# IP is Dead, Long Live IP for Wireless Sensor Networks

Jonathan W. Hui
University of California at Berkeley
Arch Rock Corporation
jwhui@cs.berkeley.edu

David E. Culler
University of California at Berkeley
Arch Rock Corporation
culler@cs.berkeley.edu

## ABSTRACT

A decade ago as wireless sensor network research took off many researchers in the field denounced the use of IP as inadequate and in contradiction to the needs of wireless sensor networking. Since then the field has matured, standard links have emerged, and IP has evolved. In this paper, we present the design of a complete IPv6-based network architecture for wireless sensor networks. We validate the architecture with a production-quality implementation that incorporates many techniques pioneered in the sensor network community, including duty-cycled link protocols, header compression, hop-by-hop forwarding, and efficient routing with effective link estimation. In addition to providing interoperability with existing IP devices, this 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. Our results outperform existing systems that do not adhere to any particular standard or architecture. 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 wireless sensor networks going forward.

## Categories and Subject Descriptors

C.2.1 [**Computer-Communications Networks**]: Network Architecture and Design—*Wireless communication*; C.2.2 [**Computer-Communications Networks**]: Network Protocols; C.2.6 [**Computer-Communications Networks**]: Internetworking—*Standards*

## General Terms

Design, Measurement, Performance, Reliability, Security, Standardization

## Keywords

network architecture; internet; internetworking; wireless; sensor networks; IP; IPv6; 6LoWPAN; media management

## 1. INTRODUCTION

As wireless sensor network (WSN) 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" [19]. The Internet architecture was denounced for several reasons including the following [19]:

- 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." WSNs will be tailored to the sensing task at hand.

In addition, it was argued that to tackle the challenges of WSNs the traditional interfaces and layers of system abstraction should not be assumed [24]. By providing a framework for defining abstractions and allowing the community to progress, new network abstractions were expected to emerge [30]. Indeed, by introducing the Active Message Dispatch ID at the head of each message, rather than a conventional header format, TinyOS [49] 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 WSN, so WSNs were organized in a manner similar to IrDA and USB, rather than 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 WSNs are used in practice.

Over this same period, the Internet has evolved as well. In 1998, RFC 2460 defined IPv6 [12]. The large address space not only provided for a huge number of devices, it eliminated many of the artificial naming constraints. This enabled the definition of an adaptation layer in RFC 4944 (6LoWPAN) that carried the meaning of IPv6 addresses in a compact form using small IEEE 802.15.4 short addresses [34]. The IPv6 prefix generalized the notion of a subnet. The various layer-two bootstrapping, discovery, and autoconfiguration mechanisms used with IPv4 were consolidated into the IPv6 framework and went directly at the issue of vast numbers of unattended devices in a changing environment. Finally, the systematic use of options provided for compact headers in the common case,


*SenSys'08,* November 5–7, 2008, Raleigh, North Carolina, USA.

while permitting a natural means of extending IP where the existing standards do not fully solve the problem at hand.

This paper provides three primary contributions:

1. We develop a complete IPv6-based network architecture for WSNs that allows end-to-end communication between nodes 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 WSN community.

3. We present the implementation of a complete, efficient, and production-quality IPv6 solution for WSNs and show that this general network architecture can outperform existing systems that do not adhere to any particular standard or architecture.

We start by providing a high-level overview of the network architecture, describing the expected network organization and defining the IP link, and then show how to support the architecture efficiently using many techniques that have been pioneered in the WSN community. Starting with the link, we describe the mechanisms necessary for the network-layer to efficiently form a network, configure routes, and forward datagrams. We then develop an adaptation layer that compacts IPv6 datagrams and fragments them when they do not fit in a single IEEE 802.15.4 frame. At the network layer, we describe adaptations to Neighbor Discovery and Autoconfiguration used to configure and maintain IPv6 networks; mechanisms used to efficiently forward datagrams while maintaining high end-to-end delivery rates; and a routing protocol that maintains small and constant state with little communication overhead. Using a production-quality implementation, we evaluate the performance of a real-world application deployment. Comparing the results to existing data, we show that this implementation achieves lower energy cost, lower per-hop latency, and higher data reception rate while communicating more data. Finally, we revisit the assumptions that have formed the basis of so much WSN research.

## 2. RELATED WORK

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 [5].

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 [14]. uIP has evolved to include a low-power link built on IEEE 802.15.4, showing that IP was feasible for WSNs. However, the proof-of-concept implementation did not take advantage of the numerous mechanisms developed within the WSN community [16].

Seeing that it was possible to implement IP on small devices, the IETF formed the 6LoWPAN working group 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 [25, 34]. Our initial work significantly influenced the design of RFC 4944. In this paper, we continue our effort to complete an IPv6-based network architecture for WSNs.

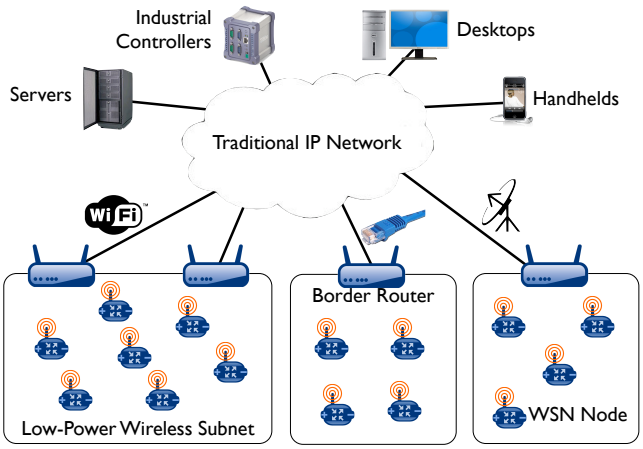



**Figure 1: Extending the Internet Architecture.** *Communicating natively with IPv6, nodes can communicate end-to-end with each other and any arbitrary IP device over the wide-area at the network layer. Border routers connect WSNs to other IP networks using traditional wired or wireless (e.g., WiFi and satellite) links and do not require any application-specific state.*

MSRLab6 [27] and NanoStack [42] validated the feasibility of RFC 4944 in WSNs, 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 [32]. None of these address operation over duty-cycled links nor do they incorporate existing WSN techniques. All of them also emulated a single IP link that spans the entire WSN, which requires additional configuration, forwarding, and routing mechanisms at the link layer. In this paper, we develop a complete IPv6-based network architecture, that maintains configuration, forwarding, and routing within the network layer. Using many existing WSN techniques, we show that our implementation can outperform existing systems that do not adhere to any particular standard or architecture.

In general, WSN 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 [15, 17, 38]. 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 WSNs 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 paper, we show how to adapt the IPv6 network architecture to WSNs and demonstrate what we can achieve with such an architecture.

# 3. AN IPV6 ARCHITECTURE

IPv6 is the designated successor of IPv4 as the network protocol for the Internet [12]. 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) allows hosts to autoconf IP addresses and other configuration parameters without a human in the loop [9, 35, 47]. Various layer-two protocols (e.g., ARP and DHCP) have been pulled into the IPv6 framework [47]. IPv6 also supports a richer set of communication paradigms, including a scoped addressing architecture and multicast into the core design [11]. IPv6 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 WSNs?

We claim that *IPv6 is better suited to the needs of WSNs 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 WSN 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, we believe IPv6 allows more efficient implementations than IPv4. The structure of IPv6 addresses are more amenable to cross-layer compression. Inclusion of necessary functionality (e.g., DHCP) that were previously outside the IP framework allows us to utilize the same packet processing, forwarding, and routing used for delivering any other IP datagram. Autoconf and ICMPv6 were designed to address scalability, visibility, and unattended operation, all features necessary in production WSNs. IPv6 does require some adaptation to better support operation in WSNs, and IPv6's support for a simple header options framework in almost every core protocol allows us to do just that.

