

Opening the Sensornet Black Box

Jung Il Choi, Jung Woo Lee, Megan Wachs, and Philip Levis
Computer Systems Laboratory
Stanford University
Stanford, CA 94305

Abstract

We argue that the principal cause of sensornet deployment and development difficulty is an inability to observe a network’s internal operation. We further argue that this lack of visibility is due to the activity and resource constraints enforced by limited energy. We present the Mote Network (MNet) architecture, which elevates visibility to be its dominant design principle. We propose a quantitative metric for network visibility and explain why network isolation and fairness are critical concerns. We describe the Fair Waiting Protocol (FWP), MNet’s single-hop protocol and show how its fairness and isolation can improve throughput and efficiency. We present the Pull Collection Protocol as a case study in designing multihop protocols in the architecture.

1 Introduction

Observing what occurs deep within a low-power sensornet is hard. This difficulty hinders development: the inability to monitor the internals of a sensornet transforms development from methodical debugging to a guessing game. The inherent energy constraints of these networks make observation hard. With unlimited energy, a node could keep detailed logs and send large amounts of debugging information. Sensornets are grey-box systems, where an operator has a few limited pieces of information with which to diagnose a problem or failure.

Two of our experiences developing network protocols illustrate this challenge. The first experience was when the TinyDB developers observed very high packet loss rates in a small test network deployed in the Intel Research Berkeley lab. Their hypothesis was that this was due to overflowing send queues. To test the hypothesis, they queried the network on the current queue depth. Because nodes with full queues were unable to enqueue responses to the query, this method never observed full queues. Working with the TinyDB developers, we discovered the cause of the queue overflows — transient routing loops — only after many hours with the TOSSIM simulator.

We had similar challenges when developing CTP [1], the collection protocol in TinyOS 2.0 [2]. Initial tests of isolated components and full protocol simulations were promising. The first real-world test was abysmal: 4% data yield. This sparked a three-week effort to determine the causes. The eventual solution was to integrate a comprehensive logging system that reported every important event to a testbed UART backchannel. This timestamped, high-fidelity view of the protocol explained when, where, and why every packet

was dropped. Once we could observe the internal operation of the network, it became easy to identify causes quickly and unambiguously. For the same tests, CTP now has a minimum yield of 98.2%.

To judge whether other users have shared similar experiences, we surveyed papers and technical reports describing deployment experiences and canvassed a subset of the low-power sensornet research community through the tinyos-help mailing list. In several cases we directly contacted the authors for details. Section 2 describes an overview of the results. More often than not, those queried *could not definitively identify the cause of deployment failures*. Furthermore, we found that the dominant identifiable cause was insufficient isolation between systems or protocols. Based on this observation, we argue that increasing network visibility will simplify system and network design as well as deployment. This simplification will lead to larger, more complex, and more advanced systems.

This paper proposes the Mote Network (MNet) architecture, whose design addresses the fundamental difficulties in deploying low-power sensornets. The *visibility principle* defines the cardinal goal of the MNet architecture:

“Minimize the energy cost of diagnosing the cause of a failure or behavior.”

The visibility principle has broad implications to system and network design. First, the simplest way to reduce the energy cost of diagnosis is to reduce the set of possible causes. For example, operating systems typically isolate processes to simplify debugging and diagnosis. But unlike an OS, which isolates computational processes that share a processor on a single node, the MNet architecture must isolate protocols across multiple nodes that share a wireless channel. Furthermore, in order to simplify diagnosis, the MNet architecture must *enforce* this protocol isolation. The simplest way to enforce isolation is to allow only one protocol to operate. Since this is not desirable, the second implication for the MNet architecture is that it must provide fairness between protocols: all protocols must have a chance to operate.

In the internet domain, TCP-friendly congestion control [9, 21] is an example of network isolation and fairness. Under common conditions, TCP-friendliness ensures that each of n flows receives approximately $\frac{1}{n+1}$ of the available bandwidth. Just as an OS isolates processes by giving them an equal virtualized share of the processor, TCP-friendly congestion control isolates applications by giving each an equal virtualized share of the network.

