Traffic management in wireless sensor networks: Decoupling congestion control and fairness ☆

Swastik Brahma [a], Mainak Chatterjee [a,*], Kevin Kwiat [b], Pramod K. Varshney [c]

[a] Department of EECS at the University of Central Florida, Orlando, FL, United States
[b] Information Directorate at Air Force Research Laboratory, Rome, NY, United States
[c] Department of EECS at Syracuse University, Syracuse, NY, United States

ARTICLE INFO

ABSTRACT

In this paper, we propose a distributed congestion control algorithm for tree based communications in wireless sensor networks, that seeks to adaptively assign a *fair* and *efficient* transmission rate to each node. In our algorithm, each node monitors its aggregate output and input traffic rate. Based on the difference of the two, a node then decides to increase (if the output rate is more) or decrease (if the input rate is more) the bandwidth allocable to a flow originating from itself and to those being routed through it. Since the application requirements in sensor network follow no common trait, our design *abstracts* the notion of fairness, allowing for the development of a generic utility controlling module. Such separation of the utility and fairness controlling modules enable each one to use a separate control law, thereby portraying a more flexible design. The working of our congestion control is independent of the underlying routing algorithm and is designed to adapt to changes in the underlying routing topology. We evaluate the performance of the algorithm via extensive simulations using an event-driven packet level simulator. The results suggest that the proposed protocol acquires a significantly high goodput of around 95% of the actual transmission rate, converges quickly to the optimal rate, and attains the desired fairness.

© 2011 Elsevier B.V. All rights reserved.

1. Introduction

Transmission control protocols are a key enabling technology in many of today's sensor network applications. Applications such as of habitat monitoring [15], structural health monitoring [7,8,10], and image sensing [12] are high data-rate applications that heavily rely on congestion control techniques which are an integral part of any transmission control protocol. Well designed congestion control techniques allow efficient transmission of significant volumes of data from a large number of nodes along one or more routes towards the data processing centers (usually known as 'sink' in sensor network terminology). In these high data-rate applications, often bulk data is generated in addition to the constantly sensed data. For example, in structural health monitoring, a set of sensors is deployed in a civil structure such as a building. Each sensor measures structural vibrations continuously and transmits the data to the sink at a certain rate. When the sensors detect a significant anomaly, they generate and send out volumes of data at a much higher rate. Without congestion control, under such traffic characteristics, network collapse due to congestion is inevitable.

A common symptom of congestion in sensor networks is the increase in buffer drop rate and packet delay – similar to traditional wired networks. In addition to these, another key result of congestion in wireless sensor networks is the degradation of radio channel quality. This is because concurrent data transmissions over different radio links interfere with each other, causing channel quality to depend not just on noise, but also on traffic densities. Moreover, the time varying nature of the radio channel and asymmetry in communication links make it harder for even regulated traffic to reach the sink. Another consequence of congestion in multihop wireless sensor networks is that the network gets biased towards delivering data from nodes nearer to the base station and is grossly unfair towards nodes further away.

Congestion also has deleterious effects on energy efficiency. As the offered traffic load crosses a certain point of congestion, the number of bits that can be sent with the same amount of energy decreases. The network ends up wasting energy by transmitting packets from nodes upstream in the network towards the sink, only to be dropped. Thus the goal is not only to achieve high channel utilization, but also to communicate efficiently in terms of energy consumed per unit of successful communication.

Furthermore, providing fairness among flows is also highly desirable. However, the notion of fairness is hard to ground in sensor networks since the application requirements follow no common trait. Fairness is concerned with the relative throughput of the flows

sharing a link. However, how to exactly apportion the available bandwidth among the flows is dictated by the requirements of the application. Thus, it would be beneficial if the notion of fairness can be *abstracted* while designing a congestion control scheme for sensor networks. This would facilitate *de-coupling* the control of *network utilization* from the management of *fairness among different flows*. Such a separation would allow easier implementation of different fairness models according to the needs of the application at hand without considering the design of the module controlling utilization of the network. Moreover, such a design would also be beneficial for sensor networks which can support multiple concurrent applications and also systems with multiple users [11], where there is a need to treat flows from different applications or users in an inequitable manner. In general, de-coupling of the two modules simplifies the design and analysis of each one by reducing the requirements imposed, and enables re-designing one of the modules without considering the other. Though, such separation of the utility and the fairness controllers have been proposed for traditional wired networks in [6], designing such a scheme for wireless sensor networks is not a trivial extension of previous work.

In this paper, we propose a distributed and adaptive mechanism for controlling congestion in sensor networks which seeks to find an optimal transmission rate for the nodes, that is both fair and maximally efficient. In our scheme, we use two separate modules to control *utility* of the network and *fairness* among flows. Each node monitors its aggregate input and output rates. Based on the discrepancy between the two, a node first computes the aggregate increase (if the output rate is more) or decrease (if the input rate is more) in traffic required. Next, the fairness controlling module acts on this aggregate change required and apportions it into individual flows to achieve the desired fairness. Note that the utility controlling module is completely indifferent to how the total change in traffic required is distributed among the flows. Now, to communicate the efficient transmission rate of a flow as calculated by the intermediate nodes along its path to the flow's source, we piggyback control information into data packets and further utilize the broadcast nature of the wireless medium. This enables a child node to overhear the bandwidth allocated to a flow by its parent. The child node, on its part, also computes the bandwidth that can be allocated to the flow by itself, and then compares (with the bandwidth allocated to the flow downstream) and propagates the smaller rate upstream. The proposed protocol adjusts its aggressiveness according to the spare bandwidth in the network. This reduces oscillations, provides stability, and ensures efficient utilization of network resources. Decoupling the utility and fairness controlling modules in our protocol opens new avenues for service differentiation using schemes that provide desired bandwidth apportioning. The scheme also supports multiple concurrent applications and is also highly robust to changes in the underlying topology and routing dynamics and can adapt to changes in them. Moreover, the protocol also has an additional advantage of improving channel quality by inducing a *phase-shifting effect* among neighboring nodes, which introduces a slight jitter to the periodicity of the application thereby breaking the synchronization among periodic streams of traffic. We analyze the performance of the algorithm via extensive simulations using an event-driven packet level simulator written in Java. For simplicity, we build our work upon the sensor network *de facto* standard MAC layer, CSMA, and link-quality based path selection. Our simulation results suggest that the steady state behavior of the protocol can acquire a goodput of around 95% of the actual transmission rate for all nodes, achieve the optimal transmission rate quickly, and furthermore is able to attain fairness among all nodes.

The rest of the paper is organized as follows. Section 2 provides a brief literature review of related works in this area. Section 3 formally defines the problem being looked at and also lays down the design rationale. The congestion control algorithm is presented in Section 4. Section 5 analyzes the steady state performance of the algorithm via extensive simulations. Finally, Section 6 concludes the paper.

2. Related research

Prior work in sensor networks literature has broadly looked at two qualitatively different problems, viz, *congestion mitigation* and *congestion control*. In general, congestion mitigation looks at the following problem. If in a sensor network, the nodes are provisioned to sense the environment and send periodic samples at a fixed rate, then, when the aggregate traffic exceeds the network capacity, how should the nodes regulate their transmissions so that the network goodput and fairness degrade gracefully. This is different from congestion control, which seeks to find an optimal fair rate for the sensor nodes that is also maximally efficient. In this case, when the nodes transmit data at the optimal rate, the network is maximally utilized, and the per node goodput is close to the sending rate.

Adaptive Rate Control (ARC) [19] monitors the injection of packets into the traffic stream as well as route-through traffic. Each node estimates the number of upstream nodes and the bandwidth is split proportionally between route-through and locally generated traffic, with preference given to the former. The resulting bandwidth allocated to each node is thus approximately fair. Also, reduction in transmission rate of route-through traffic has a back-pressure effect on upstream nodes, which in turn can reduce their transmission rates.

In [16], the authors propose Congestion Detection and Avoidance (CODA). CODA uses several mechanisms to alleviate congestion. In *open-loop hop-by-hop backpressure*, when a node experiences congestion, it broadcasts back-pressure messages upstream towards the source nodes, informing them of the need to reduce their sending rates. In *closed-loop multi-source regulation*, the sink asserts congestion control over multiple sources. Acknowledgement (ACKs) are required by the sources to determine their sending rates when traffic load exceeds the channel capacity. In general, open-loop control is more appropriate for transient congestion, while, closed loop control is better for persistent congestion.

