The Case for a Network Protocol Isolation Layer

Jung Il Choi, Maria A. Kazandjieva, Mayank Jain, and Philip Levis
Computer Systems Laboratory
Stanford University
Stanford, CA USA
jungilchoi, mariakaz, mayjain@stanford.edu, pal@cs.stanford.edu

Abstract

Network protocols are typically designed and tested individually. In practice, however, applications use multiple protocols concurrently. This discrepancy can lead to failures from unanticipated interactions between protocols.

In this paper, we argue that sensor network communication stacks should have an isolation layer, whose purpose is to make each protocol's perception of the wireless channel independent of what other protocols are running. We identify two key mechanisms the isolation layer must provide: shared collision avoidance and fair channel allocation.

We present an example design of an isolation layer that builds on the existing algorithms of grant-to-send and fair queueing. However, the complexities of wireless make these mechanisms insufficient by themselves. We therefore propose two new mechanisms that address these limitations: channel decay and fair cancellation. Incorporating these new mechanisms reduces the increase in end-to-end delivery cost associated with concurrently operating two protocols by more than 60%. The isolation layer improves median protocol fairness from 0.52 to 0.96 in Jain's fairness index. Together, these results show that using an isolation layer makes protocols more efficient and robust.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*; C.4 [**Performance of Systems**]: Reliability, availability, and serviceability

General Terms

Design, Performance, Reliability

Keywords

Protocol Isolation, Wireless Network Architecture, Fairness, Isolation Layer

1 Introduction

Imagine a general-purpose operating system without process isolation. As a single memory bug can corrupt any and all programs, an administrator must heavily test every system configuration before use. Debugging failures is exceedingly difficult: one must explore the entire system.

Unfortunately, this is the situation in sensor network protocols today. Applications run multiple protocols which can and do interact, often in negative ways. Protocols seem to work well in the lab but perform poorly in deployment. Identifying the exact cause of a failure is extremely costly and difficult. A developer has to consider the entire cross-product of possible interactions: in practice, deployments often cannot definitively state the cause of observed failures [35].

The experiences of researchers at Delft University are one example of inter-protocol interactions causing a network wide failure [24]. Bursty packet transmissions from Deluge [17] suddenly changed link qualities and caused the MintRoute [39] routing topology to collapse, resulting in a 2% data delivery ratio at the gateway. This class of failures is not unique to this example; it stems from a more fundamental cause – the inter-protocol interference that current sensornet systems cannot prevent. Even if one can find the exact cause of a bug, it is hard to fix it since it is not the fault of any single protocol. For event-driven sensor networks which have long periods of quiescence punctuated by bursts of high traffic, such as seismic detection networks [37], providing isolation within these bursts of activity is critical.

Just as an operating system isolates processes, a network architecture must isolate protocols. Isolation means that a network protocol should minimally affect the behavior of other protocols. The packet reception ratios a protocol observes should be independent of what other protocols are running. If a network architecture provides isolation, then it minimizes inter-protocol interactions, making it easier to design protocols and compose services into larger, more complex applications.

Protocol isolation is not a sufficient requirement by itself. For example, a network architecture could provide perfect isolation by only allowing one protocol to transmit. While such an architecture would have no inter-protocol collisions, this perfect isolation comes at the cost of starvation. Just as operating systems schedules the CPU between processes, a network architecture must also schedule access to the wireless channel between protocols. This scheduling should provide *protocol fairness*, which gives each protocol a fair share of the channel, such that no protocol starves. Simple MAC-layer fairness, where each node gets an equal share of the channel, is insufficient, as protocols receive a share of the channel proportional to their number of transmitting nodes. This fairness penalizes protocols with coordinated transmitters, such as Deluge. A protocol fairness scheme should provide fair access to the wireless medium for every protocol, while making sure that no protocol is starved at any node. Section 4.1 discusses these aspects in detail.

This paper proposes introducing an isolation layer into the networking stack. An isolation layer sits between the MAC and network layers, coordinating medium access to provide protocol isolation and fairness. The paper presents the details of an example isolation layer implementation that achieves isolation and fairness using two mechanisms, *shared collision avoidance* and *fair queueing* across protocols. Shared collision avoidance allows multiple network protocols to avoid collisions with one another. Fair queueing guarantees that no protocol starves. Combining these two mechanisms in a separate software layer provides an isolated channel environment without requiring significant modifications to existing MAC or network protocols.

To provide its shared collision avoidance, the isolation layer uses grant-to-send (GTS), a recently proposed collision avoidance algorithm. We defer an overview of grant-to-send to Section 3 and a more detailed description to a technical report [8], but in summary it allows a protocol to tell nearby nodes how long it expects a packet recipient to use the channel. Many network protocols today incorporate local collision avoidance mechanisms, such as packet suppression and retransmission timers. For example, a unicast routing protocol tells nearby nodes it expects the packet to be forwarded, while a binary dissemination protocol like Deluge tells nearby nodes that it expects to hear a flurry of packets. The key insight is that pushing this functionality into the isolation layer sitting below these protocols allows them to share information and avoid inter-protocol collisions.

To provide protocol fairness, the isolation layer extends the classic fair queueing algorithm by Demers et al. [10] and fair scheduling by Vaidya et al. [34] with two novel mechanisms: fair cancellation and channel decay. These mechanisms are necessary in order to allocate a shared wireless channel – rather than the private wired channel of traditional fair queueing – among multiple nodes with different packet loads for broadcast as well as unicast protocols.

Since multiple nodes can be the source of packets for a particular protocol, each node needs to track the cumulative channel occupation time over all senders. Lost packets lead to inconsistent accounting between nodes, resulting in uneven channel shares for different protocols. Channel decay periodically adjusts local node state on how long protocols have used the channel: this smoothes out inconsistencies caused by the lossiness of the wireless channel. Fair cancellation minimizes the adverse effect Vaidya's delay algorithm has on throughput. These protocol fairness mechanisms can be trivially extended to provide per-protocol-instance fairness, as Section 6.2 shows.

