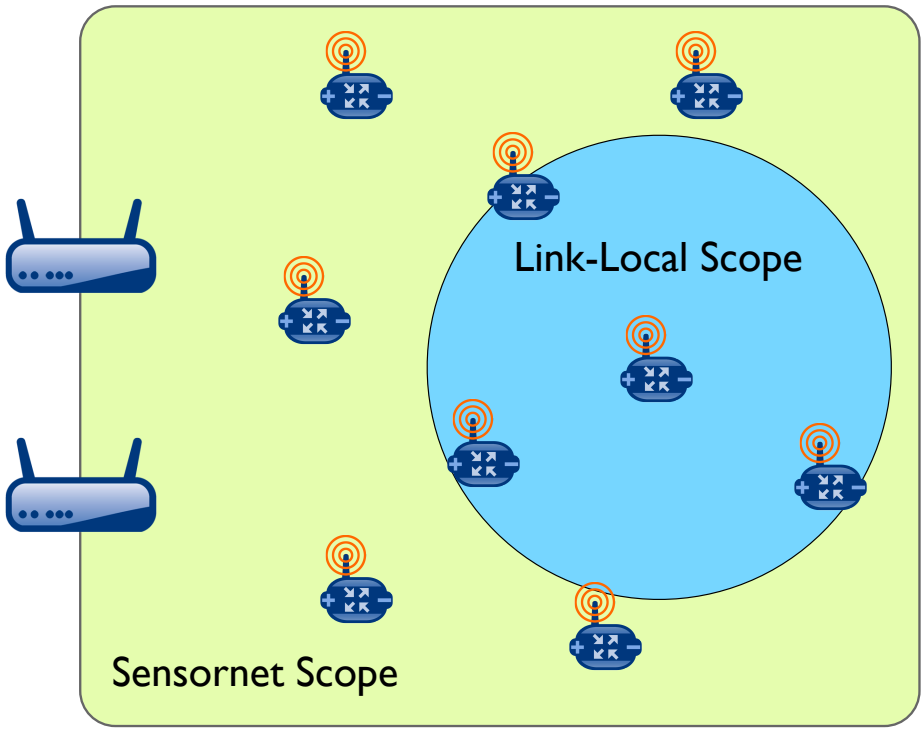


Figure 7.5: **Multicast Scopes.** Link-local scope is defined by the radio communication range, giving a fuzzy membership set due to time-varying characteristics of wireless communication. Sensornet scope scope, however, is well-defined and includes all nodes within the sensornet.

with the sequence number, a flag is used to indicate that the forwarder does not have any multicast forwarding state and the current sequence number is not known.

Receiving an older sequence number indicates that a neighbor does not have the latest multicast message. Because edge nodes do not periodically advertise their multicast state, they send Router Solicitations containing a protocol option to request transmissions from a neighboring router. When receiving an older sequence number or indication that a neighbor does not have any multicast state, the forwarder schedules transmission of the multicast message to occur just after the next advertisement. Coupling the multicast transmission with the Trickle-based advertisements randomizes the transmission time and allows suppression mechanisms to avoid duplicate transmissions. If the next advertisement is suppressed as a result of overhearing duplicate transmissions, transmission of the multicast datagram is also suppressed. If the multicast datagram requires fragmentation, the link transmits every fragment with each forwarding attempt. Stream-

ing optimizations allow efficient communication of fragmented datagrams, even when sent to the link-layer broadcast address. Nodes do not begin advertising a newer sequence number until they receive all fragments for the datagram corresponding to the latest sequence number.

The multicast forwarder implements a single forwarding channel, where nodes only try to synchronize to a single multicast message. Newer multicast messages override the propagation of older multicast messages. By enforcing a single forwarding channel, the multicast forwarder supports energy-efficient and reliable delivery with minimal forwarding state. Nodes maintain a single message buffer and an associated sequence number. Additional forwarding channels may be used if needed.

7.3.3 Related Work

IP multicast must provide high enough best-effort reliability to allow upper layers to communicate effectively. While multicast routing protocols face scalability issues in maintaining multicast distribution trees, efficient forwarding along the tree is relatively simple and consists of sending a message once on each link. Supporting reliable delivery for multicast, however, is not so simple and requires a feedback mechanism from all nodes in the multicast group back to the source. The goal of reliable multicast protocols is to mitigate this *feedback-implosion*.

Reliable Multicast Transport Protocol (RMTP) relies on acknowledgments from end nodes to deliver messages reliably [104]. To address feedback-implosion, RMTP utilizes a hierarchy of *designated receivers* that cache messages and listen for acknowledgments, an approach that is reminiscent of hop-by-hop recovery techniques commonly used for unicast delivery. Scalable Reliable Multicast (SRM) and Pragmatic General Multicast (PGM) both rely on negative acknowledgments (NACKs) instead [56, 156]. The use of NACKs allow these protocols to mitigate feedback-implosion through suppression techniques. While the suppression techniques were originally intended for wired networks, Trickle adapts them for use in ad-hoc wireless networks.

The shared communication medium of wireless, ad-hoc networks brings a significant challenge in supporting multicast communication. Limited memory and a dynamic topology has generally led most

to rely on flooding techniques to distribute messages. However, naive flooding can lead to the broadcast storm problem, where neighboring nodes saturate the channel while forwarding [123]. Some IP protocols that implement their own floods rely on selecting a subset of nodes to act as Multi-Point Relays [18, 129]. Simplified Multicast Forwarding (SMF) carries the topology maintenance concept forward to support general multicast communication [108]. However, maintaining such a topology can consume significant resources, especially when the topology is dynamic. Others protocols rely on probabilistic [107, 151] and adaptive [71] methods to avoid maintaining a forwarding topology. However, all of these ad-hoc protocols only provide best-effort delivery. If a node falls out of connectivity or fails to receive messages during the propagation period, it will fail to receive the message.

Trickle [103] builds on a large body of work in epidemic protocols, originally proposed for synchronizing databases [33]. Trickle's success as a fundamental primitive for dissemination is underlined by its use in many forwarding protocols proposed for sensornets. These protocols each target different applications, including dissemination of small application scripts [119], large application binaries [78], small set of attributes and commands [169], and even large numbers of attributes [105]. These protocols differ only in the metadata representation and the data transfer process when an inconsistency is detected.

Our use of Trickle in an IP framework brings the research cycle full circle. Trickle builds on a significant body of work originally intended for wired networks in the IP framework. Trickle adapted them for operation in sensornets. With this dissertation, we have finally brought Trickle back into an IP framework designed for sensornets.

7.4 Summary

In this chapter, we presented the design of an energy-efficient forwarder for both unicast and multicast datagrams. The network layer must provide high enough best-effort datagram delivery to allow end-to-end mechanisms to communicate effectively and achieve reliable transport. IP does not specify the mechanisms used to achieve high best-effort delivery and the mechanisms we presented in this chapter are

not specific to IP. The flexibility of IP allowed us to choose the mechanisms necessary to support sensornets, especially those developed within the sensornet community. The forwarder utilizes hop-by-hop mechanisms, where local detection and control allow quicker and more appropriate reactions. The unicast forwarder utilizes hop-by-hop retransmissions, streaming, congestion control, and QoS to achieve energy-efficient forwarding with high end-to-end success rates. The multicast forwarder employs the Trickle algorithm to achieve energy-efficient and density-aware flooding.

With the forwarder in place, the router is the last remaining piece of the network layer we have yet to address. The router is responsible for maintaining network-layer reachability by discovering and maintaining potential routes and configuring the forwarding table with the appropriate next-hop destinations. Subtle interactions between the forwarder and router can assist in improving the performance of the routing protocol while remaining energy-efficient. For example, by intelligently altering the forwarding table and utilizing data traffic, the router can avoid explicit probes to estimate link qualities and detect routing state inconsistency. In the following chapter, we discuss an efficient routing protocol for typical sensornet workloads.

Chapter 8

Network Layer - Routing

In this chapter, we address the final component of our IPv6-based network layer: routing. We present a baseline routing protocol designed for typical sensornet constraints and workloads. By concentrating routing state at border routers, our protocol to achieve $O(1)$ per-node routing state and $O(N)$ network-wide communication costs. Overall communication overhead in practice is reduced further by piggybacking on already existing traffic. The router piggybacks routing information on Trickle-based Router Advertisements and utilizes ambient data traffic to generate link quality estimates, enable loop and suboptimal route detection, and provide default route information to border routers.

8.1 Inferring a Connectivity Graph

For the network layer to provide reachability, it must configure routes to the destinations in use. The challenge is to ensure that the network forms and maintains consistent paths towards destinations in use that generally minimize some routing metric (e.g. distance, latency, utilization, or some combination). While an administrator may configure static routes by manually configuring forwarding table entries, doing so quickly becomes infeasible in large scale networks especially with dynamic topologies. Instead, most

practical networks today use dynamic routing protocols that discover the network topology and propagate routing information in an effort to form consistent and efficient paths to various destinations.

Ad-hoc wireless networks make dynamic routing protocols especially challenging because the underlying connectivity graph is not strictly defined. Traditional wired networks have relatively stable topologies with links that are either up or down, providing either very high success rates (e.g. greater than 99.9%) or no communication at all. In wireless networks, however, links to neighboring nodes are determined by environmental factors and often have a wide range of loss rates that vary over time. Even links with relatively high loss rates often provide some limited communication, and the routing protocol must decide whether or not to use those links. Furthermore, with time-varying link qualities due to changing physical and electromagnetic factors, the routing protocol must continuously evaluate links, consider their effects in the overall routing metric and path cost, and adapt if necessary.

Sensornets have different design considerations that those traditionally assumed by existing IP routing protocols. While existing intra-domain routing protocols for wired networks have proven their operation in large-scale networks, they often have large memory and communication requirements. Link-state protocols require nodes to maintain state about the entire network, with each node periodically flooding its link-state to all other nodes [20]. Loop-free distance vector protocols assume reliable, low-latency communication to ensure routing state consistency [3]. IP routing protocols developed for mobile ad-hoc networks (MANETs) are developed for wireless links, but many require frequent floods to discover and maintain routes. Assuming high and uncorrelated node mobility, MANET protocols often spend little effort in evaluating link qualities and some do not address asymmetric links well. Many assume that link estimation is entirely performed by lower layers, independently of the routing above. Sensornets, however, are more often composed of stationary nodes, giving nodes greater opportunity to evaluate links.

The strict resource constraints of sensornet nodes make the routing problem even more challenging. Limited memory often implies that a node can only maintain state about a limited set of nodes, far less than the number of nodes in the network and sometimes less than the number of neighbors within radio range. Limited throughput, small link MTUs, and limited energy constrain how often nodes can communicate with

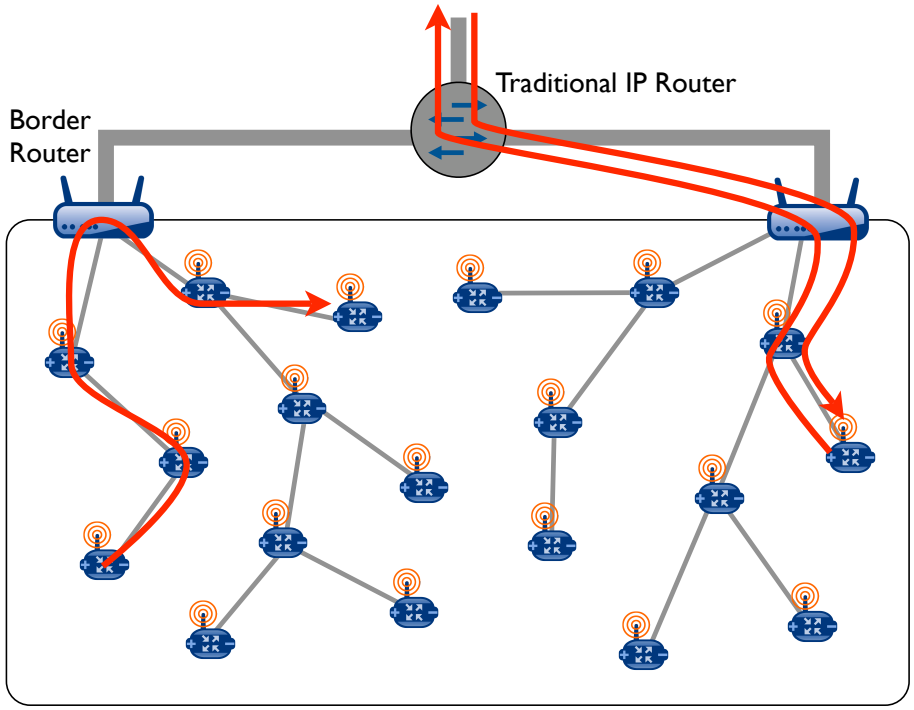


Figure 8.1: **Routing Architecture.** The IP framework aligns well with routing under partial information. Default routes provide reachability to any arbitrary IP devices by relying on other possibly more capable devices to maintain routing information. The IP framework takes no position on how routes are formed or whether they are optimal. Our routing protocol forms default routes to border routers and reverses those routes to provide reachability to all nodes in the sensornet. Because typical sensornet application workloads only require communication through a border router, the routes are optimal. By concentrating routing mechanisms at the border routers and accepting suboptimal routes for communication between arbitrary sensornet nodes, state and communication requirements for sensornet nodes remains small.

neighboring nodes. These constraints limit the discovery and evaluation of links to neighboring nodes. They also limit the amount of routing information the routing protocol can communicate and maintain to form coherent paths.

Limited resources implies that nodes must perform routing with partial information. In an IP framework, this means that nodes have next-hop information for a limited set of destinations and a default route for all others. IP requires basic reachability but takes no position on how routes are formed or if they are optimal.

In this chapter, we show that the IP framework aligns well with typical sensornet constraints and workloads. The routing protocol configures default routes to a border router. Each node then provides the border router with their chosen default routes, allowing the border router to maintain a spanning tree with host

routes to each node. Using the spanning tree, the border router forwards datagrams by including a routing header in datagrams it forwards. We show the overall routing architecture in Figure 8.1.

By concentrating routing effort at the border routers, our baseline routing protocol has minimal resource requirements for nodes within the sensornet. Sensornet nodes maintain small and constant state for configuring a single route: the default route. While memory requirements at border routers is linear with the number of nodes, border routers are often less constrained. Communication overhead for sensornet nodes is also minimal. The only communication requirements are occasional broadcasts from routers and unicast transmissions from leaf nodes to border routers. Using Trickle, the number of broadcast transmissions is low in steady state and scales well with network density (around a couple transmissions per hour in a given radio region). Overall communication overhead in practice is reduced further by piggybacking on already existing traffic. The router piggybacks routing information on Router Advertisements and utilizes ambient data traffic to generate link quality estimates, enable loop and suboptimal route detection, and provide default route information to border routers.

With minimal resource requirements, the routing protocol provides optimal routes when communicating through a border router, a traffic pattern typical to sensornet application workloads. The tradeoff with operating on such limited information is sacrificing route optimality in the general case. Note that IP says nothing about route optimality, just that the network must provide reachability. In some cases, this may not be an important case to consider. However, towards the end of this chapter, we discuss how to effectively handle such cases if needed.

8.2 Background

A routing protocol is responsible for discovering routes to destinations of interest. Traditionally, the router on a node provides route information to the forwarder through the forwarding table, but may also provide route information in the datagram itself. For example, the router may include a list of nodes to traverse in order to reach the destination. To discover routes, dynamic routing protocols must first discover usable

links and important properties of those links to neighboring nodes. Nodes must then propagate this topology information so that nodes can select routes while minimizing some metric. Dynamic routing protocols generally operate in a distributed fashion to provide better scaling characteristics, but can make it challenging to maintain consistent routing state across the network and quickly converge on coherent routing decisions.

Routing protocols generally fall into two classes: *distance-vector* and *link-state*. The earliest protocols were distance-vector protocols, implementing the Distributed Bellman-Ford (DBF) algorithm to compute paths [70]. Each node communicates its routing table to neighboring nodes, allowing neighbors to determine the cost of routing through the advertising node for each advertised destination. Nodes then select the neighbor with the minimum routing cost for each destination. DBF is simple, but can experience long convergence times, where inconsistent routing state can cause routing loops and the count-to-infinity problem. Simple mechanisms were developed to detect route inconsistencies involving only two nodes [70]. More complex mechanisms were developed to guarantee routing state consistency, but require reliable and in-order communication [3].

Link-state protocols were developed to address the convergence time problems often seen with distance-vector protocols. Link-state protocols have every node build a map of the entire network and compute routes using shortest-path algorithms over that map [20]. Each node discovers links to neighboring nodes and floods this information to all other nodes so that they can build a complete network topology. Link-state algorithms are widely used today in wired networks because they converge quickly and do not experience the count-to-infinity problem, only experiencing state inconsistencies when new information propagates through the network. The tradeoff is significantly greater state and communication requirements, which scale polynomially with the number of links in the network. For powerful routers with large memory and high bandwidth links, this is a sensible design point.

Both distance-vector [15, 91, 133, 134] and link-state [18, 129] protocols have been proposed for MANETs. Unlike routing protocols designed for wired networks, MANET protocols were designed specifically for the broadcast nature of wireless links and networks with high and uncorrelated mobility. As a result, MANET protocols often rely on flooding to distribute routing information and discover routes,

using sequence numbers to ensure that the flood terminates. Distance-vector protocols rely on the sequence numbers for loop-free route formation and often rely on link reversal, where routes are computed by reversing the path used for discovery. Link-state MANET protocols rely on floods to distribute topology information to all nodes, similar to those designed for wired networks. However, MANET protocols reduce state and communication requirements by dynamically selecting a subset of nodes to act as forwarding nodes. Only these forwarding nodes maintain topology information and forward messages, providing somewhat better scalability with node density.

MANET routing protocols are ill-suited for the needs of sensornets. MANET protocols are optimized for shortest-path routing between any arbitrary nodes and assume high and uncorrelated node mobility. These assumptions have led MANET protocols to rely on floods to discover and maintain routes, as well as increased routing state to achieve optimal shortest-path routing. Furthermore, these costs often grow polynomially with network size, properties that make MANET protocols infeasible for sensornets. Instead, sensornets generally have much more structured communication and much greater spatial stability. These characteristics offer significant opportunities to reduce resource requirements for dynamic routing protocols. Quasi-stationary networks also provide greater opportunities for link quality estimation.

Dynamic routing protocols for sensornets have primarily focused on minimizing routing state. Collection routing protocols achieve constant per-node state requirements by only optimizing communication towards a single destination [140, 169, 186]. Optimizing for a single destination allows the routing protocol to only evaluate the subset of links that offer the greatest opportunity for providing a good route to that destination. Hierarchical routing protocols only maintain state about neighboring nodes by routing along a tree-based topology [39, 131, 172].

