# IPv6 in Low-Power Wireless Networks

**Article** *in* Proceedings of the IEEE · December 2010

DOI: 10.1109/JPROC.2010.2065791 · Source: IEEE Xplore

**2 authors:**

Jonathan W. Hui
University of California, Berkeley
**33** PUBLICATIONS **7,756** CITATIONS
SEE PROFILE

David E. Culler
University of California, Berkeley
**422** PUBLICATIONS **79,415** CITATIONS
SEE PROFILE

INVITED PAPER

# IPv6 in Low-Power Wireless Networks

*Recent developments have made it possible, pragmatic, and efficient for the IPv6 protocol to be applied to low-power multihop wireless networks, especially regarding the metrics—such as low memory footprint, high reliability, and low energy use—that are most important for embedded applications.*

By Jonathan W. Hui and David E. Culler, *Fellow IEEE*

**ABSTRACT** | With deeply embedded wireless sensors, a new tier of the Internet is emerging that will extend into the physical world. These wireless sensor nodes are expected to vastly outnumber conventional computer hosts as we see them today, but their strict resource constraints are unlike other technologies already common to the Internet. As wireless sensor network research took off, many in the field eschewed the use of IP as inadequate and in contradiction to the needs of wireless sensor networking. Since then, the field has matured and IP has evolved. In this paper, we show that the convergence of Internet Protocol Version 6 (IPv6) and low-power multihop wireless networking is possible, pragmatic, and efficient—especially in regard to the metrics that matter most for embedded applications, low memory footprint, high reliability, and low energy usage. Using real commercial deployments, we show that it is possible to simultaneously achieve an average duty cycle of $<$ 0.4%, average message delivery rate of $>$ 99.9%, and average per-hop latency of $<$ 125 ms over 12 months in different environments.

**KEYWORDS** | Ad hoc networks; IEEE 802.15.4; internet; internetworking; IP networks; IPv6; low-power wireless networks; mesh networks; ROLL; RPL; wireless mesh networks; wireless networks; wireless sensor networks; 6LoWPAN

## I. INTRODUCTION

The Internet of the physical world has arrived. Some 40 years after it was created to connect computers to one another and allow file transfer, remote login, and access to distant computation, we find computational processing embedded in almost every device, machine, appliance, and instrument. And, they are increasingly able to communicate. These "hosts" hardly resemble their classic Internet forebearers, and rather than the human-generated information and documents that are exchanged over the classic Web, these deeply embedded computers present physical information—sensor readings, observations, actions, and events that occur over time at particular points in the real world. The web of real world data is used to optimize production, improve safety, and reduce energy consumption, waste, and pollution.

The emergence of this new tier of the Internet has largely been enabled by a decade or so of intense research on low-power wireless embedded networks, or *sensornets*. But early on, that thrust explicitly eschewed the design principles and constraints of the Internet architecture, arguing that conventional layering was impractical for the resource constrained devices that were being embedded in the physical world; that the underlying physical communication structure was essential to applications that would utilize such information and should not be abstracted away; and that without a human being in close attendance, these devices would need to configure themselves into networks without manual intervention. Ironically, this freedom of thought produced innumerable good ideas locked away in disjoint, noninteroperable little stovepipes with little opportunity to impact the real world. At the same time, portions of the Internet design community were pushing on these very issues of accommodating huge numbers of hosts, autoconfiguration, and extensibility to embrace unanticipated innovation in IP version 6.

In this paper, we show that these two lines of development are, in fact, highly complementary. They can be brought together to obtain a powerful confluence of

Manuscript received November 17, 2009; revised July 20, 2010; accepted August 2, 2010. Date of publication September 13, 2010; date of current version October 20, 2010.

**J. W. Hui** is with Arch Rock Corporation, San Francisco, CA 94107 USA (e-mail: jhui@archrock.com).

**D. E. Culler** is with the Computer Science Division, University of California at Berkeley, Berkeley, CA 94720 USA (e-mail: culler@cs.berkeley.edu).

Digital Object Identifier: 10.1109/JPROC.2010.2065791

0018-9219/$26.00 ©2010 IEEE

design. The best ideas arising from sensor networks can be realized within the Internet architecture, and in doing so they bring value across a range of communication link technologies and technological changes, and they can apply to a diverse array of applications (even many that have yet to emerge) because they support widely useful primitives, rather than highly specific solutions. However, this confluence is not by any means automatic. It involves working through a broad swath of the Internet stack and carefully adapting or extending its elements to meet the austere requirements of the embedded domain.

## II. BACKGROUND

### A. Sensornet Research

Over the past decade, sensornet researchers tackled a number of important networking challenges. With the observation that radio idle-listening dominates system energy consumption, numerous duty-cycled link protocols based on sampled listening [1] and scheduling [2] were proposed. Because the radio network topology must be inferred by observing communication with neighboring nodes, significant work went into link quality estimation that utilized link-layer information [3], physical-layer information [4], as well as their combination with routing information [5]. While the Internet was successful at utilizing only end-to-end feedback mechanisms, sensornet researchers demonstrated the need for hop-by-hop feedback to achieve greater visibility into the network, react quicker to local conditions, and avoid the high costs of end-to-end mechanisms [6], [7]. The need to disseminate information across the network shows up in a wide variety of application- and network-layer scenarios, and the Trickle algorithm provided a means to do so in an efficient and density-aware manner that avoided the broadcast-storm problem [8].

Much of this work did not constrain itself 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 [9]–[11]. But by being agnostic to the network protocol in use, these architectures did not define broader networking issues of discovery, naming, addressing, configuration, and management.

### B. Non-IP Network Solutions

Many past standardization efforts for low-power wireless networking (e.g., ZigBee [12], Z-Wave [13], and WirelessHART [14]) also believed that IP was too large and ill-suited for the needs of sensornets, leading them to develop *ad hoc* solutions not based on IP. Without a common network layer, these systems require an application-layer gateway to communicate with other systems or even the wider Internet. These application gateways are complex to design and manage. Mapping between two protocols is not straightforward, requiring significant functional and semantic translation. Mechanisms that do not have a similar mapping on both sides must be dropped. Because such translations are often stateful, application gateways typically represent a single point of failure for the sensornet.

Speaking a common network protocol end-to-end provides much greater deployment flexibility and robustness. Stateless routers can deliver messages using any number of paths, creating greater redundancy. New application protocols can be introduced without changing the network infrastructure. The network can also be composed of different link technologies (e.g., 802.15.4, WiFi, and Ethernet), applying them where they are most appropriate. This flexibility and scalability is more important than ever as we grow the Internet into new operational domains.

### C. Internet Protocol Version 6

Internet Protocol Version 6 (IPv6) is the designated successor of IPv4 as the network protocol for the Internet [15]. To overcome the dwindling unallocated address space and in anticipation that networked devices will vastly outnumber conventional computer hosts, IPv6 expands the IP address space from 32 to 128 b. The most widely used link technologies evenly split the IPv6 address space into a subnet prefix that uniquely identifies the subnet within the Internet and an interface identifier that uniquely identifies an interface within the subnet. Recognizing the growth in link throughput and to reduce packet transmission overhead, IPv6 increases the minimum maximum transmission unit (MTU) requirement from 576 to 1280 B. To increase protocol efficiency and eliminate the need for *ad hoc* link-layer services to bootstrap the subnet, IPv6 includes scoped multicast as an integral part of its architecture. Core IPv6 components, such as neighbor discovery (ND) [16], use link-local scoped multicast for address resolution and router discovery.