We evaluate the effectiveness of this isolation layer and its mechanisms through a series of testbed experiments. We use end-to-end delivery cost, defined as the total number of transmissions per goodput, to quantify the effectiveness of isolation. Concurrent operation of a collection and a dissemination protocol shows that end-to-end delivery cost can increase by up to 72% compared to the case where each protocol runs in isolation. The isolation layer with GTS can reduce this increase in cost by more than 60%. In addition, our protocol fairness techniques restore fairness in the testbed. For example, the ratio of transmitted bytes between a collection and a dissemination protocol is improved from 1:21 to 1:3.7 in a single-hop scenario.

To the best of our knowledge, this paper is the first attempt to explore a new domain of the fairness problem, protocol fairness in a wireless channel. As more multihop network protocols that exploit the broadcast nature of the wireless medium emerge, such as MORE [7] and MIXIT [21], fairly scheduling and queueing them is critical for extensible systems that can leverage multiple protocol options.

Enforcing per-protocol fairness at the isolation layer does not preclude fairness on other layers, such as per-flow, per-node, or per-user fairness. In fact, these fairness principles are meaningful only on top of per-protocol fairness. Per-node fairness in collection protocol is of limited use when a dissemination protocol starves the collection protocol.

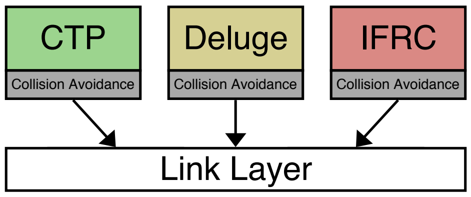
Fairness and isolation are orthogonal concerns to energy efficiency: they only affect system behavior when a network is under contention. Many ultra-low power monitoring networks, for example, have such low utilization in their report cycle that fairness and isolation algorithms do not alter their behavior. However, these principles allow networks to better handle periods of significant load, such as during reprogramming events. Similarly, in event-driven networks that exhibit long periods of dormancy punctuated by periods of intense network-wide activity, fairness and isolation can prevent unforeseen interactions and failures during important events. Correspondingly, fairness and isolation neither aid nor hinder energy efficiency, except for possibly preventing failures that cost energy.

The rest of this paper is structured as follows. Section 2 introduces the abstraction of an isolation layer and the mechanisms we use to implement it. Section 3 presents GTS and describes how it can be used as a shared collision avoidance mechanism. Section 4 examines different kinds of fairness and how to achieve protocol fairness within the isolation layer. With experiments in various settings, Section 5 and Section 6 explore how GTS and fairness schemes affect protocol behavior and isolation.

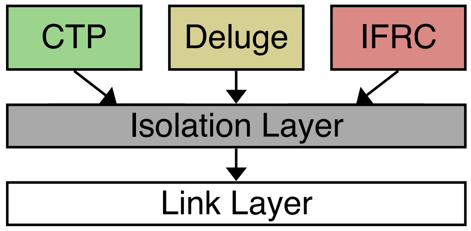
2 Isolating Network Protocols

This section argues that protocol isolation is necessary. In addition, it identifies two key principles that any proposed isolation layer should employ – a shared collision avoidance mechanism and fairness across protocols.

The ability to use multiple layer 3 protocols concurrently is a common requirement in large wireless sensor systems. Rather than use a single layer 3 protocol, such as IP, these networks improve their energy efficiency by using a variety



(a) Current network architecture with individual collision/congestion avoidance mechanisms



(b) Network architecture with isolation layer of shared mechanisms

Figure 1. Individual collision avoidance mechanisms in the current architecture do not work across protocols; a new isolation layer can organize and share collision information.

of protocols, each one optimized for a different workload. MintRoute [40], CTP [2], MultihopLQI [1], IFRC [29] and other collection protocols build minimum-cost trees to data sinks such as gateways. These collection protocols establish flows up a tree to pull data out of a network. Dissemination protocols, such as Deluge [17] and MNP [36], use leader elections and flurries of broadcasts to push data – e.g. new programs – into a network. Applications often use additional protocols beyond these two basic data flows, such as time synchronization [27] for data time-stamping and geographic [20] or coordinate [6, 12, 26] routing protocols for in-network storage [11, 30].

Typically, however, each of these protocols is designed, developed, and evaluated separately, hoping that it operates as well when other protocols are present. Designing a system with multiple protocols as building blocks can easily run into unforeseen interactions and complications. This discrepancy makes the design phase complicated, performance unpredictable, and debugging difficult.

This paper argues that improving the isolation between layer 3 protocols will greatly simplify the design and implementation of efficient wireless sensor systems. Just as an operating system simplifies building complex system on a single node by isolating processes, a network that isolates protocols would enable each one to be developed, tested, and optimized independently.

Given multiple layer 3 sensornet protocols, there are two ways to achieve protocol isolation. The first is to design all layer 3 protocols such that they respect each other channel requirements. This is similar to the way non-TCP protocols in the Internet are required to be TCP-friendly [16]. The second option is to implement a new mechanism that sits between layers 2 and 3 and ensures that no one protocol interacts badly with the rest minimizing the modifications to the layer 3 protocols. Changing all existing sensornet network protocols would be infeasible. Thus, this paper uses the second route and shows that it is an effective way to achieve protocol isolation.

2.1 Shared Collision Avoidance Mechanism

Layer 3 protocols often have mechanisms to avoid interference as in Figure 1(a). For example, CTP delays the next transmission for a random period (16∼31ms) to give time to the previous packet to be forwarded out of interference range by using a transmission timer that prevents back-to-back transmissions. Deluge can suppress transmissions from neighbors while a node is receiving data bursts [35], since data bursts can easily collide with these transmissions.

These mechanisms do not work across protocols: a node can transmit non-CTP traffic during CTP's backoff or a neighbor can send non-Deluge traffic to a node that is receiving a binary update. MintRoute [39] has been reported to break when Deluge coexists in a deployment [24]. That is, the bursts of data packets cause excessive collisions with the control packets of MintRoute, collapsing the network topology. Section 5 also shows that concurrent operation of CTP and Deluge decreases the efficiency of both protocols, increasing the end-to-end delivery cost by 24% and 72% respectively. Many other deployments have also reported low data yields [3, 4, 31, 32]. While the causes are mostly uncertain, we believe many of them to be the interactions between network protocols.

