StrawMAN: Making Sudden Traffic Surges Graceful in Low-Power Wireless Networks

Fredrik Österlind, Niklas Wirström, Nicolas Tsiftes,
Niclas Finne, Thiemo Voigt, Adam Dunkels
{fros,niwi,nvt,nfi,thiemo,adam}@sics.se
Swedish Institute of Computer Science

Abstract

To ensure a long lifetime, sensor networks must operate with a low duty cycle, but the duty cycle must be high enough for the network to handle expected peak traffic loads. Being dimensioned for peak traffic rather than regular traffic leads to shorter lifetimes than necessary. We present StrawMAN, a simple mechanism that enables low duty cycle networks to gracefully handle sudden traffic surges. We show that StrawMAN provides a significantly lower power consumption compared to an existing backoff-based mechanism, while being able to scale up to situations with full link-level saturation. As sensor networks get more diverse in both applications and traffic loads, we see StrawMAN as an important addition to the existing set of low-level sensor network mechanisms.

1 Introduction

To meet the requirements of modern deployments, sensor networks must combine long lifetime with reliability and robustness against unexpected events such as sudden traffic surges. To achieve a long lifetime, sensor networks must operate under a low duty cycle, but the duty cycle must cope with expected peak traffic loads [10]. Since sensor network traffic may be bursty due to unexpected events or spatial and temporal correlation of monitored events [6], and that routing fluctuations temporarily may alter the traffic load and pattern, networks must operate with a higher duty cycle than required by its steady state operation.

We present StrawMAN, a simple mechanism that enables a low duty cycle while gracefully tolerating sudden traffic surges. StrawMAN works at the MAC layer where it provides an efficient contention resolution mechanism for simultaneous transmitters. StrawMAN has no overhead in low-traffic situations and its overhead grows linearly with increased traffic loads in high-traffic situations.

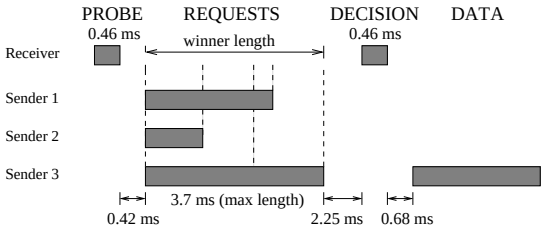

Figure 1. **In StrawMAN, transmitters draw straw for permission to send: channel access is granted to the transmitter that transmits the longest REQUEST signal; "the longest straw wins". After drawing straws, the winning transmitter sends its packet and the process is repeated until all transmitters have transmitted their packets. The timing in the figure are measured on the Tmote Sky.**

Sensor network MAC protocols typically use transmitter-initiated collision avoidance mechanisms with exponential backoff [13]. In such mechanisms, each transmitter probes the radio medium before sending to avoid transmitting a packet when another packet is in the air. If another packet is detected, the transmitter refrains from sending its packet and backs off for a random time. This makes such protocols prone to hidden terminal problems and leads to problems with bursty traffic and sudden traffic surges. Bursty traffic increases the number of collisions, causing transmitters to back off, which incurs additional delays and reduced throughput. The Request To Send / Clear To Send (RTS/CTS) mechanism can be used to solve the hidden terminal problem. Data transmissions are preceded by a transmission request message (RTS). If the medium is available and the transmission is granted (CTS), any potentially interfering neighbor refrains from accessing the medium for the duration of the data transmission. RTS/CTS in sensor networks has been shown to have high overhead. [10]

To resolve contention between multiple transmitters, StrawMAN uses a distributed selection mechanism based on the principle of drawing straws. When a collision is detected, the simultaneous transmitters draw straw for the permission to send. Each transmitter independently draws a random number from a fixed interval. To communicate the length of their straws, the transmitters simultaneously send an RTS signal, whose length is proportional to the length of their randomly drawn number. The receiver measures the length of the transmitted RTS signals, and sends a CTS a fixed

time after it has detected the end of the longest transmission. The transmitter that drew the longest straw then can send its packet. The process is repeated until all transmitters have sent their packets. Figure 1 shows the mechanism.

We have implemented StrawMAN with the receiver-initiated Low-Power Probing radio duty cycling protocol [7, 13] in Contiki. We present results from experiments on Tmote Sky hardware that demonstrate the feasibility of StrawMAN, as well as simulation and experimental results that show that StrawMAN scales with increased traffic load without requiring a higher duty cycle configuration.