Unlike the Internet, sensornets have many multihop protocols, not all of which are end-to-end flows. This diversity calls for the “narrow waist” protocol of the architecture to be single-hop (layer 2) rather than multihop (layer 3) [7]. Protocol friendliness must correspondingly move down the stack, from transport (layer 4) to multihop (layer 3). We can take a lesson from the complications that UDP traffic introduces to the Internet’s stability [16]. Rather than require every transport protocol to implement certain mechanisms, the architecture can mandate it by incorporating them into a unifying narrow waist protocol.

In order to isolate protocols, the narrow waist must prevent a protocol from transmitting when its transmission will interfere with another protocol. The MNet architecture uses Fair Waiting Protocol (FWP) to achieve this goal. FWP can grant the channel to a packet recipient, suppressing all other nearby nodes and greatly reducing inter-protocol interference. Section 3 gives a brief description of FWP’s mechanisms for providing network isolation and fairness. We refer the reader to a technical report for further details [6].

The visibility principle of the MNet architecture also applies to higher level protocols. Section 4 is a case study of designing a network protocol in the MNet architecture. It describes the Pull Collection Protocol (PCP), a tree collection protocol that gives each node a fair share of the available bandwidth to the root. The case study shows how the visibility principle affects protocol design decisions and how FWP can enable high-bandwidth packet exchanges without sacrificing its isolation or fairness properties.

Section 5 discusses some implications of the architecture, states areas of intended future work, and briefly touches on major issues such as power conservation.

2 Background

The difficulty in deploying mote-based sensornets has motivated a large spectrum of research, from program analysis [24] to programming languages [11] to entire system architectures [10]. To better understand *why* developers encounter so many software problems, we reviewed the existing deployment literature and surveyed developers through mailing lists and personal communication. We broadly clump the observed failures into four major classes.

System interactions. Often, components that were designed to work in isolation interfered with each other at a systems level. Conflicting network protocol snooping requirements led to MAC protocol failures [18]. In some cases, base station failures – due to battery exhaustion [3, 18] or unpredicted program behavior [3, 12] – disconnected many motes from the network [4, 30].

Network saturation and congestion were major causes of correlated failures [3, 5, 12, 15, 25, 29]. Collisions occurred under heavy load [25], light but correlated load [5], or when routing protocols self-interfered [14]. Congestion affected link symmetry by congesting one direction of a link [12]. These problems were often attributed to environmental causes, such as weather or RF interference, but these hypotheses were uncertain [4, 5, 17, 30].

Protocol conflicts and failures. In some cases, a single protocol could create failures across the entire network. For example, Deluge could saturate the network and prevent other data transfers [18].

Unknown. In many cases the reason for failure was not known or could not be determined [5, 18, 29]. The many possible sources of problems can cause deployments to have significant debugging and remote querying logic which comprises up to 80% of the total source code [32].

2.1 Deployment Performance

Low-level failures degrade application-level performance. The Great Duck Island network had a median node data yield of 58% [27]. A deployment in a California redwood forest reported a median data yield of 40% [28]. A deployment designed to use the latest out-of-the box components reported a frustrating 2% data yield [18]. A more recent deployment at a volcano in Ecuador, nominally built with more mature technology, reported a median data yield of 68% [30].

Some losses had clear causes, such as a base station failure. Each deployment used a routing collection tree, and the cause of many losses remains unknown. In some cases, extensive post-facto analysis lead to reasonable hypotheses, but these cannot be validated [28].

2.2 Management and Debugging

The difficulties in understanding the causes of system failure have motivated several management and debugging tools. These tools, layered on top of existing systems, improve visibility by gathering data that would otherwise be internal to the network. They range in complexity from network snooping [12] to lightweight RPC [32].

The Sympathy system builds on these approaches, providing an expert system that can diagnose failures from gathered metrics [22]. The challenge that Sympathy faces is the cost of gathering needed information: it either requires frequent updates of node state metrics or a way to query the metrics.

