Beyond Interoperability – Pushing the Performance of Sensor Network IP Stacks

JeongGil Ko[1], Joakim Eriksson[2], Nicolas Tsiftes[2], Stephen Dawson-Haggerty[3],
jgko@cs.jhu.edu, joakime@sics.se, nvt@sics.se, stevedh@eecs.berkeley.edu,
Jean-Philippe Vasseur[4], Mathilde Durvy[4], Andreas Terzis[1], Adam Dunkels[2] and David Culler[3]
jpv@cisco.com, mdurvy@cisco.com, terzis@cs.jhu.edu, adam@sics.se, culler@eecs.berkeley.edu

[1] Department of Computer Science, Johns Hopkins University
[2] Swedish Institute of Computer Science (SICS)
[3] Computer Science Division, University of California, Berkeley
[4] Cisco Systems

Abstract

Interoperability is essential for the commercial adoption of wireless sensor networks. However, existing sensor network architectures have been developed in isolation and thus interoperability has not been a concern. Recently, IP has been proposed as a solution to the interoperability problem of low-power and lossy networks (LLNs), considering its open and standards-based architecture at the network, transport, and application layers. We present two complete and interoperable implementations of the IPv6 protocol stack for LLNs, one for Contiki and one for TinyOS, and show that the cost of interoperability is low: their performance and overhead is on par with state-of-the-art protocol stacks custom built for the two platforms. At the same time, extensive testbed results show that the ensemble performance of a mixed network with nodes running the two interoperable stacks depends heavily on implementation decisions and parameters set at multiple protocol layers. In turn, these results argue that the current industry practice of interoperability testing does not cover the crucial topic of the performance and motivate the need for generic techniques that quantify the performance of such networks and configure their runtime behavior.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols; C.2.6 [**Computer-Communication Networks**]: Internetworking—*Standards*

General Terms

Experimentation, Performance, Standardization

Keywords

IPv6, 6LoWPAN, RPL, IETF, Interoperability, Sensor Network, TinyOS, Contiki

1 Introduction

For wireless sensor networks to be widely adopted by the industry, hardware and software implementations from different vendors need to interoperate and perform well together. While IEEE 802.15.4 has emerged as a common physical layer that is used both in commercial sensor networks and in academic research, interoperability at the physical layer is not enough: multiple layers of the stack must interoperate. Likewise, interoperability without high performance (e.g., high packet reception ratio) is inadequate given the critical nature of some of the industrial applications envisioned for wireless sensor networks (e.g., process control) and the severe resource limitations of mote platforms.

Academia has paid little attention to interoperability, since it has focused on producing systems and network stacks that help in attaining research results (e.g., [1, 6]). Moreover, research deployments have been homogeneous in both hardware and software, providing little incentive for work towards interoperability. This course of research has filled an important need: it has allowed the community to focus on the fundamental problems, leaving standardization and interoperability to the time when the fundamental issues were addressed.

The Internet Protocol (IP), which has proven its interoperability and extensibility as the protocol underlying the global Internet over the last 30 years, is seen by many as a promising solution to the interoperability problem in low-power and lossy networks [5, 12, 19, 21]. In support of this effort, the Internet Engineering Task Force (IETF) recently specified a number of protocols and adaptation layers that allow IPv6 to run over IEEE 802.15.4 link layers. In particular, the 6LoWPAN working group specified header compression and fragmentation for IPv6 over IEEE 802.15.4 [16] and the IETF

RoLL working group designed the RPL protocol as a proposed standard for IPv6 routing in low-power and lossy networks (LLNs) [21, 22]. These specifications provide a firm foundation for interoperable systems.

We present two independent implementations of the IPv6 stack for LLNs, one for the Contiki operating system and another for TinyOS. Both efforts implement the necessary parts of the IPv6 protocol, IPv6 header compression and fragmentation with the 6LoWPAN adaptation layer, routing over LLNs with the RPL protocol, as well as a set of protocols from the TCP/IP protocol suite [3, 5, 8]. We demonstrate that the two implementations interoperate, but more importantly we show that protocol interoperability is not enough: implementation decisions and protocol settings result in performance degradations that traditional interoperability testing methods do not detect.

The industry's standard practice for interoperability testing comprises two main methods. First, so-called *bake-offs*, or interoperability events, during which systems from different vendors are tested against each other. Second, conformance testing, where systems are tested against a reference implementation. For example, the IP for Smart Objects (IPSO) Alliance [7], formed to promote the use of IP for WSNs, has to this date held three interoperability testing events. These events have relied on vendors to either be physically present so their devices can communicate directly, or be virtually present during the same time, to allow network-level communication between devices. Furthermore, the IPv6 Ready forum provides a reference implementation of the IPv6 stack that sensor network IP software have used for conformance testing [8, 9].

We argue that such interoperability events are not enough since they cannot, nor are they supposed to, detect performance degradations such as the ones we discovered in our work. Detecting performance reductions requires large-scale experiments with controlled parameters and visibility at the network layer. For these reasons, network simulation should be an additional tool in the interoperability tester's toolbox since it allows interoperability testing at network scale with full control over all network parameters and full visibility into the network.

The contributions of this paper are two. First, we demonstrate IPv6 interoperability between the leading sensor network operating systems, TinyOS and Contiki. Interoperability is highly relevant to the industry but has not received much attention in academic research. Second, we show that interoperability between independent implementations of LLN stacks is not enough: implementation choices and protocol settings can affect system performance in unexpected ways. Our results show that although two RPL implementations interoperate, one must look deeper into the network performance as subtle variations in underlying components may affect the network's packet delivery performance.

The rest of the paper is structured as follows. We discuss the need for interoperability in wireless sensor networks and present the IPv6 architecture for sensor networks in Section 2. In Section 3 we present our IPv6 implementations for Contiki and TinyOS and in Section 4 we discuss how we leverage the Contiki simulation environment to perform interoperability testing. We run Contiki and TinyOS in a set of testbed experiments and simulation setups in Section 5, demonstrating both that our systems interoperate and that system performance can be affected by inconsistent decisions across implementations. We discuss our findings and their implications for future work in Section 6. We review related work in Section 7 and conclude the paper in Section 8.

2 The Case for Interoperability in Wireless Sensor Networks

In the early sensor network vision, networks were homogeneous and interoperability was not a concern. Likewise, early industry efforts used proprietary technologies with no provisions for working with other systems. But to reach widespread commercial adoption of sensor networks, interoperability is essential. Without interoperability, customers are locked in with one vendor.