2 Related Work

The literature contains several mechanisms for handling sudden traffic surges in sensor networks. Existing mechanisms typically work at many layers, either requiring information sharing between layers or negatively impacting the performance of upper layers. Examples include ESRT [12], which requires interaction between the application layer and the MAC layer; and the adaptive MAC layer by Woo and Culler [15], which uses random backoff to avoid surges, thus causing severe performance reduction for the application layer. By contrast, StrawMAN does not require application awareness or cross-layer interactions, and traffic surges have a low impact on the application layer performance.

Operating at the MAC-layer, StrawMAN shares parts of its heritage with traditional MAC layer protocols. Carrier Sense Multiple Access (CMSA) is a MAC-layer mechanism that is often used in sensornets due to its simplicity and low overhead in comparison with Time Division Multiple Access (TMDA) [16]. CSMA uses per-packet collision detection and contention resolution to resolve the conflict between simultaneous transmitters. To avoid the hidden terminal problem, Request-To-Send/Clear-To-Send (RTS/CTS) protocols are typically used, but have a considerable overhead [2, 10]. StrawMAN uses an RTS/CTS-like mechanism to resolve conflicts, but allows multiple simultaneous transmitters. We are not aware of other RTS/CTS mechanisms that support multiple simultaneous transmitters.

Inspired by recent work that exploits low-level radio effects [3, 4], StrawMAN extracts information from the collision of multiple radio transmissions by measuring the duration of the longest transmission. Extracting information from collisions has been used in other MAC-layer mechanisms. Demirbas et al. use radio collisions to implement Pollcast and Countcast, network primitives that enable sensornet voting among immediate neighbors [3]. Dutta et al. show that collisions of identical 802.15.4 packets do not necessarily lead to data corruption, and implement an anycast-like primitive called Backcast [4]. Whitehouse et al. [14] present a mechanism for recovering partial information from semi-collided packets. Unlike these protocols, StrawMAN does not attempt to extract any information from the packets that are involved in the collision—only the length of the longest packet is used. StrawMAN is therefore not bound to any specific radio modulation or encoding technique.

StrawMAN's arbitration mechanism is similar to that of dominance protocols such as CAN, but uses dynamic priorities rather than static. Dominance protocols have previously

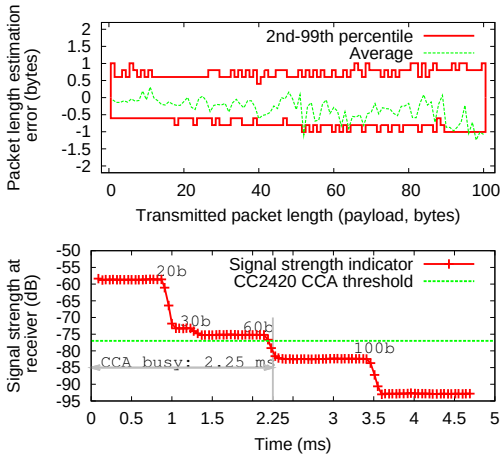

Figure 2. Packet length estimation using CCA performs well: 97% of short-range packet length estimations are within 2 bytes (upper graph); the 60-byte packet is correctly measured and is not affected by simultaneous transmissions (lower graph).

been evaluated in the wireless domain [9, 11]. WiDoM [9] and BitMAC [11] both require the underlying physical medium to be based on On-Off-Keying (OOK) modulation. By contrast, StrawMAN does not depend on any specific underlying modulation mechanis, and handles the hidden terminal problem by its inherent RTS/CTS mechanism.

Our StrawMAN implementation samples from a uniform distribution. In the context of random backoff, Jamieson et al. [5] concludes that the optimal distribution is a function of the number of contenders, and instead proposes a geometrical distribution. We consider this work a good starting point for determining the optimal distribution for StrawMAN.

3 Packet Length Estimation

To evaluate the feasibility of using RTS signal length measurements as a foundation of StrawMAN, we perform two initial experiments on the Tmote Sky platform and its CC2420 radio transceiver. The CC2420 samples the radio signal strength at 62.5 kHz, averages readings over eight samples, and makes the output available under the name Received Signal Strength Indicator (RSSI). If the RSSI value is over a configurable threshold, the CC2420 sets one of its output pins, the so-called Clear Channel Assessment (CCA) pin, to high. Both the RSSI and the CCA are available even when the CC2420 cannot make sense of the radio traffic, e.g. during a collision of two radio packets.