In [14], the authors propose the event-to-sink reliable transport protocol. ESRT allocates transmission rates to sensors such that an application-defined number of sensor readings are received at the sink, while ensuring the network is uncongested. The rate allocation is centrally computed at the base station. ESRT monitors the local buffer level of sensor nodes and sets a congestion notification bit in the packets it forwards to the sink if the buffer overflows. If a sink receives a packet with the congestion notification bit set, it infers congestion and broadcasts a control signal informing all sources to reduce their common reporting frequency. However, this approach suffers from a few drawbacks. Firstly, since the sink must broadcast this control signal at a high energy to allow all the sources to hear it, an on-going event transmission can be disrupted by this high powered congestion signal. Moreover, rate regulating all sources as proposed in [14], is fine for homogeneous applications, where all sensors in the network have the same reporting rate but not for heterogeneous ones. Even with a network where all the sources have a uniform reporting rate, ESRT always regulates all sources regardless of where the hotspot occurs in the sensor network. The control law used by ESRT is based on empirically derived regions of operation, and does not attempt to find a fair and efficient rate allocation for the nodes.

Fusion [5] is a congestion mitigation technique that uses queue lengths to detect congestion. Fusion uses three different techniques to alleviate congestion, viz, hop-by-hop flow control, rate limiting, and a prioritized MAC. Hop-by-hop flow control prevents nodes from transmitting if their packets are only destined to be dropped

downstream due to insufficient buffer spaces. Rate limiting meters traffic being admitted into the network to prevent unfairness towards sources far away from the sink. A prioritized MAC ensures that congested nodes receive prioritized access to the channel, allowing output queues to drain. Fusion focuses on congestion mitigation and does not seek to find an optimal transmission rate for the nodes that is both fair and efficient.

In [13], the authors proposed the Interference Aware Fair Rate Control protocol (IFRC). IFRC is a distributed rate allocation scheme that uses queue sizes to detect congestion, and further shares the congestion state through overhearing. Congestion Control and Fairness for Many-to-one Routing in Sensor Networks [3] is another rate allocation scheme that uses a different mechanism than IFRC. Both IFRC and [3] are tangentially related to our work in the sense that they attempt to find optimal transmission rates for all nodes, such that, congestion collapse is avoided. Note that, our algorithm has greater flexibility than IFRC and [3], since many different traffic allocation policies can be implemented in our congestion control scheme, without changing the basic congestion control module (the utility controller). Moreover, IFRC suffers from the additional drawback of having sophisticated parameter tuning for stability, unlike ours.

In [11], the authors propose the Rate Controlled Reliable Transport protocol (RCRT). This protocol is built for loss-intolerant applications that require *reliable* transport of data from the source nodes to the sink. RCRT uses end-to-end explicit loss recovery by implementing a NACK based scheme. Furthermore, RCRT places all congestion detection and rate adaptation functionality in the sinks, thereby producing a centralized congestion control scheme.

The authors in [9] proposes a congestion control mechanism, in which, the buffer in each node is adjusted according to the transmitting downstream nodes in order to minimize packet drop; the algorithm automatically adjusts a node's forwarding rate to avoid packet drops due to congestion. The algorithm resolves the fairness problem by allocating equal bandwidth to the sources. The authors in [22] propose a rate-based fairness-aware congestion control (FACC) protocol, which controls congestion and achieves approximately fair bandwidth allocation for different flows. Their congestion control is based on probabilistic dropping based on queue occupancy and hit frequency. Our congestion control is in contrast to these works as it abstracts the notion of fairness, allowing it to assume different fairness models, such as weighted fairness.

In [23], the authors propose a hop by hop predictive congestion control scheme for WSNs. Their algorithm detects the onset of congestion using queue utilization and a channel estimator algorithm that predicts the channel quality. Flow control is then achieved by a backoff interval selection scheme.

Though, in this paper, we focus on congestion control in sensor networks, it will not be out of place to discuss some recent work on congestion control in wireless network in general. The authors in [17] propose a cross-layer optimization scheme for congestion control in multi-hop wireless networks. They implement a differential backlog based MAC scheduling and router-assisted backpressure congestion control scheme using real off-shelf radios. In [18], the authors focus on fair bandwidth sharing between end-to-end flows, while maintaining an efficient overall throughput in the network. They propose a dynamic rate allocation solution that is based on a simple radio sharing model.

In the next section, we will formulate our problem and also discuss the rationale behind our solution approach.

3. Problem description and design rationale

Let us consider a set of N sensor nodes, numbered 1 through N. In the simplest version of the problem, each node has an infinite amount of data to be sent to a single base station. The nodes can originate data traffic, as well as route traffic originated by other nodes. Thus, each node can act both as a *source*, and a *router*. The nodes sample the environment at periodic intervals, encode the information into data packets, and send them out to a central *base station* or *sink*. Let the flow originating from Node i be f_i, and let r_i be the rate at which flow f_i is injected into the network. We seek to adaptively assign the rate r_i to flow f_i that is both *fair* and *efficient*. Note that, r_i is the rate at which node i injects flow f_i into the network, and does not include the rate at which node i forwards traffic.

We assume that the sensor nodes run a contention based MAC protocol. The default MAC in TinyOS, a widely used sensor network operating system, uses carrier sensing for collision avoidance. Our implementation is based on this MAC, though it can be extended to other MAC protocols as well, such as those that use RTS/CTS [21]. We further assume that the MAC layer provides link-layer retransmissions. Specifically, the MAC protocol used in this paper, senses the medium before trying the send a packet. If the medium is busy, the protocol performs a random backoff before attempting to send again. Collisions are detected using ACKs. If a collision occurs, retransmission of the packet is attempted. Our congestion control protocol works well in a regime where the wireless loss rate over the communication links is such that link layer retransmissions recover most packet losses.

We further assume that the sensor nodes run a routing protocol [20] that builds a routing tree rooted at the base station (or sink). Fig. 1 shows an example of such a tree. The working of our congestion control algorithm is not dependent on the exact choice of the routing tree formed. However, the performance of the algorithm would benefit from a tree formed based on link quality metrics. In the rest of the paper, we assume that a routing tree has been formed by the routing protocol rooted at the sink. Our congestion control algorithm is able to adapt to changes in the underlying routing topology.

3.1. Design rationale

A novel approach adopted in this paper is the decoupling of the notion of *fairness* from *utility* (or efficiency) of the network. Conceptually, fairness and efficiency are independent. Efficiency involves only the aggregate traffic behavior. When the input traffic rate equals the link capacity, no queue builds up and utilization is optimal. Fairness, however, is concerned with the relative throughput of flows sharing a resource. Thus, in our model we use two separate modules for controlling utility and fairness, viz.,
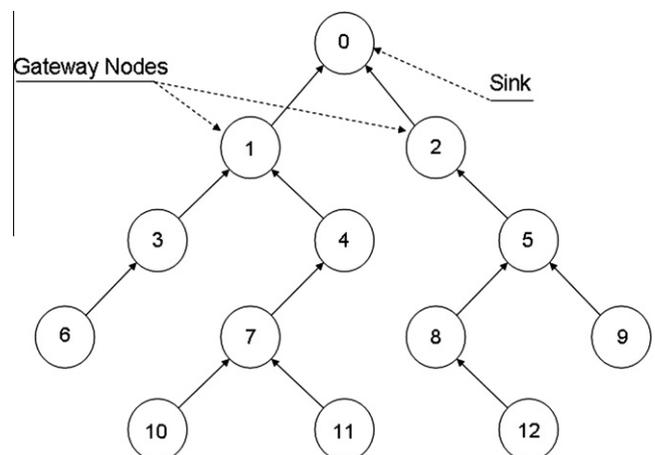

Fig. 1. An example routing tree.

the *utility controlling module* and the *fairness controlling module*. The use of two separate modules allows modification of one of the modules without redesigning the other one. For example, the fairness controlling module can implement any fairness model without changing the underlying utility module. This is particularly attractive in sensor networks where the application requirements follow no common trait. For example, even within a typical application like temperature monitoring, while some nodes might be sending critical information such as news of a fire, others might be sending regular periodic updates. In such scenarios, the application might require more information to be received at the sink from the critical regions, than from regions which appear safe, i.e., the application requires a higher data rate from nodes sending the critical data packets. Furthermore, as envisioned by the authors in [11], future sensor networks are likely to evolve and support multiple concurrent applications running on each mote, such as, motes deployed in a building having both a temperature sensor as well as a vibration sensor. Decoupling utilization from fairness helps abstract the fairness model, and allows for the development of a generic utility controlling module.

3.2. Design goals

Our congestion control has been designed for sensor networks with high data-rate requirements as exemplified by the applications given in Section 1. As mentioned earlier, not only are we interested in high channel utilization, but also in communication efficiency in terms of energy consumed per unit of successful communication. Moreover, fairness is also highly desirable. The four design goals of our congestion control algorithm are as follows.