The Internet Protocol (IP) has recently been suggested as a way to attain interoperability in low-power sensor networks [5, 8, 12, 19, 21]. By running IP, sensor networks leverage knowledge, systems, software, and equipment from the global Internet. Experience has shown that IP is both lightweight enough to run on even severely resource constrained systems [5, 8] and that the power consumption and performance is on par with that of heavily optimized sensor network protocols [12, 19]. Furthermore, IPv6 allows emerging commercial WSN applications to scale. The address space of IPv6 is large enough to accommodate billions of systems and the built-in auto-configuration mechanisms simplify network deployment and management.

Open standards are of primary importance for widespread commercial adoption of sensor network technology. With proprietary specifications and non-standards, customers are at risk of vendor lock-in, where they become completely dependent on one particular vendor. This is a significant business risk: should the vendor choose to discontinue its product line, the customer must make an expensive switch to a different technology. With open standards, customers may choose among multiple vendors.

At the same time, open standards are useful only if multiple implementations exist. This is one of the tenets of the standardization work within the IETF, which requires specifications to have multiple independent implementations before the specification is moved towards publication in the standard track. In turn, the existence of multiple implementations means that interoperability is essential to making the standard widely adopted. In fact, interoperability is important for both vendors, who can use it as a market advantage, and customers, who can feel that the purchased equipment will work with other vendors' equipment.

The need for interoperability has been highlighted by several recent industrial efforts. The IPSO Alliance [7] was formed to promote the use of IP for connecting smart objects and has established an interoperability program at both the IPv6 and the lower layers of the IPSO stack. The ZigBee/IP effort aims to formulate a subset of the IPSO standards for specific applications.

2.1 Emerging Applications

The evolution of wireless sensor networks is largely driven by a set of emerging applications [21]. The smart grid is intended to improve the electrical power grid and save considerable amounts of energy for society as a whole. Building and home automation improves the quality of indoor environments in terms of temperature, air quality, and lighting, while saving energy in the process. Industrial automation improves the quality of industrial processes. Smart cities allow new services inside increasingly populated cities, such as automatic parking management and pollution monitoring. Wireless sensors are an integral part of all these applications.

The smart grid covers a wide range of topics, ranging from efficient high-voltage current transmission to lightweight power measurement of individual devices in homes and offices. Communication is a central part of the smart grid. Power meters transmit their readings to the utility companies. Small-scale producers, such as homes with solar cells, communicate with utilities to negotiate both power levels and pricing. Individual devices, such as washing machines, ask for the current price rates and adjust their operation to reduce cost. Due to the need for interoperability, the need for open standards, and the large-scale nature of the smart grid, many standardization organizations, such as NIST in the U.S., have proposed the using IPv6 as the basis for all smart grid communication.

In building and home automation, communication is used for both sensing and control. Temperature, humidity, and presence sensors provide input to control algorithms with the goal of improving the conditions inside buildings while reducing their power consumption. The control algorithms send commands to heaters, lighting controllers, shutters, air coolers, and other actuators. Given the large number of different sensors and actuators even in small deployments, interoperability between different vendors is critical.

In industrial automation, predictability and real-time reliability are important aspects, but also interoperability and the ability to communicate with existing applications. In industrial automation, sensors are used for both process monitoring and for assisting a mobile workforce to make informed decisions on the factory floor. Workers with hand-held computers will need to communicate with sensors and actuators. Even though many industrial automation installations may consist of equipment from a single vendor, they will need to be augmented with new equipment over time, highlighting the need for interoperability.

For smart cities, communicating sensors will be used for tracking vehicles and parking spaces, measuring and tracking pollution, and for controlling traffic and street lights. These large scale systems will comprise different types of devices and networks. 3G and 4G networks will be used for long-range links. WiFi and WiMax for shorter range links, where throughput is important and power consumption is not an issue. Low-power wireless networks, such as IEEE 802.15.4, will be used for short-range multi-hop networks. With this large number of systems and vendors, interoperability is essential for the success of these efforts.

2.2 IPv6 for Lossy, Low-Power Networks

TCP/IP is arguably a highly successful network architecture that has proven its stability, scalability, and ability to support a plethora of applications. For wireless sensor networks, the IP protocol stack can provide interoperability between devices within the WSN and between the WSN and existing IP-based systems.

2.2.1 Link Layers

Emerging wireless sensing applications use different link layers. Some installations use existing network technologies such as Ethernet or WiFi but many use emerging low-power wireless technologies such as IEEE 802.15.4 for installation efficiency. Finally, others use existing wired infrastructure such as power-line communication (PLC).

Low-power wireless communication technologies, such as IEEE 802.15.4 and PLC, can be classified as low-power and lossy networks (LLNs). Low-power operation is important for both technologies: (i) wireless systems are often powered by batteries whereby the system's energy consumption determines its lifetime, (ii) power consumption determines the physical size of the power transformers used in PLC systems. Moreover, both networks have similar properties in terms of lossiness, whereas loss rates vary by orders of magnitude over both short and long time scales. In turn, the differences between LLNs and the mostly stable links that IP expects have prompted the design of new link-layer adaptation mechanisms and routing protocols for IP-based LLNs.

2.2.2 Header Compression with 6LoWPAN

IPv6 was designed for high-bandwidth networks where performance in terms of throughput and packet forwarding speed is crucial. For this reason, the IPv6 headers were designed for efficient parsing in hardware, with fixed size header fields aligned at word boundaries. On the other hand, LLN links typically have low bandwidth and small maximum frame sizes and applications generate little data. In this context, the IPv6 header size can be problematic.

Header compression is a well-known technique to reduce the header size of IP packets. The 6LoWPAN IETF working group defined a header compression scheme for IPv6 over IEEE 802.15.4 [16], commonly called 6LoWPAN. IEEE 802.15.4 has a maximum frame size of 127 bytes and since IPv6 requires the link layer to support a minimum packet size of 1280 bytes, 6LoWPAN also provides a link-layer fragmentation and reassembly mechanism. While originally defined for IEEE 802.15.4, 6LoWPAN has been subsequently used for other link layers such as PLC.

2.2.3 Routing with RPL

The IPv6 Routing Protocol for Low-power and Lossy Networks (RPL) is the routing protocol for IPv6-based LLNs proposed by the IETF's RoLL working group [22]. RPL is designed for networks with significantly higher packet loss rates than those assumed by existing routing protocols. RPL is primarily designed for many-to-one (collection) traffic patterns, a common traffic pattern for LLN applications, but works well for point-to-multipoint and point-to-point traffic as well. Being optimized for low-speed links, RPL focuses on maintaining low control plane overhead. Finally, RPL supports multi-topology routing: a set of virtual routing topologies that can be built that are optimized for different metrics and sets of constraints.