The MNet architecture seeks to achieve the same goal as these systems – improving the visibility of a network – but takes a completely different approach. Rather than try to improve the visibility of an obfuscated network by adding additional layers on top of it, it improves the visibility of the network architecturally. The MNet architecture complements and seeks to improve all of these existing tools by using preventative measures: it simplifies Sympathy decision trees and reduces the need for RPC queries.

2.3 Visibility Metric

In order to compare how well protocols follow the visibility principle, we must have a quantifiable metric. Because the visibility principle deals with diagnosing the cause of a specific behavior, a protocol may have better or worse visibility for different behaviors. For example, a protocol may be designed for good visibility into why packets are dropped, but may provide poor visibility into why nodes reboot.

As an initial attempt to quantify visibility, we propose applying Sympathy’s decision tree approach. A protocol’s visibility for a behavior of interest can be quantified by measur-

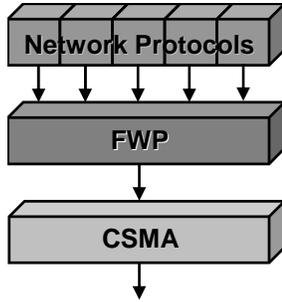


Figure 1. FWP sits between network protocols and a CSMA MAC.

ing the cost to reach a hypothesis for the cause. This cost corresponds to the energy cost of traversing the decision tree for that behavior. A protocol that has a smaller tree, or one that is less expensive to traverse, follows the visibility principle better. We leave whether causes are equally weighted or not as an open question, and expect that initially each is equally weighted.

2.4 Isolation and Fairness

The visibility principle leads to two major design criteria for network protocols: isolation and fairness. Isolation simplifies reasoning. For example, isolation between processes in an OS can make failures due to another program exceedingly rare. In a network architecture, isolation between protocols can make failures due to another traffic pattern similarly rare. Isolation by itself, however, is insufficient. While isolation ensures that two separate elements do not conflict, it does not promise that both of them can operate. In addition to isolation, the architecture must provide fairness. In combination, these two criteria allow debugging tools to shrink the decision tree and lead to more efficient diagnosis. The next section describes grant-to-send, the low-level protocol mechanism that the MNet architecture uses to provide a sound basis for these goals.

3 FWP: The Narrow Waist

The Fair Waiting Protocol (FWP) provides isolation and fairness between multihop protocols in the MNet architecture. Because many sensornet protocols are not end-to-end, FWP is a single-hop protocol. It sits between network protocols and a CSMA/CA MAC (Figure 1). FWP controls which packets to submit to CSMA and when to submit them. CSMA randomizes which node acquires a clear channel, and FWP inserts delays before CSMA to adjust a node's probability of acquisition based on which protocol that node wishes to transmit [6].

3.1 Mechanism

FWP's main goal is to isolate protocols, such that protocols do not interfere with each other's operation. This problem is unique to wireless sensornets because they employ multiple layer 3 protocols such as collection, routing, dissemination, and synchronization. The key insight behind FWP is that layer 3 protocol isolation requires inter-protocol collision avoidance. For example, a collection protocol

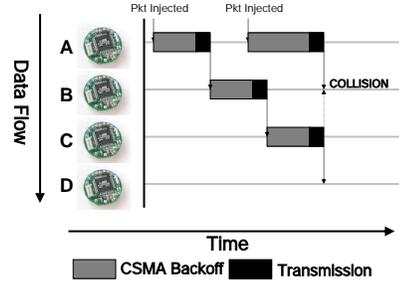


Figure 2. An example scenario of intra-path collision. Solid lines are received packets, dashed lines are overheard packets. When node A and its grandparent send packets it collides at node B.

