Interference-Aware Fair Rate Control in Wireless Sensor Networks

Sumit Rangwala, Ramakrishna Gummadi, Ramesh Govindan, Konstantinos Psounis
University of Southern California
{srangwal, gummadi, ramesh, kpsounis}@usc.edu

ABSTRACT

In a wireless sensor network of N nodes transmitting data to a single base station, possibly over multiple hops, what distributed mechanisms should be implemented in order to dynamically allocate *fair* and *efficient* transmission rates to each node? Our interference-aware fair rate control (IFRC) detects incipient congestion at a node by monitoring the average queue length, communicates congestion state to exactly the set of *potential interferers* using a novel low-overhead *congestion sharing* mechanism, and converges to a fair and efficient rate using an AIMD control law. We evaluate IFRC extensively on a 40-node wireless sensor network testbed. IFRC achieves a fair and efficient rate allocation that is within 20-40% of the optimal fair rate allocation on some network topologies. Its rate adaptation mechanism is highly effective: we did not observe a single instance of queue overflow in our many experiments. Finally, IFRC can be extended easily to support situations where only a subset of the nodes transmit, where the network has multiple base stations, or where nodes are assigned different transmission weights.

Categories and Subject Descriptors: C.2.1 [Computer Communication Networks]: Wireless communication, C.3 [Special-Purpose and Application-Based Systems]: Embedded Systems

General Terms: Design, Algorithms

Keywords: Congestion Control, Rate Control, Fairness, Wireless, IFRC, Sensor Network

1. INTRODUCTION

Congestion control has been an active area of networking research for several decades now. Relatively less attention has been paid to congestion control in the emerging area of wireless sensor networks. In this paper, we examine the following simple but important problem: In a sensor network of N sensor nodes each transmitting data, possibly over multiple hops, to a base station, what distributed mechanisms must the sensor network implement in order to dynamically allocate *fair* and *efficient* transmission rates to each node?

One motivation for this problem comes from the application of wireless sensor networks for structural health monitoring [17, 21, 22]. In this application, a set of sensors is deployed on a large civil structure (such as a building or a bridge). Each sensor continuously measures structural vibrations at several hundred samples a second, a rate that is comparable to the nominal data rate of today's low-power sensor radios. When a significant response is detected, as when the structure is vibrated using a mechanical shaker, every sensor transmits a time series of recorded samples to a base station. Civil engineers find an application such as this to be extremely useful for conducting short to medium term experiments on test structures; they analyze the structural response to derive models of the structure, or assess the level of damage in the structure. In the absence of rate control, this sensor network could suffer congestion collapse.

A more general motivation comes from an evolved understanding of the structure and application of sensor networks. Early sensor network designs assumed a flat network of sensors supporting low-rate periodic sensing. More recently, *tiered* sensor networks [28] have been proposed for use in high data-rate applications (*e.g.,* acoustic [2, 8], imaging [23]). In tiered networks, the lower-tier consists of tiny wireless sensors that transmit data to the closest upper-tier node (usually an embedded 32-bit system with an 802.1x radio) [7]. In such networks, when an event is sensed, a relatively large number of nodes might wish to transmit significant volumes of data (either raw samples, or processed information) along one or more *trees* towards base stations. Rate control will prevent congestion collapse in these situations.

In this paper, we examine the design of a distributed and adaptive mechanism for fair and efficient rate control in wireless sensor networks. In general, the design of such a mechanism is complicated by multiple nodes concurrently accessing a time varying, shared wireless channel. As such, the channel arbitration used by the MAC layer and the quality of paths determined by the routing protocol can impact the quality of any solution to rate control. For simplicity, we build our work upon the sensor network *de facto* standard MAC layer (CSMA) and routing layer (link-quality based path selection). We defer to future work the examination of an optimal "cross-layer" design—one which jointly designs the MAC layer, the routing layer, and a rate allocation scheme.

Within our chosen framework, one important challenge is to design a mechanism by which each node can locally detect all flows that can contend for channel capacity, fairly adapt its own rate such that the capacity is not exceeded, and signal all relevant flows to do so as well. In general, such interference-aware rate allocation in wireless networks is difficult for arbitrary communication patterns [14]. As we show in this paper, we can leverage the tree-based traffic pattern prevalent in wireless sensor networks to obtain a distributed, adaptive, and fair rate control mechanism. Specifically,

This material is based upon work supported by the National Science Foundation under Grant No. 0520235.

SIGCOMM'06, September 11–15, 2006, Pisa, Italy.

we make the following contributions:

Design: Our interference-aware fair rate control (IFRC) technique (Section 4) is a collection of inter-related mechanisms for distributed and adaptive fair and efficient rate allocation for tree-based communication in wireless sensor networks. IFRC monitors average queue lengths to detect incipient congestion, and uses an AIMD control law to ensure convergence to fairness. Our main contribution is the design of a low-overhead, yet surprisingly efficient, *congestion sharing* mechanism. Congestion sharing itself is not a new idea; it has been used to achieve MAC-layer fairness [3], and to help TCP connections exchange network state [1]. However, designing such a mechanism in the context of sensor networks is not a trivial extension of prior efforts: to achieve fairness, all flows (even those originating at a distant node) that pass through a congested node need to be throttled, and rapidly signaling all relevant nodes is a non-trivial task. What is more, any flow that affects the rate at which the congested node sends data to the base station may need to be throttled, but such flows may not traverse the congested node at all. Precisely identifying the set of these flows is an important contribution of this paper.

Analysis: We analyze the steady state behavior of IFRC to rigorously derive IFRC parameters for stability and fast convergence. We validate our analysis with experiments which demonstrate that parameter choices within the bounds predicted by our analysis results in a stable system, while parameter choices outside this bound result in instability.

Implementation and experimentation: We have implemented IFRC in TinyOS (the widely-used sensor node OS), and demonstrate, using extensive experiments on a 40-node testbed (Section 6), that IFRC achieves fair rate control. IFRC's performance is within 20–40% of the maximum sustainable fair rate using a fixed tree and a CSMA MAC layer. IFRC can be trivially extended to more general versions of our problem: where each sensor node sends to the nearest of many base stations, where only a subset of the nodes transmit data, or where an application might require a *weighted* fair rate allocation. Finally, IFRC's rate adaptation is highly effective: we noticed no packet drops due to queue overflows even with relatively small buffer sizes in *any* of our experiments.

2. RELATED WORK

IFRC draws inspiration from the Internet congestion control literature, and borrows liberally from the work on TCP [13] and active queue management [6, 18]. Sharing congestion information has been employed in other contexts: for adjusting MAC-layer backoffs to ensure fair channel access [3], and for sharing path congestion state among Internet flows to improve TCP performance [1]. However, our problem setting is quite unlike any considered in these areas; to our knowledge, the problem of achieving a fair and efficient rate when many wireless nodes send data, possibly over multiple hops, to a base station has not been studied in either the Internet or ad-hoc networks literature before. Internet fair queuing mechanisms [5, 20] do not extend easily to shared wireless links. Much of the literature on TCP for ad-hoc wireless networks [9, 16, 25] focuses on mechanisms that distinguish between packet loss due to routing dynamics or wireless packet corruption, and packet loss due to congestion. The one exception to this is the work of Xu *et al.* [33], which extends RED to improve TCP fairness in wireless ad-hoc networks. They introduce the idea of a distributed queue comprised of all the nodes that contend for a channel. Each node locally computes its RED drop probability based on this distributed queue. This implicit congestion sharing is similar to our mechanism; however, as we show later, the set of nodes contributing to congestion at a node extends beyond the one-hop neighborhood of a node, and IFRC's congestion sharing mechanism takes this into account.

Prior work in the sensor networks literature has looked at two qualitatively different problems: *congestion mitigation*, and *congestion control*. Broadly speaking, mitigation attempts to solve the following problem. Consider a sensor network in which each sensor is tasked to periodically transmit samples at some fixed rate. When the aggregate sensor traffic exceeds network capacity, how should nodes *regulate* their transmissions such that the network goodput and fairness *degrade* gracefully (rather than exhibit congestion collapse at a point close to network capacity)? This is qualitatively different from congestion control, which seeks to *find* an optimal fair rate allocation to sensor nodes that is also maximally *efficient* (so that, when nodes send at this rate, the network is fully utilized, and the per node goodput is close to the sending rate).

Fusion [11] is a congestion mitigation scheme that uses queue lengths to measure levels of congestion, as we do, but also uses a different set of mechanisms than the congestion sharing mechanism we have considered; it applies hop-by-hop backpressure using a token-based regulation mechanism, and a prioritized medium access layer that allows congestion at local nodes to drain quickly. With this combination, Fusion achieves higher goodput and fairness at high offered loads than in the absence of these mechanisms. CODA [30] is another congestion mitigation strategy that uses slightly different mechanisms than Fusion. It senses both channel and buffer occupancies for measuring congestion levels; Fusion uses only buffer occupancy because, to a first order, a CSMA MAC layer with link-layer retransmissions will cause congestion to be reflected in backed up buffers. Beyond that, CODA considers two strategies: open-loop back pressure for transient congestion, and an end-to-end acknowledgment based approach for persistent congestion. Unlike Fusion, CODA does not explicitly focus on per-source fairness. Because they are focused on congestion mitigation, neither Fusion nor CODA implements an AIMD control law that IFRC uses to converge to a fair and efficient rate. Also, instead of their explicit hop-by-hop back-pressure, in IFRC, nodes exchange congestion indicators, and distributedly converge to a fair and efficient rate.