At the core of RPL routing is the *Destination-Oriented Directed Acyclic Graph* (DODAG). This DODAG, rooted at an edge router, is used by all nodes in a RPL network to maintain a default route to the DODAG's root. In order to construct and maintain the DODAG, RPL specifies ICMPv6 messages called DODAG Information Objects (DIOs). DIO messages carry reachability information and DODAG Information Solicitations (DISes) which request DIO transmissions from neighboring nodes. DIOs also carry the *DODAG rank* of the nodes that generated the messages. This DODAG rank value represents the relative distance of the DIO's originator from the root and is essential to the DODAG given that nodes use it to determine their preferred parents.

RPL attempts to support different application requirements through multiple *objective functions* (OFs), used for parent selection. These OFs are carried in DIOs and describe how a node should compute its DODAG rank value given a set of metrics. An objective function can be tailored so that RPL computes routing paths that achieve application-specific goals, such as providing latency-sensitive routes for real-time applications. As a basis for interoperability, RPL includes default Objective Function 0 (OF0), that selects routes based on the hop-count to the DODAG root [20].

3 IPv6 in Contiki and TinyOS

While protocol design can simplify interoperability, only when two independent implementation work together one can claim that interoperability has been demonstrated. We present two independent implementations of the IPv6 LLN stack: one for Contiki and another for TinyOS. While the two implementations were developed independently they share many design principles. Both implementations implement the IPv6 protocol, including ICMPv6, with support for the UDP and TCP transport protocols. Underneath the IPv6 layer, both stacks implement the 6LoWPAN layer. Finally, each stack provides its own implementation for the RPL routing protocol (version 18 at the time of writing [22]).

Figure 1 shows the software architecture of our two implementations. Empty arrows show the path of packets through the stack, while filled arrows correspond to function calls. The structure of the two stacks is similar: the IPv6 layers call the RPL routing modules when receiving ICMPv6 messages and discovering neighbors. RPL modules call the IPv6 stacks to install routes to the IPv6 routing tables. At the same time, there are also subtle differences between the two stacks. Link layer feedback is handled in the Link Estimator module in Contiki, whereas link layer feedback is handled as part of the TinyOS BLIP module. However, both RPL modules respond similarly to link layer feedback, recalculating routes accordingly. TinyOS validates RPL data headers by exchanging information between BLIP and TinyOS, whereas this is performed fully within the ContikiRPL module. In both cases, the observable result is indistinguishable: the differences in structure are due to implementation choices.

The IPv6 stack in Contiki is has been certified under the IPv6 Ready Silver logo program [8] and its TCP implementation fulfills all the RFC requirements [5]. ContikiRPL has successfully completed interoperability testing through the IPSO Alliance's interop program, where it was used on three different platforms and ran over two different link layers, IEEE 802.15.4 and the Watteco low-power power-line communication module. Also, at least two sensor network deployments have used ContikiRPL. The TinyOS IPv6 stack supports IPv6, ICMPv6, DHCPv6, UDP, and TCP [3] and has been used in several deployments [4, 14].

Both stacks provide means to route packets from the sensor network to other networks. Contiki and TinyOS include implementations of the Point-to-Point Protocol (PPP) for tunneling packets over serial links. Contiki additionally supports Serial Line IP (SLIP) and Ethernet adaptation.

3.1 The IPv6 / RPL Interface

TinyRPL and ContikiRPL interact directly with their respective IPv6 implementations, BLIP and uIPv6. In both cases, the IPv6 stack forwards packets while the RPL layer installs entries in the forwarding tables. TinyOS uses RPL option header support [13] and uses BLIP's exposed packet forwarding plane to query TinyRPL for protocol-specific forwarding decisions.

3.2 RPL Objective Function Support

Objective functions in RPL determine how parent selection and forwarding decisions are made. Multiple objective functions exist, each aiming to optimize different metrics [21, 22]. ContikiRPL and TinyRPL support both the OF0 and the MRHOF objective functions and are capable of supporting any of the objective functions documented as Internet Drafts by the IETF RoLL working group through modular design.

3.3 RPL Link Quality Estimation

TinyRPL and ContikiRPL use per-link ETX to estimate link quality. Both implementations update ETX value E_p after sending packet p using an EWMA filter

$$E_p = \alpha T_p + (1-\alpha)E_{p-1},$$

where T_p is the number of transmission attempts for packet p, provided by the underlying MAC layer, and $\alpha \leq 1$ is an implementation-defined constant. ContikiRPL uses $\alpha = 0.2$ and initializes the ETX of new neighbors to 3.5, while TinyRPL uses $\alpha = 0.5$ and an initial ETX of 5.0, adapting faster to changes in the number of packet retransmissions.

In the case of OF0, the ETX estimates are used as tie-breakers among multiple potential parents with the same hop-count, while MRHOF uses the ETX of the end-to-end path to select minimum cost routes.

In ContikiRPL, an external neighbor information module updates link cost estimates through a callback function. Within one second of such an update, ContikiRPL recomputes the path cost to the sink via the updated link and checks with the selected objective function whether to switch the preferred parent. Whenever the ETX of a node in the parent set changes, TinyRPL immediately selects the node with the lowest metric as the desired parent in the DODAG.