Hierarchical approaches embed topology information in the names of nodes. IP's use of variable prefix lengths is intended to directly support hierarchical network organizations. However, relying on a strict hierarchy can lead to worst-case routing stretch that scales with the diameter of the network. Coordinate-based routing protocols attempt to achieve more optimal routes between arbitrary node pairs. Geographic routing protocols rely on knowledge of physical node locations so that nodes only need to maintain state

about neighboring nodes, but suffer when connectivity in the coordinate space is not complete [93, 100, 101]. Virtual coordinate routing protocols generate coordinates based on the underlying connectivity, but require additional resources to determine connectivity and construct a topology [58, 111, 122, 147]. A challenge with coordinate-based protocols is that they bind node names to the routing topology, making it more difficult to adapt to link quality changes or node mobility.

Utilizing link quality estimates in the path cost is an important component of routing protocols for sensornets. Some protocols take a simplistic view, relying only on physical-layer metrics such as received signal strength indicator (RSSI) or the chip correlation value [140, 169]. Physical-layer link metrics are convenient because they are computed by the radio for each received frame, but can have high variance and do to directly indicate bidirectional packet error rates. Other protocols attempt to compute packet error rates directly by maintaining state about communication failures. Early methods relied on broadcast messages with sequence numbers, allowing neighboring nodes to compute packet error rates in a single direction [58, 186]. Most recently, a link estimator proposed the use of link-layer acknowledgments to compute bidirectional packet error rates [57]. Directly measuring packet error rates provides the most relevant data, but requires more time, energy, and state to compute. As a result, the link estimator combined both physical-layer metrics and direct evaluation of packet error rates.

The strict resource constraints of sensornets have significant implications on routing protocols. Limited memory and communication capabilities imply that routing protocols must operate with partial information. In turn, partial information implies that the router can only optimize routes to a small number of destinations and accept suboptimal routes for arbitrary node pairs. While routing protocols seek to develop and maintain a global structure, nodes must be able to make quick local decisions. Rather than attempting to maintain fully consistent routing information across the network, nodes should make optimistic decisions and resolve inconsistencies if they occur. In general, we believe these tradeoffs are necessary for distributed protocols to operate effectively under strict resource constraints. In the following sections, we present a routing protocol for typical sensornet workloads that has minimal state and communication requirements.

8.3 Default Routes

In this section, we describe how the routing protocol selects and maintains default routes. The routing protocol configures and maintains default routes towards border routers, utilizing ICMPv6 Router Advertisements to discover neighboring routers and communicate routing information. The router maintains a routing table to manage a set of default route possibilities and sorts them based on path cost and confidence in the link quality estimate. The router typically selects the top entry to use as the default route, but may choose an alternate entry to support re-routing or occasionally search for better routes. While routes may be selected based on ETX [22] or other metrics, state inconsistencies are detected using hop count relative to the border router. The hop count provides a more stable indicator that minimizes dependencies on the performance of individual links along the path. By piggybacking on ambient data traffic, the router requires little communication overhead and does not require any explicit control messages. State requirements are also constant.

8.3.1 Discovering Potential Routes

The router uses Router Advertisement messages to announce the presence of routers and allow nodes to discover neighboring routers. While traditional IPv6 routing protocols use explicit datagrams specific to the routing protocol, our router piggybacks information on existing Router Advertisements using a protocol option. The needs of advertisement messages for Neighbor Discovery and routing protocols are complimentary - both need to discover neighboring nodes and propagate information over multiple hops. By piggybacking on messages already required for Neighbor Discovery, the router adds minimal communication overhead. Furthermore, our Neighbor Discovery protocol defined in Chapter 6 already adjusts the Router Advertisement transmission period using the Trickle algorithm. We add hooks to allow the router to reset the transmission period.

To discover nodes quickly, the router may transmit ICMPv6 Router Solicitation messages to request Router Advertisements from neighboring nodes. The router may wish to discover nodes when an external

event requires the node to join a network quickly or when the number of entries in the routing table falls below some threshold. In the latter case, nodes may wish to also transmit Router Advertisements to communicate a change in routing state. As a result, we add a *Router Solicitation* flag to the Router Advertisement message so that nodes can propagate updated routing information and solicit Router Advertisements from neighbors using a single message.

The routing protocol's routing option carries any information that the routing protocol needs to advertise. The routing protocol includes hop count relative to the nearest border router and a path metric for selecting routes. The hop count information is used to detect routing inconsistencies and makes the mechanism for detecting route inconsistencies independent of the path metric. We discuss its use further in Section 8.3.4.

In this chapter, we consider a simple path metric: expected number of transmissions (ETX) to reach the Border Router [22]. The ETX metric is useful because it captures link qualities along the path towards the destination as well as hop count. However, we do envision more complex metrics that may include other link-specific properties as well as node-specific properties. While traditional routing protocols are mostly concerned about link properties, node resources (e.g. memory and energy) can vary widely in sensornets and routing protocols should take advantage of extra resources whenever possible. We also envision support for multiple routing topologies with different metrics by including multiple routing options in the Router Advertisement. The router is also not limited to configuring default routes and may include additional information to route to other destinations. However, because we are focused on developing a baseline routing architecture, we do not address these more complex mechanisms in this chapter.

8.3.2 Managing the Routing Table

The router stores state about potential routes that it discovers in the routing table. Of these potential routes, the goal is to select one to insert as the default route entry in the forwarding table. The separation between routing and forwarding tables is especially important in wireless networks - the router must spend some time to evaluate a link and compare it to other possibilities before using it to route datagrams. When

inserting a potential route into the routing table, the router pins the associated entry in the link's neighbor table. Doing so ensures that the link layer will maintain link quality estimates for that link, which is essential to include in the path cost when selecting a default route. Pinning the neighbor also notifies the link layer of links that are more likely to be used, allowing the link to more effectively optimize transmissions over them (e.g., by learning and maintaining receiver schedules).

Limited memory makes selecting a default route challenging. The longer an entry remains in the table, the more information the link layer gains about that link. For newly discovered neighbors, the link only provides very coarse information of the link quality: perhaps a single sample of RSSI and chip correlation for the received Router Advertisement. Both have high variance and are not true indicators of packet error rates [157]. With each transmission over a link, the link layer can compute a new link quality estimate with higher confidence. While the router should favor links with higher confidence in link quality estimates, the router should also be accepting of new links in the case that they may provide a lower routing cost. Limited memory means that the router may have to evict routing table entries and the associated link quality estimates that took time and energy to generate.

To deal with the tension between favoring more experienced links and accepting new discoveries, the router sorts the routing table first by confidence in the link quality estimate then by path cost (which includes the link quality estimate). Routes with a low path cost and high confidence in the link quality estimate appear at the top of the list, while routes with higher path cost and lower confidence appear at the bottom. Newly discovered routes always appear at the end of the list, as the link layer provides very low confidence for those link quality estimates. However, routes with low path cost bubble up the list as confidence in the link estimate increases. During normal operation, the routing table serves as a low-pass filter for newly discovered routes. However, the router also allows quick route formation during bootstrap as well. In the initial case where the list is empty, all route entries appear with low confidence and sorted based on what little information is known.

The router evaluates an entry in the routing table whenever the link sends or receives a message over that link, which generates updated link quality estimates. Managing the routing table involves three

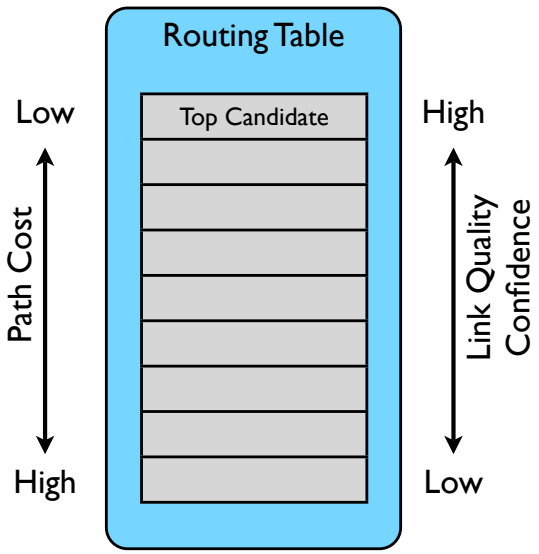


Figure 8.2: **Managing the Routing Table.** While the router should prefer entries with a high confidence in the link quality estimate, the router should also be accepting of new links that could provide a lower path cost. The router sorts the routing table by confidence in link quality estimate and the advertised path cost. The router only inserts entries at the bottom of the list and good routes with high link quality will bubble up the list as confidence in the link quality estimate increases. The routing table serves as a low-pass filter for accepting new routes.

basic operations: (i) *insertion* into the routing table, (ii) *promotion* within the routing table, and (iii) *removal* from the routing table. New entries are always inserted at the end of the list and only if the physical-layer information (RSSI and chip correlation) are above a threshold that will likely provide an acceptable link success rate. This threshold may be adaptive based on information gathered about the environment, such as the noise floor. If the routing table is full, the router chooses whether or not to evict the bottom entry. The router does not evict the bottom entry if the link quality confidence is below a minimal threshold. High churn in the bottom entry could prevent the router from discovering new routes. If the link quality confidence is above the threshold, the router replaces the bottom entry if one of the following is satisfied:

1. The advertised path cost for the new route is significantly less than the advertised path cost of the bottom entry.

2. The advertised path cost for the new route and bottom entry are similar and the new link has significantly better physical-layer link quality estimates than the bottom entry.

The advertised path cost is what the neighbor advertised in its Router Advertisement and does not include any link quality information to that neighbor. The router compares physical-layer information for the link as that is the only information it has about the link to compare. The router also does not combine the advertised path cost and physical-layer information into a single metric because the latter does not provide any direct indication of bidirectional packet success rates.

The router promotes entries in the routing table by moving them up one position in the list, but only if the entry has higher confidence in the link success rate and a lower path cost than the entry it is swapping with. Note that the path cost incorporates the advertised path cost and the link success rate into a single metric. We measure confidence using a simple metric: the number of transmission attempts over that link. However, standard statistical methods that also incorporate standard deviation to compute confidence intervals may also be used at the cost of greater computational requirements. The router evaluates the promotion of an entry each time a transmission attempt occurs over that link, which causes the link layer to update the link success rate estimate. The router promotes an entry only if one of the following is satisfied:

1. The entry has a significantly lower path cost and a higher confidence in link success rate than the entry above.

2. The entry has a similar path cost than the entry above and the confidence in link success rate is above an acceptable threshold.

The latter case allows the router to easily support load balancing by utilizing different routes and distributing the load between neighboring nodes. The acceptable confidence level may be adaptive based on observed environmental factors or other entries in the routing table.

The router removes an entry from the list when its link success rate falls below a threshold. Removing nodes from the list promotes all entries below it. If the removed entry was selected as the default route, the router selects a new default route. One challenge is to ensure that the router does not prematurely remove entries due to transient changes in link quality. In the following section, we discuss how the router

performs re-routing and occasionally searches for better routes. In doing so, the router will attempt to utilize other neighbors rather than continuing attempts to the same failing neighbor.

8.3.3 Selecting a Default Route

The router normally selects the top entry in the routing table to use as the default route in the forwarding table. However, the router may occasionally choose other entries for two reasons: (i) to support re-routing when consecutive transmission attempts to the top entry fails and (ii) to probe other candidates, increase confidence in the link success rate, and search for candidates to promote without using explicit control messages. While using other entries, the router continues to advertise the path cost when routing through the top entry.

The router detects repeated failures by tracking the link success rate of the default route it has configured. If the success rate goes down after several consecutive attempts, the router switches into a re-routing mode by randomly selecting alternative entries in the routing table to serve as the default route, as shown in Figure 8.3. By switching entries, the router attempts to utilize receiver diversity to route around any issues that may be causing transmission failure to the top entry. Switching entries also allows the router to search for better routes to promote. By changing the default route, the forwarder utilizes different links and allow the link layer to generate updated link success rates.

The re-routing mechanism represents an example of where the router is allowed to make quick local decisions when the prior optimal global decision is failing. Assuming that the route information is up-to-date, routing loops will not occur when selecting entries with a hop count less than or equal to the top entry. As a result, the router should favor entries that do not generate any routing loops. In some cases, however, it may be necessary to accept the possibility of routing loops and choose an entry with a higher hop count. Routing loops may occur when routing information is inconsistent. Randomly selecting entries while re-routing helps minimize the occurrence of routing loops.

The router must spend some effort to search for lower cost routes and keep link quality estimates up-to-date for entries in the routing table. Explicitly sending a message and receiving an acknowledgment

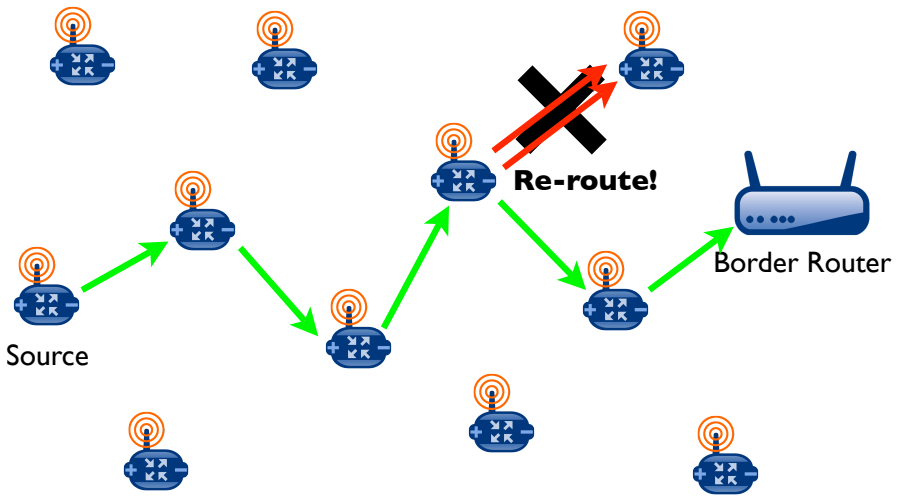


Figure 8.3: **Re-routing.** If the router detects failures on the current default route, the router begins to select other entries in attempt to utilize receiver diversity to forward the packet.

provides the most representative way to estimate link success rate. More frequent probes provide more information about the link, but is also more costly. Link success rates are also time dependent and if links are not used in the near future, then it is less important to keep link quality estimates up-to-date. Ideally, the router's effort in link quality estimation should scale with the traffic load.

Rather than relying on explicit control messages, the router also generates link quality estimates by dynamically changing the default route entry in the forwarding table. The router occasionally configures the default route with other entries to continuously search for routes with similar or lower path costs, even when the top candidate is performing well. For a small, random fraction of messages that utilize the default route, the router configures the default route to other routes that have a hop count less than or equal to the top entry, as shown in Figure 8.4. If multiple candidates exist, the router rotates between them. However, the router continues to use an entry until a transmission failure to that node occurs, allowing a route with lower advertised path cost to quickly bubble up the list if the link is good. Only testing entries with equal or lower hop count ensures that routing loops do not occur. A typical network with moderate density often has multiple neighbors that provide similar hop counts. Allowing the router to probe entries with equal hop count naturally supports load balancing, by distributing forwarding load to different next-hop neighbors over time.

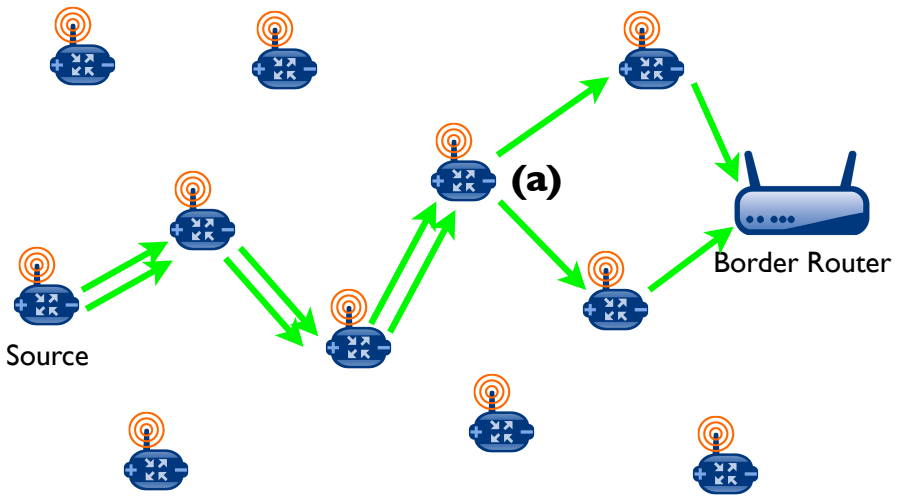


Figure 8.4: **Updating Link Quality Estimates.** If one or more routing entries have a hop count less than or equal to the top entry, the router selects those as default routes for a small, random fraction of forwarded data packets. In doing so, the router can update link quality estimates and continuously search for better routes without requiring explicit probe messages. Router (a) in this figure forwarded the last two datagrams on different links.

Using ambient data traffic, the router does not generate any extra messages to maintain link quality estimates and search for lower cost routes. The rate of updating link quality estimates also scales naturally with the traffic rate. While it is possible for the router to stop evaluating links when there is no traffic at all, we expect that sensornet applications will generate some minimal baseline traffic for management purposes. Other control protocols also require periodic traffic to maintain soft state. We present one example for maintaining host routes in Section 8.4.

8.3.4 Maintaining Route Consistency

Routing information may become inconsistent whenever changes in have not yet been communicated to other nodes in the network. Inconsistent routing information may result in the network using higher cost routes than it could be if the routing information was up-to-date. In the worst case, inconsistent routing information can lead to routing loops. Some existing routing protocols take a proactive approach in actively communicating any changes is routing information, but doing so can be resource intensive especially relative to the low data rates and constraints of sensornets [3, 20]. Instead, our routing protocol takes a passive

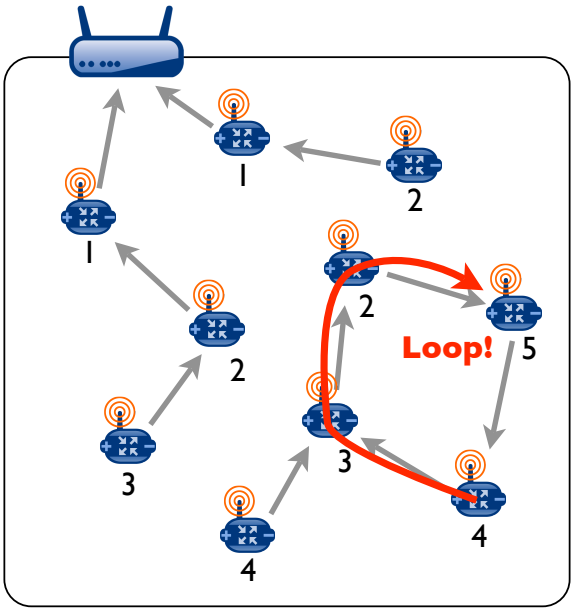


Figure 8.5: **Detecting Routing Loops.** The router detects routing loops by including the expected hop count of the next-hop destination in each forwarded packet. An increase in the hop count indicates the possibility of a routing loop and inconsistent routing information.

approach where nodes communicate updated routing information when an inconsistency is detected. This follows our design principles of allowing nodes to make optimistic decisions locally and resolve inconsistencies only when they occur.