IFRC is closer in spirit to prior work on sensor network congestion control. In early work, Woo *et al.* [31] examine an AIMD rate adjustment strategy in which the additive increase is proportional to the number of descendants of a node, and multiplicative decrease is performed whenever a node detects (by promiscuously listening) that its parent is unable to forward its traffic. Thus, a parent indicates congestion by dropping some traffic from its child, forcing the latter into multiplicative decrease. IFRC is different from this work in that it detects incipient congestion and applies aggressive rate cutting to avoid dropping packets. It also employs a more sophisticated congestion sharing strategy beyond just a signal from a parent to its children.

ESRT [24] allocates transmission rates to sensors such that an application-defined number of sensor readings are received at a base station, while ensuring that the network is uncongested (and therefore efficient). Unlike IFRC, ESRT's rate allocation is centrally computed at the base station: periodically, the base station counts the number of received sensor readings and re-tasks the sensors by broadcasting a new transmission rate. ESRT uses a sophisticated control law based on empirically derived regions of operation, and does not attempt to find a rate allocation that is maximally efficient.

More recently, Ee and Bajcsy [4] have designed a congestion control scheme in which each node estimates the channel capacity

by measuring the time to transmit a packet, and then divides up the available capacity among the nodes in its subtree. Each node enforces overall fairness by using a non-work-conserving weighted-fair queue for each of its children. Besides mechanistic differences (IFRC does not measure channel capacity by measuring the time to transmit a packet because that measurement can be skewed by local packet processing overhead), IFRC is fundamentally different from this scheme since it achieves both fairness and *efficiency*. By contrast, this scheme's non-work-conserving scheduler does not promote efficiency; nodes whose flows do not pass through the congested regions cannot fully utilize the available bandwidth.

Several other pieces of work in the sensor and ad hoc network literature are tangentially related to IFRC. Sridharan and Krishnamachari [26] explore the static allocation of fair and efficient rates for a TDMA-based network. Li *et al.* [15] demonstrate that fairness at the medium access layer does not lead to transport layer fairness. Zhang *et al.* [35] examine priority adjustment techniques to reduce the retransmission priorities of unacknowledged packets, with the aim of improving overall goodput. Jain *et al.* [14] analyze the tractability and performance benefits of interference-aware transmission scheduling in multi-hop wireless networks, but do not consider distributed rate control mechanisms. Finally, several pieces of work have considered reliable point-to-point or point-to-multipoint delivery in sensor networks [10, 19, 27, 29]; we hope to leverage some of this work in the design of an end-to-end reliability mechanism that would complement IFRC.

3. MOTIVATION AND DEFINITIONS

Consider a network of N sensor nodes, with each node uniquely identified by an integer in the range $1 \ldots N$. In the simplest version of our problem, each node has infinite data to send to a single base station. This data can traverse multiple hops before reaching the base station. Thus, a sensor node originates traffic (is a *source*), and may forward traffic sent by other nodes. The traffic originated by source i and sent to the base station is the i-th flow, denoted by f_i. We seek to adaptively assign a *fair and efficient* rate r_i to f_i (or equivalently, to node i). Specifically, r_i is *the rate at which node i sources flow f_i*, and *does not include the rate at which node i forwards traffic*. In Section 4.2, we explore other problem formulations: supporting multiple base stations; allowing only a subset of the nodes to send; and, assigning different priorities to flows.

We assume that the sensor nodes run a contention-based medium access layer. The default MAC layer in TinyOS (the widely-used sensor node operating system) uses carrier-sense for collision avoidance, and our implementation is built upon this MAC. Our design can be extended to other contention-based MACs, such as those that use RTS/CTS [34]. We have not considered whether IFRC extends easily to token-based and TDMA MACs.

We also assume that the sensor nodes run a routing protocol [32] that builds a tree to the base station. The solid lines in Figure 1 depict such a tree. The correctness of IFRC is not sensitive to the particular choice of routing protocol, but its performance is. In general, IFRC will result in higher overall throughput on routing protocols that use link-quality metrics to establish the tree than on those that do not. In what follows, we assume that a sink tree has been constructed by the routing protocol, and, for ease of exposition, that the sink tree is stable. IFRC is designed to adapt to changes in the underlying routing tree.

Finally, we assume that the MAC layer provides link-layer retransmissions. Our current IFRC design performs well in a regime where the wireless loss rates on the tree links are such that link-layer retransmissions recover from most packet losses. Outside this regime, IFRC needs an end-to-end feedback mechanism to determine the effective goodput that each node receives. We explore the impact of wireless loss rates on IFRC's performance in Sections 4.3 and 6.2.

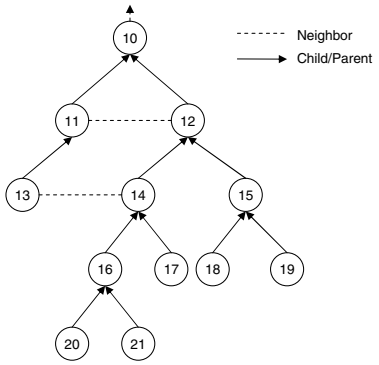

Figure 1: An example routing sub-tree. Dashed lines represent a neighbor that is neither a parent nor a child of a node.

3.1 Fair and Efficient Rate Allocation

What do we mean by a *fair and efficient* rate? Intuitively, we would like each flow to receive a fair share of the total channel capacity. However, in non-uniform deployments, spatial reuse effectively defines several wireless contention domains, in each of which the fair share may be different depending upon the number of flows traversing the domain. IFRC's goal is to assign to each flow *at least* the most congested fair share rate. In addition, it allows flows passing through less restrictive contention domains to send at higher rates in order to achieve overall network efficiency.

It is possible to make this intuition more precise using a constructive definition, described below. Central to this constructive definition, and a major contribution of this paper, is the precise identification of the set of nodes whose flows share the channel capacity at a given node. We call all such nodes potential interferers of the given node. Before defining potential interferers, we introduce the following definition:

Interfering links: A link l_1 interferes with a link l_2 if a transmission along l_1 prevents the initiation or the successful reception of a transmission along l_2.

Then, we define a potential interferer as follows:

Potential interferer: A node n_1 is a potential interferer of node n_2 if a *flow* originating from node n_1 uses a link that interferes with the link between n_2 and its parent.

In tree-based wireless communication, the potential interferers are not merely the set of neighbors of a node, as we now show. Figure 1 shows a network of wireless nodes; solid arrows describe the routing tree, and dashed lines indicate neighboring nodes (those that can overhear each others' transmissions). Consider node 16 in Figure 1, which transmits data to its parent 14. Clearly, links $20 \rightarrow 16$, $21 \rightarrow 16$ and $14 \rightarrow 12$ interfere with the link $16 \rightarrow 14$. Moreover, links $13 \rightarrow 11$, $17 \rightarrow 14$ and $12 \rightarrow 10$ interfere with $16 \rightarrow 14$ as well: when a packet is being transmitted on any of these links, 14 *cannot safely listen to transmissions from* 16. Thus, any flow traversing these links shares the channel capacity at 16. For Figure 1 this includes flows originating from 16, 20, 21, 14, 13, 17, 12 as well as flows originating from 15, 18 and 19 as these flows traverse $12 \rightarrow 10$. These nodes form the set of potential interferers of node 16. Interestingly, node 11 is *not* a potential interferer of node 16, since the flow originating at node 11 does not traverse a link that

interferes with transmissions along $16 \rightarrow 14$ and therefore does not share the channel capacity at node 16. This example illustrates an important point: in tree-based communication, *the potential interferers of a node include nodes not just in the node's subtree or its neighbor's (parent included) subtrees, but also includes nodes in its parent's neighbor's subtrees*.

We now constructively define the fair and efficient rate allocation that IFRC strives to achieve. (In effect, IFRC achieves a max-min fair rate allocation, but we resort to this constructive definition since it cleanly motivates IFRC's novel congestion-sharing mechanism, which we describe in Section 4.1.2.) To do this, consider the sink tree built by the routing protocol (as in Figure 1), and define F_i to be the set of flows routed through node i. This set includes f_i (the flow from source i), and all the flows originating at the descendants in the subtree rooted at i. Also define a neighbor of node i to be any node which is within a single hop of i; this includes i's parent, its children, and other tree nodes that can hear i's transmissions (represented by dashed lines in Figure 1). Finally, for the purpose of this definition, assume that radios have a well-defined nominal data rate B. Then, IFRC attempts to achieve the following rate allocation:

1. At each node i, define $\mathscr{F}_i$ to be the *union* of F_i and all sets F_j, where j is either a neighbor of i, or a neighbor of i's parent. Following our discussion above, $\mathscr{F}_i$ includes flows from all of i's potential interferers. Allocate to each flow in $\mathscr{F}_i$ a fair and efficient share of the nominal bandwidth at i, B. (We do not define how to compute this share, since we are merely interested in defining the network-wide fairness and efficiency. Note that more efficient allocations may be possible than those that merely evenly divide B across all the flows by, for example, carefully scheduling non-interfering flows.) Denote by $f_{l,i}$ the rate allocated at node i to flow l. Repeat this calculation for each node.