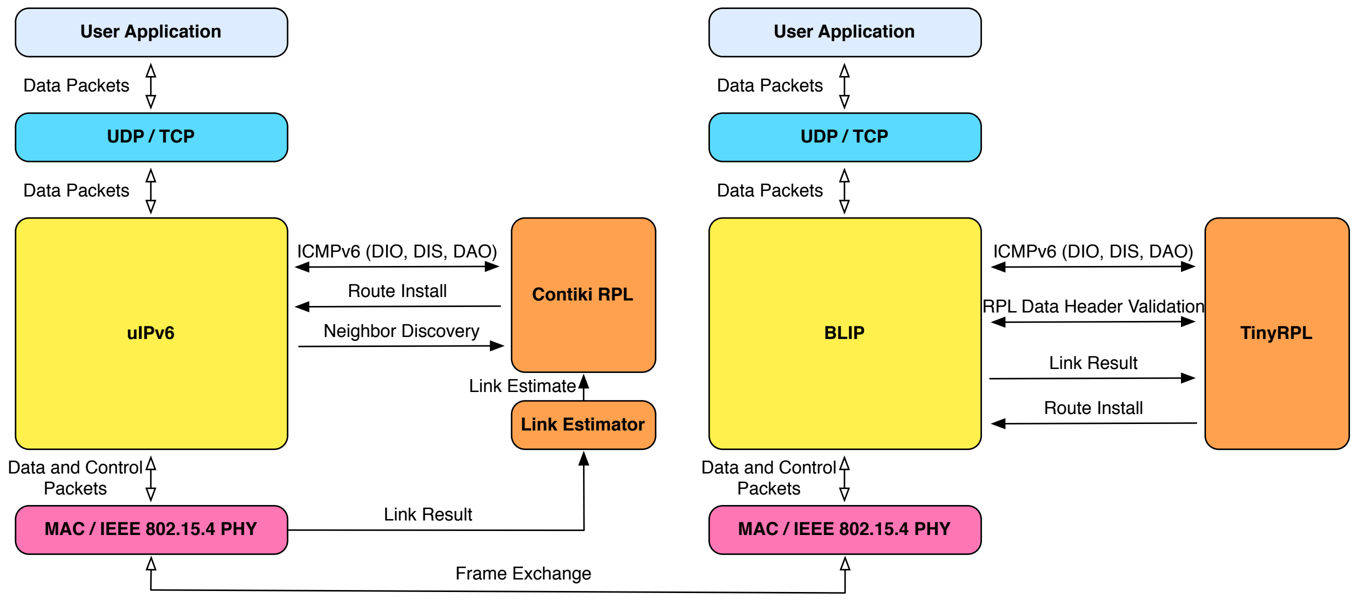

Figure 1. The IPv6 LLN implementations in Contiki and TinyOS 2.x. uIPv6 in Contiki and BLIP in TinyOS provide support for IPv6. ContikiRPL and TinyRPL implement the RPL routing protocol and communicate with their respective packet forwarding engines.

4 An Interoperability Simulation Environment

We leverage both a WSN testbed and a controllable simulation environment as tools to test the interoperability of the two implementations presented above. On one hand, the testbed provides the ability to validate performance in real wireless channel environments. However, relying solely on testbed results complicates the task of examining the network's behavior and diagnosing the root causes of performance degradations. For this purpose, while simulators lack the realism of testbeds, we leverage their ability to provide repeatability and full control over the parameter space. Furthermore, unlike a testbed setup, which is by necessity confined both in time and in space, a simulation can be made arbitrary complex. Moreover, interoperability testing can be accelerated by running multiple simulations in parallel.

We selected the Contiki simulation environment in our work (`http://www.sics.se/contiki/`). The Contiki simulation environment consists of two parts: the Cooja network simulator, which allows large-scale network simulation with a range of radio models, and the MSPsim mote emulator, which combines cycle-accurate emulation of MSP430-based motes such as the Tmote Sky with bit-accurate emulation of the CC2420 radio transceiver. With MSPsim, the exact same binary code that runs on the mote can be executed in simulation. Likewise, Cooja can simulated a network of MSPsim nodes with exact timing of all message exchanges.

The Contiki simulation environment allows combining different software stacks into the same simulation. This is a necessity for testing the interoperability of different protocol implementations, as interoperability relies on the simultaneous execution of different software implementations. The Contiki simulation environment also supports different hardware as part of the same simulation, allowing interoperability testing across hardware platforms. In addition to the MSPsim emulator, the simulation environment also contains the AvroraZ emulator, which emulates the AVR-based MicaZ mote.

Combining timing-accurate network simulation with cycle-accurate binary emulation allows testing both the interoperability of different implementations and the performance of the resulting system as a whole. By contrast, a testbed can prove the interoperability of two systems, but complicates the diagnosis of performance issues. Finally, binary emulation allows interoperability testing between systems that are shipped only in binary form by the software vendor. Although binary-only software releases are uncommon in academia, this frequently occurs in the commercial sector as it allows vendors to maintain trade secrets and source code.

5 Evaluation

The purpose of this evaluation is to explore the performance implications of interoperability. Specifically, we are interested in any potential performance shortcomings of interoperable systems compared to state-of-the-art, yet custom protocols as well as performance degradations that emerge when multiple implementations have to interoperate.

We perform our experiments using two different tools. First, using a medium-sized indoor testbed, we measure the performance of different IPv6 implementations, when running separately and together, under realistic wireless channel conditions. Second, we use the Contiki simulation environment as a way to control all of the experiments' parameters.

5.1 Initial Testbed Experiments

We first evaluate the two systems by measuring the packet reception ratio (PRR) in an indoor testbed consisting of 51

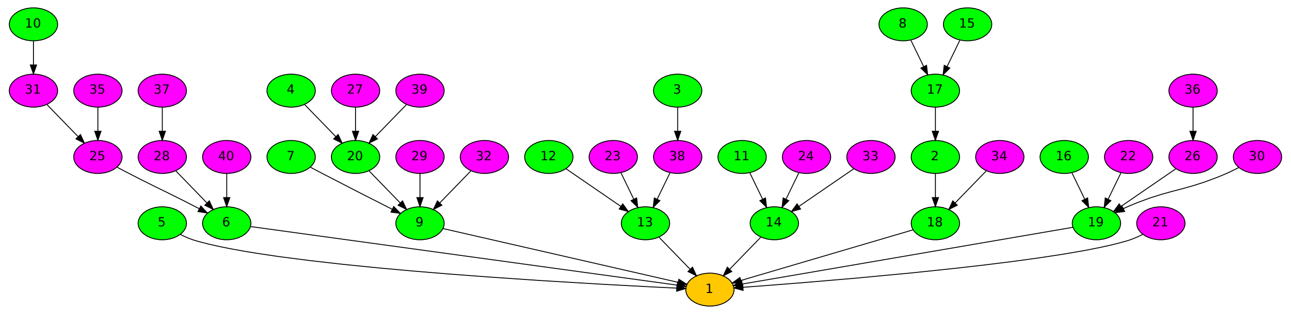

Figure 2. The simulation topology consists of 39 sender nodes and one root. The figure shows a sample RPL topology graph with both Contiki (lighter shade) and TinyOS nodes (darker shade).

Routing Protocol	Average PRR (Std. Dev.)
TinyOS IPv6	97.84% (1.06)
Contiki IPv6	95.96% (0.92)
TinyOS + Contiki	87.01% (1.23)

Table 1. Packet reception ratio (PRR) of the two systems operating individually and together in the same network. The mixed network has 26 TinyOS nodes (including the network's root) and 25 Contiki nodes. When the two protocols are combined with their default parameter configurations PRR decreases by ∼9-10%.

TelosB motes distributed over two floors of an office building. The wireless channel fluctuations mostly happen during daytime, driven by people moving throughout the building. We evaluate the performance of the IPv6 implementations in TinyOS and Contiki separately and present results for a heterogeneous network consisting of 26 TinyOS nodes, one of which is the root, and 25 Contiki nodes. All non-root nodes transmit one packet every eight seconds towards the root. We tested each network configuration three times, over three consecutive days. In all cases, we used the OF0 RPL objective function.