While IPv6 was designed for more capable nodes in mind, we claim that *IPv6 is better suited to the needs of sensornets than IPv4 in every dimension*. The large IPv6 addresses are more amenable to cross-layer compression. Inclusion of necessary bootstrapping functionality [e.g., dynamic host configuration protocol (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. Realizing the need to evolve, IPv6 network protocols were designed to support protocol options. Local IPv6 protocols that made link-specific assumptions were specified only for those links, allowing adaptations for different link technologies. This generality and extensibility of the IPv6 networking protocols allows us to utilize mechanisms that have become so pervasive in the sensornet community. Sampled listening, hop-by-hop recovery, and collection routing are just a few mechanisms that allow us to implement an

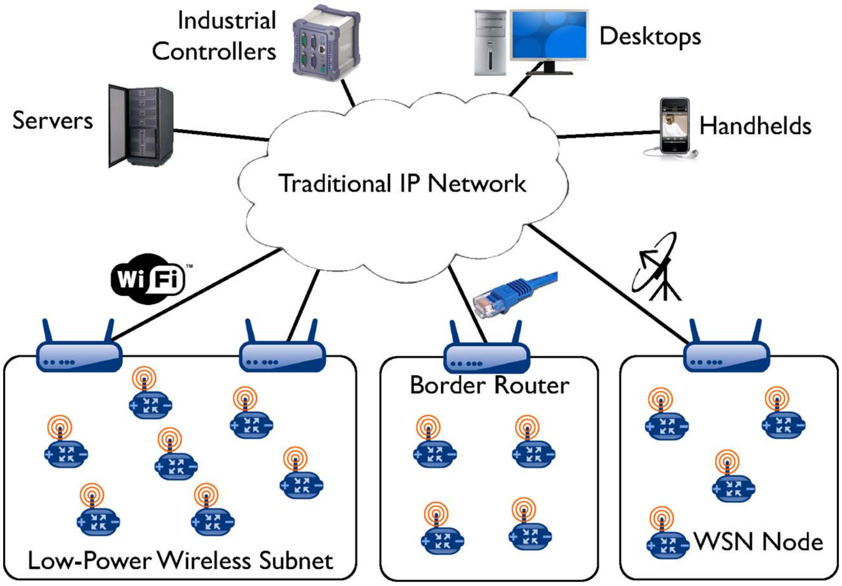


**Fig. 1.** *IPv6 network architecture.*

IPv6-based network that is no less efficient than existing non-IP-based solutions.

## III. NETWORK ARCHITECTURE

Incorporating sensornets into an IP network is no different than with any other network technology. As shown in Fig. 1, a sensornet simply forms another subnet within an IP-connected network. We define a *sensornet subnet* by the set of nodes that can communicate within the sensornet. *Edge routers* connect sensornet subnets to other subnets possibly built on different link technologies. While the sensornet network may only support IPv6, the edge router may implement IPv4-to-IPv6 translation [17]. But most importantly, edge routers only forward datagrams at the network layer and do not maintain any application-layer state.

From an internetworking perspective, we have changed nothing about IP. The challenges come from supporting IP and its essential services within a sensornet subnet. While IP has shown great success in utilizing a wide range of link technologies (including wireless), the resource constraints inherent to sensornets challenge common assumptions.

### A. Route-Over Versus Mesh-Under

An IP link is defined by those nodes that are reachable over a single IP hop. The most widely used link technologies (e.g., Ethernet, WiFi, and point-to-point) provide a *single broadcast domain* where all communication is *transitive* within the subnet (if A can send to B and B can send to C, then A can send to C) and any interface can transmit a single datagram to any number of interfaces within the subnet. Today, Ethernet and WiFi typically emulate the above properties through link-layer forwarding and routing. Ethernet subnets are often composed of multiple physical media connected by Ethernet switches. WiFi subnets are composed of clients that can only communicate directly with an access point at the physical layer due to physical limitations of radio communication and security mechanisms.

At the core of bringing IPv6 to sensornets is whether to emulate a single broadcast domain. This has been referred to as the "mesh-under versus route-over" debate—*mesh-under* referring to the use of link-layer mesh routing protocols to emulate a single broadcast domain and *route-over* referring to the use of IP routing to maintain reachability between nodes within a sensornet subnet.

Conceptually, a single broadcast domain simplifies network-layer services required to form and maintain a subnet. Any node can send a single IP message to any subset of nodes in the subnet without any need to understand the physical topology. This simplifying property is used by IPv6 ND to query whether an IP address is in use, determine the link address for a given IP address, and redirect traffic to other neighbors [16]. Each of these mechanisms is implemented through a single multicast request to neighboring nodes and a unicast response.

Emulating a single broadcast domain was largely successful with Ethernet and WiFi due to their relatively high capabilities and simple network topologies. There is little issue in providing subnet-wide multicast and both Ethernet and WiFi can easily and reliably deliver any subnet-wide multicast traffic generated by the network layer. Furthermore, there is little variation in link-layer communication properties between different pairs of nodes within the subnet. In particular, a host need not consider the link topology in selecting among multiple default routers. A host simply assumes that the link layer can provide the same level of service regardless of which default router it chooses.

However, emulation mechanisms have not always been successful, especially with more complex link technologies. Subnet emulation for asynchronous transfer mode (ATM) networks did not take hold because they could neither adequately hide nor give necessary visibility into the complex dynamics of connection-based communication [18]. For example, routing control functions at both the link and network layers interact poorly. For example, when communication fails at the physical layer, the link-layer routing protocol will react by discovering new routes. At the same time, without visibility into the link layer, the network layer may prematurely detect communication failure and generate additional messages to select a new default router, only to make matters worse. The link layer then must configure routes to the new destination, wasting any existing work in repairing routes to the old destination.

Like ATM networks, dynamic radio properties typical to sensornets lead to more complex forwarding and routing dynamics than those observed with Ethernet or WiFi. Strict resource constraints make it difficult to hide the complex dynamics of forwarding and routing within a sensornet. IP-based protocols would not have the necessary visibility to make informed decisions based on the underlying radio topology. Unlike Ethernet or WiFi, communication properties between any two pair of nodes can vary greatly especially when their path lengths differ by multiple hops. Basic tasks such as determining reachability and selecting a default router must take into account the underlying radio topology.

Supporting a broadcast service across multiple radio hops within a sensornet is also costly. A simple multicast message sent to the link-local all-nodes multicast address would require delivery to all nodes within the sensornet subnet. While it may be possible to reduce transmission costs by maintaining a multicast topology, maintaining the topology itself also carries a significant cost. Furthermore, every node within the sensornet must incur the cost of receiving the message.

The lack of visibility also removes any ability for IP-based protocols and applications to utilize the subnet's spatial structure to optimize performance. Existing work in data fusion that aggregates information from multiple sources along a forwarding topology would no longer apply.

The challenges in emulating a single broadcast domain led us to a "route-over" sensornet subnet architecture that equates a broadcast domain with the radio transmission range. The result is a sensornet subnet that may be composed of overlapping broadcast domains where each radio hop is an IP hop, as shown in Fig. 2. By limiting the broadcast domain to the physical properties of the radio, IP-based protocols and applications can operate in concert with the spatial structure of the network.

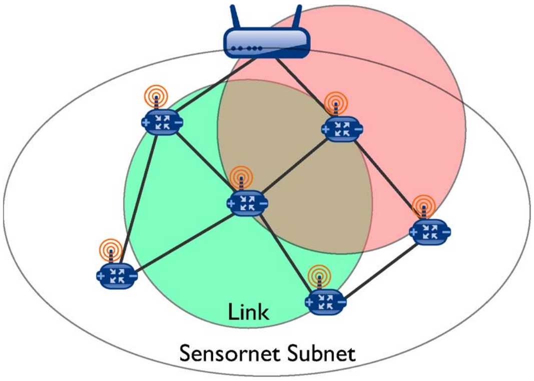


**Fig. 2.** *Subnet composed of overlapping link-local scopes.*

### B. IPv6 Address Architecture

Every IPv6 address (other than the unspecified address) has a scope within which the address may be used as a unique identifier for an interface [19]. RFC 4007 defines two scopes: *link local* for uniquely identifying interfaces on a single link and *global* for uniquely identifying interfaces across the Internet. The extent of a link is defined as those nodes that can communicate at the link layer [15].