To deal with undesired interactions, some protocols simply assume inter-protocol collisions do not exist. Flush [22] assumes it has complete control of the channel and supports a single flow. While such strict partitioning may be acceptable for application-level workloads, it precludes concurrent services such as management, time synchronization, code distribution, or localization.

Another approach is building vertical software stacks that tweak and couple existing protocol implementations. During a recent deployment [37], researchers had to rewrite significant portions of the MultihopLQI protocol in order to make it work with time synchronization and network event detection. Some systems consider the problem so acute that they explicitly prohibit introducing new layer 3 protocols; the Tenet system architecture explicitly constrains low power sensors to only use data collection trees, citing the "fragility and unmanageability" of introducing data fusion protocols [14].

We propose a shared underlying mechanism to isolate network protocols as shown in Figure 1(b). The isolation layer organizes collision avoidance information of network protocols so that a requirement to avoid collisions from one protocol can be respected by all other protocols. For example, Deluge can notify the isolation layer of the collision vulnerable period of data bursts, and the isolation layer can suppress MintRoute's control packets accordingly, avoiding loss of the control packets.

Thus, the goal of this mechanism is to:

Minimize the effect of concurrently running protocols on the packet delivery ratio observed by any given protocol.

In practice, we use the end-to-end cost of delivering a packet to measure the degree of protocol isolation. We chose this metric over packet delivery ratio on a single link since layer 3 mechanism of choosing the next hop affects link packet delivery ratio; layer 3 protocols can switch to a longer path when a link becomes congested.

In operating systems, designers do not worry about the case where some part of the memory is corrupted by other processes. With an isolation mechanism in place, network protocols can be designed and deployed without worrying about unforeseen protocol interactions.

2.2 Fairness

Shared collision avoidance itself is not enough. The behavior of a system would be far from intended if a protocol becomes starved due to other protocols. For example, if a time synchronization protocol starves due to heavy transmissions from a collection protocol, all delivered data could be meaningless. As traditional systems put much caution in fairly distributing CPU time among processes, network protocols must get a fair share of the bandwidth.

Fairness research has concentrated on IP-based, store-and-forward networks with per-flow fairness [10, 13, 19, 25, 28, 34]. However, it is not immediately obvious how to apply this kind of fairness schemes to per-protocol fairness. Since each network protocol has a unique goal and an optimized traffic pattern, there does not exist a unified end-to-end metric that can work across protocols. For example, delivering one packet in a collection protocol is not directly comparable to delivering one packet in a dissemination protocol.

Therefore, the mechanisms in this paper strive to achieve a single-hop fairness goal: distribute the channel around a node equally for each protocol. Also, counting the number of bits each protocol access the channel is not enough because collision avoidance introduces wait times. When one protocol makes other protocols wait longer yet sends the same number of bits, it must be penalized.

Thus, the second principle of an isolation layer is to:

Provide a fair share of channel time around each node to each protocol, including wait times.

A major challenge is that all nodes can be data sources of each protocol and this does not fit the traditional notion of flow fairness. Since the channel around each node has a different set of sources for each protocol in multihop network, the state inconsistency is inherent. In this case, it may not be even possible to achieve a perfect fairness at all.

In the following two sections, we explore specific mechanisms for implementing an isolation layer in accordance to the two principles above.

3 Shared Collision Avoidance

Grant-to-send (GTS) is a collision-avoidance mechanism proposed by Choi et al. [8] that can be used to provide isolation between multiple network protocols. Every link-layer packet contains a grant duration field, which grants the channel around the transmitter to the receiver of the packet. That is, upon overhearing or transmitting a packet, nodes can only transmit after the grant duration has expired, and only the receiver of the packet can transmit immediately.

The local collision avoidance mechanisms of CTP can be expressed using GTS. Section 2 introduced CTP's transmission timer – a simple mechanism that delays the next packet transmission to give time for the previous packet to be forwarded out of interference range. This mechanism can be replaced by GTS by including a grant of one packet time in each sent packet. Since a grant silences its originator, the recipient of the packet will have time to forward the data up the collection tree.

The lack of a layer 3 collision avoidance mechanism in Deluge has been shown to cause collision losses during a bursty transmission. Therefore, an improved version, V-Deluge [35], augments a request packet to silence neighbors while the data exchange takes place. This approach is equivalent to the grants that GTS supports; a network architecture that employs GTS can use regular Deluge where request packets carry a grant for the duration of the data burst.

GTS is a general mechanism that can be effectively applied to protocols that exhibit correlated packet transmissions. Ideally, the grant included in one packet will prevent collisions for the packet or packets that follow. We believe that because of its generality, GTS can act as a common language that different network protocols can use to communicate collision avoidance information with each other.

However, the authors of GTS do not explore what happens when multiple protocols use GTS at the same time. The key difference between using V-Deluge and GTS is that the first affects only packets from Deluge but the latter affects packets from every protocol since GTS is a MAC mechanism. This property enables GTS to be used as a shared mechanism for the isolation layer, enforcing the grants on all protocols. For example, if Deluge requires silence while a node receives a burst of data packets, all other protocols are held in the isolation layer, providing silence as required.

When GTS is used by multiple network protocols concurrently, it is possible that a node is granted by one protocol but sends on behalf of another. GTS specifies neither the destination of the granted packet, nor the protocol it should come from. This is important since otherwise one protocol could easily starve the rest or two nodes could take over the channel. At the same time, this underspecification still leaves space for unfairness in the way channel time is shared between protocols and nodes. The following section discusses this issue in more detail.

Ideally, GTS requires overhearing of all packets in the vicinity of a node. In practice, nodes can have their radios turned off during periods of low contention. In such periods, a node may not overhear packets, and hence not update its GTS timer. However, since a node in low-power state is not contending for the channel, collision avoidance is not crucial in this case.

4 Fairness

GTS introduces a significant fairness problem: a single packet's grant can vary by two orders of magnitude (1-255ms). If the underlying transmission scheduler operates on a packet basis, then protocols which issue large grants can request a much larger share of the channel than those which issue small grants.