2. Assign to f_l the minimum of $f_{l,i}$ over all nodes i.

We now apply this definition to node 16. Clearly, f_{16} belongs to $\mathscr{F}_{16}$. f_{20}, f_{21}, f_{14}, being flows from neighbors of node 16, belong to $\mathscr{F}_{16}$. f_{13}, f_{17}, f_{12}, being flows from neighbors of 16's parent, also belong to this set. Finally f_{15}, f_{18}, f_{19} also belong to $\mathscr{F}_{16}$ as they are routed through a neighbor of 16's parent 14. If, in this network, 16 is the most congested node, then all nodes with flows in $\mathscr{F}_{16}$ should be constrained to send at a rate no greater than that assigned to f_{16}. f_{11} can send at a higher rate since it is not in $\mathscr{F}_{16}$.

Intuitively, if node 16 is congested, the network should perform two actions: (i) reduce the arrival rate to this node, that is, reduce the rate of flows from nodes 16, 20 and 21, and (ii) increase the service rate of this node, that is, reduce the contention for the wireless channel between node 16 and 14. For the latter to happen, the capacity of 16 to transmit should increase (its neighbors 20, 21 and 14, and their descendants 17 should reduce the rate of their flows) and the capacity at 14 to receive packets should increase (its neighbors 17, 13 and 12 and their descendants 15, 18 and 19 should decrease the rate of their flows).

Before describing IFRC's rate control mechanisms, we consider two questions. Is fairness the appropriate design goal for sensor network data transport? Unless we get more deployment experience, this question is difficult to answer. Fairness is a reasonable initial design goal, and design insights from IFRC will be useful in designing other rate adaptation mechanisms for sensor networks. Furthermore, simple extensions to fairness, such as weighted fairness, can be easily realized with IFRC (Section 4.2). Secondly, should sensor network congestion control be rate-based or window-based? The former seems more natural for sensor networks, given sensors often generate periodic traffic. That said, many of IFRC's mechanisms can be adapted to window-based congestion control.

4. IFRC DESIGN

In this section, we discuss the design of IFRC, including the detailed node-level algorithms for congestion detection, signaling, and rate adaptation in IFRC. Extensions to, and limitations of, IFRC appear at the end of the section.

4.1 IFRC Mechanisms

In IFRC, each node i adaptively converges to a fair and efficient rate r_i for flow f_i using a distributed rate adaptation technique. It achieves this by accurately sharing congestion information with potential interferers. IFRC consists of three inter-related components: one that measures the level of congestion at a node, another that shares congestion information with potential interferers, and a third that adapts the rate using an AIMD control law.

We describe these mechanisms in this subsection.

4.1.1 Measuring Congestion Levels

Various techniques have been proposed in the wireless and sensor network literature to measure the level of congestion experienced by a node. Broadly speaking, these techniques either directly measure the channel utilization around a node [30], or measure the queue occupancy at the node [11], or a combination thereof. However, recent work [11] reports that, for a traffic pattern in which multiple sources send to a single sink, queue occupancy is a sufficiently accurate indicator of congestion. Intuitively, with a MAC layer that uses carrier-sense, MAC backoffs and retransmissions effectively cause queues to build up at a node, so queue lengths are a reasonable indicator of congestion.

At each node, IFRC maintains a single queue of packets from all flows passing through the node. This includes flows from all descendants of the node in the routing tree and that sourced by the node itself. Following much of the prior work on congestion control, IFRC uses an exponentially weighted moving average of the instantaneous queue length as a measure of congestion: $avg_q = (1-w_q) \times avg_q + w_q \times inst_q$. The average queue length is updated whenever a packet is inserted into the queue.

Conceptually, IFRC detects incipient congestion by using a simple thresholding technique with hysteresis. Thus, if avg_q exceeds a certain upper threshold U, the node is said to be congested, and the node halves its current r_i (the precise rate adaptation behavior is described in detail in Section 4.1.3), and then starts additively increasing its r_i. The node, however, remains in a congested state until avg_q falls below a lower threshold L.

In practice, a single upper threshold is too coarse-grained to effectively react to congestion. When avg_q crosses U, halving r_i and signaling others to accordingly adjust their rates may still leave node i in a congested state (for example, when the network size doubles instantaneously). Thus, IFRC employs multiple thresholds $U(k)$, defined by $U(k) = U(k-1) + I/2^{k-1}$, where k is a small integer and I is a constant increment of the queue length. When avg_q is increasing the node halves its r_i whenever avg_q crosses $U(k)$ for any k. Since the difference between $U(k)$ and $U(k-1)$ decreases geometrically with increasing k, the rate halving becomes more frequent as the queue size increases. In this manner, a node continues to aggressively cut its r_i until its queue starts to drain. (In TCP, the same effect is achieved when the sender treats each dropped packet as a congestion signal, and halves its window in response). Section 6 shows how this scheme effectively signals congestion while eliminating any packet drop due to queue overflow in our experiments.

4.1.2 Congestion Sharing

In Section 3, we introduced a constructive definition for IFRC's rate adaptation goal. To achieve that goal, it is necessary to share i's congestion state (its current average queue length) with other potential interferers. Each node can do this by explicitly transmitting its queue length to its potential interferers. Since some of its potential interferers can be many hops away from node i, the design of such a mechanism is a little tricky. IFRC uses a simpler congestion sharing mechanism, described below, that achieves the same goal.

In IFRC, node i includes the following information in the header of each outgoing transmission packet: its current r_i; its current average queue length, using which other nodes can infer i's congestion state; a bit indicating whether any child of i is congested; the smallest rate r_l among all its congested children (the reason for this information will become clear in a moment); and l's average queue length. Including this information in *every* packet enables robust signaling in the face of packet loss.

Using this information, all recipients of each packet (we assume that nodes snoop transmissions, a common assumption in many wireless protocols) can determine whether i (or any of its children) is congested or not. All neighbors of i (its children, its parent, and other nodes that can hear i's transmissions) receive this packet. However, some potential interferers of i (the neighbors of i's parent) may *not* receive this packet. How, then, does IFRC converge to the fair rate?

IFRC introduces two simple constraints on the value of r_i at node i:

Rule 1: r_i can never exceed r_j, the rate of i's parent j.

Rule 2: Whenever a congested neighbor j of i crosses a congestion threshold $U(k)$ (for any k), i sets its rate to the lower of r_i and r_j. The same rule is applied for the most congested child l of any neighbor of i, *i.e.,* i sets its rate to the lower of r_i and r_l where l is the most congested child of i's neighbor.

Why do these two constraints achieve the goal described in Section 3? Consider a congested node i. By rule 1 above, all descendants of i will eventually be notified of i's congestion and reduce their rates to that of r_i. By rule 2, any other neighbor of i, including its parent, will set its own rate to r_i. By the same rule, all neighbors of i's *parent* will set their rates to r_i since when i's parent sends any transmission it indicates that one of its children (namely i) is congested. Finally, recursively, descendants of a non-parent neighbor, and descendants of the parent's neighbors will also reduce their rate to r_i, again by rule 1.

More specifically, let us assume node 16 in Figure 1 is congested. We require all nodes with flows in $\mathscr{F}_{16}$ to have the same rate as r_{16}. By rule 1, 16's children 20 and 21 will set their rates to r_{16}. By rule 2, 16's neighbor 14 will set its rate equal to r_{16}. 14 will include information about a congested child in its outgoing packets. Based on this information and using rule 2, nodes 13, 17 and 12 will set their rate to equal to r_{16}. All the remaining nodes with flows in $\mathscr{F}_{16}$, namely node 15, 18 and 19 will eventually have their rates set to r_{16} by rule 1 since one of their ancestors has set its rate to r_{16}.

Finally, IFRC's state scales well, since, with this congestion sharing mechanism, a node need only maintain state proportional to the number of neighbors.

4.1.3 Rate Adaptation

Finally, in this section, we describe some of the details of IFRC's rate adaptation mechanism. Before we do this, it is worth emphasizing that the averaged value of r_i (which oscillates within a narrow range during steady state operation) represents the long-term fair and efficient rate at which node i can *originate* traffic. r_i itself is *not* the maximum instantaneous rate at which a node can originate traffic; the instantaneous rate may be affected by MAC-layer transmission scheduling mechanisms.

In IFRC, nodes increase their r_i's additively. Specifically, every $1/r_i$ seconds, a node i increases its rate by δ/r_i. We discuss the choice of the parameter δ and the choice of the $1/r_i$ control interval in Section 5. If i is congested, then when threshold $U(k)$ is crossed, the node halves its current r_i. It does this at most once for each $U(k)$ during one congestion epoch; an epoch begins when the node transitions to a congested state, and ends when it transitions to a non-congested state. The latter happens when, in a congested state, the average queue length goes below the lower threshold L.

