Energy-Efficient Congestion Detection and Avoidance in Sensor Networks

CHIEH-YIH WAN
Intel Research
and
SHANE B. EISENMAN
Columbia University
and
ANDREW T. CAMPBELL
Dartmouth College

Event-driven sensor networks operate under an idle or light load and then suddenly become active in response to a detected or monitored event. The transport of event impulses is likely to lead to varying degrees of congestion in the network depending on the distribution and rate of packet sources in the network. It is during these periods of event impulses that the likelihood of congestion is greatest and the information in transit of most importance to users. To address this challenge we propose an energy efficient congestion control scheme for sensor networks called *CODA (COngestion Detection and Avoidance)* that comprises three mechanisms: (i) receiver-based congestion detection; (ii) open loop hop-by-hop backpressure; and (iii) closed loop multi-source regulation. We present the detailed design, implementation, and evaluation of CODA using simulation and experimentation. We define two important performance metrics (i.e., energy tax and fidelity penalty) to evaluate the impact of CODA on the performance of sensing applications. We discuss the performance benefits and practical engineering challenges of implementing CODA in an experimental sensor network testbed based on Berkeley motes using CSMA. Simulation results indicate that CODA significantly improves the performance of data dissemination applications such as directed diffusion by mitigating hotspots, and reducing the energy tax and fidelity penalty on sensing applications. We also demonstrate that CODA is capable of responding to a number of congestion scenarios that we believe will be prevalent as the deployment of these networks accelerates.

Categories and Subject Descriptors: C.2.1 [**COMPUTER-COMMUNICATION NETWORKS**]: Network Architecture and Design—*Wireless Communications*

General Terms: Design, Experimentation, Performance

Additional Key Words and Phrases: Sensor networks, Congestion control

1. INTRODUCTION

Sensor networks come in a wide variety of forms, covering different geographical areas, being sparsely or densely deployed, using devices with a variety of energy constraints, and implementing an assortment of sensing applications. One application driving the development of sensor networks is the reporting of conditions within a region where the environment abruptly changes due to observed events, such as target detection, earthquakes, floods, or fires, and in habitat monitoring. Sensor networks may typically operate under light load, but can suddenly become active in response to a detected event. Some applications may only generate light traffic from small regions of the sensor network (e.g., target

Author's address: C.-Y. Wan, Intel Research, Hillsboro, OR, USA; email: chieh-yih.wan@intel.com
This paper extends work published in the proceedings of ACM SenSys 2003.

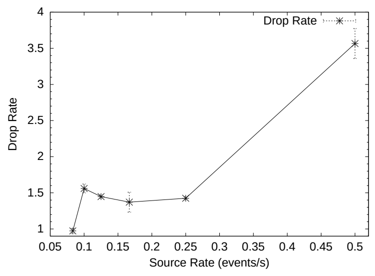


Fig. 1. Total number of packets dropped by the sensor network per data event packet delivered at the sink as a function of the source rate.

detection) while others (e.g., fires, earthquakes detection) may generate large waves of impulses, potentially across large sections of the sensing area. Although a sensor network may spend only a small fraction of time dealing with impulses, it is during this time that the information it delivers is of greatest importance. Sensor networks exhibit a unique *funneling effect*, a traffic pattern where events are generated *en masse* and then must be quickly moved toward a relatively small number of physical sink points that are attached to the regular communication infrastructure. This leads to a number of significant challenges including increased transit traffic intensity, congestion, and packet loss (and therefore energy and bandwidth waste) at nodes closer to the sink, disrupting the performance (i.e., fidelity) of the sensing application.

The transport of event impulses is likely to lead to varying degrees of congestion in sensor networks. Figure 1 shows the impact of congestion on data dissemination in an experimental sensor network testbed running Surge, a commonly used application included in the TinyOS distribution [TinyOS 2007]. Our testbed comprises 48 Mica2 motes arranged in a 6x8 grid. Node spacing and transmission power are set such that one-hop neighbors achieve $>$ 80% delivery, while two-hop neighbors achieve $<$ 20% delivery. In this way, a fairly strict and dense multi-hop radio environment is constructed for experimentation. Surge periodically reports ADC readings to the sink at a rate that is programmable over-the-air using a control message. The Surge application employs the services of the MultiHopRouter [Woo et al. 2003] component to set up and maintain a forwarding tree, based on packet-time granularity link quality estimation. Figure 1 illustrates that as the source rate increases beyond a certain network capacity threshold (0.25-0.5 events/s in this network), congestion occurs more frequently and the total number of packets dropped per received data packet at the sink becomes severe. The plot shows that even with low to moderate source event rates there is a large drop rate observed across the sensor network. For example, with a source event rate of 0.1 events/s in the network 1.5 packets are dropped across the sensor field for every data event packet received at the sink. The drop rates shown in Figure 1 represent not only significant packet losses in the sensor network, but more importantly, energy wasted by the sensing application.

In traditional computer networks, throughput and delay are two important performance metrics that impact the users' experience. Therefore, the objective function for control mechanisms adopted to control the traffic is often defined as maximizing the ratio of throughput to delay [Ramakrishnan and Jain 1995], i.e., the power. However, in the context of sensor networks, because of its limited resources and application specific nature (some applications may only generate light traffic from small regions of the sensor network (e.g., target detection) while others (e.g., fires, earthquakes detection) may generate large waves of impulses, potentially across the whole sensing area), we observe that maximizing this ratio does not necessarily result in the optimal performance. Rather, the objective of sensor networks is to maximize the operational lifetime while delivering acceptable data fidelity to the applications.

In response to this, future congestion control mechanisms for sensor networks must be capable of balancing the offered load, while attempting to maintain acceptable fidelity (e.g., rate of events) of the delivered signal at the sink during periods of transient and more persistent congestion. A number of distinct congestion scenarios are likely to arise. First, densely deployed sensors generating impulse data events will create persistent hotspots proportional to the impulse rate beginning at a location very close to the sources (e.g., within one or two hops). In this scenario, localized, fast time scale mechanisms capable of providing backpressure from the points of congestion back to the sources could be effective. Second, sparsely deployed sensors generating low data rate events will create transient hotspots potentially anywhere in the sensor field but likely farther from the sources, toward the sink. In this case, fast time scale resolution of localized hotspots using a combination of localized backpressure (between nodes identified in a hotspot region) and rate limiting techniques could be more effective. Because of the transient nature of congestion, source nodes may not be involved in the backpressure. Third, sparsely deployed sensors generating high data-rate events will create both transient and persistent hotspots distributed throughout the sensor field. In this final scenario, a combination of fast time scale actions to resolve localized transient hotspots, and closed loop rate regulation of all sources that contribute toward creating persistent congestion could be effective.

—*Congestion detection*. Accurate and efficient congestion detection plays an important role in the congestion control of wireless networks. CODA uses a combination of the present and past channel loading conditions, and the current buffer occupancy, to infer accurate detection of congestion at each receiver with low cost. Sensor networks must know the state of the channel since the transmission medium is shared and may be congested with traffic between other nodes in the neighborhood. Listening to the channel to measure local loading incurs high energy costs, if performed all the time. Therefore, CODA uses a sampling scheme that activates local channel monitoring at the appropriate time to minimize cost while forming an accurate estimate. Once congestion is detected, nodes signal their upstream neighbors via a backpressure mechanism that is discussed next.

—*Open loop, hop-by-hop backpressure*. In CODA a node broadcasts backpressure messages as long as it detects congestion. Backpressure signals are propagated upstream toward the source. In the case of impulse data events in dense networks it is very likely that backpressure will propagate directly to the sources. Nodes that receive backpressure signals can throttle their sending rates based on the local congestion policy (e.g., silence for a random time or AIMD, etc.). When an upstream node (toward the source) receives a backpressure message it decides whether or not to further propagate the backpressure upstream, based on its own local measured network conditions.

—*Closed loop, multi-source regulation*. In CODA, closed loop regulation operates over a slower time scale and is capable of asserting congestion control over multiple sources from a single sink in the event of persistent congestion. When a source event rate is less than some fraction of the maximum theoretical throughput of the channel, the source regulates itself. When this value is exceeded, however, a source is more likely to contribute to congestion, and therefore, closed loop congestion control is triggered. The source only enters sink regulation if this threshold is exceeded. At this point a source requires constant, slow time-scale feedback (e.g., ACK) from the sink to maintain its rate. The reception of ACKs at sources serve as a self-clocking mechanism allowing sources to maintain their current event rates. In contrast, failure to receive ACKs forces a source to reduce its own rate. Once a source has determined congestion has passed it takes itself out of sink regulation under its own direction.

The paper is organized as follows. Section 2 discusses a number of important design considerations for mitigating congestion in sensor networks including MAC and congestion detection issues. Section 3 details CODA's backpressure and rate regulation mechanisms. Following this, an implementation of CODA is evaluated in an experimental sensor testbed in Section 4. We define three important performance metrics (i.e., energy tax, fidelity penalty, and power) to evaluate the impact of CODA on the performance of sensing applications. Because CODA is designed to interwork with existing data dissemination schemes, we also evaluate it using one well-known dissemination mechanism. Section 5 presents our performance evaluation of CODA working with directed diffusion [Intanagonwiwat et al. 2000] using the ns-2 simulator. Section 6 presents the related work. Finally, some concluding remarks and future work are discussed in Section 7. This paper represents an extended version of a paper [Wan et al. 2003] presented in ACM SenSys 2003.

2. DESIGN CONSIDERATIONS

In what follows, we discuss the technical considerations that underpin the design of CODA while the detailed design is presented in Section 3. We discuss the MAC and congestion detection considerations with a focus toward CSMA/contention-based schemes, given that TDMA and other schedule-based schemes (e.g., [Clare et al. 1999][Rajendran et al. 2003]) can control and schedule traffic flows in the network to provide collision-free communication. However, we note that congestion can still occur in scheduled access networks when the incoming traffic exceeds the node capacity and the queue overflows. Further, the new objective function for congestion control (discussed in the previous section) demands new feedback control mechanisms even for TDMA/schedule-based networks. These new control mechanisms (Sections 3.1 and 3.2) can be used seamlessly on both contention-based and schedule-based networks.

Medium access control plays a significant role in the performance of managing impulses of data in a wireless shared medium, including the detection of congestion. A growing number of sensor networks use CSMA or variants for the medium access control. For example, the widely used Berkeley motes [Hill et al. 2000] use a simple CSMA MAC as part of the TinyOS [TinyOS 2007] platform. In [Ye et al. 2002] the authors proposed a modified version of CSMA called S-MAC, which combines TDMA scheduling with CSMA's contention-based medium access, without a strict requirement for time synchronization. S-MAC uses virtual carrier sense to avoid hidden terminal problems, allowing nodes other than the sender and receiver to enter sleep mode (during the NAV after the RTS/CTS exchange), thus saving energy. A collision-minimizing CSMA MAC is proposed in [Tay et al. 2004] that is optimized for event-driven sensor networks. The authors propose to utilize a non-uniform probability distribution for nodes to randomly select contention slots such that collisions between contending stations are minimized.

