Congestion Avoidance Based on Lightweight Buffer Management in Sensor Networks

Shigang Chen, *Member*, *IEEE*, and Na Yang

Abstract—A wireless sensor network is constrained by computation capability, memory space, communication bandwidth, and above all, energy supply. When a critical event triggers a surge of data generated by the sensors, congestion may occur as data packets converge toward a sink. Congestion causes energy waste, throughput reduction, and information loss. However, the important problem of congestion avoidance in sensor networks is largely open. This paper proposes a congestion-avoidance scheme based on lightweight buffer management. We describe simple yet effective approaches that prevent data packets from overflowing the buffer space of the intermediate sensors. These approaches automatically adapt the sensors' forwarding rates to nearly optimal without causing congestion. We discuss how to implement buffer-based congestion avoidance with different MAC protocols. In particular, for CSMA with implicit ACK, our $1/k$-buffer solution prevents hidden terminals from causing congestion. We demonstrate how to maintain near-optimal throughput with a small buffer at each sensor and how to achieve congestion-free load balancing when there are multiple routing paths toward multiple sinks.

Index Terms—Sensor networks, network communication.

1 Introduction

1.1 Motivation

Wireless sensor networks have a wide range of applications in habitat observation [1], [2], health monitoring [3], object tracking [4], [5], battlefield sensing, etc. They are different from traditional wireless networks in many aspects [6]. For example, the sending rate of a sensor is determined not only by the channel capacity but also by the activity of the neighbor sensors as well as the lifetime requirement (due to limited energy supply). Intense study has been carried out in recent years on physical layer [7], [8], MAC layer [9], [10], [11], and network layer [12], [13], [14], [15]. However, the important problem of congestion avoidance in sensor networks remains largely open. When a sensor receives more data than it can forward, the excess data has to be buffered. *Congestion* occurs when the limited buffer space is full and, consequently, the received data has to be dropped. *Congestion control* studies how to recover from a congestion. *Congestion avoidance* studies how to prevent congestion from happening, which is the subject of this paper.

A different type of congestion occurs when an area is densely populated with sensors, causing frequent radio collision if many sensors attempt to send simultaneously. The classical solutions for this problem are exponential random backoff and virtual carrier sensing. While the proposed techniques can also greatly reduce the chance of radio collision (Section 3.6), the main focus of this paper is on buffer-based congestion, which can easily happen in a sensor network where the packets converge towards a sink. As shown in Fig. 1, an intermediate sensor x close to the sink is likely to have multiple upstream sensors. With CSMA, *the upstream sensors collectively have more chance to forward packets to x than x can send out.* The excessive packets received by x will eventually cause buffer overflow. Consequently, hotspots may form around the bottleneck sensors.

Congestion causes many problems. When a packet is dropped, the energy spent by upstream sensors on the packet is wasted. The further the packet has traveled, the more the waste is. When the buffer at a sensor x is already full, if the upstream neighbors attempt to send data to x, their efforts (and energy) are deemed to be wasted and, worse yet, counter-productive. For instance, their RTS packets may collide with nearby transmissions, causing throughput reduction of other sensors. Finally, and above all, the data loss due to congestion may jeopardize the mission of the application.

1.2 Related Work

Sensor networks typically operate under light load and suddenly become active in response to certain important events such as fire outbreak, earthquake, or enemy movement. This sudden surge of data from hundreds or even thousands of sensors must be delivered to a small number of sinks, which may cause congestion especially near the sinks. While fusion techniques [14] can be used for data aggregation, applications may require some specific information (e.g, the exact locations of reporting sensors) to be kept [6], which sets a limit on how much the fusion can do.

Experiments [16] showed that maintaining an operating point that does not exceed the capacity of a sensor network is critical to improving performance in both networking and application metrics. However, congestion control in sensor

• *The authors are with the Department of Computer and Information Science and Engineering, University of Florida, Gainesville, FL 32611. E-mail: {sgchen, nyang}@cise.ufl.edu.*

Manuscript received 12 Feb. 2005; revised 31 May 2005; accepted 2 Sept. 2005; published online 26 July 2006.
Recommended for acceptance by I. Stojmenovic, S. Olariu, and D. Simplot-Ryl.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDSSI-0078-0205.

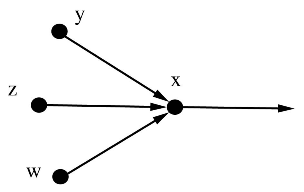


Fig. 1. CSMA causes buffer overflow.

networks had not received serious study until recently. Most current control mechanisms are rate-based.

In directed diffusion [13], interests are propagated from sinks to data sources; the reverse paths are used for forwarding data packets to the sinks. Based on the quality of data delivery, a sink reinforces certain paths by sending new interests, which increase the data rates on those paths. Although it is not designed specifically for congestion control, directed diffusion may adapt for this task (to some degree) by reinforcing paths with small delays, but this strategy will penalize distant data sources. In addition, the sink-initiated control reacts slower than the on-the-spot localized congestion control.

In ESRT [17], a sensor sets a congestion-notification (CN) bit in the packet header if its buffer is about full. The sink periodically computes a new *reporting rate* (at which each source is supposed to report data) based on a reliability measurement, the received CN bits, and the previous reporting rate. It then broadcasts the new reporting rate to all data sources. Treating all sources equally is suboptimal. To remove all congestions, the reporting rate has to be set according to the worst hotspot in the network. In that case, the noncongested sources will be constrained by a conservative reporting rate.

CODA [18] provides a comprehensive discussion on congestion control and proposes an open-loop hop-by-hop backpressure mechanism and a close-loop multisource regulation mechanism. For hop-by-hop backpressure, each sensor detects congestion by monitoring the channel utilization and the buffer-occupancy level. In response to congestion, it sends backpressure messages to neighbors, which may drop packets, reduce its sending rate, and further propagate backpressure. For multisource regulation, if a source sensor reports data at a rate greater than a preconfigured threshold, the sensor must receive a continuous stream of ACKs from the sink in order to maintain that rate. This provides the sink a means to regular the source rates by deciding how many ACKs to broadcast. The above mechanisms are reactive. They do not prevent congestion from happening.

The work by Ee and Bajcsy [19] assumes a tree routing structure from all data sources to a sink. Each sensor receives and forwards packets from its upstream neighbors; each upstream neighbor is the root of an upstream subtree. The sensor learns the number of data sources in each of those upstream subtrees, measures its own downstream forwarding rate, computes per-source fair rate, which is propagated upstream such that the data sources do not send packets beyond the rate.

Fusion [20] consists of three congestion mitigation techniques. The first technique is called hop-by-hop flow control, which resembles backpressure [18] but replaces the explicit control packets with a piggybacked congestion bit carried by all packets. When overhearing the congestion bit to be set, the upstream neighbors (virtually) stop transmitting until the congestion bit is unset. The second technique is called rate limiting, which meters traffic being admitted to the network to prevent unfairness. The third technique is called prioritized MAC, which ensures that congested nodes receive prioritized access to the channel.

One common problem of rate-based congestion control in sensor networks is the difficulty for an upstream sensor to determine the right amount of rate deduction in response to a downstream congestion. The prior work has largely avoided the discussion of this problem. The traditional AIMD approach (additive increase multiplicative decrease) relies on periodic rate adjustment. Due to environmental dynamics (e.g., background radio interference, multipath fading, and a change in the number of active neighbors), the bandwidth available to a congested sensor changes all the time, which would constantly cause upstream sensors to perform rate adjustment. It is much desired to have a new approach that allows the upstream sensors to quickly adapt their rates to near-optimal ones without explicit, slow-converging rate-based control.

1.3 Our Contributions

This paper attempts to answer the following questions: Can we eliminate buffer-based congestion in a sensor network? How to maintain near-optimal throughput without congestion? Does the buffer size have to be large? Can we avoid explicit rate signaling between sensors? How to ensure fairness in buffer access? How to achieve congestion-free load balancing through multipath routing?

First proposed by Kung et al. [21] for flow control in ATM networks, the basic idea is that a sender should transmit a packet only when it knows that the receiver has the buffer to store the packet. We describe how this idea can be used for congestion avoidance in a sensor network. We design simple yet effective approaches that prevent data packets from overflowing the buffer space of the downstream sensors, and discuss how to implement them with various MAC protocols. In particular, for CSMA with implicit ACK, one has to overcome the hidden-terminal problem. We propose a $1/k$-buffer solution that ensures hidden terminals do not cause congestion. Through Markov-chain analysis and simulations, we demonstrate that the proposed approaches automatically adapt the sensors' forwarding rates to nearly optimal without causing congestion. They can produce much larger network throughput than the rate-based approaches. We also address the fairness issue when multiple sensors try to access the buffer space of the same downstream sensor, and study how to achieve congestion-free load balancing when there are multiple routing paths towards multiple sinks.