Table 1 presents the PRR from these initial testbed experiments. The results point to three important findings. First, Contiki and TinyOS, when independent, achieve high PRR (>95%), which is what one would expect from a good protocol implementation. Second, the results of the TinyOS + Contiki configuration suggest that interoperability is possible. Third, even though the two implementations do interoperate, the PRR of the heterogeneous network is∼9-10% lower than either of the two homogeneous networks.

This result is somewhat surprising given that the root node controls the timer values that nodes use to generate their periodic control traffic. It does so by including these values in the DIO messages it transmits, indirectly affecting the parent selection mechanism as well as the link estimation and selection processes. We note that using a Contiki node as root did not change the results appreciably.

5.2 A Deeper Understanding with Simulation

To understand the causes underlying the results from the previous section we move to the Contiki simulation environment. While Cooja provides multiple radio models, we use a unit disk graph model with Bernoulli losses in which the loss probability is proportional to the square of the node distance. The loss probability at the edge of the transmission range is configurable and we use three different configurations: one with no path loss; another with 50% loss at the edge of the transmission range, resulting in an average link PRR of 78%; and a third with 100% loss at the edge of the transmission range, resulting in an average link PRR of 56% among all the connected links in the network. We realize that this loss model does not accurately capture the channel loss behaviors seen in real-world environments. On the other hand, we do not attempt to emulate reality with this model. Instead, we use it to create the network dynamics (i.e., packet transmissions, parent changes) necessary to examine the two implementations' behaviors.

Figure 2 shows a sample RPL DODAG for the 40 TelosB mixed-implementation network simulated in Cooja. We change the nodes' positions between simulation runs, while keeping the traffic pattern identical to the one used in Section 5.1.

We first measure the PRR of mixed TinyOS and Contiki networks. To do so, we vary the distribution of Contiki and TinyOS nodes in 10-node increments. We test each node configuration with a Contiki and a TinyOS root and run each experiment ten times with different random seeds. We measure the end-to-end PRR and present the results in Figure 3.

Similar to the testbed results in Table 1, Figure 3 exhibits a considerable decrease in PRR for mixed networks. This agreement suggests that the performance degradation is not an artifact of the wireless channel's vagaries, but rather due to an underlying interoperability issue.

We note that the results in Table 1 and Figure 3 were gathered with the two implementations using their default MAC-layer parameters and network queue sizes. Specifically, TinyOS nodes had a retransmission interval of 50 milliseconds and a queue size of 10 packets, while Contiki nodes had a queue size of four packets and the interval between retransmissions was 128 milliseconds. One would then ex-

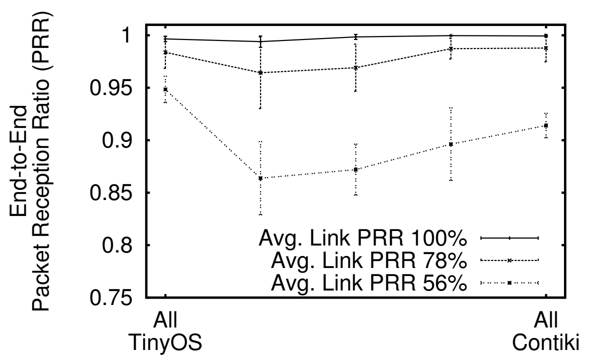

Figure 3. PRR using the original MAC-layer parameters (Contiki: retransmission interval = 128 ms, queue size = 4; TinyOS: retransmission interval = 50 ms, queue size = 10). The performance of the homogeneous networks remains high, while the performance of the mixed networks is significantly lower. This effect is pronounced when link losses are high.

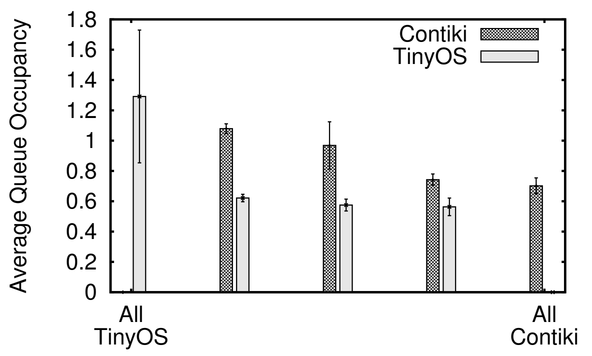

Figure 4. Average queue occupancy of Contiki and TinyOS nodes in the lossiest link environment (avg. link PRR=56%). Contiki and TinyOS nodes use their default queue size and MAC-layer parameters.

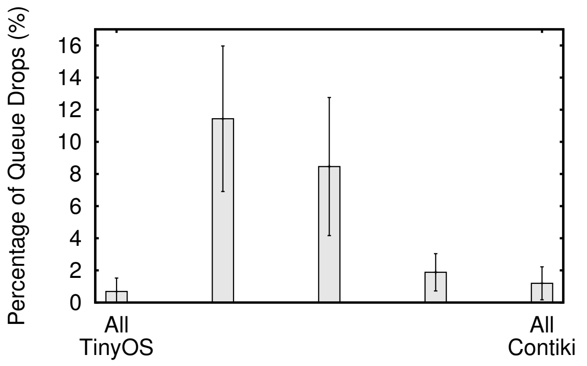

Figure 5. Percentage of packets lost due to queue drops when running TinyOS and Contiki with their default configurations (avg. link PRR=56%). The majority of the losses seen in Figure 3 are due to queue drops. In all cases, except for the “All TinyOS” case, 100% of the queue drops happen at Contiki nodes.

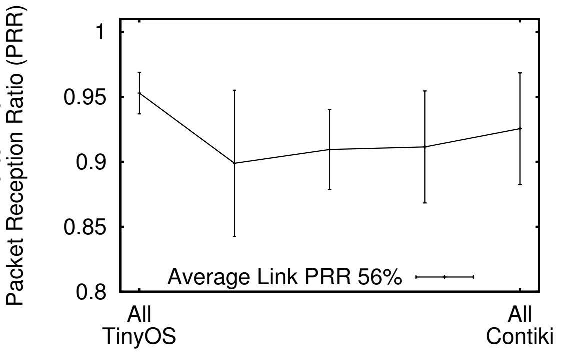

Figure 6. PRR results for lossy channel conditions (avg. link PRR=56%), when both MAC layers use the same retransmission interval (50 ms), same number of retransmissions (10), but different queue sizes (Contiki = 4, TinyOS = 10). The difference in queue sizes affects performance significantly.

pect Contiki to perform worse, given its smaller queue size. Surprisingly, a homogeneous network of Contiki nodes performs better than networks with mixed Contiki and TinyOS nodes. This effect is even more prominent as the average link quality decreases.