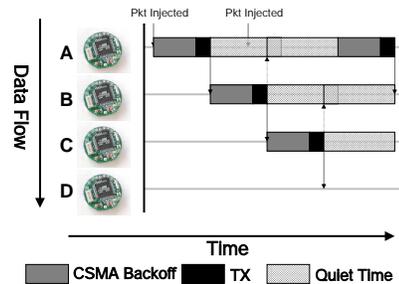


Figure 3. Grant-to-send mechanism example. Solid lines are received packets, dashed lines are overheard packets. With grant-to-send mechanism, A sends a packet to B with a nonzero grant-to-send. A, the transmitter, must be quiet for the duration of the grant-to-send, but B, the receiver, is not suppressed. When B sends to C, both B and A are suppressed. A must wait until both grants have expired. Thus even if A has a packet to send, the transmission is suppressed while previous packets exit the collision range. With this mechanism, FWP aims to clear the channel for the receiver and the protocol which it selects to send.

should not fail because a dissemination protocol's packet bursts corrupt routing beacons.

Unlike layer 2 approaches, such as RTS/CTS, FWP aims to utilize layer 3 information to avoid collisions across multiple hops of a data flow. Figure 2 shows a simple example scenario of packet loss due to an intra-path collision. To avoid collision, nodes must wait until their grandparents forward the previous packet [33, 19]. However, even if every protocol introduces waits between its packets to prevent self-interference, each protocol can transmit during another's waiting period. Correctly preventing self-interference across protocols requires a shared mechanism between them.

FWP enables layer 3 protocols to share information by placing an additional collision avoidance layer between layers 2 and 3. An FWP transmission request includes a *grant-to-send* value along the packet to transmit. FWP puts this value in a packet as a one byte header. A grant-to-send is a quiet time during which only the recipient of the packet may transmit. During this quiet time, nodes that overheard or transmitted the packet may not use the channel. Thus a transmission *grants* the channel around the transmitter for the recipient *to send*. Figure 3 shows an example of FWP operating across a route. Although packets are injected at the same time as in Fig. 2, B's grant to C forces A to wait, preventing

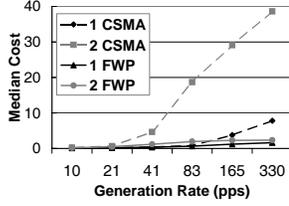


Figure 4. Median packet delivery costs - retransmissions per successful transmission - for 1 and 2 instances of CTP running on bare CSMA and over FWP on 165-node network. FWP effectively isolates the two instances from each other to reduce packet retransmissions.

self-interference along the path. When all quiet times expire, FWP submits a packet to the underlying CSMA.

In practice, FWP does not prevent all packet collisions. Since FWP makes transmission estimates based on history, it only prevents collision on the tails of forwarding flows. RTS/CTS can be a perfect solution for collision avoidance. However, RTS/CTS does not easily support broadcasts, a common primitive in sensor networks such as Deluge [13]. Furthermore, for the small datagrams typical of sensor networks, control overhead of RTS/CTS can be large. In contrast, FWP preserves all flexibilities of CSMA with a small overhead of one byte per packet.

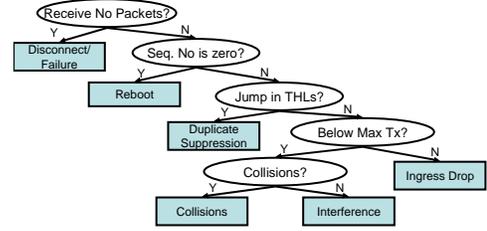
Figure 4 shows how FWP’s isolation affects the performance of two collection protocols running on the 165-node motelab testbed [31]. The figure shows the cost of delivering packets with TinyOS 2.0’s Collection Tree Protocol (CTP) [1] running over plain CSMA and over FWP. In the FWP experiment, CTP data packets have a quiet time of one packet time and CTP routing beacons have a quiet time of zero. While both handle a single instance of CTP well, two instances of CTP running over bare CSMA interfere with each other heavily. This is because CTP has built-in rate-limiting mechanisms that prevent self-interference, but these methods are ineffective when another protocol is simultaneously using the channel. FWP isolates the two instances of the protocol, resulting in lower packet delivery costs. Its suppression mechanism enforces rate-limiting across protocols, limiting the sending rate to what the network can handle. Because FWP drastically reduces the effects of protocols on one another, it simplifies debugging and makes identifying causes of failure easier.