When i discovers its r_i to be higher than that of its parent j, it sets its rate to r_j. When i overhears a neighbor l's transmission that indicates that l, or one of l's children p has crossed a threshold $U(k)$, its sets r_i to either r_l or r_p as necessary. Both these sets of rules follow from the discussions in the previous two sections. In either case, i does *not* change its own congestion state.

All nodes start from a fixed initial rate r_{init}. For faster convergence, IFRC implements a multiplicative rate increase initially. This technique is similar to TCP's slow start. While node i is in slow start, it adds ϕ to its rate every $1/r_i$ seconds, thus multiplicatively increasing its r_i. It exits this mode when one of three conditions is true. If, in slow start, i itself becomes congested, it halves r_i. If its r_i equals or exceeds that of its parent, it sets r_i to its parent's rate and transitions to additive increase. Finally, if its rate is constrained by congestion sharing, i sets its r_i to that of the constraining node and transitions to additive increase. These three conditions are consistent with the congestion sharing rate adaptation mechanisms discussed above.

Slow start behavior is also invoked, for rapid recovery, when r_i equals or goes below r_{init}. This can happen in one of two ways. If i's average queue length increases continuously, it will repeatedly halve its rate (Section 4.1.1). Alternately, i's rate can be constrained by a neighbor to r_{init}.

4.1.4 Base Station Behavior

The base station is a distinguished node in IFRC, since it never sources any traffic, and therefore does not need a separate allocated rate. It does, however, need to facilitate congestion sharing between its children. For this, it maintains a "rate" r_b and enforces congestion sharing by adapting r_b to congestion indicators from its children. Its children are constrained by rule 1 in Section 4.1.2 to a rate no greater than r_b.

The base station implements a slightly different algorithm to adapt its rate r_b. It decreases its rate only when any of its children j crosses $U(k)$ for any k. Unlike other nodes, it does *not* decrease its rate when any of its non-child neighbors or any child of a neighbor is congested. Since the base station does not source or forward any traffic, the rate of its children (determined by r_b) does not affect the congestion status of the congested neighbor (or the congested child of a neighbor) of the base station.

Initially, IFRC sets r_b equal to the nominal data rate of the wireless channel. IFRC is not sensitive to the choice of value for this rate, since each node independently measures congestion by monitoring their own queue lengths. The initial value of r_b merely needs to be high enough that the base station's children can determine their fair rates without being constrained by the configured value. The base station *always* additively increases r_b: every $1/r_b$ seconds, it increments r_b by δ/r_b. Since the base station does not source any traffic, it broadcasts a control message containing this rate as a way of sharing congestion information. To limit control

message overhead, the base station broadcasts this message after at least m packets have been received from the child with the highest rate. Intuitively, this lets the fastest child perform m additive increases before hearing from the base station. In our implementation, $m = 5$. However, when r_b is decreased in response to congestion at a child, the base station immediately broadcasts the control message. This message is broadcast twice, to increase the likelihood that the children receive the "bad news" quickly (in the event that a child doesn't receive both these transmissions, it will eventually get one of the periodically transmitted control messages).

4.2 Extensions to IFRC

So far, we have assumed that every node in the network has data to send to a single base-station. Our IFRC design is flexible enough to accommodate multiple base stations, weighted fairness, or only a subset of nodes transmitting data.

Multiple Base Stations: Consider a network with multiple base stations. Each sensor node may be able to send to more than one base station, but picks the one with the best path and the system converges to one routing tree for each base station. In this scenario, a potential interferer for node i may be on a different routing tree than the one i is on. IFRC must account for this in adapting to a fair and efficient rate.

To accommodate multiple base stations, IFRC requires a simple modification to the base station behavior (Section 4.1.4). Figure 2 motivates this modification and shows two routing trees rooted at node 0 and node 10. When node 11 is congested, node 10, following the base station behavior described above, sets r_{10} to r_{11}. r_1 should also to be set to r_{10} as node 1 is a potential interferer of node 11. But since node 10 is not congested (it is the base station), Rule 2 in Section 4.1.2 is never triggered at node 1.

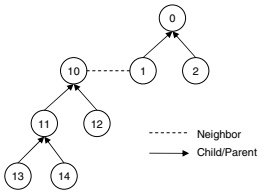

Figure 2: Scenario with multiple base stations (nodes 0 and 10)

Our fix to this modifies the control packet transmitted by the base station to include, in place of the current rate of the most congested child, the base station's own rate. With this change, when node 11 is congested, r_{10} resets its rate to r_{11} and sends a control packet indicating that r_{11} is congested. This triggers Rule 2 at node 1 if r_1 is greater than r_{11}. This modification is backward compatible to the single base station case. All our experiments have incorporated this modification. In discussing other extensions to IFRC below, we revert to a single base-station scenario, and believe that these other extensions will extend without modification to multiple base stations.

Weighted Fairness: Consider a network where node i is assigned a weight w_i. We say a rate allocation scheme is fair if the normalized rate for each flow, $\frac{f_i}{w_i}$, is fair according to our definition in Section 3. We have implemented weighted fairness as follows, without needing to change any aspect of IFRC's rate adaptation, congestion sharing, or queue management behavior: each node sends traffic at a long-term rate $w_i \times r_i$, instead of just r_i. This is equivalent to adding $(w_i - 1)$ leaf nodes to the parent of i each sending at rate r_i; this intuition explains why no modifications are needed to IFRC. We experimentally demonstrate this strategy in Section 6.2.

When only a subset of nodes transmit: Finally, IFRC works (without modification) when only a subset of nodes are sending data in the network. Each node i maintains r_i exactly as it would if it had data to send. Intuitively, a subset of nodes sending data can be considered as a special case of weighted fairness where nodes that have data to send have weight 1, and nodes that don't have weight 0. We experimentally demonstrate this extension in Section 6.2.

4.3 Discussion

As we discussed briefly in Section 3, IFRC currently works well in regimes where hop-by-hop loss recovery (using a limited number of retransmissions) suffices to recover from most end-to-end losses. Hop-by-hop loss recovery is used in many real-world scenarios today to improve delivery reliability [22, 28]. Furthermore, many, but not all, of our experiments in real radio environments fall in this regime. (Our experiments were conducted in relatively harsh wireless environments with loss rates in the 15-40% range, with paths of more than five hops, and where five retransmissions or less were sufficient to recover from losses in, relatively speaking, most—92% or more—cases.) In this regime, the average r_i is effectively the *goodput* that the node receives. Outside this regime, however, since IFRC is essentially an open-loop system and does not get feedback about the actual goodput each node receives, there is no way to ensure globally fair allocation of goodput. For this purpose, as well as for reliability reasons, IFRC needs to be complemented with an end-to-end reliability mechanism.

Designing a scalable end-to-end reliability mechanism is a challenge, and we have left that to future work. When integrated with such a mechanism, we expect the rate adaptation parameters and some of the details of those algorithms to change, but the congestion sharing algorithms to stay the same.

IFRC cannot detect the reduction of channel capacity caused by interfering transmissions from non-neighboring nodes. This kind of interference is harder to detect at the higher layers of software, and manifests itself as packet loss. The interference rejection capabilities of spread spectrum radios mitigates this problem somewhat, and hop-by-hop recovery deals with its effects.

Node battery conservation is an important issue in the design of sensor network systems. Many existing sensor network systems duty-cycle network nodes [28] to conserve energy. As long as the underlying routing protocol maintains or re-constructs routing paths when nodes resume from sleep, IFRC will work unmodified. Although duty-cycling achieves significant energy conservation, other research has proposed finer-grain energy conservation methods [34], for example, by turning off radios to avoid overhearing transmissions. IFRC assumes promiscuous listening for congestion sharing, and so will not work with such methods.

IFRC will work without modification when intermediate nodes perform in-network aggregation [12]. In general, aggregation reduces the volume of data transmitted by nodes; in IFRC, nodes will adaptively detect the availability of, and use, the available capacity.

5. PARAMETER SELECTION IN IFRC

IFRC converges to a fair rate using AIMD, but its long-term stability, efficiency, and convergence properties crucially depend upon the choice of its parameters. The choice of many IFRC parameters is influenced by IFRC's open-loop design. In reliable transport protocols like TCP, acknowledgments used for error recovery are received after one round trip time. This RTT is a natural timescale at which rate can be adapted. Since IFRC is not coupled with an end-to-end feedback mechanism (Section 3), it departs from traditional congestion control protocols in its parameter choices.

Intensity of AIMD: In IFRC, node i increases its rate r_i every $1/r_i$ seconds by δ/r_i. Other additive functions are possible. We choose this because, intuitively, $1/r_i$ is the inter-packet transmission time, and thus it provides a natural timescale for making a control decision. Hence, we have:

$$r_i\left(t+\frac{1}{r_i(t)}\right) = r_i(t)+\frac{\delta}{r_i(t)}.$$

It is easy to see that the function $r_i(t)$ is a linear function with slope δ, that is, $r_i(t) = \delta t$.

The choice of δ dictates the intensity of the additive increase, and is thus crucial for both stability and speed of convergence. We now present a simple analysis of the steady state behavior of the system, which we use to deduce the appropriate value of δ.