By avoiding IP link emulation, the traditional IPv6 address architecture is no longer applicable to sensornet subnets. While the topological span of an IP link is defined by those interfaces within radio range, a link-local address assigned to an interface must be unique among interfaces reachable within two hops or more. Because time-varying radio characteristics make the set of two-hop neighbors an ill-defined set that changes over time, the extent of uniqueness is best expanded to that of the entire sensornet subnet.

As a result, our IPv6 address architecture for sensornet subnets defines the link-local scope as those interfaces reachable within a single radio range and all addresses (including link-local addresses) within the sensornet must be unique across the sensornet subnet.

## IV. IPv6 ADDRESS AUTOCONFIGURATION

In IPv6, nodes may autoconfigure an IPv6 address two ways: 1) statelessly by combining a 64-b IEEE EUI-64 unique identifier with an IPv6 address prefix (e.g., link-local or subnet ID) server, or 2) by using DHCPv6 to assign an address. In both cases, the specifications require nodes to verify the address' uniqueness. RFC 4862 defines a duplicate address detection (DAD) mechanism for verifying

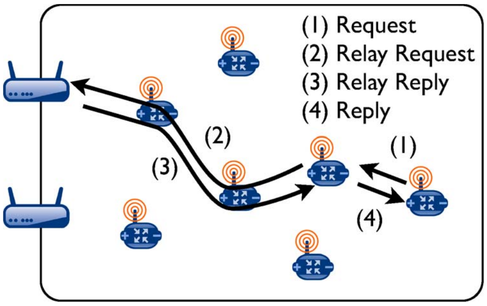


**Fig. 3.** ***DHCPv6 request/reply exchange.***

uniqueness of an address within a link [20]. On links that support a single broadcast domain, performing DAD is done simply by multicasting a query to all link-local nodes asking if an address is in use. However, because sensornet subnets are composed of overlapping link-local scopes, a single link-local multicast is not sufficient.

One option to ensure IPv6 addresses uniqueness is by extending the traditional DAD multicast to cover the sensornet subnet, but doing so requires a costly multihop flood and routing a response back to the source if a duplicate is found. Alternatively, each subnet could avoid the need for multicast by implementing a central IPv6 address registration service that maintains all addresses in use within the subnet. Before using an address, a node must attempt to register an address with the service. Nodes should specify a lifetime with each registration and periodically refresh their registrations for robust operation.

However, implementing an IPv6 address registration service would duplicate much of the core functionality already provided by DHCPv6. A DHCPv6 server maintains entries for all addresses in use within its domain. Nodes obtain a leased address by making requests to the DHCPv6 server and may renew leases over time. If the DHCPv6 server cannot be reached within a single IP hop, neighboring sensornet routers can proxy DHCPv6 messages so that nodes do not have to use a global address before its uniqueness is verified, as shown in Fig. 3.

Because DHCPv6 already implements the mechanisms needed to ensure uniqueness of IPv6 addresses, we do not implement a separate *duplicate detection* service. While IPv6 does require duplicate detection for addresses assigned by DHCPv6, IPv6 does allow disabling of DAD if the costs outweighs the benefits. The addition of an IPv6 address registration service does not provide any real benefit. A centralized IPv6 address registration service suffers the same common failure modes as DHCPv6. For example, having multiple instantiations can lead to duplicate address assignments. Falling back to our initial approach of utilizing multihop multicast is far too costly.

## V. IPv6 NEIGHBOR DISCOVERY

The IPv6 ND protocol supports mechanisms necessary for neighboring IPv6 nodes to communicate, such as discovering each other's presence, determining each other's link-layer addresses, discovering neighboring routers, and maintaining reachability information to active neighbors [16]. In this section, we discuss how to efficiently support these mechanisms within a sensornet subnet.

### A. Neighbor Table

IPv6 ND uses a *neighbor cache* to maintain neighbor information such as the link-layer address mapping, reachability information, and whether the neighbor is a router or a host. As with any cache, its utility depends on the difference between cache *hits* and *misses*. With IPv6 ND, for example, always incurring a cache miss would require an address resolution exchange before sending any unicast transmission. Instead, we use a *neighbor table* where nodes can only perform unicast communication with neighbors resident in the neighbor table. Because the routing protocol determines relevant neighbors when forming routes, the insertion/eviction policies are left to the routing protocol. While limited memory bounds the number of neighbors a node can communicate with, it also makes the energy cost of ND operations predictable.

### B. Address Resolution

A node must determine the link-layer address of a neighboring node before it can send unicast IP messages to it. IPv6 ND resolves IP to link-layer address mappings by sending a link-local multicast query and processing the response from the target neighbor. But by using a *table* rather than a *cache*, the link-layer address mappings for next-hop neighbors that a node may communicate with are always resident in memory. An explicit query mechanism is not required since there is no possibility of a cache miss. Nodes obtain the link-layer address information from router advertisement (RA) messages sent by neighboring nodes Section V-D.

### C. Neighbor Unreachability Detection (NUD)

NUD allows a node to determine if it can successfully send datagrams to IP neighbors. IPv6 ND performs NUD by sending a unicast neighbor solicitation (NS) message to the target address. The target confirms reachability by sending a unicast neighbor advertisement (NA) message back to the source. Rather than requiring a full NS/NA exchange to maintain reachability between routers, we utilize link-layer acknowledgments in place of the NA message. IPv6 ND could not assume the use of link-layer acknowledgments because 1) it is intended for links (e.g., Ethernet) that do not provide such mechanisms, and 2) such mechanisms are usually not sufficient to confirm that the target received the datagram at the IPv6 layer. We are able to maintain the semantics of reachability because

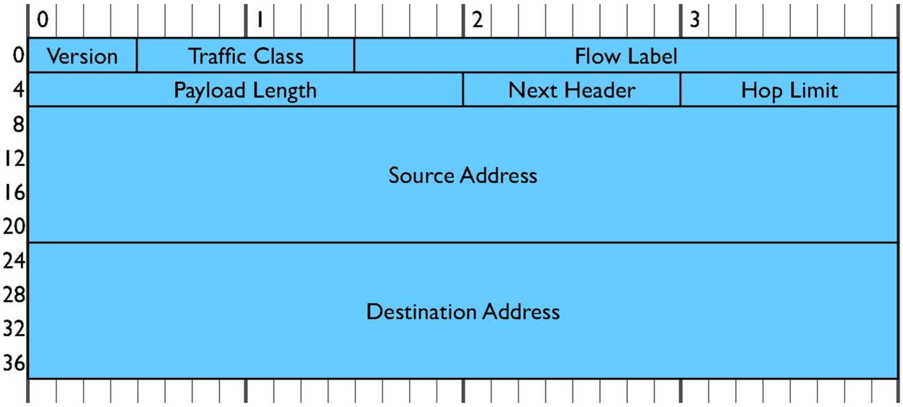


**Fig. 4.** ***IPv6 header.***

we extend our link-layer acknowledgments to indicate if the datagram was received by the network layer, as we will discuss in Section VIII.

### D. Router Discovery

A typical sensornet configuration may consist of both hosts and routers. While a routing protocol is used to establish paths among routers, IPv6 ND specifies a protocol that allows *hosts* to discover and maintain default routes. Routers periodically multicast RA messages to the link-local all-nodes address, to announce their presence. Nodes may multicast a router solicitation (RS) message to the link-local all-routers address, asking for routers to announce themselves more quickly.

We use the same RS/RA messages within the sensornet subnet. However, we utilize a Trickle timer to drive RA transmissions [8]. On each transmission, routers double the RA transmission interval up to a maximum threshold. Routers receiving an RS message reset the transmission interval to a minimum threshold. The adaptive transmission period allows the discovery mechanism to react quickly when needed but also minimize overhead for basic maintenance.

### E. Challenges With Emulating a Single Broadcast Domain

Mapping IPv6 ND mechanisms in an efficient and robust manner would have been much more difficult had we attempted to emulate a single broadcast domain where all nodes within a sensornet subnet appear as neighbors at the network layer. Take the case of selecting a default router. When emulating a single broadcast domain, selecting a default router is equivalent to selecting an edge router that may be multiple radio hops away. Without any link-layer information, a node can only select edge routers by observing network-layer communication properties.

