Mitigating Congestion in Wireless Sensor Networks

Bret Hull, Kyle Jamieson, Hari Balakrishnan
MIT Computer Science and Artificial Intelligence Laboratory
The Stata Center, 32 Vassar St., Cambridge, MA 02139
{bwhull, jamieson, hari}@csail.mit.edu

ABSTRACT

Network congestion occurs when offered traffic load exceeds available capacity at any point in a network. In wireless sensor networks, congestion causes overall channel quality to degrade and loss rates to rise, leads to buffer drops and increased delays (as in wired networks), and tends to be grossly unfair toward nodes whose data has to traverse a larger number of radio hops.

Congestion control in wired networks is usually done using end-to-end and network-layer mechanisms acting in concert. However, this approach does not solve the problem in wireless networks because concurrent radio transmissions on different "links" interact with and affect each other, and because radio channel quality shows high variability over multiple time-scales. We examine three techniques that span different layers of the traditional protocol stack: hop-by-hop flow control, rate limiting source traffic when transit traffic is present, and a prioritized medium access control (MAC) protocol. We implement these techniques and present experimental results from a 55-node in-building wireless sensor network. We demonstrate that the combination of these techniques, *Fusion*, can improve network efficiency by a factor of *three* under realistic workloads.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols.

General Terms

Measurement, performance, design, experimentation.

Keywords

Wireless sensor networks, congestion control, flow control, rate limiting, network performance.

This paper is based upon work supported by the National Science Foundation under Grant No. 0205445. This work was also supported by the MIT Project Oxygen partnership, Intel Corporation, and a Sloan Foundation Fellowship.

SenSys'04, November 3–5, 2004, Baltimore, Maryland, USA.

1. INTRODUCTION

Provisioning a wireless sensor network so that congestion is a rare event is extremely difficult. Sensor networks deliver myriad types of traffic, from simple periodic reports to unpredictable bursts of messages triggered by external events that are being sensed. Even under a known, periodic traffic pattern and a simple network topology, congestion occurs in wireless sensor networks because radio channels vary in time (often dramatically) and concurrent data transmissions over different radio "links" interact with each other, causing channel quality to depend not just on noise but also on traffic densities. Moreover, the addition or removal of sensors, or a change in the report rate can cause previously uncongested parts of the network to become under-provisioned and congested. Last but not least, when sensed events cause bursts of messages, congestion becomes even more likely.

In traditional wired networks and cellular wireless networks, buffer drops and increased delays are the symptoms of congestion. Over the past many years, researchers have developed a combination of end-to-end rate (window) adaptation and network-layer dropping or signaling techniques to ensure that such networks can operate without collapsing from congestion. In addition to buffer overflows, a key symptom of congestion in wireless sensor networks is a degradation in the quality of the radio channel caused by an increase in the amount of traffic being sent in *other* parts of the network. Because radio "links" are not shielded from each other in the same way that wires or provisioned cellular wireless links are, traffic traversing any given part of the network has a deleterious impact on channel quality and loss rates in other parts of the network. Poor and time-varying channel quality, asymmetric communication channels, and hidden terminals all make even well-regulated traffic hard to deliver. In addition, under traffic load, multi-hop wireless sensor networks tend to severely penalize packets that traverse a larger number of radio hops, leading to large degrees of unfairness.

This paper studies three congestion control techniques that operate at different layers of the traditional protocol stack, and shows that the adverse effects of network congestion can be greatly alleviated when they operate in concert. The first technique is *hop-by-hop flow control*, in which nodes signal local congestion to each other via backpressure, reducing packet loss rates and preventing the wasteful transmissions of packets that are only destined to be dropped at the downstream node. The second technique is a *source rate limiting* scheme to alleviate the serious unfairness toward sources that have to traverse a larger number of wire-

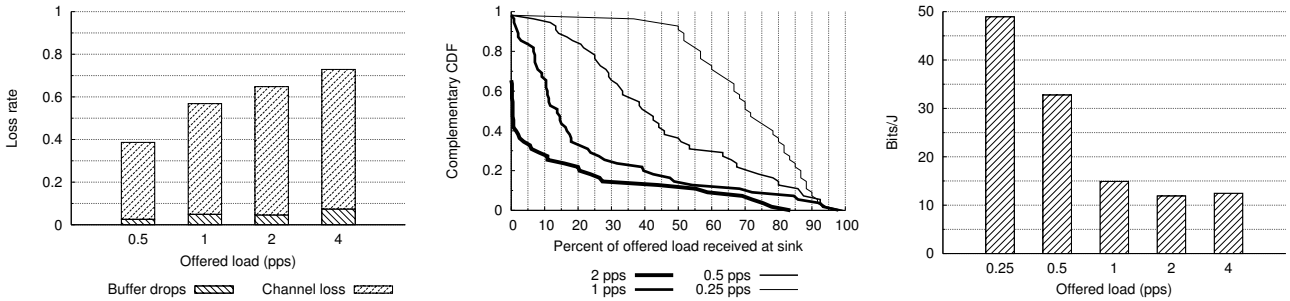

Figure 1: Congestion collapse in a testbed deployment with no congestion control strategy. Channel and buffer loss rate as a function of per-node offered load (*left*). Percentage of each node's offered load that is received at the sink (complementary CDF, *center*). Network-wide bits successfully transmitted per unit energy (*right*).

less hops. The third technique is a *prioritized MAC* layer that gives a backlogged node priority over non-backlogged nodes for access to the shared medium, thus avoiding buffer drops. We combine these techniques into a strategy called *Fusion*. In isolation, each technique helps somewhat, but when acting in concert, Fusion dramatically improves network efficiency, fairness, and channel loss rates. These experimental findings, together with the design details of the aforementioned mechanisms, are the primary contributions of this paper.

In developing the techniques, we borrow heavily from previous work. For example, our hop-by-hop flow control scheme was inspired by work done on wired networks and by recent work [22] that applies the idea to wireless sensor networks. Our source rate limiting scheme bears some similarity to, and was inspired by, previous work as well [24]. However, the details of our schemes and their synergistic operation, as well as a detailed experimental evaluation of these approaches both individually and in concert, are important novel contributions of our work. We evaluate each scheme in a 55-node indoor wireless sensor network testbed. In our testbed, Fusion improves efficiency by a factor of three and fairness by a factor of more than two.

In the next section, we present the results of experiments that highlight the congestion problem in wireless sensor networks and describes various performance metrics that we use to evaluate different congestion control methods. We have developed a network stack in TinyOS that implements several variants of the three congestion control techniques mentioned above; Section 3 discusses how they operate together to control congestion. In Section 4, we study each congestion control mechanism in isolation and in concert, over different traffic patterns. Section 5 presents a detailed comparison with related work, and Section 6 summarizes our main results and describes future directions.

2. THE CONGESTION PROBLEM

This section diagnoses the two key symptoms of congestion collapse in wireless sensor networks. The following results are derived from an indoor, Mica2 wireless sensor network testbed, which we describe in detail in Section 4. Every node generates data at a constant rate, which other sensors forward over a multihop network to a single sink. As the offered load increases, loss rates quickly increase. Figure 1 (*left*) shows network-wide loss rates for various offered loads. We separate losses due to wireless channel errors from losses caused by a lack of buffer space at a sensor node. We see that channel losses dominate buffer drops and increase quickly with offered load. This dramatic increase in loss rates is one of the two symptoms of congestion collapse.

The second symptom of congestion collapse is starvation of most of the network due to traffic from nodes one hop away from the sink. Figure 1 (*center*) illustrates this phenomenon. Given a percentage of packets p received from a given node at the sink, the complementary CDF plots the fraction of sensors that deliver at least p percent of their data to the sink. We see that as the offered load increases, a decreasing number of nodes get a disproportionately large portion of bandwidth.

Congestion collapse has dire consequences for energy efficiency in sensor networks, as Figure 1 (*right*) shows. When offered load increases past the point of congestion, fewer bits can be sent with the same amount of energy. The network wastes energy transmitting bits from the edge towards the sink, only to be dropped. We call this phenomenon *livelock*.

2.1 Metrics

Based on these quantitative observations, we propose a number of measures to evaluate the performance of sensor networks under congestion: *network efficiency* η, *node imbalance* ζ, *aggregate sink received throughput*, *network fairness* ϕ, and *median packet latency*.

We define *efficiency* as the number of hops "useful" packets travel, divided by the total number of packet transmissions in the network (Equation 1 in Table 1). A *useful* packet is any packet that eventually reaches a sink. Efficiency is important to measure because network bandwidth is limited, and energy is often a scarce resource in sensor networks. Motivated by the above energy result (Figure 1, *right*), this definition extends previous notions of link efficiency [29, §5.3] to a multi-hop sensor network setting. The denominator of the metric (total number of transmissions) includes all retransmissions, as well as transmissions associated with packets that are eventually dropped or corrupted.