Collision avoidance alone, however, does not provide isolation. If a protocol sends a burst of packets with maximum quiet time, other protocols would suffer suppression indefinitely. As stated in section 2.4, we also need to provide fairness across protocols. FWP adopts fair queueing defined by Demers et al [8]. As the grant durations can be viewed as a channel occupation, FWP keeps track of channel usage of protocols by adding quiet times and air times. When multiple transmission requests are submitted to FWP, it selects the packet with the least channel usage. Therefore, protocols with more packets or larger quiet times are penalized to give priority for protocols that have occupied the channel less.

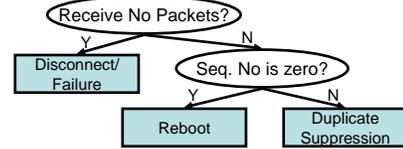
Unlike programming abstractions such as SP, FWP is a protocol and is therefore OS, language, and platform-

Disconnection	Temporarily or permanently broken link.
Destruction	Depleted batteries or permanent hardware failure.
Reboot	Software failure loses packets in RAM.
Egress drop	Retransmit threshold is reached.
Ingress drop	Receiving a packet when the queue is full.
Suppression	Temporary loops cause nodes to mistake looped packets as duplicates and drop them.

Table 1. Causes of packet loss for collection tree protocols. Temporary disconnections can introduce huge latencies, which may or may not actually drop packets. For example, if a disconnected node thinks it has no parents, it will not encounter egress drops, but if it erroneously thinks it has parents, it will.



(a) Traditional Routing Protocol Decision Tree



(b) PCP Decision Tree

Figure 5. Decision trees for identifying causes of data loss.

independent. While our current implementation is for the CC2420 radio under TinyOS 2.0, we do not foresee challenges porting it to other OSes or CSMA layers.

Our initial experiments with CTP suggest that FWP can support and improve visibility for existing protocols by providing isolation and rate-limiting within a local node neighborhood. The next section describes a new protocol which is designed with FWP in mind, and explains how we can leverage the properties of the underlying network to create a more visible system.

4 PCP: A Design Example

To demonstrate the implications of this network architecture, we present the design of a tree collection protocol, Pull Collection Protocol (PCP), which runs over FWP and follows the visibility principle. In a collection protocol, a major behavior of interest is why packets are not received from a source. Table 1 lists the conditions that prevent traditional routing protocols, such as MintRoute [34] or ARC [33], from delivering packets.

Following the visibility principle, we want to minimize the energy required to diagnose the cause of packet losses. This corresponds to the energy required to traverse the decision tree shown in Figure 5(a). We can minimize the cost both by removing leaves from the tree, and by minimizing the number of additional queries or updates required to traverse the remaining portion.

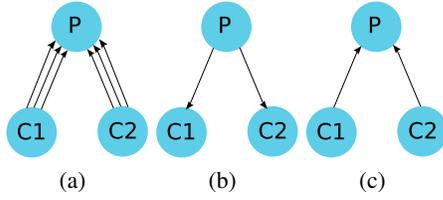


Figure 6. Traditional Push-Based Method for Rate Limiting: a) Children send data to parent at high rate. b) Parent sends rate-limiting information. c) Children send data to parent at reduced rate.

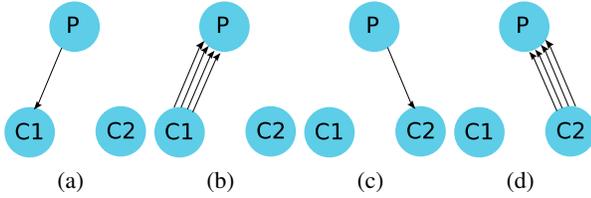


Figure 7. Pull-Based Method for Rate Limiting: a) Parent sends grant-to-send to one child. b) Child sends a burst of packets to the parent. c) Parent sends grant-to-send to another child. d) Child sends a burst of packets to the parent.