- *Network Efficiency*: Our primary design goal is to operate the network at an efficient operating point. As network load increases beyond a certain congestion point, congestion collapse occurs and no useful work gets done by the network. Nodes at the edge of the network transmit packets towards the sink, only to be dropped. This not only decreases goodput of the flows but also wastes precious energy reserve of the nodes as energy is invested in transmitting packets only to be dropped later, thereby decreasing the number bits that can be sent with the same amount of energy. At the same time, we need to ensure that the sensor motes send data at a sufficiently high rate, without of course contributing to congestion. Thus, our congestion control algorithm seeks to adaptively find a *fair* transmission rate for all nodes that is also *maximally efficient*.
- *Flexible Design*: Sensor networks deliver myriad types of traffic and the requirements of the applications vary greatly from one to another. In some homogeneous deployments, all nodes transmit data at the same rate, and it might be necessary to split up the available bandwidth equally among them. However, this is not always the case. For deployments in which some nodes require a higher bandwidth allocation (like those sending images), available bandwidth might need to be disproportionately split up among the flows sharing a resource. This calls for a need to abstract away the notion of fairness, allowing it to assume different fairness models, without changing the design of the module controlling the utilization of the network. This design is more flexible in the sense that it allows for the development of a generic utility controlling module, abstracting away the fairness module.
- *Support concurrent applications*: Future sensor networks are likely to support multiple concurrent applications and also evolve to become multi-user systems [11]. To allow for such evolution, it is important that the transmission control protocols provide explicit support for the same. In these systems, it might be very well necessary to treat traffic from different

applications in a different manner. Abstracting the fairness module inherently facilitates such differentiation by allowing the fairness component to assume any fairness model, where available bandwidth can be disproportionately apportioned among competing flows.
- *Robustness*: Finally, it is important that any congestion control be robust to changes in underlying routing dynamics. Moreover, the algorithm also needs to adapt to changes in network topology which may be caused by nodes entering or leaving the system. For example, more nodes may be deployed or existing ones may die due to exhaustion of their energy reserve. All of these may require the rate allocations of the nodes to be changed dynamically without requiring external intervention.

4. Congestion control design

In this section, we develop a simple distributed algorithm for congestion control in sensor networks. Before we begin we first define a few terms.

Feedback delay: Feedback delay of a flow is the time interval, measured starting from the time node i starts transmitting flow f_i at a rate r_a to the time the control signal arrives and changes the rate to r_b.

Gateway node: A node is called a gateway node if it is one hop away from the sink (see Fig. 1).

Control interval: Control interval is the time period over which a node takes a control decision regarding the increase or decrease in the transmission rates of the flows originated by itself and of the flows being routed through it. In our implementation, this interval has been taken to be the average feedback delay of the flows passing through the gateway node.

4.1. Overview

Each node i, which originates flow f_i, maintains and sends the current rate r_i at which the flow is being injected into the network, and an estimate of the *feedback delay* of the flow by appending a congestion header to each packet being sent in the flow. Fig. 2 shows the congestion header. This allows the intermediate nodes to be aware of the transmission rates of the flows passing through it, and their feedback delays. In our mechanism, the nodes monitor their input traffic rate. Based on the difference between the link capacity and the input traffic rate, the nodes compute the desired increase or decrease in the rate of origination of the flows sharing the link. Note that the bandwidth that can be acquired by a flow f_i, originating from node i, is constrained by the *minimum* bandwidth that can be allocated to the flow by the nodes along the path of the flow. Thus, the effective transmission rate of f_i that needs to be
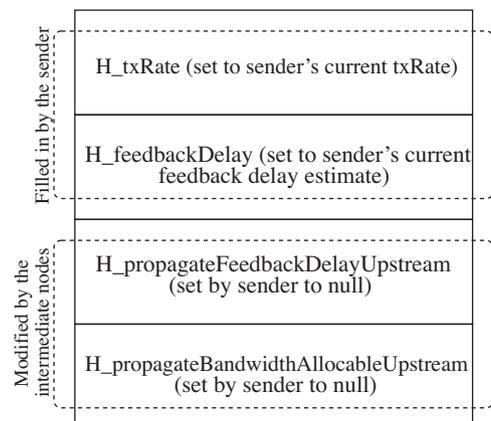

Fig. 2. Congestion header.

conveyed to node i is the minimum transmission rate that can be sustained by the nodes along the path of the flow. To enable this, we take advantage of the broadcast nature of the wireless medium by assuming that the transmission of a packet by a parent node can be heard by its children. When a node (say j) transmits a packet belonging to flow f_i, it writes into the header in the packet, the minimum of the bandwidths that can be allocated to f_i by itself and by the nodes downstream (nearer to the sink) to node j. Node j knows the minimum bandwidth allocated to f_i by its downstream nodes by overhearing transmission of its parent. Finally, the congestion signal that reaches the source node is, thus, the minimum bandwidth allocable to the flow based on the spare bandwidths of the nodes along the path of the flow. Next, we formally present the congestion control algorithm.

4.2. Congestion control algorithm

The congestion control algorithm can be described by the following steps executed at each node every control interval:

(1) Measure the average rate r_{out} at which packets can be sent from the node, the average aggregate input rate r_{in}, and Q, which is the minimum number of packets in the output queue seen by an arriving packet in a control interval.
(2) Based on the difference between r_{out} and r_{in}, and Q, compute Δr, which is the total change in aggregate traffic required as: $\Delta r = \alpha \times (r_{out} - r_{in}) - \beta \times \left(\frac{Q}{t_{CI}}\right)$ where, α and β are constants, and t_{CI} is the control interval.
(3) Apportion Δr into individual flows to achieve fairness.
(4) Compare the bandwidth computed for each flow with the bandwidths advertised by its parent. Use and propagate the smaller rate upstream.

An important point that needs to be clarified is when does the congestion control procedure gets invoked at the various nodes. The congestion control algorithm gets invoked every control interval at the gateway nodes. For any other node, the algorithm is invoked when the transmission rate of its parent changes. Notice that, this essentially makes the congestion control to be invoked at *all* the nodes every control interval. We will elaborate further on the rationale behind such a control behavior and on the estimation of the control interval later in Section 4.2.6.

The congestion control algorithm outlined above requires:

(1) estimation of average aggregate output rate.
(2) estimation of average aggregate input rate.
(3) computation of the total change in aggregate traffic required (Δr) to control efficiency.
(4) apportioning Δr into individual flows to obtain desired fairness.
(5) propagation of rate upstream.
(6) estimation of the control interval.

We now explain in detail the working of each one of the above mentioned requirements in the following sections.

4.2.1. Estimation of average output rate
Let t_{out} seconds be the time required to transmit a packet, measured starting from the time the packet was sent by the network layer to the MAC layer to the time when the MAC layer notifies the network layer that the packet was successfully transmitted. This is shown in Fig. 3, where we assume that the MAC protocol in use is CSMA/CA with an acknowledgement based scheme. Then, we note that the effective rate r_{out} packets per second is the inverse of the time interval t_{out} seconds, i.e., $r_{out} = \frac{1}{t_{out}}$.
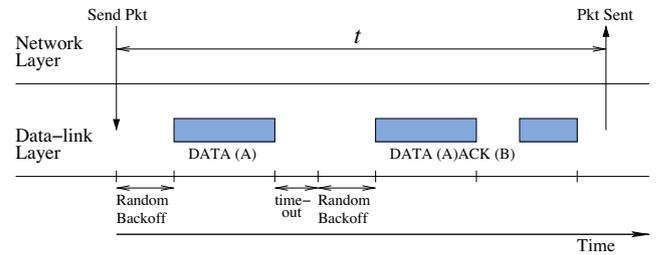

Fig. 3. Determination of time t taken to successfully transmit a data packet, considering CSMA as the MAC protocol. The sender of a packet is denoted in parenthesis.

The value of t_{out} obtained per packet transmitted is one particular instance of the average time taken to transmit a packet. We compute the average value by using the exponential moving average formula below,

$$\overline{t_{out}^i} = (\alpha_{out}) \cdot T_{out} + (1 - \alpha_{out}) \cdot \overline{t_{out}^{i-1}} \tag{1}$$

where, $\overline{t_{out}^i}$ is the exponential moving average value of the variable t_{out} in the ith iteration, α_{out} is the weight and T_{out} is the current value of the variable t_{out}. Calculating $r_{out} = \frac{1}{t_{out}^i}$ gives us the average rate at which packets can be transmitted from a particular node at any instant.