We first measure how well we are able to measure the length of a transmission using only the information provided by the RSSI and the CCA, without receiving the packet. This indicates how well we can use multiple simultaneous transmissions as a way to transmit RTS signals. The experimental setup consists of two nodes placed close to each other. One node transmits packets at random intervals, and the other continuously samples the CC2420 transceiver's Clear Channel Assessment (CCA) pin. The packet payload is varied between 1 and 100 bytes. 350 test runs are performed for each payload. The results of the first experiment are shown in the upper graph of Figure 2. The graph shows the 2nd and 99th

percentiles of the estimation errors; 97% of the estimations have an error of less than 1 byte.

We now turn to evaluating the effect of transmitter distance on the packet length estimation. In the second experiment five nodes are placed in line, ten meters apart. One node is the receiver, four are transmitters. The receiver periodically sends a command packet to the transmitters, commanding them to immediately send a packet to the receiver. Response packets are sent with a lower output power than the command packet. The receiver samples the RSSI and the CCA for a total of 5 ms. The payload sizes of reply messages differ between the four nodes, and are proportional to the distance from the probing node: 20 bytes, 30 bytes, 60 bytes, and 100 bytes, respectively. The lower graph of Figure 2 shows the results, demonstrating that the CCA estimate can be used to correctly estimate packet lengths even with concurrent transmissions.

4 The StrawMAN Mechanism

StrawMAN is conceptually similar to both RTS/CTS and random backoff. Like RTS/CTS, StrawMAN uses request messages, and explicitly notifies all contenders who won access and is allowed to transmit its data packet. The two major differences between StrawMAN and RTS/CTS are that StrawMAN allows several nodes to simultaneously request access to the channel, and therefore requires the receiver to initiate the contention round.

Random backoff and StrawMAN both randomly schedule channel access among contenders. Both mechanisms let contenders pick a random number each contention round, that effectively determines who wins channel access. Unlike random backoff algorithms, StrawMAN nodes are actively transmitting a REQUEST message with a length corresponding to the random pick. Since each contender transmits at least once every contention round, a StrawMAN node knows that it has no already contending immediate neighbors if the radio medium is free for a single contention round.

StrawMAN separates data packet transmissions from channel access arbitration, which enables shorter arbitration phases. In comparison to random backoff algorithms, StrawMAN can use a significantly higher resolution – on the order of transmitted bytes instead of packets – and hence shorter arbitration rounds. Although StrawMAN requires two additional transmissions (REQUESTs and DECISION) before transmitting the data packet, and only allows a single data packet transmission per contention round, the average contention round durations are significantly shorter then with Binary Exponential Backoff as shown in Section 5.1.

Each StrawMAN contention period consists of four consecutive messages: PROBE, REQUEST, DECISION, and DATA (see Figure 1). A PROBE is sent by the receiver to notify neighbors that he is ready to receive data. All neighbors that have data for the receiver contend for the channel by an immediate REQUEST. Multiple REQUESTs may thus collide at the receiver. Each REQUEST message consists of a small header and a random length payload. In this work we sample from a uniform distribution – determining the optimal distribution is regarded future work. The receiver samples the channel for activity during the REQUESTs, and estimates the payload length of the longest REQUEST. The receiver then sends a DECISION message containing the length estimate. The contender whose REQUEST length matches the one specified in the DECISION is granted channel access, and sends its DATA message. Another contention round is initiated when the DATA has been received, or after a timeout. The PROBE message has dual purpose: it also acknowledges the last received DATA packet. Note that if two contenders pick the same random length and hence are both granted channel access leading to a DATA collision, the timeout will trigger another contention period, and both data packets will be retransmitted due to the lack of acknowledgement.

5 Evaluation

We evaluate three aspects of StrawMAN: the performance implications of the discrete length of the RTS signals – a limitation of our hardware platform; StrawMAN's performance in face of the funneling effect; and how well it scales to traffic loads that incur full link layer saturation.

We have implemented StrawMAN in Contiki 2.4 by integrating it with Contiki's Low Power Probing (LPP) MAC protocol. StrawMAN LPP is configured to trigger *only when a packet collision is detected*. The StrawMAN LPP is not fully optimized; it does not preload data packets [8], and it does not use CC2420's hardware acknowledgments. We expect the performance to increase with such optimizations.