The router detects possible *routing loops* by using hop count information relative to border routers. By including hop count information in the Router Advertisement, nodes learn the hop count of their neighbors relative to each border router. The forwarder includes the expected hop count of the next-hop destination when forwarding along the default route by including it in an IPv6 Hop-by-Hop Option or 6LoWPAN header depending on what layer forwarding occurs. Receiving a message with a hop count greater than the node's current hop count indicates an inconsistency and the possibility of a routing loop, as shown in Figure 8.5.

The router detects possible *path inefficiencies* and opportunities to improve route selection by observing significant differences in the advertised path cost. The path cost can increase significantly when links on the route experience higher loss rates than before. Similarly, the path cost may decrease if a lower cost route along the path was found (e.g., by removal of an obstruction). To detect changes in the path cost, the

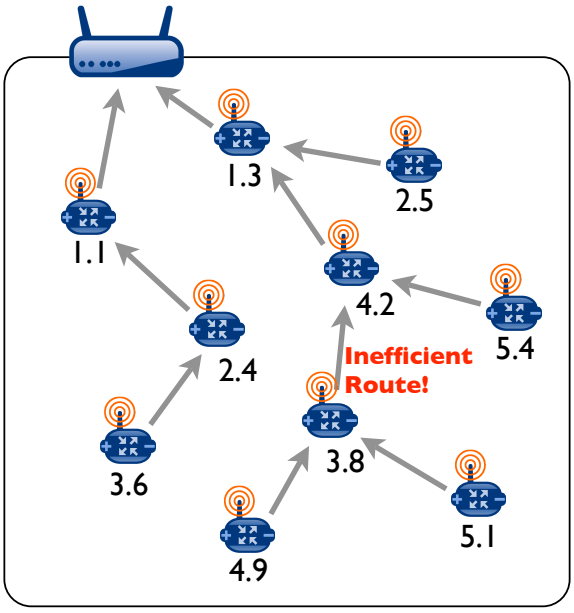


Figure 8.6: **Detecting Routing Inefficiencies.** The router detects inefficient routes by including the advertised routing cost from the next-hop destination in forwarded packets. A routing cost that is significantly different indicates that nodes may utilize more efficient routes if they are updated with more recent path costs. Note that an increase in path cost does not necessarily indicate a routing loop. By decoupling loop detection, the threshold for difference in path cost represents a tradeoff between how much effort to place in keeping path cost information up-to-date and possibly using less efficient routes due to stale routing information.

forwarder also includes the last advertised path cost received from the next-hop destination when forwarding datagrams along the default route. Receiving a message with a significantly different path cost indicates that the default route selection may be suboptimal, since the sender is using stale routing information, as shown in Figure 8.6.

In both cases, the router responds by resetting the Router Advertisement period to more quickly communicate updated routing information to neighboring nodes. Doing so, however, increases communication costs. By using both hop count and advertised path cost, the router can react differently to the possibility of routing loops and routing inefficiencies. In the case of routing inefficiencies, the router can make a tradeoff between sending updated routing information or allowing suboptimal routes. However, the occurrence of routing loops can prevent the network layer from providing reachability and the router should attempt to resolve them quickly. By decoupling the hop count and path cost, the router can avoid reacting to changes in path cost when the hop count indicates no existence of routing loops.

8.4 Host Routes

Default routes provide reachability from sensornet nodes to the border router and any other IP devices connected to other IP networks. To provide reachability to sensornet nodes, our routing protocol also forms host routes to each individual sensornet node. To efficiently construct and maintain host routes, our protocol concentrates routing efforts at border routers. Sensornet nodes report their default routes to the nearest border router. The border router then reverses those routes to form host routes back to each sensornet node. Link reversal is possible because default routes are only selected based on bidirectional connectivity. The border router forwards a datagram to nodes within the sensornet by inserting a routing header that contains the route to the destination. Using source-based routing at the border router, sensornet nodes do not need to maintain any state for host routes. While nodes must periodically unicast route information to the border router, flood operations are never required.

The combination of default routes and host routes at the border router allows the network-layer to provide reachability between a sensornet node and any arbitrary IP device, including sensornet nodes within the same sensornet, sensornet nodes in other sensornets, and any other IP device connected to other IP networks. Note that routes to and from external IP devices are optimal from a sensornet perspective, as the metric for selecting default routes is to minimize the cost of forwarding datagrams to border routers. Communication with external devices is typical for many sensornet applications. Data collection applications usually communicate data to a central data server. Control applications often receive control instructions from a central server as well. In the following sections, we describe how our routing protocol configures and maintains host routes.

8.4.1 Learning Host Routes

Sensornet nodes provide information of their default routes by periodically sending *record-route* messages to the border router using their default routes. As a side benefit, the periodic record-route messages allow the router to update link quality estimates and refine routing decisions for the default route. Using

an option in the Router Advertisement, the border router indicates how often nodes should send registration messages back to the border router. Nodes may also generate registration messages when the top entry changes, to ensure reachability from the border router.

The record-route message contains an ordered list of nodes that forwarded the message. Each node that forwards the message appends its address to the list. The record-route message may be included in any datagram by using an IPv6 Hop-by-Hop Option header when forwarding at the network layer or a 6LoWPAN adaptation header when forwarding below. When there is ambient data traffic, the forwarder piggybacks a record-route message in enough datagrams to satisfy the advertised registration period. If the existing traffic rate is less than the advertised registration period, the node must generate its own datagrams simply to communicate record-route information.

Because record-route messages provide default route information for the entire path between the source and the border router, nodes can avoid generating their own record-route messages whenever they have recently processed and forwarded a record-route message. Suppressing record-route messages allows the cost to scale with the number of leaf nodes rather than the total number of nodes. This savings are most significant in long linear networks that have a very small number of leaf nodes.

8.4.2 Border Routing

Using the default route information provided by each sensornet node, border routers can generate a spanning tree of the entire network and use it for generating host routes back to each node. When a border router receives a datagram destined for a node within the sensornet, it performs a lookup in the spanning tree to determine a route to the destination. If no valid route is available for that node, the border router generates an ICMP Host Unreachable error. If the destination is within radio range of the border router, the border router forwards the datagram as normal by setting the link header's destination address to the destination. If the destination is multiple hops away, the border router inserts a routing header that contains a list of addresses in the packet to reach the final destination. Nodes forward the packet by processing the routing header to determine the next-hop destination for the packet.

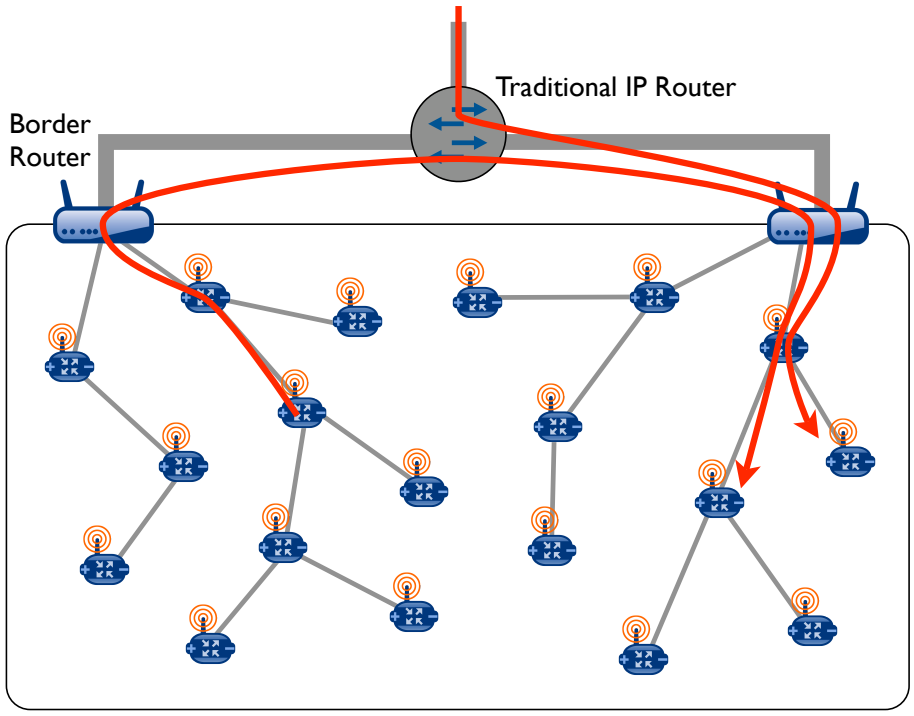


Figure 8.7: **Routing Among Multiple Border Routers.** Multiple border routers can support a network by sharing IP host routes between them. Because sensornet nodes select routes to the nearest border router, border routers naturally forward datagrams to the border router nearest the destination, taking advantage of the more capable network that connect border routers.

The address list may be included in an IPv6 Routing header when forwarding at the network layer or in a 6LoWPAN adaptation header when forwarding at the link layer. In both cases, each address entry is equivalent a short 16-bit link-layer address. The same techniques used to compress IPv6 headers can be used in the routing header as well. While using short addresses makes the address list more compact, it does require nodes to assign short addresses to the wireless interface. When using an IPv6 Routing header, the header is similar to a Type 0 routing header [31] but each address entry represents the lower 16 bits of the Interface Identifier of a routable IPv6 address. The global routing prefix is assumed to be the same for all addresses in the list. The Type 0 routing header has been deprecated for security reasons and we respect those security concerns by creating a new routing type and only allowing use of the routing header within the sensornet [2]. Border routers should not forward any datagrams that already include a routing header.

In many cases, it may be desirable to support a sensornet using multiple border routers, as shown in Figure 8.7. By adding border routers, the network administrator can increase energy efficiency, reduce channel utilization, and reduce communication latency by using border routers to lower the average number of hops between sensornet nodes and the nearest border router. Adding border routers can also increase network robustness by providing more than one border router for nodes to communicate through. If a border router fails, nodes routing through that border router will reconfigure their default routes and register themselves to a different border router.

IP trivially supports routing between multiple border routers using standard IP-based mechanisms. Border routers simply need to exchange host routes among each other. The border routers may be directly connected over a highly capable link (e.g. Ethernet), in which case they simply need to advertise host routes across the link. Neighbor Discovery Proxy mechanisms may also be used to effectively form routes between border routers [167]. Using ND Proxy, border routers respond to ND queries as if they are the node itself giving the illusion that all sensornet nodes are on-link. As a result, routers will forward datagrams to the appropriate border router. When border routers are not connected to the same link, either the transit network needs to configure host routes for sensornet nodes or the border routers must be directly connected using tunnels to form an overlay network that emulates a single IP link. All configurations allow the surrounding network to forward datagrams to the appropriate border router before injecting it into the sensornet.

8.5 Discussion

8.5.1 Overhead Analysis

The routing protocol presented in this chapter represents a simple baseline that provides reachability to any arbitrary IP device inside or outside the sensornet with minimal overhead while also establishing optimal routes for typical sensornet application workloads. By pushing routing state to the border routers, the routing state requirement on sensornet nodes is small and constant - that required for configuring default routes (64 bytes). Memory requirements at the border router is linear with the number of sensornet nodes,

but border routers generally have greater memory capacity. Note, however, that the border router's state requirements are independent of network density because the border router only maintains a spanning tree rather than complete connectivity information. When using multiple border routers, the state requirements can be easily distributed across them using standard IP mechanisms.

The only communication requirements are occasional broadcasts from routers and unicast transmissions from leaf nodes to border routers. Using Trickle, the number of broadcast transmissions is low in steady state and scales well with network density (typically around a couple transmissions per hour in a given radio region). Local route repairs limits the effect of a link failure to only a subset of the network. If a link fails, the router performs re-routing. In the best case, only a single node needs to update its default route. In the worst case, all descendants of that node may also have to update their routes, but the repairs are localized to those descendants. Overall communication overhead in practice is reduced further by piggybacking on ambient traffic required for other functions. The router piggybacks routing information on Router Advertisement messages and utilizes ambient data traffic to generate link quality estimates, enable loop and suboptimal route detection, and provide default route information to border routers.

The routing architecture is able to minimize communication and memory overhead for sensornet nodes by relying on more capable border routers. By concentrating routing efforts at the border routers, sensornet nodes only need to communicate with and maintain state for the nearest border router. In contrast, completely distributed protocols require nodes to communicate with and maintain state about any destinations in use and possibly all other nodes, often through floods. In Chapter 9, we provide empirical data for energy and memory requirements of a real-world application deployment.

8.5.2 Improving Routing Stretch

The baseline routing architecture achieves minimal communication overhead with minimal and constant state requirements at the cost of suboptimal routes when sensornet nodes communicate to other nodes within the same sensornet. A common metric for describing the optimality of a route is the *routing stretch*, which describes the ratio of the selected route's path cost to the optimal path cost. We simplify the

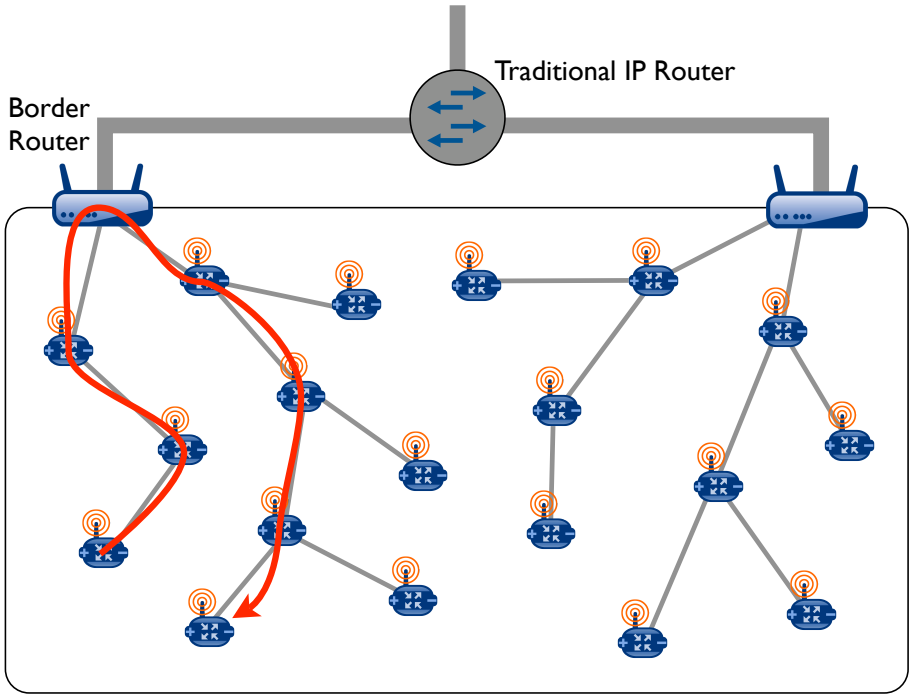


Figure 8.8: **Worst-Case Routing Stretch for Baseline.** Concentrating routing efforts at border routers minimizes the routing protocol's state and communication requirements, but creates suboptimal routes between nodes within the same sensornet. The worst-case routing stretch is bounded by $2D$ and occurs between two neighboring nodes furthest away from a border router.

discussion by using hop count as the path cost and denote D to represent the diameter of the network in hops. With the baseline routing architecture, the worst-case routing stretch is bounded by $2D$, which occurs between two neighboring nodes that are furthest away from the router, as shown in Figure 8.8.

For some sensornet applications, the worst-case routing stretch also happens to be the most common case for direct node-to-node communication. Due to the physical nature of sensornet applications, nodes that are within physical proximity are more likely to communicate directly with each other. A light switch, for example, will most likely control a lamp within the same room and often within a single radio hop. Fortunately, this common case is the simplest case to optimize - nodes simply discover neighboring nodes and insert them into the forwarding table. For routers, there is no added communication cost as Router Advertisements already provide the discovery mechanism. Edge nodes need to send Neighbor Advertisements to announce themselves. Neighbor state, however, scales with the number of neighboring nodes. Ideally application-layer knowledge should indicate important neighbors to maintain, similar to how the router indi-

cates what neighbors to the link by pinning potential default routes in the link's neighbor table. This simple optimization reduces the upper bound on routing stretch to $\frac{2D}{2} = D$ and occurs for nodes separated by two hops.

We can generalize the solution to optimize routes that involve more than one hop by allowing nodes to discover n-hop neighbors through scoped discovery mechanisms. In doing so, we can reduce the upper bound to $\frac{2D}{n+1}$. However, unlike the 1-hop special case, supporting two or more hops requires routers to exchange additional routing information. Specifically, routers must advertise routing costs for each potential destination within the n-hop scope and scales with node density. The added costs may be feasible for a small scope, but quickly become infeasible for larger scopes.

An alternative approach is to generalize the routing mechanisms concentrated at the border routers. Rather than only providing information about default routes, nodes could provide the border router information about links to neighboring nodes as well. Implementing a centralized link-state routing protocol, the border router can configure routes by injecting forwarding table entries in individual nodes. As with the baseline architecture, concentrating routing efforts at the border routers reduces both communication and state requirements relative to a completely distributed approach. However, doing so comes with some cost. Nodes need to source their own registration messages to communicate neighbor tables rather than using a record-route message. The need to communicate with a border router can add delay to configuring a route when the network is deep. Furthermore, when the number of neighbors exceeds a node's neighbor table size, the border router may have to operate on incomplete information. We leave practical methods for achieving optimal point-to-point routes with minimal communication and state overhead for future work.

8.6 Related Work

There has been significant work in dynamic routing protocols for ad-hoc wireless networks. MANET routing protocols generally focus on networks with high and uncorrelated mobility, requiring floods and quick discovery mechanisms to discover routes effectively. Like traditional routing protocols, MANET protocols

Protocol	IPv6	DSDV	AODV	DSR	OLSR	GPSR	MintRoute	S4
Per-node State	$O(1)$	$O(N)$	$O(PN)$	$O(PN)$	$O(N^2)$	$O(1)$	$O(N)$	$O(\sqrt{N})$
Route Formation	$O(N)$	$O(N^2)$	$O(PN)$	$O(PN)$	$O(N^2)$	n/a	$O(N^2)$	$O(N\sqrt{N})$
Local Recovery	yes	no	no	no	no	yes	yes	yes
Routing Stretch	$O(D)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$
Link Estimation	yes	no	no	no	yes	no	yes	yes
Name Binding	no	no	no	no	no	yes	no	yes

Table 8.1: **Routing Protocol Comparison.** "IPv6" represents the routing protocol presented in this chapter. Per-node state represents the amount state for an individual sensornet node. Route formation is the network-wide communication cost of generating and maintaining routes. Local recovery indicates if recovery from link and node failures remain localized to a subset of the network. Routing stretch represents the quality of the chosen routes compared to the optimal route. Link estimation represents whether or not the protocol evaluates link success rates and includes them in the path cost. Name binding represents whether or not node names are bound to the routing topology.

have focused on providing optimal point-to-point routes. As a result, many MANET protocols can have state or communication requirements that scale polynomially with the number of destinations or nodes. Routing protocols designed specifically for sensornets generally focus more on bounding the communication and state requirements, and do so by accepting suboptimal routes. Sensornet protocols also rely on the more stationary nature of sensornets to utilize more effective link quality estimation for inclusion into the path cost.