4.2.2. Estimation of average input rate
Let t_{in} seconds be the inter packet arrival time at a node, measured starting from the time a packet was enqueued to the time when the next packet is successfully enqueued. Then, the effective aggregate input rate r_{in} at a node is the inverse of the time interval t_{in}, i.e., $r_{in} = \frac{1}{t_{in}}$. We compute the average value of t_{in} using the exponential moving average formula below,

$$\overline{t_{in}^i} = (\alpha_{in}) \cdot T_{in} + (1 - \alpha_{in}) \cdot \overline{t_{in}^{i-1}} \tag{2}$$

where, $\overline{t_{in}^i}$ is the exponential moving average value of the variable t_{in} in the ith iteration, α_{in} is the weight and T_{in} is the current value of the variable t_{in}. Calculating $r_{in} = \frac{1}{t_{in}^i}$ gives us the average aggregate input rate at a node at any instant.

4.2.3. Controlling efficiency
In controlling efficiency, we seek to maximize link utilization, while minimizing buffer drop rates and persistent queues. The efficiency controlling component considers only the aggregate traffic and does not take into account fairness issues.

The efficiency controller computes the required increase or decrease of the aggregate transmission rate of the traffic (Δr) in a control interval (in packets per second). This is computed as:

$$\Delta r = \alpha \times (r_{out} - r_{in}) - \beta \times \left(\frac{Q}{t_{CI}}\right) \tag{3}$$

where, t_{CI} is the control interval of the node, α and β are constant parameters and Q is the persistent queue size. We use 0.4 and 0.226 as values of α and β respectively, based on the work done in [6]. Q is computed as the minimum number of packets seen by an arriving packet in a control interval. Note that the quantity, $(r_{out} - r_{in})$, can be positive, negative or equal to zero. When $(r_{out} - r_{in}) > 0$, the link is underutilized and positive feedback needs to be sent to increase the transmission rates of the flows. When $(r_{out} - r_{in}) < 0$, link is congested and negative feedback is required to decrease the transmission rates. For the case when $(r_{out} - r_{in})$ is equal to zero, i.e., the input capacity matches the link capacity, we have to provide feedback in a manner which drains the persistent

queue size Q. This is why the aggregate feedback has been made proportional to Q.

It is worth noting here that the values of the coefficients α and β used are for linear feedback systems with delay. α and β influence the stability of the feedback system, which can be studied from their open-loop transfer function in a Nyquist plot. The values of α and β used are such that the system satisfies the Nyquist stability criterion.

4.2.4. Controlling fairness

The task of the fairness controlling module is to apportion the feedback computed by the efficiency controlling module into the individual flows. For the sake of simplicity, in this section, we treat all flows equally. Section 4.3 discusses further on the issue of differentiable treatment of flows by the fairness controller. Though there could be other choices, we use AIMD to make the system converge to fairness. There are three cases:

- $\Delta r > 0$: In this case, the positive feedback to be sent is divided equally among the flows so that increase in throughput of all the flows is the same. If we assume that there are n flows comprising the route-through traffic of a node, then each one of the flows should get $1/(n+1)$th fraction of the spare bandwidth, considering that the node also originates traffic of its own. Thus, we can write:

$$r_i(t+1) = r_i(t) + \frac{\Delta r}{n+1} \qquad (4)$$

- $\Delta r < 0$: In this case, the negative feedback to be sent is allocated in such a way that the decrease in throughput of a flow is proportional to its current throughput. Thus,

$$r_i(t+1) = m \cdot r_i(t) \qquad (5)$$

  where, $0 < m < 1$. Here m is inversely proportional to the magnitude of Δr, $|\Delta r|$. This means that, as the discrepancy between the output rate and the input rate increases, the throughput of the flows is cut down more and more aggressively.
- $\Delta r = 0$: This case corresponds to the efficiency being near optimal. Bandwidth shuffling is done in this case, such that, the total traffic rate, and consequently the efficiency, does not change, yet the throughput of each individual flow changes gradually to approach the flow's fair share.

4.2.5. Propagation of rate upstream

As mentioned earlier, congestion control is invoked at the *gateway nodes* every control interval. For all the other nodes, congestion control is invoked when the transmission rate of the parent node changes. It is quite evident that, this, in effect, leads to congestion control being invoked at all the nodes every control interval. In order to propagate the congestion signal upstream, starting from the gateway nodes, we make use of the broadcast nature of the wireless medium, which enables a child node to overhear transmissions of its parent. When a node (say j), transmits a packet of a flow f_i, it writes into a header (H_propagateBandwidthAllocableUpstream as shown in Fig. 2) in the packet, the minimum of the bandwidths allocable to the flow by itself (which it computes when its congestion control is invoked) and that advertised by its parent. Note that, the bandwidth advertised by its parent is, in turn, the minimum of the bandwidth allocable to f_i by the nodes downstream to node j. Thus, the transmission rate for flow f_i that node j writes into the header of a packet of f_i, is the minimum bandwidth allocable to the flow among nodes downstream starting from node j. In this fashion, the feasible transmission rate, that ultimately reaches node i (which originates flow f_i), is the minimum bandwidth allocable to the flow by the intermediate nodes along the path of the flow.

4.2.6. Estimation of control interval

Control theory states that a controller must react as quickly as the dynamics of the controlled signal. Otherwise, the controller will lag behind the system being controlled and will be ineffective. Thus, the controller of our congestion control scheme makes a single control decision every average *feedback delay* period (averaged over the feedback delays of the flows passing through a node). This is motivated by the need to observe the results of the previous control decisions before attempting a new control. For example, if an intermediate node tells an upstream node to increase its transmission rate, the former should wait to see how much spare bandwidth remains before asking it to increase again.

As explained before, *feedback delay* of a flow is the time interval, measured starting from the time node i starts transmitting flow f_i at a rate r_a to the time the control signal arrives and changes the rate to r_b. Feedback delay of a flow is, thus, equal to the delay experienced by the packets of the flow to go from the source node to one of the gateway nodes plus the time taken by the congestion signal to propagate upstream from the gateway node back to the source node. Now, the time taken by the packets to travel from source nodes to the gateway nodes is comprised of the queueing delays of the packets plus the average time taken to transmit a packet successfully by the MAC layer at the various intermediate hops. Further, time taken by the congestion signal to propagate upstream from the gateway node to the source node is comprised of the summation of the inter packet arrival interval of the packets of the flow at the head of the output queue at the intermediate nodes and the time taken to transmit a packet successfully by the MAC layer at these nodes. To understand why, recall that the congestion signal propagates upstream by utilizing the broadcast nature of the wireless medium, enabling a child node to overhear transmission of its parent. Thus, when the minimum bandwidth allocable to a flow changes at a parent node, its child comes to know about it *at most* after the instantaneous inter packet arrival interval of the packets of the flow at the head of the output queue at the parent plus the time taken to transmit a packet by the MAC layer at the same.

From the above discussion it is clear that, if a flow k passes through n intermediate hops, numbered 1 through n, then the feedback delay of flow k is given by,

$$\sum_{i=1}^{n} \left(queueingDelay_i^k + interPktArrInterval_i^k + 2 \cdot macPktTxTime_i^k \right)$$

$$(6)$$

where, $queueingDelay_i^k$ is the average queueing delay at node i of the packets of flow k, $interPktArrInterval_i^k$ is the average time interval between two packets of flow k arriving at the head of the output queue, and $macPktTxTime_i^k$ is the average time taken by the MAC layer of node i to transmit a packet.

Now, what is required is to compute the above quantity in a distributed manner. For this, each node keeps a moving average of each one of the three terms in Eq. (6) for each flow passing thorough it. When a node (say j) transmits a packet of a flow k, it writes into a header (H_propagateFeedbackDelayUpstream as shown in Fig. 2) in the packet the summation of the three terms as estimated by itself for flow k and the value for feedback delay for flow k as advertised by its parent. The value for feedback delay advertised by its parent is in turn the summation of the three terms of the nodes downstream to j. In this fashion, the source node can be made aware of the feedback delay of the flow it is originating. The source node, on its part, maintains the feedback delay experienced by its flow and sends them out to the intermediate nodes via a header (H_feedbackDelay as shown in Fig. 2) in every packet.

As mentioned earlier, gateway nodes are those which are one hop away from the sink. These nodes have a more holistic view

of the network since traffic from every other node has to go via one of these nodes to get to the sink. The gateway nodes use the average feedback delay of the flows passing through them as the control interval. For the other nodes in the network, the congestion control procedure is invoked when the transmission rate of its parent changes. In this scheme, effectively, the congestion control algorithm gets invoked for all the nodes every average feedback delay of flows passing through the gateway node, which is the control interval.