Figure 4 plots the average queue occupancy for Contiki and TinyOS nodes in the lossiest simulation setup (i.e., average link PRR =56%). It is clear that Contiki has higher queue occupancy than TinyOS when the two systems coexist. One can then imply that Contiki nodes take longer to deplete their queues due to their longer MAC-layer retransmission intervals. Thus Contiki nodes are more likely to overflow their queues when intermixed with TinyOS nodes.

Figure 5 plots the percentage of packets that are lost due to queue drops. These results suggest that the majority of the losses in Figure 3 are due to queue drops. Moreover, we note that in all, other than the “All TinyOS”, configurations 100% of the queue drops happened at the Contiki nodes. Notice that most queue drops occurred when 25% of the network’s node ran Contiki. The reason is that in such a configuration most of the children of a Contiki node will be TinyOS nodes that aggressively retransmit their packets, leading to queue overflows and packet drops. As the ratio of TinyOS nodes decreases, Contiki nodes face less pressure in their queues and PRR increases.

In conclusion, our findings suggest that different MAC-layer retransmission intervals and message queue sizes between the two implementations cause the performance degradation in mixed networks.

Our results thus far imply that when considering the interoperability of LLN IPv6 implementations one should not focus at a single layer, such as routing, but on system-wide interoperability. Would performance deteriorations, such as the one we discovered, disappear by using the same standardized MAC protocol? While MAC protocols can specify values for retransmission intervals, queue size are typically implementation-specific, constrained by the hardware resources available and affected by run-time aspects. In other words, even fully compliant systems can have different net-

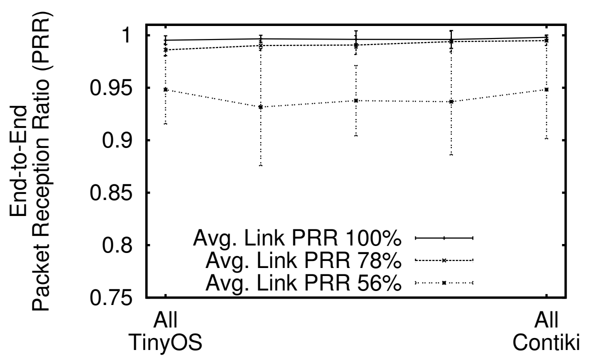

Figure 7. PRR is consistently high across all configurations when Contiki and TinyOS use the same values for MAC-layer parameters and queue sizes.

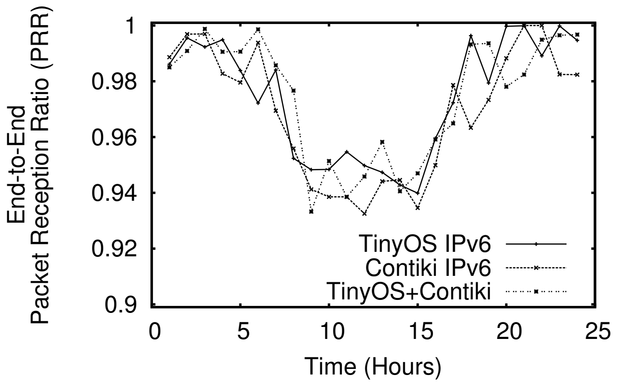

Figure 8. Per-hour PRR for the TinyOS and Contiki IPv6 stacks along with a 50/50 mix configuration over a single 24 hour period. PRR decreases during daytime hours and increases during periods with lower human activity.

work queue sizes.

To evaluate the effect of queue size on performance, we ran a simulation in which we configured Contiki and TinyOS to have the same MAC-layer retransmission intervals (50 milliseconds) but different queue sizes (four packets for Contiki and 10 for TinyOS). Figure 6 presents the PRR measurements for this scenario, using the lossiest link setup. A similar decrease in performance is evident in this case as well, although slightly less pronounced than in Figure 3.

Next, we investigate the case when we configure Contiki and TinyOS with the same MAC-layer retransmission (50 milliseconds) and queue size parameters (10 packets). Figure 7 presents the PRR results for low, medium, and high quality links. We notice that for all link conditions, the PRR performance is consistent, in contrast to Figures 3 and 6. This result suggest that achieving high performance in interoperable systems requires going further than simply testing standards compliance.

5.3 A Point of Comparison: Native Data Collection Protocols

To provide a point of comparison for the performance results of the IPv6 implementations, we compare them to those of the native Contiki and TinyOS data collection protocols, the Contiki Collect protocol and the TinyOS Collection Tree Protocol (CTP) [11]. Both Contiki Collect and TinyOS CTP are state-of-the-art address-free data collection protocols that provide a way for nodes to send data packets towards a data sink. Nodes do not need to know the address of the sink: the protocol takes responsibility for delivering the data to the closest sink. Both TinyOS CTP and Contiki Collect use ETX, finding paths that minimize the number of packet transmissions to reach the root. Moreover, neither CTP nor Contiki Collect are IPv6-based. CTP uses the TinyOS active messages network stack [1] while Contiki Collect uses the Contiki Rime stack [6].

At this point we wish to highlight that our comparison is explicitly unfair: the native data collection protocols use reliability mechanisms, such as extensive hop-by-hop recovery, which are not part of the best-effort UDP protocols used in our IPv6 experiments. Also, the native protocols use ETX-based route selection techniques that are known to outperform the hop count-based mechanisms that the RPL implementations use. We thus expect the IPv6-based networks to perform worse than the native data collection networks.

5.3.1 Packet Reception Ratio

Table 2 shows the daily, minimum, and maximum per-hour PRR computed over three 24-hour runs in an indoor testbed for TinyOS CTP, Contiki Collect, TinyOS IPv6 stack, Contiki IPv6 stack, and a heterogeneous network consisting of an equal number of IPv6-based TinyOS and Contiki nodes. We see that on average, the daily performance of the mixed IPv6 implementations are on par with CTP and Contiki Collect.

As Table 2 suggests, the PRR performance of the IPv6-based systems fluctuates more than that of CTP and Contiki Collect. Using Figure 8, in which we plot the per-hour PRR of TinyOS IPv6, Contiki IPv6 and the mix of the two implementations over 24 hours, we can see that the heterogeneity of interoperable implementations does not affect the performance of the network on a temporal scale. Instead, the results are due to differences in the protocols as such, not in their implementation.

5.3.2 Overhead and End-to-end Efficiency

Knowing that the average PRR performance of interoperable IPv6 implementations is comparable with custom sensor network data collection protocols, we set the next goal of our evaluation to measure the overhead caused by operating an interoperable network.