We study StrawMAN both in simulation and in experiments with real hardware. For simulations, we use the COOJA/MSPSim simulator, which allows both idealized high-level simulation of Java-based sensor network code as well as cycle-accurate Tmote Sky simulation with bit-accurate simulation of the CC2420 radio chip.

5.1 Request Length Resolution

Before evaluating the performance of StrawMAN, we must first quantify the effect of the discrete length of the StrawMAN RTS signals. Our discussion has so far used ideal, continuous-length signals, but an implementation must by necessity use discrete-length signals. The Contiki StrawMAN implementation uses a fixed resolution: each request transmission has a random payload from 0 to 100 bytes in 2-byte intervals. To measure the effect of discrete-length signals, we perform a high-level simulation.

We compare StrawMAN with the scalable Binary Exponential Backoff (BEB) mechanism. We use simulation to avoid the limitations of sensornets: the simulated environment has infinite buffer size, no channel-detection delay, and a perfect radio environment with symmetric links. Instead of using the Contiki StrawMAN implementation in this experiment, we use a high-level implementation in Java intended to only exercise the scalability.

We measure completion time: the time for all contenders to successfully send their data packets to the receiver. In the idealized environment, data packets and StrawMAN request packets both last for exactly 1 time unit. To remove effects unrelated to scalability, data probe packets, decision packets, and acknowledgments have zero duration.

The network consists of a single receiver and a varying amount of contenders, each with a single data packet. The contenders are positioned randomly at a constant distance

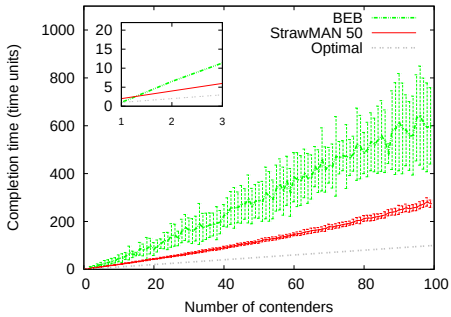

Figure 3. StrawMAN with resolution 50 has a significantly lower completion time than Binary Exponential Backoff (BEB).

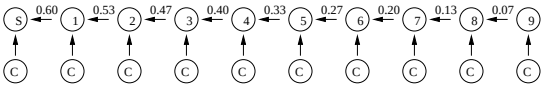

Figure 4. The collection network has one sink (*S*), and 19 collector nodes that generate data at random in the interval 0 - 60 seconds. The average rate of data packets per second over each link are shown in the figure, excluding protocol overhead and retransmissions.

from the receiver, thus forming a circle. The radio transmission range is equal to the circle radius: each contender can communicate with the receiver, but is outside the range of most of the other contenders. This network setup is chosen to spur the hidden terminal problem.

Figure 3 shows the resulting completion time. We expect BEB to scale linearly with the number of contenders, whereas the limited resolution of StrawMAN should limit contender scalability. For less than 100 simultaneous contenders, however, StrawMAN has a significantly lower completion time than BEB. Further, the completion time of BEB has a significantly higher variance than StrawMAN.

5.2 Increasing Traffic Concentration: The Funneling Effect

To evaluate the performance of StrawMAN with increasing traffic concentration and congestion, we devise an experiment that exhibits a strong funneling effect [1]. Sensornets experience funneling regions at near-sink nodes that forward traffic from a large portion of the network. Since radio collisions occur more frequently in funneling regions, the need for channel access arbitration is crucial.

StrawMAN solves contention in linear time with less overhead than Binary Exponential Backoff (BEB) (Section 5.1). We evaluate StrawMAN's dependency between channel check rate and data load, and compare with BEB.

We simulate the network of Tmote Sky nodes shown in Figure 4. The purpose of the network setup is to show how StrawMAN's radio duty cycle and data collection latency is affected by the data load and the configured channel check rate. The backbone nodes (1-9) both generate and forward data from the collector nodes (denoted *C*) towards the sink node (*S*.) We run experiments with StrawMAN LPP, and measure the resulting duty cycle and the average per-transmitter data collection latency. We include simulation results with BEB for comparison.