To reduce the number of leaves in the tree, PCP eliminates causes of packet loss. PCP uses a novel approach to limit ingress drops. Unlike traditional push-based protocols (Fig. 6), in PCP, sinks pull data from the network (Fig. 7). Parents use FWP’s grant-to-send mechanism to request a burst of packets from a child. The grant-to-send ensures that there is no interference from other protocols while the child transmits. Children keep their buffers full by requesting packets from their children. This is similar to 802.11e’s PCF protocol; however, grant-to-send allows PCP to work over wireless hops while PCF requires a wired backchannel.

PCP’s design requires that packets be sent in bursts from the child. This is because experimental results have shown that it can be advantageous to send packets in bursts, because links remain stable and link estimations are more reliable [26]. The use of FWP, with its limits on channel usage and enforcement of fairness between protocols, seems in opposition to this goal. Actually, FWP can facilitate bursts, because it can guarantee that the channel will be clear for a node to send many packets for an arbitrarily long time. The isolation property ensures that another protocol will not disturb the stream of packets until the transmitter’s grant-to-send expires.

PCP removes another leaf from the decision tree by eliminating ingress drops. It does this by allowing infinite retransmissions of a packet. Because all data packets have the same destination, there is no reason to penalize one packet in favor of another. Because PCP is isolated from other protocols, we can expect that the losses due to collisions and interference from other protocols are minimal. Thus, PCP shortens the decision tree into the one shown in Figure 5(b).

PCP’s design allows the traversal of the remainder of the tree without additional queries. The source node fills in the sequence number of the packet and sets the Time-Has-

Lived(THL) to 0. The THL field is incremented at each hop the packet takes. If no packets are received from a node after many requests, it is dead or disconnected. The sequence number field of the packet, which is used by the protocol itself for duplicate suppression, can indicate if a node rebooted if the sequence number returns to 0. Observing this can explain correlated packet loss from a subtree.

The open research question for PCP is how to avoid maintaining too much per-child state. We are experimenting with probabilistic methods of counting and voting [20] to allow balanced pulling from children without maintaining state.

5 Extensions and Limitations

Increasing visibility has a cost. Fairness and isolation introduce delay, thereby increasing latencies. Our results for CTP suggest that delay can improve network performance under heavy load by preventing collisions. When load is very light, protocols can use quiet times of zero, reverting to a standard CSMA with fair queueing. Exploring quiet time selection algorithms is a clear area of future work: we plan to revisit a range of protocols from the literature and investigate how they could be optimized within the MNet architecture.

5.1 Low Power

Our goal is to start with a simple, flexible architecture which allows optimization later, as has been the case in successful abstractions such as files, threads, and TCP. Reducing power consumption is one such critical optimization, and instances of the MNet architecture can use many different approaches.

One way to save power is to use low power listening with packet bursts. Prior work [26] showed packet bursts can require fewer retransmissions. Packet bursts also work well with low power listening, as a single long preamble can be amortized over many packets. Grant-to-send interferes with this approach when bursts are transmitter-driven, as a transmitter must keep quiet after transmitting a single packet with a nonzero grant-to-send. However, PCP showed a way in which receiver-driven grant-to-send can be used to request an uninterrupted burst of packets.

5.2 Security

Designing a new network architecture from scratch allows us to incorporate security from the beginning. FWP raises several open questions such as snooping encrypted MAC frames and detecting cheaters or colluding suppressors. Fairness provides simple mechanisms to detect egregious cheaters – they aren’t fair – and we are currently studying how to take advantage of FWP’s isolation and fairness to introduce security into the narrow waist.

5.3 Isolation and Fairness

The need for network isolation affects system implementation. For example, if an operating system does not allocate packet buffers fairly to protocols, then it is possible they will not be able to offer equal loads, thereby compromising fairness. Similarly, if an OS does not isolate the software of the protocols from one another, then a failure in one can cascade, increasing the size of the diagnosis decision tree. With inter-

protocol isolation, a network monitoring protocol can work even when others malfunction.