## 3.1 Architecture Overview

In this paper, we show that the IPv6 network architecture is general enough that it can be applied efficiently in the WSN space. We consider a network organization where a WSN is composed of a collection of low-power wireless nodes that may require multihop communication to reach each other. Each WSN node serves as an IP router, but typically operates with a single interface. As shown in Figure 1, we assume that WSNs will typically operate on the edge of IP networks, acting as stub networks with all nodes assigned a common prefix. While nodes may be mobile, they generally remain within the WSN.

The WSN may be connected to other IP networks through one or more *border routers* that forward IP datagrams between different media. Connectivity to other IP networks may be provided through any arbitrary IP link, including Ethernet, WiFi, GPRS, or satellite. Border routers may also implement IPv4-to-IPv6 translation to support interoperability with IPv4 networks. 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 WSNs to other networks. These application gateways must understand any application profiles that may be used in the WSN, and any changes to the application protocols used in the WSN require changes on the gateway. In contrast, border routers remain agnostic to the set of applications deployed in the network.

IP's layered model means that peers communicate in terms of the capabilities provided by the layer 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

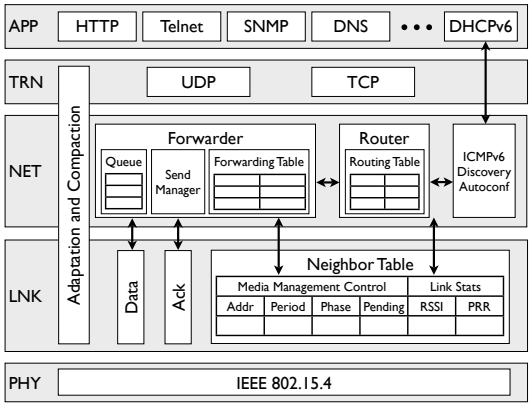



**Figure 2: Software Architecture.** *Each node implements a full network stack, respecting IP's layered model while using the proper mechanisms to support efficient communication in WSNs.*

applications with an effective communication channel. 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 WSN community and cast them in a software architecture that preserves the layering and functionality separation, as shown in Figure 2. The following sections describe each component in detail.

## 3.2 Avoiding IP Link Emulation

An IP link is defined by those nodes that are reachable over a single IP hop. Reachability may be provided by a direct connection at the physical layer (e.g., basic Ethernet) or emulated over different physical communication domains (e.g., switched Ethernet). In either case, IP-based protocols generally assume three properties:

- **Always-on**: The IP link provides a *connectionless* communication service, allowing a node to communicate with any other node attached to the same IP link at any time without the need to establish a connection.
- **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.
- **Single Broadcast Domain**: The IP link provides *transitive* reachability for all nodes on the link (if A can send to B and B can send to C, then A can send to C).

These basic assumptions of IP links are supported by the vast majority of link technologies in use today (e.g., Ethernet, WiFi, and point-to-point). In many cases, the properties are emulated by forwarding the message at the link layer to achieve greater reliability or forwarding the message to make the link appear as a single broadcast domain. However, as we have seen with ATM networks, IP link emulation has not always been met with success [2]. A lesson learned from the ATM experience is that link emulation can place too much policy in the link layer, and does not give the network layer necessary visibility into complex link-layer dynamics.

Drawing on the ATM experience, our IP link model exposes only basic functionality to the network layer. We equate an IP link to those neighbors reachable within a single radio transmission. The result is a WSN composed of overlapping link-local scopes. Doing so violates the single broadcast domain assumption, but gives IP-based protocols the necessary visibility into the radio connectivity

to meaningfully discover and communicate with their radio neighbors, and ultimately allow routing or app-level protocols to build higher-level structures based on the link connectivity. We also expose aspects of the unreliable nature of wireless communication to the network layer, giving the network layer better control of forwarding policies.

# 4. LINK LAYER

While IEEE 802.15.4 specifies a low-power wireless link standard with numerous implementations on the market, the industry has not yet come to agreement on a link protocol for duty-cycled operation in a multihop network. As this is required for interoperability at higher layers, we develop a duty-cycled link protocol, while keeping in mind the use of an IPv6 network layer above. Xu et. al. observed that *idle-listening* completely dominates system energy consumption when the radio is not duty-cycled [53]. Low-power radios consume as much power when receiving, or even just listening, when compared to transmitting [45].

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. In the remainder of this section, we develop a duty-cycled link protocol and a software abstraction that presents the MMC to the IPv6 network layer. In following sections, we describe how the network layer makes use of the MMC.

## 4.1 Emulating an Always-On Link

An *always-on* link provides a connectionless communication service. 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 does not require state synchronization. In general, these capabilities enable a more robust network.

Two mechanisms have emerged from existing work in duty-cycled links: *sampled listening* [18, 28, 37] and *scheduling* [50, 54]. Sampled listening trades reduced listening cost for increased transmission cost, assuming that 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. Sampled listening supports the always-on abstraction - nodes do not maintain any state to communicate with neighbors. 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. Indeed, scheduling can be used to optimize sampled listening, and we apply similar techniques to increase efficiency [18, 55].

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.

## 4.2 Media Management Control

Our MMC builds on B-MAC [37] and WiseMAC [18]. Sampled-listening provides the baseline communication mechanism, supporting a robust always-on abstraction and broadcast communication. We also 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 further lower transmission costs. Finally, we expose the link's capabilities through a simple, but expressive link abstraction derived from SP [38]. While we specifically address IEEE 802.15.4, the protocol is not fundamentally constrained to it.

### *4.2.1 Sampled Listening*

Sampled listening requires two primitives: (i) a wakeup signal when transmitting and (ii) channel sampling. Because existing 802.15.4 radios provide a packet-based interface, we implement the chirp signal using short *chirp frames* [6]. The chirp is an 802.15.4-compliant frame and contains a *destination address* and a *rendezvous time* that indicates the time remaining until the actual data frame transmission. Addressing information significantly reduces overhearing costs, as unintended destinations can abort reception early. The rendezvous time allows the destination to power down until the data frame begins transmission. This reduces the receive cost to that of receiving a chirp frame and the data frame, making the receive cost independent of the chirp signal length.

### *4.2.2 Synchronous Acks*

For IP to efficiently achieve high best effort datagram delivery, synchronous acks must be used in low-power devices where loss rates greater than 10% are common. Unfortunately, the ack frame defined in 802.15.4 is insufficient for use in production networks: (i) the ack frame contains no addressing information, which can lead to false positives in acked transmissions; (ii) ack frames cannot be secured, allowing an attacker to easily inject acks [41]; and (iii) ack frames cannot carry a payload, which is useful for hop-by-hop feedback needed by various protocols. Instead, we define a new ack frame as a IEEE 802.15.4 data frame. Because the new ack frame is a data frame, the already defined addressing and security mechanisms can be used. The new ack frame can also carry a payload, which we utilize for scheduling optimizations and network layer optimizations in Section 7 and Section 8.

### *4.2.3 Scheduling*

Nodes may monitor the channel sample period of their neighbors to reduce the chirp signal length. We include the sample period and phase in the payload of each ack. Using these acks, a node can synchronize to any neighbor after a single acked transmission. If the destination's schedule is known, the chirp duration can be reduced to a small synchronization guard time, which is progressively loosened based on the expected drift rate up to the receiver's sample period. More frequent communication reduces the average transmission cost, as synchronization occurs more frequently. Localized scheduling provides greater flexibility, robustness, and increases channel efficiency when channel sample schedules do not overlap. Unlike purely scheduled approaches, the local schedules

act as a *hint* and is required to communicate. Note that localized scheduling does not preclude the use of global coordination. A central manager can be used to collect and modify the schedules of individual nodes.

#### 4.2.4 Streaming

To increase throughput and energy efficiency, we apply streaming of multiple frames per channel sample [37]. Utilizing 802.15.4's Frame Pending bit, a transmitter can signal that another data frame will immediately follow. A node can then send data frames back-to-back without delay after sending a single chirp signal, allowing both sender and receiver to amortize wakeup costs across multiple frames.