The rest of the paper is organized as follows: Section 2 defines the sensor network model and discusses radio collision. Section 3 proposes our congestion avoidance scheme. Section 4 describes the congestion avoidance algorithms and Section 5 analyzes the impact of buffer size on throughput. Section 6 presents the simulation results. Section 7 draws the conclusion.

2 Background

2.1 Network Model

A sensor network consists of a set of sensors and a set of base stations (also called *sinks*) for data collection. Two sensors are neighbors if they are in the transmission range of each other and can directly communicate with certain reliability. We assume there exists a neighbor discovering protocol. For example, each sensor periodically transmits a beacon packet identifying itself, so that every sensor knows the set of its neighbors. The sensors share the same wireless media, and each packet is transmitted as a local broadcast in the neighborhood. We assume the existence of a MAC protocol, e.g., based on CSMA or TDMA, which resolves the media contention and ensures that only the intended receiver keeps the packet and other neighbors discard the packet. Although there may exist asymmetric communication links, only symmetric ones are used for sending data. That is because for x to transmit a packet to y, x must know the existence of y as a neighbor, which means that x can hear y's beacon. Moreover, some MAC protocols such as CSMA/CA can only work with symmetric communication links.

The sensors are statically located after deployment. We do not consider mobile sensors that form a dynamic ad hoc network. We study data packets sent from sensors to sinks. While there may be a subset of sensors generating data, all sensors will serve to relay the packets toward the sinks. Assume that the sinks are connected via an external network to a data collection center. It makes no difference which particular sink a packet is delivered to. Suppose all data packets have the same size. The size of a buffer space is counted as the number of packets that the buffer can store.

2.2 Collision and Congestion

Radio collision and buffer overflow are two main types of congestion in a sensor network. Solutions against collision include CSMA, TDMA, CDMA, etc. In [22], we use simulations to show that, with an appropriate size of minimum contention window, the classical approach of exponential random backoff can effectively control the radio collision problem to an insignificant level. But, solving media contention does not necessarily mean solving congestion. The shared media access among sensors brings a new congestion scenario that is not present in a wired network. More specifically, CSMA's "fair" media access directly contributes to buffer overflow. Refer to Fig. 1, where data sources y, z, and w send packets to the sink via x. If y, z, w, and x each obtain a fair share of channel capacity, x will receive three packets for every packet it sends out. Consequently, its internal packet queue will build up and eventually overflow. Therefore, it is not sufficient for the data sources to slow down to a level that does not cause serious collision. They must slow down further such that x is able to send at a higher rate that matches the combined rate of y, z and w. How to achieve this in a dynamic environment where the channel capacity and the contention from neighbors may change at any time? This is the question that the paper wants to answer.

3 Buffer-Based Congestion Avoidance

3.1 Basic Scheme

The key for congestion avoidance is to make sure that a sensor y sends a packet to another sensor x only when x has the buffer space to hold the packet. Below, we describe a basic congestion avoidance scheme.

Let N_x be the set of neighbor nodes of x. The residual buffer of x changes when x receives a packet from or forward a packet to a neighbor sensor. To keep N_x updated with x's buffer size, whenever x sends out a packet, it piggybacks its current buffer state in the frame header, for example, using one bit to indicate if the buffer is full or using a few bits to store the size of the residual buffer. Note that both data and control packets sent by x can piggyback the buffer state.

Consider a neighbor sensor $y \in N_x$. When y receives or overhears a packet from x, it caches the buffer state of x. When y has a packet to forward x, only if x's buffer is not full, y forwards the packet. Otherwise, y withholds the packet until it overhears a packet from x, piggybacking a nonfull buffer state.

The proposed basic scheme avoids packet drop due to buffer overflow. It quickly adapts the data rates at the sources and the forwarding rates at the immediate sensors to near-optimal values. For example, suppose an object entering a field triggers a large number of sensors to track its movement. How fast should those sensors send data to the sink? If the sending rate is too small, the system may lose track of the object amidst other moving objects. If the sending rate is too large, it may cause congestion and be counter-productive. Suppose the sensors initially attempt to send as fast as they can. When the buffer at an intermediate sensor x is filled, by our scheme the forwarding rates of its upstream sensors are forced to slow down, in accordance to x's forwarding rate. When the buffers at the upstream sensors are filled up, the further upstream sensors are forced to slow down. This process repeats towards the furthest sensors and eventually the whole network adapts toward the maximum congestion-free throughput.

Our buffer-based scheme is simple but effective. It eliminates the complicated rate-based signaling that is required by many existing congestion control approaches, yet it can produce much larger network throughput (Section 6) and, unlike the rate-based approaches, it does not drop packets. In the following, we describe how to implement the basic scheme with various MAC protocols. We do not intend to exhaustively discuss all MAC protocols, but rather choose a few examples to show in principle how the problems can be solved.

3.2 CSMA/CA and CSMA with ACK

First, we consider CSMA/CA. Virtual carrier sensing is used to reduce the probability of radio collision due to hidden terminals. Data transmission requires RTS-CTS-DATA-ACK exchange between two neighboring sensors. One bit in each packet is used to piggyback whether the sender's buffer is full. A sensor y forwards a data packet to a neighbor x only when it learns that x's buffer is not full. Now, consider the following two cases:

- *Case 1: y may not overhear packets sent by x due to temporary radio interference. Therefore, its knowledge about x's buffer may be stale.*
- *Case 2: When y wakes up from the sleeping mode, its knowledge about x's buffer may be stale.*

There are two approaches that handle the above cases:

- *Approach 1:* The one-bit buffer state can be piggybacked by the neighbor discovery messages that are exchanged periodically between neighbors. The state information will be resynchronized between x and y as long as they remain neighbors of each other. The time for resynchronization is determined by the frequency at which the neighbor discovery messages are transmitted.
- *Approach 2:* y attempts to transmit *if it has not overheard x's buffer state for a period of time T* (Case 1) or *if it has not overheard x's buffer state since it wakes up from a sleep period of at least T* (Case 2), where T is a system parameter. Under these two conditions, y thinks that its knowledge of x's buffer state may be stale. If y has a data packet to send, it transmits a RTS packet to x, which replies with a CTS packet, piggybacking its current buffer state. Upon receipt of CTS, y sends DATA packet(s) if the buffer is not full. Basically, the RTS/CTS exchange resynchronizes the buffer state before DATA is sent. If y does not receive CTS due to collision, it performs the exponential random backoff as usual.

Congestion avoidance in CSMA with ACK is handled similarly. For Approach 2, because there is no RTS, a DATA packet is transmitted when the buffer information is suspected to be stale. If the receiver drops the packet due to buffer overflow, without hearing the ACK the sender performs exponential random backoff and retransmit, the packet later.

3.3 CSMA with Implicit ACK and TDMA with Fixed Schedule

For a sensor network with small packet size, the control packets in RTS-CTS-DATA-ACK exchange constitute a significant overhead. Woo and Culler suggested that RTS and CTS should be used only when the media contention level is high. In addition, the acknowledgement can be free when the sender of the packet overhears the transmission of the same packet by the receiver [10], which saves ACK at the cost of slightly increased holding time of the packet at the sender. This is called *implicit ACK*. Therefore, when the media contention level is not high, CSMA can be reduced to DATA packets only for the purpose of energy efficiency. An alternative approach of implementing implicit ACK is for data packets to carry an one-byte acknowledgement field in their headers, such that a data packet forwarded by a downstream sensor acknowledges a (different) packet just received from an upstream neighbor.

Another scenario that has only DATA packets is TDMA with fixed transmission schedule, where each sensor is assigned fixed time slots. Far from optimal in terms of total throughput, a simple TDMA may still be a viable choice for low-cost sensor networks that have very limited on-board resources (e.g., energy, CPU, and memory), which make simplicity a higher priority than network throughput.

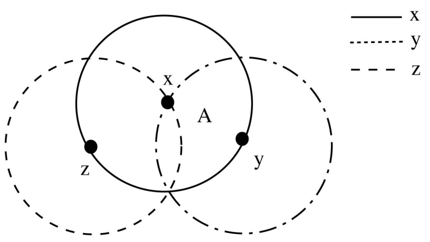


Fig. 2. Hidden-terminal problem.

For the above two MAC protocols, only DATA packets are available to piggyback the buffer state. Without ACK packets piggybacking, we have to modify the basic scheme as follows: When a sensor x sends out a DATA packet, it piggybacks its residual-buffer size in the frame header. When a neighbor $y \in N_x$ overhears a frame from x, it caches the residual-buffer size of x. When y overhears a packet that is sent by another sensor to x, it reduces the residual-buffer size of x by one.[1] In addition, y's knowledge about x's residual buffer may be stale when it misses packets due to radio interference or sleeping. The approaches that handle these cases in Section 3.2 should be applied here as well.