8.6.1 Comparison to Existing Protocols

In this section, we compare our routing protocol to other representative routing protocols designed for ad-hoc wireless networks. We consider the state and communication requirements for configuring default routes and host routes as well as the routing stretch. We also consider whether or not the protocols support localized recovery mechanisms, consider link quality estimates in their path cost, and whether or not the protocol binds node names to the routing topology. In this analysis, we denote N as the number of nodes in the sensornet, D as the network diameter, P as the number of communicating pairs, and n as the number of neighbors for a given node. We summarize the comparison in Table 8.1.

Destination-initiated routing protocols, such as DSDV [134] and MintRoute [186], require all nodes to initiate floods in order to provide reachability to all nodes in the network. Because destinations do not know whether other nodes would like to communicate with them, every node must initiate floods to maintain

reachability. As a result, destination-initiated protocols require $O(N)$ state and $O(N^2)$ communication, which is infeasible for typical sensornet constraints.

Source-initiated routing protocols are reactive, allowing their costs to scale with the number of communicating pairs. However, configuring default routes requires all nodes to form routes to a border router, implying that the number of communicating pairs P is equal to the number of nodes in the network N. We expect N to be the lower bound on P with typical application workloads, as each sensornet node typically reports application and management data to a central server. For most sensornet application workloads, we expect P to be larger than N in order to provide reachability from the border router to individual sensornet nodes, necessary for configuration and management (e.g., DHCPv6) as well as control applications. In the worst-case, the number of communicating pairs P can be N^2 and makes the cost of such protocols infeasible for typical sensornet constraints.

Routing protocols based on geographic coordinates have excellent scaling characteristics, with constant state requirements and potentially no communication requirements to setup routes [93]. However, geographic protocols bind the names of nodes to the geographic topology. Furthermore, the geographic topology may not reflect the actual radio connectivity and can fail to provide an effective route for delivering datagrams.

In comparison, our baseline routing protocol achieves requires $O(1)$ state and $O(N)$ communication costs, but does so at the cost of increased routing stretch. Our routing protocol allows local recovery to link failures through the use of re-routing. Unlike coordinate-based routing protocols, our routing protocol does not create a binding between routing topology and node names.

8.6.2 Mobile Ad-Hoc Networks (MANET)

DSDV is a proactive, distance-vector protocol where each node maintains and advertises route information for each destination [134]. While fairly similar to RIP, DSDV differs mostly in how it avoids routing loops. Along with path cost, the route information for each destination includes a sequence number to indicate freshness, which implements a logical flood from each destination. Routes are selected first by sequence number then by path cost. Due to communication delays, the router must delay choosing new routes

to allow other neighboring nodes to advertise updated route information and the delay can be dependent on the specific network deployment. DSDV repairs link failures by relying on periodic DSDV floods from the destination. While the construction of default routes is similar to DSDV for a single destination, our routing protocol is more closely related to RIP, as it does not utilize sequence numbers to ensure loop-free routes. Furthermore, our routing protocol does not require continuous floods. Nodes asynchronously broadcast routing information and the transmission rate depends on local network conditions.

AODV is a reactive distance-vector protocol that does not generate any control traffic when nodes are not communicating with each other [133]. To communicate with a destination, the source node must first discover a route to it by initiating a flood. Nodes forwarding the discovery message cache a reverse route back to the source. After receiving the discovery message, the destination replies back to the source to configure the routes between them. Nodes only maintain state about paths between communicating node pairs. Like DSDV, discovery messages include a sequence number to avoid routing loops. AODV repairs link failures by initiating new discovery floods. DYMO improves on AODV by specifying more compact packet formats, smaller neighbor table entries, and optimizations that reduce control overhead [15]. However, like AODV, DYMO still requires nodes to discover destinations through floods. To configure default routes, every node must initiate a discovery flood and cache discovery request for all other nodes. In contrast, our routing protocol never uses floods to discover nodes.

DSR is a reactive distance-vector protocol where the each packet includes a route to the destination [91]. Similar to AODV and DYMO, DSR discovers routes through flooding. However, the DSR discovery reply provides the source with the full route to the destination node. As a result, only the source node needs to maintain routing information and does so by storing a spanning tree for nodes that it must route through to reach all destination nodes. DSR repairs link failures by initiating new discovery floods. Like DSR, our routing protocol maintains source routes but only at border routers. Our protocol differs in that it does not attempt to provide shortest-paths between any arbitrary node pair. As a result, our protocol does not require discovery floods and sensornet nodes do not need to maintain routes for destinations other than the default route.

OLSR is a proactive link-state protocol designed for MANETs [18]. The basic operation is similar to OSPF [20]: nodes discover their link-local topology, flood the information to all other nodes, and build a topology map for the entire network. However, OLSR reduces communication and state requirements by selecting Multipoint Relays (MPRs) to operate as forwarding nodes. MPRs reduce the number of nodes that participate in floods and allow nodes to only maintain state about links that involve at least one MPR. In the best case, MPRs make communication and state requirements independent of density. TBRPF is another proactive link-state protocol designed for MANETs [129]. Each node computes a source tree to generate paths to individual nodes based on available topology information and advertise a subset of the source tree to neighboring nodes. While OLSR optimizes communication by having nodes select neighbors as MPRs, TBRPF nodes elect themselves as forwarders if they are selected as a next-hop by neighboring nodes. TBRPF further improves communication by using differential updates that only reflect changes. Even with the optimizations, state and communication requirements are still $O(N^2)$ in the worst case. While per-node state requirements scale with the number of nodes in the average case, even these resource requirements are infeasible for typical sensornet applications.

Neighbor discovery is an important process for any routing protocol. The MANET working group is currently developing a Neighbor Hood Discovery Protocol (NHDP) designed explicitly for MANETs [19]. The NHDP protocol relies on periodic HELLO broadcasts to discover 1-hop and 2-hop neighbors. Transmitters may include sequence numbers to allow receiving nodes to compute unidirectional link success rates. Unfortunately, unidirectional link quality estimates are most relevant to the transmitter. Computing bidirectional link success rates using NHDP requires receivers to maintain state about all nodes that may transmit to them, limiting the in-degree of a node. Our routing protocol, however, utilizes link-layer acknowledgments to efficiently provide bidirectional success rates to the transmitter, eliminate state requirements at the receiver, and remove any bounds on in-degree.

8.6.3 Sensornets

Many sensornet applications involve a small number of known destinations (often one) to collect sensor data. For this reason, sensornet routing protocols often take a distance-vector approach because the communication and state complexity mostly scales with the number of destinations. MintRoute is a distance-vector routing protocol designed for sensornets [186]. MintRoute is similar to RIP in that it is destination initiated and nodes maintain route information only for destinations that advertise themselves. MintRoute differs from RIP in that MintRoute incorporates link success rates into a minimum expected transmissions (MTX) cost metric, rather than simple hop counts. Unlike MANET protocols, MintRoute assumes a relatively stable topology that allows nodes to generate link success rate estimates over time. However, like NHDP, nodes must maintain state about all neighbors to compute bidirectional success rates. Our routing protocol is similar to MintRoute in the default route construction, but performs link-estimation using link-layer acknowledgments so that routers only need to maintain state for their default routes.

Other distance-vector routing protocols utilize link quality indicators provided by the link-layer. MultihopLQI is a distance-vector routing protocol similar to MintRoute but uses a LQI metric computed from the radio's chip correlation metric [140]. Drain is a similar protocol but utilizes RSSI information [169]. Both incorporate physical-layer information into the path cost. Because these physical-layer metrics are computed for each received message, the routing protocol can be more agile. Direct estimation of link success rates are more representative, but take more time and state to compute.

The relative tradeoffs between physical-layer information and sampling link success rates led to a proposed link estimator that utilized both [57]. Physical-layer information is used to indicate if a link is above or below a usable threshold, while link success rates are used to select a specific route from the table. In deciding what neighbors to evict, the link estimator can also ask the network layer to compare between two neighbors and the routing layer can also pin a neighbors in the table. All of the interfaces convey information using only a single bit. Our routing protocol uses similar mechanisms, but differs slightly in how those mechanisms are presented. Rather than a single bit for physical-layer information, our link layer allows the

router to directly compare full LQI and RSSI values to choose neighbors that are likely to have a higher quality links.

There is a significant body of work that focuses on minimizing resource requirements when shortest-path, any-to-any communication is required. With geographic routing, nodes are assigned coordinates based on their physical location within the network [93, 100, 101]. Using greedy algorithms, nodes can forward a datagram by only maintaining state about their neighbors and forwarding the datagram to the neighbor that minimize the distance from the destination. However, because the geographic coordinate space does not reflect the underlying connectivity, greedy forwarding can be inefficient when choosing a link with high loss rates. Worse yet, such protocols can fail when no neighbor provides a smaller distance than the current node. Routing protocols based on virtual coordinates attempt to address the disconnect between physical coordinates and radio connectivity by dynamically generating and maintaining a virtual coordinate space [58, 111, 122, 147]. Some protocols, however, still do not generate a complete coordinate space and must resort to recovery algorithms (e.g. flooding) when the greedy algorithm fails. Hierarchical routing protocols generate a spanning tree and assign names to nodes based on their location in the tree [39, 131, 172].

All coordinate-based routing protocols embed location information in the names of nodes, binding the node name to the routing topology. As a result, node names must be dynamic and must be changed whenever the routing topology changes. Before nodes can send to a specific destination, nodes must first lookup the destination's location. Any environmental changes or node mobility that may cause the destination's location to change requires additional lookups. In contrast, our baseline routing protocol does not couple the node name with the routing structure or underlying connectivity.

8.7 Summary

In this chapter, we presented a baseline routing protocol designed for typical sensornet constraints and workloads. The baseline routing protocol concentrates routing state at border routers to minimize resource requirements among sensornet nodes. By only maintaining state for a fixed set of potential default

routes to the nearest border router, our baseline routing protocol never initiates floods, requires small and constant state, and supports local recovery. Our protocol only requires $O(1)$ per-node routing state and $O(N)$ network-wide communication costs. In contrast, most existing protocols require in $O(N)$ state and $O(N^2)$ communication requirements with others having even worse scaling characteristics. Our protocol achieves better scaling characteristics by sacrificing routing stretch. However, our protocol maintains optimal paths for the expected common case of communicating through a border router. To further reduce costs, our protocol piggybacks all routing information on ambient traffic and uses that traffic to generate link quality estimates, reducing the need to generate any additional traffic.

We have finally developed a complete IPv6 network layer for sensornets that includes configuration and management, forwarding, and routing. Using this architecture and the mechanisms that implement it, the network layer can provide reachability with high best-effort datagram delivery between a sensornet node and any other IP device (e.g., nodes within the sensornet and traditional IP devices outside). In the following chapter we provide a brief discussion of transport-layer protocols, respecting that UDP and TCP are the most dominant and widely-deployed today. Chapter 9 provides an empirical analysis of the entire a production-quality implementation in a real application deployment. Through the evaluation, we demonstrate the efficiency of the routing architecture and all other networking components and show that it can outperform existing systems that do not adhere to any architecture or standard.

Chapter 9

Evaluation

In this chapter, we demonstrate what is possible to achieve with an IPv6-based network architecture for sensornets. To evaluate our IPv6 architecture for sensornets, we implement all of the functionality described in this dissertation as well as RFC-compliant UDP and TCP for transport protocols above. It is hard to ignore the ubiquity of UDP and TCP, and doing so provides end-to-end interoperability with existing IP devices. But because we've constrained ourselves to the IPv6 architecture and maintained high best-effort datagram delivery with low latency, implementing RFC-compliant UDP and TCP is straightforward.

We evaluate both high-level systems aspects of our IPv6-based network architecture and in-depth aspects of critical details. Our production-quality implementation is built using TinyOS 2.x [126] on the TelosB platform [139]. The TelosB consists of a 16-bit TI MSP430 MCU with 48KB ROM and 10KB RAM and a 2.4 GHz, 250 kbps TI CC2420 IEEE 802.15.4 radio. We use AES-128 authentication and encryption (CCM, ENC-MIC-32), as this is important in any production deployment.

We begin by showing both ROM and RAM requirements for our implementation. We then conduct a detailed power analysis of link-layer communication primitives and demonstrate general properties of the link. We pay careful attention to the details of the link layer, as it is the link's capabilities that establish the baseline for the network layer. Building on the link layer, we also conduct a power analysis of the network and application layers, analyzing the cost of forming and maintaining a network as well as the cost of supporting

Component	ROM	RAM
CC2420 Driver	3149	272
802.15.4 Encryption	1194	101
Media Access Control	330	9
Media Management Control	1348	20
6LoWPAN + IPv6	2550	0
Checksums	134	0
SLAAC	216	32
DHCPv6 Client	212	3
DHCPv6 Proxy	104	0
ICMPv6	522	0
Unicast Forwarder	1158	315
Multicast Forwarder	352	4
Message Buffers	0	2048
Router	2050	64
UDP	450	6
TCP	1674	48

Table 9.1: **ROM and RAM Requirements for Communication Components.**

typical application workloads. Using a real-world application, we validate the power model and show that it outperforms other systems that do not adhere to any architecture or standard.

9.1 Memory Footprint

A complete, production-quality implementation that provides all of the functionality described in this dissertation and support for one UDP socket and one TCP connection consumes 24,038 bytes of ROM and 3,598 bytes of RAM. These numbers include the entire run-time required to support the IPv6 network stack (e.g., OS-level services). The breakdown for communication-specific components is shown in Table 9.1 and are similar to uIP [40]. The memory footprint easily fits within our chosen hardware platform and also supports less capable microcontrollers depending on how much memory is required for the application.

9.2 Link-Layer Energy Cost

The communication primitives at the link layer are: channel sample, overhear, receive, and transmit. For each primitive, we plot in Figure 9.4 the instantaneous current draw over time for the entire TelosB

platform. Each primitive requires the MCU to wakeup, enable the radio's voltage regulator and crystal oscillator, set radio configuration parameters, and enable the receiver. The receiver is enabled for transmissions to perform CSMA. All of this takes about 2ms, the effects of which can be seen in every profile.

The cost of a channel sample determines the listen cost to support a given communication latency. Figure 9.1(a) displays a channel sample, with the receiver only enabled for 640us. From the CC2420 datasheet, it takes 192us to enable the receiver, another 128us before the RSSI measurement becomes valid, and two TX-RX turnaround times (192us each) due to the CC2420's receive-after-transmit implementation. Slightly shorter times are seen in practice because datasheets represent the worst-case. Significant improvements can be made if the radio's design supported continuous transmissions.

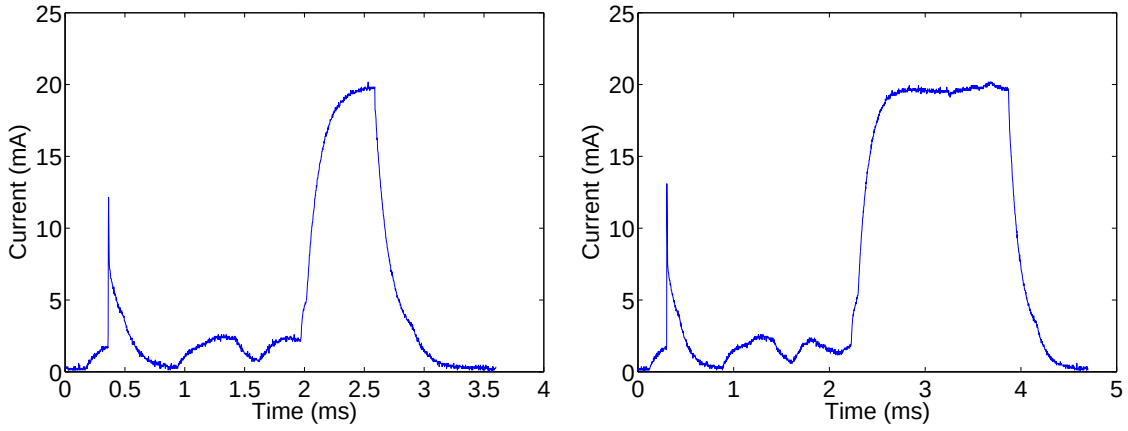
Figure 9.1(b) displays the profile of overhearing, which involves receiving a chirp frame and processing the destination address. Figures 9.2(a) and 9.2(b) display the profiles for transmitting 26 and 127 byte frames. These sizes represent the smallest and largest frame sizes supported by the IPv6 stack. The receive profile includes receiving a chirp and data frame and transmitting an ack frame. The transmit profile includes a CCA check, data frame transmission and reception of an ack. Figures 9.3(a) and 9.3(b) display the profile for transmitting a data frame with a chirp length of 62.5ms and 125ms, respectively. Figures 9.4(a) and 9.4(b) display the profiles for receiving 26 and 127 byte frames.

Using empirical measurements for low-level communication primitives, we build a model that accurately reflects average power draw of the entire platform. At a high level, average power draw P_{total} consists of three basic components: listen P_{listen}, receive P_{rx}, and transmit P_{tx}.

$$P_{total} = P_{listen} + P_{rx} + P_{tx}$$

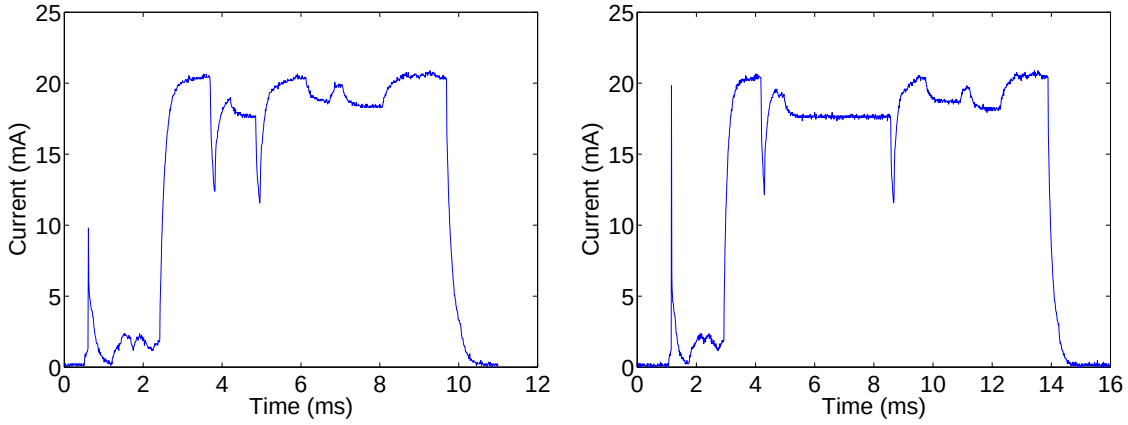
Listen represents the baseline power draw for a node and places a upper bound on the node's lifetime. Included in this baseline is the sleep power. The average power draw for listening can be determined by the sleep power draw P_{sleep} and the product of the channel sample frequency f_{sample} and channel sample energy cost E_{sample}.

$$P_{listen} = P_{sleep} + f_{sample}E_{sample}$$



(a) Channel Sample (b) Overhear

Figure 9.1: **Current Draw Profile for Channel Sampling and Overhearing.** The profiles only display the time period during the actual listening action. In reality, the radio is off for relatively long periods before and after listening.

