# Opening the Sensornet Black Box

Technical Report SING-06-03

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 network visibility to be the dominant network design principle. We propose a quantitative metric for network visibility and explain why network isolation and fairness are critical concerns. We descibe 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 (PCP) as a case study in designing efficient multihop protocols in the architecture.

## 1. INTRODUCTION

This paper argues that the often-cited difficulty in programming low-power sensornets is due to a lack of visibility into their operation and behavior. While other factors contribute to making development challenging, the principal cause is the fact that an operator cannot easily and inexpensively observe what is happening within a network. This lack of visibility is a direct result of energy constraints. 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 routing protocol experiences present good examples of this phenomenon. The first involved a situation where the TinyDB developers observed very high packet loss rates in a small test network deployed in the Intel Research Berkeley lab. To test the hypothesis of overflowing send queues, the test query had queue depth as a field. As a full queue prevented the query from enqueueing a packet reporting its size, however, they were unable to observe a full queue. The cause of the queue overflows, — transient routing loops — was only discovered after many hours with the TOSSIM simulator.

The TinyOS net2 Working Group had similar challenges when developing CTP [1], the collection protocol in TinyOS 2.0 (T2) [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 reports every important event to a testbed UART backchannel. This timestamped, high-fidelity view of the protocol explains 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 unambigiously. For the same tests, CTP now has a minimum yield of 98.2%.

This paper proposes the Mote Network (MNet) architecture, whose design addresses the fundamental difficulties in deploying mote systems. The principal principle of the MNet architecture can be summarized as:

> "Minimize the energy cost of diagnosing
> the cause of a failure or behavior."

The visibility principle has broad implications to system and network design. Isolation emerges as a fundamental design goal, as it minimizes unforeseen interactions, thereby simplifying the system and reducing the number of possible causes. System isolation is a long-standing topic in operating systems (e.g., processes and virtualization) and a major consideration in sensornet OS design [11, 13]. Where system isolation deals with interactions on a single node, network isolation addresses the problem of protocols interacting or interfering across many nodes.

In the internet domain, one common example of network isolation is TCP-friendly congestion control [9, 22]. Under certain conditions, TCP-friendliness ensures that each of $n$ flows receives approximately $\frac{1}{n+1}$ of the available bandwidth. Just as an operating system isolates processes by giving them an equal virtualized share of the processor, TCP-friendly congestion control isolates applications by giving them 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 [17]. Rather than require every transport protocol to implement the needed mechanisms, the architecture can mandate it by incorporating them into the unifying narrow waist protocol.

To judge whether our visibility hypothesis is valid, 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 to have further 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.

In Section 2 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. In short, it will lead to improved research and technological progress. Section 2 concludes with a suggestion of a quantitative measure of network visibility and how two factors – fairness and isolation – are critical to improving this metric.

Section 3 describes the Fair Waiting Protocol (FWP), the unifying protocol of the MNet architecture. FWP sits on top of a CSMA MAC and isolates network protocols using a novel grant-to-send (GTS) mechanism. A grant-to-send transmission can grant the channel to the recipient by enforcing a quiet time on other nearby nodes. FWP uses GTS quiet times to allocate the channel fairly. Section 3 gives a brief description of FWP's mechanisms to address issues that arise from inconsistent views of the channel. We refer the reader to a tech report for further details [6].

Section 4 is a case study of how one might design a network protocol in the MNet architecture. It describes the Push 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 be used to 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 [26] to programming languages [12] 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 have led to MAC protocol failures [19]. In some cases, base station failures – due to battery exhaustion [3, 19] or unpredicted program behavior [3, 14] – disconnected many motes from the network [4, 34].

**Network saturation and congestion** were major causes of correlated failures [3, 5, 14, 16, 27, 31]. Collisions occur under heavy load [27], light but correlated load [5], or when routing protocols self-interfere [15]. Congestion affected link symmetry by congesting one direction of link [14]. These problems were often attributed to environmental causes, such as weather or RF interference [4, 5, 18, 34].

**Protocol conflicts and failures.** In some cases, a single protocol can create failures across the entire network. For example, Deluge can saturate the network and prevent other data transfers [19].

**Unknown.** In many cases the reason for failure is not known or could not be determined [5, 19, 31]. The presence of so many possible sources of problems leads many deployment codes to have significant debugging and remote querying code which in some cases comprises 80% of the total source code [35].