Our efficiency metric penalizes failed transmissions, buffer drops, retransmissions due to lost acknowledgments, and channel losses. Efficiency penalizes a dropped packet to a greater extent if it travels a greater number of hops toward a sink. Efficiency therefore measures the *fraction of transmissions in a sensor network that contribute to a packet's*

Metric	Definition	Parameters
Efficiency	$\eta = \frac{\sum_{u \in U} \mathbf{hops}(u)}{\sum_{p \in P} \sum_{h \in \mathbf{hops}(p)} \mathbf{xmits}(p,h)}$ (1)	U is the set of useful packets, P is the set of all packets, $\mathbf{hops}(p)$ ranges over each hop packet p takes, and $\mathbf{xmits}(p,h)$ counts the number of transmissions packet p undergoes at hop h.
Fairness	$\phi = \frac{\left(\sum_{i=1}^{N} r_i\right)^2}{N \sum_{i=1}^{N} r_i^2}$. (2)	The average rate of packets delivered from the ith sensor is denoted r_i. N is the number of sensors in the network.
Imbalance	$\zeta = \frac{\text{packets received at } i}{\text{packets received at } i\text{'s parent from } i}$ (3)	This metric is defined per-node i; packet counts are taken over the entire experiment.

Table 1: A summary of the metrics we use to evaluate the performance of a sensor network.

eventual delivery at the sink. Efficiency also measures the fraction of transmissions whose energy is not wasted.

Related to efficiency is *imbalance*, a per-node measure of how well each node delivers its children's packets to its parent. We define the *imbalance ζ at node i* in Table 1. When $\zeta = 1$ for node N, N receives and transmits equal amounts of data from its children to its parent. A large ζ indicates that N receives more data from its children than it successfully transmits. Imbalance differs from a simple buffer drop count, because wireless drops directly influence it.

While efficiency and imbalance capture how efficiently a sensor network delivers data in terms of its use of transmission opportunities and energy, they do not measure the overall rate of data delivery. We therefore report aggregate and median per-node throughput, both measured at the sink.

Achieving fairness is desirable because in many sensing applications there is a decreasing marginal utility of increasing a sensor's report rate. In other words, it is often more important to hear a low rate of traffic from N sensors spread out across a geographical region than a high rate from one sensor. Achieving fairness in multi-hop wireless networks is difficult and often comes at the cost of reduced aggregate throughput [12, 15]. We measure fairness ϕ with the index [8] shown in Table 1.

Median packet latency is important to measure because many applications require that the time between sensing and reporting be minimal. Traffic overload in a sensor network often increases latency. It is important to note that while adding buffering and flow control helps to alleviate congestion, it also increases the queuing delay of packets at each hop, increasing end-to-end latency.

3. MITIGATING CONGESTION

Our congestion control scheme, which we call *Fusion*, integrates three techniques: hop-by-hop flow control, rate limiting, and a prioritized MAC. Hop-by-hop flow control is designed to prevent nodes from transmitting if their packets are only destined to be dropped due to insufficient space in output queues at downstream nodes. Rate limiting meters traffic being admitted into the network to prevent unfairness toward sources far from a sink. A prioritized MAC ensures that congested nodes receive prioritized access to the channel, allowing output queues to drain. While these techniques do not explicitly rely on topology information, we focus on single-sink, spanning-tree topologies in this paper.

We also note that the application of these techniques is difficult in the wireless domain. First, contention for the wireless channel occurs at both the sender and the receiver. Indoors, this contention is particularly problematic, since radio reflection makes propagation erratic and causes interference between two seemingly-disjoint sets of nodes. Second, there is a natural trade-off between channel utilization and fairness. By allowing nodes that provide transit for large amounts of traffic to transmit, we essentially allocate bandwidth to those nodes with the greatest contention. Finally, the wireless channel is inherently lossy, making distributed control of data flows even more challenging.

3.1 Hop-by-hop flow control

Hop-by-hop flow control has been proposed in wired local-area and wide-area networks [14, 16, 17], as well as in sensor networks [11, 22]. In our implementation, each sensor sets a *congestion* bit in the header of every outgoing packet. By taking advantage of the broadcast nature of the wireless medium, our implementation provides congestion feedback to all nodes in a radio neighborhood with every transmission. As a result, this implicit feedback obviates the need for explicit control messages that could use a large fraction of available bandwidth. Hop-by-hop flow control has two components: congestion detection and congestion mitigation. We first discuss two methods of detecting congestion, queue occupancy and channel sampling.

A simple way to detect congestion relies on monitoring a sensor's queue size: if the fraction of space available in the output queue falls below a high water mark α (in our implementation, $\alpha = 0.25$), the congestion bit of outgoing packets is set; otherwise the congestion bit is cleared. This technique, which we evaluate in Section 4 as *queue occupancy*, incurs little additional overhead.

CODA [22] proposes an alternate way to detect congestion. When a packet is waiting to be sent, the sensor samples the state of the channel at a fixed interval. Based on the number of times the channel is busy, it calculates a utilization factor. If utilization rises above a certain level, the congestion bit is set. Otherwise, the congestion bit is cleared.

This method, which we evaluate in Section 4 as *channel sampling*, requires the radio to continuously carrier sense the shared medium.

Congestion mitigation is the mechanism by which nodes in a given radio neighborhood throttle their transmissions to prevent queues at their next-hop node from overflowing. When a sensor overhears a packet from its parent with the congestion bit set, it stops forwarding data, allowing the parent to drain its queues. Without such a feedback mechanism, packet buffers could easily be overrun when a wave of traffic flows through the network. If a path experiences persistent congestion, hop-by-hop backpressure will eventually reach the source, allowing application-level flow control (described later in Section 3.4), to throttle the source rate.

One problem that arises when backpressure needs to propagate through multiple hops relates to the communication of congestion state. When a parent sets its congestion bit, its children stop transmitting. This prevents the children from informing their own children when they become congested. We fix this by allowing a congested node to send out one packet once it has heard a packet from its parent with the congestion bit set. A congested node may also send one additional packet per received packet, to compensate for children not hearing a packet that indicates congestion.

3.2 Rate limiting

Due to the variability of channel conditions and workloads, points of congestion can occur anywhere in the network. These points of congestion usually result in an increase in the noise floor accompanied by a precipitous drop in the packet delivery rate. As network diameter grows in size, it becomes particularly problematic if transit traffic is dropped due to congestion, because the network has expended a significant amount of energy and bandwidth transmitting the packet over many hops (a problem referred to as livelock). Moreover, there is a natural tendency for the network to deliver traffic originating close to a sink at the expense of traffic sourced deeper inside the network. This rise in the noise floor and increase in unfairness are precisely the properties that source rate limiting addresses.

The rate limiting scheme we evaluate works as follows. Note that we make the simplifying assumption that all sensors offer the same traffic load and that the routing tree is not significantly skewed. A more general approach that better handles variable rates would require nodes to propagate their rates. For simplicity, we utilize a completely passive approach that relies on monitoring transit traffic to determine source rates. Each sensor listens to the traffic its parent forwards to estimate N, the total number of unique sources routing through the parent. We then use a token bucket scheme to regulate each sensor's send rate. A sensor accumulates one token every time it hears its parent forward N packets, up to a maximum number of tokens. The sensor is allowed to send only when its token count is above zero, and each send costs one token. This approach rate-limits the sensor to send at the same rate of each of its descendants.

We evaluate this simple rate limiting scheme in Section 4 both in isolation, and in concert with other congestion control mechanisms.

3.3 The MAC layer

Although sensors can react to congestion using the above network-layer mechanisms, they cannot always react to congestion fast enough to prevent buffer losses without some help from the MAC layer. A carrier sense multiple access (CSMA) MAC can aid congestion control.

A standard CSMA MAC layer gives all sensors competing to transmit an equal chance of success. However, during times of congestion, this approach can lead to reduced performance due to a congested sensor's inability to quickly propagate congestion control feedback to its neighbors. For example, consider a high fan-in scenario, where several sensors are forwarding through a common parent. On average, the parent sensor will gain access to the channel only after half its neighbors have transmitted. However, since the parent is congested, it may not have enough buffer space available to store packets from its children. Hence the parent has no choice but to drop packets that its children forward it. Consequently, it is imperative that congested sensors have *prioritized access* to the wireless medium.

In order to address this issue, we adopt a technique that Aad and Castelluccia advocate [1], making the length of each sensor's randomized backoff (before every transmit cycle) a function of its local congestion state. If a sensor is congested, its backoff window is one-fourth the size of a non-congested sensor's backoff window, making it more likely that a congested sensor will win the contention period, allowing queues to drain and increasing the likelihood congestion control information will propagate throughout a sensor's neighborhood.

3.3.1 The hidden terminal problem

Hidden terminals occur when two senders that are not in radio range transmit to a common receiver. One way of reducing collisions between hidden terminals is to exchange RTS/CTS control packets before communicating. Although these control packets can collide, and some non-colliding transmissions may be stopped, the RTS/CTS exchange eliminates most data packet collisions. The added cost of the RTS/CTS exchange is worthwhile when data packets are substantially larger than control packets. However, in sensor networks, data packets are usually small [6], and on some platforms the RTS/CTS exchange would incur a 40% overhead [24]. Consequently, we do not evaluate this mechanism.