Furthermore, sensornet protocols vary in their traffic patterns: in routing protocols like CTP, every node may be trying to send to their next hop, while in dissemination protocols like Deluge, a single node sends a burst of packets. MAC-layer fairness says that each node should have an equal chance of acquiring the channel, but this policy penalizes protocols, such as Deluge, that centralize their transmissions.

The isolation layer provides protocol fairness with two classic algorithms: fair queueing by Demers et al. [10], decides which packet a node should send, and fair scheduling by Vaidya et al. [34], extends this algorithm to the shared medium of wireless by controlling when a node sends.

By themselves, these two algorithms are insufficient: the lossy nature of a wireless channel and the realities of packet radio behavior introduce problematic edge conditions and scheduling challenges. The isolation layer introduces two novel mechanisms, channel decay and fair cancellation, to address these challenges. The rest of this section describes these classic algorithms, their limitations, and the isolation layer's solutions in greater detail. First, however, we define metrics for fairness in order to quantify the issues fairness encounters in wireless networks.

4.1 Metrics

Traditionally, fairness and fair queueing are concerned with whether the throughput on a wired channel is allocated fairly. This model assumes that a single node transmits on the channel. In wireless, however, many nodes share the same channel. Furthermore, unlike fairness across unicast flows, fairness across protocols means many neighbors may have transmissions.

We define three fairness metrics, all measured by Jain's Fairness Index (JFI)[1] [18]: channel fairness, transmit fairness, and protocol fairness. Let O_N^p be how long protocol p has occupied the channel at node N, and T_N^p be the amount of time node N has spent sending packets of protocol p. We define the metrics as follows:

- **Channel fairness** refers to the time that a protocol's packets occupy in the wireless channel. Channel fairness for a node N is $\forall x : JFI(O_N^x)$, that is, the JFI over channel occupancy by protocols at that node.
- **Node fairness** is the time different nodes spend sending a particular protocol. Node fairness is the MAC notion of fairness: a scheme is node fair if each node with packets to send for a protocol receives an equal share of the channel. Node fairness for a protocol p is therefore $\forall Y : JFI(T_Y^p)$, that is, the JFI over node transmissions for a protocol.
- **Transmit fairness** is the time a particular node spends sending protocols. It is the traditional notion of fairness in queueing: it represents what share of a node's transmit time each protocol occupies. Transmit fairness for a node N is therefore $\forall x : JFI(T_N^x)$.

A primary goal of the isolation layer is channel fairness: each protocol should receive a fair share of the medium. One

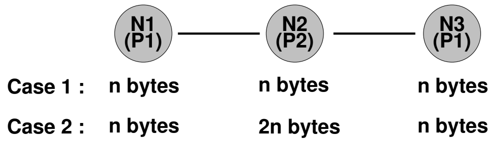


Figure 2. A multihop example where it is impossible to obtain channel fairness for all nodes. N1 and N3 wish to transmit protocol P1, while N2 wishes to transmit protocol P2. Case 1 achieves channel fairness only for N1 and N3, while Case 2 achieves fairness only for N2.

challenge that protocol fairness in wireless networks introduces is that all nodes are potential sources. The simplest way to achieve channel fairness is to have a single node transmit, reducing a distributed problem to a centralized one. But doing so has a node fairness of zero: all other nodes starve.

Furthermore, multihop networks can have workloads where it is impossible to simultaneously provide fairness for one or all of the metrics at all nodes. Figure 2 shows an example case for channel fairness in a 2-hop 3-node network. In case 1 in the figure, where all nodes transmits n bytes, N1 and N3 see a total of n bytes from both protocols, but N2 sees $2N$ bytes from P1 and n bytes from P2. N1 and N3 achieves channel fairness but N2 does not. In case 2, where N2 transmits $2N$ bytes, N2 sees $2N$ bytes from both protocols, but N1 and N3 see a total of n bytes from P1 and $2n$ from P2. In this case, channel fairness is achieved only at N2.

While the isolation layer places channel fairness as its most important goal, it does so while balancing its needs for node and transmit fairness as well.

4.2 Basic Algorithm

The isolation layer uses the basic idea of simple fair queueing [10]. A node N maintains a table of how long each protocol p has occupied the channel (O_N^p). On each transmission or reception, the node adds time to the associated table entry. The isolation layer queues packets such that the next packet is always from the protocol with the smallest O_N^p.

Traditionally, the update is proportional to a packet's length. However, since GTS packets reserve the channel and suppress other transmitters, the isolation layer includes grant durations in its calculations. When the isolation layer updates O_N^p, it adds not only the packet airtime but also that packet's grant duration. This scheme is equivalent to virtual tags from the fair queueing literature [10, 25, 34], with the exception of using grant durations to update channel time.

Fair queueing is insufficient for channel fairness when there are multiple transmitters. For example, if there are 4 nodes, and three wish to transmit protocol P_1 while only one wishes to transmit protocol P_2, then three times out of four a P_1 node will win CSMA backoff and transmit. In addition to fair queueing, fairness requires fair scheduling, which controls when nodes transmit.

For fair scheduling, we borrow ideas from Vaidya et al. [34] and Luo et al. [25] and introduce an additional wait time before CSMA backoff. This wait the isolation layer to skew the probability that a given node will win CSMA.

While good starting points to establish protocol fairness,

[1] $JFI = \frac{(\sum x_i)^2}{n \cdot \sum {x_i}^2}$, where n is the number of elements. The worst fairness is $1/n$, and the best is 1.

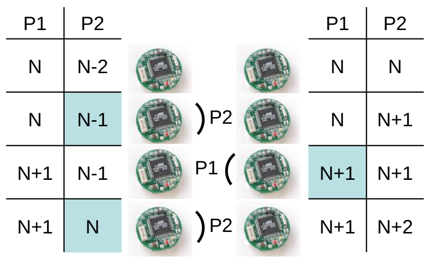


Figure 3. An example case of the ping-pong effect with two nodes sending two protocols. The two tables show the state of each node over time. A state inconsistency, which can be caused by packet losses, can result in a low transmit fairness, with the left node sending only P2 and the right node sending only P1.

each of these algorithms has limitations when used in multi-hop wireless networks: in practice they can exhibit poor fairness. The next two subsections examine these limitations in detail, and describe the mechanisms the isolation layer uses to achieve good channel, transmit, and node fairness.