Even with the above modification, y may still lose track of the accurate size of x's residual buffer due to the hidden terminal problem. Fig. 2 gives an example. The transmission ranges of three sensors (x, y, and z) are shown by three circles, respectively. Suppose x advertises (by piggybacking on a DATA packet) that it can hold one more packet. Both y and z want to send x a packet. Suppose z sends its packet first. Because y is outside of z's transmission range, it will not overhear the packet and still think that x can hold a packet. When y sends its packet to x, congestion happens and the packet must be dropped.

To better *illustrate* our idea, we consider an *idealized* network with each sensor having the same circular transmission range, and propose a $1/6$-buffer *solution* for the hidden-terminal problem, i.e., every sensor advertises only one sixth of its residual-buffer size. The general case with irregular transmission ranges will be studied in the next section.

Theorem 1. *Suppose all sensors have the same circular transmission range. Hidden terminals do not cause buffer congestion when the* $1/6$-buffer *solution is used.*

Proof. Consider an arbitrary sensor x. We can prove the theorem by showing that no congestion will occur at x between *any two consecutive transmissions* by x. Without losing generality, suppose two consecutive transmissions are made at times t_0 and t_1. Let L be the residual-buffer size that x advertises at t_0. The actual residual-buffer size at the time is $6L$. We prove that, before the next transmission by x, the data sent to x will not exceed $6L$ and, consequently, no congestion will occur.

Refer to Fig. 3, where transmission ranges are shown by circles. Only neighbor sensors in the solid circle can send data to x. The solid circle can be divided into six convex regions, A, B, C, D, E, and F, which overlaps

1. This would be unnecessary if x sent an ACK packet, which would carry x's reduced buffer size and be overheard by y.

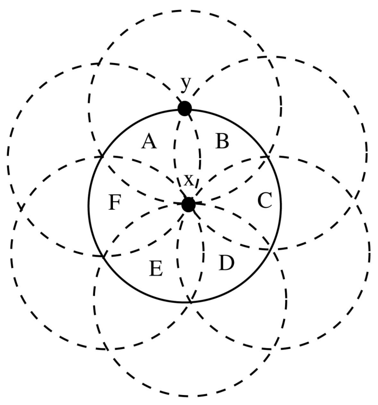


Fig. 3. 1/6-buffer solution.

partially. For example, A partially overlaps with B and F. The sensors in each region can overhear each other's transmission and, therefore, the number of data packets sent from the region to x cannot exceed L, based on the above congestion avoidance scheme. There are six such regions. Hence, the total number of packets sent from all neighbors to x cannot exceed $6L$ before x makes another buffer advertisement at t_1. □

One may ask why we choose to advertise $1/6$ of the residual buffer, instead of $1/4$, $1/5$, $1/7$, etc. As shown in Fig. 3, the neighborhood of x can be divided into six convex regions *with the sensors of each region able to overhear each other*, which is a key property needed by the proof of Theorem 1. The property is not true for any less number of regions. On the other hand, any larger number of regions will also support the proof but lead to unnecessarily low buffer advertisement, e.g., a $1/7$-buffer solution would advertise one seventh of the residual buffer.

In practice, it is likely that a sensor can advertise more than one sixth of its residual buffer. Refer to Fig. 3. Assume that the sink is to the right side of x and geographic routing [23], [24], [12], [25] is used. Because a packet is always routed to a neighbor that is closer to the sink, only sensors in three regions to the left of x (i.e., A, F, and E) will send packets to x. Following the same argument used in the proof, x is able to advertise one third of its residual buffer without causing congestion, i.e., a $1/3$-buffer solution.

3.4 Adaptive 1/k-Buffer Solution

Our derivation of the value $1/6$ is based on an idealized assumption that all sensors have the same circular transmission range. In reality, the radio transmission range is highly irregular. It not only depends on the power level, but also on environmental dynamics such as radio interference and physical obstacles. To handle the general case, we propose an adaptive $1/k$-buffer solution, where k is modified dynamically by the sensor who advertises its residual buffer. On one hand, k can be as low as three when geographic routing is used. On the other hand, k can be larger than 6 in the worse case where the radio transmission range is highly irregular. Our adaptive $1/k$-buffer solution works as follows. Each sensor initializes k to be 6. If there is no buffer overflow for a long time, the sensor concludes that k is set too conservatively and it reduces k by one. Whenever there is buffer overflow, the sensor knows that k is set too aggressively and it increases k by one. The idea is to dynamically adjust k to a minimum value that does not cause buffer overflow. Due to traffic dynamics, periodic packet drops may happen as k is cyclically increased and then decreased. Although a small amount of packet drops can be tolerated by many applications, it is desirable to minimize data loss. One approach is to enlarge the no-overflow time period for k's reduction to be large enough such that the packet overflow rate becomes negligibly small. Another approach is to (at least temporarily) stop decreasing k when k's value keep oscillating up and down.

It may seem underperformed to advertise just one kth of the residual buffer, and it might appear that $(k-1)/k$ of the whole buffer could be left unused. That is not true because a sensor always makes revised advertisement through piggybacking. If $(k-1)/k$ of the whole buffer is left unused, then $1/k$ of the unused will be advertised. Eventually, most buffer space will be utilized even in the worst case where all upstream sensors come from one convex region in Fig. 3. Only when x's residual-buffer size becomes $(k-1)$ or less, it advertises zero, which stops all upstream sensors from transmitting to x.

Furthermore, our simulations demonstrate that a sensor only needs to allocate a small buffer in order to achieve high throughput and avoid congestion. Intuitively, a large buffer would be needed to absorb busty traffic that would otherwise be lost, but it is not necessarily essential for achieving good long-term average throughput for periodic CBS traffic that will not be lost due to buffer overflow, thanks to the congestion avoidance scheme. As an illustrative example, even one buffer slot may achieve good throughput if the two nodes take turn to send. Confirming this intuition, we found in our simulations that conservative buffer advertisement by the $1/k$ solution has negligible impact on data throughput. For a buffer of just 12 slots (each holding one packet) and using only two header bits to piggyback the buffer state (0, 1, or 2), our congestion avoidance scheme considerably outperforms the existing congestion control schemes.

3.5 Fairness in Buffer Access

As shown in Fig. 4, if CSMA is used, all upstream sensors (y, z, and w) compete fairly for transmission and each is statistically guaranteed a fair share of bandwidth for sending data to x. On the other hand, TDMA may result in extreme unfairness. In the lower plot of Fig. 4, suppose all sensors are in each other's transmission range, x runs the $1/6$-buffer solution with just six buffer slots, and the periodic transmission schedule is in the order of x, y, z, and w. Every time when x advertises one residual slot, y may immediately send its packet. Overhearing the packet, z and w think the buffer at x is now full and will not send their packets. The order of transmission gives y priority over z and w, allowing the former to starve the latter.

The basic idea of our solution is to provide a mechanism for the first node in the transmission schedule that cannot send to reserve the next available buffer slot at the receiver. After this node sends a packet, the next node in the schedule has the opportunity to reserve, and this process continues such that the nodes take turn to reserve the buffer slots at the receiver in

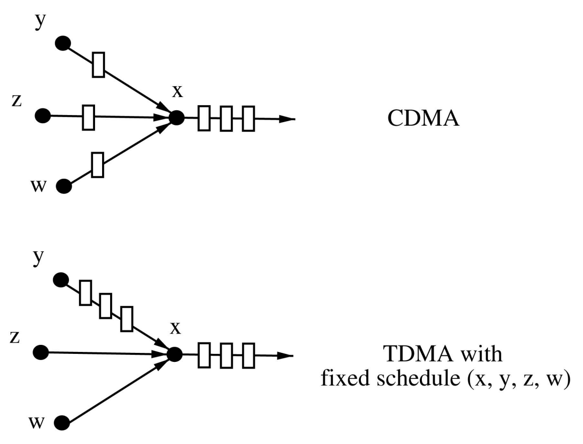


Fig. 4. Fairness among upstream sensors.

the order that they appear in the schedule. We use the previous example to explain our reservation mechanism. Based on the transmission schedule, when z has its turn to send a packet, if it finds that x's buffer is full because y just transmitted a packet, z will instead transmit a control message called NEXT(z, x) if nobody else has done so. The message is to announce that z should get the next available buffer slot of x. When the next node w in the schedule hears NEXT(z, x), it will keep silent without sending its NEXT(w, x). y also hears NEXT(z, x) and records the message. When x advertises a residual slot again, although y is still scheduled ahead of z, it will give up the chance of transmission and remove the recorded NEXT(z, x). z now has the chance to send x its packet. Overhearing z's packet, w learns that x's buffer becomes full and it cannot send a packet to x. It will instead send NEXT(w, x) to reserve the next available buffer slot of x. The message will be recorded by both y and z, which will give up their chance in the following round. After w sends, y has its chance to send and then the above reservation process repeats. The lifetime of a recorded NEXT message is one round of the TDMA schedule. This is to prevent a dead node from indefinitely holding the right of transmitting to x.