4.3. Discussion

This section brings out some important points about the design of the congestion control algorithm.

4.3.1. Notes on the control laws

As mentioned earlier, a novel approach adopted in this paper is decoupling the modules controlling utilization and fairness. The utility controller uses a Multiplicative-Increase Multiplicative-Decrease (MIMD) law, increasing the traffic rate proportional to the spare bandwidth in the network. This allows the traffic to quickly acquire available bandwidth in the network. On the other hand, the fairness controller uses Additive-Increase Multiplicative-Decrease (AIMD), which has been shown to be able to converge to fairness [1]. Thus, our design facilitates each module to use a different control law, thereby portraying a more flexible approach.

4.3.2. The phase shifting effect

From a control theory perspective, the system forms a closed loop and is in constant oscillation. When the aggregate traffic input rate at a node becomes more than its aggregate output rate, the node decreases the bandwidth allocable to all its upstream nodes, including itself. This gets propagated through the network, and ultimately the queues start draining. This causes fewer nodes to transmit, thus reducing interference among neighboring nodes and the time taken to successfully transmit a packet decreases. The node, then advertises increase in bandwidth of the flows, and gradually the transmission rate of the flows increase, thus luring congestion. This cycle goes on repeating and the instantaneous transmission rates of the nodes fluctuate around the optimal value. This fluctuation has an important effect of *shifting phase* among neighboring nodes. Thus, the nodes will be injecting packets at different times. This helps reduce interference among neighboring nodes, thus, improving channel quality and the average retransmission count of the nodes at the MAC layer accordingly decreases.

4.3.3. Weighted fairness

As explained earlier the notion of fairness has been abstracted in our design, allowing it to assume different fairness models. For example, it is possible for the fairness module to implement *weighted fairness* [13], without changing the utilization controlling module. In *weighted fairness*, each node i is assigned a weight w_i. A rate allocation scheme is said to be fair if the normalized rate for each flow, $\frac{r_i}{w_i}$, is fair. This simply means that instead of assigning each flow an equal share of the spare bandwidth, we split up the available bandwidth proportionally to each flow's weight. However, in this paper, we treat all flows equally for the sake of simplicity, and do not investigate further the implementation of different fairness modules, leaving it as a future work.

4.3.4. Subset of senders

Note that in our algorithm, congestion control is invoked at nodes other than gateway nodes, when the transmission rate of the parent node changes. However, this would require all nodes in the network to be transmitting data for this algorithm to work.

**Table 1**
Parameter values used in the simulations.

Parameter	Values
Channel Bit-rate	38.4 kbps
Topology considered	10×10 grid
Link loss probability	0.1
Initial pkt generation rate	1 pkts/s
Queue Size	25
Data pkt size	64 bytes
Beacon pkt size	10 bytes
MAC ACK pkt size	5 bytes
Tx/Sensing Range	25 m
DIFS	50 μs
SIFS	10 μs
Slot time	20 μs
Maximum Retx threshold	7

In order to remove this constraint, when a node not transmitting any data detects a change in the transmission rate of its parent, it transmits a *dummy packet*, after invoking its own congestion control, indicating to its child node of a 'change' in its transmission rate.

4.3.5. Influence of MAC

Though we assume a sensor node to run the CSMA/CA protocol (which, for example, is used by the CC2420 radio stack found in TinyOS 2.x) to access the medium, our solution would still be valid if a node uses a MAC protocol that is based on a sleep/wake cycle. In case of such MACs, a node is required to transmit a dummy packet periodically (the periodicity is determined by the control interval), for indicating to its child node of a 'change' in its transmission rate.

Since we are appending 4 bytes of header, the packet size grows from 60 to 64 bytes; thereby incurring an increase of 6.66% in the packet size. Experiments in [4] have shown a linear relation between number of data bytes transmitted and energy consumed. Thus, the increase in energy consumption in transmitting a packet is also 6.66%. Note that, the power to transmit a packet also
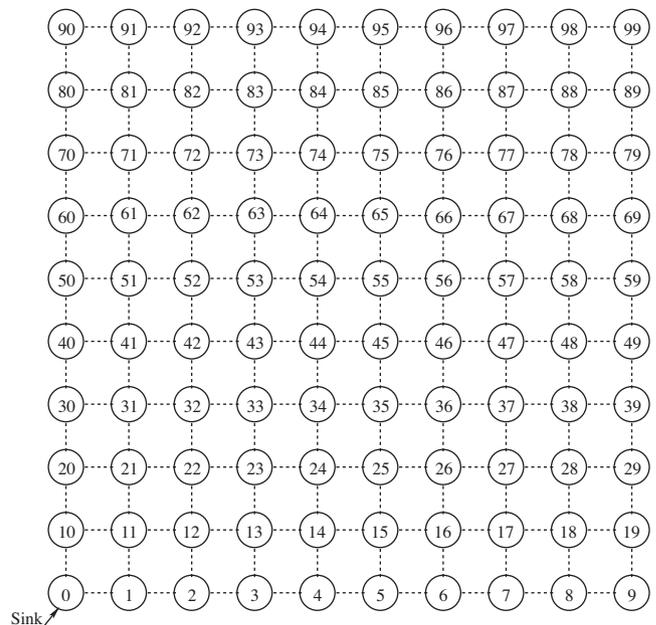
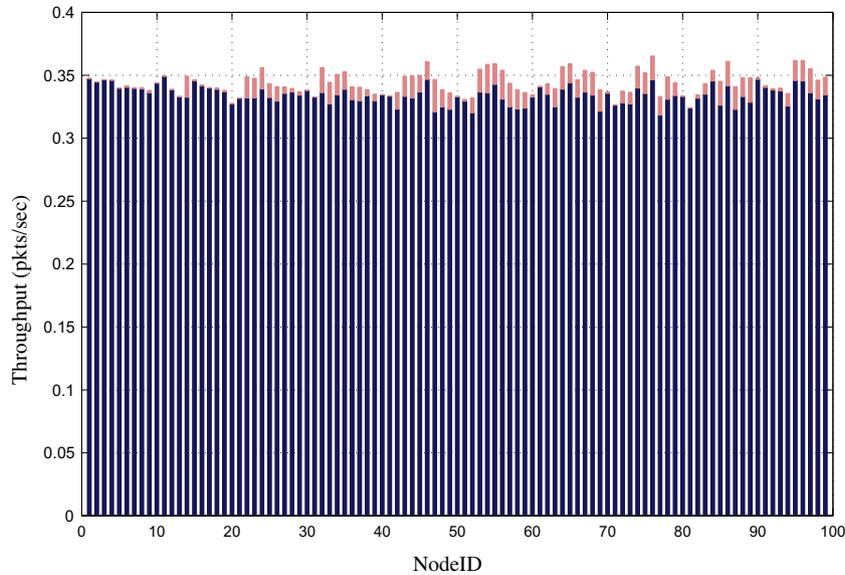**Fig. 4.** 10×10 grid used in the simulations.

Fig. 5. Per flow goodput in the 100 node simulation.

depends on factors other than the size of the packet. The total energy consumed depends not only on the energy consumed during transmission but also during contention and also when a node is idle (listening) mode. While energy is spent on all these activities, the predominant energy consumption is due to transmission of bits. In this paper, we consider only this dominating factor. For transmission of a single bit, E_b is consumed when the noise floor is N_O.

Finally, our congestion control algorithm works well in regimes where hop-by-hop loss recovery recovers from most end-to-end packet losses. Many real world scenarios use hop-by-hop loss recovery to improve delivery ratio [10,15]. Outside this regime, the congestion control does not get feedback about the actual goodput the nodes receive, and thus cannot guarantee globally fair allocation of goodput.

The next section evaluates the performance of the algorithm via extensive simulations.

5. Performance evaluation

To evaluate the performance of the congestion control algorithm, we implemented a packet level simulator in Java. The simulator is event-driven and implements CSMA/CA as the MAC protocol.