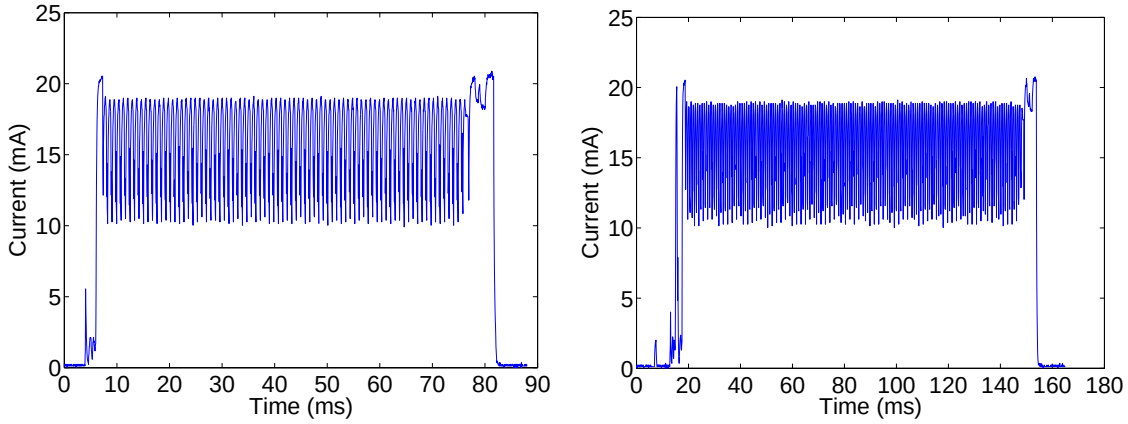


(a) Transmit Len = 26 bytes (b) Transmit Len = 127 bytes

Figure 9.2: **Current Draw Profile for Transmitting.** The transmissions do not include a wakeup signal. (a) shows a transmission of a 26-byte frame and (b) shows a transmission of a 127-byte frame.

Receive represents the average power draw due to receiving data frames. The receive cost is independent of the channel sample period and whether the frames are unicast or broadcast. The receive cost can be modeled as the product of the reception frequency f_{rx} and the reception energy cost E_{rx}. To simplify the model, we always assume worst-case frame sizes of 127 bytes.

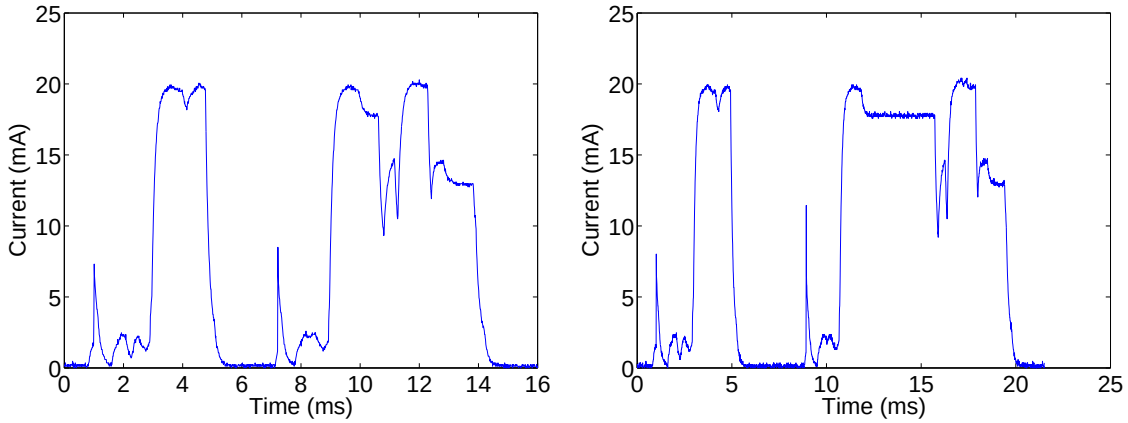
$$P_{rx} = f_{rx} E_{rx}$$



(a) Transmit Chirp = 62.5ms

(b) Transmit Chirp = 125ms

Figure 9.3: **Current Draw Profile for Transmitting with a Chirp Signal.** (a) shows a transmission with a 62.5 ms chirp signal and (b) shows a transmission of a 125 ms chirp signal. Both transmit a 26-byte data frame.

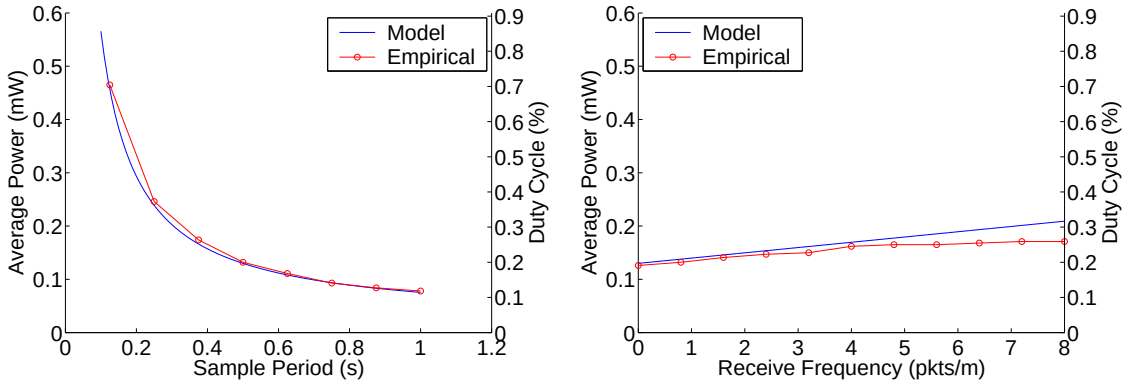


(a) Receive Len = 26 bytes

(b) Receive Len = 127 bytes

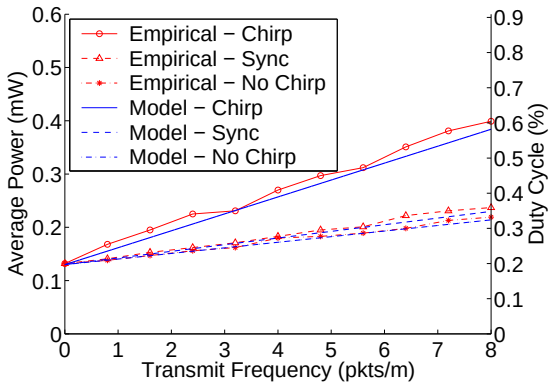
Figure 9.4: **Current Draw Profile for Receiving.** (a) shows a transmission with a 26-byte data frame and (b) shows a transmission with a 127-byte data frame. Both only incur the cost of receiving a chirp and data frame, significantly reducing the energy cost and making the energy cost independent of the channel sample period.

Transmit represents the average power draw due to transmitting data frames. The transmit cost is highly dependent on the chirp duration. When transmitting a broadcast frame or to neighbors that don't have associated synchronization information, the chirp duration is equal to the channel sample period. When synchronization information is known, the chirp duration is the synchronization guard time which increases



(a) Listen

(b) Receive



(c) Transmit

Figure 9.5: **Average Power Draw for Listening, Receiving, and Transmitting**. Figure (a) shows the average power draw when only listening. Figures (b) and (c) show the average power draw when receiving and transmitting relative to the packet rates and assumes $T_{sample} = 0.5$s.

over time since the last received ack from that neighbor. In this implementation, the minimum guard time is 2ms and we assume a frequency tolerance $f_\Delta = \pm 20$ppm. Note that the minimum of guard time and channel sample period is used to determine the chirp duration. We simplify the power model for transmissions by accounting for the frequency of broadcast messages f_{txb}, the frequency of unicast messages f_{txu}, and assume that a node transmits all messages to duty-cycled nodes.

$$P_{txb} = f_{txb}\left(E_{tx} + E_{cb} + E_{cd}\frac{1}{f_{sample}}\right) \tag{9.1}$$

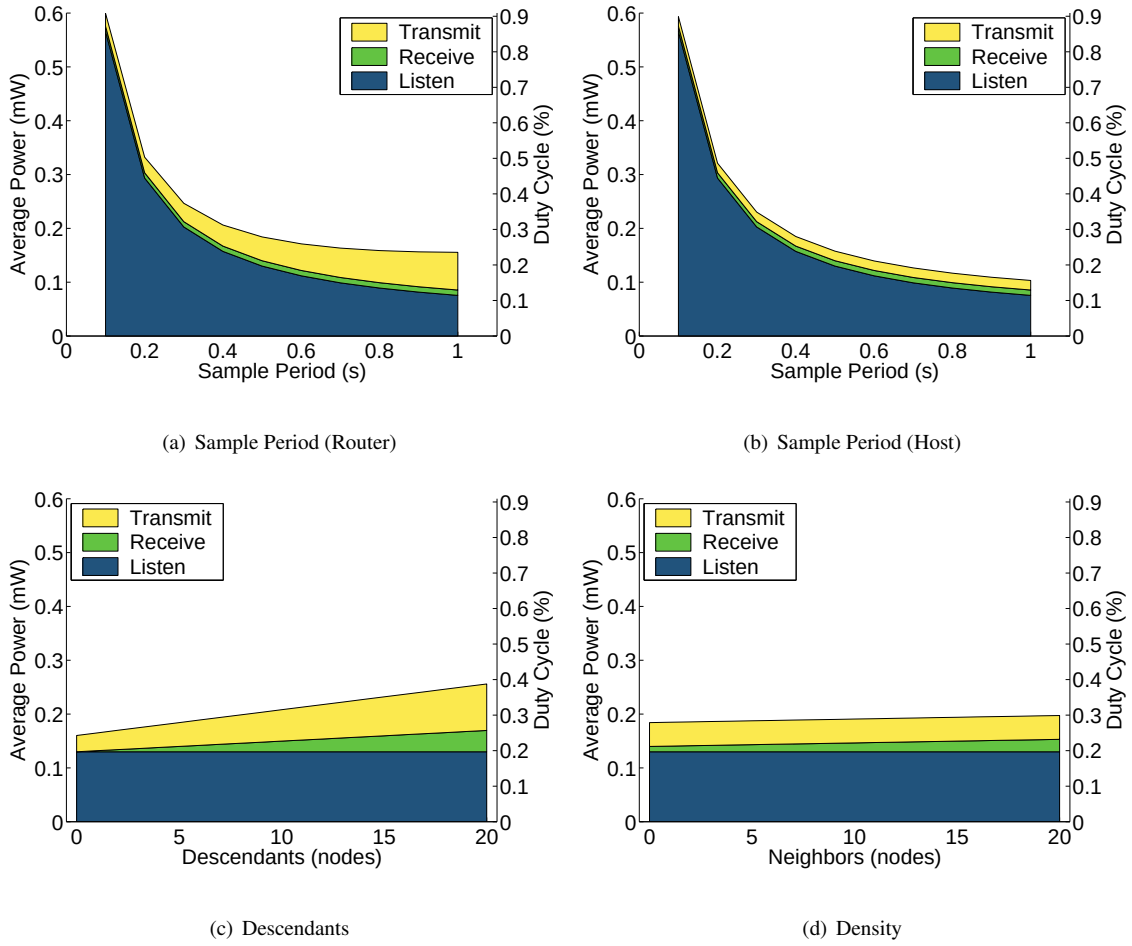
$$P_{txu} = f_{txu}\left(E_{tx} + E_{cb} + E_{cd}\left(2 + \frac{f_\Delta}{f_{txu}}\right)\right) \tag{9.2}$$

We verify the models for listen, receive, and transmit by measuring the average power draw of a node only doing those respective operations. As shown in Figure 9.5, the model predicts the measured values to within 2% on average. These results show significantly less average power draw than other results using the CC2420 radio. X-MAC has a channel sample cost 30 times larger because it inserts acks into the preamble stream and does not support synchronization [13]. Current LPL implementation variants in TinyOS 2.x are not optimized to minimize gaps between transmissions, resulting in a minimum channel sample duration that is 10 times larger [126]. SCP-MAC presented similar but limited results for basic primitives for the MicaZ platform [191].

9.3 Network-Layer Energy Cost

To model the average power draw for a node maintaining network connectivity, we need to determine the frequency of transmitting broadcast and unicast messages as well as receiving messages. The only control messages generated by the network layer are the ICMPv6 RA messages and Record Route messages. While RA messages are broadcast messages sent using Trickle, we simplify the model by assuming $f_{ra} = \frac{\tau_h}{2}$ where τ_h is the maximum Trickle period. RRO messages are unicast periodically, with frequency f_{rr}. We assume no suppression for both RA and RRO messages. Finally, we assume that most traffic will follow default routes and that synchronization information is known when sending unicast messages.

The transmission and reception frequency depend on a number of parameters. The frequency of forwarding messages depends on how many nodes route through a given node, and is determined by the *descendants* D of a node. The number of neighboring routers N affects the receive cost due to the RA reception frequency f_{ra}. A node may be configured as a router or host-only, which determines whether or not a node forwards messages and sends RA messages. Using the model, we analyze the average power draw



(a) Sample Period (Router)

(b) Sample Period (Host)

(c) Descendants

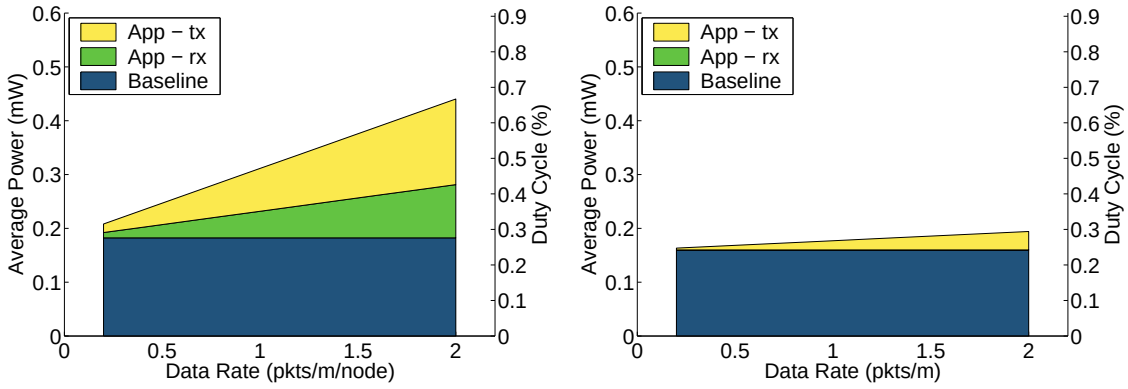
(d) Density

Figure 9.6: **Network Maintenance.** Each figure shows the power draw of listen, receive, and transmit. Figure (a) shows power draw of a router while varying T_{sample} with $D = 5$ and $N = 5$. Figure (b) shows power draw for a host-only node while varying T_{sample} with $D = 0$ and $N = 5$. Figure (c) shows power draw for a router while varying D with $T_{sample} = 0.5$s and $N = 5$. Figure (d) shows power draw for a router while varying N with $T_{sample} = 0.5$s and $D = 5$.

for host-only and router nodes. From these observations, we have the following relationships:

$$\begin{aligned} f_{rx} &= N f_{ra} + D f_{rr} \\ f_{txb} &= f_{ra} \\ f_{txu} &= (1 + D) f_{rr} \end{aligned}$$

Using the model, we analyze the average power draw of a node doing nothing except maintaining network connectivity. We plot the average power draw for routers and hosts (hosts neither neither forward



(a) Router

(b) Host-Only

Figure 9.7: **Application Power Consumption.** Figures (a) and (b) show average power draw for a router and host, respectively, while varying f_{app} with $T_{sample} = 0.5$s, $D = 5$, and $N = 5$.

datagrams nor send RA messages) as a function of channel sample period, descendants, and node density in Figure 9.6. Figure 9.6(a) shows the average power draw of a router relative to the channel sample period, achieving a duty-cycle of about 0.23% when $T_{sample} = 1$s. As the channel sample period increases, the listen cost decreases and the receive cost remains constant. The transmit cost increases because the RA broadcast scales with the channel sample period. RA messages are required for router discovery and no synchronization information is assumed. A host-only node benefits greatly by not needing to transmit RA messages, achieving a duty-cycle of about 0.16% when $T_{sample} = 1$s, as shown in Figure 9.6(b). Figure 9.6(c) shows the average power draw of a router relative to D. Both P_{rx} and P_{tx} increase linearly, but P_{listen} still accounts for the majority of P_{total} even when $D = 20$. Figure 9.6(d) shows the average power draw of a router relative to the number of neighbors. Both P_{rx} and P_{tx} increase linearly, but at a much slower rate than D.

9.4 Application-Layer Energy Cost

To evaluate the average power draw of a node in an application environment, we consider the data-collection scenario typical to many sensornet applications. Both host-only and router nodes source UDP datagrams with a fixed period to an external data server through border routers. The two parameters that affect

transmission and reception frequency are: (i) the frequency of application datagrams sourced by individual nodes f_{app} and (ii) the number of collection flows D that a node is forwarding (for host-only nodes, this is zero). We augment the model to incorporate application-level traffic and plot average power draw relative to the datagram generation rate in Figure 9.7.

$$\begin{aligned} f_{rx} &= N f_{ra} + D\,(f_{rr} + f_{app}) \\ f_{txb} &= f_{ra} \\ f_{txu} &= (1 + D)\,(f_{rr} + f_{app}) \end{aligned}$$

Figures 9.7(a) and 9.7(b) display the average power draw of a router and host-only node. In both cases, the network functions account for a significant fraction of the average power draw and listening forms the largest component. For a host-only node, the application accounts for a small fraction of the average power draw even when $f_{app} > 2$ packets per minute. For the router, application traffic accounts for more than half of the average power draw when $f_{app} > 1.5$ packets per minute.

9.5 A Real-World Application

To validate the power model, we use a real-world home monitoring application. A snapshot of the deployment layout is shown in Figure 9.8. The application consists of 15 nodes deployed in refrigerators, solar power inverters, outdoors, and indoors in or near the intended sense point. A border router connects the sensornet to an Ethernet LAN. Sensornet nodes report data to a data server connected to the Ethernet LAN using UDP/IPv6 datagrams. A DSL/Ethernet router connects the data server and sensornet nodes to the Internet. Arbitrary nodes within the Internet can ping nodes and query the status of sensors using both IPv4 and IPv6.