3.6 Collision Reduction

The proposed congestion-avoidance scheme stops upstream sensors from transmitting when the downstream neighbors do not have the required buffer to hold the packets, which would be dropped anyway if they were transmitted. By eliminating unhelpful transmission and keeping the upstream sensors silent, it not only saves energy but also helps reducing radio collision with other sensors transmitting in the neighborhood. As an example, refer back to Fig. 3. Consider the radio transmissions to x and assume that x has six buffer slots. According to the $1/6$-buffer solution, when y sends a packet to x, it will silence all neighbors of x in regions A and B,[2] which covers 39 percent of x's neighborhood. If y is closer to x, instead of locating at the perimeter, then an even larger portion of x's neighbors will overhear y's packet and stop transmitting to x. After at most six packets are sent to x, all neighbors will stop transmitting for a period before x empties its buffer and makes a new advertisement. During this period, not only the neighbors avoid wasting energy in transmitting to x, but their silence reduces the chance of collision with parallel transmissions (to sensors other than x) and, thus, helps to improve the overall throughput.

4 Congestion Avoidance Algorithms

4.1 Congestion Avoidance with Multiple Paths

We first consider the case of multipath routing to one sink. Suppose each sensor x knows a list of neighbor sensors, denoted as R_x, which can be used to forward packets to the sink. The first sensor in the list R_x has the highest preference to be used, the second sensor has the second highest preference, and so on. As described in the previous section, each sensor advertises its residual-buffer size (based on the $1/k$-buffer solution in case of CSMA with implicit ACK). x records the residual-buffer sizes advertised by the nodes in R_x. When x overhears a packet sent to $y \in R_x$, it reduces y's residual-buffer size by one. When x has a packet to be sent to the sink, it executes a congestion-free multipath forwarding routine (CFMF). The routine attempts to find the first sensor in the list R_x whose buffer is not full. If none in R_x can hold the packet and TDMA is used, it attempts to reserve the next available buffer slot of a sensor in R_x by sending a NEXT message.

```
$x$.CFMF(packet)
1. **iterate** $y \in R_x$ in the order of preference **do**
2.    **if** $y$'s residual buffer is not full **then**
3.       forward the packet to $y$
4.       return
5. **if** (TDMA)
6.    **iterate** $y \in R_x$ in the order of preference **do**
7.       **if** NEXT(*, $y$) has not been recorded **then**
8.          send NEXT($x$, $y$)
9.          return
10. re-execute upon receipt of new buffer advertisement
      from $R_x$
```

If geographic routing [23], [24], [25] is used, R_x is defined as

$$R_x = \{y \mid d(y) < d(x), y \in N_x\}. \tag{1}$$

The nodes in R_x are sorted in the ascending order of the distance to the sink. When some communication links become unreliable, the corresponding nodes in R_x should be removed or given lower preference until the links regain reliability. If $|R_x|$ is large, we only keep the first k sensors in the list, where k is a system parameter. The void problem is resolved by the right-hand rule [24].

Without geographic routing, R_x may be constructed as follows. The sink periodically broadcasts a beacon, which carries a unique identifier s and a hop counter c initialized to be zero. When a sensor x receives a beacon for the first time (from a neighbor y), it increases the hop counter by one, records this hop count as d_x, inserts y to R_x, and broadcast the beacon with d_x to its neighbors. When x receives a subsequent beacon with d_u from a neighbor u, it checks if $d_u + 1 \leq d_x$.[3] If so, x inserts u to R_x together with

2. Those neighbors keep silence without transmitting to x, but may transmit to other sensors in their neighborhoods.

3. The number of nodes in R_x can be increased by allowing all u nodes with $d(u) \leq d(x) \wedge u < x$ to be inserted to R_x. The condition $u < x$ is to prevent a routing loop from being formed.

the value of d_u. The beacon is then discarded. Using nodes in R_x as the next hops makes sure that the packets do not travel backward away from the sink. The nodes u in R_x are sorted in the ascending order of their d_u values with the exception that nodes with low link reliability are given low preference.

The sink increases the broadcast identifier s by one for each subsequent broadcast. Each sensor keeps the largest identifier that it has seen. It resets R_x when receiving a beacon with a larger identifier, and discards all received beacons with smaller identifiers.

4.2 Congestion Avoidance with Multiple Sinks

We now consider the case of multipath routing to multiple sinks. Let B be the set of sinks. A separate routing structure is constructed for each sink. Specifically, every sensor x maintains a separate list R_x^b of next hops for every $b \in B$ in the same way as described previously. It also creates a single preference list R_x for all sinks. R_x is sorted based on the (geographical or hop) distance from a neighbor to its closest sink. More specifically, R_x is a list of tuples, $\langle y, b, d\rangle$, sorted in the ascending order of d, where y is a next-hop neighbor, b is the closest sink to y, and d is the distance from y to b.

When x is ready to send a locally originated packet, it first executes the following routine to identify a sink to which the packet will be delivered.

```
$x$.Target_Sink()
1. **iterate** $\langle y, b, d\rangle \in R_x$ in the order of preference **do**
2.    **if** $y$'s residual buffer is not full **then**
3.       return $b$
4. **if** (TDMA)
5.    **iterate** $\langle y, b\rangle \in R_x$ in the order of preference **do**
6.       **if** NEXT(*, $y$) has not been recorded **then**
7.          send NEXT($x$, $y$)
8.          return $b$
9. re-execute upon receipt of new buffer advertisement from
   $R_x$
```

The above algorithm attempts to identify the closest sink to which the routing path is not congested. Consider a path from x through a neighbor y to a sink b'. If the path is persistently congested, all the intermediate sensors including y will eventually fill up their buffers, and Target_Sink() will avoid selecting b' (Lines 2-3). After a sink is identified, the problem is reduced to the case of multipath routing for a single sink, which has been addressed previously. R_x is used by a source sensor x to select a target sink. R_x^b is used by an intermediate sensor x to route a packet to a selected sink b.

Some optimization may be done. Let b be the sink selected by the source of a packet. Suppose after receiving the packet an intermediate sensor x finds that none of the neighbors in R_x^b can hold the packet. According to x.CFMF(), the packet has to wait until R_x^b releases some buffer, even though a sensor in $R_x^{b'}$ currently has the required buffer space, where b' is a different sink. A naive optimization is to allow any intermediate sensor to switch the destination sink of a packet. This may, however, cause routing loops. A compromise is to allow a packet to switch its destination sink only for a small number of times. It can be easily implemented by a TTL field in the packet header, whose value is decreased by one for each switch.

5 Analysis

We analyze the throughput of a sensor when the proposed congestion avoidance scheme is implemented with CSMA/CA or CSMA with ACK, where the information piggybacked in ACK from a sensor ensures that all neighbors know its current residual-buffer size. To simplify the analysis, we do not consider the two cases that cause stale buffer information (Section 3.2).

We model the change of a sensor x's residual-buffer size as a discrete-time finite-state Markov chain. The possible states are $\{0, 1, 2, \ldots, m\}$, where m is the maximum buffer size. There are only two types of events that changes the state of x's residual buffer:

- Event 1: x successfully transmits a data packet.
- Event 2: One of x's upstream neighbors successfully transmits a data packet to x.

Event 1 increases x's residual buffer by one. Event 2 decreases x's residual buffer by one. Event 1 and Event 2 do not happen at the same time because x cannot receive while transmitting. Other events (i.e., transmissions with neither sender nor receiver being x) may also occur in the neighborhood. They are irrelevant to and, thus, ignored in the analysis.

If the current state is $i \in \{1, 2, \ldots, m-1\}$, let a be the probability for the next event being Event 1, and $(1-a)$ be the probability for the next event being Event 2. The value of a is determined later. The transition probability from state i to state $i-1$ is therefore a, and the transition probability from state i to state $i+1$ is $(1-a)$. The transition probability for state 0 (empty buffer) to state 1 is 100 percent, and the transition probability for state m (full buffer) to state $m-1$ is also 100 percent. The transition probabilities between other states are zero. Let $P = (p_{ij})$, for $i, j \in [0..m]$, be the matrix of transition probabilities.