Recall that $r_i(t)$ is the rate at which node i generates traffic at time t. Let $r_{st,i}$ be the maximum sustainable rate of node i, $r_{min,i}$ be the minimum rate of node i, and $r_{max,i}$ be its maximum rate. Note that all these rates do not include the rate by which node i forwards traffic generated by other nodes. Figure 3 shows $r_i(t)$ during the additive increase phase. The rate increases from $r_{min,i}$, crosses the maximum sustainable rate $r_{st,i}$, and, once a congestion signal is received, it decreases from $r_{max,i}$ to $r_{min,i}$ by half.

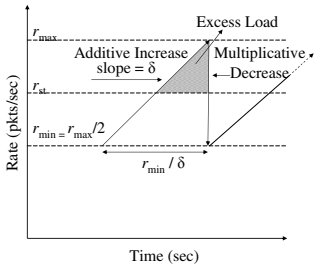

Figure 3: AIMD behavior of $r_i(t)$

For stability, we require that the amount of data transmitted when r_i is above r_{st} (the "excess load" in Figure 3) be no more than the unexploited transmission opportunity when r_i is below r_{st} (the "underutilized capacity" in Figure 3), *i.e.*, $r_{st,i} > \frac{r_{min,i}+r_{max,i}}{2}$. Also, since, our multiplicative decrease factor equals $1/2$, $r_{max} = 2\times r_{min}$ and we have:

$$r_{st,i} > \frac{3r_{max,i}}{4}. \tag{1}$$

To avoid r_i jumping from $r_{min,i}$ to $r_{max,i}$ in a single step, we require $\delta/r_{min,i} \ll r_{min,i}$, or,

$$\delta = \varepsilon r_{min,i}^2,$$

where $0 < \varepsilon < 1$ is a small positive number whose value we derive below.

When the average rate of nodes in the network is below their sustainable rate, $r_{st,i}$, the load on the network is less than the capacity and thus the network is able to support the load without any queue build-up at the nodes. When the rate of nodes increases beyond their sustainable rate, the excess load will cause queues to build up at the congested nodes. The excess number of packets that a node sends during this time is equal to the area of the shaded region in Figure 3, which equals $(r_{max,i}-r_{st,i})^2/(2\delta)$ for each node i.

Let's focus on one congested node, call it node j. Let I_{ij} be an indicator function that equals one if the packets from node i traverse node j, and equals zero otherwise. Then, the total number of excess packets that are accumulated at node j equals

$$\sum_i \frac{(r_{max,i}-r_{st,i})^2 I_{ij}}{2\delta}.$$

This assumes that the r_i values increase and decrease in synchrony at all network nodes. This is an idealized assumption that would only be true if feedback were instantaneous. As Figure 8 shows, this assumption holds approximately in our experiments, because per-hop latencies are small relative to the timescale of additive increase.

Intuitively, the formula above measures the excess work on node j's queue due to the increase in the packet arrival rate at this queue. Notice that, in the above calculation, there is an underlying assumption that the service rate of this queue remains the same independently of the network congestion. But this assumption does not hold for wireless networks because of contention. In particular, contention reduces the service rate of the queue and creates further queue build up. To see this consider a flow from a potential interferer of j. This flow may use various links that interfere with the link between j and its parent. A transmission of an excess packet of this flow along one of these interfering links will prevent j from transmitting a packet it would otherwise transmit, which is *as if* one more packet is added at node j's queue. Precisely analyzing such contention is a hard problem in general. Here, we model its effect by replacing the indicator function I_{ij} with a function f_{ij} whose value reflects not only the arrival of an excess packet, but also the number of time slots that node j cannot transmit due to the transmission of an excess packet by one of j's potential interferers.

The congested node will signal congestion only when its average queue increases beyond the threshold $U(0)$. Let U_0 denote the instantaneous queue length required for the EWMA-averaged queue length to exceed $U(0)$. Then, for congestion signaling, we require

$$\sum_i \frac{(r_{max,i}-r_{st,i})^2 f_{ij}}{2\delta} > U_0. \tag{2}$$

Further, IFRC decreases its rate every time a threshold is crossed. Therefore, to avoid raising multiple congestion signals, that is, to avoid multiple multiplicative decreases, we also require that the excess load is less than the second threshold. Thus,

$$\sum_i \frac{(r_{max,i}-r_{st,i})^2 f_{ij}}{2\delta} < U_1, \tag{3}$$

where U_1 is the instantaneous queue length required for the EWMA-averaged queue length to exceed $U(1)$, starting from an empty queue.

In the analysis so far we have ignored the latency associated with the time that it takes for the congestion signal to travel from node j to all its potential interferes. Let i be a potential interfere of j and s_i be the number of rate updates performed at node i before it receives the congestion information from node j. Assume that the rate at node i when congestion occurred at node j was at least $r_{st,i}$ and approximate the increase in r_i due to the s_i updates by $s_i\delta/r_i$. Then, we have

$$r_{max,i} > r_{st,i} + \frac{s_i\delta}{r_{st,i}}. \tag{4}$$

Equations (1), (2), and (3) guarantee system stability, and, whenever there is congestion, that one and only one signal is going to be propagated to all relevant nodes. Together with Equation (4) that takes the propagation delay of the congestion signals into account, they dictate some operational constraints that must be satisfied.

Without loss of generality, assume all node weights are equal, that is, in steady state $r_{st,i}$, $r_{max,i}$, and $r_{min,i}$ are the same for all

i. By substituting r_{st} in Equation (2) using its lower bound from Equation (1), and letting $F_j = \sum_i f_{ij}$ we get $\frac{r_{st}^2 F_j}{18\delta} > U_0$. Similarly, by substituting r_{max} in Equation (3) using its lower bound from Equation (4) and letting $\bar{s}$ be the average value of s_i we get $\frac{\bar{s}^2 \delta F_j}{2r_{st}^2} < U_1$. Now, recall that $\delta = \varepsilon r_{min}^2$ to get

$$\varepsilon < \frac{F_j (r_{st}/r_{min})^2}{18U_0}, \text{ and } \varepsilon < \frac{2U_1 (r_{st}/r_{min})^2}{\bar{s}_j^2 F_j}.$$

The value of r_{st}/r_{min} ranges between 1.5 and 2. (The lower bound comes from Equation (1) and the fact that $r_{max} = 2 \times r_{min}$, and the upper bound from the fact that $r_{max} = 2 \times r_{min} > r_{st}$.) Any value in this range can be legitimately substituted into the above inequalities. However, a particular choice of value determines the efficiency of the system. When the value is 1.5, the average rate assigned to each node will be r_{st}, but when the value is 2, the system achieves only 75% efficiency. We choose the former, sacrificing convergence time for efficiency (notice that choosing 1.5 yields a lower value of ε, resulting in longer convergence times) which yields:

$$\varepsilon < \frac{F_j}{8U_0}, \tag{5}$$

and

$$\varepsilon < \frac{9U_1}{2\bar{s}^2 F_j}. \tag{6}$$

Intuitively, when F_j is small, the first inequality (Equation (5)) determines ε. This happens in small networks or in sparse networks with low contention. In larger networks, the second inequality (Equation (6)) determines ε.

In Section 6, we conduct experiments that validate our analysis. In particular, we show that choosing ε using the above inequalities results in stable operation, while choosing ε that minimally violates them results in instability.

In general, however, choosing an ε for an arbitrary network is tricky, since F_j is a function of the topology and the tree. In our experiments, we have been using a rule-of-thumb value of $n \log n$ for F_j, where n is the number of nodes in the network. $n \log n$ is a good approximation of F_j for a reasonably balanced tree in a network where every node can hear every other node, and where the congested node is close to the root. In the worst case—in a chain topology in which all links interfere with each other—F_j can be n^2.

Choice of Other Parameters: We now turn to a brief discussion of how to set other parameters. Our choices for these parameters are based on intuition, rather than rigorous analysis, but which we have validated through experimentation. We have left the analysis to future work.

Clearly, r_{init} should have a value that is smaller than the steady state rate achievable for a given network. We conservatively set r_{init} to be one order of magnitude smaller than the steady state rate. Now, in a wireless network of n nodes, depending upon the network topology and the routing tree, the maximum achievable rate, r_{st}, can vary greatly. In a sparse network with a balanced tree, nodes can hear only their parent and children. In this case r_{st} is $O(1/n)$. When each node can hear all other nodes, and the tree is balanced, r_{st} is of the order of $O(1/(n \log n))$. If instead, the tree degenerates to a chain topology, r_{st} is $O(1/n^2)$. This is the worst case. We expect most networks of interest to be closer to the second category, and thus conservatively set r_{init} to equal $B/(10n \log n)$, where B is the nominal data rate of the radio.

The parameter ϕ determines the rate of multiplicative increase during slow-start. We conservatively set it to $r_{init}/8$ to ensure a small slow-start overshoot over the sustainable rate, without compromising the speed at which slow-start hunts for that rate.