First, we examine the quality of the links that the different protocols select in Figure 9. Specifically, Figure 9 plots the average number of transmission attempts of the link to the parent node that each implementation selects. The results suggest that CTP selects paths with the lowest per-link transmission attempts and the links that Contiki Collect selects are the second best. In addition, we can notice that the links selected by the three IPv6-based configurations are very similar among themselves and not far for from CTP or Contiki Collect.

We also measure the end-to-end efficiency of the packets that reach the root. Figure 10 presents the average end-to-end number of transmission attempts of the successfully received packets and also present the average number of hops

Routing Protocol	Avg. Daily PRR (Std. Dev.)	Min. per-hour PRR	Max. per-hour PRR	ReTx Limit
TinyOS CTP	99.82% (0.11)	98.09%	100%	30
Contiki Collect	99.45% (0.12)	98.26%	100%	10
TinyOS IPv6	97.84% (1.06)	93.46%	100%	10
Contiki IPv6	98.01% (1.01)	92.79%	99.99%	10
TinyOS + Contiki (IPv6)	97.45% (1.20)	91.98%	100%	10

Table 2. Average daily packet reception ratio (PRR), minimum and maximum per-hour PRR computed over three 24-hour runs in an indoor testbed for five different network configurations: TinyOS CTP, Contiki Collect, TinyOS IPv6, Contiki IPv6, and a 50/50 mix of TinyOS and Contiki nodes. The Contiki IPv6 and TinyOS IPv6 nodes use the same MAC-layer retransmission interval and queue length parameters. The PRR performance of the interoperable IPv6-based configuration is comparable with TinyOS CTP and Contiki Collect, which are the state-of-the-art proprietary routing protocols included with each operating system. The ∼2% difference in the average PRR performance is due to the naive routing metric used by both RPL implementations (i.e., the hop-count based OF0).

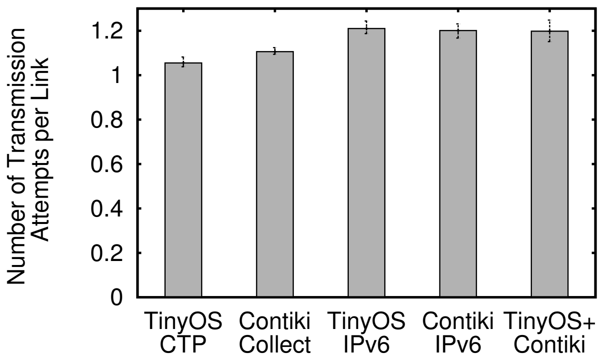

Figure 9. Number of transmission attempts of links selected by TinyOS CTP, Contiki Collect, TinyOS IPv6, Contiki IPv6, and a 50/50 mix of TinyOS IPv6 and Contiki IPv6. Each bar represents the average number of transmission attempts for links in the network and the error bars indicate the maximum and minimum values observed over three testbed runs. The overhead of uniform and mixed IPv6 configurations is on par with that of state-of-the art proprietary routing protocols.

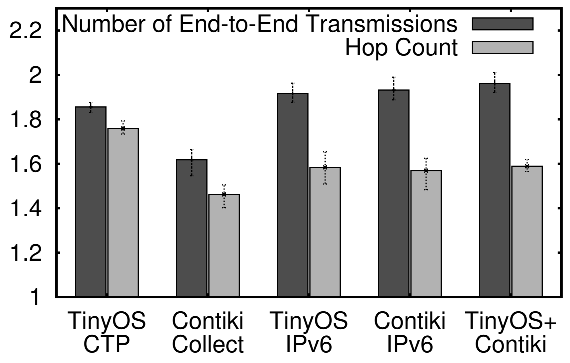

Figure 10. Average number of transmissions and per-packet hop count for TinyOS CTP, Contiki Collect, TinyOS IPv6, Contiki IPv6, and a 50/50 mix of TinyOS IPv6 and Contiki IPv6. The error bars indicate the maximum and minimum values observed over three testbed runs. The overhead of all IPv6 configurations is comparable to that of state-of-the-art proprietary protocols.

that these packets traveled. We can notice that the end-to-end number of transmission attempts of Contiki Collect is the most efficient. We attribute this to Contiki Collect incorporating an active probing scheme for precise link quality estimations. While the four-bit link estimator [10] that TinyOS CTP uses relies only on beacon sequence numbers to estimate ETX between nodes that do not exchange data packets, Contiki Collect sends explicit unicast probe messages to the neighbors that lack link quality samples when computing the ETX. This allows Contiki Collect to make more frequent and accurate link quality measurements thereby promptly discovering paths that can deliver packets with lower costs. The price that IPv6-based implementations pay is tolerable. Considering that the IPv6 implementations all use the OF0 objective function, which selects parents based hop count, using an objective function, such as MRHOF, that effectively minimizes path costs can potentially improve PRR.

6 Interoperability Experiences

The analysis of the performance of IPv6 implementations revealed that interoperability testing should be done at multiple layers of the protocol stack and that the performance of the resulting system must also be evaluated. If testing stops after interoperability is achieved, the performance of a network with multiple implementations might suffer.

6.1 Leverage Simulation

Network-scale simulation is an invaluable tool for interoperability performance testing. Although simulators cannot be expected to fully mimic the behavior of real wireless channels, they are effective in discovering the root causes underlying performance degradations caused from implementation-specific design choices. In this regard, our findings motivate the need for techniques that go beyond basic interoperability testing. The development of tools that quantify the expected performance of multiple "black-box" implementations and protocols that maximize performance by adjusting protocol parameters once a system is deployed are research directions that can have large impact in practice.

6.2 Distinguishing Causes for Performance Degradation

In our experience, both interoperability issues and protocol decisions cause performance deteriorations. For example, we found that native collection protocols outperform the standards-based interoperable IPv6 implementations. However, the reason in this case is that CTP and Contiki Collect use a better link metric (ETX) than the one RPL used in our tests (hop-count). Interestingly, we used the OF0 objective function, even though both RPL implementations support the ETX-based MRHOF objective function, because it simplified interoperability testing. Resolving the tension between ease of interoperability and high performance is an interesting avenue of research.

6.3 Intricate Interoperability Bugs

This paper extends our previous work on interoperability testing of IPv6 implementations for LLNs [15] through testbed experiments, a deeper analysis of the results, and a quantitative comparison with CTP and Contiki Collect.