### 4.3 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 abstraction is to take guidance from the network layer to optimize link-layer performance and enable the network layer to support efficient end-to-end datagram delivery. We build on our earlier work with SP [38], but simplify the abstraction because we assume a single network layer above. By exposing capabilities rather than specific mechanisms, the abstraction preserves the layered model.

As shown in Figure 1, the link layer maintains a neighbor table that holds link-specific state about neighbors, including link addresses, schedules, frame pending indicator, and link quality statistics. The MMC uses an LRU policy when inserting new neighbors, but allows the network layer to pin neighbors that are most relevant from a routing perspective. Link quality information includes both physical layer (RSSI) and link success rates that the network layer can use in selecting routes (Section 8). In addition to the neighbor table, the link abstraction augments the data-path by providing feedback on each transmission and reception. The link indicates whether a transmission attempt was acked and, if so, provides RSSI of the ack. The link also provides RSSI for each received frame.

## 5. ADAPTATION & COMPRESSION

The IEEE 802.15.4 frame only supports 127 bytes of payload and around 80 bytes in the worst case (when including extended addressing and full security information). IPv6 has a base header of 40 bytes and a minimum link MTU requirement of 1280 bytes. As a result, we must use an adaptation layer to fragment IPv6 datagrams when they do not fit within a single frame and compress IPv6 headers to make header overhead reasonable. In the layered architecture, this adaptation layer sits at layer 2.5 (between the link and network).

We took an initial step at such an adaptation layer in RFC 4944, defining a format for fragmenting IPv6 datagrams and compressing IPv6 headers [34]. In this section, we build on our work by generalizing the header compression to support additional communication paradigms. Our header compression can reduce a 48 byte UDP/IPv6 header down to 6 bytes in the best case.

### 5.1 Header Compression

Unlike traditional IP header compression, RFC 4944 header compression is stateless and places no binding state between a compressor/decompressor pair. Stateless compression gives WSN nodes the necessary flexibility to communicate with any neighbor in compact form at any time. RFC 4944 compresses headers in two ways. First, by making assumptions about common values for IPv6 header fields in WSNs (Version is 6, Traffic Class and Flow Label are 0, Next Header is UDP, TCP, or ICMPv6, and IPv6 prefixes are

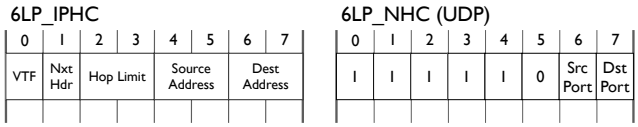



**Figure 3: Header Compression Format.** *The encoding formats specify what fields are carried inline.*

link-local). Second, by removing redundant information across layers (Payload Length and Interface Identifiers (IID) are derived from the link header). Eliding the IID is an example where IPv6 makes the architecture more feasible than IPv4. RFC 4944 also defines UDP header compression using similar techniques.

Nodes must be configured with global addresses to communicate over multiple hops. However, RFC 4944 does not efficiently compress headers when communicating outside of link-local scope or when using multicast. Any prefix other than the link-local prefix must be carried inline. Any suffix must be at least 64 bits when carried inline even if derived from a short 802.15.4 address. To provide better compression over a broader range of scenarios, we generalize RFC 4944 compression by defining a new compression format for IPv6 headers (6LP_IPHC) and arbitrary next headers (6LP_NHC).

#### 5.1.1 IPv6 Header Compression

6LP_IPHC compresses global prefixes by assuming that an entire WSN is assigned a *Common Global Prefix (CGP)*, which can be done using mechanisms defined in Section 6. Nodes use this shared context to elide the prefix whenever communicating with the CGP. The other commonly used prefix is the link-local prefix. To support both simultaneously, we utilize different 6LoWPAN Dispatch Types to distinguish between the CGP and link-local prefix [34].

The 6LP_IPHC encoding is only 1 byte. Two bits are used for each IPv6 address, indicating one of four possibilities: (i) full 128-bit address carried inline, (ii) 64-bit IID carried inline, (iii) bottom 16-bits of IID carried inline, and (iv) fully elided. When the prefix is elided, it is assumed to be the CGP or link-local prefix. The encoding uses one bit to indicate whether Version, Traffic Class, and Flow Label are elided. One bit is used to indicate if Next Header is elided and 6LP_NHC is in use. Two bits are used to compress the Hop Limit, indicating whether 1, 64, 255, or carried inline.[1]

IPv6 multicast is core to the IPv6 architecture and frequently used for discovery and configuration (Section 6). We compress well-known multicast addresses down to 16 bits by dividing the short address namespace: the first bit being 0 indicates a unicast address whereas the first three bits being '101' indicates a multicast address. Of the remaining 13 bits in the compressed multicast address, 4 bits carry the multicast scope inline and 9 bits are used to identify the well-known multicast address.

#### 5.1.2 Next Header Compression

When the IPv6 Next Header is compressed, a 6LP_NHC encoding follows the compressed IPv6 header. The 6LP_NHC encoding starts with a variable-length identifier that indicates the Next Header value and compression format. This paper initially defines UDP header compression within this framework. Like RFC 4944, the encoding uses 1 bit to compress the upper 12 bits of each UDP port but always elides UDP Payload Length, compressing an 8-byte UDP header down to 4 bytes in the best case.

#### 5.1.3 Compression Efficiency

6LP_IPHC and 6LP_NHC can compress a 48-byte UDP/IPv6 header down to 6 bytes when communicating over link-local uni-

[1]Chosen values are those most commonly used by IPv6 protocols.

cast, 8 bytes with link-local multicast, 11 bytes with global unicast and multicast within the WSN, and 25 bytes when communicating with arbitrary IP devices outside the WSN. With stateful compression at the border routers, the format can support a 11-byte header when communicating with arbitrary IP devices outside the WSN. Using ICMP, the border router can establish short address identifiers for arbitrary IP devices with WSN nodes.

# 6. ICMPV6, DISCOVERY, & AUTOCONF

The IP network layer includes three fundamental services: (i) configuration and management, (ii) forwarding, and (iii) routing. In this section, we discuss the first of those three. IPv6 anticipated the need to configure and manage large numbers of nodes, which is paramount in WSNs. The ICMPv6 framework supports a family of network layer control and management protocols [9]. Utilizing these, rather than creating specialized link-layer equivalents, provides the necessary functionality to manage a robust network. But while IPv6 has the right concepts, the solutions defined in RFCs were designed to operate over a single IP link supporting a single broadcast domain. In this section, we describe necessary extensions to existing discovery and autoconf mechanisms.

## 6.1 Neighbor Discovery (ND)

IPv6 nodes use ND to discover each other's presence, determine each other's link-layer addresses, find routers, and configure network parameters. Routers periodically multicast Router Advertisements (RA) to announce their existence and propagate network parameters to all hosts on the link (e.g., global prefixes for generating global addresses). However, as currently defined, RAs are only intended to operate over a single IP link and do not propagate information over multiple IP links. To effectively configure a WSN with overlapping link-local scopes, we extend RAs to propagate network parameters over multiple hops with little resource demand.

Border routers are the point of entry to WSNs and network parameters originate at border routers. Our architecture extends the use of RAs to disseminate network parameters over multiple hops. Maintaining network parameters is a state consistency problem, and we use the Trickle algorithm to provide low maintenance overhead [31]. The Trickle period resets whenever new parameters are discovered or Router Solicitation (RS) messages are received. To support Trickle, we extend RA options by including a sequence number that indicates freshness of the information. Nodes always accept information with the latest sequence number. Trickle is simple and scalable, and allows us to avoid many of the magic constants specified in existing standards [35]. We envision that Trickle can be applied to other IP protocols with similar benefits.

We further reduce the overhead of ND by making simplifying assumptions. Because IIDs embed link-layer addresses, nodes do not need to resolve and maintain address mappings when communicating with neighbors. Our ND does not maintain reachability information for neighboring nodes, as WSN protocols may have their own notion of reachability and have more information on which neighbors are meaningful. Using other established mechanisms to maintain IPv6 address uniqueness (described in the following section) allows us to avoid using ND for detecting duplicates. As a result, the need for communicating Neighbor Solicitation and Advertisement messages is reduced to discovering neighboring hosts.