Woo and Culler propose an interesting strategy to alleviate hidden terminals [24] in tree-based topologies. When a node overhears its parent finish sending a packet, it waits for one packet-time plus a guard time, to avoid a likely hidden terminal collision with its grandparent. We evaluate this *delay* strategy with our other congestion control strategies in Section 4.

3.4 Application adaptation

Applications play an important role in preventing congestion. When the networking stack is not ready to accept additional data, it signals applications via send failures. It is then up to the application to respond appropriately. Some applications will simply wait until the stack is ready again (the strategy we evaluate). Others may adjust their send rate via an AIMD controller or similar mechanism. Generally, applications will only allow small numbers of packets outstanding in the networking stack at once. Doing so prevents locally-generated traffic from starving route-through traffic.

4. EXPERIMENTAL EVALUATION

In this section we evaluate our congestion control suite in a 55-node indoor wireless sensor network testbed. Each node is a Crossbow Mica2, which has an Atmel ATmega128L microcontroller with 4 KB of RAM, 128 KB of flash, and a CC1000 radio. The radio operates at 433 MHz, transmits at 38.4 Kbps, and uses Manchester encoding. Each node was attached to a Crossbow MIB600 interface board that provides both power and an Ethernet backchannel for programming and data collection. As Figures 5 and 6 show, we deployed nodes over an area of 16,076 square feet on one floor of our office building, with liberal coverage throughout the floor and a higher than average density in the top-left corner of the floor. We use Motelab [23] to manage the testbed.

Characterizing the size of each node's neighborhood is difficult because of the vagaries of radio propagation. To characterize neighborhood size, we measure channel quality between all pairs of nodes in an unloaded network for several different transmit power levels. Note this measurement is performed without a routing protocol nor any other network stack modifications. One-by-one, each node sends a train of broadcast probe packets (without link-level retransmissions). We define the size of a node's neighborhood $N(x)$ to be the expected number of nodes that will hear any given transmission. This value can be calculated using Equation 4, where x is any node in the network, $\mathcal{N}$ is the set of all nodes, and p_{xy} is the probability node y successfully receives node x's transmission.

$$N(x) = \sum_{y \in \mathcal{N} - \{x\}} p_{xy} \tag{4}$$

Figure 2 shows node neighborhood sizes in our testbed. Note that the average neighborhood size increases linearly with an exponential increase in power. For our experiments, we selected a transmit power level of -10 dBm, which is significantly lower than the default power level of 0 dBm. Our goal was to reduce radio power and increase spatial reuse, while maintaining a connected network.

We summarize each congestion control strategy in Table 2, providing experimental parameters for each. For every strategy except no congestion control, we use the MAC with prioritization features described in Section 3.3. For the no congestion control strategy, we use the unmodified MAC included with TinyOS (B-MAC). We evaluate each strategy under three workloads: a periodic workload, a high fan-in workload, and a correlated-event workload. We evaluate each technique using the metrics described in Section 2.

The sensors in our testbed run TinyOS 1.1, augmented with our own single-destination DSDV [18] routing protocol. Each sensor monitors the channel quality from its neighbors by overhearing their transmissions, and selects routes that minimize the expected number of transmissions (ETX [4]) required to deliver data to destinations. Each sensor uses an eight packet queue to buffer outgoing traffic. The MAC used in the radio stack is derived from the base TinyOS distribution, with modifications as described in Section 3.3. Our link-layer sends data semi-reliably with a maximum retransmission count of three.

All packets sent through the wireless sensor network are 36 bytes in length. The only other traffic running through the network during the experiments is infrequent routing updates, which are sent every ten seconds. In addition, we set the radio frequency to reduce the amount of outside interference from other wireless sensor networks in the building. We allow routes to stabilize for two minutes prior to the data collection phase of our experiments, and we pin routes for the duration of the experiment once the stabilization phase completes. Pinning routes ensures that the routing protocol does not influence the outcome of our experiments.

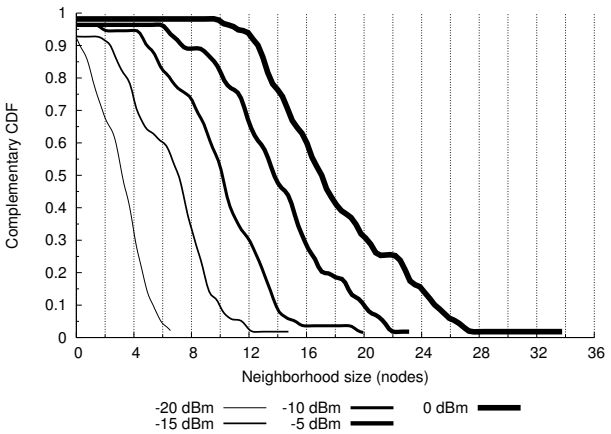

Figure 2: Neighborhood size (complementary CDF) as computed from Equation 4, as a function of radio power level, for a 55-node indoor wireless sensor network testbed deployed over an area of 16,076 square feet on one floor of an office building.

For each experimental data point in the periodic and correlated-event workload results, we report an average and confidence interval over 15 runs to one sink (node 6). Our high fan-in results average over 5 runs. We combine runs taken during different times of the day and on different days of the week. The traffic statistics collection phase of each run lasts four minutes for the periodic and high fan-in workloads, and one minute for the event experiment.

We evaluate all metrics as per-sensor offered load ranges from 0.25 packets per second to four packets per second for periodic and high fan-in workloads, and as event size ranges from one to eight packets for the correlated-event workload. Since the link-level throughput for the Mica2 caps at approximately 40 packets per second and our network has 55 sensors, we can be sure that our network becomes congested at four packets per second.

4.1 Periodic workload

The *periodic workload* models a typical monitoring sensor network in which readings are generated at fixed time intervals. Deployments exhibiting this traffic pattern are quite common in practice [7, 13, 20]. In this workload, each sensor sources traffic at some offered load, and helps to forward other sensors' traffic to a sink. To avoid synchronizing periodic reports from different sensors, we introduce a random delay, which is large relative to the report rate, at the beginning of the experiment. Figure 3 shows a snapshot of the routing topology used to forward packets toward the sink.

We note here that it is deceptively difficult to provision a wireless sensor network to obviate the need for an congestion control, even under this simple workload. Even though a network designer knows the aggregate offered load *a priori*,

Strategy	Remarks	Parameters
Queue occupancy	Hop-by-hop flow control using queue occupancy (described in Section 3.1; α indicates the fractional queue occupancy at which congestion is indicated.	Queue size of eight, $\alpha = 0.25$.
Channel sampling	Hop-by-hop flow control using channel sampling (described in Section 3.1). N indicates the number of epochs of length E that the channel is sensed for; α indicates the EWMA averaging parameter.	$N = 4$, $E = 100$ milliseconds, $\alpha = 0.85$.
Delay	The queue occupancy strategy augmented with the *delay* technique as described in Section 3.3.1. After overhearing the end of a parent's transmission, we backoff for τ milliseconds, a bit more than one packet-time on the Mica2's CC1000 radio [3].	$\tau = 40$ milliseconds.
Rate limiting	We implement a simple *rate limiting* strategy as described in Section 3.2.	None.
Fusion	This strategy simultaneously combines the queue occupancy, forwarding delay, and rate limiting algorithms.	As above.
No congestion control	No congestion control related changes are made to the network layer or MAC. Transmission is attempted as soon as data enters the outgoing queue (after carrier sense and MAC backoff).	None.

Table 2: Summary of congestion control strategies evaluated in this paper and their relevant parameters.

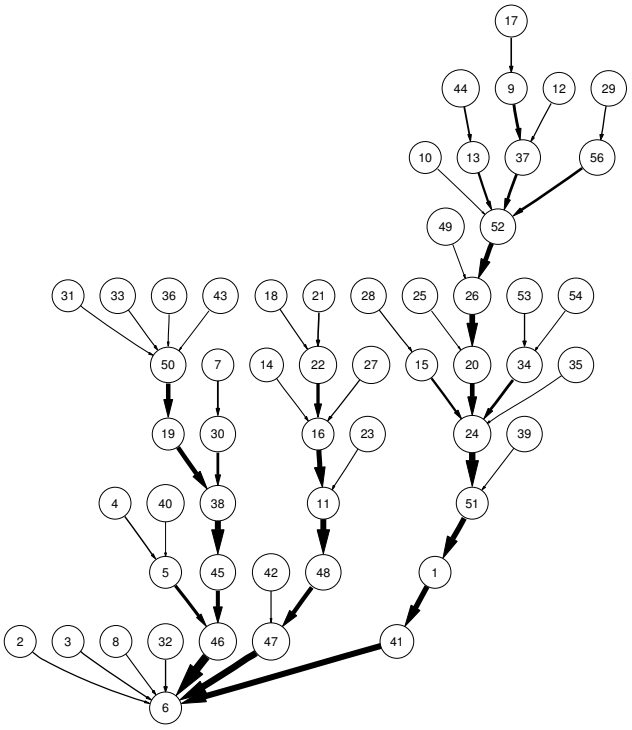

Figure 3: Routing topology in one run of an experiment with Fusion flow control and each node offering one packet per second, as formed by the ETX path selection metric. The thickness of each edge is proportional to the number of packets the node at the head of the edge received from the node at the tail of the edge.