Finally, for a given choice of $U(0)$, the choice of w_q dictates the burst lengths U_0 and U_1 that each node is able to accommodate. The relationship between these two parameters and the burst length has been explored in [6]. In our case, a reasonable rule of thumb for U_0 is $N/2$, and U_1 is N; intuitively, if the congested node is close to the root of the tree, we would like to be able to accommodate one additive increase step (which, in IFRC, is one packet) at either that node or its child (when a node is congested, queues start to build up at its child too), from each node in the network.

Our parameter choices in Section 6 have been derived based on the discussions in this section.

6. EVALUATION

In this section, we present the main results from an extensive performance evaluation of our implementation of IFRC on a 40-node wireless sensor testbed.

6.1 Implementation and Methodology

We have implemented, in TinyOS 1.1, all the features of IFRC described in Section 4. Our implementation is about 1200 lines of code, and consists of two modules, QueueManagement and RateControl. The former module maintains the packet queue, and implements packet forwarding functionality. It also measures the node's congestion level as described in Section 4.1.1, and signals the RateControl module when the node's congestion state changes. The RateControl module implements congestion sharing (Section 4.1.2) and rate adaptation (Section 4.1.3). In addition, the module maintains a neighbor table to track neighbor congestion states and rates. The RateControl module also promiscuously snoops network packets for congestion sharing. In our hardware platform, putting the radio in promiscuous mode disables the chip-level acknowledgments usually provided by the radio hardware. Since hop-by-hop recovery is essential for IFRC, we added MAC-layer acknowledgments, but expect IFRC to work well with chip-level acknowledgments as well.

We evaluated our IFRC implementation on a 40-node indoor wireless sensor network testbed. Each sensor node in our testbed is a Moteiv Tmote with an 8MHz Texas Instruments MSP430 microcontroller, 10KB RAM and a 2.4GHz IEEE 802.15.4 Chipcon Wireless Transceiver with a nominal bit rate of 250Kbps. These motes are deployed over 1125 square meters of a large office floor They have a USB back-channel that we use for logging experimental data. The motes collectively form a connected topology (Figure 4) with a diameter of eight hops. We only depict links that have a loss rate of 40% or lower.

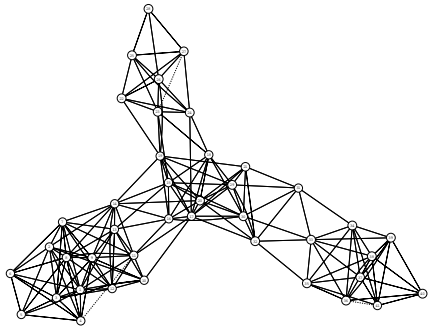

Figure 4: Testbed connectivity graph

We have tested IFRC on several routing trees constructed on

various topologies with up to 40 nodes, and with different base station locations. Here, we present a subset of these results. Table 1 shows the parameters used in our experiments (unless otherwise specified); these parameter choices were driven by the discussion in Section 5.

Parameter	Value
Lower Threshold (L)	4 pkts
Upper Threshold ($U(0)$)	8 pkts
Queue Increment (I)	8 pkts
Additive Increase Parameter (ε)	0.025 per pkt
Slow-start initial rate (r_{init})	0.02 pkt/sec
Slow-start mult. incr. (ϕ)	0.0025 pkt/sec
MAX_RETRANS	5
Queue Size	64 pkts
EWMA Weight (w_q)	0.02
Packet Size	32 bytes

Table 1: IFRC parameters used in the experiments

A routing protocol which computes routes using link reliability estimators can incur route changes under heavy traffic. During preliminary experiments, we have found IFRC to work well in dynamic topologies. However, in order to study the steady-state behavior of IFRC, we modified the existing TinyOS routing protocol (MultiHopLQI) to terminate the route computation after an initial, reasonable tree is found. For each experiment, then, we ran this modified routing protocol to fix an initial tree, after which we ran IFRC on it. This procedure results in the same routing tree with substantially similar link qualities across different IFRC experiments.

We ran each experiment for *at least* an hour. For each experiment, we logged every packet transmission and reception, and every change in rate, at each network node. In addition, we logged every packet received at the base station. This fairly detailed logging helps us visualize IFRC behavior in several ways. However, during and across runs, the quality of wireless links can and does change. This is beyond our control, of course, but we compensate by running long experiments at consistent times of day (usually late at night or in the early morning hours).

We plot the per-flow *goodput* as the total number of packets received from a node at the base station divided by the duration of the experiment. We plot the *rate adaptation* at each node, by plotting r_i as a function of time. For some experiments, we visualize *instantaneous queue size* at nodes as a function of time. For some experiments, we also present a per-flow *packet reception* plot, which shows the number of packets received at the sink as a function of time.

6.2 Results

In this section, we present experimental results that validate our IFRC design, demonstrate that IFRC can be extended in the ways we have discussed in Section 4.2, and analyze the consequences of poor parameter choices for IFRC.

IFRC on a 40-node network: We now discuss the performance of IFRC on a 40-node network. All 40 nodes are transmitting data, and there is one base station. All flows are treated equally.

Figure 5 shows the routing tree generated during one experimental run. Packet reception ratios are depicted on each link and vary between 66% and 96%. This tree is *9 hops* deep, which is slightly greater than the diameter of the underlying topology. This surprisingly large depth is because of the way the routing protocol can prefer shorter links with higher qualities over poorer but longer links. The depth of the tree, the variable fanout at various nodes, the high variability in packet loss rates, and the unbalanced layout make this a good routing topology to study IFRC performance.

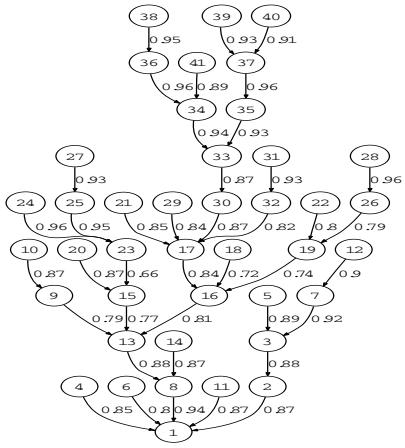

Figure 5: Routing tree and link qualities used in the experiments

Figure 6 shows the per-flow goodput received at the sink. Each bar represents the average rate of transmitted packets from the corresponding node. The lighter 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, darkly shaded, section of the bar). We make several observations from this graph. First, it validates the basic design of IFRC, and shows that nodes receive approximately fair rates and fair goodput. In this topology, it turns out that nodes 13 and 8 are congested, and every other node is a potential interferer of one of these nodes. For this topology, hop-by-hop recovery resulted in fewer than 8% of the packets being lost end-to-end. Second, node 1 is the base station, and the bar corresponding to node 1 indicates the base station's signaling control traffic rate. This represents pure overhead, and is less than 1 packet in 10 seconds. Figure 7 shows the packet reception

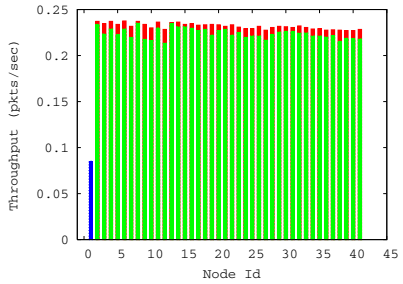

Figure 6: Per flow goodput in the 40-node experiment

plot for all the nodes in the network. The slope of this curve around a particular point gives us an estimate of the instantaneous goodput received by the node at that point. Notice that the reception plot is relatively smooth, with minor variations attributable to AIMD.

Figure 8 shows the rate adaptation at each node, including the base station. All nodes adapt their rates in near-synchrony, as the rate plots for various nodes overlap with each other (cf. the discussion in Section 5). The slow start phase and the classic AIMD sawtooth behavior is clearly visible in the graph. A closer look (not shown) reveals that the rate adaptation at different nodes is slightly de-synchronized by network latencies, but this effect does not show up at the time scale of the graph. Finally, there are several small measurement artifacts in the graph, caused by loss of experimental

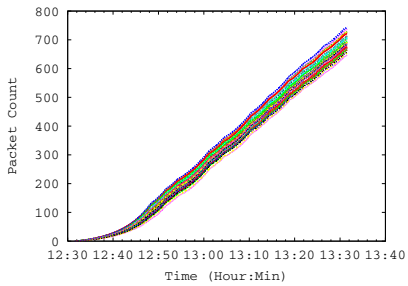

Figure 7: Packet reception in the 40-node experiment

data. For example, the horizontal line starting at about 12:45 shows that the rate plot for one of the nodes is slightly distorted. This results from an infrequent data logging queue overflow problem at the USB ports on one of our motes, due to software driver issues beyond our control.

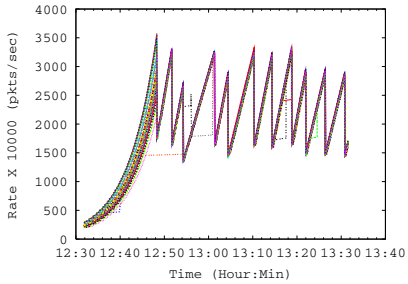

Figure 8: Rate adaptation in the 40-node experiment

In subsequent experiments, unless otherwise specified, we use the same routing tree (Figure 5) as the one used for Figure 8.