## 6.2 Autoconfiguration

IPv6 anticipated the need for easy configuration and management of node addresses and other parameters. IPv6 defines both *stateless* [47] and *DHCPv6* [13] autoconf methods - the former disseminates parameters to all nodes in a network, while the latter selectively assigns parameters to individual nodes. Unlike traditional IPv6 networks, the addresses must be unique within the entire WSN. Node mobility or changing connectivity make it insufficient to only maintain uniqueness among neighboring nodes.

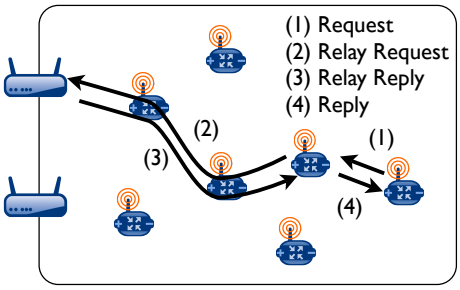



**Figure 4: DHCPv6 Autoconfiguration.** *DHCPv6 allows selective distribution of IPv6 addresses and configuration parameters to individual nodes. Relay Agents form an application overlay, allowing nodes to communicate over link-local scope before receiving a global address.*

Stateless autoconf generates IPv6 addresses by concatenating a prefix with an IID, the latter derived from 802.15.4 link addresses. While the link-local prefix is well-known, global prefixes are communicated using RA messages. The challenge is ensuring that the configured address is unique within the WSN, which works best when the uniqueness of link addresses is maintained by the link layer (e.g., PAN coordinator). In this case, stateless autoconf is attractive because nodes can configure addresses with very low network-layer cost - that of receiving a single RA. However, the link layer must also implement configuration, forwarding, and routing simply to manage addresses across the WSN.

By using DHCPv6, we can directly leverage the IPv6 infrastructure that we've built, rather than requiring additional functionality at the layers below. DHCPv6 allows nodes to request information from a central server, providing a flexible framework to distribute configuration parameters. Because the server maintains a central view of all nodes in the network, it can trivially ensure the uniqueness of both link-local and global addresses. Interestingly, DHCPv6 operates naturally in a multihop WSN without extensions. Every WSN router operates as a DHCPv6 Relay Agent, forwarding replies between clients and a DHCPv6 server. Relay Agents create an application-level overlay, allowing clients to communicate requests as they would in traditional networks, as shown in Figure 4.

# 7. 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 to the appropriate interface. The router is responsible for managing entries in the forwarding table. In contrast to much WSN work, which typically integrates forwarding and routing together, our IPv6-based architecture for WSN maintains the separation, as shown in Figure 2.

The IP network layer must provide high best-effort datagram delivery and enable end-to-end mechanisms to achieve reliable transport. But IP does not take a position on the use of specific mechanisms or the software architecture used to implement them. Forwarding mechanisms can have a significant influence on overall performance and is well studied in both wired and wireless contexts. Many IP-based mechanisms have been proposed to improve throughput, efficiency, and fairness of the network. Some rely on interactions with transport-layer mechanisms [33]. Others address

issues in dealing with relatively high loss rates in wireless links [3]. Significant work in WSNs have addressed similar issues outside the IP context [26, 51].

A key concept common to much existing work in wireless networks is the use of hop-by-hop mechanisms for reliability and flow control. Compared to end-to-end mechanisms (e.g., TCP), hop-by-hop mechanisms have greater visibility in network conditions, are more capable of addressing highly variable link qualities in both spatial and temporal dimensions, and do not require costly end-to-end communication.

The primary goals of our forwarder design is to provide *energy efficient* and *high end-to-end success rates*. Energy efficiency is the ratio of the best achievable energy cost relative to the realized energy cost of delivering a datagram. Existing forwarder designs address throughput, fairness, or efficiency, but none address *energy efficiency*. Existing work often equates number of transmissions with energy efficiency, but doing so does not consider that transmission costs vary by an order of magnitude or more depending on what optimizations the link layer can apply. Most existing efforts with hop-by-hop forwarding rely on broadcasts or snooping to communicate feedback. Broadcast transmissions can be significantly more expensive than unicast. Snooping requires nodes to listen to all neighboring transmissions, which is costly in duty-cycled operation.

## 7.1 Unicast Forwarder

To maximize energy efficiency and reliability, the unicast forwarder applies three orthogonal mechanisms: *hop-by-hop recovery*, *streaming*, and *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. Because IP does not specify the mechanisms or software architecture, we were able to select and compose those mechanisms that allow efficient forwarding in WSNs. The mechanisms we use to implement IP forwarding are not specific IP. The only portion that is IP specific is the use of options headers within the datagram.

### 7.1.1 Hop-by-Hop Recovery

Hop-by-hop recovery is used to increase energy efficiency and end-to-end delivery rates. Dropped datagrams may be due to link transmission failures or queue overflows and can prevent the network layer from achieving high best-effort delivery rates or the transport layer from achieving efficient end-to-end reliability. Our hop-by-hop recovery takes a custody-transfer approach, with link-layer acks indicating whether or not the network-layer was able to accept the message. Piggybacking network-layer information on link acks provides effective feedback without relying on broadcast or continuous snooping.

Contrary to traditional IP forwarding implementations, the network layer (not the link) is responsible for retransmitting datagrams and allows the network layer to support *re-routing*. The forwarder performs a next-hop lookup on *every* transmission attempt. Doing so allows the forwarder to take advantage of changes the router may make to the forwarding table, and allows the router to be fairly dynamic in those changes. This is especially important in WSNs where link qualities may be highly variable.

### 7.1.2 Streaming

The forwarder uses streaming to reduce the average transmission cost and intentionally delays latency-tolerant datagrams to increase the benefits of streaming. When submitting datagrams to the link, the forwarder indicates whether other packets for the same next-hop destination will follow. While delaying datagrams will increase queue occupancy, hop-by-hop recovery prevents dropped datagrams due to queue overflows. Streaming also takes advantage of the temporal properties of link quality - if the first transmission succeeds, it is likely that the remaining transmissions will succeed. Finally, streaming reduces overall channel utilization, by eliminating wakeup signals.

### 7.1.3 Congestion Control

The forwarder employs mechanisms to detect and mitigate congestion, as congestion can cause queues to become full and decrease energy efficiency due to forwarding failures. Congestion is detected when the queue is full, which differs from existing approaches that indicate congestion before the queue is full for three reasons [20]: (i) hop-by-hop recovery prevents dropped messages due to queues overflows, (ii) allowing full use of the queue maximizes streaming benefits, and (iii) congestion feedback utilizes the same network-layer information contained in link acks. Congestion control uses the feedback to adjust transmission rates using an additive-increase, multiplicative decrease control. Because datagrams are not dropped until the next-hop can accept it, congested nodes will appropriately apply back-pressure all the way to the application source if necessary.

### 7.1.4 Quality of Service

The forwarder's mechanisms of reliable forwarding, streaming, and congestion control may induce higher latencies due to forwarding delays or higher queue occupancies. There are times, however, when communication latency is more important than energy efficiency. To address this, we include three basic QoS mechanisms that allow upper layers to specify whether energy efficiency is of concern. First, upper layers must explicitly tag datagrams as latency-tolerant to take advantage of the energy-saving mechanisms. Unmarked datagrams are never delayed for streaming and retransmission timeouts are less conservative. Second, upper layers may tag datagrams as *high priority* to allow the forwarder to evict low priority messages when full. Finally, the forwarder permits *queue reservations* for different traffic classes, allowing the forwarder to provide some level of service to different traffic classes. Information about the datagram is placed in an IPv6 Hop-by-Hop Option header. When no information is known, forwarders assume the datagram is not latency-tolerant, has low priority, and is assigned a default traffic class.