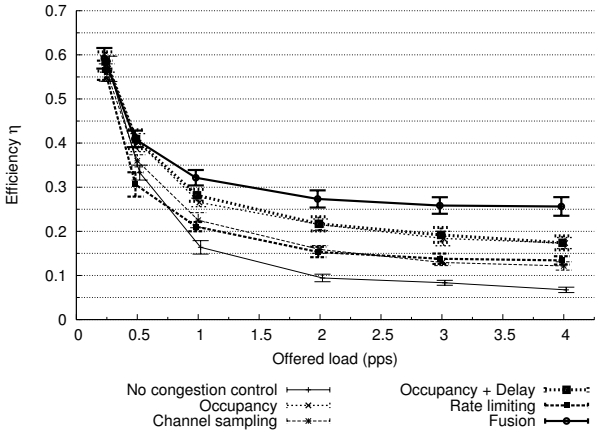

Figure 4: Average network efficiency η versus per-sensor offered load under a periodic workload. 99% confidence intervals are shown.

the unpredictable nature of radio makes it is almost impossible to know the actual forwarding topology of the network, making fine-grained channel bandwidth allocation difficult.

4.1.1 Periodic workload: Network efficiency

Figure 4 shows how network efficiency η (defined in Equation 1) varies with per-sensor offered load for each congestion control strategy we examine. First, note the decreasing trend for each congestion control strategy. This trend is expected, because as the number of transmissions increases, the noise floor of the network rises, increasing the probability of packet corruption and retransmission. Additionally, the probability of collisions due to MAC contention and hidden terminals increases. These factors drive η down as offered load grows.

Rate limiting offers an incremental improvement in efficiency for the same reason that the efficiency vs. offered load trend is downward: fewer packet transmissions. In particular, rate limiting reduces the offered load near the sink of the network, where congestion and contention are worst. However, as the offered load increases, hidden terminals and interference prevent this strategy from exceeding the efficiencies of the other strategies.

Hop-by-hop flow control offers an additional improvement in efficiency, but succeeds for a different reason than rate limiting. Rather than reducing contention between the transit and leaf nodes of the network, hop-by-hop flow control improves efficiency by throttling transmissions based on queue contention in the local neighborhood. In addition, as offered load increases, queue occupancy congestion detection consistently outperforms channel sampling. This suggests that queue occupancy is at least as good as channel sampling as an indicator of congestion.

Combining these two flow control strategies in the Fusion scheme yields the highest gains in efficiency. Hop-by-hop flow control helps to throttle transmissions at every link in the network, while the rate limiting mechanism meters traffic being admitted into the network. Together, these two strategies complement each other, achieving a high level of efficiency *even after* the network reaches saturation.

4.1.2 Periodic workload: Imbalance

With an offered load of four packets per second, we plot distributions of node imbalance ζ for different congestion control strategies in Figure 7. These results summarize node imbalances over multiple runs of the periodic experiment.

Without any congestion control strategy, the network has many hotspots: approximately five nodes (the 90th percentile) have an imbalance greater than 50. Furthermore, the tail of the imbalance CDF without congestion control is very heavy, indicating the presence of pathological hotspots in the network that are not successfully forwarding any traffic they receive. Rate limiting also exhibits a heavy tail, indicating that rate limiting does little to help these pathological nodes.

Channel sampling and occupancy-based congestion control both remove pathological hotspots from the network. We see a marked synergistic improvement when we combine the congestion control strategies in the Fusion scheme.

Figures 5 and 6 show the amount of traffic received by each node in one experimental run. The figures show a periodic workload of four packets per second using Fusion and no congestion control strategies, respectively. The thickness of each link is proportional to the number of packets the node at the head of the edge receives from the node at the tail of the edge.

In Figure 5, note that the relatively thick edges in the graph form a forwarding backbone over which most traffic is sent. The rate limiter helps to shape traffic such that leaf nodes, especially those near the sink, do not overload the network. In addition, a conservation of packets property holds: for nodes that route traffic, several thin incoming links usually result in a much thicker outgoing link.

In contrast, in Figure 6 (with no congestion control) there is no clear forwarding backbone, nor does the conservation of packets property hold. For example, one or more thick edges entering a node lead to one thin edge exiting the same node. This implies a large number of buffer or wireless drops, and

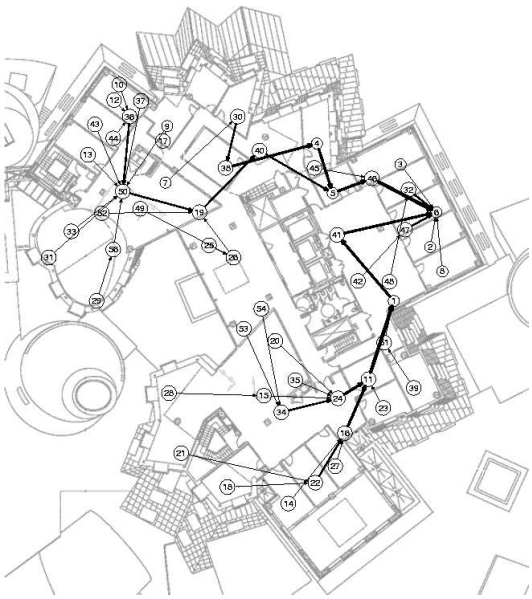

Figure 5: Traffic flow in one run of a Fusion congestion controlled experiment with each node offering four packets per second. The thickness of each edge is proportional to the number of packets the node at the head of the edge receives from the node at the tail of the edge.

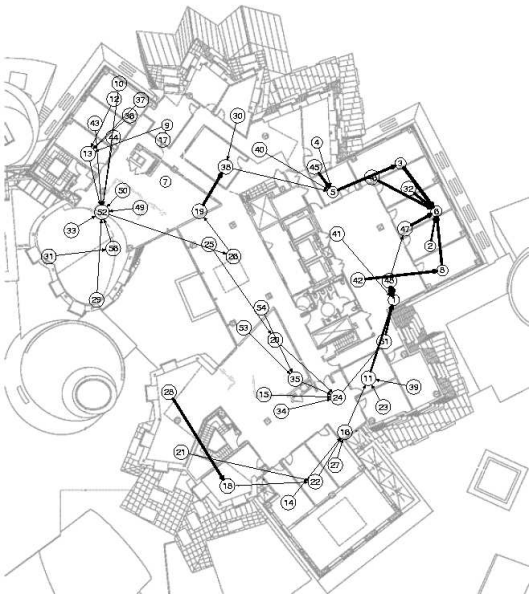

Figure 6: Traffic flow in one run of an experiment with no congestion control and each node offering four packets per second. The thickness of each edge is proportional to the number of packets the node at the head of the edge receives from the node at the tail of the edge.

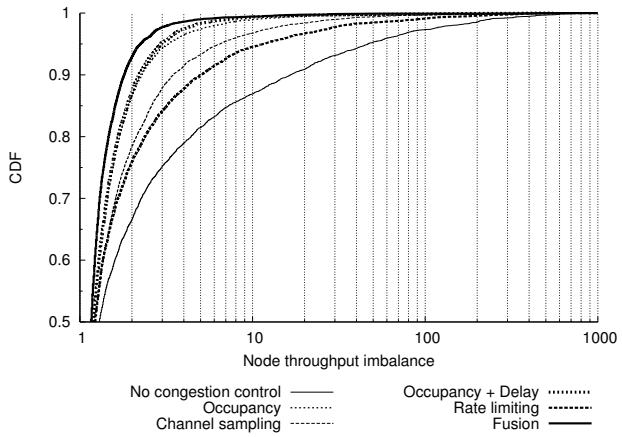

Figure 7: Node imbalance ζ (CDF) for different flow control strategies. Each node offers four packets per second.

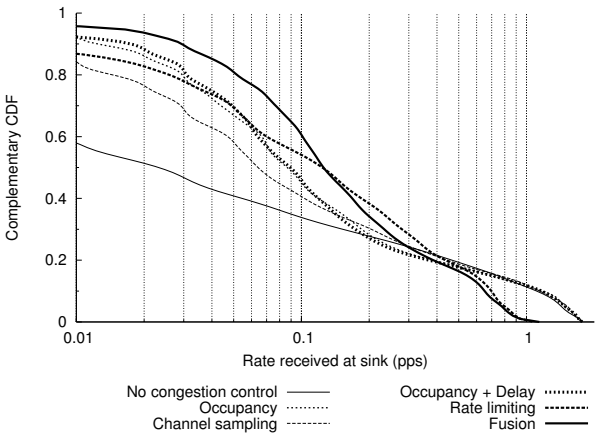

Figure 9: Per-node throughput received at the SAP (complementary CDF) under a periodic workload. Each node offers two packets per second.