4.3 Fair Queueing

To explain a major challenge fair queueing encounters in a wireless network, we start with a very simple case: two nodes both send packets from two protocols P_1 and P_2 as fast as possible. If there are no packet losses, this network will achieve node, channel, and transmit fairness. Each node has an equal chance of acquiring the channel, and on doing so will transmit the protocol with a smaller channel time.

Packet losses, however, complicate this situation. If a node fails to hear a transmission, then the channel occupancy tables on the two nodes become inconsistent: the transmitter has incremented channel occupancy but the receiver has not. Because each node is queueing packets based on its own local view of the channel, an inconsistency can lead to a "ping-pong" effect, where the two nodes disagree on which protocol has used the channel less. It can cause low transmit fairness.

Figure 3 shows an example of the ping-pong effect in the two node, two protocol case. For simplicity, a transmission of each protocol increments the value of O_N^p by one. Suppose that both protocols have accessed the channel N times but, for some reason, N1 has missed two packets from P2. When N1 accesses the channel again, it sends a packet from P2 trying to equalize the protocols. Upon hearing this packet, N2 increments the channel access history for P2 by one. If N2 accesses the channel next, it will send P1, balancing out its protocol table. However, this packet reverts the effort of N1 trying to achieve equality. Thus, at the next chance N1 will transmit P2 again, and this cycle goes on. Eventually, N1 will be biased for P2, and N2 for P1. This communication schedule has perfect channel fairness but low transmit fairness.

Unfortunately, the ping-pong effect is not an edge case that rarely happens; inconsistency can occur from packet losses, collisions, or even different boot times. Furthermore, in multihop networks, each node typically has a different view of the channel as it hears transmissions from a different set of neighbors.

Prior work by Luo et al. [25] showed how, in the case of unicast flows, a transmitter can embed the flow's channel occupancy in a data packet. Nodes overhearing a packet can use the information to update their table and restores consistency. But in the case of protocol fairness, virtual tags are not easily synchronizable because there can be many transmitters: doing so would require each node to maintain $O(np)$ space, where n is the number of neighbors and p is the number of protocols.

4.4 Channel Decay

The isolation layer solves the problems of the ping-pong effect using the simple, local mechanism of *channel decay*, where it periodically decays the times in the channel occupancy table by half. Channel decay has a similar effect as an exponentially weighted moving average (EWMA), bounding the effect of past packet losses. At each decay interval, the inconsistency between protocol tables is halved, while consistent table entries remain consistent.

Channel decay provides a second benefit: it bounds the time interval over which the isolation layer computes fairness. If a protocol has a channel occupancy of t, then after $log(t)$ intervals channel decay will have reduced this occupancy to zero.

4.5 Fair Scheduling

Prior sections examined how packet losses can lead to poor fairness in even a simple two node, two protocol workload. A fair queueing mechanism can achieve fairness for uniform loads because locally optimal decisions are globally optimal. This section examines how networks with non-uniform loads introduce further complications.

For example, suppose there is one node sending Protocol 1 (P1), two nodes sending P2, and four nodes sending P3. All protocols have identical grant durations and packet lengths. Each node has a single protocol to send, so local fair queueing always sends that protocol. As MAC-layer node fairness means each node receives an equal share of the channel, the differing number of senders causes the channel shares for the protocols to be 1:2:4. Running such an experiment on TelosB nodes showed a poor channel fairness of 0.7748.

Fair scheduling changes the channel access behavior by penalizing protocols that have used the channel more. After the isolation layer decides which protocol to send based on fair queueing, it introduces a penalizing delay before CSMA backoff called a protocol penalty. Protocols with a greater channel use have a larger protocol penalty. Delaying packet transmissions is an established technique for fairness in flow-based networks [28, 34]. In these algorithms, whenever a node receives a packet, it cancels its current transmission and resubmits it to the CSMA layer.

Determining the length of the penalizing delay requires two decisions: quantifying a protocol's share of the channel and deciding the actual value of the delay from the share. For the first decision, the isolation layer uses the following equation:

Name	Definition	Description
Null	$f(x) = 0$	No penalty
Linear	$f(x) = x - 1$	Linear function
Log	$f(x) = 10log_{10}x$	Log function
Exp	$f(x) = 10exp(x - 10)$	Exponential function
Prob	$f(x) = 10 - 10\sqrt{\frac{2}{1+x^2}}$	TX prob. of $1/x^2$ with two nodes
Const	10ms delay for successive TX	Penalty for successive TX

Table 1. Various examples of the penalty function for 802.15.4. For all cases, the minimum value is zero for $x = 1$ and the maximum value is 10ms.

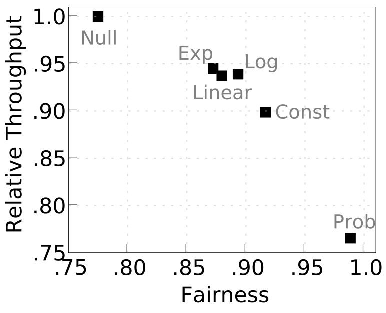


Figure 4. Throughput-fairness tradeoff of various penalty functions when three protocols have 1, 2, and 4 senders in single-hop topology.

$$\text{Share}(x) = \frac{P_x}{\min_j(P_j : P_j \neq 0)}$$

where P_i denotes the channel occupancy of protocol i in the protocol table. When the load is uniform, P_x is always $\min_j(P_j : P_j \neq 0)$: it is the minimum value in the table.

The above equation can lead to unnecessarily large penalties when some protocols fall idle. For example, if one protocol only sends packets at boot, its P_i will always be a small but non-zero value. Other protocols will have large shares, and therefore large penalties, in the chance that this protocol may transmit again. Channel decay, however, solves this problem: over time, inactive protocols will have their channel occupancy reach zero. Following the above equation, the isolation layer does not consider these inactive protocols when calculating shares.