FWP provides fairness between protocols, but not within them. For example, IFRC [23] and ARC [33] protocols provide *node* fairness, in that they seek to give each node in a collection tree an equal share of the bandwidth to the collection sink. FWP provides fairness at the level of single-hop communication: protocols built on top of it (such as IFRC and ARC) can provide higher levels of fairness as needed, with the knowledge that FWP will give them a fair share of local bandwidth.

Acknowledgements

We would like to thank Rodrigo Fonseca and Sukun Kim, whose rate control research provided the inspiration for FWP. This work was supported by generous gifts from the Intel Research and DoCoMo Capital, scholarships from the Samsung Scholarship and the Korea Foundation for Advanced Studies, the National Science Foundation under grant #0615308 (“CSR-EHS”), a Stanford Graduate Fellowship, and a Stanford Terman Fellowship.

6 References

- [1] TEP 123: Collection Tree Protocol. <http://www.tinyos.net/tinyos-2.x/doc/>.
- [2] TinyOS 2.0. <http://www.tinyos.net/tinyos-2.x/>.
- [3] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita. A line in the sand: A wireless sensor network for target detection. *Computer Networks (Elsevier)*, 46, 2004.
- [4] R. Beckwith, D. Teibel, and P. Bowen. Unwired wine: Sensor networks in vineyards. In *Proceedings of IEEE Sensors*, 2004.
- [5] P. Buonadonna, D. Gay, J. Hellerstein, W. Hong, and S. Madden. Task: Sensor network in a box. In *Proceedings of the Second European Workshop on Wireless Sensor Networks (EWSN)*, 2005.
- [6] J. I. Choi and P. Levis. Grant to send: Fairness and isolation in low-power wireless. Technical Report SING-06-01, Stanford Information Networks Group, 2006.
- [7] D. Culler, P. Dutta, C. T. Ee, R. Fonseca, J. Hui, P. Levis, J. Polastre, S. Shenker, I. Stoica, G. Tolle, and J. Zhao. Towards a sensor network architecture: Lowering the waistline. In *Proceedings of the Tenth Workshop on Hot Topics in Operating Systems (HotOS-X)*, 2005.
- [8] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. In *Proceedings of the ACM SIGCOMM*, 1989.
- [9] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the internet. *IEEE/ACM Transactions on Networking*, 1999.
- [10] O. Gnawali, B. Greenstein, K.-Y. Jang, A. Joki, J. Paek, M. Vieira, D. Estrin, R. Govindan, and E. Kohler. The TENET architecture for tiered sensor networks. In *Proceedings of the Fourth ACM Conference On Embedded Networked Sensor Systems (SenSys)*, 2006.
- [11] R. Gummadi, N. Kothari, R. Govindan, and T. Millstein. Kairos: a macro-programming system for wireless sensor networks. In *Proceedings of the Twentieth ACM symposium on Operating systems principles*, 2005.
- [12] T. He, S. Krishnamurthy, L. Luo, T. Yan, L. Gu, R. Stoleru, G. Zhou, Q. Cao, P. Vicaire, J. A. Stankovic, T. F. Abdelzaher, J. Hui, and B. Krogh. Vigilnet: An integrated sensor network system for energy-efficient surveillance. *ACM Transactions on Sensor Networks (TOSN)*, 2006.
- [13] J. W. Hui and D. Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. In *Proceedings of the Second ACM Conference On Embedded Networked Sensor Systems (SenSys)*, pages 81–94, New York, NY, USA, 2004. ACM Press.
- [14] S. Kim, R. Fonesca, P. Dutta, A. Tavakoli, D. Culler, P. Levis, S. Shenker, and I. Stoica. Flush: A reliable bulk transport protocol for multihop wireless networks. Technical Report UCB/EECS-2006-169, University of California, Berkeley, 2006.
- [15] P. Kimelman. Personal communication, 2006.
- [16] E. Kohler, S. Floyd, and M. Handley. Designing dccp: Congestion control without reliability. In *Proceedings of the ACM SIGCOMM*, 2006.