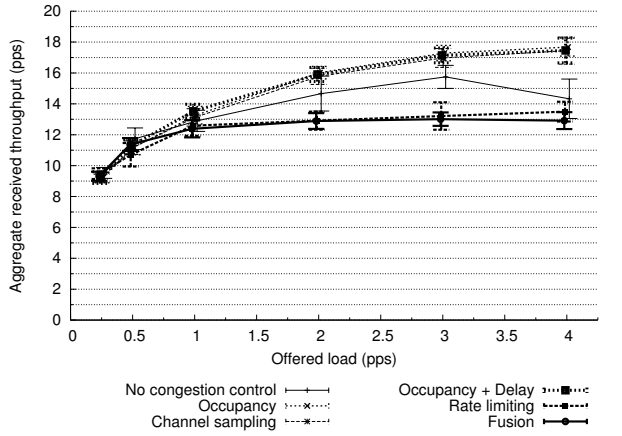

Figure 8: Average aggregate received throughput (measured at the sink) versus per-sensor offered load under a periodic workload. 99% confidence intervals are shown.

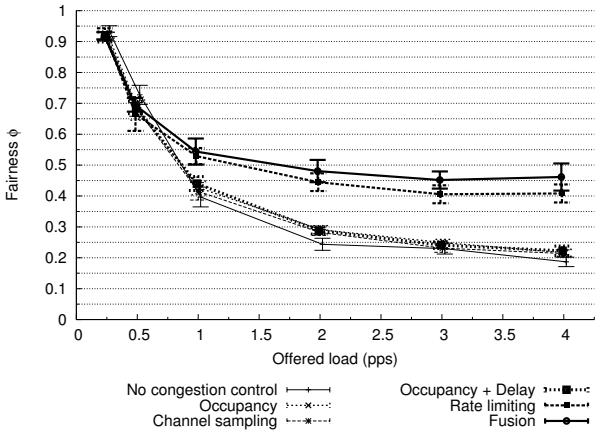

Figure 10: Average fairness ϕ versus per-sensor offered load under a periodic workload. 99% confidence intervals are shown.

explains the pathological hotspots we see in Figure 7. Also note the lack of thick edges in the densely-populated upper-left hand corner of the map. This suggests a large number of wireless collisions in that geographic region.

4.1.3 Periodic workload: Throughput and fairness

Next, we measure the aggregate received throughput at the sink, without regard to which nodes deliver the data. Figure 8 shows that as offered load increases, using a non-rate limiting congestion control strategy results in the highest throughput. This follows because while rate limiting can slow nodes down, so can network inefficiencies.

The throughput trend is of secondary importance, however, since fairness decreases substantially without congestion control. Figure 9 shows the distribution of throughputs that the sink receives from each node at an offered load of two packets per second. Note that without congestion control, more than 40% of the nodes deliver less than one packet every 100 seconds, making that part of the network almost useless. While congestion control sacrifices the high throughput of a minority of nodes, it distributes bandwidth between nodes more fairly.

Table 3 shows the report period that a given percentage of nodes in the network can achieve when every sensor offers four packets per second. In the second column, we see that without any congestion control strategy, no throughput guarantees can be made about one quarter of the nodes. In contrast, nodes using an occupancy congestion control strategy can cover 90% of the network. If, however, we are only interested in the throughput 10% of the nodes in the network can achieve, no congestion control is the best strategy. This regime, however, is unlikely to be of much interest to sensor network designers, because 10% of the sensors would provide poor network coverage, particularly because these nodes are the ones closest to the sink.

As we vary offered load between 0.25 and four packets per second, we see the same trends for aggregate throughput. Without any form of congestion control, aggregate throughput increases as show in Figure 8. However, the network mostly delivers data from nodes one hop away from the sink, resulting in a decrease in fairness as shown in the right hand

	Maximum period					
Coverage	No congestion control	Occupancy	Channel sampling	Occupancy + Delay	Rate limiting	**Fusion**
100%	∞	∞	∞	∞	∞	∞
95	∞	108 s	∞	123 s	∞	**43 s**
90	∞	62 s	100 s	71 s	250 s	**35 s**
75	∞	29 s	38 s	33 s	31 s	**15 s**
50	38 s	12 s	13 s	12 s	8.5 s	**7.2 s**
25	4.2 s	3.4 s	3.8 s	3.2 s	2.9 s	**3.0 s**
10	840 ms	870 ms	980 ms	950 ms	1.4 s	**1.5 s**

Table 3: Network coverage by congestion control strategy. Given that a network designer wants the network coverage shown in the left hand column, that percentage of sensors operating under different congestion control strategies can achieve at least the report rate show in the rightmost columns. The offered load in the network is periodic at two packets per second per sensor.

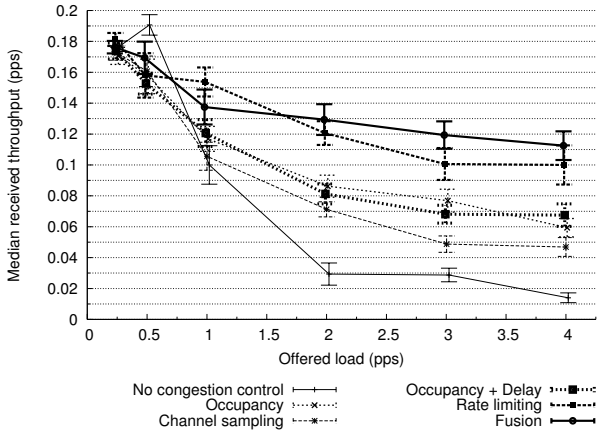

Figure 11: Median throughput as a function of offered load under a periodic workload. 99% confidence intervals are shown.

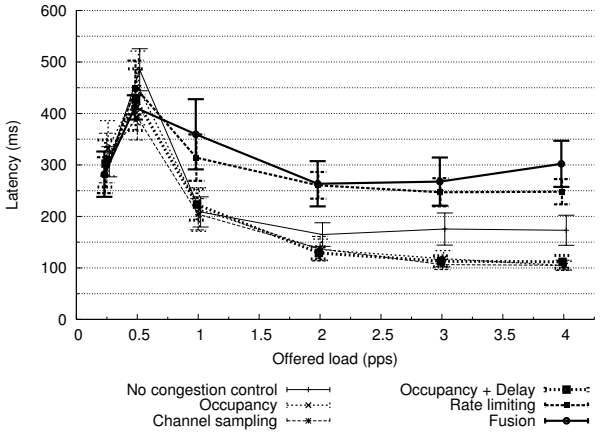

Figure 12: Median packet latency as a function of offered load under a periodic workload. 95% confidence intervals are shown.

side of Figure 10. Without a rate control policy, congestion control mechanisms suffer a similar fate, because nodes in the core of the network have more opportunities to inject packets into the network, since they have fewer hops over which hop-by-hop backpressure can act. Rate-limiting dramatically improves fairness because space opens up in node transmission queues for traffic from the edges of the network.

Figure 11 shows median node throughput as a function of per-node offered load. Below an offered load of 0.5 packets per second, the network is in an underloaded state, and median received throughput increases with offered load. Above one packet per second, sensors need a congestion control strategy if more than half can provide any traffic at all. This result quantifies the high degree of unfairness that sensors at the edges experience when the network is an a state of congestion collapse. At least half the nodes running the Fusion strategy are able to maintain at least 0.1 packets per second as offered load increases, because rate control prevents the core of the network from overwhelming the edges.

4.1.4 Periodic workload: Latency

Figure 12 shows how the median packet latency varies with offered load. We measure latency from the application-level transmit on the sensor to the moment at which the sink receives the packet. Since we only measure the latency of packets received at the sink, this metric must be viewed with fairness in mind. The primary source of latency is queuing delay, which increases as the number of hops a packet must traverse grows. As we increase offered load from zero to 0.5 packets per second, queues start to build and latency increase. Beyond an offered load of one packet per second for all strategies except Fusion, fairness declines sharply, since the edge's packets are dropped. Thus latency decreases with increasing offered load, since the sensors that get through are in the core of the network, closer to the sink. Since the Fusion strategy is the fairest, as offered load increases, a greater proportion of the packets received are from the edges (many hops away) and consequently latency is higher.

4.1.5 Periodic workload: Sources of loss

Figure 13 shows network-wide wireless channel loss as a function of offered load. We calculate loss rate by dividing the sum transmission count (including retransmissions) by the sum reception count. As expected, the wireless loss rate increases for all strategies as the offered load increases. This trend is caused by rise in the noise floor, and possibly an increase in the number of collisions due to hidden terminals.

The no congestion control strategy suffers from the highest loss rates, which approach more than 80% at four packets per second. Channel sampling and rate limiting occupy the middle ground, with loss rates approaching 60% and 70%, respectively. The occupancy-based congestion con-

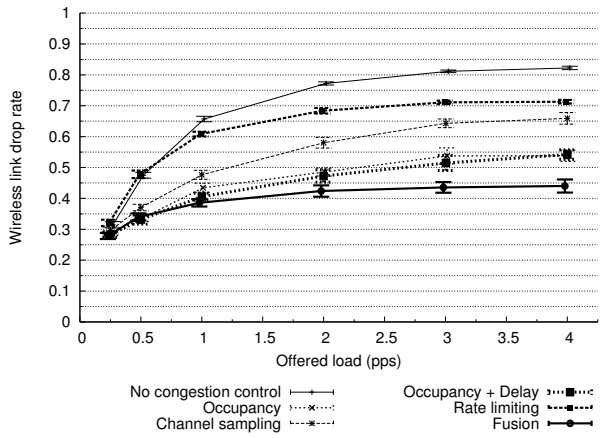

Figure 13: Average network-wide wireless link drop rate (before retransmissions) versus per-sensor offered load under a periodic workload. 99% confidence intervals are shown.