The second decision the isolation must make is assigning a protocol penalty based on a channel share. The isolation layer uses the following equation:

$$\text{Penalty(x)} = f(\text{Share}(x))$$

Table 1 lists six possible functions for $f(x)$. While the functions are described in terms of the TinyOS 2.1 MAC layer for 802.15.4, which has a 10ms maximum backoff, they are easily generalizable to other CSMA schemes. Note that since the local mechanisms of the isolation layer do not know the number of nodes contending for the channel, deriving a specific channel access probability from penalty functions is not possible.

A strawman algorithm that will achieve fairness is to sort protocols by channel occupancy and assign penalties based on the maximum CSMA backoff. The protocol with the least occupancy has no delay, the second least delays a maximum backoff, the third delays twice the maximum backoff, and so on. This provides fairness, but greatly harms throughput. In the case of the TinyOS 2.1 CSMA layer, backoff is very large ($\sim$10ms) with respect to the packet air time($\sim$2ms).

Figure 4 shows throughput and fairness of the various penalty functions. We can observe a tradeoff between throughput and fairness, with various functions achieving different points on the curve. The slope patterns of the penalty functions decide which function gains in fairness or in throughput. In this experiment, all protocols have unlimited packets. This means that there always exists a node which tries to send the protocol with the minimum channel usage. Therefore, the initial slope of the penalty function is the deciding factor for this case.

4.6 Fair Cancellation

The fair scheduling algorithm described above requires resetting the backoff timer whenever nodes hear transmissions. As TinyOS uses a long backoff interval compared to packet airtimes, cancelling a transmission greatly harms throughput: a packet must be reloaded into the radio, and go through another backoff. This is in contrast to 802.11 networks, where packets can be loaded very quickly on high speed buses and backoffs are very small. For example, Section 6.2 presents an example case where always cancelling transmissions can reduce throughput by as much as 36%.

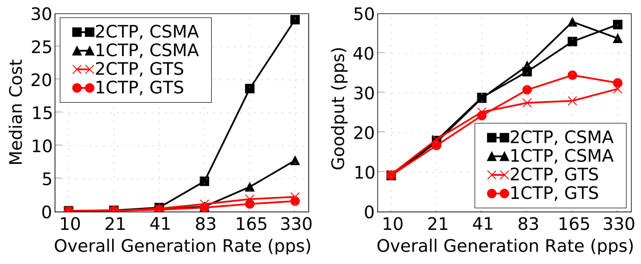
In order to achieve fair scheduling without harming throughput, the isolation layer uses *fair cancellation.* On hearing a transmission, nodes update the channel access table normally. However, they do not cancel the packet in CSMA layer if it belongs to the protocol has the minimum channel occupancy after the update. Since this is the packet that should be transmitted next, cancelling it limits the overall packet rate of the network. By not cancelling packets of the minimum protocol, fair cancellation prevents the standard fair scheduling algorithm from harming throughput.

Because grants inherently suppress and cancel transmissions, fair cancellation is only relevant in transmission scenarios where there are no outstanding grants, i.e. when protocols are using standard CSMA, and for the recipients of GTS packets.

4.7 Summary of Fairness Schemes

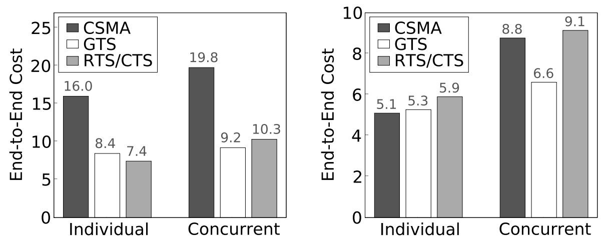
In summary, the isolation layer uses four mechanisms to provide fairness:

- Fair Queueing (FQ) : Mechanism for selecting the next packet in the link layer queue internally. Selects the protocol with the smallest channel occupancy.
- Channel Decay : Multiplicatively decrease all values in the channel occupancy table periodically. Channel decay smooths out table inconsistencies, preventing the ping-pong effect. It also serves as a time-windowing mechanism to determine active protocols.
- Fair scheduling and protocol penalty (PP): Before entering CSMA backoff, the isolation layer delays packets according to the channel occupancy of the protocols.



(a) Median node ETX. (b) Goodput

Figure 5. Median node ETX and goodput of one and two instances of CTP, with and without GTS. Node ETX is defined as the number of transmissions required per successful packet. Collision avoidance in CTP is ineffective for concurrent operation.



(a) Cost for CTP (b) Cost for Deluge

Figure 6. End-to-end delivery cost when each protocol runs alone and concurrently. End-to-end delivery cost for Deluge is the number of data packet transmissions needed to disseminate a page to a node. GTS achieves low costs for individual as well as concurrent operation.

- Fair Cancellation (FC) : Instead of restarting CSMA backoff for all packets, the isolation layer restarts the backoff only for packets which are not from the minimum occupancy protocol. Fair cancellation prevents transmissions from limiting the packet rate by forcing the next packet to re-enter backoff. Fair cancellation applies to the packets already performing CSMA backoffs when non-suppressing packets are received/overheard.

5 Isolation Evaluation

Section 2.1 discussed the need to minimize interactions between protocols via an isolation layer; Section 3 proposed GTS as a shared collision avoidance mechanism for this layer. This section compares the performance of a network in which no isolation exists with one that uses GTS.

5.1 Two instances of CTP

In the first experiment, two separate instances of CTP run concurrently on the Motelab testbed [38]. The experiment uses 165 nodes that generate data at varying rates. In reality, an application would use a single CTP with two different identifiers for data. However, having two CTPs is an effective scenario to reveal if the mechanisms of CTP will work when other protocols are concurrently operating. The GTS grant duration value used is one packet time, 10ms and CTP's transmit timer is disabled when GTS is used because GTS effectively replaces it.

Figure 5 shows the median node ETX for one and two instances of CTP with and without GTS, with aggregate packet generation rate on the x-axis. We use node ETX instead of end-to-end delivery cost because as the network generates up to 330 packets per second only nodes near the root can deliver packets to the root, and we take median to neutralize the effect of isolated nodes.