When emulating a single broadcast domain, simply determining if an edge router is reachable with IPv6 ND is costly and challenging to make robust. Because the edge router may be multiple hops away, NUD requires a full NS/NA exchange and our link-layer acknowledgment optimization cannot be applied. Selecting an appropriate round-trip delay depends on the radio topology and can vary greatly when a sensornet subnet spans many hops. Additionally, when a link-layer path fails, the link-layer routing protocol will attempt to fix the path. However, an IPv6 ND may prematurely determine that an edge router is unreachable. In fact, a premature decision could make matters worse if a node generates additional traffic to select a new edge router or if the link-layer must discover routes to a new edge router. Similar unintended interactions of routing protocols between layers also prevented the widespread adoption of local area network (LAN) emulation over ATM [18].

## VI. ADAPTATION AND HEADER COMPRESSION

IPv6 requires the link to carry a payload of up to 1280 B [15]. Low-power radio links often do not support such a large payload—an IEEE 802.15.4 frame only supports 127 B of payload and around 80 B in the worst case (with extended addressing and full security information). Additionally, the IPv6 base header, shown in Fig. 4, is relatively large at 40 B. To handle these issues, IPv6 over low-power wireless personal area networks (6LoWPAN) introduces an adaptation layer that sits at layer 2.5 (between the link and network layers). 6LoWPAN defines a header encoding to support fragmentation when IPv6 datagrams do not fit within a single frame and compresses IPv6 headers to reduce header overhead [21].

### A. Fragmentation

The fragmentation mechanism is simple and only provides the ability to encode a datagram using multiple link frames. It does not include end-to-end recovery of lost fragments expecting that link-layer acknowledgments will provide sufficient delivery success rates. The fragmentation header is 4–5 B and contains three fields: *datagram size* indicates the size of the datagram being fragmented, *datagram tag* identifies all fragments for a particular datagram, and *datagram offset* indicates the fragment's location within the datagram. The datagram offset is always elided on the first fragment since its value is always zero.

### B. Header Compression

Traditional header compression techniques used in IP networks are flow based, observing which portions of the header change rarely across packets within a flow and eliding those portions when possible. Flow-based techniques are commonly used on point-to-point links and work well because the same compressor and decompressor are used for the lifetime of the flow. In sensornets, however, the path of a flow may change frequently due to time-varying dynamics of low-power wireless communication. When using traditional flow-based techniques, changing a next-hop route would require migrating compression state to the new route, limiting the effectiveness of header compression.

As a result, we developed a new compression format that does not require per-flow state [22]. This new format statelessly compacts headers in two ways: first, by removing redundant information across the link, network, and transport layers (IPv6 payload length is derived from lower layers, the interface ID of IPv6 addresses may be derived from link-layer addresses, and IPv6 Version is always 6 by virtue of using 6LoWPAN); and second, by assuming common values for header fields and defining compact forms of those values (traffic class and flow label are 0, hop limit is 1, 64, or 255, link-local IPv6 addresses are common, and commonly used multicast addresses only differ in a few bytes).

The header compression format also allows stateful compression for arbitrary IPv6 address prefixes and is effective for a couple reasons. First, nodes within the sensornet subnet all share the same subnet ID. Second, many applications involve all sensornet nodes communicating with a single common destination. In both cases, prefix information used for compression is common to all nodes and not specific to individual flows. To support stateful compression, each node maintains a *context table*, each entry containing an IPv6 prefix. The table may be populated by carrying context information in the same DHCPv6 messages used to assign addresses.

## VII. IPv6 ROUTING

To deliver datagrams beyond link-local scope, nodes must use an IP routing protocol to establish and maintain reachability towards IP destinations. The routing protocol is responsible for managing the forwarding table, which the forwarder uses to determine the next-hop destination when forwarding datagrams. In contrast to much existing sensornet work, we maintain the traditional separation between routing and forwarding in the IP architecture.

The routing problem for sensornet subnets differs from traditional IP networks for a few reasons. First, because today's most widely used links support a single broadcast domain, most existing routing protocols must only provide reachability between subnets. However, because our sensornet subnets may be composed of multiple IP hops, sensornet nodes also need to implement IP routing protocols to communicate within the subnet. Second, resource constraints on sensornet nodes require routing protocols to operate with limited and stale information. Memory constraints limit the amount of routing state nodes can store. Limited energy and channel capacity constrains how often nodes can communicate routing information and perform link quality estimation. Third, the wireless topology is time-varying and ill-defined. The routing protocol must infer the link topology using measurements generated from actual radio communication.

To address the resource constraints and low-power wireless characteristics, we developed a new routing protocol that incorporates various techniques developed for sensornets. At a high level, the protocol orients the link topology by constructing a directed acyclic graph (DAG). The roots of the DAG are administratively chosen and are typically nodes that forward the most traffic (e.g., edge routers). An *up route* is one that follows an edge toward the root and a *down route* is one that follows an edge away from the root. By routing along the DAG, all paths take the form of zero, one, or more up routes and zero, one, or more down routes, as shown in Fig. 5.

Utilizing a DAG allows the routing protocol to minimize resource requirements. State complexity is small

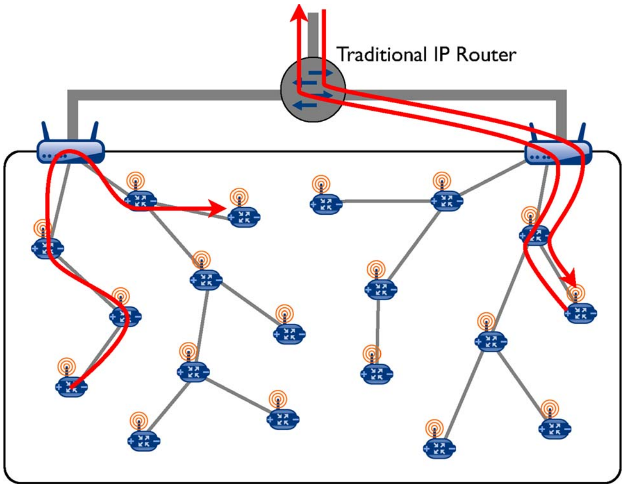


**Fig. 5.** ***Routing along a DAG.***

and constant because nodes need only maintain up routes to its DAG parents. Communication requirements remain minimal because nodes only propagate DAG topology information towards the DAG root when establishing down routes. Furthermore, unlike reactive protocols, no costly flood-based mechanism is needed to discover arbitrary destinations.

### A. Establishing Up Routes

The protocol spends most of its effort in establishing good up routes to form the DAG. Using a distance-vector protocol, nodes choose parents that minimize their path cost to one or more DAG roots. This routing protocol differs from prior work in its mechanisms that perform robust link estimation, avoid the count-to-infinity problem, and communicate routing information with low overhead.

*1) Link Estimation:* The routing protocol discovers potential up routes by receiving route advertisement messages and utilizes both the advertised route metric and link quality estimate (LQE) to determine if the link should be considered. While many radios can provide some information about the received signal quality of the route advertisement [e.g., received signal strength indication (RSSI) or chip correlation], these measurements are limited because they are easily biased by environmental factors and only test reception from that neighbor. Instead, edges within the DAG must be selected based on their bidirectional connectivity, recognizing that link-layer acknowledgments are commonplace in low-power wireless and that datagrams will flow in both directions along the DAG.

Generating LQEs consumes resources, requiring 1) communication with neighboring nodes to make observations, and 2) memory to store those observations and compute LQEs. Furthermore, confidence in the LQE depends on the number of samples. In the general case, it is infeasible for nodes to maintain LQEs for all neighbors because network-wide communication and memory costs grow polynomially with network density.