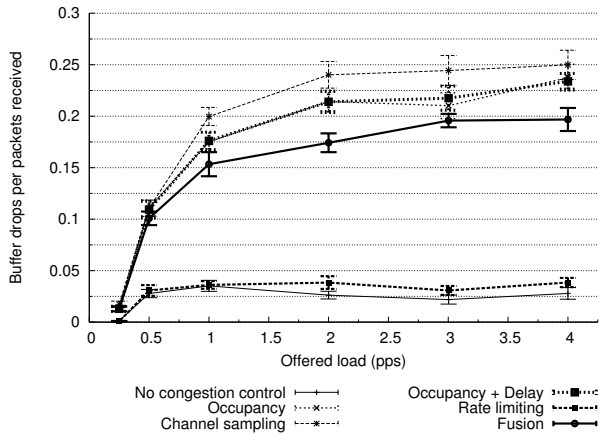

Figure 14: Total network-wide buffer drops per network-wide received packets as a function of per-sensor offered load under a periodic workload. 99% confidence intervals are shown.

trol strategies perform even better, keeping loss rates under about 50%. The Fusion scheme performs the best overall, with close to a 50% decrease in drops at four packets per second. Surprisingly, there is no statistical difference in wireless drop rate between the occupancy + delay strategy and the occupancy strategy. This observation suggests that either hidden terminals are not the primary problem, or that the delay strategy does not often avoid them.

Figure 14 shows total network-wide buffer drops per packets received, as a function of offered load. This measures the probability that a packet will be dropped due to buffer overflow, given that it was successfully received. Surprisingly, the buffer drop rate is substantially higher for strategies that include hop-by-hop flow control. This trend is a direct result of the substantially higher wireless loss rates of the strategies that do not include hop-by-hop flow control, no congestion control and rate limiting. With these later strategies, nodes do not receive enough packets to fill up their forwarding queues and trigger buffer drops.

4.1.6 Periodic workload: Energy considerations

Performance under a power-saving workload is important to low-power monitoring applications where a large number of sensors periodically sleep (forgetting their congestion state) in between low-rate packet transmissions. There are a wide variety of power-saving strategies proposed for sensor networks. CSMA-based sleep-wakeup algorithms [2, 26] elect a dynamic "backbone" of nodes that forwards traffic for the other power-saving nodes. Other proposals synchronize nodes' sleep-wakeup cycles [27, 21]. In TDMA-based sensor networks, data sources can sleep in all inactive time slots. For any of the above types of networks, the key challenge is designing and evaluating congestion control algorithms that can function soon after the node wakes up.

The congestion control techniques presented in this paper rely on channel overhearing to propagate congestion state. When combined with power-saving, it is not immediately clear how the reduction in overhearing time resulting from various sleep intervals will impact congestion control performance. To measure this impact, we constructed a topology and strategy that simulates the effects of the sleep-wakeup cycle. In this power-saving workload, a core set of nodes is selected to forward traffic. The remaining leaf nodes send traffic at a low data rate (one packet per second), sleeping for a given fraction of the period. After waking up, nodes forget their congestion state, listen for the remaining fraction of the period, then send.

Our results (omitted for brevity) for varying listen/sleep intervals indicate that sleep-wakeup power-saving strategies have a statistically insignificant impact on network performance. In particular, efficiency is dependent on offered rate, but not listen period. Unlike many protocols that rely on overhearing, hop-by-hop flow control is not cumulative: nodes are only concerned with the most recently propagated congestion state. Additionally, in deep multihop networks, transit nodes—which are on continuously and can fully benefit from congestion control—transmit the majority of packets.

4.2 High fan-in network experiments

Our high fan-in experiments evaluate congestion control in a wireless network where mitigating congestion is very difficult. We choose only a small subset of the nodes to perform the task of routing traffic in the network. Nodes still choose routes using the ETX metric, but only ten nodes out of 55 advertise routes to the sink. Figure 15 shows a representative routing topology used in one run of our experiments. Non-uniform deployments—where a high degree of fan-in is expected—motivate this workload. Heterogeneous sensor capabilities, such as differences in processor, memory, and power ability, can also motivate this choice of topology.

Note that in comparison with the topology formed in Figure 3 under the periodic workload, this topology has a higher degree of fan-in and a smaller network diameter. The high fan-in makes hop-by-hop congestion control more difficult, since each child must each receive congestion feedback before the aggregate demand on a parent decreases.

4.2.1 High fan-in network: Network efficiency

Figure 16 shows network efficiency versus per-sensor offered load as offered load ranges from 0.25 to four packets per second. Comparing Figure 16 with Figure 4 (network ef-

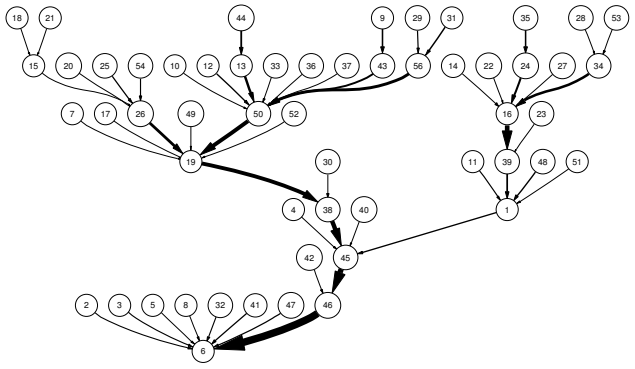

Figure 15: Routing topology in one run of a high fan-in experiment with Fusion congestion control and each node offering one packet per second, as formed by the ETX path selection metric, restricted to gateway nodes. The thickness of each edge is proportional to the number of packets the node at the head of the edge received from the node at the tail of the edge.

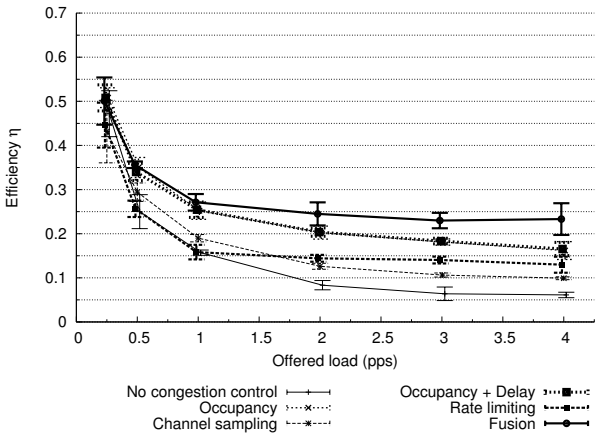

Figure 16: Average network efficiency η versus per-sensor offered load in a high fan-in network. 95% confidence intervals are shown. The curves for Occupancy and Occupancy + Delay overlap.

ficiency under the periodic workload), we make the following observations.

Even at low offered load, network efficiency in the high fan-in network is lower than network efficiency in the periodic network. It is unlikely that poor link selection in the high fan-in network causes the lowered efficiency, because the wireless link drop rates for both the high fan-in and normal networks are equal at low transmission rates. The likely explanation, therefore, is wireless contention and network congestion at high fan-in nodes.

As offered load increases, the trend continues, with efficiency in the fan-in network marginally lower than in the normal network. Note that in a high fan-in topology, congestion control techniques work together to increase performance. Fusion outperforms all strategies at most offered loads.

4.3 Correlated-event workload

The *correlated-event workload* captures the effect of spatially-correlated events in a sensor network. Detection and tracking applications often use an event-oriented reporting model rather than sending a periodic stream of readings. In particular, this workload models the effects of a single, synchronized impulse of traffic on network performance.

Instead of collecting analog data from each sensor in our testbed, we model an event-based workload using the following methodology. At the beginning of the experiment, we run a distributed level-aware, sender-receiver based time synchronization protocol similar to the Timesync Protocol for Sensor Networks (TPSN) [5]. Using an Ethernet backchannel, we verified synchronization of 91% of the testbed to within 29 ms (less than one packet-time) and all 55 nodes in our testbed to within 56 ms (between one and two packet-times).

After the synchronization phase, all sensors in the testbed simultaneously send B packets back-to-back (or, if the channel is carrier-sensed busy, as fast as possible) at the scheduled time of each event. There is sufficient time (20 seconds) between each event to let traffic drain completely from the network.

We evaluate this correlated-event workload by varying the traffic burst size B. It is important to note that for this workload, which consists of a single impulse of traffic, rate limiting provides no improvement in network performance. Our rate limiting algorithm is designed to operate when there is significant steady-state traffic present throughout the network. As a result, we omit the rate limiting and Fusion schemes from our data sets for clarity and investigate in detail performance of the hop-by-hop variants.