2.1 CSMA Considerations

2.1.1 *Throughput Issues.* The theoretical maximum channel utilization for the CSMA scheme is approximately [Bertsekas and Gallagher 1991]:

$$S_{max} \approx \frac{1}{(1+2\sqrt{\beta})} (for\ \beta = \frac{\tau C}{L} \ll 1), \qquad (1)$$

The performance of CSMA is highly dependent on the value of β, which is a measure of radio propagation delay and channel idle detection delay. τ is the sum of both radio propagation delay and channel idle detection delay in seconds, C is the raw channel bit rate and L is the expected number of bits in a data packet. If nodes can detect idle periods quickly, in other words have a very small β value, then CSMA can offer very good channel utilization regardless of the offered load.

Equation (1) gives the channel capacity of CSMA within one hop. In [Li et al. 2001] the authors show that an ideal ad hoc multi-hop forwarding chain should be able to achieve 25% of the throughput that a single-hop transmission can achieve. This observation has important implications in the design of our congestion detection and closed loop regulation mechanisms, as discussed in Section 2.2 and Section 3.2, respectively.

2.1.2 *Hidden Terminals.* CSMA suffers from the well-known hidden terminal problem in multi-hop environments. IEEE 802.11 utilizes virtual carrier sense (VC), namely an RTS/CTS exchange, to eliminate hidden terminals. In order to reduce the signaling overhead incurred by adding VC, IEEE 802.11 does not exchange RTS/CTS for small packets. In sensor networks, packets are usually small in nature (i.e., on the order of few tens of bytes) because of the low duty cycle requirement and traffic characteristics [Pottie and Kaiser 2000]. Therefore, the signaling cost is high if the RTS/CTS exchange is used for every message. Furthermore, sensor nodes have a limited energy budget making the energy cost of doing this prohibitively high.

Usually, nodes other than event source nodes and the forwarding nodes will be silent most of the time. Therefore, loss due to hidden terminals is less likely to occur when the workload of the network is low. In [Woo and Culler 2001], the authors show that in general, when nodes use a randomized pre-transmit delay coupled with appropriate jitter in sending/forwarding packets, the probability of hidden terminals is low even in dense networks. In S-MAC [Ye et al. 2002], an RTS/CTS exchange is used in an aggregated manner (i.e., not for every single packet) to reduce the energy cost.

In the context of sensor networks, the VC scheme is costly and mostly unnecessary during normal operations[1]. There is a need, however, to devise a scheme that can work satisfactorily with or without the VC for collision avoidance, that incurs low cost or no cost during normal operations, and yet is responsive enough to quickly resolve congestion. In Section 3.1, we discuss such a scheme.

2.1.3 *Link-layer ARQ.* In the IEEE 802.11 MAC, a packet will be kept in the sending buffer until an ACK is received or the number of retransmissions exceeds a certain thresh-

[1]A user may omit the VC for data packets but retain it for critical signaling messages (e.g., routing protocol control packets) in order to reduce overhead.

old. This mechanism increases the link reliability at the expense of energy and buffer space. However, both of these resources are scarce in sensor nodes where support for reliability may not always be necessary under normal operations (i.e., due to the application-specific nature of sensor networks not all data packets require strict reliability[2]). Today different sensor platforms utilize different radio technologies; some radios support low-overhead synchronous ACK [Levis et al. 2004] (e.g., the RFM radio used in Mica) and some radios include built-in link-layer ACK supporting higher data rates up to 250 Kbps (e.g., the IEEE 802.15.4 radio used in Telos [TinyOS 2007]), while in others supporting ACK could be costly (e.g., the Chipcon radio used in Mica2 [Levis et al. 2004]) in terms of energy and bandwidth consumption, since for these the ACK is sent as a full packet after a non-negligible rx-to-tx switch time, which requires re-contention for the channel.

We believe there is a need for separation between reliability and congestion control in the design of sensor networks protocols. The use of VC and link-layer ARQ as a reliable means of communication are essential for critical information exchange (e.g., routing signaling), but they are not necessarily relevant during congestion. In sensor networks, energy expenditure is more important than occasional data loss because of the natural redundancy inherent in disseminated sensor data. The main objective function is therefore to minimize energy expenditure. This is in contrast to TCP where the lost data is always recovered. In our design, congestion control elements do not explicitly look at loss (unlike TCP), allowing CODA to decouple reliability from congestion control mechanisms. CODA is capable of working with or without reliability elements, such as link-layer ARQ, providing flexibility in support of applications' needs and the radio technology used. CODA is not proposed as a TCP replacement but as a different possibility for transport in wireless sensor networks.

2.2 Congestion Detection

Accurate and efficient congestion detection plays an important role in congestion control of sensor networks. There is a need for new congestion detection techniques that incur low cost in terms of energy and computation complexity. Several techniques are possible.

2.2.1 *Buffer Queue Length.* Queue management is often used in traditional data networks for congestion detection. However, without link-layer ACK (some applications might not require this and hence would omit it to save the overhead, as discussed above), buffer occupancy or queue length cannot be used as a reliable indication of congestion. To illustrate this, we perform an ns-2 simulation of the simple IEEE 802.11 wireless 5-node network shown in Figure 2. In the simulation, nodes 1 and 4 each start sending (1 second apart in simulation time) CBR traffic (64 byte data packets) that consumes 50% of the channel capacity through node 2 to node 3 and 5, respectively. One of the sources stops sending data after 10 seconds. We run two simulation trials, one with the VC enabled (including link ARQ), the other with it disabled and no link ARQ. Nodes have queues of length ten.

Figure 3 shows the time series traces for both channel loading and buffer occupancy as well as the packet delivery ratio measured at the intermediate node 2. It is clear from the plot that the channel loading almost immediately rises to 90% during the time both

[2]For example, applications that generate periodic workload can often reasonably assume that subsequent reports will supersede any lost data.

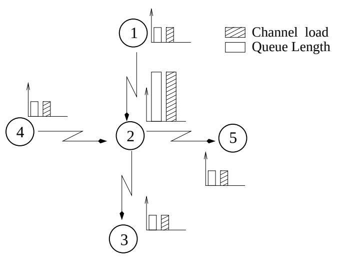


Fig. 2. A simple IEEE 802.11 wireless network of 5 nodes illustrates receiver-based congestion detection.

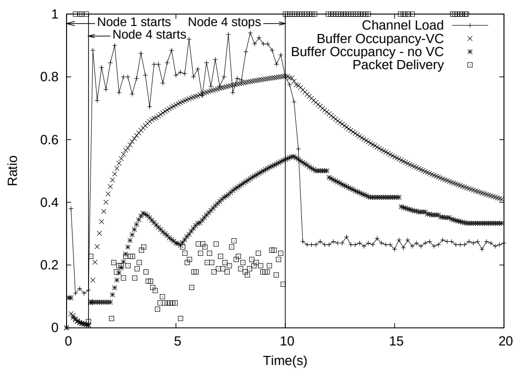


Fig. 3. Channel load and buffer occupancy time series traces with and without virtual carrier sense (VC) + LL-ACK, and packet delivery trace with VC.

sources are on. Congestion occurs and the packet delivery ratio drops from 100% to around 20% during this period. Note that the buffer occupancy grows at a slower rate during this congestion period, particularly in the trace corresponding to the simulation where the VC is disabled. The buffer occupancy (without link ACK) even drops at around 5 seconds into the simulation, which provides false information about the congestion state. This is because without the link-layer ACK, the clearing of the queue at the transmitter does not mean that congestion is alleviated since packets that leave the queue might fail to reach the next hop as a result of collisions. Note that CSMA does not guarantee collision-free transmissions among neighboring nodes because of the detection delay [Bertsekas and Gallagher 1991].

This simple simulation shows that the buffer occupancy alone does not provide an accurate and timely indication of congestion even when the link ARQ is enabled, except in the extreme case when the queue is empty or about to overflow. The first case indicates good traffic conditions and the latter one signals serious congestion. As shown in the figure, the queue takes a much longer time to grow beyond a high watermark level (e.g., 0.8) that signifies congestion compared to the channel load. We argue that this bimodal effect and detection latency is not responsive enough and too coarse to provide accurate, timely and efficient congestion control, especially in the case of event-driven sensor networks where short-lived hotspots are likely to occur across different time-scales. Therefore, we propose augmenting buffer monitoring with *channel load measurement* for fast and reliable congestion detection in sensor networks.

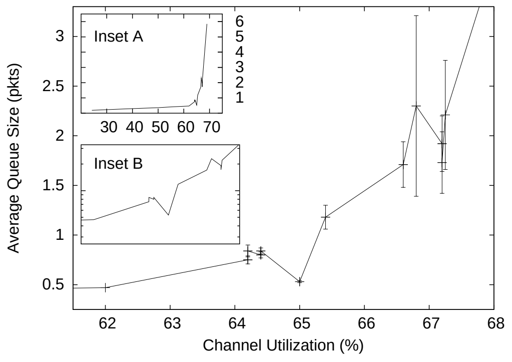


Fig. 4. Queuing performance of a real sensor network of Mica motes.

2.2.2 *Channel Loading.* In CSMA networks, it is straightforward for sensors to listen to the channel, note when the channel is busy, and calculate the local fractional channel load. Since S_{max} in Equation (1) gives the optimal utilization of the channel, if one senses that the channel loading reaches a certain fraction of the channel capacity, this would indicate a high probability of collision [Li et al. 2001].

Listening to the channel consumes a significant portion of energy [Xu et al. 2001] in a node. Therefore, performing this operation all of the time is not practical in sensor networks. In Section 3.1, we propose a sampling scheme that activates local channel monitoring only at the appropriate time to minimize the energy cost while forming an accurate estimate of conditions.

Channel loading and buffer occupancy provide a good indication of congestion detection for hop-by-hop flow control, but the scope of this control is inherently local. Yet, hop-by-hop flow control has limited effect, for example, in mitigating large-scale congestion caused by data impulses from sparsely located sources that generate high-rate traffic. To understand this limitation in a practical sensor network, we study the channel load and queue performance using a simple two-hop Mica mote [Hill et al. 2000] topology (see Section 4.2). We generate data packets at different rates that drive the network to different levels of congestion and measure the average queue size of the nodes in a small neighborhood that share the wireless medium. A plot of the measured average queue size against the channel load (utilization) is shown in Figure 4. The main figure shows a magnified view of the knee of the entire curve (Inset A). Inset B shows a log-log plot of the knee. We see that the queue size is very small ($\ll 1$) for all channel loads before the channel saturates at a utilization of approximately 70%. Note that the curve resembles a typical M/M/1 queue, except that it saturates at a utilization far lower than one, which is a limitation imposed by the channel idle detection delay (this result is further confirmed when we measure the β value in Section 4.1).