Using simple sensors, the home monitoring system allows the home owner to be more aware of various items within the home. For example, a magnetic reed switch connected directly to a sensornet node can sense the opening and closing of doors. A temperature sensor combined with a power meter attached to

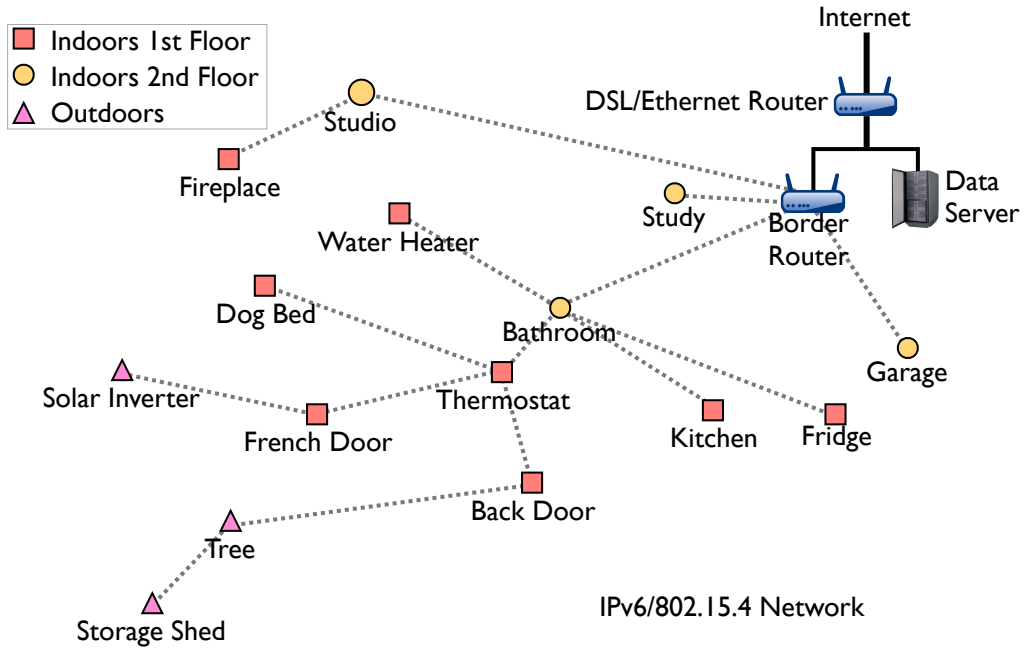
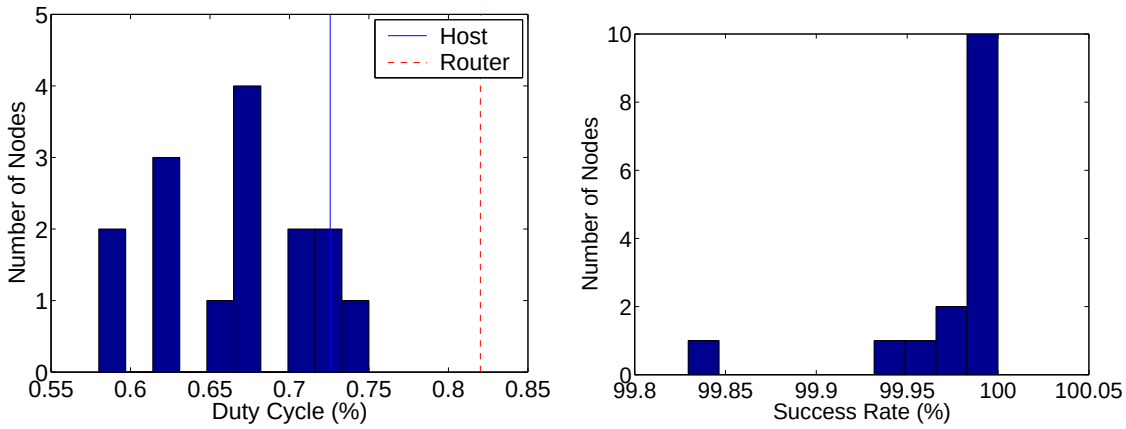


Figure 9.8: **Real-World Application**. The home monitoring application consists of 15 nodes, monitoring the state of various areas and components of a home. The sensornet is connected to an Ethernet LAN through a border router. Nodes report data to a data server connected to the Ethernet LAN using UDP/IPv6 datagrams. A DSL/Ethernet router connects the data server and sensornet nodes to the Internet.

the refrigerator can provide enough information to compute its efficiency. Sensors attached to solar power inverters provide information of how much energy solar panels are generating.

Nodes periodically report environmental data (e.g., temperature and humidity) as well as a variety of sensors attached to each node. Each node also reports network statistics, including uptime, active time for the radio and MCU, number of datagrams sourced, and routing topology. Application traffic was about one datagram per minute per node, each datagram nearly filling a full 802.15.4 frame.

We logged network statistics over a continuous 4 week period. The routing topology consisted of 4 nodes within 1-hop of the border router, the remaining nodes being 2 to 5 hops away. The link was configured with $T_{sample} = 0.125$s. Using this information, we compute the average duty-cycle for each node. As shown in Figure 9.9(a), all nodes had an average duty-cycle between 0.59% and 0.74%. Because the power model does not account for suppression mechanisms, the empirical numbers are *better* than predicted. We achieved



(a) Duty Cycle

(b) Success Rate

Figure 9.9: **Real-World Application**. Figures (a) and (b) show data from a deployed home monitoring application over a continuous 4-week period, where $T_{sample} = 0.125$s. Figure (a) shows a histogram of average duty-cycles over all three weeks with the model's prediction for leaf and router nodes. Figure (b) shows a histogram of the packet success rate by the server.

this with a high success rate for datagrams delivered to the server. All but one node had a success rate above 99.94% with an aggregate success rate of 99.98%, as shown in 9.9(b).

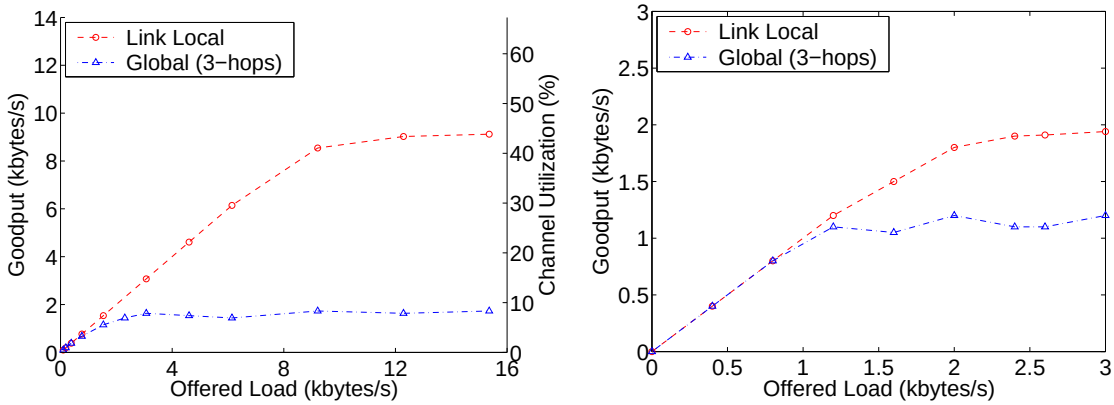
We compare results from our application deployment to published data from prior sensornet deployments in Table 9.2, showing that we were able to achieve better performance in average duty-cycle, per-hop latency, and data reception rate with a higher traffic load. NanoStack, another 6LoWPAN-based IPv6 stack, uses the standard IEEE 802.15.4 MAC and does not support duty-cycled operation for forwarding nodes [154]. Similarly, ZigBee [195] does not support duty-cycled operation for forwarding nodes and also does not provide end-to-end IP interoperability. uIP [40, 42] showed the feasibility of IP in sensornets, but existing work focused more on CPU performance rather than overall system energy consumption and networking performance in a typical sensornet setting.

9.6 Goodput and Latency

Using the same application deployment, we also evaluate the goodput and latency performance of link-local and global communication over UDP and TCP. Streaming optimizations allowed our implementa-

Deployment	Year	Report Period (m)	Duty Cycle	Latency (s)	Reception Rate
GDI [161]	2003	20	2.2%	0.54-1.085	28%
Redwoods [170]	2004	5	1.3%	300	49%
FireWxNet [67]	2005	15	6.7%	900	40%
WiSe [162]	2006	30	1.6%	60	33%
Dozer [14]	2007	2	1.67%	15	98.8%
SensorScope [8]	2008	2	1.11%	120	95%
IPv6	2008	1	0.65%	0.125	99.98%

Table 9.2: **Performance of prior sensornet deployments.** *Report period is the time delay between data transmissions. Duty cycle is the fraction of time the radio spent in the active state. Worst-case per-hop latency is determined by the radio's wake period. The reception rate is the fraction of data received at the collection point. "IPv6" represents the home-monitoring application presented here.*



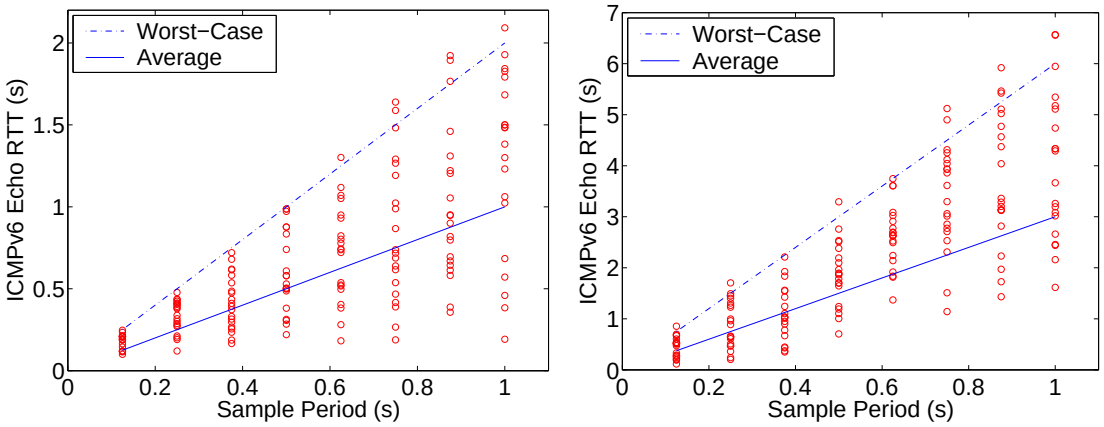
(a) UDP Goodput

(b) TCP Goodput

Figure 9.10: **Goodput.** Figures (a) and (b) shows goodput and channel utilization for UDP communication within link-local and global scope, respectively.

tion to achieve higher goodput than other duty-cycled solutions. Figure 9.10(a) shows the achievable goodput for UDP is 9 kbytes/s for link-local and 1.7 kbytes/s over three hops. However, the achievable goodput is still much lower than the theoretical 250 kbps, due to the use of AES-128 encryption and software-based acks. Figure 9.10(b) shows the achievable goodput over TCP is 1.9 kbytes/s for link-local and 1.2 kbytes/s over three hops, which is significantly lower than UDP because TCP requires communication in both directions.

Figures 9.11(a) and 9.11(b) shows that the expected communication latency measured with Ping is linear with T_{sample}. Note that the expected round-trip per-hop communication latency is half of the channel-sample period. With synchronization, communication latency depends on when the next channel sample



(a) Link Local Ping

(b) Global Ping

Figure 9.11: **Latency.** Figures (a) and (b) shows the distribution of round-trip times for ICMPv6 Echo Request/Reply messages over link-local and global scope, respectively.

occurs at the receiver. Per-hop latencies that are less than a second allows the network to support human time scales for configuration and management. Application events and alerts can also propagate quickly if needed.

9.7 The Cost of IP

We have shown that an IPv6-based network architecture can be implemented more efficiently than existing systems that do not adhere to any particular architecture or standard. The question remains: what is the cost of an IPv6-based network architecture? The adaptation layer requires 7-12 bytes for a typical UDP/IPv6 header. Putting this into perspective, the marginal cost of transmitting 1 byte is only 1.67uJ while the cost of transmitting a packet is 630uJ and an order of magnitude greater when sending chirps. Even so, some header overhead is required by any network architecture (protocol identifiers, end-to-end integrity checks, and application dispatch). RA messages require a few transmissions per hour, but similar mechanisms are necessary for configuration and routing in non-IP settings as well. Similar with periodic transmissions to a root. Despite this concern of transmission overhead, we have shown that the idle-listening cost still dominates in many cases.

Chapter 10

Conclusions

10.1 Architectural Retrospective

In the past decade, wireless sensor network research and Internet architecture, especially IPv6, have both progressed substantially. Thus, it makes sense to revisit the assumptions that have formed the basis of so much sensornet research. By constructing a high quality sensornet using IPv6 as a foundation, we find that the two areas of development are in fact highly complementary. Most of the unique requirements of sensornets are well served by the IPv6 architecture, in many cases better than by any efforts that were carried out without concern for any particular network standard. However, in several instances while the architecture was appropriate, the specific RFCs and implementations were not. Extensions had to be created to encompass particular needs, and these extensions are naturally supported by the systematic use of options.

Sensornet research has focused much more on network protocol algorithms and mechanisms, rather than on networking in the broader sense. The IPv6 forms of layering, addressing, header formats, configuration, management, routing, and forwarding provide the missing structure. And, in many cases the mechanisms developed in the sensornet space provide elegant solutions to problems that have long challenged conventional IETF approaches. Not only do we find a close analog to the structures for dissemination and

collection that are so common in sensornets, but Trickle mechanisms provide an elegant means of reaching consensus, responding quickly to changes, and becoming passive in the steady state.

In particular, we find that we need not forsake the many virtues of layering to meet the severe resource constraints of sensornet nodes. If anything, the focus obtained by a layered approach tends to produce better solutions than when many degrees of freedom are addressed simultaneously in an ad-hoc manner. We were able to obtain extremely low power consumption, small footprint, good throughput, low latency, and high reliability with a layered solution. However, the interfaces between the layers cannot be oblivious to the nature of the constraints and challenges, to provide enough expressiveness for the layers to cooperate effectively.

The sheer numbers of nodes, unattended use, and need for ease of configuration and management are well aligned with the mechanisms provided by ICMPv6. While many aspects of discovery and configuration in IPv4 relied on external layer 2 services like BOOTP and DHCP, these have been systematically incorporated into the IPv6 architecture with enhanced autoconf and discovery that are enabled by the larger, simpler namespace and multicast support. We find that ICMPv6 capabilities far exceed anything that has been proposed specifically for sensornets. Indeed, if sensornets are not to incorporate the IP architecture, much of this functionality will need to be reinvented for these networks to go into production.

However, most of these IPv6 solutions assume that all nodes are a single hop from a designated agent. They needed to be extended to service the multihop case. Perhaps the surprising result is that by treating each node as a router and hence defining a network architecture of overlapping link-local scopes, the existing IPv6 protocols can be naturally extended with simple options. Going in, it seemed that emulating the more common single broadcast domain over multiple hops would preserve more of the existing IPv6 mechanisms. Our experience was that we needed to have layer 2 versions of this same functionality to support the emulation, so it was much more natural to keep layer 2 extremely simple and utilize the routing capability that is by definition built in to layer 3.

While it remains unclear whether localized algorithms and in-network processing will become prevalent, rather than relatively straightforward data collection, logging, alarms, and configuration, the use of

an IPv6 architecture as we have formulated it is entirely consistent with the desire to support such algorithms. By constructing the architecture as overlapping link-local scopes that provide global routing, localized algorithms are simply implemented as UDP datagrams to various well-defined multicast addresses or to unicast addresses.

It also remains unclear whether naming will become primarily data-centric and if so whether this will need to be addressed within the network stack or by application overlays. But regardless, by providing a clean IPv6 architecture for sensornets, this debate becomes essentially the same as the data-centric debate in the rest of the Internet. What we did find was that the large, simple address space, use of multicast groups defined in terms of that address space, and regularity of the architecture actually made the resource constrained solution easier than in the IPv4 setting. Compression and elision of header information where it can be reconstructed from the layer 2 header in the presence of some simple assumptions of shared context becomes straightforward and efficient.

While sensornets are indeed more application specific than traditional networks, we find that the networking mechanisms and the architecture are not. What is application specific is how the use of those mechanisms is optimized and how the network is organized within that architectural framework. Whether it is deterministic scheduling of communication or streaming data to reduce overhead, simple mechanisms that support a wide variety of use can exploit the application specific structure. However, the mechanisms also provide a more general safety net when the network's behavior is not following the specific pattern. For example, it can be fully operational at low power with always-on behavior while nodes communicate to determine that schedule. Or, it can fall back to sample listening when the schedule drifts.

The relatively simple application characteristics that are typical of sensornets can be exploited to simplify protocols and their implementations to achieve small resource demand. Such solutions may be suboptimal for the arbitrary any-to-any transfers that are the primary design point for conventional networks.

10.2 Research Impact

Broad architectural issues are best addressed in agreement with the community. Throughout this work, we have been actively involved in standards organizations such as the IETF and the IEEE. We are currently working with the IEEE 802.15.4e working group and have proposed many of the ideas from Chapter 4 for inclusion in the IEEE 802.15.4 standard. The 6LoWPAN working group within the IETF published a proposed standard for communicating IPv6 datagrams within IEEE 802.15.4 frames. Our work initial work led to the header formats specified in RFC 4944, which incorporates the adaptation-layer header formats presented in Chapter 5. However, there still remains significant work within 6LoWPAN to complete an IPv6-based network architecture and we plan to continue the process. Our header compression mechanisms presented in Chapter 5 are serving as the basis for header compression improvements to RFC 4944. Furthermore, our work in Chapter 6 is significantly influencing the design and specification of Neighbor Discovery for 6LoWPAN networks.

Even though the 6LoWPAN working group is far from complete, its results are already propagating outside the IETF. Other standards organizations, such as ISA [84], have adopted the 6LoWPAN header format and an IPv6-based network architecture for use in industrial automation applications. Academic efforts have already implemented an IPv6-based network stack based on RFC 4944 [27, 68]. Numerous commercial entities have also done the same, including Hitachi [76], Jennic [88], Sensinode [154], and Sun Microsystems [160].

A variant of the implementation that we described and evaluated in this dissertation serves as the core technology for Arch Rock Corporation. Our production-quality implementation has been in use for over a year in real customer deployments. We have also used the same technology in academic settings. At the University of California, Berkeley, the technology was used to teach an undergraduate class titled: “The Internet of Everyday Things” [26]. By supporting familiar IP-based communication mechanisms, students were able focus on building web-based applications using a broad set of existing IP-based tools, rather than on the intricacies of a non-standard network stack and their integration into traditional networks.

10.3 Last Words

Supporting IP provides invaluable interoperability with existing IP devices as well as being able to utilize the broad body of existing IP tools when connecting sensornets to other IP networks (i.e., firewalls, proxies, caches, etc.). We have shown that an IPv6-based network architecture can be implemented more efficiently in sensornets than what has been demonstrated to date without adherence to any particular standard. Our implementation was able to achieve an average duty-cycle of 0.65%, average per-hop latency of 62ms, and a data reception rate of 99.98% over a period of 4 weeks in a real-world home-monitoring application where each node generates one application packet per minute.