4.3.1 Correlated-event workload: Network efficiency

Even at small event sizes, the strong time correlation between events necessitates some kind of congestion control. This trend stands in contrast to that of the periodic and high fan-in workloads in which the benefits of congestion control were mainly seen at higher offered loads. Figure 17 shows network efficiency as each sensor's traffic burst size B varies between one and eight packets. As expected, a downward trend still exists for all strategies as network load increases. However, even at an event size of one, the occupancy + delay strategy yields close to an 80% gain in network efficiency over no congestion control. As B increases, the relative increase in network efficiency (versus no congestion control) increases for all strategies. At an event size of eight packets, the best strategy is approximately two times better than the baseline.

There is a clear benefit to using hop-by-hop flow control for a correlated-event workload to combat congestion caused by the wave of data flowing from the leaves to the sink.

Buffer occupancy augmented with a forwarding delay for hidden terminal reduction (occupancy + delay) performs better than all other strategies at small B. Without the delay strategy, synchronization in the correlated-event workload makes collisions much more likely. The forwarding delay allows a node's grandparent, which normally would exist as a hidden terminal and be prone to collision, a larger window to successfully complete its transmission. At larger values of B, the increase in network efficiency due to the

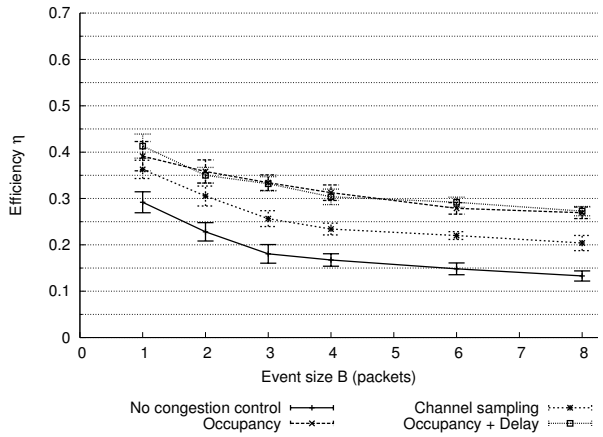

Figure 17: Average network efficiency η as a function of event size B under a correlated-event workload. 99% confidence intervals are shown.

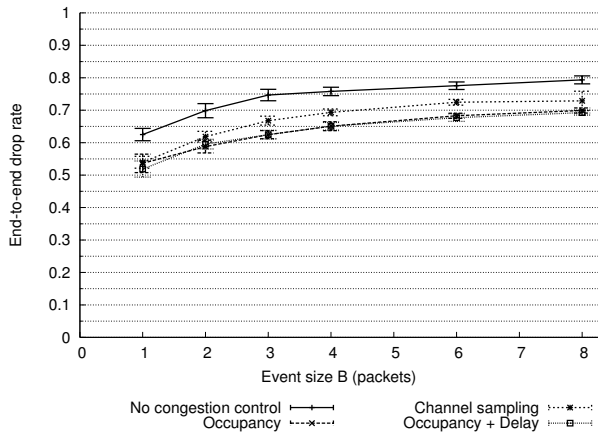

Figure 18: End-to-end drop rate as a function of event size B under a correlated-event workload. 99% confidence intervals are shown.

forwarding delay is negligible. For most event sizes, channel sampling performs worse than all strategies except for no congestion control.

4.3.2 Correlated-event workload: Drop rate

For many types of applications designed to detect discrete, non-repeating events, the end-to-end packet drop rate is an important measure of performance. This stands in contrast to the periodic workload where often it is reasonable to assume that subsequent reports will supersede any lost data. Figure 18 shows the end-to-end drop rate as a function of event size B for various congestion control strategies. Note how all strategies perform better than no congestion control. In some cases, the lack of any congestion control can increase the end-to-end drop rate by almost 35%. Hop-by-hop flow control alleviates congestion in this workload because the backpressure desynchronizes packet transmissions.

4.3.3 Correlated-event workload: Latency

In Figure 19 we note that packet latency rises when using any of the congestion control strategies. This increase is to

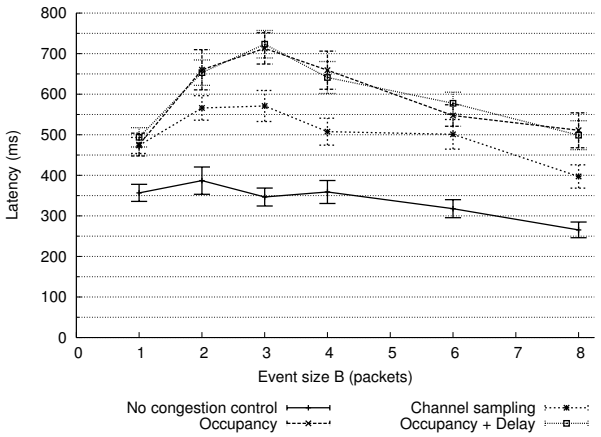

Figure 19: Median packet latency as a function of event size B under a correlated-event workload. 95% confidence intervals are shown.

be expected because all congestion control strategies operate by delaying transmissions. By decreasing the rate at which queues drain, wireless contention and collisions are reduced at the cost of increased queuing delays. For small event sizes, the increase in latency is in the neighborhood of 40%. However, the maximum increase in latency is seen when the event size is three, resulting in a latency increase of over 100% when compared to no congestion control.

Whether or not these increases in latency are actually meaningful depends on the application. It is important to note that latencies can by significantly reduced by decreasing the size of the forwarding queues, at the cost of increased losses when burst sizes are large.

5. RELATED WORK

Wan *et al.* propose CODA [22], a congestion control system for sensor networks. CODA detects congestion by periodically sampling the channel load and comparing the fraction of time that the channel is busy to the optimal channel utilization. The system responds to congestion with a combination of hop-by-hop flow control and closed-loop regulation. In our work, we experimentally evaluate CODA's congestion detection mechanism (channel sampling) and one of its congestion mitigation mechanisms (hop-by-hop flow control). We make all our comparisons in a large sensor network testbed, expanding on previous small-scale testbed or simulation-based congestion control studies. We find that when used alone, channel sampling-based congestion detection performs worse than queue occupancy-based congestion detection. We also find that augmenting a hop-by-hop flow control mechanism (such as CODA) with rate limiting is beneficial.

Woo and Culler propose a rate control mechanism [24] that admits traffic into the network using an AIMD controller. When a node hears that a packet it had previously sent was forwarded, it additively increases its transmission rate. When it does not hear a previous transmission being successfully forwarded (presumably after a timeout), it multiplicatively reduces its transmission rate. We evaluated a similarly motivated rate control mechanism. We found that rate limiting increases fairness, but its benefits to the net-

work (as measured by our metrics from Section 2) are most significant when used in combination with other congestion control techniques.

Lu *et al.* propose RAP [11], a real-time communication protocol for sensor networks. The network layer of RAP ensures fairness between nodes and improves the ratio of packets that make their latency deadlines. To accomplish this task, RAP provides support for deadline- and distance-aware packet scheduling. Packets originating from sources deeper in the network have higher priority than packets originating from sources close to the sink. While RAP focuses on the network's ability to meet deadlines, our work focuses on managing overload and congestion in a sensor network.

Sankarasubramaniam *et al.* propose ESRT [19], the Event to Sink Reliable Transport Protocol. Their system addresses congestion control in the context of reliable delivery. ESRT keeps a network operating near its optimal load by broadcasting one-hop control messages to sources from the sink. The consequent assumption is that a data sink can reach all sources via a high-powered one-hop broadcast, which reduces overall network capacity. In contrast, our hop-by-hop flow control does not require a high-powered broadcast message to be sent by a sink.

Lemmon *et al.* study overload in sensor-actuator networks connected by a wired bus [10]. The key difference between their overload problem from ours is that the communication network they consider is a shared bus, with no potential for spatial reuse. Additionally, their sensor nodes do not forward each other's traffic.

Zhao and Govindan conduct a comprehensive study of packet delivery performance in wireless sensor networks [29]. Their study focuses on the physical and link layers, evaluating packet loss, packet loss correlations, and link asymmetry. Our study of congestion complements their work, studying end-to-end performance when sensors participate in multihop routing and congestion avoidance protocols. Our congestion control algorithms operate in a network with a wide range of link loss rates and asymmetries, as in their work.

Woo *et al.* examine routing in sensor networks [25], studying link estimation and neighborhood table management in particular. We use these mechanisms in our network layer implementation to support our congestion control algorithms.

Yi and Shakkottai propose a fair hop-by-hop congestion control algorithm for multihop wireless networks [28]. They build a theoretical model and provide a simulation-based evaluation of their distributed algorithms. They make the assumption that simultaneous transmissions can occur over links in the same radio neighborhood, using orthogonal code division-multiplexing channels. Such approaches require sophisticated code management algorithms. In the sensor networks we analyze, all nodes operate at the same frequency, and hence parallel transmissions within the same radio neighborhood are not possible.

Hop-by-hop flow control protocols have been extensively studied in the context of ATM and local-area networks [9, 14, 16, 17]. The motivation in these high-speed networks is to avoid the burst behavior of end-to-end protocols like TCP at small round-trip times. In sensor networks, hop-by-hop flow control is attractive because it allows good congestion adaptation without incurring losses or requiring the expensive end-to-end acknowledgments that are unnecessary for many streams that don't require TCP-style reliability.