A complementary approach to both the queue occupancy and channel load monitoring congestion detection techniques is proposed in [Tay et al. 2004]. There the authors describe Sift, a MAC protocol tailored for impulse-dominant traffic patterns that draws backoff durations from a non-uniform distribution to minimize the chance of collision. While useful at avoiding collisions, Sift still needs help resolving network congestion.

3. CODA DESIGN

Hotspots (i.e., congestion) can occur in different regions of a sensor field due to different congestion scenarios that arise. This motivates the need for CODA's open loop hop-by-hop backpressure and closed loop multi-source regulation mechanisms. These two control mechanisms, while insufficient in isolation, complement each other. Different rate control functions are required at different nodes in the sensor network depending on whether they are sources, sinks, or intermediate nodes. Sources know the properties of the traffic they inject while intermediate nodes do not. Sinks are best placed to understand the fidelity rate of the received signal, and in some applications, sinks are powerful nodes that are capable of performing sophisticated heuristics. The goal of CODA is to maintain low or no cost operations during normal conditions, but be responsive enough to quickly mitigate congestion around hotspots once it is detected. In what follows, we discuss CODA's backpressure and multi-source regulation mechanisms.

3.1 Open Loop Hop-by-Hop Backpressure

Backpressure is the primary fast time scale control mechanism when congestion occurs. The main idea is to use the components mentioned in Section 2.2 to do local congestion detection at each node with low cost. Once congestion is detected, the receiver will broadcast a backpressure signal to its neighbors and at the same time make local adjustments to prevent propagating the congestion downstream.

A node broadcasts backpressure signal as long as it detects congestion. Backpressure signals are propagated upstream toward the source. In the case of impulse data events in dense networks it is very likely that the backpressure may propagate directly to the sources. Nodes that receive backpressure signals could throttle their sending rates (e.g., be silent for a random period of time) or regulate data rates based on some local congestion policy (e.g., AIMD).

When an upstream node (toward the source) receives a backpressure signal, based on its own local network conditions it determines whether or not to further propagate the backpressure signal upstream. For example, nodes do not propagate the backpressure signal if they are not congested.

We use the term *depth of congestion* to indicate the number of hops that the backpressure signal has traversed before a non-congested node is encountered. The depth of congestion can be used by the routing protocol and local packet drop policies to help balance the energy consumed during congestion across different paths. Two simple schemes can be used: (i) consider the instantaneous depth of congestion as an indicator to the routing protocol to select better paths, thereby reducing traffic over the paths suffering deep congestion, (ii) silently suppress or drop important signaling messages associated with routing or data dissemination protocols (e.g., interests [Intanagonwiwat et al. 2000], data advertisements [Heinzelman et al. 1999], etc.). The latter scheme would help to *push* event flows out of congested regions and away from hotspots without explicitly coupling congestion control and routing. Further investigation of using depth of congestion to assist routing is out of the scope of this paper.

3.1.1 *Receiver-based Detection.* Both a nearly overflowing queue and a measured channel load higher than a fraction of the optimum utilization are good indications of congestion. The latter provides a probabilistic indication of congestion by observing how closely the channel load approaches the upper bound.

Monitoring the queue size comes almost for free except for a little processing overhead, but it provides only a bimodal indication (see the abrupt transition from < 1 to infinity in Figure 4) with non-negligible latency. Listening to the channel either to measure the channel loading or to acquire signaling information for collision detection provides a fast and good indication but incurs high energy cost if performed all the time. Therefore, it is crucial to activate the latter component only at the appropriate time in order to minimize cost.

Consider the typical packet forwarding behavior of a sensor network node and its normal radio operational modes. The radio stays in the listening mode except when it is turned off or transmitting. When a carrier is detected on the channel, the radio switches into the receiving mode to look for a transmission preamble and continues to receive the packet bit stream. Before forwarding this packet to the next hop, CSMA requires the radio to detect an idle channel which implies listening for a certain amount of time. If the channel is clear during this period, then the radio switches into the transmission mode and sends out a packet. There is no extra cost to listen and measure channel loading when a node wants to transmit a packet since carrier sense is required anyway before a packet transmission. Based on this observation, we conclude that the proper time to activate the detection mechanism is when a node's send buffer is not empty. In other words, a node's radio might be turned off most of the time according to some node coordination schemes (e.g., GAF [Xu et al. 2001], SPAN [Chen et al. 2002], S-MAC [Ye et al. 2002], etc.), but, whenever receiving or transmitting a packet, the radio must reside in the listening mode for a time.

Figure 2 illustrates a typical scenario in sensor networks in which hotspots or congestion areas could be created. In this example, nodes 1 and 4 each send CBR traffic that consumes 50% of the channel capacity through node 2 to node 3 and 5, respectively. Packets that are received by node 2 stay in its queue because of the very busy channel and are eventually dropped. This simple example shows that in a congested neighborhood, a receiver's (e.g., node 2, the forwarding node) buffer occupancy is high or at least non-empty. A node that activates the channel loading measurement during the moment when its buffer is not empty is highly responsive with almost no cost. The channel loading measurement will stop naturally when the buffer is cleared, which indicates with high probability that any congestion is mitigated and data flows smoothly around the neighborhood. Based on this observation, there is little extra cost to measure the channel loading if a node activates channel monitoring only when it is "receiving" a packet and needs to forward it later on. The only time CODA needs to do this is when a node has something to send, and it has to do carrier sense anyway for those situations.

3.1.2 *Minimum Cost Sampling.* Each node uses a simple channel sampling scheme to measure local channel load over a number of sampling *epochs*, where each epoch has a length E equal to a small number of packet transmission times. A node initiates channel sampling when it has a packet to transmit, and probes the MAC for at least one epoch time to measure the channel load. Within each epoch, a node uses non-invasive probing of the MAC states to determine when the fraction of time the radio is busy. The approach allows the radio to be turned off at any time (e.g., during the backoff interval) to save energy, and is also more accurate and faster reacting than calculating the average backoff per transmitted packet. To calculate a longer term channel load average, each node calculates $\overline{\Phi}$ as the exponential average of Φ_n (the measured channel load during epoch n) with parameter α

over the previous N consecutive sensing epochs:

$$\overline{\Phi}_{n+1} = \alpha\overline{\Phi}_n + (1-\alpha)\Phi_n, (n \in \{1,2,...,N\}, \overline{\Phi}_1 = \Phi_1).$$

If the packet transmit buffer is cleared before n counts to N, then the average value is ignored and n is reset to 1. The tuple (N,E,α) offers a way to tune the sampling scheme to accurately measure the channel load for specific radio and system architectures. In Section 4.2, we describe and demonstrate the tuning of these three parameters in an experimental sensor network testbed comprised of Berkeley Mica motes, concluding that congestion detection is not very sensitive to these parameter.

3.1.3 *Backpressure Signal.* In CODA, the channel-measurement based congestion detection is a preventive approach triggering a node to send a message as a backpressure signal as soon as the sensed channel load builds above a threshold (before the neighborhood is heavily congested), or when the buffer occupancy reaches certain high watermark level. At the same time, the node's local congestion policy is triggered. This threshold can simply be S_{max}, as shown in later evaluation sections. Although there is no guarantee that all neighboring nodes will get this message, at least some nodes will get it probabilistically. A node will continue broadcasting this message up to certain maximum number of times with minimum separation as long as congestion persists. The overhead of this method is limited by this maximum suppression rate. Alternatively, a node can set a *congestion bit* in the header of every outgoing packet [Hull et al. 2004] instead of sending explicit backpressure messages. However, this scheme requires all nodes to overhear traffic from the neighborhood, which may be difficult to realize with MACs that duty-cycle to save energy.

The backpressure message can also serve as an on-demand "Clear To Send" (CTS) signal, so that all other neighbors except a single sender (which could be picked randomly, or a node can assign more chances to more desirable senders) can be silenced at least for a single packet transmission time. This deals with hidden terminals and supports an implicit priority scheme in CODA. The "chosen node(s)" embedded in the suppression message can be selected based on data type (all nodes can share a priority list of data types) or other metrics that essentially assign the chosen sender(s) a higher priority to use the bandwidth.

Because of the funneling effect in sensor networks, particularly for sparsely located sources, congestion is most likely to occur at downstream sensors closer to the sink. Therefore, upstream sensors located closer to the sources within the propagation funnel (i.e., data flowing from multiple sources toward a sink) are likely to experience lower channel load, and hence a low queue occupancy according to Figure 4. As a result, the backpressure signal would most likely stop propagating before it reaches the sources. Therefore, the hop-by-hop backpressure mechanism alone is not enough to mitigate large-scale congestion. To address this, a mechanism that resembles end-to-end closed loop control that allows a user to control the desired reporting rate of an application is proposed in the next section.

3.1.4 *Impact on Fairness.* CODA is implemented as a shim in the link layer, directly above the MAC and as such does not impact fairness of the underlying MAC. (The suppress message is broadcasted and has a fair chance to reach every node in the contention region.) Further, CODA supplies an explicit priority access mechanism for different nodes or application traffic types, as demonstrated in Figure 9. In [Ee and Bajcsy 2004], Ee and Bajcsy propose the use of per-child queue to provide a hard notion of fairness. Our "chosen node" priority access scheme is currently used to prioritize traffic from a specific node but can also be used to provide a “soft fairness” in the following way: each node learns its neighbors (e.g., the neighbor table maintained in SP [Polastre et al. 2005]) and randomizes the “chosen node” among its neighbors across different epochs. This provides a notion of fairness, without requiring per child queues.

3.2 Closed Loop Multi-Source Regulation

In sensor networks there is a need to assert congestion control over multiple sources from a single sink in the event of persistent congestion, where the sink acts as a 1-to-N controller over multiple sources. Note that backpressure alone cannot resolve congestion under all scenarios because our design does not propagate the congestion signal in cases where nodes do not locally experience congestion - to do so would be very costly in terms of power and bandwidth consumption.

We conjecture that pure window-based end-to-end control schemes like TCP are not well suited to sensor networks. In addition to the excessive end-to-end acknowledgment overhead, the traffic model is mismatched with the applications (i.e., the data traffic is usually CBR in nature and might experience a sudden increase in the data rate when an interesting event occurs). In TCP, since every incoming ACK increases the transmission window size, low-rate CBR can inflate the window to a very large size that could easily overwhelm the network when an event-based burst of traffic arises. To avoid this TCP characteristic and the high cost of per-packet ACKs, we propose an approach that would dynamically regulate all sources associated with a particular data event. Under normal operation sources would regulate themselves at predefined rates (e.g., based on the data dissemination protocol [Intanagonwiwat et al. 2000] [Heinzelman et al. 1999]) without the intervention of closed loop sink regulation.