## 7.2 Multicast Forwarder

The multicast forwarder implements a simple controlled flood using Trickle [31]. Trickle's sequence number is included in an IPv6 Hop-by-Hop Option header. Nodes buffer a single datagram for continuous retransmissions until the maximum transmission period is reached. Retransmitting the most recent datagram increases the delivery rates and suppression reduces the cost of forwarding.

# 8. ROUTING

The router is responsible for establishing reachability, forming paths to destinations that minimize some routing metric by maintaining forwarding table entries, as shown in Figure 2. Ad-hoc wireless networks make routing challenging because there is no strictly defined topology. Instead, the router must infer a topology from varying link connectivity and account for links that are neither good nor bad but in between. Resource constraints add to the challenge by limiting how often the router can probe neighbors and how much routing information it can maintain or communicate.

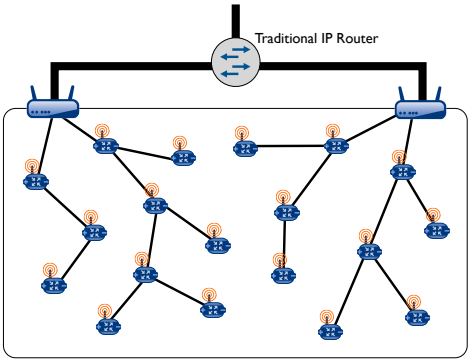



**Figure 5: Utilizing Default Routes.** *By utilizing default routes in both directions, WSN nodes maintain small and constant routing state with low communication overhead.*

Traditional routing protocols for wired and wireless networks are ill-suited for WSNs. State-of-the-art protocols for wired networks rely on complete link-state information that require large amounts of memory [22] or complex synchronization protocols to converge quickly [1]. MANET protocols are designed for ad-hoc networks with high and uncorrelated mobility, relying on floods to discover routes and recover from link failures [8, 36]. Proposed protocols for large-scale ad-hoc networks often reduce state by constraining the routing topology and sacrificing route optimality [52]. Location-based protocols embed location information into the names of nodes so that nodes only need to maintain the locations of their neighbors, but binds names to the underlying topology [29].

Some WSN routing protocols directly address the range of link qualities by incorporating link quality estimates in the routing metric. Some protocols use physical layer measurements provided by (e.g., RSSI or LQI), but these measurements often have high variance and may not provide a good correlation to PRR [40]. Other protocols send control messages to directly compute PRR, but require more time, energy, and state to compute [52]. Recent work shows that these two methods can be combined effectively [21].

Limited resources mean 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 routability but takes no position on how those routes formed or if they are optimal. In this section, we show that the IP framework aligns well with typical WSN constraints and workloads. Routers configure default routes towards a border router, as shown in Figure 5. Border routers maintain host routes to every node in the WSN by learning the default route graph and reversing the links. By concentrating routing effort at the border routers, our routing protocol maintains small and constant state on WSN nodes with almost no control message overhead, yet provides optimal routes when communicating with traditional IP devices outside the WSN.

## 8.1 Default Routes

### *8.1.1 Discovering Routes*

The default router maintains a routing table, separate from the forwarding table. This distinction is important in WSNs, as the routing table is used to evaluate potential routes without being used for forwarding. The router uses RA messages to discover candidate routes and communicate routing information to neighboring nodes. The routing information includes the sender's distance in hops and estimated number of transmissions (ETX) [10] relative to the nearest border router, however other metrics may be included in to support more complex decision algorithms.

### *8.1.2 Managing the Routing Table*

The router inserts potential routes into the routing table with the eventual goal of selecting one as the default route. To gather link quality information for those routes, the router pins the associated entries in the link layer's neighbor table. Doing so also allows the link layer to apply energy-saving optimizations when transmitting to those neighbors. The router uses measured PRR as a link quality estimate, but computing PRR requires state, energy, and time. Nodes inserted into the routing table initially begin with no existing knowledge of the PRR to that node. However, each transmission to a node updates its measured PRR and increases confidence in the link quality estimate.

The router accounts for the range of link quality confidence by sorting the routing table based on *path cost* and *confidence in the link quality estimate*, the top entry providing a good combination of both. In some cases, the router will choose a route that advertises a higher path cost than others simply because it has higher confidence in the link quality estimate. The router removes entries when the link quality estimate drops below a threshold.

### *8.1.3 Selecting Default Routes*

In most cases, the top entry in the routing table is selected as the default route. However, the router may choose to deviate from this choice for two reasons. First, the router may choose to re-route by configuring an alternate route when a few consecutive transmission to the top candidate fails. Second, the router may choose to probe another candidate that may potentially provide a better route to increase confidence in its link quality estimate. The router randomly selects a small fraction of forwarded datagrams to serve as link quality probes and uses hop count information to avoid creating loops. By using existing data traffic, the router does not need to generate explicit control messages to generate link quality estimates. Furthermore, link quality estimates only occur when paths are being utilized and the amount of link quality information scales with the data traffic.

### *8.1.4 Maintaining Route Consistency*

Route information can become inconsistent and lead to routing loops or suboptimal routes. To address this problem, the forwarder tags each datagram with the expected hop count and ETX of the next hop using an IPv6 Hop-by-Hop Option. Receiving a lower than expected hop count signals the possibility of a routing loop. Receiving a significantly different ETX indicates the possibility of inefficient routes, but does not necessarily indicate the presence of routing loops. In both cases, the router resets the RA Trickle timer to update route information more quickly.

Compared to ETX, hop count provides a more effective mechanism for loop detection. Hop count has less variability because it is not subject to varying link qualities on the path. Furthermore, an increase in ETX may only indicate that an increased path cost down stream has not yet fully propagated upstream. The router also uses hop count to avoid routing loops when selecting alternate routes for re-routing or load balancing. The use of hop count when detecting routing loops makes the detection mechanism independent of the route metric and thresholds for detecting inefficient paths.

## 8.2 Host Routes

Border routers maintain host routes to every node in the WSN and route datagrams to WSN nodes by inserting an IPv6 Routing Header that contains a path to the destination. WSN nodes need not maintain any routing information for anything other than their default routes. Because default routes are selected based on their bidirectional connectivity, border routers can easily generate host

routes by learning the default route graph and reversing its links. WSN nodes provide the border router with default route information by including a Record Route Option (RRO) in the IPv6 Hop-by-Hop Option header of any datagram. The RRO contains a list of IPv6 addresses, identifying hosts that have forwarded the datagram. Because all nodes have the same prefix, the RRO only requires 2 bytes per entry. The same compression methods used in Section 5 apply not only to layer 2.5, but also to layer 3. Nodes that have recently forwarded an RRO suppress their own transmissions, allowing the overhead to scale with the number of leaves rather than the number of nodes.

In some cases, multiple border routers may be desirable to increase redundancy in egress points for the WSN or energy efficiency by reducing the average number of hops to an egress point. Because nodes individually select default routes to a single border router, the each border router may only contain host routes for a subset of the WSN. Border routers must effectively communicate their host route state to other border routers. This may be done by advertising host routes using standard IP routing protocols or by proxying Neighbor Discovery [46]. This choice in utilizing standard mechanisms emphasizes the advantage of directly supporting the IPv6 architecture.

## 8.3 Routing Overhead

The routing protocol configures both default and host routes with minimal resource requirements. 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 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 RAs and utilizes ambient data traffic to generate link quality estimates, enable loop and sub-optimal route detection, and provide default route information to border routers.

By pushing routing state to the border routers, the routing state on WSN nodes is small and constant - that required for configuring default routes (64 bytes). Routing state at border routers scales with the number of WSN nodes, but border routers generally have greater memory capacity. In Section 10, we provide empirical data for energy and memory requirements of an actual application deployment.