## 2.1 Deployment Performance

Low-level failures degrade application-level performance. The Great Duck Island network had a median node data yield of 58% [29]. A deployment in a California redwood forest reported a median data yield of 40% [30]. A deployment designed to use the latest out-of-the box components reported a frustrating 2% data yeild [19]. A more recent deployment at a volcano in Ecuador, nominally built with more mature technology, reported a mean data yield of 68% [34].

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 they cannot be validated [30].

## 2.2 Management and Debugging

The difficulties in understanding the causes of system failure have motivated several management and debugging tools. These tools improve visibility by layering on top of existing systems and providing a mechanism to gather data that would otherwise be internal to the network. They range in complexity from network snooping [14] to lightweight RPC [35].

The Sympathy system builds on these approaches, providing an expert system that can diagnose failures from gathered metrics [24]. 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 goals as these systems – improving the visibilty 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. As an initial attempt to do so, we propose borrowing Sympathy's decision tree approach. A protocol's visibility can be quantified by measuring the energy cost to traversing the decision tree in order to reach a hypothesis. A protocol that has a smaller or less expensive tree 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 MNet uses to provide a sound basis towards these goals.

# 3. FWP: THE NARROW WAIST

Fair Waiting Protocol (FWP) [6] is a narrow waist protocol which sits between network protocols and a CSMA/CA MAC. FWP provides network protocol isolation and fairness by controlling which packets to submit to CSMA and when to submit them. Since not all sensornet protocols are end-to-end, FWP is single-hop.

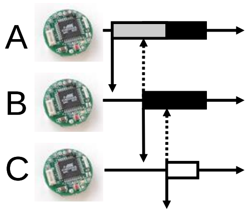


**Figure 1:** GTS Mechanism example. Solid lines are received packets, dashed lines are overheard packets. Boxes represent quiet times. Node A sends a packet to B with a nonzero quiet time. A, the transmitter, must be quiet, but B, the receiver, is not supressed. When B sends to C, both B and A are supressed. A must wait the maximum of the two quiet times. With this mechanism, FWP aims to clear the channel except for the receiver and the protocol which it selects to send.

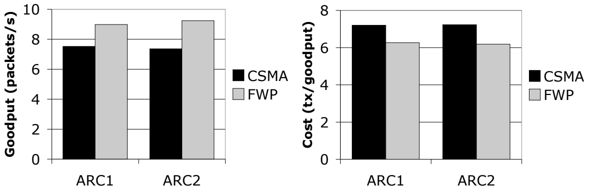
FWP provides isolation using a grant-to-send (GTS) mechanism. GTS adds one additional byte to the normal packet header to indicate a post-transmission quiet time. During this time, only the recipient of the packet may transmit, suppressing transmissions by the sender and overhearing nodes. Figure 1 shows a simple example of this mechanism in a multihop line-topology network. One basic use of GTS is to enforce traffic rate-limiting [25] across protocols.

FWP provides fairness between protocols with Demers et al.'s fair queueing algorithm [8]. FWP uses the grant duration and packet transmission times to estimate how long the channel has been occupied by a protocol. When there are requests for transmission from multiple network protocols, FWP chooses the protocol with the least channel occupancy and submits it to CSMA. Protocols which reserve the channel for long periods send fewer packets.

Inconsistent views of the channel, however, can have detrimental effects on fairness. Channel views are inherently different in multihop networks, and packet losses complicate this even further. Periodically decaying the protocol channel utilizations filters out small aberrations and restores fairness. In a 40-node testbed experiment, periodic decaying channel utilization improves the median Jain fairness index from 0.40 to 0.99.

CSMA is fair to nodes, not protocols. Differences in traffic patterns between motes can result in unfairness between protocols at the channel level. The algorithm by Vaidya et al. for providing MAC-level fairness [32] indicates that delay needs to be introduced before beginning CSMA. Therefore, in addition to deciding which packet to transmit, FWP also adjusts when it submits it to the CSMA layer. A larger range of delay values improves fairness at the cost of throughput. We are currently exploring delay functions to better understand the tradeoffs.