When the source event rate (r) is less than some fraction η of the maximum theoretical throughput (S_{max}) of the channel the source regulates itself (e.g., based on the data dissemination protocol [Intanagonwiwat et al. 2000] [Heinzelman et al. 1999]) without the intervention of closed loop sink regulation. When this value is exceeded ($r \geq \eta S_{max}$), a source is more likely to contribute to congestion and therefore closed loop control is triggered. The threshold η here is not the same as the threshold that used in local congestion detection, in fact η should be much smaller because of the result suggested in [Li et al. 2001]. The source only enters sink regulation if this threshold is exceeded. At this point a source requires steady periodic feedback (e.g., ACKs) from the sink to maintain its rate (r). A source enacts sink regulation by setting the regulate bit in the event packets it forwards toward the sink. Reception of packets with the regulate bit set forces the sink to send “aggregated ACKs” (e.g., 1 ACK per 100 events received at the sink) to regulate *all sources* associated with a particular data event. ACKs could be sent in an application specific manner. For example, the sink could send the ACK only along paths it wants to reinforce in the case of a directed diffusion [Intanagonwiwat et al. 2000] application. The reception of ACKs at sources serves as a self-clocking mechanism allowing the sources to maintain the current event rate (r).

When a source sets its regulate bit it expects to receive an ACK from the sink at some predefined rate, or better, a certain number of ACKs over a predefined period allowing for the occasional loss of ACKs (e.g., due to transient congestion). If a source receives a prescribed number of ACKs during this interval it maintains its rate (r). When congestion builds up ACKs can be lost, forcing sources to drop their event rate (r) according to some rate decrease function (e.g., multiplicative decrease, etc.). The sink can stop sending ACKs

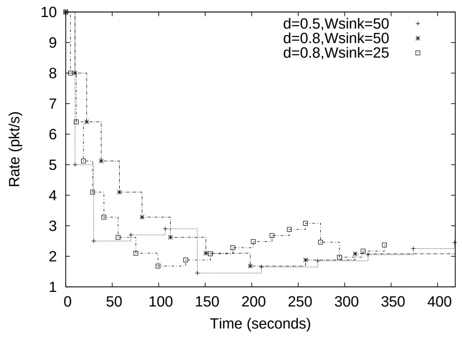


Fig. 5. Closed loop control model. The impact of W_{sink} and the multiplicative decrease factor d.

based on its view of network conditions. The sink is capable of measuring its own local channel loading (ρ) conditions and if this is excessive ($\rho \geq \gamma S_{\max}$) it can stop sending ACKs to sources.

Because the sink expects a certain reporting rate for applications with periodic traffic [Intanagonwiwat et al. 2000] and certain known impulse event types, it can also take application-specific actions when this rate is consistently less than the desired reporting rate (i.e., the fidelity of the signal [Tilak et al. 2002]). In this case the sink infers that packets are being dropped along the path due to persistent congestion and stops sending ACKs to sources. When congestion clears the sink can start to transmit ACKs again, and as a result, the event rate of the source nodes will increase according to some rate increase function (e.g., additive increase). Additionally, since the sink is a point of data collection and in some networks is powerful in comparison to sensors, it can maintain state information associated with specific data types. By observing packet streams from sources at the sink, if congestion is inferred the sink can send explicit control signals to those sources to lower their threshold value η to force them to trigger sink regulation even at a lower rate than others, (i.e., other more important observers). This provides an implicit priority mechanism as part of the closed loop congestion control mechanism.

When the event rate at the sources is reset (e.g., via reinforcement [Intanagonwiwat et al. 2000]) to a value (r) that is less than some factor η of the maximum theoretical throughput (S_{max}) of the channel then the sources begin again to regulate themselves without the need of ACKs from the sink. Such a multi-modal congestion control scheme provides the foundation for designing efficient and low cost control that can be practically implemented in sensor networks based on the Berkeley motes series [Hill et al. 2000], as discussed in Section 4. Overall, closed loop multi-source regulation works closer to the application layer and operates on a much larger (order of magnitude) time-scale than its open loop counterpart.

3.2.1 *A Hybrid Window-based and Rate-based Algorithm.* CODA's closed loop control can be realized as a combination of window-based and rate-based schemes. We define the drop rate (i.e., number of packets dropped in the network per received packet at the sink) as an energy metric called the *energy tax* or E_{Tax}. The packet loss rate p is thus $\frac{E_{Tax}}{1+E_{Tax}}$. With a source event rate of r, the expected number of event packets received at the sink, which is a measure of application fidelity, is $r(1-p)$ or $\frac{r}{1+E_{Tax}}$. The application fidelity is approximately inversely proportional to E_{Tax}.

Recall a key objective of sensor networks is to maximize the operational lifetime while delivering acceptable data fidelity to the applications. This demands a mechanism to control the network so that the energy tax does not exceed an acceptable value, which is an application-specific choice. This is the rationale for CODA's closed loop control. Under overload conditions, assume that the network does not drop ACKs from the sinks, (i.e. ACKs are delivered through high priority queues), and the majority of packet loss in the network is due to congestion. We can then realize this objective through a hybrid rate-based and window-based algorithm. This algorithm governs the window sizes at both source and sink with the E_{Tax} in the following equation:

$$W_{src} = r(\tau_f + \tau_b) + W_{sink}(1 + E_{Tax}) \tag{2}$$

W_{src} is the window size or the number of event packets a source is allowed to send at the current rate r without receiving an ACK from the sink. W_{sink} is the window size or the number of accumulated event packets a sink receives before it sends an aggregated ACK. r is the source rate during the current observation cycle and $(\tau_f + \tau_b)$ is the sum of the forward and backward one-way delays between a source and the sink. The algorithm is such that, if a source does not receive an ACK after it has sent out W_{src} event packets at rate r, it should decrease its rate from r to $d \cdot r$ ($d < 1$ multiplicative decrease). If later an ACK is received at the source within the next observation cycle W_{src}, then the source increases its rate from r to $r + b$ (additive increase). In other words, this control scheme ensures that a source would cut its rate whenever the perceived energy tax rises beyond an acceptable value E_{Tax}. W_{sink} determines the control overhead and the length of the decision period that controls the convergence time of the rate control algorithm. To understand the tradeoff between the control overhead and the convergence time, we numerically evaluate Equation (2), simulating a network that experiences congestion when the source rate exceeds 3 pkt/s but no congestion when the source rate is below 1.5 pkt/s.

In the simulation environment defined in Section 5.1, we evaluate the impact of two values of multiplicative decrease factor d and two values of W_{sink}. In Figure 5, for a fixed W_{sink} (e.g., equal to 50 or 2% control overhead for sending ACKs), we observe that the source rate with a smaller d (i.e., 0.5) drops more quickly than a source with a larger d value (i.e., 0.8). However, the rate with a smaller d oscillates and thus takes a longer time to restore and converge to an acceptable rate that avoids congestion. Therefore, a smaller d can reduce the energy tax but most likely will hurt the fidelity because of the longer convergence time. On the other hand, a larger d would have a larger energy tax because of the slower rate reduction, even though it could achieve higher data fidelity because of the finer levels of granularity of rate reduction and thus can converge faster to an acceptable rate. Note that W_{sink} controls the length of the "observation cycle" and thus a smaller W_{sink} can accelerate the rate reduction process. In Figure 5, we can see that a smaller W_{sink} (i.e., 25) causes the rate of a source with $d = 0.8$ to decrease as fast as $d = 0.5$. This allows the algorithm to achieve the same reduction in energy tax while maintaining high fidelity, at the expense of higher control overhead (i.e., an increase from 2% to 4%) because of the smaller value of W_{sink}. We study these parameter tradeoffs in our mote testbed and discuss the result in Section 4.5 under real-world experimental conditions.

4. EXPERIMENTAL SENSOR NETWORK TESTBED

In this section, we discuss experiences implementing CODA on a real sensor system using the TinyOS platform [TinyOS 2007] on Mica motes [Hill et al. 2000]. CODA is implemented as a shim at the link layer, just above the MAC. We report evaluation results, including measuring the β value, tuning the parameters for accurate channel load measurement, and finally, evaluating CODA with a generic data dissemination application.

The sensor device has an ATMEL 4MHz, low power, 8-bit microcontroller with 128K bytes of program memory, 4K byte of data memory, and a 512KB external flash serves as secondary storage. The radio is a single channel RF transceiver operating at 916 MHz and is capable of transmitting at 10 Kbps using on-off-keying encoding. All our experiments use a Non-Persistent CSMA MAC on top of the Mica motes.

4.1 Measuring the β Value

An important decision that must be made when using CODA's open loop control mechanism described in Section 3.1 is the congestion threshold at which we should start applying backpressure. In the following, we describe the first step in making this decision, a simple one-time procedure to determine the maximum channel utilization achievable with the radio and MAC protocol being used.

As noted in Equation 1 in Section 2.1, for the CSMA MAC protocol, the channel utilization in a wireless network depends on the propagation delay between the nodes with the maximum physical separation that can still interfere with each other's communications, and the channel idle detection delay. In sensor networks, the maximum physical separation is typically tens of meters or less and as such the propagation delay is negligible for most purposes. Thus, if the channel idle detection delay is also negligible, CSMA should provide almost 100% utilization of the offered load of the channel. However, in practice, the utilization is much less due to the latency in the idle channel detection at the MAC layer. We can use the parameter β as defined in Equation 1 to predict how much this latency degrades the maximum channel utilization.

We measure the β value for the Mica mote using a simple experimental setup involving two motes both running TinyOS [TinyOS 2007]. Stopwatches inserted in the MAC provide the basis for the measurement of β. Figure 6 illustrates the placement of the stopwatches within the receive and transmit flows of the Mica MAC layer state machine. Mote A starts its watch when the MAC receives a packet to be sent from the upper layers of the network stack and stops its watch when it detects the start-symbol of an incoming packet from mote B. The locations of the stopwatch trigger points in the mote B MAC are the same as in mote A, but the operations are reversed. It starts the watch when it receives a packet and stops it when it starts to transmit.