Figure 9 shows the instantaneous size of the queue at every node. While it is difficult to decipher the detailed behavior of each flow from the graph, we clearly see that the *instantaneous* size varies a fair bit, but never grows beyond 20. Each of our nodes has a buffer size of 64, and, so, in this experiment, no packets are lost due to queue overflow. In fact, *in all the experiments that we have conducted, we have never seen a single instance of buffer overflow*. This results from IFRC's aggressive rate cutting at each $U(k)$ (Section 4.1.1).

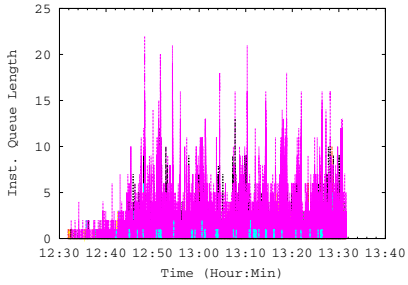

Figure 9: Instantaneous queue size in 40-node experiment

Optimality of IFRC: How close is IFRC's rate allocation to the optimal? There are several ways to answer that question, but the one we have chosen is to ask: What is the maximum fair rate sustainable on the routing tree of Figure 5? This can help us understand how much inefficiency IFRC introduces. To understand this, we programmed each node to send at a fixed rate R, without IFRC, but kept buffer sizes the same as in our above experiment, and used link-layer retransmissions to recover from losses. Starting with a value of R corresponding to the fair rate in Figure 6, we then increased R and measured the goodput received by each node. Our conjecture was that beyond some rate R', the network would result in unfair goodput, and R' would be the benchmark we could use to calibrate IFRC.

Figure 10 shows two distinct plots in the same graph. The x-axis depicts the successive rates used in the experiment. The solid line is the ratio of the maximum goodput received by any node to the minimum goodput. The dotted line measures the largest queue seen at a node during an experiment.

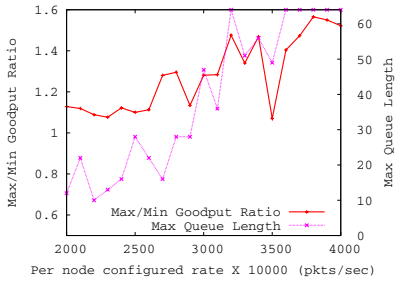

Figure 10: Optimality test for the 40-node experiment

Consider the goodput ratio curve. It remains flat initially, and then starts increasing. If we define the point at which it starts increasing to be R' (about 0.27 packets/sec), then IFRC achieves about 80% of the maximum sustainable fair rate. Roughly speaking, that value of R' also represents the knee of the maximum queue size curve. However, real-world experiments are often messy; in our graph, there are two higher rates that appear to be anomalously non-monotonic, but still unfair. If we conservatively assume that R' is closer to 0.36 packets/sec (beyond which the largest queue always overflows, as can be seen from the second, queue size plot), then IFRC still achieves about 60% of the maximum sustainable fair rate. Either way, this is extremely encouraging, and validates the basic motivation and premise of our design.

As an aside, the nominal bandwidth of the Tmote radio is about 80-90 packets/sec. However, IFRC's fair rate for our topology amounts to a total goodput of about 8.8 packets/sec. Why this apparent discrepancy? First, our goodput numbers do not account for loss in channel capacity due to lost transmissions and re-transmissions. Second, our link-layer acknowledgments reduce some of the network capacity. Finally, the TinyOS MAC layer implements a random, and not exponential backoff. The default random backoff parameter in TinyOS is not sufficient for the densities and traffic levels we use, so we increased it by a factor of three. These last two changes effectively reduce the nominal bandwidth of the radio to about 50 packets/sec. Now, in our experiments, we have measured the total traffic at each node (defined to be all packets received, overheard, and transmitted by that node, at the transport layer). We find that the busiest node has a total traffic of close to 32 packet/sec, close to the nominal bandwidth. Collision-related losses handled purely by the radio at the MAC layer partially contributed to the remaining difference.

Dynamics and Startup Behavior: IFRC dynamically adapts to node addition and removal. We would have liked to validate this behavior by turning nodes on and off. However, as we have discussed above, IFRC can interact with link quality based route selection, and turning notes on and off would trigger changes to the routing tree. So, we conducted two simple experiments to demonstrate how IFRC adapts to the addition or removal of nodes.

In the first experiment, all the nodes were turned on at the beginning of the experiment, and the routing tree was fixed, but only a fourth of the nodes started sending data initially. The others started sending data about 20 minutes into the experiment. Figure 11 shows the rate adaptation of the nodes in this experiment. The reduced overall fair share rate when nodes are added to the network is clearly visible in this plot. The horizontal line in the left half of the picture represents the r_i's of inactive nodes. The graph also shows another interesting feature: two successive multiplicative decreases occur at around 9:09. Such a behavior is sometimes caused by a parent and a child reaching a congested state within a short time of each other. When the parent gets congested, the child reduces its rate. Again, when the child's own congestion threshold is exceeded, the child halves its rate again.

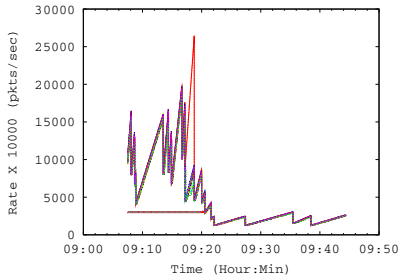

Figure 11: Node addition: Rate adaptation

The second experiment was similar, but all nodes started sending data at the beginning, and three-fourths of the nodes stopped about 20 minutes into the experiment. Nodes that continue transmitting data obtain higher fair share rates, as shown in Figure 12. This figure also has an interesting feature: a slight de-synchronization at 10:30 of the additive increase rates of some nodes. This is caused by packet loss; if a node loses a couple of transmissions from its parent, its rate can then lag behind that of its parent for some time.

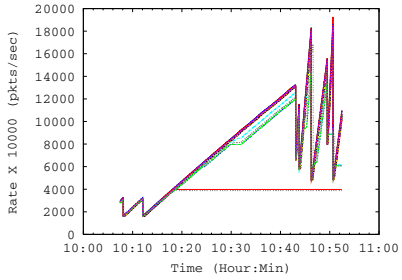

Figure 12: Node deletion: Rate adaptation

IFRC Extensions: In this section, we validate the performance of the IFRC extensions discussed in Section 4.2.

Figure 13 plots the per-flow goodput for an experiment in which some nodes were assigned different weights than others. Each node whose id is divisible by 4 was assigned a weight of 2 and all the other nodes were assigned a weight 1. The figure clearly demonstrates that IFRC is effective in allocating rates conforming to the configured weights. All other nodes whose identifiers are divisible by 4 get twice the fair share rate dictated by the most congested node. (As with other goodput graphs, the lighter section of the bar measures goodput received by each node, while the bar for node 1 denotes the base station's control traffic overhead.)

Figure 14 shows the per-flow goodput for an experiment in which only nodes whose identifiers were divisible by 3 were allowed to

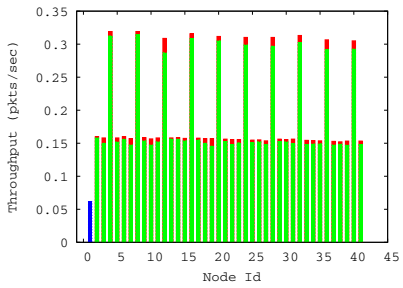

Figure 13: Per flow goodput with weighted fairness

transmit data. As expected, all transmitting nodes receive at least the fair rate. Furthermore, the fair share per-flow goodput is approximately a factor of four higher in this case than when all nodes transmit; IFRC adapts to the increased overall available capacity and allocates it fairly to the transmitting nodes.

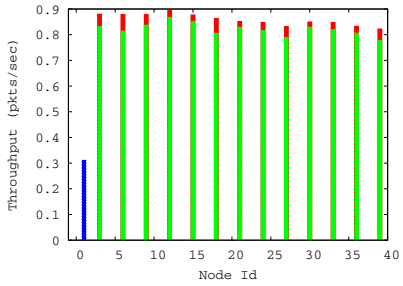

Figure 14: Per flow goodput with only a subset of senders

Finally, we ran an experiment with *two* base stations (Figure 15). Nodes numbered higher than 33 have node 41 as the sink, and the rest send to another sink (node 1). As Figure 15 shows, the two sets of nodes get different fair share rates. The most congested node in the larger cluster is 17; the smaller cluster is not constrained by this node. This experiment further illustrates IFRC's efficiency. Nodes 4, 5 and 6 receive a little less than *twice* the fair share rate because, in this topology, these nodes are not the potential interferers of node 17.