The results for CSMA demonstrate that the collision avoidance mechanism of CTP does not work across protocols. When the traffic load is light, both standard CSMA and GTS operate efficiently. With standard CSMA, as the traffic load increases, the network with two instances of CTP starts to break while the one with a single instance degrades gracefully.

GTS improves the link quality of a single instance of CTP. Moreover, with two instances, GTS can achieve similar link quality as if each protocol was alone. With 330 pps generation rate, CSMA experiences 273% increase in ETX when it is divided into two instances, while GTS shows an ETX increase of only 39%. In addition, while CSMA suffers a 174-fold increase in ETX when the generation rate is increased from 10 to 330 pps, GTS shows a 15-fold increase which is only 8.8% of the standard CSMA case.

The price for managing the cost of CTP is loss of goodput, as Figure 5(b) shows. The large default CSMA backoff forces collision avoidance mechanisms to be conservative as well.

5.2 CTP and Deluge

The next network scenario examines a collection and a dissemination protocol operating concurrently on the Mirage testbed [9] with 64 nodes. The traffic pattern of CTP simulates event detection that triggers a portion of the nodes to transmit information regarding the event. The initial event is a radio packet and 14 nodes that hear the trigger packet initiate infinite data generation. Source nodes are fixed throughout the experiments to eliminate the effect of variations.

To find out how well isolated each protocol is, we compare two metrics: mean end-to-end delivery cost for CTP and the total number of data packet transmissions per node per page for Deluge. We consider each protocol operating in isolation and both operating concurrently. Although Deluge cannot utilize link layer acknowledgements due to broadcasts, if nodes miss data packets they must request the missing packets again. Thus the number of total Deluge data packet transmissions is an indication of link quality.

Figure 6 shows the end-to-end delivery cost of CTP and Deluge. When CTP operates individually, GTS achieves 47% reduction in the cost without drop in goodput (not shown). This suggests that GTS can successfully replace the already-existing collision avoidance mechanism of CTP. For Deluge, the difference in its efficiency is insignificant for CSMA and GTS, within 4%. This is because the modified version of Deluge [35] used in this experiment has most of the GTS functionality included in the Deluge protocol itself.

When the two protocols operate concurrently, with

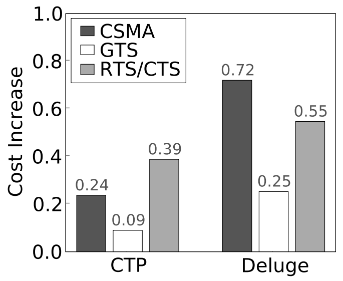
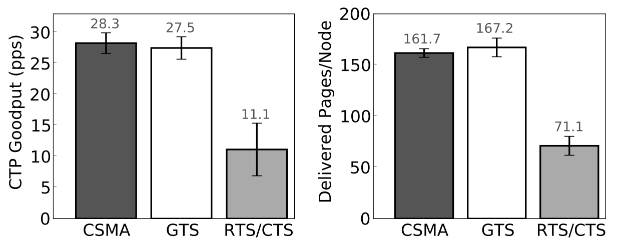


Figure 7. End-to-end delivery cost increase when CTP and Deluge run concurrently. GTS reduces the cost increase compared to CSMA by over 62% and RTS/CTS by over 53%.



(a) CTP Goodput (b) Number of disseminated pages per node

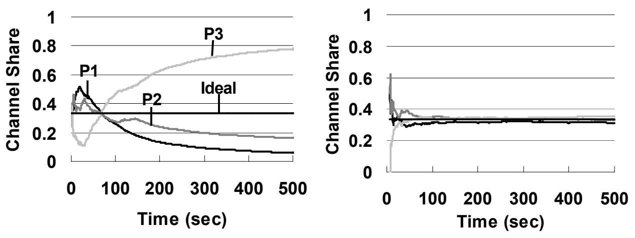
Figure 8. Goodput for CTP and Deluge for concurrent operation. Grant-to-send does not introduce goodput drop.

CSMA only, CTP's cost increases by 24% and Deluge's cost increases by 72%. GTS reduces the cost increases by 62% and 65% for CTP and Deluge respectively, achieving a cost increase of 9% for CTP and 25% for Deluge. At layer 2.5, GTS reduces collisions across layer 3 protocols.

RTS/CTS can be another candidate for the collision avoidance mechanism in the isolation layer, as will be discussed in more detail in Section 7. With RTS/CTS, although individual operation of CTP gives a slightly lower cost than GTS, concurrent operation gives a higher cost. For Deluge, the cost for the concurrent case is even higher than CSMA's. This is because RTS/CTS does not support broadcast packets; the data packets from Deluge are not protected and RTS/CTS exchanges on top of CTP data packets induce even more interference for Deluge packets. In contrast, GTS can provide protection for broadcast packets, achieving the reduction in the cost increase by 53% and 76% for CTP and Deluge respectively.

Figure 8 shows the goodput of CTP and Deluge when operating concurrently, confirming that GTS does not sacrifice performance. Deluge's goodput refers to the number of total disseminated pages per node across the network.

Overall, GTS achieves the best of both worlds. Collision avoidance is as effective as RTS/CTS, with the addition of protection for broadcast packets. At the same time, GTS preserves the efficiency of CSMA in terms of goodput.



(a) Plain fair queueing (b) Fair queueing with channel decay

Figure 9. Channel shares of three protocols transmitted on a node in a 5-node single hop experiment with uniform loads. Simple fair queueing does not provide transmit fairness due to ping-pong effect, but channel decay restores fairness.

6 Fairness Evaluation

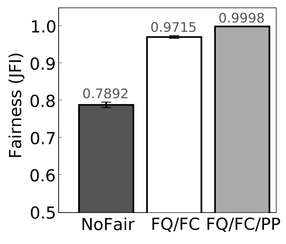
The previous section evaluated how well GTS isolates concurrently operating protocols from one another. This section evaluates the isolation layer's fairness through the mechanisms presented in Section 4, paying particular attention to channel decay and fair cancellation.