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 \\ a & 0 & 1-a & 0 & 0 & \ldots & 0 & 0 & 0 & 0 \\ 0 & a & 0 & 1-a & 0 & \ldots & 0 & 0 & 0 & 0 \\ 0 & 0 & a & 0 & 1-a & \ldots & 0 & 0 & 0 & 0 \\ & & \ldots & & & \ldots & & & \ldots & \\ 0 & 0 & 0 & 0 & 0 & \ldots & a & 0 & 1-a & 0 \\ 0 & 0 & 0 & 0 & 0 & \ldots & 0 & a & 0 & 1-a \\ 0 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 & 1 & 0 \end{pmatrix}.$$

Let $\pi = (\pi_0, \pi_1, \ldots \pi_m)$ be the stationary distribution of the Markov chain, which satisfies $\pi P = \pi$. $\sum_{i=0}^{m} \pi_i = 1$, and π_i is the stationary probability for x's residual-buffer size being i in the stochastic process of the Markov chain. We have

$$(\pi_0, \pi_1, \dots . \pi_m) \cdot \begin{pmatrix} 0 & 1 & 0 & 0 & \dots & 0 & 0 & 0 \\ a & 0 & 1-a & 0 & \dots & 0 & 0 & 0 \\ 0 & a & 0 & 1-a & \dots & 0 & 0 & 0 \\ & & \dots & & \dots & & \dots & \\ 0 & 0 & 0 & 0 & \dots & a & 0 & 1-a \\ 0 & 0 & 0 & 0 & \dots & 0 & 1 & 0 \end{pmatrix} = (\pi_0, \pi_1, \dots . \pi_m).$$

It can be rewritten as

$$\begin{aligned} \pi_0 - a\pi_1 &= 0 \\ (1-a)\pi_i - a\pi_{i+1} &= 0, \quad i \in [1..m-1] \\ (1-a)\pi_{m-1} - \pi_m &= 0 \\ \sum_{i=0}^{m} \pi_i &= 1. \end{aligned}$$

Solving the linear equations, we have

$$\begin{aligned} \pi_0 &= \frac{a^{m-1} - 2a^m}{2(1-a)^m - 2a^m} \\ \pi_i &= \frac{a^{m-1} - 2a^m}{2(1-a)^m - 2a^m} \frac{1}{a} \left(\frac{1-a}{a}\right)^{i-1}, \quad i \in [1..m-1] \\ \pi_m &= \frac{a^{m-1} - 2a^m}{2(1-a)^m - 2a^m} \left(\frac{1-a}{a}\right)^{m-1}. \end{aligned}$$

Let $n(\geq 1)$ be the expected number of upstream neighbors that have data to transmit to x at any given time. Let p be the probability that at least one sensor in R_x has a nonfull buffer. According to our congestion avoidance scheme, x will compete for media access only when at least one neighbor in R_x has a nonfull buffer.[4] When that is the case, the conditional probability for the next event being Event 1 is about $\frac{1}{n+1}$, which is the result of access competition between x and its n upstream neighbors. Consequently, $a = \frac{1}{n+1}p$. Note that we simplify the analysis by using the expected number n, instead of the variable, instantaneous number of upstream neighbors that have nonempty buffers.

The possible value range of a is $(0, \frac{1}{2}]$. π_0 is the probability of x's buffer size being zero. It is a function of a. For $a \in (0, \frac{1}{2}]$, it can be shown that π_0 is monotonically increasing with respect to a. Intuitvely, the chance for x's buffer to be empty is greater when x has more chance to send, i.e., a is greater. Therefore, we have

$$\pi_0(a) = \pi_0\left(\frac{1}{n+1}p\right) \leq \pi_0\left(\frac{1}{n+1}\right) = \frac{1}{2\sum_{i=0}^{m-1} n^i}.$$

Sensor x achieves its maximum data throughput when its buffer is always nonempty such that it competes for media access whenever allowed. Now, with buffer-based congestion avoidance, the probability for the buffer to be empty is π_0, which can also be interpreted as the percentage of time when x does not compete for media access, i.e., an upper bound on the percentage of lost throughput. In other words, because the fraction of time that x competes for media access is at least $(1 - \pi_0)$, the fraction of maximum possible throughput that is actually realized is at least $(1 - \pi_0)$. Therefore, with buffer-based congestion avoidance, the ratio of the actual throughput to the maximum throughput is at least

$$1 - \pi_0 \geq 1 - \frac{1}{2\sum_{i=0}^{m-1} n^i},$$

which is plotted in Fig. 5. The ratio is close to 100 percent with a modest buffer size (≥ 12).

It is a much harder problem to analyze the $1/k$-buffer solution in CSMA with implicit ACK. We study it next by simulations.

4. If the buffers at x's next-hop neighbors are all full, the packet sent by x will be dropped and, consequently, does not contribute to data throughput anyway.

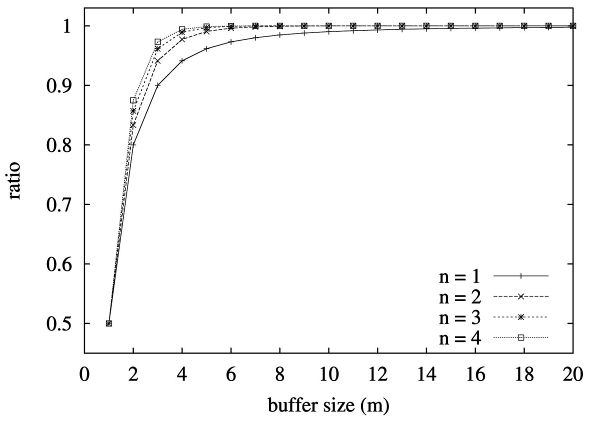


Fig. 5. Ratio of actual throughput to optimal throughput.

6 Simulation Results

We perform extensive simulations to evaluate the proposed congestion avoidance scheme (including the $1/k$-buffer solution) in sensor networks with CSMA. Packet collision caused by random media access is resolved by exponential random backoff. With a static or dynamic set of data sources, we compare our scheme with others in terms of the following performance metrics: *accumulated packet loss*, *achievable source rate*, *average routing distance per packet*, *average routing delay per packet*, and *average energy expenditure per packet*.

The default simulation parameters are described as follows. Five hundred sensors are randomly placed in a $1,000 \times 1,000$ area. The transmission ranges of the sensors are randomly chosen from $[100, 200]$. The transmission rate is 512 kbps. Due to limited energy supply, a sensor should not send data continuously at a high rate because it shortens its life time. The sustainable rate of a sensor is configured to be 10 packets per second. Each data packet is 30 bytes long. The buffer space at each sensor can hold 12 packets. Five base stations (sinks) are evenly spaced along one edge of the deployment area. There are 100 data sources, randomly selected from the 500 sensors. Each data source generates new data at an initial rate of four packets per second; it may generate at a lower rate due to congestion control. Unless explicitly specified otherwise, all parameters are assumed to take their default values. We use the beacon-based routing algorithm described in Section 4.1 to maintain the shortest routing paths (stored by R_x at every sensor x). Note that there may be multiple shortest paths from a sensor to

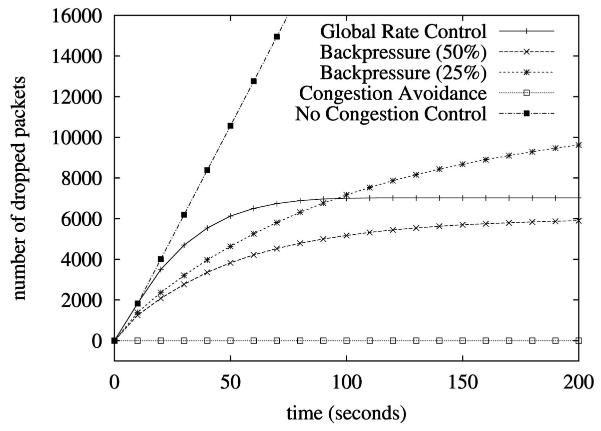


Fig. 6. Accumulated packet loss over time.