Figure 5 shows the radio duty cycle of the backbone

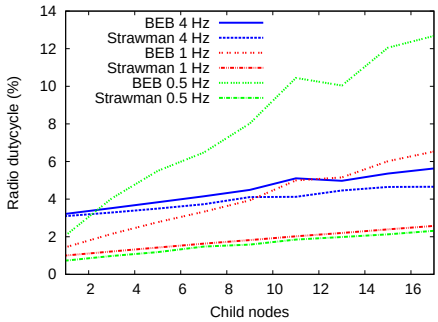

Figure 5. The radio duty cycle for StrawMAN is lower for the same channel check rate.

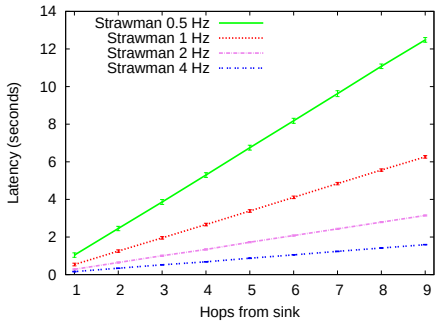

Figure 6. Data delivery latency.

nodes, ordered by number of child nodes. The duty cycle increases semi-linearly with increased traffic for both StrawMAN and BEB. All three StrawMAN lines appear parallel to each other, which indicates that the network data load and the configured channel check rate are independent from each other. By contrast, BEB's duty cycle increases more with lower channel check rates. The BEB graphs are included only to highlight the behavior of StrawMAN: in a network that uses BEB with corresponding network data load, the configured channel check rate should be considerably higher.

Figure 6 shows the data packet latency for StrawMAN with different channel check rates. The figures show that latency of StrawMAN's data packets depend on the distance from the sink and the configured channel check rate. The variance is low, indicating that packet latency is not affected by increased data load

5.3 Pushing the Envelope: Link Saturation

To understand the performance limits of StrawMAN, we perform an experiment where multiple transmitters transmit at full speed to a single receiver. The purpose of the experiment is to understand how StrawMAN handles the extreme end of a sudden traffic surge: the case where the traffic amount exceeds the available link bitrate. Ideally, we expect StrawMAN to provide each contending sender with its fair share of the medium and to keep the link saturated.

We run our experiment on a small single-hop network with 5 Tmote Sky nodes running StrawMAN LPP. Four nodes are configured to *always* contend for permission to transmit data to the fifth node. The objective of this experimental setup is both to stress-test the implementation on real

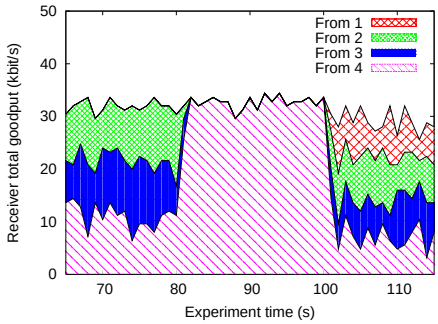

Figure 7. The total application goodput at the sink node is not significantly affected by the number of channel contenders, nor by adding or removing contenders.

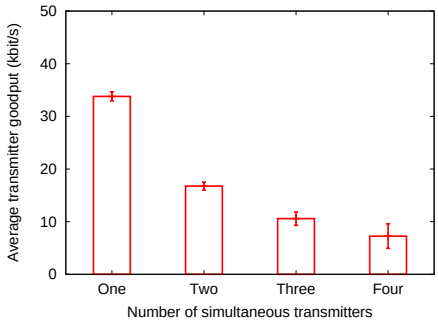

Figure 8. The goodput scales well with number of channel contenders. The transmitter goodput with a single contender is 33.8 kbit/s. With four contenders, the goodput per transmitter is 7.3 kbit/s (29.0 kbit/s at the sink).

hardware, and to evaluate the fairness between contenders.

The number of active contenders during the experiment varies in intervals of 20 seconds. Each data packet has a payload of 100 bytes. We measure the goodput, i.e., the rate at which application data arrives at the receiver.

Figure 7 shows an excerpt of the application goodput during the experiment. Up until 80 seconds, three nodes are active. Between time 80 and 100, we stop all senders except node 4. All four nodes are activated at time 100 seconds.

Figure 8 shows the average transmitter goodput in the experiment. The overall goodput is highest with only one active transmitter: 33.8 kbit/s. The goodput per transmitter when having 4 transmitters is 7.3 kbit/s, totalling 29.0 kbit/s. The amount of invalid packet length estimations increased with the number of nodes, causing the throughput to decrease and the variance to increase, which we believe accounts for the decrease in overall goodput. This goodput variance demonstrates that StrawMAN is fair to the four contenders.