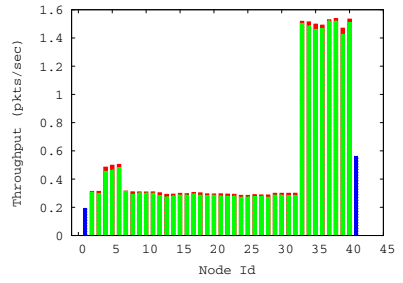

Figure 15: Per flow goodput with multiple sinks

Validating Design Choices and Parameter Settings: In Section 3, we discussed why link-layer retransmissions are essential for IFRC. In all of our experiments, we have found that the five retransmission limit we use is sufficient to ensure fairness. (We emphasize that our topology encounters fairly high loss rates, with some links experiencing upwards of 30% loss rates.) Figure 16 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 lighter section of each bar). This, as we have discussed in Section 4.3, is because nodes receive no feedback about the goodput they receive.

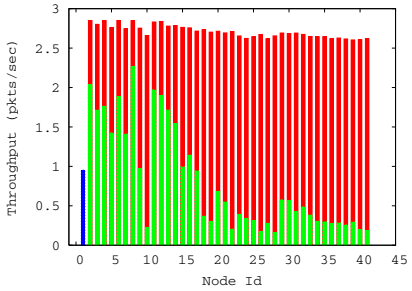

Figure 16: Per flow goodput with no link-layer retransmissions

Finally, we conducted two experiments to validate the analysis of Section 5. Table 2 summarizes our findings. Our first experiment involved running IFRC on two nodes. For this experiment, we chose a nominal value of $U_0 = 10$ and $U_1 = 20$, which gives a value of $w_q < 0.32$. We conservatively set w_q to be 0.3. With these values, the first inequality determines ε, giving an $\varepsilon < 0.0125$. We then increased ε slowly, until the system became unstable—the queue didn't drain completely after one multiplicative decrease and the subsequent additive increase, triggering another multiplicative decrease. This happened at $\varepsilon = 0.0167$. For this experiment, the ε value predicted by our analysis is conservative (in that the predicted value results in stability), but tight.

Experiment	Predicted ε	Smallest unstable ε
2 nodes	0.0125	0.0167
30 nodes	0.147	0.2

Table 2: Validating the analysis of Section 5

For a larger, 30-node network, we conducted a similar experiment, but with a small difference. In a large network, as we have discussed before, it is difficult to estimate F_j or $\overline{s_j}$, so a reasonable rule of thumb for the former is $n \log n$ and for the latter is a small number (we chose 3, which is the about the average network radius). Following Section 5, we set $U_0 = 15$ and $U_1 = 30$, giving an $\varepsilon < 0.147$. For this large network, we ran IFRC using ε values of 0.05, 0.1, and 0.2. Instability occurred at 0.2. Thus, for this larger network, choosing ε using our analysis and our rules of thumb resulted in a conservative, but stable choice.

While these experiments are not conclusive, they are highly encouraging in that they both validate the analysis, and our rules of thumb for picking IFRC parameters for a given network.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we have described the design and implementation of IFRC, the first practical interference-aware rate control mechanism for wireless sensor networks. IFRC incorporates a novel low-overhead congestion sharing mechanism that leverages the prevalent tree-based communication pattern in wireless sensor networks. IFRC is fair and efficient in realistic wireless environments with packet loss rate over 30%, and its queue management strategy completely prevents packet drops due to queue overflow (at least in the experiments we have conducted).

Our design of IFRC points to some directions for future work: the integration of an end-to-end reliability mechanism with IFRC, a more complete validation of our analysis of IFRC parameters, and a complete analysis of the impact of other parameters on IFRC performance.

References

[1] H. Balakrishnan, H. Rahul, and S. Seshan. An Integrated Congestion Management Architecture for Internet Hosts. In *SIGCOMM '99*.

[2] P. Bergamo, S. Asgari, H. Wang, D. Maniezzo, L. Yip, R. E. Hudson, K. Yao, and D. Estrin. Collaborative Sensor Networking Towards Real-Time Acoustical Beamforming in Free-Space and Limited Reverberance. In *IEEE Trans. Mob. Comput.*, 3(3):211–224, 2004.

[3] V. Bharghavan, A. J. Demers, S. Shenker, and L. Zhang. MACAW: A Media Access Protocol for Wireless LAN's. In *SIGCOMM '94*.

[4] Cheng Tien Ee and Ruzena Bajcsy. Congestion Control and Fairness for Many-to-One Routing in Sensor Networks. In *SenSys '04*.

[5] A. Demers, S. Keshav, and S. Shenker. Analysis and Simulation of a Fair Queueing Algorithm. In *SIGCOMM '95*.

[6] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, 1(4), 1993.

[7] R. Govindan, E. Kohler, D. Estrin, F. Bian, K. Chintalapudi, O. Gnawali, S. Rangwala, R. Gummadi, and T. Stathopoulos. Tenet: An Architecture for Tiered Embedded Networks. *CENS Technical Report 56*, 2005.

[8] B. Greenstein, A. Pesterev, C. Mar, E. Kohler, J. Judy, S. Farschi, and D. Estrin. Collecting High-Rate Data Over low-rate Sensor Network Radios. *CENS Technical Report 55*, 2005.

[9] G. Holland and N. Vaidya. Impact of Routing and Link Layers on TCP Performance in Mobile Ad-Hoc Networks. In *IEEE WCNC '99*.

[10] J. W. Hui and D. Culler. The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale. In *SenSys '04*.

[11] B. Hull, K. Jamieson, and H. Balakrishnan. Mitigating Congestion in Wireless Sensor Networks. In *SenSys '04*.

[12] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In *MobiCom '00*.

[13] V. Jacobson. Congestion Avoidance and Control. *ACM CCR*, 18(4), 1988.

[14] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu. Impact of Interference on Multi-Hop Wireless Network Performance. In *MobiCom '03*.

[15] J. Li, C. Blake, and Douglas S. J. De Couto and Hu Imm Lee and Robert Morris. Capacity of Ad-Hoc Wireless Networks. In *Mobicom '01*.

[16] D. Kim, C.-K. Toh, and Y. Choi. TCP-BuS: Improving TCP Performance in Wireless Ad-Hoc Networks. In *ICC '00*.

[17] V. Kottapalli, A. Kiremidjian, J. P. Lynch, E. Carryer, T. Kenny, K. Law, and Y. Lei. A Two-Tier Wireless Sensor Network Architecture for Structural Health Monitoring. In *Proc. of SPIE's 2003*.

[18] S. Kunniyur and R. Srikant. Analysis and Design of an Adaptive Virtual Queue (AVQ) Algorithm for Active Queue Management. In *SIGCOMM '01*.

[19] P. Levis, N. Patel, D. Culler, and S. Shenker. Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks In *NSDI '04*.

[20] P. E. McKenney. Stochastic Fairness Queuing. In *INFOCOMM '90*.

[21] K. Mechitov, W. Y. Kim, G. Agha, and T. Nagayama. High-Frequency Distributed Sensing for Structure Monitoring. In *INSS '04*.

[22] J. Paek, K. Chintalapudi, J. Cafferey, R. Govindan, and S. Masri. A Wireless Sensor Network for Structural Health Monitoring: Performance and experience. In *EmNetS '05*.

[23] M. Rahimi, R. Baer, J. Warrior, D. Estrin, and M. B. Srivastava. Cyclops: In-situ Image Sensing and Interpretation in Wireless Sensor Networks. In *SenSys '05*.

[24] Y. Sankarasubramaniam, O. Akan, and I. Akyildiz. ESRT: Event-to-Sink Reliable Transport in Wireless Sensor Networks. In *MobiHoc '03*

[25] P. Sinha, N. Venkitaraman, R. Sivakumar, and V. Bharghavan. WTCP: A Reliable Transport Protocol for Wireless Wide-Area Networks. In *Mobicom '99*.

[26] A. Sridharan and B. Krishnamachari. Max-Min Fair Collision-Free Scheduling for Wireless Sensor Networks. In *Workshop on Multihop Wireless Networks (MWN'04), IPCCC*.

[27] F. Stann and J. Heidemann. Rmst: Reliable Data Transport in Sensor Networks. In *SNPA '03*.

[28] R. Szewczyk, A. Mainwaring, J. Polastre, and D. Culler. An Analysis of a Large Scale Habitat Monitoring. In *SenSys '04*.

[29] C.-Y. Wan, A. T. Campbell, and L. Krishnamurthy. PSFQ: A Reliable Transport Protocol For Wireless Sensor Networks. In *WSNA '02*.

[30] C.-Y. Wan, S. B. Eisenman, and A. T. Campbell. CODA: Congestion Detection and Avoidance in Sensor Networks. In *SenSys '03*.

[31] A. Woo and D. Culler. A Transmission Control Scheme for Media Access In Sensor Networks. In *Mobicom '01*.

[32] A. Woo, T. Tong, and D. Culler. Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks. In *SenSys '03*.

[33] K. Xu, M. Gerla, L. Qi, and Y. Shu. Enhancing TCP Fairness in Ad-Hoc Wireless Networks Using Neighborhood RED. In *MobiCom '03*.

[34] W. Ye, J. Heidemann, and D. Estrin. An Energy-Efficient MAC Protocol for Wireless Sensor Networks, In *Infocom '02*.

[35] H. Zhang, A. Arora, Y. ri Choi, and M. G. Gouda. Reliable Bursty Convergecast in Wireless Sensor Networks. In *MobiHoc '05*.