The tradeoff with minimal state and communication is routing stretch. The worst case routing stretch is bounded by $2D$ and occurs for neighboring nodes furthest from the border router, where $D$ is the network diameter. However, the worst-case is the easiest case to optimize for. Discovering and inserting neighbors into the forwarding table reduces the worst-case bound to $D$. Generalizing this includes $n$-hop neighbors, reducing the worst-case bound to $\frac{2D}{1+n}$. Due to the physical nature of WSNs, far away nodes are less likely to communicate and it may not be worth the additional cost of maintaining near-optimal routes across all nodes. Nodes could also provide more complete link state information to the border router and the border router could assist in providing forwarding table entries to WSN nodes.

# 9. TRANSPORT LAYER

The transport layer provides end-to-end communication between application end points. It is hard to ignore the ubiquity of UDP and TCP, and for this reason, it is almost a requirement to implement them. Doing so allows WSN nodes to communicate end-to-end with existing and unmodified IP devices. But because we've preserved the IPv6 network architecture, protocol layering, and high best-effort datagram delivery with low latency, implementing RFC-compliant UDP and TCP is straightforward. But as with traditional IP networks, other transport protocols may be invented that are designed specific applications. Having an established architecture only helps to frame the problem.

| 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 1: ROM and RAM Requirements for Communication Components.**

# 10. EVALUATION

To evaluate our IPv6 architecture for WSNs, we implement all of the functionality described in this paper from the link through the transport layer. We evaluate both high-level systems aspects of our IPv6-based network architecture and in-depth aspects of critical details. Our implementation is built using TinyOS 2.x [49] on the TelosB platform [39]. 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.

A complete, production-quality implementation that provides all of the functionality described in this paper 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 runtime required to support the IPv6 network stack (e.g., OS-level services). The breakdown for communication-specific components is shown in Table 1 and are similar to uIP [14].

We use the implementation to conduct a detailed power analysis of the IPv6-based network architecture. To evaluate the sensitivity to a number of parameters, we build a power model for the network layer using empirical data from link-layer communication primitives. We then validate the power model using a real-world low-rate data collection application and show that it outperforms what has been demonstrated to date.

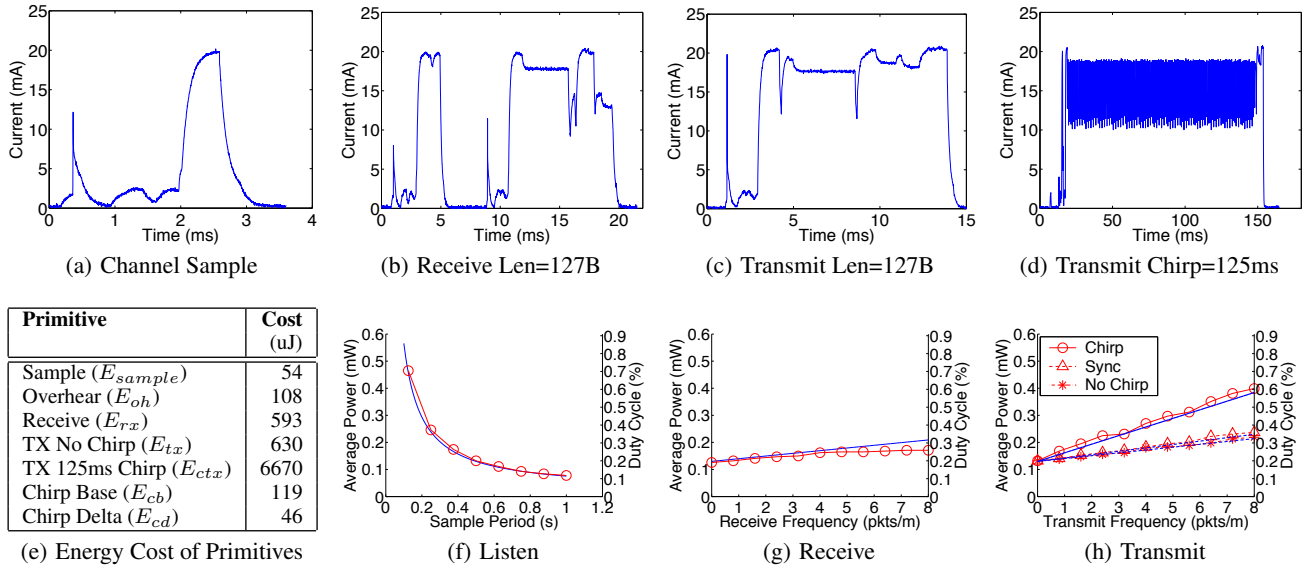
## 10.1 Link Energy Cost

We model the average power draw of the complete system $P_{total}$ using the listen, receive, and transmit costs by

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

Listen represents the baseline power draw for a node and places a upper bound on the node's lifetime. We model the average power draw for listening $P_{listen}$ using the sleep power draw $P_{sleep}$, the channel sample frequency $f_{sample}$, and channel sample energy cost $E_{sample}$ by

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




(a) Channel Sample (b) Receive Len=127B (c) Transmit Len=127B (d) Transmit Chirp=125ms

| Primitive | Cost (uJ) |
|---|---|
| Sample ($E_{sample}$) | 54 |
| Overhear ($E_{oh}$) | 108 |
| Receive ($E_{rx}$) | 593 |
| TX No Chirp ($E_{tx}$) | 630 |
| TX 125ms Chirp ($E_{ctx}$) | 6670 |
| Chirp Base ($E_{cb}$) | 119 |
| Chirp Delta ($E_{cd}$) | 46 |

(e) Energy Cost of Primitives (f) Listen (g) Receive (h) Transmit

**Figure 6: Link Power Model.** *Figures (a)-(d) show the TelosB's instantaneous current draw over time for representative link primitives. Figure (e) shows the energy cost of each primitive by integrating the current-draw profile. Figure (f) shows the average power draw when only listening. Figures (g) and (h) show the average power draw when receiving and transmitting relative to the packet rates and assumes $T_{sample} = 0.5s$. The model predicts the measured values to within 2% on average.*

In this model, we always assume worst-case frame sizes of 127 bytes for both receive and transmission costs. Receive represents the average power draw due to receiving data frames. The receive cost is independent of chirp length and we model the cost using the reception frequency $f_{rx}$ and the reception energy cost $E_{rx}$ by

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

Transmit represents the average power draw due to transmitting data frames. The transmit cost is dependent on the chirp duration, which is equal to the channel sample period when the receiver's schedule is unknown or sending broadcasts. When the receiver's schedule is known, the chirp length is a guard time that increases over time but resets when receiving an ack from that neighbor. The minimum guard time for our implementation is 2ms and we assume a frequency tolerance $f_\Delta = \pm 20$ppm. We simplify the power model for transmissions by accounting for the frequency of broadcast messages $f_{txb}$ and the frequency of unicast messages $f_{txu}$. The transmit costs are given by

$$P_{txb} = f_{txb}\left(E_{tx} + E_{cb} + E_{cd}\frac{1}{f_{sample}}\right)$$
$$P_{txu} = f_{txu}\left(E_{tx} + E_{cb} + E_{cd}\left(2 + \frac{f_\Delta}{f_{txu}}\right)\right).$$

Using empirical measurements for low-level communication primitives, we build a model that accurately reflects average power draw of the entire hardware platform. The communication primitives at the link layer are: channel sample, overhear, receive, and transmit. We measured the current draw profile for each primitive by using an oscilloscope to measure the voltage drop over time across a sense resistor placed in series with the TelosB's power supply. We plot representative profiles in Figures 6(a)-6(d).