The presence of broad-based IPv6 support in the same technological context as prior more limited, more idiosyncratic solutions means that we can re-examine a much broader set of research questions. The question of in-network processing, aggregation, compression, and query processing are not constrained by the underlying networking capability. It is fairly straightforward to implement any of these. Their effectiveness can be strongly separated from questions of whether they should be implemented as application level overlays or somehow more deeply integrated into the network stack. Similarly, many of the long-standing sensornet questions can be examined in a much more general setting, and if the answer is "open a TCP connection" or "send a UDP datagram to where it needs to go", that answer is acceptable as well.

On the other hand, a number of new questions emerge about how the Internet architecture should change or evolve now that it is supporting a new class of applications. It is hard to ignore the ubiquity of UDP and TCP - they must be implemented to provide end-to-end interoperability with existing IP devices. However, many suggest that neither UDP or TCP transports are quite right for periodic readings, reports, and configuration actions. Many of the industrial protocols have encountered these tensions in conventional wired links, but they take on a new flavor in the low-power wireless setting. Another example is the presence of groupcast, especially when actuation is involved. The needs of building automation and home automation, for example, appear to be quite different from audio and video streaming where IP multicast is common. Heterogeneity in deployments becomes much more natural, since IP routing by definition supports crossing

a variety of links. In these or a wide variety of other studies, the IPv6 architecture provides a framework to develop specific solutions and the mechanisms for doing so effectively, even with severe resource constraints, without each study needing to develop yet another MAC, routing protocol, and transport. Thus, it would seem that the two lines of development are not just technically complementary, combining may accelerate progress in both.

Bibliography

[1] ITU-T Recommendation I.363.5, B-ISDN ATM Adaptation Layer (AAL) Type 5 Specification, August 1996.

[2] J. Abley, P. Savola, and G. Neville-Neil. Deprecation of Type 0 Routing Headers in IPv6. RFC 5095 (Proposed Standard), December 2007.

[3] R. Albrightson, J. Garcia-Luna-Aceves, and J. Boyle. EIGRP-A Fast Routing Protocol Based on Distance Vectors. 1994.

[4] G. Armitage, P. Schulter, M. Jork, and G. Harter. IPv6 over Non-Broadcast Multiple Access (NBMA) networks. RFC 2491 (Proposed Standard), January 1999.

[5] ATMForum. LANEmulation over ATMVersion-2 LUNI Specification, December 1995.

[6] Tuomas Aura. Cryptographically Generated Addresses (CGA). RFC 3972 (Proposed Standard), March 2005. Updated by RFCs 4581, 4982.

[7] Hari Balakrishnan, Srinivasan Seshan, Elan Amir, and Randy H. Katz. Improving TCP/IP performance over wireless networks. In *MobiCom '95: Proceedings of the 1st annual international conference on Mobile computing and networking*, pages 2–11, New York, NY, USA, 1995. ACM.

[8] Guillermo Barrenetxea, François Ingelrest, Gunnar Schaefer, Martin Vetterli, Olivier Couach, and Marc Parlange. SensorScope: Out-of-the-Box Environmental Monitoring. In *IPSN '08: Proceedings*

of the 2008 International Conference on Information Processing in Sensor Networks (ipsn 2008), pages 332–343, Washington, DC, USA, 2008. IEEE Computer Society.

[9] C.G. Bell, R. Chen, and S. Rege. Effect of Technology on Near Term Computer Structures. *Computer*, 5(2):29–38, February 1972.

[10] Jeremy Bentham. *TCP/IP lean: web servers for embedded systems*. CMP Media, Inc., USA, 2000.

[11] C. Bormann, C. Burmeister, M. Degermark, H. Fukushima, H. Hannu, L-E. Jonsson, R. Hakenberg, T. Koren, K. Le, Z. Liu, A. Martensson, A. Miyazaki, K. Svanbro, T. Wiebke, T. Yoshimura, and H. Zheng. RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed. RFC 3095 (Proposed Standard), July 2001. Updated by RFCs 3759, 4815.

[12] R. Braden. Requirements for Internet Hosts - Communication Layers. RFC 1122 (Standard), October 1989. Updated by RFCs 1349, 4379.

[13] Michael Buettner, Gary V. Yee, Eric Anderson, and Richard Han. X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks. In *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 307–320, New York, NY, USA, 2006. ACM Press.

[14] Nicolas Burri, Pascal von Rickenbach, and Roger Wattenhofer. Dozer: ultra-low power data gathering in sensor networks. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 450–459, New York, NY, USA, 2007. ACM.

[15] Ian Chakeres and Charlie Perkins. Dynamic MANET On-demand (DYMO) Routing. draft-ietf-manet-dymo-13 (work in progress), April 2008.

[16] S. Cheshire, B. Aboba, and E. Guttman. Dynamic Configuration of IPv4 Link-Local Addresses. RFC 3927 (Proposed Standard), May 2005.

[17] J. Chu and V. Kashyap. Transmission of IP over InfiniBand (IPoIB). RFC 4391 (Proposed Standard), April 2006.

[18] T. Clausen and P. Jacquet. Optimized Link State Routing Protocol (OLSR). RFC 3626 (Experimental), October 2003.

[19] Thomas Clausen, Christopher Dearlove, and Justin W. Dean. MANET Neighborhood Discovery Protocol (NHDP). draft-ietf-manet-nhdp-07 (work in progress, July 2008.

[20] R. Coltun, D. Ferguson, and J. Moy. OSPF for IPv6. RFC 2740 (Proposed Standard), December 1999.

[21] A. Conta, S. Deering, and M. Gupta. Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification. RFC 4443 (Draft Standard), March 2006. Updated by RFC 4884.

[22] Douglas S. J. De Couto, Daniel Aguayo, John Bicket, and Robert Morris. A high-throughput path metric for multi-hop wireless routing. In *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 134–146, New York, NY, USA, 2003. ACM.

[23] M. Crawford. Transmission of IPv6 Packets over FDDI Networks. RFC 2467 (Proposed Standard), December 1998.

[24] M. Crawford. Router Renumbering for IPv6. RFC 2894 (Proposed Standard), August 2000.

[25] M. Crawford, T. Narten, and S. Thomas. Transmission of IPv6 Packets over Token Ring Networks. RFC 2470 (Proposed Standard), December 1998.

[26] David Culler, Jonathan Hui, and Prabal Dutta. The Internet of Everyday Things. http://inst.eecs.berkeley.edu/~cs194-5/sp08/.

[27] Stephen Dawson-Haggerty. b6lowpan. http://tinyos.cvs.sourceforge.net/tinyos/tinyos-2.x-contrib/berkeley/b6lowpan/.

[28] S. Deering. ICMP Router Discovery Messages. RFC 1256 (Proposed Standard), September 1991.

[29] S. Deering, D.L. Estrin, D. Farinacci, V. Jacobson, Ching-Gung Liu, and Liming Wei. The pim architecture for wide-area multicast routing. *Networking, IEEE/ACM Transactions on*, 4(2):153–162, Apr 1996.

[30] S. Deering, B. Haberman, T. Jinmei, E. Nordmark, and B. Zill. IPv6 Scoped Address Architecture. RFC 4007 (Proposed Standard), March 2005.

[31] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard), December 1998.

[32] M. Degermark, B. Nordgren, and S. Pink. IP Header Compression. RFC 2507 (Proposed Standard), February 1999.

[33] Alan Demers, Dan Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry. Epidemic algorithms for replicated database maintenance. In *PODC '87: Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*, pages 1–12, New York, NY, USA, 1987. ACM.

[34] C. DeSanti, C. Carlson, and R. Nixon. Transmission of IPv6, IPv4, and Address Resolution Protocol (ARP) Packets over Fibre Channel. RFC 4338 (Proposed Standard), January 2006.

[35] R. Draves and D. Thaler. Default Router Preferences and More-Specific Routes. RFC 4191 (Proposed Standard), November 2005.

[36] R. Droms. Dynamic Host Configuration Protocol. RFC 2131 (Draft Standard), March 1997. Updated by RFCs 3396, 4361.

[37] R. Droms. Stateless Dynamic Host Configuration Protocol (DHCP) Service for IPv6. RFC 3736 (Proposed Standard), April 2004.

[38] R. Droms, J. Bound, B. Volz, T. Lemon, C. Perkins, and M. Carney. Dynamic Host Configuration Protocol for IPv6 (DHCPv6). RFC 3315 (Proposed Standard), July 2003. Updated by RFC 4361.

[39] Shu Du, Ahamed Khan, Santashil PalChaudhuri, Ansley Post, Amit Kumar Saha, Peter Druschel, David B. Johnson, and Rudolf Riedi. Safari: A self-organizing, hierarchical architecture for scalable ad hoc networking. *Ad Hoc Netw.*, 6(4):485–507, 2008.

[40] Adam Dunkels. Full TCP/IP for 8-bit architectures. In *MobiSys '03: Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 85–98, New York, NY, USA, 2003. ACM.

[41] Adam Dunkels, Fredrik Österlind, and Zhitao He. An adaptive communication architecture for wireless sensor networks. In *SenSys '07: Proceedings of the 5th international conference on Embedded networked sensor systems*, pages 335–349, New York, NY, USA, 2007. ACM.

[42] Adam Dunkels, Thiemo Voigt, and Juan Alonso. Making TCP/IP Viable for Wireless Sensor Networks. In *Proceedings of the First European Workshop on Wireless Sensor Networks (EWSN 2004), work-in-progress session*, Berlin, Germany, January 2004.

[43] Dust Networks. Technical overview of time synchronized mesh protocol (tsmp). http://www.dustnetworks.com/docs/TSMP_Whitepaper.pdf, June 2006.

[44] Prabal Dutta, Mike Grimmer, Anish Arora, Steven Bibyk, and David Culler. Design of a wireless sensor network platform for detecting rare, random, and ephemeral events. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, page 70, Piscataway, NJ, USA, 2005. IEEE Press.

[45] Cheng Tien Ee and Ruzena Bajcsy. Congestion control and fairness for many-to-one routing in sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 148–161, New York, NY, USA, 2004. ACM.

[46] Cheng Tien Ee, Rodrigo Fonseca, Sukun Kim, Daekyeong Moon, Arsalan Tavakoli, David Culler, Scott Shenker, and Ion Stoica. A modular network layer for sensorsets. In *OSDI '06: Proceedings of*

the 7th symposium on Operating systems design and implementation, pages 249–262, Berkeley, CA, USA, 2006. USENIX Association.

[47] S. B. Eisenman, E. Miluzzo, N. D. Lane, R. A. Peterson, G-S. Ahn, and A. T. Campbell. The BikeNet mobile sensing system for cyclist experience mapping. In *SenSys '07: Proceedings of the 5th international conference on Embedded networked sensor systems*, pages 87–101, New York, NY, USA, 2007. ACM.

[48] A. El-Hoiydi and J.-D. Decotignie. WiseMAC: an ultra low power MAC protocol for the downlink of infrastructure wireless sensor networks. In *ISCC '04: Proceedings of the Ninth International Symposium on Computers and Communications 2004 Volume 2 (ISCC"04)*, pages 244–251, Washington, DC, USA, 2004. IEEE Computer Society.

[49] Amre El-Hoiydi. Aloha with Preamble Sampling for Sporadic Traffic in Ad Hoc Wireless Sensor Networks. In *IEEE International Conference on Communications (ICC)*, New York, April 2002.

[50] Member-Sinem Coleri Ergen and Fellow-Pravin Varaiya. PEDAMACS: Power Efficient and Delay Aware Medium Access Protocol for Sensor Networks. *IEEE Transactions on Mobile Computing*, 5(7):920–930, 2006.

[51] Deborah Estrin, David Culler, Kris Pister, and Gaurav Sukhatme. Connecting the Physical World with Pervasive Networks. *IEEE Pervasive Computing*, 1(1):59–69, 2002.

[52] Deborah Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar. Next century challenges: scalable coordination in sensor networks. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 263–270, New York, NY, USA, 1999. ACM.

[53] D.J. Farber, G. Delp, and T.M. Conte. Thinwire protocol for connecting personal computers to the Internet. RFC 914 (Historic), September 1984.

[54] Maria Fazio, Claudio E. Palazzi, Shirshanka Das, and Mario Gerla. Automatic IP address configuration in VANETs. In *VANET '06: Proceedings of the 3rd international workshop on Vehicular ad hoc networks*, pages 100–101, New York, NY, USA, 2006. ACM Press.

[55] Sally Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. Netw.*, 1(4):397–413, 1993.

[56] Sally Floyd, Van Jacobson, Ching-Gung Liu, Steven McCanne, and Lixia Zhang. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Trans. Netw.*, 5(6):784–803, 1997.

[57] Rodrigo Fonseca, Omprakash Gnawali, Kyle Jamieson, and Philip Levis. Four Bit Wireless Link Estimation. In *HotNets VI: Proceedings of the Sixth Workshop on Hot Topics in Networks*, 2007.

[58] Rodrigo Fonseca, Sylvia Ratnasamy, Jerry Zhao, Cheng Tien Ee, David Culler, Scott Shenker, and Ion Stoica. Beacon vector routing: scalable point-to-point routing in wireless sensornets. In *NSDI'05: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*, pages 329–342, Berkeley, CA, USA, 2005. USENIX Association.

[59] L. Frelechoux, M. Osborne, and R. Haas. Topology optimization of IP over ATM. *Universal Multi-service Networks, 2000. ECUMN 2000. 1st European Conference on*, pages 122–131, 2000.

[60] Saurabh Ganeriwal, Deepak Ganesan, Hohyun Shim, Vlasios Tsiatsis, and Mani B. Srivastava. Estimating clock uncertainty for efficient duty-cycling in sensor networks. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 130–141, New York, NY, USA, 2005. ACM Press.

[61] J. J. Garcia-Lunes-Aceves. Loop-free routing using diffusing computations. *IEEE/ACM Trans. Netw.*, 1(1):130–141, 1993.

[62] D. Grossman and J. Heinanen. Multiprotocol Encapsulation over ATM Adaptation Layer 5. RFC 2684 (Proposed Standard), September 1999.

[63] M. Gupta and N. Melam. Authentication/Confidentiality for OSPFv3. RFC 4552 (Proposed Standard), June 2006.

[64] Silvia Hagen. *IPv6 Essentials*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2002.

[65] G. P. Halkes, T. van Dam, and K. G. Langendoen. Comparing energy-saving MAC protocols for wireless sensor networks. *Mob. Netw. Appl.*, 10(5):783–791, 2005.

[66] HART Communications Foundation. WirelessHART Overview. http://www.hartcomm2.org/hart_protocol/wireless_hart/wireless_hart_main.html, 2007.

[67] Carl Hartung, Richard Han, Carl Seielstad, and Saxon Holbrook. FireWxNet: a multi-tiered portable wireless system for monitoring weather conditions in wildland fire environments. In *MobiSys '06: Proceedings of the 4th international conference on Mobile systems, applications and services*, pages 28–41, New York, NY, USA, 2006. ACM.

[68] M. Harvan and J. Schnwlder. M.S. Thesis, Connecting Wireless Sensor Networks to the Internet a 6lowpan Implementation for TinyOS 2.0, May 2007.

[69] E. S. Hashem. Analysis of Random Drop for Gateway Congestion Control. November 1989.

[70] C.L. Hedrick. Routing Information Protocol. RFC 1058 (Historic), June 1988. Updated by RFCs 1388, 1723.

[71] Wendi Rabiner Heinzelman, Joanna Kulik, and Hari Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 174–185, New York, NY, USA, 1999. ACM.

[72] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister. System architecture directions for networked sensors. *SIGPLAN Not.*, 35(11):93–104, 2000.

[73] Jason L. Hill and David E. Culler. Mica: A Wireless Platform for Deeply Embedded Networks. *IEEE Micro*, 22(6):12–24, 2002.

[74] Robert Hinden and Stephen Deering. IP version 6 addressing architecture.

[75] Robert M. Hinden. IP next generation overview. *Commun. ACM*, 39(6):61–71, 1996.

[76] Hitachi. 6LoWPAN Network Stack. http://www.hitachi.com/.

[77] Barbara Hohlt, Lance Doherty, and Eric Brewer. Flexible power scheduling for sensor networks. In *IPSN '04: Proceedings of the third international symposium on Information processing in sensor networks*, pages 205–214, New York, NY, USA, 2004. ACM.

[78] Jonathan W. Hui and David Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 81–94, New York, NY, USA, 2004. ACM.

[79] Jonathan W. Hui and David E. Culler. Extending IP to Low-Power, Wireless Personal Area Networks. *Internet Computing, IEEE*, 12(4):37–45, July-Aug. 2008.

[80] Bret Hull, Vladimir Bychkovsky, Yang Zhang, Kevin Chen, Michel Goraczko, Allen Miu, Eugene Shih, Hari Balakrishnan, and Samuel Madden. CarTel: a distributed mobile sensor computing system. In *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 125–138, New York, NY, USA, 2006. ACM.

[81] Bret Hull, Kyle Jamieson, and Hari Balakrishnan. Mitigating congestion in wireless sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 134–147, New York, NY, USA, 2004. ACM.

[82] Hongwei Huo, Hongke Zhang, Yanchao Niu, Shuai Gao, Zhaohua Li, and Sidong Zhang. Msrlab6: An ipv6 wireless sensor networks testbed. *Signal Processing, 2006 8th International Conference on*, 4:–, 16-20 2006.

[83] Geoff Huston. IPv4 Address Report. http://www.potaroo.net/tools/ipv4/index.html, August 2008.

[84] ISA. ISA100, Wireless Systems for Automation. http://www.isa.org/isa100, 2007.

[85] V. Jacobson. Congestion avoidance and control. In *SIGCOMM '88: Symposium proceedings on Communications architectures and protocols*, pages 314–329, New York, NY, USA, 1988. ACM.

[86] Van Jacobson. Compressing TCP/IP Headers for Low-Speed Serial Links. RFC 1144 (Proposed Standard), February 1990.

[87] Christophe Jelger and Thomas Noel. Prefix continuity and global address autoconfiguration in IPv6 ad hoc networks. In *MedHoc '05: 4th Annual Mediterranean Ad Hoc Networking Workshop*, pages 311–320. Springer Boston, 2005.

[88] Jennic. 6LoWPAN Network Stack. http://www.jennic.com/.

[89] J. Jeong, S. Park, L. Beloeil, and S. Madanapalli. IPv6 Router Advertisement Option for DNS Configuration. RFC 5006 (Experimental), September 2007.

[90] P. Johansson. IPv4 over IEEE 1394. RFC 2734 (Proposed Standard), December 1999.

[91] D. Johnson, Y. Hu, and D. Maltz. The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4. RFC 4728 (Experimental), February 2007.

[92] John Jubin and Janet D. Tornow. The DARPA packet radio network protocols. *Proceedings of IEEE*, 75(1):21–32, January 1987.

[93] Brad Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks.

[94] Dina Katabi, Mark Handley, and Charlie Rohrs. Congestion control for high bandwidth-delay product networks. *SIGCOMM Comput. Commun. Rev.*, 32(4):89–102, 2002.