To address sensornet resource constraints, the routing protocol cooperates with the link layer by 1) managing the neighbor table to indicate links of interest, and 2) generating traffic to increase confidence in the LQEs. Because the routing protocol should only enable forwarding on links that provide a good LQE with high confidence, the protocol marks links as either *tentative* or *active*. Upon insertion, the link is marked as tentative and the routing protocol generates periodic probe messages to determine reachability and generate measurements to better compute the LQE (e.g., packet success rate, RSSI, and LQI in both directions). After achieving a certain confidence level, the entry is marked as active if the LQE is above threshold and may then be used for forwarding.

*2) Avoiding Count-to-Infinity:* A common issue with distance-vector protocols is the count-to-infinity problem caused by loops when propagating routing information. A common solution to this problem in wireless networks is to prohibit nodes from moving deeper in the DAG (i.e., increasing their path cost to the root), removing any possibility that a node may use one of its descendants as a parent. To allow cases when a node must increase its path costs, existing protocols use sequence numbers to allow the root to rebuild the routing topology and ensure loop-free propagation of routing information from the root. However, prohibiting nodes from selecting parents with larger path cost until the root updates the sequence number is overly strict—in many cases, nodes can select parents with larger path cost and avoid routing loops.

Unlike existing protocols, we take a more lenient approach in addressing the count-to-infinity problem. Our protocol also utilizes sequence numbers to allow roots to refresh the routing topology. However, routers have the ability to increase their path cost by a fixed amount. Allowing a limited increase in path cost avoids limiting the cost of count-to-infinity issues when they occur while giving greater flexibility in selecting parents.

*3) Trickle and Data-Path Validation:* Proactive routing protocols typically communicate routing information using periodic local broadcasts. Choosing an appropriate advertisement rate is difficult because it must trade convergence time with communication overhead. Our routing protocol uses two mechanisms to overcome this challenge: 1) the Trickle algorithm, and 2) data-path validation of routing information.

The routing protocol uses the Trickle algorithm to drive transmission of route advertisements [8]. While Trickle was originally intended for binary code updates, the goal is the same—to maintain consistency in information across a network with low overhead. On each transmission, routers double the transmission interval up to a maximum threshold. Routers reset the transmission threshold when they discover an inconsistency in routing information, such as receiving a new sequence number or a significant change in a parent's path cost. Furthermore, routers can suppress their own route advertisement if they receive similar advertisements from neighboring nodes.

To assist Trickle in detecting inconsistencies quickly, our routing protocol also relies on data-path validation. When the routing topology is consistent, datagrams flowing towards the root should always traverse nodes with decreasing path cost and datagrams flowing away from the root should always traverse nodes with increasing path cost. Exploiting this property, routers piggyback their path cost when forwarding data-plane datagrams and verify its value when receiving forwarded datagrams. If the value is not increasing or decreasing as expected, the router resets the Trickle timer in an attempt to resolve the inconsistency quickly.

### B. Establishing Down Routes

To establish down routes, each node periodically unicasts destination advertisement (DA) messages to the DAG root, providing information about each node's set of parents. By building-maintaining the parents of each node within the DAG, the root can then build paths from the root back to each node. In this case, the root can safely perform link reversal since the up edges were carefully selected based on their bidirectional connectivity. Because the root maintains all down routes, state complexity on all other nodes is small and constant (only that required to maintain the fixed number of up routes).

### C. Up/Down Routing

With strict DAG routing, all traffic first flows to the root, which then inserts source routing headers when forwarding datagrams back down the DAG. The downside of routing along a DAG is that the routing stretch (path length determined by the routing protocol divided by the optimal path length) can be up to twice the diameter of the network. Furthermore, the root can represent a point of congestion.

While any node can serve as the root, in practice, the root(s) are administratively chosen to be the edge router(s). Most sensornet applications involve significant communication with IP devices outside the sensornet network (i.e., data collection, management, or legacy devices) and vast majority of communication must flow through a relatively small number of edge routers. Thus, it is natural to place the root functionality at edge routers such that the routing protocol forms efficient paths for the most common flows. Because edge routers are a critical part of the infrastructure, they are likely to have additional resources for maintaining down routes.

## VIII. IPv6 FORWARDING

The forwarder is responsible for queuing incoming datagrams, determining the next hop towards the destination, and delivering that datagram to the next hop. Traditional IP forwarders do not perform any hop-by-hop recovery because they assume the link layer delivers datagrams with high success rates. Furthermore, traditional IP forwarders readily drop datagrams when queues are congested to improve queuing fairness between different flows and the overall responsiveness of the network.

The unreliable nature of low-power wireless links and limited buffering capability on sensornet nodes invalidate the traditional assumptions of IP forwarders. In sensornet subnets, a next-hop link may transition between usable and unusable relatively frequently. Constrained memory may limit forwarding queues to hold only a few messages, making full queues a relatively common occurrence, and freely dropping messages will significantly reduce overall end-to-end delivery rates and energy efficiency.

### A. Hop-by-Hop Recovery

Due to resource constraints and the unreliable nature of low-power wireless communication, the IP forwarder must implement hop-by-hop recovery. The two most common reasons for delivery failures are 1) link transmission failures, and 2) queue congestion at the receiver. Both failures can be detected using hop-by-hop acknowledgments that indicate successful reception by the intended receiver. While link transmission failures can be detected using link-layer acknowledgments, queue congestion requires network-layer acknowledgments. To reduce energy and channel utilization, we extend link acknowledgments with a flag to indicate if the upper layer successfully received the datagram. Explicitly communicating network-layer queuing failures when they occur allows the sender to distinguish between link-layer transmission failures and network-layer queuing failures.

The forwarder dequeues messages only on positive indication that it was successfully received at the network layer by the next hop. When link transmission failures occur, the forwarder performs another next-hop lookup and resubmits the datagram to the link layer. Doing so allows the forwarder to support *rerouting* and utilize forwarding table changes made by the routing protocol in response to transmission failures. When queuing failures occur, the forwarder performs *congestion control* by slowing the forwarding rate to the same next hop. Delaying retransmissions helps to reduce channel utilization and wasted energy caused by failed attempts. Because datagrams are not dropped, congested nodes will apply back pressure all the way to the application source if necessary. To handle unforeseen persistent failures, the forwarder may drop datagrams after a large number of consecutive failures.

### B. Quality of Service

Queue congestion can occur often due to the combination of limited buffering capabilities and hop-by-hop recovery mechanisms. A node with no available buffers cannot receive datagrams and, as a result, cannot help forward datagrams to their destination. Because the forwarder does not drop datagrams, the net effect of congested queues is a significant increase in communication delay. For some applications, communication delay for application traffic may or may not be an issue. Network protocols, however, often require low-latency delivery to propagate information and provide feedback in a timely manner. For example, routing protocols may need to communicate new routing information to repair a route. End-to-end transport or application protocols may need to provide timely feedback for rate control.

Allocating per-flow buffer space can help isolate congestion, but doing so is infeasible due to memory constraints. Instead, we observe that flows typically fall into three classes for many sensornet applications: 1) *link-local traffic* supports ND, routing protocols, and local

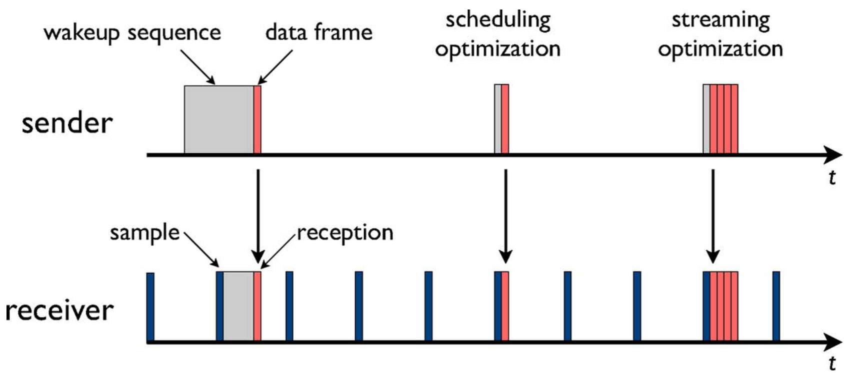


**Fig. 6.** *Sampled listening with optimizations.*

communication; 2) *upward traffic* towards edge routers for data collection; and 3) *downward traffic* away from edge routers for configuration or control traffic. Isolating these three classes ensures that the network continues to operate even when congested. Routing protocols can continue to send link-local messages to repair routes. Transport-layer messages can continue to flow upward or downward and throttle transmission rates if needed.