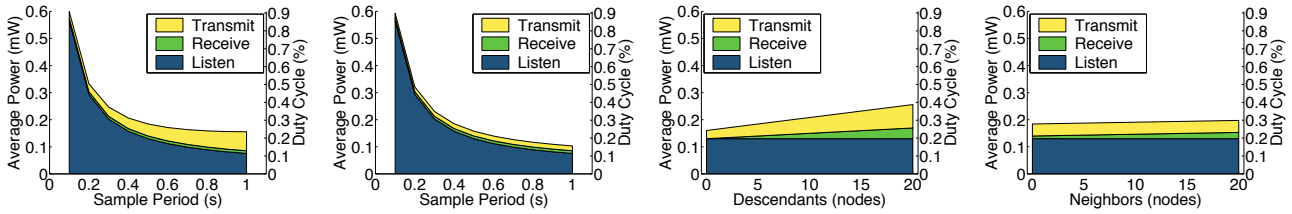
Integrating the current-draw profile and multiplying by the supplied voltage gives the energy cost of each primitive, shown in Figure 6(e). The rendezvous time in chirp frames reduces the receive cost and makes the cost independent of the chirp duration. The inclusion of addressing information in chirp frames significantly reduces overhearing costs. Transmission without chirps is slightly more costly than receiving for the same data frame length due to the carrier-sense and ack frame processing. As expected, transmissions with chirps are costly and is dominated by the chirp length. By measuring the energy cost for two different chirp lengths, we can compute the fixed $E_{cb}$ and marginal $E_{cd}$ costs for sending chirps.

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 Figures 6(f)-6(h), 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. Much of the savings is due to the low cost of channel samples, which only requires the receiver to be on for 640 us.[2] X-MAC has a channel sample cost 30 times larger because it inserts acks into the preamble stream and does not support synchronization [6]. 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 [49]. SCP-MAC presented similar but limited results for the MicaZ platform [55].

## 10.2 Network 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 $\tau_h = 30m$ (the maximum Trickle period). RRO messages are unicast periodically, with frequency $f_{rr}$. We do not model message suppression.

[2]From the CC2420 datasheet [45], 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.




(a) Sample Period (Router) (b) Sample Period (Host) (c) Descendants (d) Density

**Figure 7: Network Maintenance Costs.** *Each figure shows the power draw of listen, receive, and transmit. Figure (a) shows power draw of a router over $T_{sample}$ with $D = 5$ and $N = 5$. Figure (b) shows power draw for a host-only node over $T_{sample}$ with $D = 0$ and $N = 5$. Figure (c) shows power draw for a router over $D$ with $T_{sample} = 0.5s$ and $N = 5$. Figure (d) shows power draw for a router over $N$ with $T_{sample} = 0.5s$ and $D = 5$.*

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 $f_{ra}$. The relationships are by

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

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 datagrams nor send RA messages) as a function of channel sample period, descendants, and node density in Figure 7. Figure 7(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. 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 7(b). Figure 7(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 7(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 with $D$.

## 10.3 Application Energy Cost

To evaluate the average power draw of a node in an application environment, we consider the low rate data-collection workload typical to many WSN 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). Augmenting the model, the frequency relationships are given by

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

We plot average power draw relative to the datagram generation rate in Figure 8.

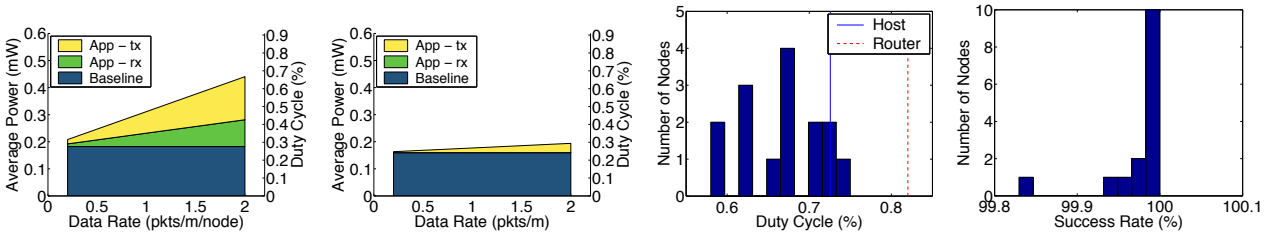
Figures 8(a) and 8(b) display the average power draw of a router and host-only node, breaking out the baseline power required for maintaining the network. In both cases, the network and link layers account for a significant fraction of the average power draw and listening forms the largest component. 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.

| Deployment | Year | RP (m) | DC | Latency (s) | DRR |
|---|---|---|---|---|---|
| GDI [43] | 2003 | 20 | 2.2% | 0.54-1.085 | 28% |
| Redwoods [48] | 2004 | 5 | 1.3% | 300 | 49% |
| FireWxNet [23] | 2005 | 15 | 6.7% | 900 | 40% |
| WiSe [44] | 2006 | 30 | 1.6% | 60 | 33% |
| Dozer [7] | 2007 | 2 | 1.67% | 15 | 98.8% |
| SensorScope [4] | 2008 | 2 | 1.11% | 120 | 95% |
| IPv6 | 2008 | 1 | 0.65% | 0.125 | 99.98% |

**Table 2: Performance of prior WSN deployments.** *Report period (RP) is the time delay between data transmissions. Duty cycle (DC) 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 data reception rate (DRR) is the fraction of data received at the collection point.*

To validate the power model, we used a real-world home monitoring application. The application consists of 15 nodes deployed in refrigerators, solar power inverters, outdoors, and indoors in or near the intended sense point. 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 7 nodes within 1-hop of the border router, the remaining half being 2 to 3 hops away. The link was configured with $T_{sample} = 0.125$s. Using this information, we compute the average duty-cycle for each node. 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 lower than predicted. We achieved 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%.

We compare results from our application deployment to published data from prior WSN deployments in Table 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 [42]. Similarly, ZigBee [56] does not sup-




(a) Router (b) Host-Only (c) Duty Cycle (d) Success Rate

**Figure 8: Application Power Consumption.** *Figures (a) and (b) show average power draw for a router and host, respectively, over $f_{app}$ with $T_{sample} = 0.5s$, $D = 5$, and $N = 5$. Figures (c) and (d) show data from a deployed home monitoring application over a continuous 4-week period with $T_{sample} = 0.125s$. Figure (c) shows a histogram of average duty-cycles over all four weeks with the model's prediction for host-only and router nodes. We achieved better results than predicted, as the model does not account for suppression. Figure (d) shows a histogram of the packet success rate by the server.*

port duty-cycled operation for forwarding nodes and also does not provide end-to-end IP interoperability. uIP [14, 16] showed the feasibility of IP in WSNs, but existing work focused more on CPU performance rather than overall system energy consumption and networking performance in a typical WSN setting.

## 10.4 Goodput and Latency

We also evaluate the goodput and latency performance of link-local and global communication over UDP and TCP. Streaming optimizations allowed our implementation to achieve higher goodput than other duty-cycled solutions. Figure 9(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 throughput is still much lower than the theoretical 250 kbps, due to the use of AES-128 encryption and software-based acks. Figure 9(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(c) and 9(d) shows that the expected communication latency measured with Ping is linear with $T_{sample}$. With synchronization, communication latency depends on when the next channel sample occurs at the receiver.

## 10.5 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 a particular architecture. The question remains: what is the cost of an IPv6-based network architecture? The adaptation layer requires 6-11 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.

# 11. ARCHITECTURAL RETROSPECTIVE

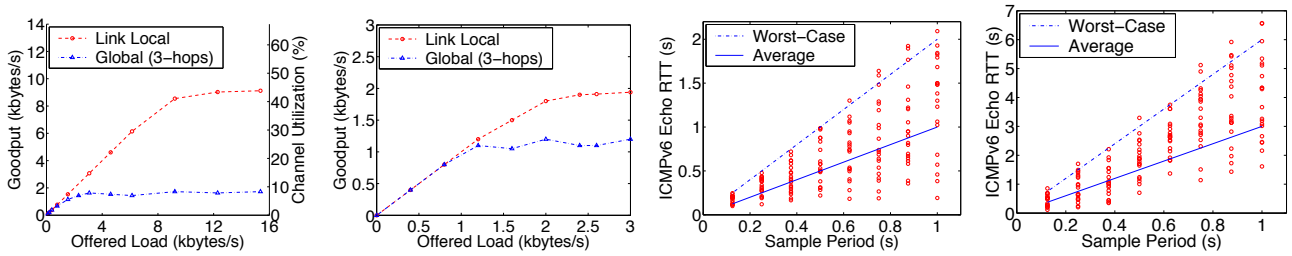
In the past decade, wireless sensor network research and Internet architecture, especially IPv6, have both progressed substantially. Thus, it makes sense to revisit the assumptions that have formed the basis of so much WSN research. By constructing a high quality WSN using IPv6 as a foundation, we find that the two areas of development are in fact highly complementary. Most of the unique requirements of WSNs 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.