A single iteration of the measurement consists of mote A sending a packet to mote B and mote B immediately reflecting the packet back to mote A. Due to the symmetry inherent in the placement of the stopwatch trigger points, β is proportional to half the difference between Stopwatch A and Stopwatch B: $\beta = \frac{(\text{StopwatchA}-\text{StopwatchB})}{(2*(\text{Packet transmission time}))}$. Over 50 iterations, we measure an average β of 0.030 ± 0.003 (with confidence level of 95%) for the Mica motes. Substituting β into Equation 1, the standard expression for CSMA throughput (S_{max}), we predict a maximum channel utilization of approximately 73%. The same measurement procedure executed on the Mica2 mote predicts a maximum throughput of approximately 36% with the default MAC in TinyOS-1.1.0. Note that the measurement of β is simply a

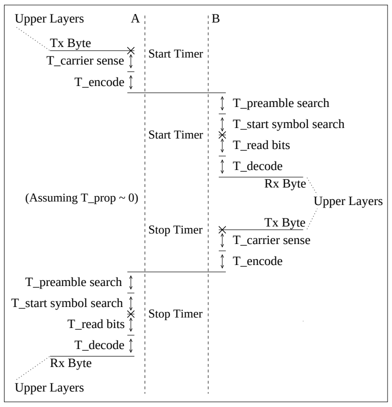


Fig. 6. MAC layer stopwatch placement for β measurement. Diagram of receive and transmit state flows in the TinyOS MAC component code. Placement of the stopwatch start/stop trigger points are marked with an X.

way to provide theoretical rationale to determine a reasonable threshold. Alternatively, one can always determine a suitable threshold experimentally.

4.2 Channel Loading Measurement and Utilization

Setting the channel loading threshold that will trigger the backpressure mechanism requires consideration of the tradeoff between energy savings and fidelity. Conserving energy implies a strategy that senses the channel load sparsely in time (fewer timer interrupts and processing). However, the channel load measurement is most accurate when sensing densely in time. As a compromise between dense and sparse sampling, we use the scheme discussed in Section 3.1.2 where the channel load is measured for N consecutive epochs of length E (with some fixed channel state sampling rate within this epoch), and an exponential average, with parameter α, is calculated to represent the sensed channel load. The problem then becomes to manipulate these three parameters (N,E,α) so that the node's sensed channel load is as close as possible to the actual channel load.

To do this optimization experimentally, we use two motes running TinyOS with a CSMA MAC. Mote S is a randomized CBR source that sends at 4 packets per second. Mote R is the receiver that senses the channel load using the scheme mentioned in the previous paragraph. The channel is sampled once per millisecond for each epoch E for a total of N epochs. Using this setup we tested all combinations of $N \in \{2,3,4,5\}$; $E \in \{100ms, 200ms, 300ms\}$ and $\alpha \in \{0.75, 0.80, 0.85, 0.90\}$. A time series average, of the exponential averages, is taken over 256 seconds for each combination (1024 packets are sent). Using this method we found that the combination $(4, 100ms, 0.85)$ yielded the average sensed channel load at mote R closest to the actual average channel load calculated by mote S with an accuracy of 0.16±0.07%. In general, we observe that the detection accuracy is not very sensitive (the difference is within 5%) to these three parameters. Therefore, manual calibration for each new CSMA-based radio might not be necessary. Our experiences with the new generation of Mica2 mote, which uses a different radio/MAC than Mica, are consistent with this conjecture.

In order to address the more realistic case of a node that both listens to and forwards packets, a third mote F is added to the previous experimental setup with all motes well

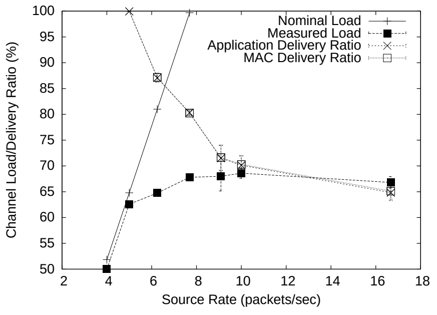


Fig. 7. A limit on measured channel load is imposed by β. Nominal load curve increases with constant slope as the source packet rate increases, while the measured load saturates at a value below 70%.

within the transmission range of each other. Mote F forwards packets sent from mote S with a small random jitter added for application phase shifting purposes [Woo and Culler 2001], and also senses the channel load using the same scheme with the same (N, E, α) parameters that mote R uses. There is now contention for the channel since there are two packet sources (motes S and F). To minimize the probability of dropping packets from the application layer because of buffer limitations, we use a buffer size of three packets at the MAC layer. This decision is based on the queue performance shown in Figure 4, where the average queue size is $\ll$ three before the channel saturates. Mote R remains as a reference to check the channel load sensed by mote F and also to track of the number of packets sent by motes S and F to calculate the delivery ratio.

With mote S sending 1024 packets, we measure the packet delivery ratio and channel load sensing accuracy using different source packet rates (*viz*. $4, 5, 6.25, 7.69, 9.09, 10, 16.67$). The average sensed channel load at R and F, along with the nominal channel load (calculated based strictly on offered load), are plotted against the source packet rate in Figure 7.

Figure 7 shows the β-dependency of the CSMA MAC on the Mica mote. We can see from the plot of the nominal channel load that the offered load is more than enough to saturate the channel at points above 7.69 packets per second (source packet rate). However, we can also observe that regardless of the source packet rate, the measured channel load/utilization saturates below 70%. This is in agreement with the limitation predicted by β (as shown in Section 4.1), if we can assume that packet collision and buffer limitation do not contribute significantly to the observed reduced channel load. To verify this assumption, we analyze the packet delivery ratio at both the MAC and application layer in Figure 7.

We define the MAC packet delivery ratio as the percentage of packets sent by the MAC layer at motes S and F that are received by mote R. The application delivery ratio is the percentage of packets sent by the application layer (i.e., passed down to the MAC queue) at motes S and F that are received by mote R. Figure 7 shows that both application and MAC delivery ratios match each other closely, indicating that nearly every packet that gets into the MAC queue is sent and received successfully, eliminating the effect of packet collision and buffer overflowing in the reduced channel load.

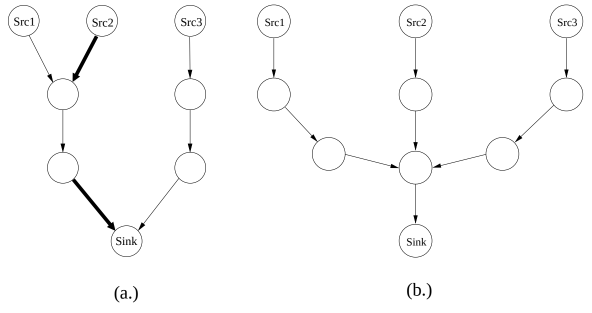


Fig. 8. Experimental sensor network testbed topologies. Packets are unicasted. In (a) nodes are well-connected, representing a dense deployment of nodes; in (b), nodes are arranged to capture the funneling effect in a larger network with sparsely located sources.

4.3 Energy Tax, Fidelity Penalty, and Power

We define three metrics to analyze the performance of CODA on sensing applications:

Average Energy Tax. This metric calculates (tot. num. of pkts. dropped in the network)/(tot. num. of pkts. rcvd. at the sinks) over a given time period. Since packet transmission/reception consumes the main portion of the energy of a node, the number of wasted packets per received packet directly indicates the energy saving aspect of CODA when compared to the case of systems without CODA.

Average Fidelity Penalty. We define the data fidelity as the delivery to the sink of the required number of data event packets within a certain time limit (i.e., event delivery rate). Fidelity penalty measures the difference between the average number of data packets received at a sink when using CODA and when using the ideal scheme discussed in the Appendix. Since CODA's control policy is to rate control the sources during periods of congestion, fidelity is necessarily degraded on the average. This fidelity difference, when normalized to the ideal fidelity obtained at the sink, indicates the fidelity penalty for using CODA. A lower fidelity penalty is desired by CODA to efficiently alleviate congestion while attempting not to impact the system performance seen by sensing applications.

Power. This metric calculates (data fidelity)/(energy tax). Traditional end-to-end congestion control schemes often define power as the throughput/delay where the objective function is to maximize the power. We borrow the same idea but maximize the power by operating the network at minimize energy tax (thereby maximizing the operational lifetime of the network) while delivering acceptable data fidelity to the applications. This is the objective of our closed loop control.

4.4 Open Loop Control

We create a simple generic data dissemination application to evaluate our congestion control scheme in a wireless sensor network. The simple application implements the open loop fast time scale component of our scheme using TinyOS and runs on our Mica mote testbed. When an intermediate (non source/sink) node receives a packet to forward, it enables channel load sensing. It disables sensing when its packet queue is emptied. If the channel load exceeds a given threshold value (e.g., 73% as discussed in Section 4.1) during the sensing period or its buffer overflows, it transmits a backpressure packet. The sources use a multiplicative rate reduction policy. When a source receives a backpressure message, it reduces

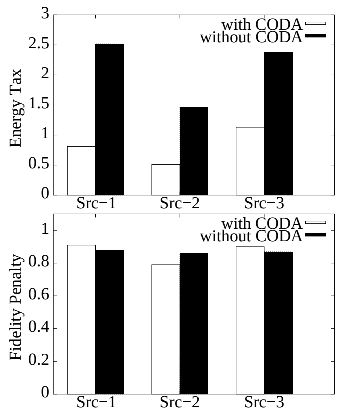


Fig. 9. Improvement in energy tax with minimal fidelity penalty using CODA. Priority of Src-2 evident from the fidelity penalty results.

its rate by half. A minimum rate is imposed, such that a higher-rate source that does not hear a particular suppress message cannot shut off a low-rate source. An intermediate node stops transmitting for a small random number of packet transmission times (on the order of $N*E/\text{pkt\_tx\_time}$ seconds) when it receives a backpressure message to allow local queues and priority upstream queues to drain, unless it is a "chosen node" (c.f. Section 3.1.3). No link-layer ACKs are used in any testbed experiments.

The experimental sensor network testbed topology is shown in Figure 8.a. Packets are unicasted, with the arrows in Figure 8.a indicating the unicast paths. The topology represents a dense deployment of motes so that the contention regions of many nodes in the graph overlap in a time-varying manner (hidden terminals can exist), nodes at the same hop-distance from the sink are mutually interfering, and the sources are isolated from the sink. The local congestion policy of the intermediate nodes can include the designation of a "chosen parent" (i.e., the chosen node, as discussed in Section 3.1.3) or set of parents, such that a backpressure message sent by this node will invoke the suppression method at its neighbors except for the chosen parent(s). This supports traffic prioritization. In Figure 8.a, the thick arrows show the "chosen paths". Paths funnel events toward the sink node. The three source nodes provide a high traffic load to the network, representing a data impulse. The source rates are chosen considering the available bandwidth to engineer congestion at each hop, allowing for the demonstration of the CODA open loop priority channel access feature (Src-1: 8pps (packets per second), Src-2: 4pps, Src-3: 7pps).

The sink node maintains received packet counters for each source. Each source node counts the number of packets it sends over the air and the number of packets the application tries to send. The difference between these last two counters measures the number of packets the MAC layer drops.