To support class-based isolation, the forwarder supports *queue reservations* by dedicating a minimum amount of buffer space to each traffic class and allows the forwarder to accept messages for uncongested classes even while others are congested. The queue reservations are configured statically on each node and the actual value chosen depends on the platform-specific properties. We found that even reserving just one message buffer per traffic class makes the network much more robust than without queue reservations. In addition to the reserved message buffers, a set of shared message buffers are allocated and can be used by any traffic queue. With shared message buffers, forwarders remain flexible to the actual proportion of traffic used by each flow. Note that host nodes do not reserve space for forwarding traffic because they do not forward datagrams. The forwarder services the classes in round-robin order to prevent starvation.

## IX. ALWAYS-ON (BUT MOSTLY OFF) LINK

IP protocols generally assume that the link is *always-on*. Such links have two important properties: 1) they can deliver datagrams (unicast or multicast) to neighboring nodes with relatively low latency, and 2) transitions between the link up and down states are an exception rather than the norm. The always-on property simplifies IP protocols because they do not need to worry about scheduling when to deliver datagrams. Multicast simplifies ND and message delivery to arbitrary subsets of neighbors.

While traditional IP links are always-on by constantly listening for packets, such idle listening is prohibitively costly in low-power wireless applications. However, by utilizing existing duty-cycling techniques, it is possible to give the illusion that a receiver is always on even while it is actually off more than 99% of the time. The tradeoff is decreased communication throughput and increased communication latency.

### A. Sampled Listening

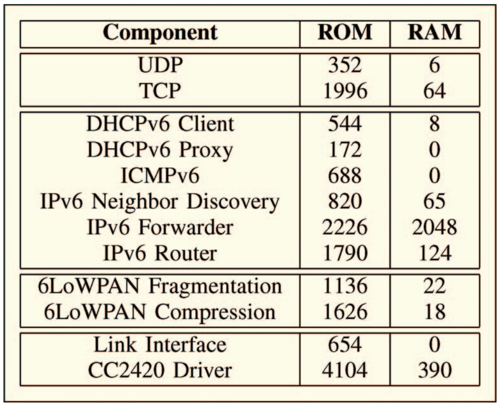
As shown in Fig. 6, sampled listening monitors the channel using short periodic receive checks (often implemented using the RSSI) to determine if a neighboring node is currently transmitting. Using short receive checks minimizes the sample period for a given duty cycle and thus communication latency as well. A node transmits frames by first transmitting a wakeup signal at least as long as the receiver's sample period. Sampled listening is both flexible and robust because it supports both unicast and multicast communication without requiring any neighbor state or global time synchronization. The tradeoff, however, is an increase in transmission cost that is proportional to the sample period.

Because existing 802.15.4 radios provide a packet interface, we implement the wakeup signal using a sequence of *wakeup frames*. A wakeup frame is an 802.15.4-compliant frame that contains a *destination address* and a *rendezvous time* to indicate the remaining time until data frame transmission. Addressing information reduces overhearing costs, as unintended destinations can abort reception early. The rendezvous time allows the destination to power down until the actual data frame transmission. This reduces the receive cost to that of receiving a single chirp frame and a data frame, making the receive cost independent of the wakeup sequence length.

### B. Scheduling

Sampled listening trades idle-listening overhead for transmission overhead. Due to uncertainty in the

Table 1 Memory Requirements for Communication Components



| Component | ROM | RAM |
|---|---|---|
| UDP | 352 | 6 |
| TCP | 1996 | 64 |
| DHCPv6 Client | 544 | 8 |
| DHCPv6 Proxy | 172 | 0 |
| ICMPv6 | 688 | 0 |
| IPv6 Neighbor Discovery | 820 | 65 |
| IPv6 Forwarder | 2226 | 2048 |
| IPv6 Router | 1790 | 124 |
| 6LoWPAN Fragmentation | 1136 | 22 |
| 6LoWPAN Compression | 1626 | 18 |
| Link Interface | 654 | 0 |
| CC2420 Driver | 4104 | 390 |

neighbor's channel sample schedule, the wakeup signal must be as long as the receiver's sample period. But by learning neighbor's sample schedules, a node can nearly eliminate the need for wakeup signals. If the destination's schedule is known, the wakeup signal is reduced to a small synchronization guard time, as shown in Fig. 6.

Knowing the *period* and *relative phase* of a neighbor's channel samples is sufficient to reduce the wakeup signal. Using extended link headers, nodes may piggyback period and phase information in both data and acknowledgment frames. The phase information indicates the amount of time since the start-of-frame was transmitted until the next channel sample event. Using the start-of-frame event provides tight time synchronization with the transmitter.

While any data traffic may carry scheduling information, discovery mechanisms provide the most benefit. For example, piggybacking schedules on RA messages allows nodes to learn the schedules of routers before probing or routing traffic through them. While the synchronization guard time grows as scheduling information ages, piggybacking schedules in acknowledgments allows nodes to (re)synchronize with neighbors and reduce guard times for subsequent transmissions.

By piggybacking schedules on data and acknowledgment frames, costly wakeup signals are only used in practice with multicast transmissions. In typical networks, the vast majority of multicast transmissions are RA messages which are relatively rare and necessary for discovering and maintaining routes.

### C. Streaming

In addition to latency, sampled listening also trades increased energy efficiency with decreased communication throughput. However, *streaming* may be used to increase both throughput and energy efficiency by allowing transmission of multiple data frames during a single sample period. By using a bit in the link header, the transmitter can indicate that additional data frames are pending and the receiver should remain on to receive them. Subsequent frames can be sent immediately without a wakeup signal, maximizing throughput and energy efficiency, as shown in Fig. 6. The streaming optimization provides the most benefit for fragmented datagrams and traffic that flows along a common route.

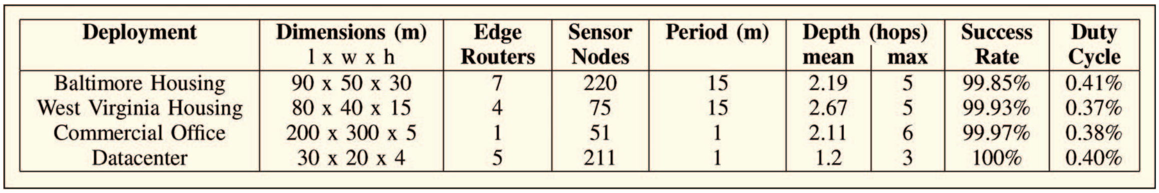
## X. EVALUATION

We implemented a production-quality IPv6 network stack for sensornets. The network stack is RFC-compliant and is the first of its kind to earn the IPv6 Ready—Phase 2 Gold designation [23]. Using this implementation, we evaluate its performance in the metrics that matter most for embedded applications—low memory footprint, high reliability, and low energy consumption. The network stack includes full implementations of IPv6 ND, forwarding, routing, ICMPv6, and all of the link and network-layer mechanisms described in this paper. We also implemented both user datagram protocol (UDP) and transmission control protocol (TCP) at the transport layer. Our implementation is built using TinyOS 2.x [24] for an Epic-based platform that consists of a 16-b TI MSP430 MCU with 48-kB ROM and 10-kB RAM and a 2.4-GHz, 250-kb/s TI CC2420 IEEE 802.15.4 radio [25].

An embedded kernel that supports one UDP socket and one TCP connection consumes 24 038 B of ROM and 3598 B of RAM. The kernel includes OS-level services required to support the IPv6 network stack. The breakdown of communication components is shown in Table 1. The network and transport layers require about 10 kB of ROM and nearly 3 kB of RAM. The vast majority of RAM consumption goes toward forwarding message buffers.