Table 1 shows the parameter values used in the simulations. We assumed that the maximum communication channel bit rate is 38.4 kbps, ACK packet is 5 bytes, beacon packet is 10 bytes and data packets are 64 bytes. The simulations have been carried out on a 100 node network, a 10×10 grid, with the sink at the lower left corner. The topology is shown in Fig. 4. The nodes have been initialized with an initial packet generation rate of 1 packets/s. Each link has a constant loss probability of 0.1, which is in addition to the loss caused by interference. Since ARQ is implemented, packets dropped due to interference or queue overflow are retransmitted. The weights used in the exponential moving average calculation of the output and input rate (see Sections 4.2.1 and 4.2.2) is set to
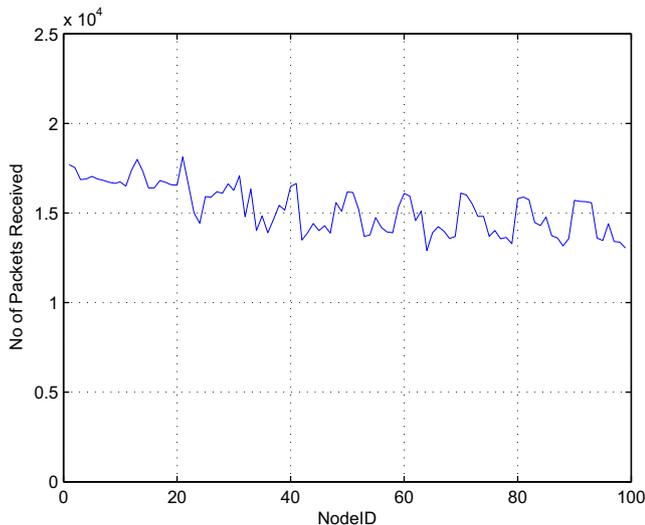

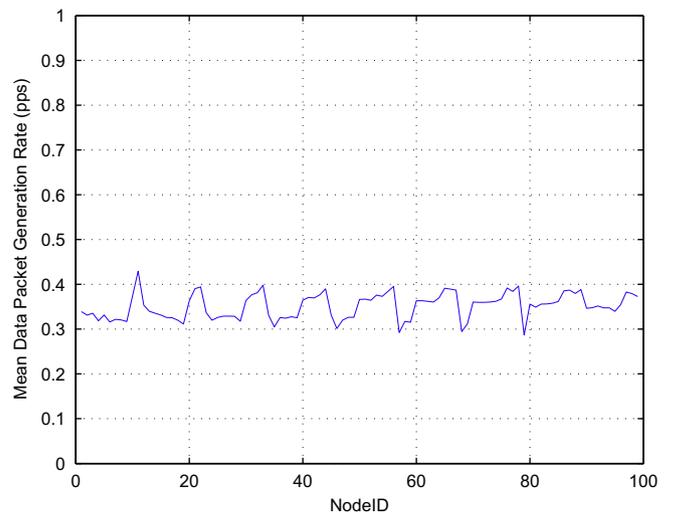**Fig. 6.** Number of packets received from each node by the sink.

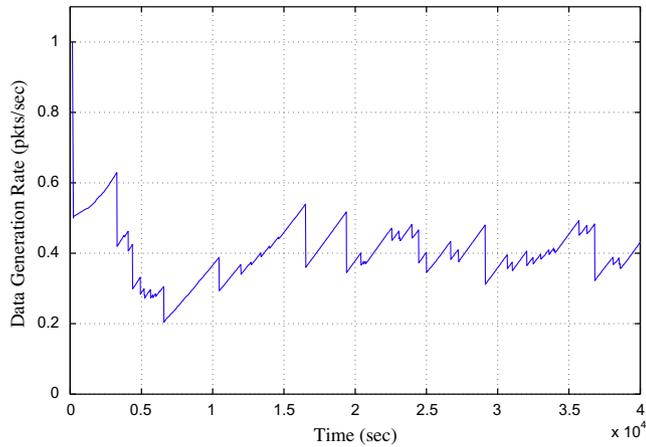**Fig. 7.** Mean data packet generation rates per node.

Fig. 8. Data packet generation rate fluctuation (Node 2).

0.1. Each node has a queue to hold packets forwarded by its children, as well as packets generated by itself. This queue has been set to hold a maximum of 25 packets.

5.1. Goodput and fairness

Fig. 5 shows the per-flow goodput received at the sink. Each bar represents the average rate of transmitted packets from the corresponding node. The darker section of each bar represents the average rate at which packets from that node were received at the base station. Packet losses account for the rest (the remaining lighter section of the bar). We make an important observation from this graph. This figure validates the basic design of the congestion control algorithm, and shows that nodes receive approximately fair rates and fair goodput.

Fig. 6 brings out the fact that the sink, over a long term, receives an equal number of packets on an average from the nodes. Without congestion control, packets received from nodes further than 2 hops starts falling drastically.

5.2. Data generation rate oscillations

Without proper congestion control, the nodes nearer to the base station will tend to have higher transmission rates than nodes further away from the sink since the traffic from the former is competing for lesser number of resources. One of the design criteria of the congestion control algorithm is to receive equitable amounts of data from all the nodes at the sink, which in other words requires, on an average, equal sending rates from the nodes. Fig. 7 tends to bring out this fact, wherein we observe that the long term average transmission rates for all the nodes fall within a narrow range of values. The spikes that we observe in Fig. 7 are prevalently at nodes 11, 22, 33, 44, 55, 66, 77, 88 and 99 (Fig. 4 shows the grid used in the simulations). This is due to the fact that traffic from relatively fewer number of nodes follow this route. Since the bandwidth along a route is split up equally among competing flows, hence these nodes tend to achieve a higher average transmission rate than the other nodes in the network.

Fig. 8 shows the variation of data packet generation rate at node 2 with time. The graph begins at 1 packets/s since the nodes are initialized at that rate. From the graph we can see that the congestion control scheme is able to adapt and obtain the effective transmission rate quickly. As can be seen from the graph, the transmission rate of the node oscillates around a mean. This oscillation around the mean causes phase shifting among neighboring nodes, as explained in Section 4.3, thus reducing interference and improving transmission rate at the MAC layer.

5.3. Subset of senders

Fig. 9 shows the per flow goodput with a subset of senders. Those nodes whose ID's are multiples of 3 were only allowed to transmit data. As can been seen from the graph, all nodes receive a fair rate. Note that the per flow goodput is significantly higher than when all nodes transmit. This is because the protocol adapts to the increased overall available capacity and allocates it fairly to the transmitting nodes.

5.4. Link layer retransmissions

Fig. 10 shows the effect of turning off link-layer retransmissions. In this figure, notice that the rate allocation is fair, but the goodput that each node gets is not (as shown by the darker section of each bar). This shows that the congestion control algorithm is dependent on link layer retransmissions to achieve fair goodput for all nodes.
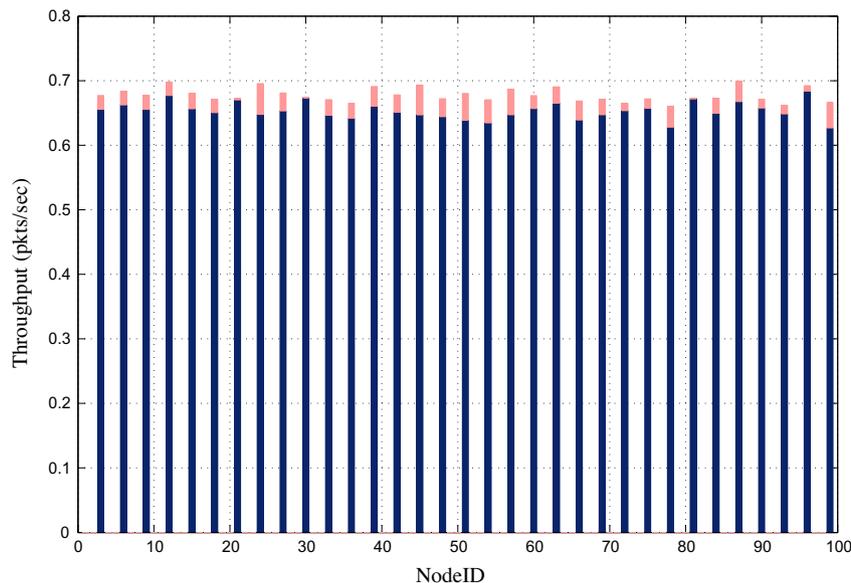

Fig. 9. Per flow goodput with only a subset of senders. Nodes whose ID's are multiples of 3 were only allowed to transmit data.
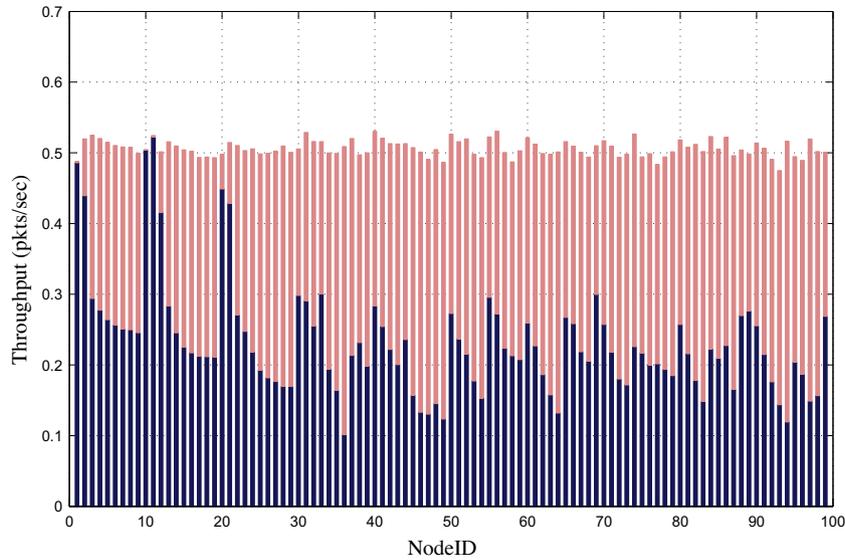
Fig. 10. Per flow goodput with no link layer retransmissions.