If every packet has a quiet time of 0, then FWP operates as a standard CSMA layer albeit with fair queueing (packet lengths are factored into channel use). FWP leaves choosing quiet times to network protocols, based on their requirements and traffic patterns. For example, single-hop protocols such as Trickle [20] may spec-



(a) ARC goodput. (b) ARC cost.

**Figure 2:** Goodput and cost for two separate ARC instances running in the presence of PSFQ and Trickle on top of CSMA and FWP. FWP increases goodput by 23-30% and decreases cost by 5-10%.

| | |
|---|---|
| **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 CTP packet loss. 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.

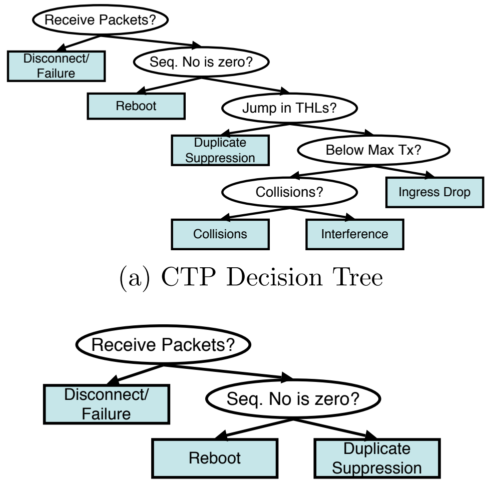
ify quiet times of 0, while routing protocols such as ARC [36] may specify quiet times that ensure no one transmits before a packet leaves the local interference range. Link quality and topology also affect the necessary quiet time, so network protocols should have adaptive mechanisms to find the best quiet time. While we have explored a few simple cases such as those above, estimating and calculating quiet time is an open area for future research. Section 4 describes one way to obtain high bandwidth using quiet times.

FWP does not aim to be a perfect protocol. CSMA race conditions preclude providing perfect isolation, and quiet times can reduce network capacity by introducing suppression and delay. However, the isolation benefits of choosing good quiet times can outweigh these costs. Figure 2 shows the performance of FWP in 40-node experiment running four protocols: separate ARC [36], PSFQ [33], and Trickle [20]. With proper parameters, there is a 23-30% increase in the goodput and 5-10% decrease in the energy/goodput for ARC. PSFQ and Trickle are minimally affected by FWP due to their low traffic rates. This result shows that isolation can not only improve visibility but also improve throughput by preventing interference.

Because FWP is a protocol rather than a programming abstraction [23], it is OS and platform-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.

# 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 follows the visibility principle. This protocol is based on T2's Collection Tree Protocol (CTP) [1]. Table 1 lists the conditions that prevent CTP from delivering packets.



(a) CTP Decision Tree

(b) PCP Decision Tree

**Figure 3:** Decision trees for identifying causes of data loss. Left branches are true; right branches are false.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C | P | reserved | | | | | | THL | | | | | | | |
| ETX | | | | | | | | | | | | | | | |
| Origin Node | | | | | | | | | | | | | | | |
| Sequence No. | | | | | | | | Protocol ID | | | | | | | |
| Data... | | | | | | | | | | | | | | | |

**Figure 4:** CTP data packet header.

Following the visibility principle, we want to minimize the energy required to traverse the decision tree shown in Figure 3(a). We can do this both by removing leaves from the tree, and by minimizing the amount of additional queries or updates required to traverse the remaining portion.

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 in CTP and other traditional pull-based protocols (Figure 5), in PCP, sinks pull data from the network(Figure 6). Parents use FWP's GTS mechanism to request packets from a child. Children keep their buffers full by requesting packets from their children. This is similar to 802.11e's PCF protocol; however, GTS allows PCP to work over wireless hops while PCF requires a wired backchannel.

PCP removes another leaf from the 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. Thus, PCP shortens the decision tree into the one shown in Figure 3(b).

PCP's design allows the traversal of the remainder of the tree without additional queries. PCP uses CTP's 8-

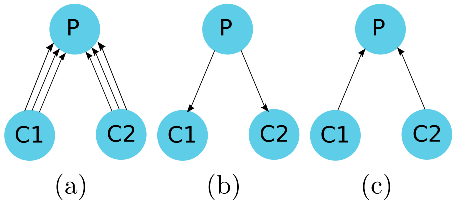


**Figure 5:** Traditional Push-Based Method for Rate Limiting: a) Children send data to parent at high rate. b) Parent sends rate-limiting information. c) Children send data to parent at reduced rate.