Using ten 120-second trials, we obtain average values for the packets received, sent, and attempted to be sent but failed (e.g., because of a busy channel, buffer overflow, etc.) for each of the three sources. From this measured data, we calculate the energy tax and fidelity penalty for each of the three sources. Figure 9 shows the result of experiments with and without CODA enabled. We can see from the figure that with a small fidelity penalty compared with a non-CODA system we can achieve a 3x reduction in energy tax on average.

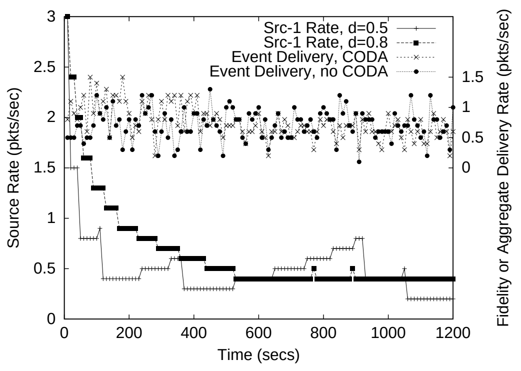


Fig. 10. Time series traces that present the rate control dynamics and the event fidelity/delivery performance of CODA. CODA's rate control scheme does not increase the degree of variability to the event delivery performance.

We observe that without CODA the fidelity penalty is the same for all three sources. With CODA the penalty for Src-2 is much less than the other two sources. In contrast with the other sources, the fidelity penalty for Src-2 is less with CODA than without CODA. The data type of Src-2 has the highest priority; the CODA suppression mechanism prioritizes Src-2 packets.

4.5 Combining Open Loop and Closed Loop Control

We reuse the application described above but add more motes to the testbed to capture the funneling effect in a larger network, as shown in Figure 8.b. The topology provides a bi-directional multi-hop environment (with time-varying characteristics) between sources and sink, and the sources are isolated from the choke point and from each other. The three branches are mostly isolated from each other except within the 1-hop neighborhood of the choke point. We implement CODA's closed loop control component, as discussed in Section 3.2, into the application running in parallel with the open loop component. The first experiment examines the rate control dynamics of CODA. Figure 10 presents time series traces taken at one of the sources in the topology, (i.e., Src1 in Figure 8.b). W_{sink} is set to 25 (representing 4% of control overhead) and we examine our closed loop model using two values for multiplicative factor d, of 0.5 and 0.8, respectively. The two time series traces (source rate) closely resemble the numerical example traces shown in Figure 5. We observe that the source rate oscillates when using a smaller $d = 0.5$ and converges more slowly when compared to its counterpart when $d = 0.8$. In the experiment, the open loop control component is running in parallel and backpressure signals are originated from the mote closest to the sink (i.e. at the funnel neck). However, we observe that none of the signals propagated back to any of the sources, confirming our postulation regarding the limitations of open loop control discussed in Section 2.2.2.

To understand the impact of our rate control algorithm on the stability of event delivery/fidelity at the sink, we plot the event delivery rate measured at the sink as time series traces in Figure 10. While the traces exhibit a high degree of variability even without any rate control (trace with no CODA), we observe that CODA rate control does not increase the degree of variation. Rather, the trace with CODA is more stable after the rate converges to a value that is determined by the E_{Tax} threshold in the closed loop model.

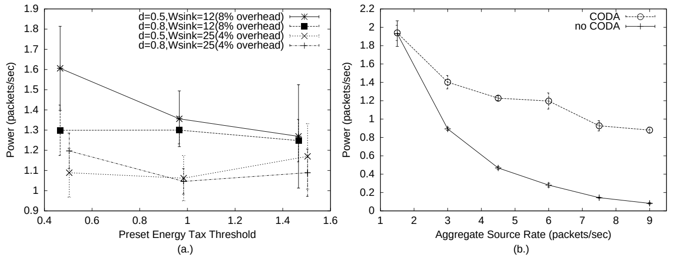


Fig. 11. (a.) shows the tradeoff between fidelity and energy tax that obtain the most benefit, i.e. maximum Power for the network; (b.) shows the comparative Power between networks with and without CODA enabled.

We next examine our closed loop control model that controls the tradeoffs between the perceived energy tax of the network and the perceived application fidelity. Recall (Section 3.2.1) that a smaller value of d yields a larger saving of energy tax but negatively impacts the data fidelity. Similarly, allowing a smaller value of E_{Tax} threshold in the network (Equation 2) would reduce W_{src}, hence a smaller observation cycle. This makes the control algorithm more sensitive to packet loss, thus reducing rate and energy tax more aggressively, but would adversely affect the data fidelity. Achieving a balance obtains the most benefit from the closed loop control. We present results based on the power metric defined in Section 4.3, using different control parameters, in Figure 11. The results clearly indicate that a smaller value of E_{Tax} almost always guarantees a higher power. Therefore, a smaller observation cycle can gain more in energy tax than it harms the fidelity. On the other hand, with a smaller value of d the gain is less stable as observed in the high degree of variability (indicated by the error bars, which represent the corresponding 95% confidence intervals).

Finally, Figure 11 presents the performance gain of CODA compared to the cases without CODA under different network workload. CODA is able to prevent the network power from degrading exponentially when the workload increases.

5. SIMULATION RESULTS

We use packet-level simulation to study the effects of network scale on the performance and dynamics of CODA.

5.1 Simulation Environment

We implemented both open loop backpressure and closed loop regulation in the ns-2 [The network simulator - ns2.] simulator in their simplest instantiation; that is, a simple AIMD function is implemented at each sensor source by an application agent. The reception of backpressure messages at the source, or, in the case of closed loop control, not receiving a sufficient number of ACKs from the sink over a predefined period of time, will cause a source to cut its rate by half (i.e., $d = 0.5$). For intermediate nodes (non source/sink), local congestion policy is such that a backpressure message will halt a node's transmission for a small random number of packet transmission times (on the order of $N{*}E/\text{pkt\_tx\_time}$ seconds) unless a node is the chosen node specified in the backpressure message, to allow local queues and priority upstream queues to drain.

In all our experiments, we use random topologies with different network sizes. We generate sensor networks of different sizes by placing nodes randomly in a square area. Different sizes are generated by scaling the square size and keeping the nominal radio

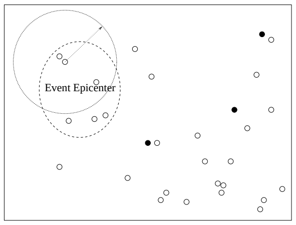


Fig. 12. Network of 30 nodes. Sensors within the range of the event epicentre, which is enclosed by the dotted ellipse, generate impulse data when an event occurs. The circle represents the radio range (40m) of the sensor.

range (40 meters in our simulations) constant in order to approximately keep the average density of sensor nodes constant. In most of our simulations, we study five different sensor fields with size ranging from 30 to 120 nodes in increments of 20 nodes. For each network size, our results are averaged over five different generated topologies and each value is reported with its corresponding 95% confidence interval.

Our simulations use a 2 Mbps IEEE 802.11 MAC provided in ns-2 simulator, with some modifications. First, we disable the use of RTS/CTS exchanges and link-layer ARQ for data packets. We do this for the reasons discussed in Section 2.1 because we want to capture the realistic cases where reliable delivery of data is not needed and the fidelity can be compromised to save energy. Although we use IEEE 802.11 in the simulation, most sensor platforms use simpler link technologies where the ARQ is not enabled by default, (e.g., Berkeley motes). Next, we added code to the MAC to measure the channel loading using the epoch parameters $(N = 3, E = 200ms, \alpha = 0.5)$, as defined in Section 3.1.2. The choice of the parameters is not crucial because the ns-2 simulator does not model the details of the IEEE 802.11 physical layer. The MAC broadcasts backpressure messages when the measured channel load exceeds a threshold of 80%. We have added code to model the channel idle detection delay with a β of 0.01, which yields a S_{max} of 80%. Closed loop multi-source regulation is implemented as an application agent attached to source-sink pairs. W_{sink} is set to 100 and the E_{Tax} threshold to 2 for the closed loop control parameters, modeling the desired energy tax parameters of a dummy application.

Finally, we use directed diffusion [Intanagonwiwat et al. 2000] as the routing core in the simulations since our congestion control fits nicely into the diffusion paradigm, and since doing so allows insight into CODA's interaction with a realistic data routing model where congestion can occur.

In most of our simulations, we use a fixed workload that consists of 6 sources and 3 sinks. All sources are randomly selected from nodes in the network. Sinks are uniformly scattered across the sensor field. A sink subscribes to 2 data types corresponding to two different sources. This models the typical case in which there are fewer sinks than sources in a sensor field. Each source generates packets at a different rate. An event packet is 64 bytes and an interest packet is 36 bytes in size [Intanagonwiwat et al. 2000], respectively. Nodes have queues of length ten.

5.2 Results and Discussion

We evaluate CODA under the three distinct congestion scenarios discussed in the Introduction section to best understand its behavior and dynamics in responding to the different types of congestion found in sensor networks. First we look at a densely deployed sensor

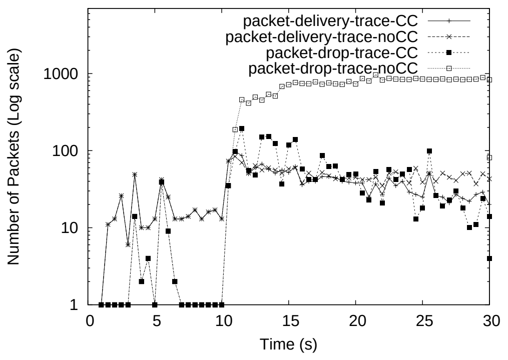


Fig. 13. Traces for densely deployed sources that generate high rate data.

field that generates impulse data events. Next, we examine the behavior of our scheme when dealing with transient hotspots in sparsely deployed sensor networks of different sizes. Last, we examine the case where both transient and persistent hotspots occur in a sparsely deployed sensor field generating data at a high rate.

5.2.1 *Congestion Scenario - Dense Sources, High Rate.* We simulate a network with 30 nodes, as shown in Figure 12, emulating a disaster-related event (e.g., fire, earthquake) that occurs 10 seconds into the simulation. Each node within the epicenter region, which is enclosed by the dotted ellipse, generates at least 100 packets per second sent toward the sinks. shown as filled black dots in the figure.

Figure 13 shows both the number of packets delivered and the packets dropped as time series traces (y axis is log scale). For the packet delivery trace, we count the number of data packets a sink receives every fixed interval of 500ms, which indicates the fidelity of the data samples. For the packet dropped trace, we count the number of data packets dropped within the whole network every 500ms.