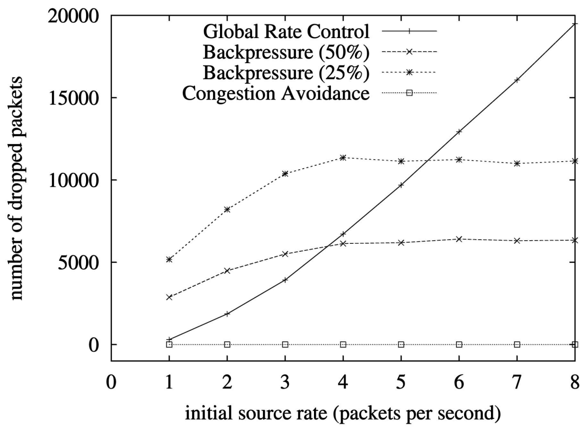


Fig. 7. Final packet loss (when all congestions are removed) with respect to initial source rate.

the sinks. The following four congestion control/avoidance schemes are implemented:

- Global Rate Control: If a sensor is congested, it sets a congestion-notification bit in the packet header. A base station periodically (once per second) broadcasts a reporting rate at which each source sensor should generate new data. Initially, the reporting rate is equal to the initial data rate of a source (four packets per second by default). If a base station receives packets whose congestion-notification bits are set, the next reporting rate will be reduced by a percentage (25 percent in the simulations). After all congestions in the network are removed, the reporting rate will stabilize.
- Backpressure: This is CODA's hop-by-hop congestion control mechanism. If a sensor x is congested (based on channel utilization and buffer level), it periodically sends backpressure messages to its neighbors, which reduce their forwarding rates to x by a percentage (25 percent or 50 percent in the simulations). If an upstream neighbor is a data source, the neighbor reduces the rate at which it produces new data by the same percentage. After all congestions in the network are removed, the rates at all sensors will stabilize.
- Congestion Avoidance: It is the scheme proposed in this paper.
- No congestion control: Data packets are dropped by congested sensors and no further action is taken.

In the following, we first compare the above four schemes in terms of packet loss and achievable source rate. We then evaluate the properties of routing distance, routing delay, and energy expenditure. We also compare the schemes with a continuously changing set of data sources. Finally, we study the impact of failed overhearing on our buffer-based congestion avoidance scheme. For each data point in the figures, we run the simulation on 50 randomly created networks and then take the average result.

6.1 Packet-Loss Comparison

The first set of simulations confirms that our Congestion Avoidance scheme seldom drops packets due to buffer overflow, but other schemes do.

Fig. 6 shows the number of overflowed packets in the network with respect to time. Backpressure (50 percent) and Backpressure (25 percent) both refer to the Backpressure scheme, but their percentages of reduction in a sensor's data rate in response to a backpressure message are 50 percent and 25 percent, respectively. Both Global Rate Control and Backpressure drop a significant number of packets during the process of congestion control, whereas few packet drops by Congestion Avoidance are observed during simulations. Backpressure (50 percent) drops a less number of packets than Backpressure (25 percent) because it reduces data rates more aggressively and, thus, removes congestions more quickly. Global Rate Control reduces data rates for all sensors in response to congestion. It sharply reduces the amount of traffic and removes the congestion more quickly than Backpressure. Congestion Avoidance tries to prevent congestion from being developed and therefore avoid packet drops.

Fig. 7 compares the final number of dropped packets (when all congestions are removed) with respect to the initial rate at which the sources generate new data. Intuitively, when the initial source rate is higher, it takes more reduction cycles (and, thus, more time) to reduce the rate to an appropriate level, which means more packet drops. This is evident for Global Rate Control, whose number of dropped packets is roughly proportional to the initial source rate. On the other hand, Backpressure is less sensitive to the initial source rate because its localized control reacts more precisely against congestion and is more flexible in directing excessive traffic to other paths.

Fig. 8 compares the final number of dropped packets (when all congestions are removed) with respect to the number of data sources. A larger number of data sources means more traffic in the network, an increased likelihood of congestion, and consequently more dropped packets. The number of packets dropped by Global Rate Control is roughly proportional to the number of data sources. The same thing is also true for Backpressure but with a much smaller slope, demonstrating the advantage of localized congestion control. On the other hand, while both are localized, our proposed buffer-based congestion avoidance has advantage over the rate-based Backpressure because it takes a preventive approach instead of a reactive one.

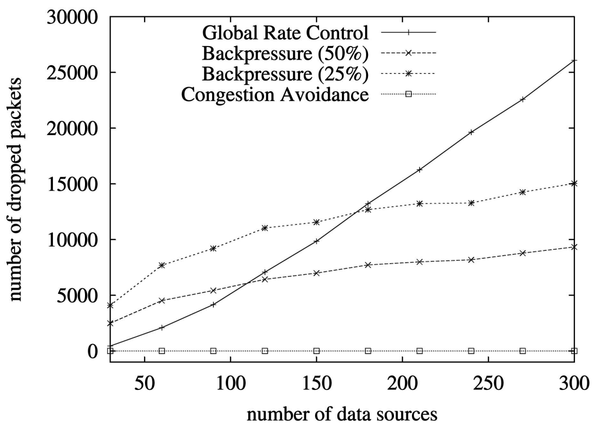


Fig. 8. Final packet loss (when all congestions are removed) with respect to number of sources.

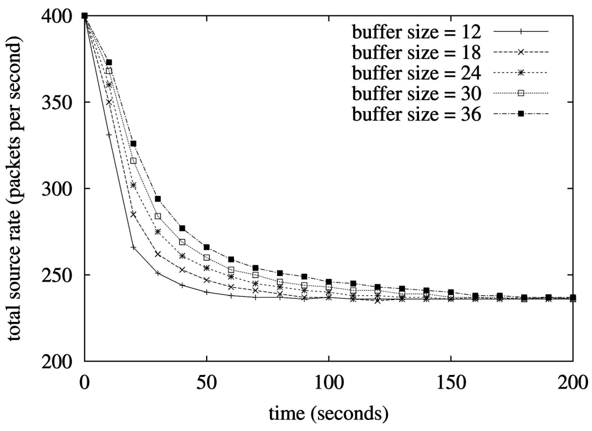


Fig. 10. Congestion Avoidance: How does the buffer size affect the total source rate?

6.2 Source-Rate Comparison

The second set of simulations demonstrates that our Congestion Avoidance scheme is able to automatically adapt the sensors' data rates according to the network conditions and achieve better congestion-free rates than other schemes.

The *total source rate* is defined as the total number of data packets generated by the data sources per second. Fig. 9 compares the total source rates of the schemes with respect to time. During the course of congestion control/avoidance, the total source rates are reduced. At around time = 200 seconds, congestions are removed in all schemes (except for No Congestion Control) and the total source rates stabilize. Congestion Avoidance achieves the best source rate due to its capability of redirecting traffic towards other downstream paths that are not congested, which is implemented by the distributed execution of x.CFMF(...), particularly Lines 1-4, at all sensors x. Global Rate Control has the worst source rate because it treats all sources in the same way as it treats the one whose downstream paths are most congested. Backpressure falls in the middle. The final source rates of Backpressure (50 percent) and Backpressure (25 percent) are similar with the latter slightly better. Combining with the results in Fig. 6, we see a tradeoff between the number of dropped packets and the total source rate. By more aggressively reducing the rates in response to congestion, Backpressure (50 percent) has a lower number of dropped packets but a smaller source rate than Backpressure (25 percent).

Fig. 10 demonstrates, for the Congestion Avoidance scheme, how the buffer size at each sensor affects the total source rate over time. After time = 180 seconds, the source rates of all simulations stabilize and they are very close to one another, which means that a small buffer size (= 12) can already achieve suboptimal performance and the gain by further increasing the buffer size is not significant, agreeing with our analytical results.

Fig. 11 compares the total source rates (after all congestions are removed) with respect to the network size. With a fixed number of data sources (100 by default), when the network size increases, the total source rates increase for all schemes. That is because a larger network has more routing paths and is less likely to be congested. The gap in source rate between Congestion Avoidance and other schemes widens when the network is larger. Fig. 12 compares the total source rates (after all congestions are removed) with respect to the number of data sources. With a fixed network size (500 by default), when the number of data sources increases, the total source rates also increase for all schemes. That is simply because more data sources

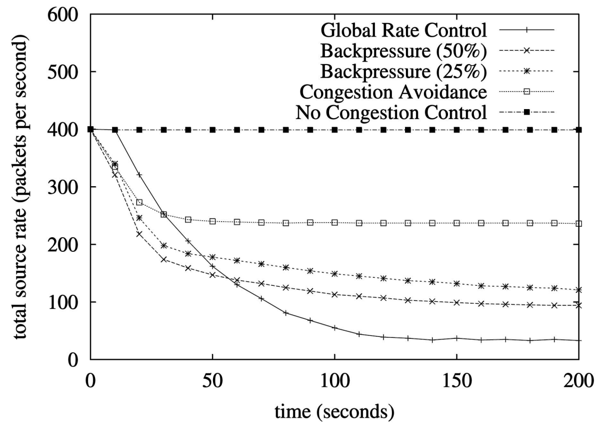


Fig. 9. How does the total source rate change over time?