6. CONCLUSION

This paper presents an experimental evaluation of three complementary congestion control strategies for wireless sensor networks. We show that unless a sensor network operating under load has some means of controlling congestion, it will face significant degradation in efficiency and fairness. As network load increases, or when channel variations cause fluctuations in achievable bandwidth, nodes must modulate their send rates based on local congestion feedback or the network will go into congestion collapse.

The metrics we use for evaluation express properties that designers and users of sensor networks should find desirable. Network efficiency quantifies the amount of energy that is wasted on transmissions that do not deliver packets. Fairness quantifies the degree of variation in send rates, which impacts sensor network coverage. Imbalance measures the inequity between packet receptions at a given node and its next hop, providing a clearer picture of the hotspots in a network and how that results in buffer and wireless drops. Latency is important because many event-driven applications require the timely reporting of sensor values.

We evaluate three techniques for mitigating congestion both in isolation and in concert. Our results show that hop-by-hop flow control with a simple queue occupancy-based congestion detection method offers substantial efficiency improvements for all types of workloads and utilization levels. This finding holds because a successful wireless transmission requires both the sender and receiver to be contention-free with respect to both the wireless channel and queue space. Implementing a rate-limiting policy results in substantial improvements to fairness. Finally, MAC enhancements support the operation of hop-by-hop flow control.

We analyze two ways of detecting congestion: queue occupancy and channel sampling. In addition to offering significantly better performance, queue occupancy requires no support from the MAC layer and is very easy to implement on different platforms.

We present Fusion, a congestion control mechanism that combines rate limiting, hop-by-hop flow control, and a prioritized MAC. Our results show the efficacy of Fusion under a variety of workloads on a 55-node deployment. A simple periodic workload benefits because it is extremely difficult to adequately provision for varied link capacities of a large scale deployment. A high fan-in network realizes gains from congestion control because the nature of that topology makes transit nodes particularly prone to buffer drops. Correlated-event workloads need congestion control to handle the sudden bursts of traffic that spatially-correlated events generate.

6.1 Future Work

The results presented in this paper point to a number of possible areas for future work. First, although the rate limiting scheme presented in Section 3.2 is effective at improving the fairness of networks under load, a robust rate limiting algorithm that correctly handles node failures, skewed routing topologies, and variable send rates would be useful.

Second, even though our implementation of hop-by-hop flow control relies on overhearing, an alternate implementation is possible with the use of link-level acknowledgments to indicate congestion state. Although we have briefly investigated this design point, we leave a performance comparison for future work.

Finally, while we offer hints as to the sources of loss in our network, more work needs to be done to find definitive answers. In particular, we are investigating whether losses occur because of hidden terminal collisions (in which case RTS/CTS might be a solution) or due to additive interference from distant sources of noise.

Acknowledgments

We thank our shepherd John Heidemann as well as the anonymous reviewers for valuable feedback. We also thank Vladimir Bychkovsky, Michel Goraczko, Sam Madden, Allen Miu, and Stanislav Rost for their valuable feedback and help setting up the testbed.

7. REFERENCES

[1] Aad, I., and Castelluccia, C. Differentiation Mechanisms for IEEE 802.11. In *Proc. of the IEEE INFOCOM Conf.* (Anchorage, AK, April 2001), pp. 209–218.

[2] Chen, B., Jamieson, K., Balakrishnan, H., and Morris, R. Span: An Energy-Efficient Coordination Algorithm for Topology Maintainence in Ad Hoc Wireless Networks. In *Proc. of the ACM MOBICOM Conf.* (Rome, Italy, July 2001), pp. 85–96.

[3] Chipcon Corporation. CC1000 Transceiver Datasheet. http://www.chipcon.com.

[4] De Couto, D. S. J., Aguayo, D., Bicket, J., and Morris, R. A High-Throughput Path Metric for Multi-Hop Wireless Routing. In *Proc. of the ACM MOBICOM Conf.* (September 2003), pp. 134–146.

[5] Ganeriwal, S., Kumar, R., and Srivastava, M. B. Timing-sync Protocol for Sensor Networks. In *Proc. of the ACM Sensys Conf.* (Los Angeles, CA, November 2003), pp. 138–149.

[6] Intanagonwiwat, C., Govindan, R., and Estrin, D. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In *Proc. of the ACM MOBICOM Conf.* (Boston, MA, August 2000), pp. 56–67.

[7] Intel Corporation. New Computing Frontiers – The Wireless Vineyard. http://www.intel.com/labs/features/rs01031.htm.

[8] Jain, R. *The Art of Computer Systems Performance Analysis*, First ed. Wiley, 1991.

[9] Lee, D., Coleri, S., Dong, X., and Ergen, M. FLORAX—Flow-Rate Based Hop by Hop Backpressure Control for IEEE 802.3x. In *5th IEEE Conf. on High Speed Networks and Multimedia Communications* (Jeju Island, Korea, July 2002).

[10] Lemmon, M. D., Ling, Q., and Sun, Y. Overload Management in Sensor-Actuator Networks used for Spatially-Distributed Control Systems. In *Proc. of the ACM Sensys Conf.* (Los Angeles, CA, November 2003), pp. 162–170.

[11] Lu, C., Blum, B. M., Abdelzaher, T. F., Stankovic, J. A., and He, T. RAP: A Real-Time Communication Architecture for Large-Scale Wireless Sensor Networks. In *Proc. of the IEEE RTAS Symposium* (San Jose, CA, September 2002).

[12] Luo, H., Lu, S., and Bharghavan, V. A New Model for Packet Scheduling in Multihop Wireless Networks. In *Proc. of the ACM MOBICOM Conf.* (Boston, MA, August 2000), pp. 87–98.

[13] Mainwaring, A., Polastre, J., Szewczyk, R., Culler, D., and Anderson, J. Wireless Sensor Networks for Habitat Monitoring. In *Proc. of the ACM WSNA Workshop* (Atlanta, GA, September 2002).

[14] Mishra, P., and Kanakia, H. A Hop by Hop Rate-based Congestion Control Scheme. In *Proceedings of the ACM SIGCOMM Conf.* (Baltimore, MD, August 1992), pp. 112–123.

[15] Nandagopal, T., Kim, T.-E., Gao, X., and Bhargavan, V. Acheiving MAC Layer Fairness in Wireless Packet Networks. In *Proc. of the ACM MOBICOM Conf.* (Boston, MA, August 2000), pp. 87–98.

[16] Noureddine, W., and Tobagi, F. Selective Backpressure in Switched Ethernet LANs. In *Proceedings of the IEEE GLOBECOM Conf.* (Rio De Janeiro, Brazil, December 1999), pp. 1256–1263.

[17] Özveren, C., Simcoe, R., and Varghese, G. Reliable and Efficient Hop-by-Hop Flow Control. In *Proceedings of the ACM SIGCOMM Conf.* (London, UK, August 1994).

[18] Perkins, C., and Bhagwat, P. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. *Computer Communication Review* (October 1994).

[19] Sankarasubramaniam, Y., Özgür Akan, and Akyildiz, I. ESRT: Event-to-Sink Reliable Transport in Wireless Sensor Networks. In *Proc. of the ACM Mobihoc Conf.* (Annapolis, MD, June 2003), pp. 177–189.

[20] University of California, Berkeley. Firebug. http://firebug.sourceforge.net/.

[21] van Dam, T., and Langendoen, K. An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *Proc. of the ACM Sensys Conf.* (Los Angeles, CA, November 2003), pp. 171–180.

[22] Wan, C.-Y., Eisenman, S., and Campbell, A. CODA: Congestion Detection and Avoidance in Sensor Networks. In *Proc. of the ACM Sensys Conf.* (Los Angeles, CA, November 2003), pp. 266–279.

[23] Welsh, M. and Werner-Allen, G. Motelab webpage. http://motelab.eecs.harvard.edu.

[24] Woo, A., and Culler, D. A Transmission Control Scheme for Media Access in Sensor Networks. In *Proceedings of the ACM MOBICOM Conf.* (Rome, Italy, 2001), pp. 221–235.

[25] Woo, A., Tong, T., and Culler, D. Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks. In *Proc. of the ACM Sensys Conf.* (Los Angeles, CA, November 2003), pp. 14–27.

[26] Xu, Y., Heidemann, J., and Estrin, D. Geography-Informed Energy Conservation for Ad Hoc Routing. In *Proceedings of the ACM MOBICOM Conf.* (Rome, Italy, July 2001), pp. 70–84.

[27] Ye, W., Heidemann, J., and Estrin, D. An Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *Proc. of the IEEE INFOCOM Conf.* (New York, NY, June 2002), pp. 1567–1576.

[28] Yi, Y., and Shakkottai, S. Hop-by-hop Congestion Control over a Wireless Multi-hop Network. In *Proc. of the IEEE INFOCOM Conf.* (Hong Kong, June 2004).

[29] Zhao, J., and Govindan, R. Understanding Packet Delivery Performance in Dense Wireless Sensor Networks. In *Proc. of the ACM Sensys Conf.* (Los Angeles, CA, November 2003), pp. 1–13.