[95] D. Katz. Transmission of IP and ARP over FDDI Networks. RFC 1390 (Standard), January 1993.

[96] Sukun Kim, Rodrigo Fonseca, Prabal Dutta, Arsalan Tavakoli, David Culler, Philip Levis, Scott Shenker, and Ion Stoica. Flush: a reliable bulk transport protocol for multihop wireless networks. In *SenSys '07: Proceedings of the 5th international conference on Embedded networked sensor systems*, pages 351–365, New York, NY, USA, 2007. ACM.

[97] Sukun Kim, Shamim Pakzad, David Culler, James Demmel, Gregory Fenves, Steven Glaser, and Martin Turon. Health monitoring of civil infrastructures using wireless sensor networks. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 254–263, New York, NY, USA, 2007. ACM.

[98] S. Kopparty, S.V. Krishnamurthy, M. Faloutsos, and S.K. Tripathi. Split TCP for mobile ad hoc networks. *Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE*, 1:138–142 vol.1, Nov. 2002.

[99] Lakshman Krishnamurthy, Robert Adler, Phil Buonadonna, Jasmeet Chhabra, Mick Flanigan, Nandakishore Kushalnagar, Lama Nachman, and Mark Yarvis. Design and deployment of industrial sensor networks: experiences from a semiconductor plant and the north sea. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 64–75, New York, NY, USA, 2005. ACM.

[100] Fabian Kuhn, Roger Wattenhofer, Yan Zhang, and Aaron Zollinger. Geometric ad-hoc routing: of theory and practice. In *PODC '03: Proceedings of the twenty-second annual symposium on Principles of distributed computing*, pages 63–72, New York, NY, USA, 2003. ACM.

[101] Ben Leong, Sayan Mitra, and Barbara Liskov. Path Vector Face Routing: Geographic Routing with Local Face Information. In *ICNP '05: Proceedings of the 13TH IEEE International Conference on Network Protocols*, pages 147–158, Washington, DC, USA, 2005. IEEE Computer Society.

[102] Philip Levis, Sam Madden, David Gay, Joseph Polastre, Robert Szewczyk, Alec Woo, Eric Brewer, and David Culler. The emergence of networking abstractions and techniques in TinyOS. In *NSDI'04:*

Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation, pages 1–1, Berkeley, CA, USA, 2004. USENIX Association.

[103] Philip Levis, Neil Patel, David Culler, and Scott Shenker. Trickle: a self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In *NSDI'04: Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation*, pages 2–2, Berkeley, CA, USA, 2004. USENIX Association.

[104] J.C. Lin and S. Paul. RMTP: a reliable multicast transport protocol. *INFOCOM '96. Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation. Proceedings IEEE*, 3:1414–1424 vol.3, 24-28 Mar 1996.

[105] Kaisen Lin and Philip Levis. Data Discovery and Dissemination with DIP. In *IPSN '08: Proceedings of the 7th international symposium on Information processing in sensor networks*, Piscataway, NJ, USA, 2008. IEEE Press.

[106] J. Loughney. IPv6 Node Requirements. RFC 4294 (Informational), April 2006.

[107] J. Luo, P.T. Eugster, and J.-P. Hubaux. Route driven gossip: probabilistic reliable multicast in ad hoc networks. *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE*, 3:2229–2239 vol.3, 30 March-3 April 2003.

[108] Joseph Macker. Simplified Multicast Forwarding for MANET. draft-ietf-manet-smf-07 (work in progress), February 2008.

[109] Alan Mainwaring, David Culler, Joseph Polastre, Robert Szewczyk, and John Anderson. Wireless sensor networks for habitat monitoring. In *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 88–97, New York, NY, USA, 2002. ACM.

[110] G. Malkin and R. Minnear. RIPng for IPv6. RFC 2080 (Proposed Standard), January 1997.

[111] Yun Mao, Feng Wang, Lili Qiu, Simon S. Lam, and Jonathan M. Smith+. S4: Small State and Small Stretch Routing Protocol for Large Wireless Sensor Networks. 2007.

[112] Karl Mayer and Wolfgang Fritsche. IP-enabled wireless sensor networks and their integration into the internet. In *InterSense '06: Proceedings of the first international conference on Integrated internet ad hoc and sensor networks*, page 5, New York, NY, USA, 2006. ACM.

[113] L.J. McLaughlin. Standard for the transmission of 802.2 packets over IPX networks. RFC 1132 (Standard), November 1989.

[114] Partho P. Mishra and Hemant Kanakia. A hop by hop rate-based congestion control scheme. *SIGCOMM Comput. Commun. Rev.*, 22(4):112–123, 1992.

[115] A. Misra, S. Das, A. McAuley, and S.K. Das. Autoconfiguration, registration, and mobility management for pervasive computing. *Personal Communications, IEEE [see also IEEE Wireless Communications]*, 8(4):24–31, Aug 2001.

[116] Mansoor Mohsin and Ravi Prakash. IP address assignment in a mobile ad hoc network.

[117] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. RFC 4944 (Proposed Standard), September 2007.

[118] N. Moore. Optimistic Duplicate Address Detection (DAD) for IPv6. RFC 4429 (Proposed Standard), April 2006.

[119] René Müller, Gustavo Alonso, and Donald Kossmann. A virtual machine for sensor networks. *SIGOPS Oper. Syst. Rev.*, 41(3):145–158, 2007.

[120] T. Narten and R. Draves. Privacy Extensions for Stateless Address Autoconfiguration in IPv6. RFC 3041 (Proposed Standard), January 2001.

[121] T. Narten, E. Nordmark, W. Simpson, and H. Soliman. Neighbor Discovery for IP version 6 (IPv6). RFC 4861 (Draft Standard), September 2007.

[122] James Newsome and Dawn Song. GEM: Graph EMbedding for routing and data-centric storage in sensor networks without geographic information. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 76–88, New York, NY, USA, 2003. ACM.

[123] Sze-Yao Ni, Yu-Chee Tseng, Yuh-Shyan Chen, and Jang-Ping Sheu. The broadcast storm problem in a mobile ad hoc network. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 151–162, New York, NY, USA, 1999. ACM.

[124] Erik Nordmark. Stateless IP/ICMP Translation Algorithm (SIIT). RFC 2765 (Proposed Standard), February 2000.

[125] W. Noureddine and F. Tobagi. Selective back-pressure in switched Ethernet LANs. *Global Telecommunications Conference, 1999. GLOBECOM '99*, 2:1256–1263 vol.2, 1999.

[126] University of California at Berkeley. TinyOS. http://www.tinyos.net/, 2004.

[127] The Institute of Electrical and Inc. Electronics Engineers. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, 1999.

[128] The Institute of Electrical and Inc. Electronics Engineers. Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs), October 2003.

[129] R. Ogier, F. Templin, and M. Lewis. Topology Dissemination Based on Reverse-Path Forwarding (TBRPF). RFC 3684 (Experimental), February 2004.

[130] Cüneyt Özveren, Robert Simcoe, and George Varghese. Reliable and efficient hop-by-hop flow control. In *SIGCOMM '94: Proceedings of the conference on Communications architectures, protocols and applications*, pages 89–100, New York, NY, USA, 1994. ACM.

[131] Guangyu Pei, Mario Gerla, and Xiaoyan Hong. LANMAR: landmark routing for large scale wireless ad hoc networks with group mobility. In *MobiHoc '00: Proceedings of the 1st ACM international*

symposium on Mobile ad hoc networking & computing, pages 11–18, Piscataway, NJ, USA, 2000. IEEE Press.

[132] Charlie Perkins. IP Address Autoconfiguration for Ad Hoc Networks. draft-perkins-manet-autoconf-01 (work in progress), November 2001.

[133] Charlie Perkins. Ad hoc On-Demand Distance Vector (AODV) Routing. RFC 3561 (Experimental), July 2003.

[134] Charlie Perkins and Pravin Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*.

[135] D. Piscitello and J. Lawrence. The Transmission of IP Datagrams over the SMDS Service. RFC 1209 (Standard), March 1991.

[136] D. Plummer. Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware. RFC 826 (Standard), November 1982.

[137] Joseph Polastre, Jason Hill, and David Culler. Versatile low power media access for wireless sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 95–107, New York, NY, USA, 2004. ACM Press.

[138] Joseph Polastre, Jonathan Hui, Philip Levis, Jerry Zhao, David Culler, Scott Shenker, and Ion Stoica. A unifying link abstraction for wireless sensor networks. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 76–89, New York, NY, USA, 2005. ACM Press.

[139] Joseph Polastre, Robert Szewczyk, and David Culler. Telos: enabling ultra-low power wireless research. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, page 48, Piscataway, NJ, USA, 2005. IEEE Press.

[140] Joseph Polastre, Gilman Tolle, and Jonathan Hui. Low power mesh networking with Telos and IEEE 802.15.4. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 319–319, New York, NY, USA, 2004. ACM.

[141] J. Postel. DoD standard Transmission Control Protocol. RFC 761, January 1980.

[142] J. Postel. Internet Control Message Protocol. RFC 792 (Standard), September 1981. Updated by RFCs 950, 4884.

[143] D. Provan. Transmitting IP traffic over ARCNET networks. RFC 1201 (Standard), February 1991.

[144] W. Prue and J. Postel. Something a host could do with source quench: The Source Quench Introduced Delay (SQuID). RFC 1016, July 1987.

[145] K. K. Ramakrishnan and R. Jain. A binary feedback scheme for congestion avoidance in computer networks. *ACM Trans. Comput. Syst.*, 8(2):158–181, 1990.

[146] Sumit Rangwala, Ramakrishna Gummadi, Ramesh Govindan, and Konstantinos Psounis. Interference-aware fair rate control in wireless sensor networks. *SIGCOMM Comput. Commun. Rev.*, 36(4):63–74, 2006.

[147] Ananth Rao, Sylvia Ratnasamy, Christos Papadimitriou, Scott Shenker, and Ion Stoica. Geographic routing without location information. In *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 96–108, New York, NY, USA, 2003. ACM.

[148] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol Label Switching Architecture. RFC 3031 (Proposed Standard), January 2001.

[149] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *ACM Trans. Comput. Syst.*, 2(4):277–288, 1984.

[150] Yogesh Sankarasubramaniam, Özgür B. Akan, and Ian F. Akyildiz. ESRT: event-to-sink reliable transport in wireless sensor networks. In *MobiHoc '03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 177–188, New York, NY, USA, 2003. ACM.

[151] Y. Sasson, D. Cavin, and A. Schiper. Probabilistic broadcast for flooding in wireless mobile ad hoc networks. *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*, 2:1124–1130 vol.2, 16-20 March 2003.

[152] Naveen Sastry and David Wagner. Security Considerations for IEEE 802.15.4 Networks. In *ACM Workshop on Wireless Security (WiSe 2004)*, October 2004.

[153] Curt Schurgers, Vlasios Tsiatsis, Saurabh Ganeriwal, and Mani Srivastava. Optimizing Sensor Networks in the Energy-Latency-Density Design Space. *IEEE Transactions on Mobile Computing*, 1(1):70–80, 2002.

[154] Sensinode. NanoStack. http://sourceforge.net/projects/nanostack/.

[155] Vipul Singhvi, Andreas Krause, Carlos Guestrin, Jr. James H. Garrett, and H. Scott Matthews. Intelligent light control using sensor networks. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 218–229, New York, NY, USA, 2005. ACM.

[156] T. Speakman, J. Crowcroft, J. Gemmell, D. Farinacci, S. Lin, D. Leshchiner, M. Luby, T. Montgomery, L. Rizzo, A. Tweedly, N. Bhaskar, R. Edmonstone, R. Sumanasekera, and L. Vicisano. PGM Reliable Transport Protocol Specification. RFC 3208 (Experimental), December 2001.

[157] Kannan Srinivasan, Prabal Dutta, Arsalan Tavakoli, and Philip Levis. Some Implications of Low-Power Wireless to IP Routing. In *HotNets V: Proceedings of the Fifth Workshop on Hot Topics in Networks*, New York, NY, USA, 2006. ACM.

[158] F. Stann and J. Heidemann. RMST: reliable data transport in sensor networks. *Sensor Network Protocols and Applications, 2003. Proceedings of the First IEEE. 2003 IEEE International Workshop on*, pages 102–112, 11 May 2003.

[159] Thad Starner and Joseph Paradiso. Human Generated Power for Mobile Electronics. *Low-Power Electronics*, pages 45–1–45–35, 2004.

[160] Sun Microsystems. SunSpotWorld. http://www.sunspotworld.com/.

[161] Robert Szewczyk, Alan Mainwaring, Joseph Polastre, John Anderson, and David Culler. An analysis of a large scale habitat monitoring application. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 214–226, New York, NY, USA, 2004. ACM.

[162] Katalin Szlavecz, Andreas Terzis, Razvan Musloiu-E., Joshua Cogan, Sam Small, Stuart Ozer, Randal Burns, Jim Gray, and Alexander S. Szalay. Life Under Your Feet: An End-to-End Soil Ecology Sensor Network, Database, Web Server, and Analysis Service. Technical Report MSR-TR-2006-90, Microsoft Research, 2006.

[163] Jay Taneja, Jaein Jeong, and David Culler. Design, Modeling, and Capacity Planning for Micro-solar Power Sensor Networks. *Information Processing in Sensor Networks, 2008. IPSN '08. International Conference on*, pages 407–418, April 2008.

[164] Abhishek Prakash Tayal and L. M. Patnaik. An address assignment for the automatic configuration of mobile ad hoc networks. *Personal Ubiquitous Comput.*, 8(1):47–54, 2004.

[165] Texas Instruments. CC2420: 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver. http://focus.ti.com/lit/ds/symlink/cc2420.pdf, March 2007.

[166] Texas Instruments. CC2500: Low-Cost Low-Power 2.4 GHz RF Trnasceiver. http://focus.ti.com/lit/ds/symlink/cc2500.pdf, September 2007.

[167] D. Thaler, M. Talwar, and C. Patel. Neighbor Discovery Proxies (ND Proxy). RFC 4389 (Experimental), April 2006.

[168] S. Thomson, T. Narten, and T. Jinmei. IPv6 Stateless Address Autoconfiguration. RFC 4862 (Draft Standard), September 2007.

[169] G. Tolle and D. Culler. Design of an application-cooperative management system for wireless sensor networks. *Wireless Sensor Networks, 2005. Proceeedings of the Second European Workshop on*, pages 121–132, 31 Jan.-2 Feb. 2005.

[170] Gilman Tolle, Joseph Polastre, Robert Szewczyk, David Culler, Neil Turner, Kevin Tu, Stephen Burgess, Todd Dawson, Phil Buonadonna, David Gay, and Wei Hong. A macroscope in the redwoods. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 51–63, New York, NY, USA, 2005. ACM.

[171] O. Troan and R. Droms. IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6. RFC 3633 (Proposed Standard), December 2003.

[172] P. F. Tsuchiya. The landmark hierarchy: a new hierarchy for routing in very large networks. In *SIGCOMM '88: Symposium proceedings on Communications architectures and protocols*, pages 35–42, New York, NY, USA, 1988. ACM.

[173] Jorge Urrutia. Routing with guaranteed delivery in geometric and wireless networks. pages 393–406, 2002.

[174] Nitin H. Vaidya. Weak duplicate address detection in mobile ad hoc networks. In *MobiHoc '02: Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, pages 206–216, New York, NY, USA, 2002. ACM Press.

[175] Tijs van Dam and Koen Langendoen. An adaptive energy-efficient MAC protocol for wireless sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 171–180, New York, NY, USA, 2003. ACM Press.

[176] D. Waitzman, C. Partridge, and S.E. Deering. Distance Vector Multicast Routing Protocol. RFC 1075 (Experimental), November 1988.

[177] Chieh-Yih Wan, Andrew T. Campbell, and Lakshman Krishnamurthy. PSFQ: a reliable transport protocol for wireless sensor networks. In *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 1–11, New York, NY, USA, 2002. ACM.

[178] Chieh-Yih Wan, Shane B. Eisenman, and Andrew T. Campbell. CODA: congestion detection and avoidance in sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 266–279, New York, NY, USA, 2003. ACM.

[179] K. Weniger. PACMAN: passive autoconfiguration for mobile ad hoc networks. *Selected Areas in Communications, IEEE Journal on*, 23(3):507–519, March 2005.

[180] K. Weniger and M. Zitterbart. IPv6 Autoconfiguration in Large Scale Mobile. In *European Wireless*, 2002.

[181] Geoffrey Werner-Allen, Konrad Lorincz, Matt Welsh, Omar Marcillo, Jeff Johnson, Mario Ruiz, and Jonathan Lees. Deploying a Wireless Sensor Network on an Active Volcano. *IEEE Internet Computing*, 10(2):18–25, 2006.

[182] Cedric Westphal. A User-based Frequency-dependent IP Header Compression Architecture. 2002.

[183] Cedric Westphal. Layered IP Header Compression for IP-enabled Sensor Networks. In *IEEE International Conference on Communications (ICC)*, 2006.

[184] Cedric Westphal and Rajiv Koodli. Stateless IP Header Compression. In *IEEE International Conference on Communications (ICC)*, 2005.

[185] Alec Woo and David E. Culler. A transmission control scheme for media access in sensor networks. In *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 221–235, New York, NY, USA, 2001. ACM.

[186] Alec Woo, Terence Tong, and David Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 14–27, New York, NY, USA, 2003. ACM Press.

[187] Ning Xu, Sumit Rangwala, Krishna Kant Chintalapudi, Deepak Ganesan, Alan Broad, Ramesh Govindan, and Deborah Estrin. A wireless sensor network For structural monitoring. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 13–24, New York, NY, USA, 2004. ACM.

[188] Ya Xu, John Heidemann, and Deborah Estrin. Geography-informed energy conservation for Ad Hoc routing. In *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 70–84, New York, NY, USA, 2001. ACM.

[189] Wei Ye, John Heidemann, and Deborah Estrin. An Energy-efficient MAC Protocol for Wireless Sensor Networks. In *21st Conference of the IEEE Computer and Communications Societies (INFOCOM)*, volume 3, pages 1567–1576, June 2002.

[190] Wei Ye, John Heidemann, and Deborah Estrin. Medium access control with coordinated adaptive sleeping for wireless sensor networks. *IEEE/ACM Trans. Netw.*, 12(3):493–506, 2004.

[191] Wei Ye, Fabio Silva, and John Heidemann. Ultra-low duty cycle MAC with scheduled channel polling. In *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 321–334, New York, NY, USA, 2006. ACM Press.

[192] Z-Wave. Z-Wave. http://www.z-wave.com/.

[193] Lixia Zhang. A New Architecture for Packet Switching Network Protocols. July 1989.

[194] Hongbo Zhou, Lionel M. Ni, and Matt W. Mutka. Prophet address allocation for large scale MANETs. *Ad Hoc Networks*, 1(4):423–434, 2003.

[195] ZigBee Alliance. ZigBee. http://www.zigbee.org/.