From the traces, it is clear that the difference in data delivery (fidelity) with and without CODA is small, while the number of packets dropped is an order of magnitude smaller (hence the energy savings) when congestion control is applied. We can also observe that the congestion is effectively relieved within 2 to 3 seconds, showing the reactivity of CODA. The delivery plot reflects the real system goodput, which is highly dependent on the system capacity, indicating the maximum channel utilization. When impulses happen, the channel is saturated so it can deliver only a fraction of the event's data. CODA's open loop backpressure (even with a very simple policy) adapts well to operate close enough to the channel saturation, as shown in Figure 13, while efficiently alleviating congestion. This greatly reduces the number of packets dropped thereby saving energy, which is the key objective function for CODA. The same simulation scenario is repeated 5 times using different topologies of the same size. Overall, using CODA obtains packet (energy) saving up to $88 \pm 2\%$ while the fidelity penalty paid is only $3 \pm 11\%$.

5.2.2 *Congestion Scenario - Sparse Sources, Low Rate.* To examine the ability to deal with transient hotspots, in these simulations all six sources send at low data rates, at most 20 packets per seconds. Four of the sources are randomly selected so that they are turned on and off at a random time between 10 and 20 seconds into the simulation.

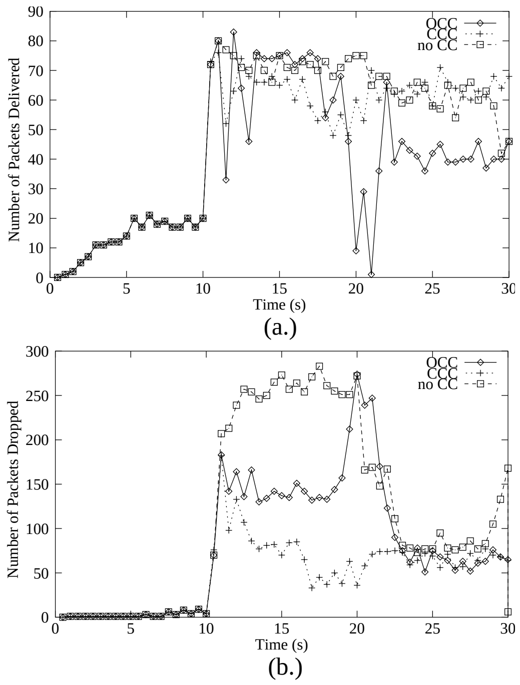


Fig. 14. (a) Packet delivery and (b) Packet dropped time series traces for a 15-node network with low rate traffic. The plots show the traces for three cases: when only open loop control (OCC) is used, both open loop and closed loop control (CCC) are enabled and when congestion control is disabled (no CC).

Figure 14 shows the packet delivery and packet drop traces for one of the simulation sessions in a network of 15 nodes. We simulate three cases: when only open loop control is used, both open loop and closed loop control are enabled and when congestion control is disabled. Observe in Figure 14(a), the difference in fidelity between the three cases is small, except for around 20 seconds in to the trace, when only open loop control is used. Figure 14(b) shows a large improvement in energy savings (i.e., packet drop reduction) especially when closed loop control is also enabled together with open loop control. Again, the figure shows that at around 20 seconds into the trace, open loop control cannot resolve congestion as there is no reduction in the number of dropped packets and with the OCC priority mechanism disabled there is low delivery during this period. This is because transient hotspots turn into persistent congestion at around 18 seconds into the trace until four of the sources turn off after 20 seconds, after which the congestion eases, queues drain, and packet delivery rebounds. Open loop control cannot deal with persistent congestion unless the hotspots are close to the sources, as discussed in Section 2.2.2. On the other hand, the trace corresponding to closed loop regulation also shows that the fidelity is maintained while effectively alleviating congestion with only a small amount of additional signaling overhead. Importantly, the signaling cost of CODA is less than 1% with respect to the number of data packets delivered to the sink.

The same behavior can be observed in Figure 15, where the two metrics (i.e., energy tax and fidelity penalty) are plotted as a function of the network size. Note that when using only open loop control, the energy savings has a large variation, indicated by the

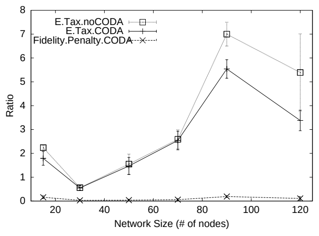
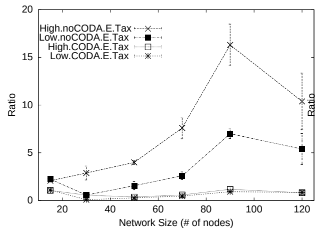
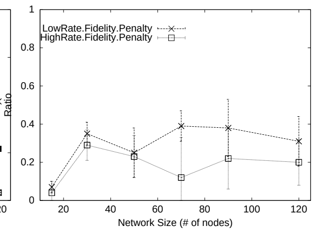


Fig. 15. Average energy tax and fidelity penalty as a function of the network size when only CODA's open loop control is used.



(a)



(b)

Fig. 16. Performance at scale. (a.) shows energy tax as a function of network size for high and low rate data traffic. The difference between the data points with and without CODA indicates the energy saving achieved by CODA. (b.) shows fidelity penalty versus network size for high and low rate data traffic.

error bars that represent 95% confidence intervals. This indicates that congestion is not always resolved, especially for larger-sized networks. This is because in larger networks, persistent hotspots, which localized open loop control is unable to resolve, are more likely to occur given the long routes between source-sink pairs. When closed loop control is also enabled, the energy savings is large, up to 500% with a small variation, and increases with the growing network size, as shown in Figure 16(a).

Overall, the gain from using open loop control in larger networks is limited. Hotspots are likely to persist when the sources are generating data at a low rate because of possible long routes and the funneling effect. Enabling closed loop control even at low source rates can improve the performance significantly, with the addition of a small overhead for the control packets from sinks. Note that the amount of overhead is only a small fraction (i.e., 1%, of the number of data packets that the sink receives). This result suggests that except for small networks, always enabling closed loop control is beneficial, regardless of the source rate. This is an important observation that guides the use of CODA's mechanisms in sensor networks.

5.2.3 *Congestion Scenario - Sparse Sources, High Rate.* We examine the performance of our scheme in resolving both transient and persistent hotspots where sparsely located sources generate high data traffic. In the simulations, all sources generate 50 packets per second data traffic over the 30 second simulation time. Both open loop and closed loop control are used throughout the simulations. Figure 16(a) shows that CODA can obtain up to 15 times energy savings. Figure 16(b) shows that CODA can maintain a relatively low fidelity penalty of less than 40% as compared to the ideal scheme. Observe that energy tax increases as the network grows in general. However, in Figure 16(a) we can see that the energy tax actually decreases when the network grows beyond the size of 100 nodes (the same can be observed in Figure 15). This is because under a fixed workload, which is the case in our simulations, a network's capacity could increase when the network grows beyond certain sizes. This is because the data dissemination paths from the sources to the sinks spread across a broader network and the funneling effect is lessened.

For this and the other two congestion scenarios discussed in Section 5.2, in our simulation environment we keep the node density constant by scaling the field size as we vary the number of nodes. While we do not investigate high fan-in scenarios in this way, we conjecture that as node density grows (and source density remains the same as discussed in our three congestion scenarios) the packet delivery rate will decrease since more time (bandwidth) is spent on contention resolution. Additionally, the packet drop rate will increase due to more hidden terminal collisions.

6. RELATED WORK

There is a growing interest in the problem of congestion in sensor networks. The need for congestion avoidance techniques is identified in [Tilak et al. 2002] while discussing the infrastructure tradeoffs for sensor networks. Tilak, Abu-Ghazaleh, and Heinzelman [Tilak et al. 2002] show the impact of increasing the density and reporting rate on the performance of the network. While the authors do not propose any congestion avoidance mechanisms, they do note that any such mechanism must converge on a reporting rate that is just sufficient to meet the performance or fidelity of the sensing application. This is an important observation in the context of sensor networks.

Some existing data dissemination schemes [Intanagonwiwat et al. 2000] [Wan et al. 2005] can be configured or modified to be responsive to congestion. For example, directed diffusion [Intanagonwiwat et al. 2000] can use in-network data reduction techniques such as aggressive aggregation when congestion is detected. Other protocols, such as PSFQ (Pump Slowly Fetch Quickly [Wan et al. 2005], a reliable transport protocol for sensor networks) can adapt the protocol (i.e., modulate its pump/fetch ratio) to avoid congestion. However, such approaches involve highly specialized parameter tuning, accurate timing configuration, and an in-depth understanding of the protocol's internal operations. There is a need for a comprehensive set of congestion control mechanisms specifically designed to best fit the unique constraints and requirements of sensor networks and their emerging applications. These mechanisms should provide a general set of components that can be plugged into applications or the MAC in support of energy efficient congestion control.

In [Woo and Culler 2001] a comprehensive study of carrier sensing mechanisms for sensor networks is reported. The authors propose an adaptive rate control mechanism that supports fair bandwidth allocation for all nodes in the network. Implicit loss (i.e., failed attempts to inject a packet into the network) is used as a collision signal to adjust the transmission rate of nodes. The paper focuses on fairness issues in access control but not congestion control. In [Hull et al. 2003] the authors assume homogeneous applications in an indoor environment where sinks are sensor access points (SAPs) that work collaboratively to collect data from a sensor field. The authors propose using a combination of a hop-by-hop flow control scheme and a SAP selection routing metric that considers packet loss probabilities, path load, and path length to select congestion-free paths to SAPs, improving the capacity of the network.

In [Sankarasubramaniam et al. 2003] an event-to-sink reliable transport protocol (ESRT) provides support for congestion control. ESRT regulates the reporting rate of sensors in response to congestion detected in the network. This paper is inspired, as our work is, by the observations of Tilak, Abu-Ghazaleh, and Heinzelman [Tilak et al. 2002] discussed above. ESRT monitors the local buffer level of sensor nodes and sets a congestion notification bit in the packets it forwards to sinks if the buffer overflows. If a sink receives a packet with the congestion notification bit set it infers congestion and broadcasts a control signal informing all source nodes to reduce their common reporting frequency according to some function. As discussed in [Sankarasubramaniam et al. 2003] the sink must broadcast this control signal at high energy so that all sources in the sensor field can hear it. Such a signal has a number of potential drawbacks, however, particularly in large sensor networks. Any on-going event transmission would be disrupted by such a high powered congestion signal to sources. In addition, rate regulating all sources in the manner proposed in [Sankarasubramaniam et al. 2003] is fine for homogeneous applications where all sensors in the network have the same reporting rate but not for heterogeneous sources. Even with homogeneous sources, ESRT always regulates all sources regardless of where the hotspot occurs in the sensor field or whether the observed hotspot impacts a path between a source and sink. A collision-minimizing CSMA MAC [Tay et al. 2004], optimized for event-driven traffic, uses a non-uniform probability distribution for contention slot selection, but does not support adaptively adjustable traffic prioritization. We believe there is a need to support heterogeneous sources and only regulate those sources that are responsible for, or impacted by, transient or persistent congestion conditions. Furthermore, we believe that closed loop regulation of sources should not use high energy but instead hop-by-hop signaling that does not interfere with on-going data dissemination.