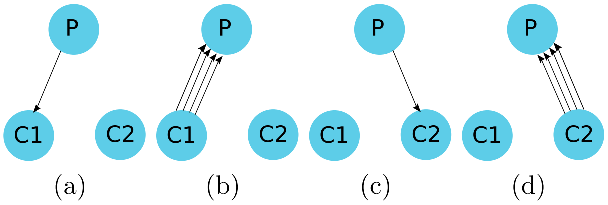


**Figure 6:** Pull-Based Method for Rate Limiting: a) Parent sends GTS to one child. b) Child sends a burst of packets to the parent. c) Parent sends GTS to another child d) Child sends a burst of packets to the parent

byte packet header format, shown in Figure 4. As with CTP, a source node fills in the sequence number and the origin field, and sets the Time-Has-Lived(THL) to 0. 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 THL field indicates the number of hops the packet has traversed so far. This is used by the protocol to avoid false positive duplicate suppression. At the sink, a sudden increase in THL of received packets means that there was a temporary routing loop in the network, so correlated losses were probably caused by false positive duplicate suppression.

Experimental results have shown that it can be advantageous to send packets in bursts, because links remain stable and link estimations are more reliable [28]. 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 quiet time. The isolation property of FWP ensures that another protocol will not disturb the stream of packets until the reserved quiet time runs out.

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 [21] 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 ARC 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 network. 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.

FWP assumes that nodes can snoop on all packets. For radios such as the ChipCon CC2420, this can come at the cost of relinquishing hardware mechansisms such as synchronous acknowledgements. Our current software FWP implementation does not have layer 2 acks, but we are actively working with the TinyOS CC2420 developers to reintroduce them.

## 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 is using low power listening with packet bursts. Prior work [28] showed packet bursts can have 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 obey the quiet time. However, PCP showed a way in which receiver-driven GTS can be used to send 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 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.

While FWP provides fairness between protocols, but not within them. For example, IFRC [25] and ARC [36] 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.

## 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] L. Gu and J. A. Stankovic. t-kernel: Providing reliable os support to wireless sensor networks. In *Proceedings of the Fourth ACM Conference On Embedded Networked Sensor Systems (SenSys)*, 2006.

[12] 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.

[13] C.-C. Han, R. Kumar, R. Shea, E. Kohler, and M. Srivastava. A dynamic operating system for sensor nodes. In *Proceedings of ACM International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2005.

[14] 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.

[15] 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.

[16] P. Kimelman. personal communication, 2006.

[17] E. Kohler, S. Floyd, and M. Handley. Designing dccp: Congestion control without reliability. In *Proceedings of the ACM SIGCOMM*, 2006.

[18] 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.

[19] 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.

[20] P. Levis, N. Patel, D. Culler, and S. Shenker. Trickle: A self-regulating algorithm for code maintenance and propagation in wireless sensor networks. In *Proceedings of the First USENIX/ACM Symposium on Network Systems Design and Implementation (NSDI)*, 2004.

[21] 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.

[22] T. J. Ott, J. H. B. Kemperman, and M. Mathis. The stationary behavior of ideal tcp congestion avoidance. *IEEE/ACM Transactions on Networking*, 1999.

[23] J. Polastre, J. Hui, P. Levis, J. Zhao, D. Culler, S. Shenker, and I. Stoica. A unifying link abstraction for wireless sensor networks. In *Proceedings of the Third ACM Conference On Embedded Networked Sensor Systems (SenSys)*, 2005.

[24] 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.

[25] 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.

[26] J. Regehr, A. Reid, and kirk Webb. Eliminating stack overflow by abstract interpretation. *ACM Transactions on Embedded Computing Systems (TECS)*, 2005.

[27] 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.

[28] 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.

[29] 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.

[30] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, , and W. Hong. A macroscope in the redwoods. In *Proceedings of the Third ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2005.

[31] 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.

[32] N. H. Vaidya, P. Bahl, and S. Gupta. Distributed fair scheduling in a wireless lan. In *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, 2000.

[33] C.-Y. Wan, A. T. Campbell, and L. Krishnamurthy. PSFQ: a reliable transport protocol for wireless sensor networks. In *Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, 2002.

[34] 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.

[35] 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.

[36] 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.