6 Conclusions

To meet both lifetime and reliability requirements, sensor networks must be able to cope with both low-intensity traffic and spurious traffic surges. We present StrawMAN, a low-power mechanism that allows network operation with a low duty cycle, yet gracefully handles higher traffic loads. We have implemented StrawMAN with the Low-Power Probing radio duty cycling protocol in Contiki and demonstrate through simulation and experiments that StrawMAN provides a lower duty cycle than other collision backoff mechanisms, while scaling up to full link layer saturation conditions. Being able to combine low power operation with reliability and robustness against traffic surges, we believe StrawMAN will be a valuable addition to the existing toolbox of low-level sensor network mechanisms.

7 Acknowledgements

This work was partially supported by VINNOVA, the Swedish Agency for Innovation Systems, by the Swedish Foundation for Strategic Research and by CONET, the Cooperating Objects Network of Excellence.

8 References

[1] G.-S. Ahn, E. Miluzzo, A. T. Campbell, S. G. Hong, and F. Cuomo. Funneling-mac: A localized, sink oriented mac for boosting fidelity in sensor networks. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Boulder, Colorado, USA, Nov. 2006.

[2] V. Bharghavan, A. Demers, S. Schenker, and L. Zhang. Macaw: a media access protocol for wireless lan's. In *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM)*, United Kingdom, 1994.

[3] M. Demirbas, O. Soysal, and M. Hussain. Singlehop collaborative feedback primitive for wireless sensor networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, 2008.

[4] P. Dutta, R. Musaloiu-E., I. Stoica, and A. Terzis. Wireless ack collisions not considered harmful. In *Proceedings of the Workshop on Hot Topics in Networks (ACM HotNets)*, 2008.

[5] K. Jamieson, H. Balakrishnan, and Y. C. Tay. Sift: a MAC Protocol for Event-Driven Wireless Sensor Networks. In *Proceedings of the European Conference on Wireless Sensor Networks (EWSN)*, Zurich, Switzerland, Feb. 2006.

[6] A. Jindal and K. Psounis. Modeling spatially correlated data in sensor networks. In *ACM Transactions of Sensor Networks*, Nov. 2006.

[7] R. Musaloiu-E., C.-J. M. Liang, and A. Terzis. Koala: Ultra-Low Power Data Retrieval in Wireless Sensor Networks. In *Proceedings of the International Conference on Information Processing in Sensor Networks (ACM/IEEE IPSN)*, St. Louis, Missouri, USA, 2008.

[8] F. Österlind and A. Dunkels. Approaching the maximum 802.15.4 multi-hop throughput. In *Proceedings of the Workshop on Hot Topics in Embedded Networked Sensor Systems (HotEmnets)*, June 2008.

[9] N. Pereira, B. Andersson, and E. Tovar. WiDom: A dominance protocol for wireless medium access. *IEEE Transactions on Industrial Informatics*, 3(2):120, 2007.

[10] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, pages 95–107, Baltimore, MD, USA, 2004. ACM Press.

[11] M. Ringwald and K. Römer. BitMAC: A Deterministic, Collision-Free, and Robust MAC Protocol for Sensor Networks. In *Proceedings of the European Conference on Wireless Sensor Networks (EWSN)*, Istanbul, Turkey, Jan. 2005.

[12] Y. Sankarasubramaniam, O. Akan, and I. Akyildiz. ESRT : Event-to-Sink Reliable Transport in Wireless Sensor Networks. In *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking and computing (MobiHOC 2003)*, 2003.

[13] Y. Sun, O. Gurewitz, and D. Johnson. RI-MAC: A Receiver-Initiated Asynchronous Duty Cycle MAC Protocol for Dynamic Traffic Loads in Wireless Sensor Networks. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Raleigh, NC, USA, 2008.

[14] K. Whitehouse, A. Woo, F. Jiang, J. Polastre, and D. Culler. Exploiting The Capture Effect For Collision Detection And Recovery. In *Proceedings of the IEEE Workshop on Embedded Networked Sensor Systems (IEEE Emnets)*, Sydney, Australia, May 2005.

[15] A. Woo and D. Culler. A transmission control scheme for media access in sensor networks. In *Proceedings of the International Conference on Mobile Computing and Networking (ACM MobiCom)*, Rome, Italy, 2001.

[16] W. Ye, J. Heidemann, and D. Estrin. An Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, New York, NY, USA, June 2002.