- [17] L. Krishnamurthy, R. Adler, P. Buonadonna, J. Chhabra, M. Flanigan, N. Kushalnagar, L. Nachman, and M. Tarvis. Design and deployment of industrial sensor networks: Experiences from a semiconductor plant and the north sea. In *Proceedings of the Third ACM Conference On Embedded Networked Sensor Systems (SenSys)*, 2005.
- [18] K. Langendoen, A. Baggio, and O. Visser. Murphy loves potatoes: Experiences from a pilot sensor network deployment in precision agriculture. In *the Fourteenth Int. Workshop on Parallel and Distributed Real-Time Systems (WPDRTS)*, 2006.
- [19] T. Moscibroda, R. Wattenhofer, and Y. Weber. Protocol design beyond graph-based models. In *Proceedings of the 5th ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets)*, 2006.
- [20] S. Nath, P. B. Gibbons, S. Seshan, and Z. R. Anderson. Synopsis diffusion for robust aggregation in sensor networks. In *Proceedings of the Second ACM Conference On Embedded Networked Sensor Systems (SenSys)*, 2004.
- [21] T. J. Ott, J. H. B. Kemperman, and M. Mathis. The stationary behavior of ideal tcp congestion avoidance. *IEEE/ACM Transactions on Networking*, 1999.
- [22] N. Ramanathan, K. Chang, R. Kapur, L. Girod, E. Kohler, and D. Estrin. Sympathy for the sensor network debugger. In *Proceedings of the Third ACM Conference On Embedded Networked Sensor Systems (SenSys)*, 2005.
- [23] S. Rangwala, R. Gummadi, R. Govindan, and K. Psounis. Interference-aware fair rate control in wireless sensor networks. In *Proceedings of the ACM SIGCOMM*, 2006.
- [24] J. Regehr, A. Reid, and K. Webb. Eliminating stack overflow by abstract interpretation. *ACM Transactions on Embedded Computing Systems (TECS)*, 2005.
- [25] T. Schmid, H. Dubois-Ferriere, and M. Vetterli. Sensorscope: Experiences with a wireless building monitoring sensor network. In *Proceedings of the Workshop on Real-World Wireless Sensor Networks (REALWSN)*, 2005.
- [26] K. Srinivasan, P. Dutta, A. Tavakoli, and P. Levis. Some implications of low-power wireless to ip routing. In *Proceedings of the Fifth Workshop on Hot Topics in Networks (HotNets-V)*, 2006.
- [27] R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler. An analysis of a large scale habitat monitoring application. In *Proceedings of the Second ACM Conference On Embedded Networked Sensor Systems (SenSys)*, 2004.
- [28] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, , and W. Hong. A microscope in the redwoods. In *Proceedings of the Third ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2005.
- [29] V. Turau, C. Renner, M. Venzke, S. Waschik, C. Weyer, and M. Witt. The heathland experiment: Results and experiences. In *Proceedings of the Workshop on Real-World Wireless Sensor Networks (REALWSN)*, 2005.
- [30] G. Werner-Allen, K. Lorincz, J. Johnson, J. Leess, and M. Welsh. Monitoring volcanic eruptions with a wireless sensor network. In *Proceedings of the Second European Workshop on Wireless Sensor Networks (EWSN)*, 2005.
- [31] G. Werner-Allen, P. Swieskowski, and M. Welsh. Motelab: a wireless sensor network testbed. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, 2005.
- [32] K. Whitehouse, G. Tolle, J. Taneja, C. Sharp, S. Kim, J. Jeong, J. Hui, P. Dutta, and D. Culler. Marionette: Using rpc for interactive development and debugging of wireless embedded networks. In *Proceedings of the Fifth International Conference on Information Processing in Sensor Networks: Special Track on Sensor Platform, Tools, and Design Methods for Network Embedded Systems (IPSN/SPOTS)*, 2006.
- [33] A. Woo and D. E. Culler. A transmission control scheme for media access in sensor networks. In *Proceedings of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, 2001.
- [34] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of multihop routing in sensor networks. In *Proceedings of the First ACM Conference On Embedded Networked Sensor Systems (SenSys)*, Los Angeles, CA, Nov. 2003.