Our earlier work was based on preliminary RPL implementations that lacked support for route poisoning and path validation. Including these components added a level of complexity to the system: an intricate bug in the route poisoning code of one of the stacks would trigger a bug in the other, causing a routing black hole. Packets were routed towards a node that had no outward routes where they were discarded. To find this bug, we studied a small set of simulation traces that exhibited the problem. The bug never manifested itself in any of our extensive testbed experiments, strengthening our belief in the utility of simulations in the testing of interoperable systems.

We hope that this work will influence protocol design, protocol implementation, and interoperability testing frameworks within organizations as the IETF, the IPSO Alliance, as well as with software vendors that implement these protocols. Both stacks are included in the TinyOS and Contiki operating system repositories, available from `http://www.tinyos.net/` and `http://www.sics.se/contiki/` respectively.

7 Related Work

Interoperability has always been a cornerstone of protocol standardization within the IETF: only protocol specifications with interoperable implementations can become standards. Experience has shown that implementation aspects have a significant impact on network performance. Paxson showed that subtle differences in implementations of TCP, the most widely used transport protocol on the Internet, have a deep impact on network performance [17]. This finding prompted work that subsequently resulted in the publication of RFC2525 [18], outlining potential TCP interoperability problems. Our aim with this work is to identify potential interoperability issues in RPL and in this way increase the stability and interoperability of future implementations.

In the sensor networking domain, mechanisms such as network types [2] and architectures such as Chameleon [6] can provide interoperability at the protocol header level, but not at the protocol logic level. Our experience shows that even when implementations interoperate at the protocol logic level, implementation decisions at multiple layers affect the performance of mixed networks.

8 Conclusions

As sensor networks move towards widespread industrial adoption, interoperability is becoming increasingly important. At the same time, the research community has largely ignored this issue, instead focusing on custom and incompatible systems. In this paper we present two interoperable implementations of the IPv6 protocol stack for LLNs for two leading sensor network operating systems: Contiki and TinyOS. We demonstrate that the two implementations interoperate, but also show that interoperability is not enough. Rather, we expose the fact that subtle differences in different layers of the protocol stack can affect the resulting system performance. Our findings imply that two systems that have not been jointly tuned can have substandard performance. Therefore, sensor network stacks should not only be tested for interoperability, but also for their expected performance in heterogeneous settings.

We envision this work as the initial step for further research on interoperable and high-performance sensor networks that can be deployed at commercial scales. Such research is important for the continued industrial and commercial adoption of wireless sensor networks. Moreover, interoperability testing of wireless sensor networks should evolve to test systems for performance as well as compatibility, as is the practice for testing equipment for wired networks.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments that helped us to improve this paper, our paper shepherd Anthony Rowe, and Omprakash Gnawali. This work was partially supported by NSF Awards #0855191, #0931843, #0932209 and the Swedish Strategic Research Foundation through the Promos project.

9 References

[1] P. Buonadonna, J. Hill, and D. Culler. Active message communication for tiny networked sensors, 2001. Available from www.tinyos.net.

[2] K. K. Chang and D. Gay. Language support for interoperable messaging in sensor networks. In *Proceedings of the 2005 workshop on Software and compilers for embedded systems*, pages 1–9, Dallas, Texas, 2005.

[3] S. Dawson-Haggerty. Design, implementation, and evaluation of an embedded IPv6 stack. Master's thesis, UC Berkeley, 2010.

[4] S. Dawson-Haggerty, X. Jiang, G. Tolle, J. Ortiz, and D. Culler. sMAP - a Simple Measurement and Actuation Profile for Physical Information. In *Proceedings of the Eighth ACM Conference on Embedded Networked Sensor Systems (SenSys)*, November 2010.

[5] A. Dunkels. Full TCP/IP for 8-bit architectures. In *Proceedings of The International Conference on Mobile Systems, Applications, and Services (MobiSys)*, San Francisco, CA, USA, May 2003.

[6] A. Dunkels, F. Österlind, and Z. He. An adaptive communication architecture for wireless sensor networks. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Sydney, Australia, November 2007.

[7] A. Dunkels and J-P Vasseur. IP for Smart Objects, September 2008. IPSO Alliance White Paper 1, available from www.ipso-alliance.org.

[8] 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 *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Raleigh, North Carolina, USA, November 2008.

[9] M. Durvy and M. Valente. Making Smart Objects IPv6 Ready: Where are we? In *Proceedings of the 2011 IAB Interconnecting Smart Objects with the Internet Workshop*, March 2011.

[10] R. Fonseca, O. Gnawali, K. Jamieson, and P. Levis. Four-bit wireless link estimation. In *Proceedings of the Workshop on Hot Topics in Networks (ACM HotNets)*, Atlanta, Georgia, USA, November 2007.

[11] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection tree protocol. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Berkeley, CA, USA, 2009.

[12] J. Hui and D. Culler. IP is Dead, Long Live IP for Wireless Sensor Networks. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Raleigh, North Carolina, USA, November 2008.

[13] J. Hui and JP. Vasseur. Rpl option for carrying rpl information in data-plane datagrams. Internet Draft (Work in Progress), IETF, 2010.

[14] X. Jiang, M. Van Ly, J. Taneja, P. Dutta, and D. Culler. Experiences with a high-fidelity wireless building energy auditing network. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Berkeley, CA, USA, 2009.

[15] J. Ko, J. Eriksson, N. Tsiftes, S. Dawson-Haggerty, A. Terzis, A. Dunkels, and D. Culler. ContikiRPL and TinyRPL: Happy Together. In *Proceedings of the workshop on Extending the Internet to Low power and Lossy Networks (IP+SN 2011)*, April 2011.

[16] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. Internet proposed standard RFC 4944, September 2007.

[17] V. Paxson. Automated packet trace analysis of TCP implementations. In *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communications (ACM SIGCOMM)*, Cannes, France, 1997.

[18] V. Paxson, M. Allman, S. Dawson, W. Fenner, J. Griner, I. Heavens, K. Lahey, J. Semke, and B. Volz. Known TCP Implementation Problems. RFC 2525 (Informational), March 1999.

[19] B. Priyantha, A. Kansal, M. Goraczko, and F. Zhao. Tiny web services: design and implementation of interoperable and evolvable sensor networks. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, pages 253–266, Raleigh, NC, USA, 2008.

[20] P. Thubert. Rpl objective function 0. Internet Draft (Work in Progress), IETF, 2010.

[21] J.P. Vasseur and A. Dunkels. *Interconnecting Smart Objects with IP: The Next Internet*. Morgan Kaufmann, 2010.

[22] T. Winter (Ed.), P. Thubert (Ed.), and RPL Author Team. RPL: IPv6 Routing Protocol for Low power and Lossy Networks. Internet Draft draft-ietf-roll-rpl-18, work in progress.