More recently, Ee and Bajcsy study the fairness issues of congestion control in sensor networks [Ee and Bajcsy 2004]. They propose a distributed congestion control algorithm in the transport layer of the traditional network stack model to ensure the fair delivery of packets to a central node. In [Hull et al. 2004], Hull *et al.* experimentally investigate the end-to-end performance of various congestion avoidance techniques in a 55-node sensor networks. They propose a strategy called *Fusion* that combines three congestion control techniques that operate at different layers of the traditional protocol stack. The first of these is a version of hop-by-hop flow control, similar to CODA's open loop control [Wan et al. 2003]. However, Fusion uses only queue monitoring for congestion detection, while CODA uses both queue monitoring and channel sampling for a faster response to congestion. Also, Fusion implements the backpressure signal using passive congestion notification, while CODA uses an explicit notification. Additionally, Fusion incorporates a source rate limiting scheme (similar to the adaptive rate control mechanism proposed in [Woo and Culler 2001], and similar in spirit to the aims of CODA's closed loop component) that meters traffic being admitted into the network, and a prioritized MAC layer that gives a backlogged node priority over non-backlogged nodes for access to the shared medium. Based on an extensive amount of experimental data from their MIST testbed the authors of [Hull et al. 2004] show the adverse effects of network congestion, and demonstrate that by integrating congestion detection and signaling techniques with source rate limiting and a prioritized MAC layer, Fusion can greatly improve the sensor network efficiency, beyond the use of congestion detection and backpressure alone.

A number of other groups have looked at the issue of congestion control in wireless networks other than sensor networks. For example, WTCP [Sinha et al. 1999] monitors the ratio of inter-packet separation for senders and receivers to detect and react to congestion in wireless LANs. SWAN [Ahn et al. 2002] forces sources to re-negotiate end-to-end flows if congestion is detected in wireless ad hoc networks. RALM [Tang and Gerla 2001] employs TCP-like congestion and error control mechanisms for multicast support in wireless ad hoc networks. While multicast congestion control and congestion control in wireless networks are of interest they do not address the same problem space as energy efficient congestion detection and avoidance for sensor networks.

7. CONCLUSION

In this paper, we have presented an energy efficient congestion control scheme for sensor networks called CODA. The framework is targeted at CSMA-based sensors[3], and comprises three key mechanisms: (i) receiver-based congestion detection, (ii) open loop hop-by-hop backpressure, and (iii) closed loop multi-source regulation. We have presented experimental results from a small sensor network testbed based on TinyOS running on Berkeley Mica motes. We defined three performance metrics, average energy tax, average fidelity penalty and power, which capture the impact of CODA on sensing applications' performance. A number of important results came out of our study and implementation. It was straightforward to measure β, channel loading at the receiver, and to evaluate CODA with a generic data dissemination scheme. We have also demonstrated through simulation that CODA can be integrated to support data dissemination schemes and be responsive to a number of different congestion control scenarios that we believe will be prevalent in future sensor network deployments. Simulation results indicated that CODA can improve the performance of directed diffusion by significantly reducing the average energy tax with minimal fidelity penalty to sensing applications. The source code for CODA is freely available from the web (http://www.comet.colulmbia.edu/armstrong).

APPENDIX

Experimentally determining the ideal network fidelity

Assume that there exists an ideal congestion control scheme that is capable of rate-controlling each source to share the network capacity equally without dropping packets. We must then find the upper bound on the network's capacity. The actual capacity of the network is application-specific depending on several factors including the radio bandwidth, MAC operations, routing/data dissemination schemes, and traffic pattern. Assume that the network is homogeneous in the sense that all wireless links are symmetrical and equal.

Def: $C_{max,i}$ *= Maximum data delivery rate of a path i associated with source i, in which the packet drop rate is minimum.*

Consider that multiple distinct sources send data toward a common sink traveling along different paths. These dissemination paths from the sources to the sink share at least one common link, the last hop to the sink. Therefore, the data dissemination capacity for a sink is limited by $max\{C_{max,i}\}$. Thus we can experimentally determine this upper bound and calculate the corresponding ideal fidelity.

Acknowledgment

This work was supported by the National Science Foundation (NSF) Wireless Technology under Award ANI-9979439, by the Army Research Office (ARO) under Award W911NF-04-1-0311, and by Intel Corporation.

[3]The two congestion control components are independent of the MAC used and can work with scheduled-based MACs.

REFERENCES

AHN, G.-S., CAMPBELL, A. T., VERES, A., AND SUN, L.-H. 2002. Supporting service differentiation for real-time and best-effort traffic in stateless wireless ad hoc networks (swan). *IEEE Transactions on Mobile Computing 1,* 3, 192–207.

BERTSEKAS, D. AND GALLAGHER, R. 1991. *Data Networks*, 2nd ed. Prentice Hall, Englewood Cliffs, NJ, USA.

CHEN, B., JAMIESON, K., BALAKRISHNAN, H., AND MORRIS, R. 2002. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *ACM Wireless Networks 8,* 5 (Sep), 85–96.

CLARE, L. P., POTTIE, G., AND AGRE, J. R. 1999. Self-organizing distributed microsensor networks. In *Proc. of 13th Annual Int'l Symp. on Aerospace/Defense Sensing, Simulation, and Controls*. SPIE, Orlando, 229–237.

EE, C. T. AND BAJCSY, R. 2004. Congestion control and fairness for many-to-one routing in sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*. ACM, New York, NY, USA, 148–161.

HEINZELMAN, W. R., KULIK, J., AND BALAKRISHNAN, H. 1999. Adaptive protocols for information dissemination in wireless sensor networks. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*. ACM, New York, NY, USA, 174–185.

HILL, J., SZEWCZYK, R., WOO, A., HOLLAR, S., CULLER, D., AND PISTER, K. 2000. System architecture directions for networked sensors. *SIGPLAN Not. 35,* 11, 93–104.

HULL, B., JAMIESON, K., AND BALAKRISHNAN, H. 2003. Poster abstract: bandwidth management in wireless sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*. ACM, New York, NY, USA, 306–307.

HULL, B., JAMIESON, K., AND BALAKRISHNAN, H. 2004. Mitigating congestion in wireless sensor networks. In *Proc. of the 2nd Conf. on Embedded Networked Sensor Systems*. ACM, Baltimore, 134–147.

INTANAGONWIWAT, C., GOVINDAN, R., AND ESTRIN, D. 2000. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*. ACM, New York, NY, USA, 56–67.

LEVIS, P., MADDEN, S., GAY, D., POLASTRE, J., SZEWCZYK, R., WOO, A., BREWER, E., AND CULLER, D. 2004. The emergence of networking abstractions and techniques in tinyos. In *NSDI'04: Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation*. USENIX Association, Berkeley, CA, USA, 1–1.

LI, J., BLAKE, C., COUTO, D. D., LEE, H., AND MORRIS, R. 2001. Capacity of ad hoc wireless networks. In *Proc. of the 7th Annual Int'l Conf. on Mobile Computing and Networking*. ACM, Rome, 61–69.

POLASTRE, J., HUI, J., LEVIS, P., ZHAO, J., CULLER, D., SHENKER, S., AND STOICA, I. 2005. A unifying link abstraction for wireless sensor networks. In *Proc. of the 3rd Conf. on Embedded Networked Sensor Systems*. ACM, San Diego, 76–89.

POTTIE, G. J. AND KAISER, W. J. 2000. Wireless integrated network sensors. *Commun. ACM 43,* 5 (May), 51–58.

RAJENDRAN, V., OBRACZKA, K., AND GARCIA, J. 2003. Energy-efficient, collision-free medium access control for wireless sensor networks. In *Proc. of 1st Conf. on Embedded Networked Sensor Systems*. ACM, Los Angeles, 181–192.

RAMAKRISHNAN, K. K. AND JAIN, R. 1995. A binary feedback scheme for congestion avoidance in computer networks with a connectionless network layer. *SIGCOMM Comput. Commun. Rev. 25,* 1, 138–156.

SANKARASUBRAMANIAM, Y., ÖZGÜR B. AKAN, AND AKYILDIZ, I. F. 2003. Esrt: event-to-sink reliable transport in wireless sensor networks. In *MobiHoc '03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*. ACM, New York, NY, USA, 177–188.

SINHA, P., VENKITARAMAN, N., SIVAKUMAR, R., AND BHARGHAVAN, V. 1999. Wtcp: a reliable transport protocol for wireless wide-area networks. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*. ACM, New York, NY, USA, 231–241.

TANG, K. AND GERLA, M. 2001. Reliable on-demand multicast routing with congestion control in wireless ad hoc networks. S. Fahmy and K. Park, Eds. *Scalability and Traffic Control in IP Networks 4526,* 1, 109–120.

TAY, Y., JAMIESON, K., AND BALAKRISHNAN, H. 2004. Collision-Minimizing CSMA and its Applications to Wireless Sensor Networks. *IEEE Journal on Selected Areas in Communications 22,* 6 (August), 1048–1057.

The network simulator - ns2. `http://www.isi.edu/nsnam/ns/`.

TILAK, S., ABU-GHAZALEH, N. B., AND HEINZELMAN, W. 2002. Infrastructure tradeoffs for sensor networks. In *Proc. of the 1st Int'l Workshop on Wireless Sensor Networks and Applications*. ACM, Atlanta, 49–58.

TinyOS 2007. `http://www.tinyos.net/`.

WAN, C.-Y., CAMPBELL, A. T., AND KRISHNAMURTHY, L. 2005. Pump-slowly, fetch-quickly (psfq): a reliable transport protocol for sensor networks. *IEEE Journal on Selected Areas in Communications 23,* 4 (April), 862–872.

WAN, C.-Y., EISENMAN, S. B., AND CAMPBELL, A. T. 2003. Coda: congestion detection and avoidance in sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*. ACM, New York, NY, USA, 266–279.

WOO, A. AND CULLER, D. 2001. A transmission control scheme for media access in sensor networks. In *Proc. of the 7th Annual Int'l Conf. on Mobile Computing and Networking*. ACM, Rome, 221–235.

WOO, A., TONG, T., AND CULLER, D. 2003. Taming the underlying challenges of reliable multihop routing in sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*. 14–27.

XU, Y., HEIDEMANN, J., AND ESTRIN, D. 2001. Geography-informed energy conservation for ad hoc routing. In *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*. ACM, New York, NY, USA, 70–84.

YE, W., HEIDEMANN, J., AND ESTRIN, D. 2002. An energy-efficient mac protocol for wireless sensor networks. In *Proceedings 21st International Annual Joint Conference of the IEEE Computer and Communications Societies*. New York, New York, USA, 1567–1576.