WSN 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 WSN space provide elegant solutions to problems that have not been well addressed by conventional IETF approaches. Not only do we find a close analog to the structures for dissemination and collection that are so common in WSNs, but Trickle mechanisms provide an elegant means of reaching consensus, responding quickly to changes, and becoming passive as things quiesce.

In particular, we find that we need not forsake the many virtues of layering to meet the severe resource constraints of WSN 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 WSNs. Indeed, if WSNs 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 could 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




(a) UDP Goodput (b) TCP Goodput (c) Link Local Ping (d) Global Ping

**Figure 9: Goodput and Latency.** *Figures (a) and (b) show goodput in kilobytes/second and channel utilization for UDP communication within link-local and global scope, respectively. Figures (c) and (d) show round-trip time for ICMPv6 Echo Request/Reply messages over link-local and global scope, respectively.*

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 this goal. 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 WSNs, 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 WSNs 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 be used to 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 WSNs 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.

## 12. CONCLUSION

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 WSNs to other IP networks (i.e., firewalls, proxies, caches, etc.). The presence of a broad-based IPv6 network architecture in the same technological context as solutions that lack any architecture means that we can re-examine a much broader set of research questions. The question of in-network processing, aggregation, and query processing are not constrained by the IPv6 network architecture. However, 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. Many long-standing WSN questions can be examined in a more general setting, and if the answer is open a TCP connection or send a UDP datagram, that answer is acceptable as well.

Similarly, 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. For example, many suggest that neither UDP or TCP transports are quite right for periodic readings, reports, and configuration actions. 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.

## 13. ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under grants #0435454 ("NeTS-NR") and #0454432 ("CNS-CRI"). Many thanks to Matt Welsh, Prabal Dutta, and our shepherd Adam Dunkels for their useful feedback.

## 14. REFERENCES

[1] R. Albrightson, J. Garcia-Luna-Aceves, and J. Boyle. Eigrp-a fast routing protocol based on distance vectors. 1994.

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

[3] H. Balakrishnan, S. Seshan, E. Amir, and R. 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.

[4] G. Barrenetxea, F. Ingelrest, G. Schaefer, M. Vetterli, O. Couach, and M. 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.

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

[6] M. Buettner, G. V. Yee, E. Anderson, and R. 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.

[7] N. Burri, P. von Rickenbach, and R. 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.

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

[9] A. Conta, S. Deering, and M. Gupta. Internet control message protocol (icmpv6) for the internet protocol version 6 (ipv6) specification. RFC 4443 (Draft Standard), Mar. 2006. Updated by RFC 4884.

[10] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. 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.

[11] S. Deering, B. Haberman, T. Jinmei, E. Nordmark, and B. Zill. Ipv6 scoped address architecture. RFC 4007 (Proposed Standard), Mar. 2005.

[12] S. Deering and R. Hinden. Internet protocol, version 6 (ipv6) specification. RFC 2460 (Draft Standard), Dec. 1998.

[13] 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.

[14] A. 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.

[15] A. Dunkels, F. Österlind, and Z. 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.

[16] A. Dunkels, T. Voigt, and J. 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, Jan. 2004.

[17] C. T. Ee, R. Fonseca, S. Kim, D. Moon, A. Tavakoli, D. Culler, S. Shenker, and I. 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.

[18] 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.

[19] D. Estrin, R. Govindan, J. Heidemann, and S. 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.

[20] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. Netw.*, 1(4):397–413, 1993.

[21] R. Fonseca, O. Gnawali, K. Jamieson, and P. Levis. Four bit wireless link estimation. In *HotNets VI: Proceedings of the Sixth Workshop on Hot Topics in Networks*, 2007.

[22] M. Gupta and N. Melam. Authentication/Confidentiality for OSPFv3. RFC 4552 (Proposed Standard), June 2006.

[23] C. Hartung, R. Han, C. Seielstad, and S. 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.

[24] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. *SIGPLAN Not.*, 35(11):93–104, 2000.

[25] J. W. Hui and D. E. Culler. Extending ip to low-power, wireless personal area networks. *Internet Computing, IEEE*, 12(4):37–45, July-Aug. 2008.

[26] B. Hull, K. Jamieson, and H. 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.

[27] H. Huo, H. Zhang, Y. Niu, S. Gao, Z. Li, and S. Zhang. Msrlab6: An ipv6 wireless sensor networks testbed. *Signal Processing, 2006 8th International Conference on*, 4:–, 16-20 2006.

[28] J. Jubin and J. D. Tornow. The DARPA packet radio network protocols. *Proceedings of IEEE*, 75(1):21–32, Jan. 1987.

[29] B. Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2000)*, pages 243–254, 2000.

[30] P. Levis, S. Madden, D. Gay, J. Polastre, R. Szewczyk, A. Woo, E. Brewer, and D. Culler. The emergence of networking abstractions and techniques in tinyos. 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.

[31] P. Levis, N. Patel, D. Culler, and S. 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.

[32] K. Mayer and W. 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.

[33] P. P. Mishra and H. Kanakia. A hop by hop rate-based congestion control scheme. *SIGCOMM Comput. Commun. Rev.*, 22(4):112–123, 1992.

[34] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. Transmission of ipv6 packets over ieee 802.15.4 networks. RFC 4944 (Proposed Standard), Sept. 2007.

[35] T. Narten, E. Nordmark, W. Simpson, and H. Soliman. Neighbor discovery for ip version 6 (ipv6). RFC 4861 (Draft Standard), Sept. 2007.

[36] C. Perkins. Ad hoc on-demand distance vector (AODV) routing. RFC 3561 (Experimental), July 2003.

[37] J. Polastre, J. Hill, and D. 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.

[38] J. Polastre, J. Hui, P. Levis, J. Zhao, D. Culler, S. Shenker, and I. 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.

[39] J. Polastre, R. Szewczyk, and D. 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.

[40] J. Polastre, G. Tolle, and J. 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.

[41] N. Sastry and D. Wagner. Security considerations for IEEE 802.15.4 networks. In *ACM Workshop on Wireless Security (WiSe 2004)*, October 2004.

[42] Sensinode. Nanostack. http://sourceforge.net/projects/nanostack/.

[43] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. 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.

[44] K. Szlavecz, A. Terzis, R. MusÇŐloiu-E., J. Cogan, S. Small, S. Ozer, R. Burns, J. Gray, and A. 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.

[45] Texas Instruments. Cc2420: 2.4 ghz ieee 802.15.4 / zigbee-ready rf transceiver. http://focus.ti.com/lit/ds/symlink/cc2420.pdf, Mar. 2007.

[46] D. Thaler, M. Talwar, and C. Patel. Neighbor Discovery Proxies (ND Proxy). RFC 4389 (Experimental), Apr. 2006.

[47] S. Thomson, T. Narten, and T. Jinmei. Ipv6 stateless address autoconfiguration. RFC 4862 (Draft Standard), Sept. 2007.

[48] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, and W. 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.

[49] University of California at Berkeley. Tinyos. http://www.tinyos.net/, 2004.

[50] T. van Dam and K. 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.

[51] A. Woo and D. 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.

[52] A. Woo, T. Tong, and D. 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.

[53] Y. Xu, J. Heidemann, and D. 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.

[54] W. Ye, J. Heidemann, and D. 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.

[55] W. Ye, F. Silva, and J. 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.

[56] ZigBee Alliance. Zigbee. http://www.zigbee.org/.