Fig. 11 shows that without congestion control each packet is retransmitted due to interference a maximum of about 6 times, a number dependent on many factors such as the topology, data packet generation rate and size of the network. In general, number of retransmissions, and therefore losses, increases with the depth of the network. The graph where congestion control is implemented shows a small number of retransmissions per packet and this number roughly stays constant for all nodes in the network.

5.5. Comparison with hop-by-hop flow control

We will now compare the performance of our algorithm with that of hop-by-hop flow control. Hop-by-hop flow control has been proposed for sensor networks in [2,16]. In this scheme, each sensor sets a congestion bit in the header of every outgoing packet. By taking advantage of the broadcast nature of the wireless medium, congestion feedback is provided to all nodes in a radio neighborhood with every transmission. Hop-by-hop flow control has two components–congestion detection and congestion mitigation. We next discuss implementation of each of these two components.

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 certain threshold, the congestion bit of outgoing packets is set; otherwise the congestion bit is cleared. We set this threshold to 25% in our implementation, i.e., the congestion bit is set when the amount of available space in the output queue is less than 25%.

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 over flowing. 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. If a path experiences persistent congestion, hop-by-hop backpressure will eventually reach the source, allowing application-level flow control. In such a case, the application adjusts its sending rate via an AIMD controller.

Fig. 12 shows the per-flow goodput received at the sink, in a simulation with 36 nodes. Fig. 12(a) shows the goodput achieved using our proposed congestion control algorithm, while Fig. 12(b) shows the goodput achieved using hop-by-hop flow control. In both figures, each bar represents the average rate of transmitted packets from the corresponding node. The darker section of each bar represents the average rate at which packets from that node were received at the base station (goodput). Packet losses account for the rest (the remaining lighter section of the bar).

As can be clearly seen from the two figures, the goodput achieved by using our proposed congestion control mechanism is higher than that achieved by using hop-by-hop backpressure. The loss rate is significantly higher when using hop-by-hop backpressure. Moreover, note that, nodes further away from the sink suffer more losses. This is because hop-by-hop backpressure is more "sluggish" in propagating the congestion signal to the source nodes, making the network unable to react to congestion in a timely manner. Also, note that, the average transmission rates of nodes deeper in the network is higher for hop-by-hop flow control than that obtained by using our proposed congestion control mechanism, while the loss rate is higher for the former. This shows that our mechanism can operate the network at an optimal point that maximizes network throughput.
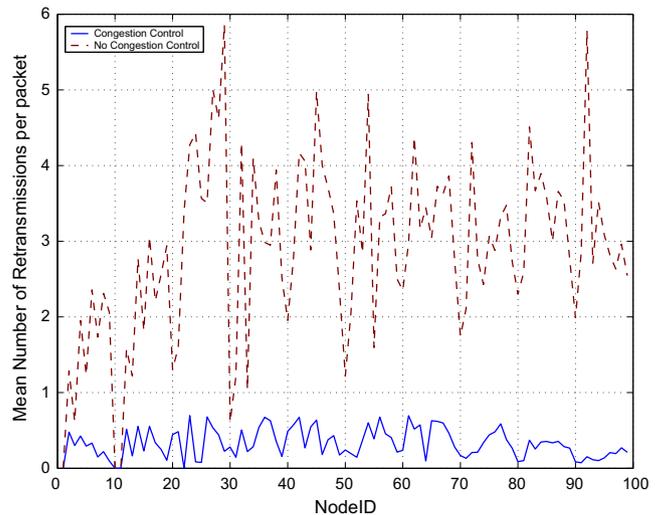

Fig. 11. Mean retransmissions per node due to interference.
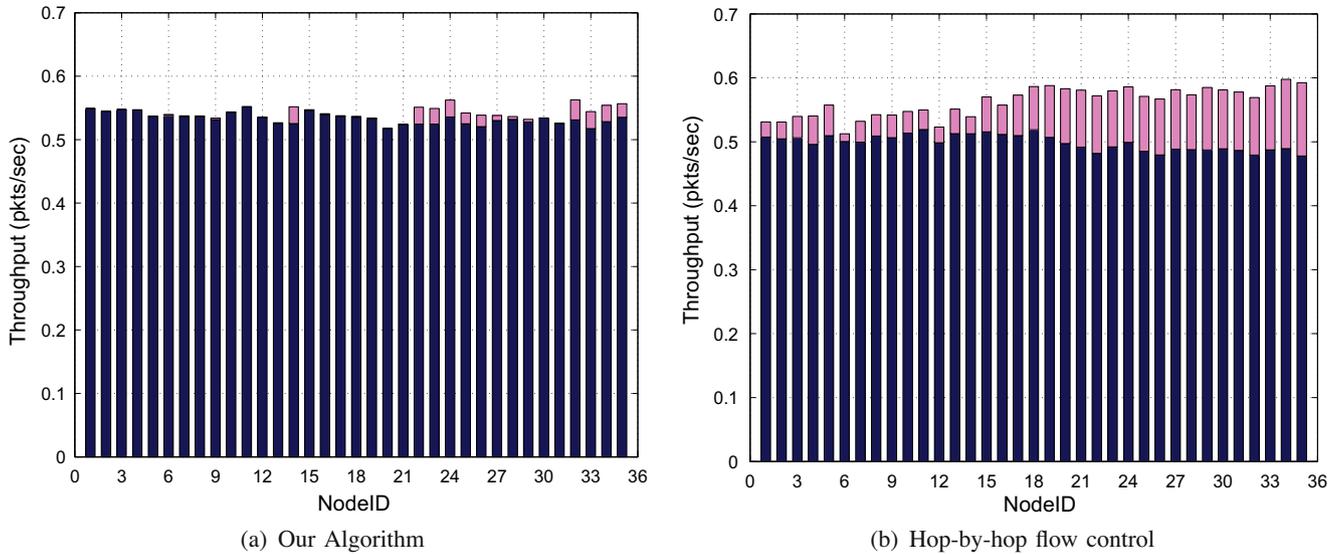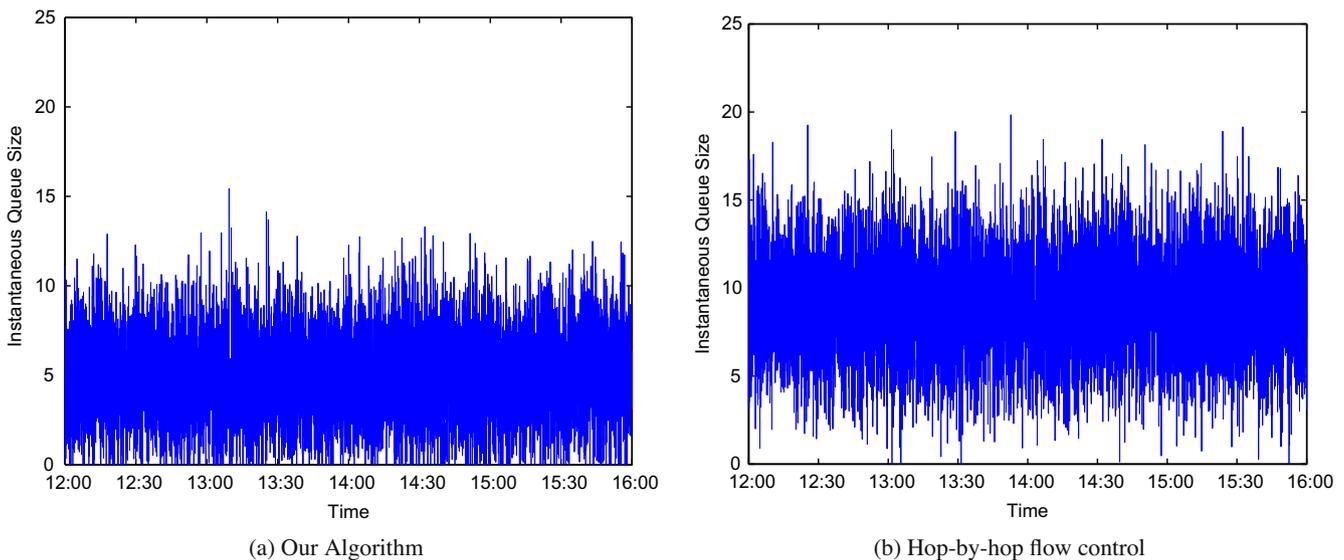
Fig. 12. Per flow goodput.