We utilized our IPv6 network stack to support commercial environmental monitoring applications in a variety of deployment environments. Characteristics of representative deployments are summarized in Table 2. In

Table 2 Production Deployments



| Deployment | Dimensions (m) | Edge | Sensor | Period (m) | Depth (hops) | | Success | Duty |
|---|---|---|---|---|---|---|---|---|
| | l x w x h | Routers | Nodes | | mean | max | Rate | Cycle |
| Baltimore Housing | 90 x 50 x 30 | 7 | 220 | 15 | 2.19 | 5 | 99.85% | 0.41% |
| West Virginia Housing | 80 x 40 x 15 | 4 | 75 | 15 | 2.67 | 5 | 99.93% | 0.37% |
| Commercial Office | 200 x 300 x 5 | 1 | 51 | 1 | 2.11 | 6 | 99.97% | 0.38% |
| Datacenter | 30 x 20 x 4 | 5 | 211 | 1 | 1.2 | 3 | 100% | 0.40% |

these applications, each node periodically reports application and network management reports to a central sever sitting outside the sensornet subnet. Each report consumes about 150 B of UDP payload, requiring the use of 6LoWPAN fragmentation when forwarding. Period in Table 2 indicates the period between reports for each node. Depth in Table 2 represents the number of hops to the nearest edge router for paths chosen by the routing protocol. In all deployments, the link layer was configured with a channel sample period of 250 ms.

Over a 12-month period, all deployments achieved a delivery success rate well over 99% and average duty cycle well below 1%. We were able to achieve these numbers because 1) we did not attempt to emulate a single broadcast domain and used a "route-over" architecture that minimizes the cost of link-local multicast and allows IP-based protocols to operate in concert with the radio topology; 2) a routing protocol that efficiently communicates routing information and avoids count-to-infinity; and 3) low-power listening techniques with scheduling optimizations. For a more thorough evaluation of our IPv6 network stack, refer to our prior publications [26].

While we have shown that an IP-based network can be implemented efficiently on sensornets, the question remains: What is the cost of an IP-based network? The adaptation layer requires less than 10 B for a typical UDP/IPv6 header. Putting this into perspective, the marginal cost of transmitting 1 B is only 1.67 uJ while the cost of transmitting a packet without a wakeup signal is 630 uJ. Even so, some header overhead is required by any network (protocol identifiers, end-to-end integrity checks, and application dispatch). Route advertisement messages are necessary for ND and routing, but similar messages are necessary in non-IP settings as well. Despite these concerns of transmission overhead, idle-listening cost at the link layer still dominates in many cases and this is independent of the use of IP above.

## XI. RELATED WORK

The trend towards connecting embedded devices to the Internet has also given rise to numerous IP stacks designed for limited memory and computation capabilities. However, early attempts achieved a small footprint through application-specific optimizations, sacrificing generality and RFC compliance [27]. Furthermore, such stacks were designed for wired networks and are only concerned with host operation. uIP demonstrated the feasibility of a nRFC-compliant IP stack for 8-b microcontrollers [28]. Seeing that it was possible to implement IP on small devices, the Internet Engineering Task Force (IETF) formed the 6LoWPAN working group to map IPv6 and support 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 [21], [29]. Our initial work significantly influenced RFC 4944.

With RFC 4944 in place, the IETF has moved to tackle other layers of the stack. The routing over low-power and lossy networks (ROLL) working group was formed to standardize an IP routing protocol tailored to the resource constraints of sensornets. Our work has significantly influenced the shape and design of this new routing protocol specification. The IETF just recently formed the Constrained RESTful Environments (CoRE) working group to standardize a application-layer protocol for sensornets. At the link layer, the IEEE 802.15.4g task group is currently working to standardize the duty-cycling techniques discussed in this paper.

Several existing implementations now support IPv6 over IEEE 802.15.4 [30]–[32]. However, no existing work has looked into the broader issues of network architecture such as a subnet model that does not support a single broadcast domain and an IPv6 addressing architecture where the communication scope does not equal the uniqueness scope. All attempted to preserve the existing IP architecture by emulating a single IP link that spans the entire sensornet. Furthermore, none of these address operation over duty-cycled links nor do they incorporate existing sensornet techniques. There are no published results that show the efficiency or effectiveness of these implementations.

## XII. CONCLUSION

We have shown that the convergence of IPv6 and low-power multihop wireless networking is possible, pragmatic, and efficient—especially in regard to the metrics that matter most for embedded applications, low memory footprint, high reliability, and low energy usage. By tackling the many challenges presented by this class of networks within the constraints of the IP architecture and its principle of layered design, rather than creating specific solutions for particular application domains and a particular link technology, we gain the potential for broad adoption, deep reduction in the total cost of ownership, interoperability with a wide range of other links, protocols, tools and applications, and longevity through advances in link technology and usage. The power of this "generic" approach is in how it permits specialization. The network designer is able to select links, switches, and management techniques to meet the performance, cost, scale, reliability, or lifetime requirements of the application and assemble potentially diverse technology in a common interoperable framework.

Utilizing the network architecture, addressing, autoconfiguration, adaptation, routing, and forwarding presented here, novel low-power wireless networking technology can join the ranks of other sophisticated technologies, such as Ethernet and WiFi, of being taken granted because it "just works" in the settings where it makes sense for it to be deployed. However, several nontechnical hurdles remain. For the traditional Internet community, the approach presented here pushes the boundaries of the Internet architecture. It observes that the broadly held, largely tacit assumption that a link is a single broadcast domain is not so essential, especially with the transition from subnets to prefixes. Routing can be optimized for common traffic patterns and be responsive to changing link-layer connectivity while still maintaining a clean, layered separation. But, it will take discipline and careful design to emerge from the broad committee process with a simple, elegant design around a few powerful mechanisms, rather than a host of knobs and flags.

For the many industrial forums that have emerged around the IEEE 802.15.4 standard, it may be difficult to accept that their specific application domains are best served not by an application specific solution over a particular link, but by a generic architecture that permits the use of a variety of links and common protocols. However, mounting quantitative evidence and rapid technical advance within such horizontally stratified layers has historically been what shaped the final outcome.

For the research community, it will be important to recognize that the presence of structure and constraints is as much an opportunity for profound innovation as their absence. Many problems that have been studied in a manner devoid of context appear anew in a particular context, with a new set of tradeoffs and criteria. It becomes possible to innovate in one area while utilizing the best available technology everywhere else, rather than what happens to be available in a local code base, and then to make clear assessments of the impact of a particular innovation.

As we look forward to 2011 and beyond, Andy Grove's famous saying would seem to apply, "Let chaos reign, then rein in the chaos." We have enjoyed a decade of tremendous innovation in the design and use of low-power wireless network technology. And we have laid the groundwork for billions of Internet worked devices. The next phase is to fully integrate these two areas of development. ■

## REFERENCES

[1] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proc. 2nd Int. Conf. Embedded Netw. Sensor Syst.*, Baltimore, MD, 2004, pp. 95–107.

[2] Dust Networks, *Technical Overview of Time Synchronized Mesh Protocol (TSMP)*, Hayward, CA, Jun. 2006. [Online]. Available: http://www.dustnetworks.com/docs/TSMP_Whitepaper.pdf

[3] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," in *Proc. 9th Annu. Int. Conf. Mobile Comput. Netw.*, San Diego, CA, 2003, pp. 134–146.

[4] J. Polastre, G. Tolle, and J. Hui, "Low power mesh networking with Telos and IEEE 802.15.4," in *Proc. 2nd Int. Conf. Embedded Netw. Sensor Syst.*, Baltimore, MD, 2004, p. 319.

[5] R. Fonseca, O. Gnawali, K. Jamieson, and P. Levis, "Four bit wireless link estimation," in *Proc. 6th Workshop Hot Topics Netw.*, 2007, DOI: 10.1.1.74.2030.