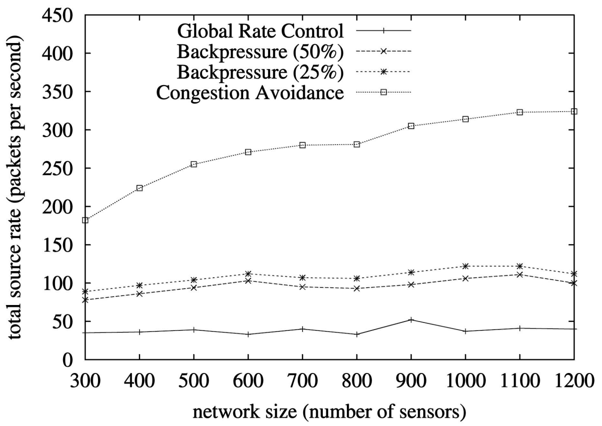


Fig. 11. Total source rate (when all congestions are removed) with respect to network size.

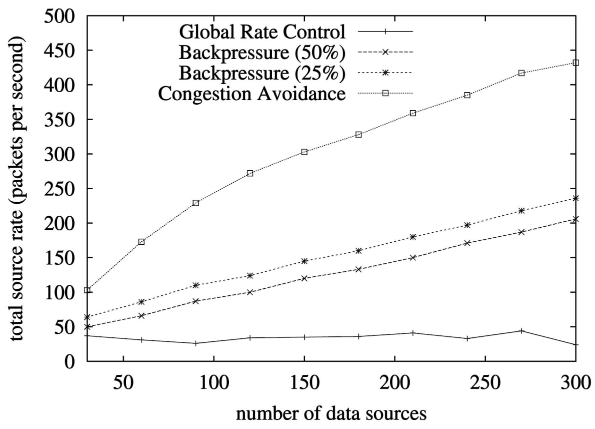


Fig. 12. Total source rate (when all congestions are removed) with respect to number of sources.

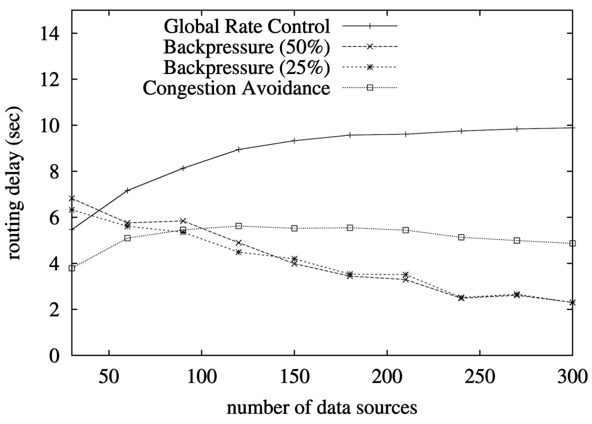


Fig. 14. Routing delay is not proportional to routing distance due to varied per-hop queuing delay.

will utilize the available bandwidths of the network paths more thoroughly.

6.3 Routing Distance, Routing Delay, and Energy Expenditure

The four congestion control/avoidance schemes use the same shortest-path routing algorithm. Hence, the routing distance from the same data source to the sinks is always the same for all schemes. If distant data sources send less, then the average routing distance per packet will be smaller, and vice versa. Global Rate Control is the fairest scheme because all data sources send at the same rate. So, its average routing distance sets a benchmark. For other schemes, the less the distant nodes send, the smaller the average routing distance will be. Fig. 13 shows that Backpressure has the smallest routing distance, which means nodes closer to the sink send much more than their fair share. The routing distance of Congestion Avoidance is much closer to that of Global Rate Control, which means distant nodes are penalized much less.

Fig. 14 shows the average routing delay per packet after the congestions are removed. The end-to-end routing delay is mostly determined by the number of hops and per hop queuing delay. Backpressure fully utilizes the buffer space. Consequently, the average routing delay decreases when the average routing distance decreases, which happens when the number of data sources increases (Fig. 13). Due to the k-buffer solution, the queue length in Congestion Avoidance will be kept smaller than the buffer size, which means it has smaller per-hop delay than Backpressure. However, Backpressure still has a smaller end-to-end delay due to its smaller routing distance (Fig. 13). Global Rate Control has the largest routing delay due to its largest routing distance.

The average energy expenditure is defined as the total number of transmissions made in the network divided by the number of packets delivered to the sinks. One transmission moves a packet one hop closer to a sink. Fig. 15 shows how the average energy expenditure changes over time. Congestion Avoidance is more energy efficient than Global Rate Control because the latter drops many packets, which waste a lot of transmissions. Backpressure has the lowest energy expenditure because it is not fair to the distant sensors. More packets are generated from the sensors close to the sinks, and it takes less numbers of transmissions to deliver them to the sinks.

6.4 Performance with Dynamic Sources

The fourth set of simulations studies the performance of the schemes when the set of data sources is dynamic. Starting

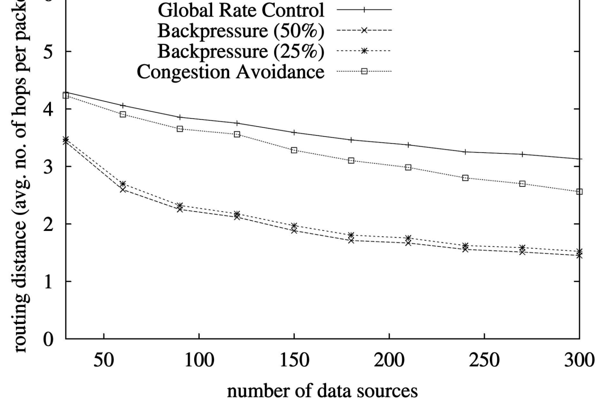


Fig. 13. Average routing distance indicates how much the distant nodes are penalized.

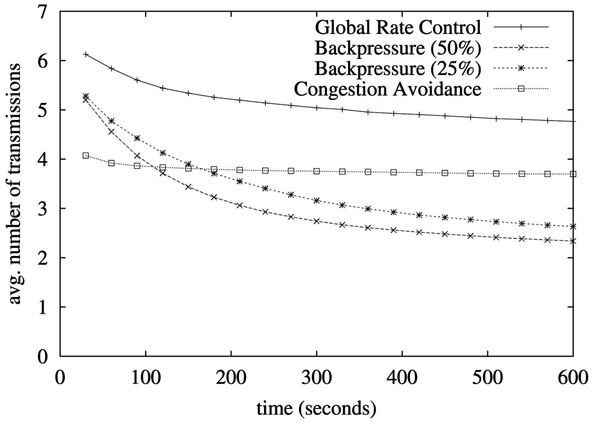


Fig. 15. Energy expenditure per packet over time.

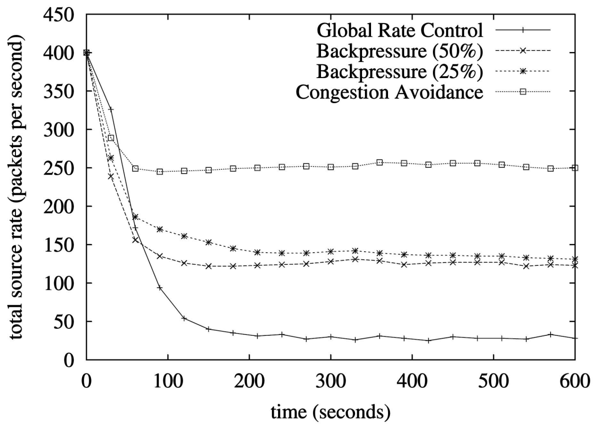


Fig. 16. Total source rate over time when the set of data sources is dynamic.

with 100 data sources, the set is continuously changing at a rate of one every five seconds, meaning that during every five seconds, on average one sensor in the set ceases to produce new data while another sensor joins the set to produce new data. Fig. 16 compares the total source rates and Fig. 17 compares the numbers of packets dropped by the schemes. The results are consistent with the previous simulations. Congestion Avoidance achieves better source rates with few packet drops. The source rate of Global Rate Control is the worst, while Backpressure drops more packets over time.

6.5 Impact of Failed Overhearing

Our last simulation studies Congestion Avoidance in a noisy environment where overhearing for buffer advertisement is unreliable. When a sensor x transmits a packet to a next-hop neighbor y, y has a certain probability of not hearing the packet due to noise interference, which causes a packet drop. Independently, another neighbor z has the same probability of not overhearing the packet, which causes stale buffer information at z because it misses the piggybacked buffer advertisement. Because of the stale information, z may send a packet to y while y's buffer is full, causing a packet drop due to buffer overflow.