Fig. 13. Instantaneous queue sizes.

Fig. 13 shows the instantaneous queue sizes at every node in a 100 node simulation. Fig. 13(a) shows the instantaneous queue sizes when using our congestion control mechanism, while Fig. 13(b) shows the same when using hop-by-hop flow control. As the figures suggest, the queue sizes when using our proposed mechanism are lesser than that obtained when using hop-by-hop flow control. This implies that the delay in delivering a packet to the sink is lesser when using our proposed mechanism in comparison to the delay a packet experiences when using hop-by-hop flow control. Also, the lower queue sizes and higher throughput achieved (Fig. 12) when using our algorithm imply that our proposed mechanism can operate the network at an optimal point, beyond which congestion sets in and network performance degrades.

Fig. 14 shows the average number of retransmissions per packet at each node due to interference when using our proposed mechanism as well as when using hop-by-hop flow control, in a 36 node simulation. As can be clearly seen from the figure, the average number of retransmissions per packet when using our algorithm is considerably less than that needed for hop-by-hop flow control.

This is because of the effect of phase shifting among neighboring nodes that our algorithm induces (refer Section 4.3). This makes nodes to inject packets at different times, thereby reducing interference among nodes and subsequently decreasing average retransmission count of a packet.

6. Conclusions

The paper has presented a distributed algorithm for congestion control in wireless sensor networks that seeks to assign a fair and efficient rate to each node. The algorithm requires each node to monitor their aggregate input and output traffic rate, based on the difference of which each node decides to increase or decrease the transmission rates of itself and its upstream nodes. The utilization controlling module computes the total increase or decrease in traffic rate. The fairness module decides on how exactly to apportion the total change in traffic rate required among the flows. The utilization controller is indifferent to it, thus abstracting the notion
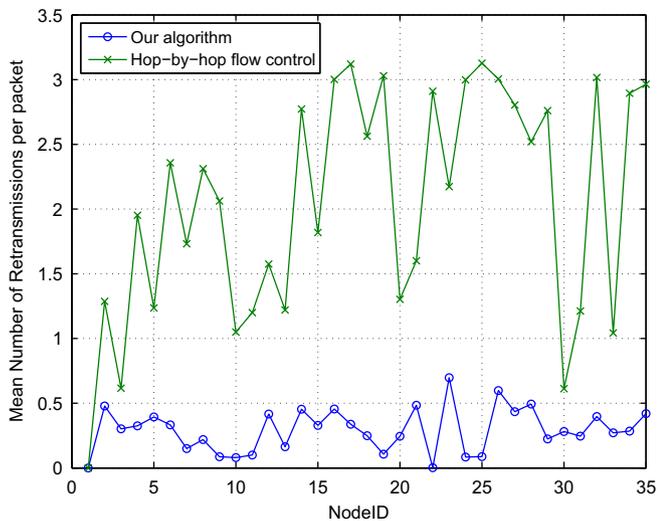
Fig. 14. Mean retransmissions per node due to interference.

of fairness, and allowing the fairness module to assume differential bandwidth allocation policies. The congestion control is invoked at each node every average feedback delay of the flows passing through the gateway node, which we define as the control period. By piggy-backing control information on data packets and by utilizing the broadcast nature of the wireless medium, the algorithm does not require the need for additional control packets. We tested our algorithm on an event-driven packet level simulator. The results indicate that the proposed congestion control mechanism can achieve remarkably high goodput, is able to attain fairness for all nodes in the network, and can acquire the optimal transmission rates quickly. Finally, our solution is independent of the underlying routing algorithm and can adapt to changes in the routing tree.

References

[1] D. Chiu, R. Jain, Analysis of the increase and decrease algorithms for congestion avoidance in computer networks, Computer Neworks and ISDN Systems 17 (1) (1989) 1–14.

[2] T. van Dam, K. Langendoen, An adaptive energy-efficient MAC protocol for wireless sensor networks, in: Proceedings of the ACM Sensys Conference, November 2003, pp. 171–180.

[3] C.T. Ee, R. Bajcsy, Congestion control and fairness for many-to-one routing in sensor networks, in: Sensys 2004.

[4] L.M. Feeney, M. Nilsson, "Investigating the energy consumption of a wireless network interface in an ad hoc networking environment, IEEE INFOCOM (2001) 1548–1557.

[5] B. Hull, K. Jamieson, H. Balakrishnan, Mitigating congestion in wireless sensor networks, in: Sensys 2004.

[6] D. Katabi, M. Handley, C. Rohrs, Congestion control for high bandwidth-delay product networks, in: Sigcomm 2002.

[7] V. Kottapalli, A. Kiremidjian, J.P. Lynch, E. Carryer, T. Kenny, K. Law, Y. Lei, A two-tier wireless sensor network architecture for structural health monitoring, in: Proceedings of SPIE 2003.

[8] K. Mechitov, W.Y. Kim, G. Agha, T. Nagayama, High-frequency distributed sensing for structure monitoring, in: Proceedings of First International Workshop on Networked Sensing Systems (INSS), 2004.

[9] V. Michopoulos, L. Guan, I. Phillips, A new congestion control mechanism for WSNs, in: IEEE International Conference on Computer and Information Technology, 2010, pp. 709–714.

[10] J. Paek, K. Chintalapudi, J. Cafferey, R. Govindan, S. Masri, A wireless sensor network for structural health monitoring: performance and experience, in: EmNetS 2005.

[11] J. Paek, R. Govindan, RCRT: rate-controlled reliable transport for wireless sensor networks, in: Sensys 2007.

[12] M. Rahimi, R. Baer, J. Warrior, D. Estrin, M.B. Srivastava, Cyclops: In situ image sensing and interpretation in wireless sensor networks, in: SenSys 2005.

[13] S. Rangwala, R. Gummadi, R. Govindan, K. Psounis, Interference-aware fair rate control in wireless sensor networks, in: Proceedings of ACM SIGCOMM Symposium on Network Architecture and Protocols, 2006.

[14] Y. Sankarasubramaniam, O. Akan, I. Akyildiz, Event-to-sink reliable transport in wireless sensor networks, in: Proceedings of the 4th ACM Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc 2003, pp 177–188.

[15] R. Szewczyk, A. Mainwaring, J. Polastre, D. Culler, An analysis of a large scale habitat monitoring, in: Sensys 2004.

[16] C. Wan, S.B. Eisenman, A.T. Campbell, CODA: Congestion detection and avoidance in sensor networks, in: First ACM conference on Embedded Networked Sensor Systems, 2003.

[17] A. Warrier, S. Janakiraman, S. Ha, I. Rhee, DiffQ: Practical differential backlog congestion control for wireless networks, in: INFOCOM 2009, pp. 262–270.

[18] R. Vannier, I. Gurin Lassous, Towards a practical and fair rate allocation for multihop wireless networks based on a simple node model, in: 11th ACM/IEEE International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM), Vancouver, Canada, October 2008.

[19] A. Woo, D. Culler, A Transmission control scheme for media access in sensor networks, in: Seventh Annual International Conference on Mobile Computing and Networking, July 2001, pp 221–235.

[20] A. Woo, T. Tong, D. Culler, Taming the underlying challenges of reliable multihop routing in sensor networks, in: SenSys 2003.

[21] W. Ye, J. Heidemann, D. Estrin, An energy-efficient MAC protocol for wireless sensor networks, in: Infocom 2002.

[22] X. Yin, X. Zhou, R. Huang, Y. Fang, S. Li, A fairness – aware congestion control scheme in wireless sensor networks, IEEE Transactions on Vehicular Technology 58 (9) (2009) 5225–5234.

[23] M. Zawodniok, S. Jagannathan, Predictive congestion control protocol for wireless sensor networks, IEEE Transactions on Wireless Communications 6 (11) (2007) 3955–3963.