[6] B. Hull, K. Jamieson, and H. Balakrishnan, "Mitigating congestion in wireless sensor networks," in *Proc. 2nd Int. Conf. Embedded Netw. Sensor Syst.*, Baltimore, MD, 2004, pp. 134–147.

[7] A. Woo and D. E. Culler, "A transmission control scheme for media access in sensor networks," in *Proc. 7th Annual Int. Conf. Mobile Comput. Netw.*, Rome, Italy, 2001, pp. 221–235.

[8] P. Levis, N. Patel, D. Culler, and S. Shenker, "Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks," in *Proc. 1st Conf. Symp. Netw. Syst. Design Implement.*, San Francisco, CA, 2004, p. 2.

[9] A. Dunkels, F. Österlind, and Z. He, "An adaptive communication architecture for wireless sensor networks," in *Proc. 5th Int. Conf. Embedded Netw. Sensor Syst.*, Sydney, Australia, 2007, pp. 335–349.

[10] 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 *Proc. 7th Symp. Oper. Syst. Design Implement.*, Seattle, WA, 2006, pp. 249–262.

[11] J. Polastre, J. Hui, P. Levis, J. Zhao, D. Culler, S. Shenker, and I. Stoica, "A unifying link abstraction for wireless sensor networks," in *Proc. 3rd Int. Conf. Embedded Netw. Sensor Syst.*, San Diego, CA, 2005, pp. 76–89.

[12] ZigBee. [Online]. Available: http://www.zigbee.org/

[13] Z-Wave. [Online]. Available: http://www.z-wave.com/

[14] HART Communications Foundation, *WirelessHART Overview*, 2007. [Online]. Available: http://www.hartcomm2.org/hart_protocol/wireless_hart/wireless_hart_main.html

[15] S. Deering and R. Hinden, *Internet Protocol, Version 6 (IPv6) Specification*, RFC 2460 (Draft Standard), Dec. 1998. [Online]. Available: http://www.ietf.org/rfc/rfc2460.txt

[16] T. Narten, E. Nordmark, W. Simpson, and H. Soliman, *Neighbor Discovery for IP Version 6 (IPv6)*, RFC 4861 (Draft Standard), Sep. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4861.txt

[17] E. Nordmark, *Stateless IP/ICMP Translation Algorithm (SIIT)*, Internet Engineering Task Force, RFC 2765 (Proposed Standard), Feb. 2000. [Online]. Available: http://www.ietf.org/rfc/rfc2765

[18] ATMForum, *LANEmulation Over ATMVersion-2 LUNI Specification*, Dec. 1995.

[19] S. Deering, B. Haberman, T. Jinmei, E. Nordmark, and B. Zill, *IPv6 Scoped Address Architecture*, RFC 4007 (Proposed Standard), Mar. 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4007.txt

[20] S. Thomson, T. Narten, and T. Jinmei, *IPv6 Stateless Address Autoconfiguration*, RFC 4862 (Draft Standard), Sep. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4862.txt

[21] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, *Transmission of IPv6 Packets Over IEEE 802.15.4 Networks*, RFC 4944 (Proposed Standard), Sep. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4944.txt

[22] J. Hui and P. Thubert, *Compression format for IPv6 datagrams in 6LoWPAN networks*, Oct. 2009. [Online]. Available: http://tools.ietf.org/html/draft-ietf-6lowpan-hc

[23] IPv6 Forum, *IPv6 Ready Logo*. [Online]. Available: http://www.ipv6ready.org/

[24] University of California at Berkeley, *TinyOS*, Berkeley, CA, 2004. [Online]. Available: http://www.tinyos.net/

[25] P. Dutta, J. Taneja, J. Jeong, X. Jiang, and D. Culler, "A building block approach to sensornet systems," in *Proc. 6th ACM Conf. Embedded Netw. Sensor Syst.*, Raleigh, NC, 2008, pp. 267–280.

[26] J. W. Hui and D. E. Culler, "IP is dead, long live IP for wireless sensor networks," in *Proc. 6th ACM Conf. Embedded Netw. Sensor Syst.*, Raleigh, NC, 2008, pp. 15–28.

[27] J. Bentham, *TCP/IP Lean: Web Servers for Embedded Systems*. Manhasset, NY: CMP Media, Inc., 2000.

[28] A. Dunkels, "Full TCP/IP for 8-bit architectures," in *Proc. 1st Int. Conf. Mobile Syst. Appl. Services*, San Francisco, CA, 2003, pp. 85–98.

[29] J. W. Hui and D. E. Culler, "Extending IP to low-power, wireless personal area networks," *IEEE Internet Comput.*, vol. 12, no. 4, pp. 37–45, Jul.–Aug. 2008.

[30] Sensinode, *NanoStack*. [Online]. Available: http://sourceforge.net/projects/nanostack/

[31] M. Durvy, J. Abeillé, P. Wetterwald, C. O'Flynn, B. Leverett, E. Gnoske, M. Vidales, G. Mulligan, N. Tsiftes, N. Finne, and A. Dunkels, "Making sensor networks IPv6 ready," in *Proc. 6th ACM Conf. Embedded Netw. Sensor Syst.*, Raleigh, NC, 2008, pp. 421–422.

[32] H. Huo, H. Zhang, Y. Niu, S. Gao, Z. Li, and S. Zhang, "MSRLab6: An IPv6 wireless sensor networks testbed," in *Proc. 8th Int. Conf. Signal Process.*, 2006, vol. 4, pp. 16–20.

## ABOUT THE AUTHORS

**Jonathan W. Hui** received the B.S. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, in 2003 and the M.S. and Ph.D. degrees in computer science from the University of California at Berkeley, Berkeley, in 2005 and 2008, respectively. For his Ph.D. dissertation, he did seminal work on IPv6 in constrained networks, having designed, implemented, and evaluated the world's first IPv6/6LoWPAN network for low-power wireless radios.

Currently, he is a Lead Engineer and Founding Engineer at Arch Rock Corporation, San Francisco, CA. His primary focus is to drive the use of IP in smart objects. At Arch Rock Corporation, he led the development of the PhyNet low-power wireless platform, the world's first commercial IPv6/6LoWPAN product and the first of its kind to receive the IPv6 Ready-Phase 2 (Gold) designation. He participates in the Internet Engineering Task Force and has authored standards specifications related to IPv6 in low-power wireless networks. He also participates in the IP for Smart Objects Alliance to promote the use of IP for smart objects.

**David E. Culler** (Fellow, IEEE) received the B.A. degree in mathematics from the University of California at Berkeley, Berkeley, in 1980 and the M.S. and Ph.D. degrees in computer science from Massachusetts Institute of Technology (MIT), Cambridge, in 1985 and 1989, respectively.

Currently, he is a Professor of Computer Science at the University of California at Berkeley, CTO of Arch Rock Corporation, San Francisco, CA, and an Associate Chair of Electrical Engineering and Computer Sciences. He has been on the faculty at the University of California at Berkeley since 1989, where he holds the Howard Friesen Chair. He was the Principal Investigator of the DARPA Network Embedded Systems Technology project that created the open platform for wireless sensor networks based on TinyOS, and was the founding Director of Intel Research, Berkeley. He has done seminal work on networks of small, embedded wireless devices, planetary-scale internet services, parallel computer architecture, parallel programming languages, and high performance communication, including TinyOS, PlanetLab, networks of workstations (NOW), and active messages. He has served on Technical Advisory Boards for several companies, including Inktomi, ExpertCity (now CITRIX online), and DoCoMo USA.

Prof. Culler is a member of the National Academy of Engineering and a Fellow of the Association for Computing Machinery (ACM), and was selected for ACM's Sigmod Outstanding Achievement Award, *Scientific American*'s "Top 50 Researchers," and *Technology Review*'s "10 Technologies that Will Change the World." He received the National Science Foundation (NSF) Presidential Young Investigators award in 1990 and the NSF Presidential Faculty Fellowship in 1992.