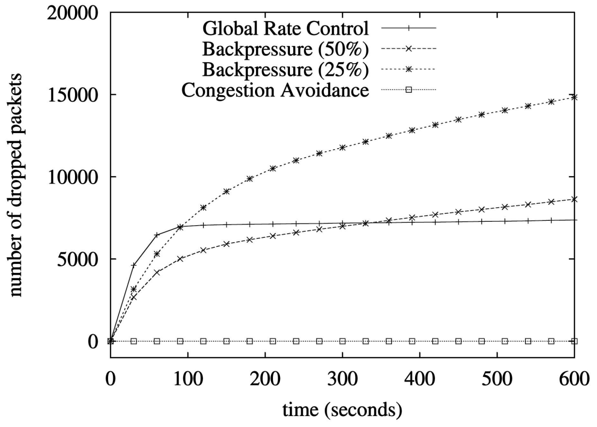


Fig. 17. Packet drop over time when the set of data sources is dynamic.

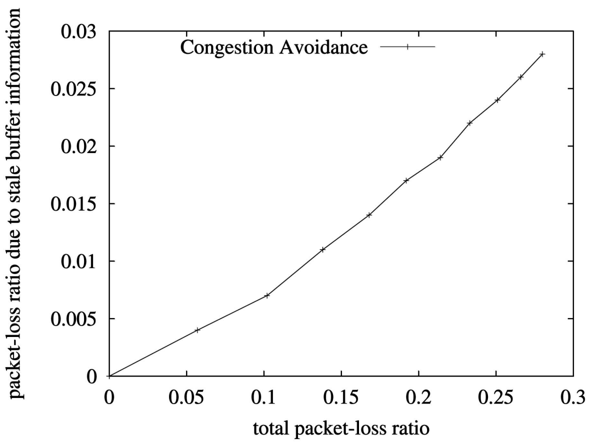


Fig. 18. Impact of failed overhearing.

In summary, there are two types of packet drops. One is due to noise interference, and the other is due to stale buffer information that causes buffer overflow. The packet-drop ratio is the total number of packet drops divided by the total number of packets generated by the sources. We can increase the packet-drop ratio in our simulation by increasing the probability of noise interference. In Fig. 18, the x axis shows the packet-drop ratio and the y axis shows the portion of the packet-drop ratio due to stale buffer information. From the figure, the stale buffer information causes relatively insignificant packet drops. For example, when the total packet loss is 19 percent (the first point to the left of 0.2 at the x axis), the contribution by stale buffer information is only 1.7 percent.

7 Conclusion

This paper proposes a buffer-based congestion avoidance scheme. We discuss how to implement such a scheme with various MAC protocols. For CSMA with implicit ACK and TDMA with a fixed schedule, we propose the $1/k$-buffer solution for the hidden-terminal problem. We address the fairness issue in buffer access and how to balance load over multiple paths to multiple sinks based on the buffer availability. We thoroughly evaluate the proposed scheme by simulations. We show by analysis and simulations that a small buffer per sensor can achieve near-optimal throughput.

References

[1] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao, "Habitat Monitoring: Application Drive for Wireless Communications Technology," *Proc. ACM SIGCOMM Workshop Data Comm. in Latin Am. and the Caribbean,* Apr. 2001.
[2] E. Biagioni and K. Bridges, "The Applications of Remote Sensor Technology to Assist the Recovery of Rare and Endangered Species," *Int'l J. High Performance Computing Applications,* special issue on distributed sensor networks, Apr. 2003.
[3] L. Schwiebert, S. Gupta, and J. Weinmann, "Research Challenges in Wireless Networks of Biomedical Sensors," *Mobile Computing and Networking,* pp. 151-165, 2001.
[4] H.T. Kung and D. Vlah, "Efficient Location Tracking Using Sensor Networks," *Proc. IEEE Wireless Comm. and Networking Conf. (WCNC),* Mar. 2003.

[5] R. Brooks, P. Ramanathan, and A. Sayeed, "Distributed Target Classification and Tracking in Sensor Networks," *Proc. IEEE,* vol. 91, no. 8, pp. 1163-1171, 2003.

[6] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A Survey on Sensor Networks," *IEEE Comm. Magazine,* Aug. 2002.

[7] C. Chien, I. Elgorriaga, and C. McConaghy, "Low-Power Direct-Sequence Spread-Spectrum Modem Architecture for Distributed Wireless Sensor Networks," *Proc. Int'l Symp. Low Power Electronics and Design (ISLPED '01),* Aug. 2001.

[8] R.J. Cramer, M.Z. Win, and R.A. Scholtz, "Impulse Radio Multipath Characteristics and Diversity Reception," *Proc. IEEE Int'l Conf. Comm. '98,* June 1998

[9] E. Shih, S. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, and A. Chandrakasan, "Physical Layer Driven Protocol and Algorithm Design for Energy-Efficient Wireless Sensor Networks," *Proc. ACM MobiCom '01,* July 2001.

[10] A. Woo and D. Culler, "A Transmission Control Scheme for Media Access in Sensor Networks," *Proc. ACM MobiCom '01,* July 2001.

[11] K. Sohrabi, J. Gao, V. Ailawadhi, and G.J. Pottie, "Protocols for Self-Organization of a Wireless Sensor Network," *IEEE Personal Comm.,* Oct. 2000.

[12] T. He, J.A. Stankovic, C. Lu, and T.F. Abdelzaher, "SPEED: A Stateless Protocol for Real-Time Communication in Sensor Networks," *Proc. Int'l Conf. Distributed Computing Systems (ICDCS '03),* May 2003.

[13] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed Diffusion for Wireless Sensor Networking," *ACM/IEEE Trans. Networking,* vol. 11, no. 1, pp. 2-16, Feb. 2002.

[14] W.R. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive Protocols for Information Dissemination in Wireless Sensor Networks," *Proc. ACM MobiCom '99,* Aug. 1999

[15] S. Hedetniemi, S. Hedetniemi, and A. Liestman, "A Survey of Gossiping and Broadcasting in Communication Networks," *Networks,* vol. 18, 1988.

[16] S. Tilak, M.B. Abu-Ghazaleh, and W. Heinzelman, "Infrastructure Tradeoffs for Sensor Networks," *Proc. First ACM Int'l Workshop Wireless Sensor Networks and Applications (WSNA '02),* Sept. 2002.

[17] Y. Sankarasubramaniam, O. Akan, and I. Akyildiz, "ESRT: Event-to-Sink Reliable Transport in Wireless Sensor Networks," *Proc. Fourth ACM Symp. Mobile Ad Hoc Networking and Computing (MobiHoc '03),* June 2003.

[18] C.-Y. Wan, S.B. Eisenman, and A.T. Campbell, "CODA: Congestion Detection and Avoidance in Sensor Networks," *Proc. ACM SenSys '03,* Nov. 2003.

[19] C.T. Ee and R. Bajcsy, "Congestion Control and Fairness for Many-to-One Routing in Sensor Networks," *Proc. ACM SenSys '04,* Nov. 2004.

[20] B. Hull, K. Jamieson, and H. Balakrishnan, "Mitigating Congestion in Wireless Sensor Networks," *Proc. ACM SenSys '04,* Nov. 2004.

[21] H.T. Kung, T. Blackwell, and A. Chapman, "Credit-Based Flow Control for ATM Networks: Credit Update Protocol, Adaptive Credit Allocation, and Statistical Multiplexing," *Proc. ACM SIGCOMM '94,* Aug. 1994.

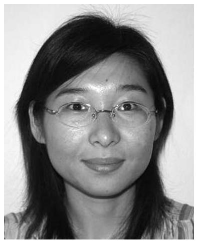
[22] S. Chen and N. Yang, "Congestion Avoidance Based on Light-Weight Buffer Management in Sensor Networks," technical report, Dept. of Computer and Information Science and Eng., Univ. of Florida, 2005.

[23] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia, "Routing with Guaranteed Delivery in Ad Hoc Wireless Networks," *Proc. Third Int'l Workshop Discrete Algorithms and Methods for Mobile Computing and Comm. (DialM '99),* Aug. 1999.

[24] B. Karp and H. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks," *Proc. ACM MobiCom '00,* Aug. 2000.

[25] Q. Fang, J. Gao, and L.J. Guibas, "Locating and Bypassing Routing Holes in Sensor Networks," *Proc. IEEE INFOCOM '04,* Mar. 2004.

Shigang Chen received the BS degree in computer science from the University of Science and Technology of China in 1993 and the MS and PhD degrees in computer science from the University of Illinois at Urbana-Champaign in 1996 and 1999, respectively. After graduation, he worked with Cisco Systems for three years before joining the University of Florida as an assistant professor in 2002. His research interests include network security, quality of service, and sensor networks. He is a member of the IEEE.



Na Yang received the BS and MS degrees in computer science from Fudan University of China in 2000 and 2003, respectively, before joining the PhD program at the University of Florida where she is a student in the Department of Computer and Information Science and Engineering. Her research interests are wireless sensor networks and quality of service.