6.1 Channel Decay

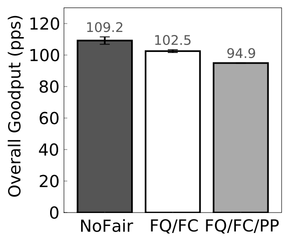
This section quantifies the negative impact of the ping-pong effect and evaluates how well channel decay can restore transmit fairness. In this experiment, three abstract protocols (denoted by P1, P2, and P3) run on five Telosb nodes (N1 through N5), all within communication range of each other. All nodes send packets from each protocol as fast as possible, such that they always have transmissions pending. All packet grants are zero: the channel occupancy table is the duration of packet airtimes. The protocols send packets of different lengths, in a ratio of 1:2:4. If the network has perfect channel and transmit fairness, each node will send packets with a ratio of 4:2:1 and the channel will observe transmissions in the same ratio.

With basic fair queueing, the scenario above achieves a perfect channel fairness of 0.9999. However, the transmit fairness is very low. Figure 9(a) shows the channel shares of each protocol transmitted by N3. The channel share curves are diverging from the ideal 1/3 line: transmit fairness is 0.52. The node transmits protocols unevenly – with a ratio of 1:3:13 instead of 1:1:1. This observation matches the previous description of the ping-pong effect, high channel fairness but low transmit fairness, and inspecting the logs verifies this is what is happening.

Figure 9(b) shows that a channel decay interval of 1s restores channel fairness to 0.9947. As mentioned in Section 4.4, the decay period determines the time window over which the isolation layer is fair: a previous occupancy of t is forgotten after $log(t)$ periods. The period also limits how long a state inconsistency persists. We leave it to future work to analytically determine the decaying period suitable for both. In our implementation and in all experiments, the decay interval is 1 second, which we have found to provide a good tradeoff for different workloads.

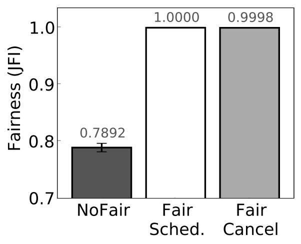


(a) Fairness

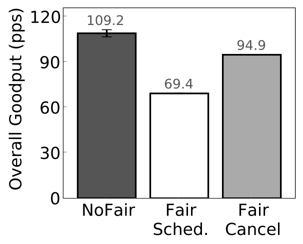


(b) Packet rate

Figure 10. Fairness and packet rate of two CTP protocols running in parallel, one with 22-byte packets and the other with 86-byte packets. The fairness schemes improves fairness at the cost of reduced throughput.



(a) Fairness



(b) Overall packet rate

Figure 11. Comparison of pure fair scheduling and fair cancellation, on top of protocol penalty and fair queueing. Fair cancellation achieves similar fairness as pure fair scheduling while reducing the packet rate drop.

6.2 Single-hop Collection

We study fairness in a single-hop network scenario of two collection protocols running in parallel. In these experiments, all grant durations are zero because there are no hidden terminals and no need for multihop collision avoidance. Therefore, fairness only depends on the number of bits each protocol sends. In such a setting, the fairness scheme directly affects metrics such as the rate at which the network sends packets.

In the first set of experiments, there are two separate copies of CTP active in the network. One copy uses a payload of 22 bytes and the other uses a payload of 86 bytes. Since the header length of CTP is 10 bytes, the network is fair when the CTP with 22 byte packets delivers three times more packets than the CTP with 86 byte packets $(3 \cdot (22+10) = (86+10) = 96)$.

Figure 10 shows results for a six-node network: 2 nodes send short CTP packets, 2 nodes send longer packets, and 2 nodes send both. 'NoFair' is the standard TinyOS stack, 'FQ' is an isolation layer with fair queueing, and 'FC' means the isolation layer also includes fair cancellation. Finally, 'PP' indicates that the isolation layer uses protocol penalties with the probability ("Prob") penalty function. We use this probability function to clearly present the effects of protocol penalty on fairness. Section 4.7 summarized each of the fairness schemes used in these experiments.

In the default TinyOS stack, the CSMA layer gives each node an equal chance of transmission and nodes serve protocols in a round-robin fashion. Therefore, both CTPs have the same number of transmit opportunities and their goodputs are proportional to their packet lengths. This results in a poor channel fairness of 0.7892, as Figure 10(a) shows. Fair queueing with fair cancellation increase channel fairness to 0.9715, and protocol penalties improve it further, to 0.9998.

Figure 10(b) shows the overall network throughput to the collection point. It shows the opposite trend as fairness, demonstrating the tradeoff between the two. For example, fair queueing with fair cancellation (FQ/FC) has a throughput of 102.5, compared to a standard CSMA layer's throughput of 109.2 packets per second. As the isolation layer introduces more delay mechanisms to improve fairness, it trades off throughput. However, as Figure 10 shows, a combination of fair queueing, fair cancellation, and protocol penalties allow the isolation layer to provide perfect channel fairness while only sending 13% fewer packets.

Due to the large backoffs of the TinyOS CSMA layer, improving fairness in this workload while maintaining the same packet rate actually decreases the throughput of the network. Increasing fairness causes there to be more short packets than long ones. However, as the CSMA backoff is much larger than a packet time, the sending rate is only slightly affected by packet length. A short packet transmission, including backoff, takes only slightly less time than a long one, even though the actual packet occupies the channel for a much shorter period.

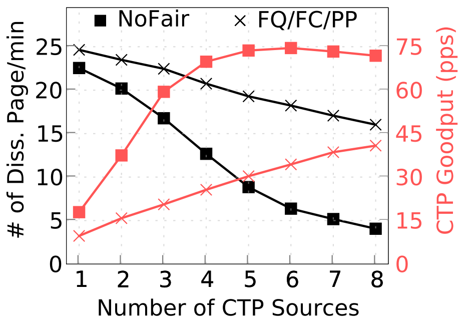
Figure 11 examines the effect of fair cancellation on fairness and network goodput. 'Fair Scheduling' denotes pure fair scheduling on top of fair queueing, where all packets are cancelled and rescheduled with a protocol penalty. 'Fair Cancel' has the same settings as 'Fair Scheduling', but replaces cancelling all packets with our fair cancellation mechanism so the minimum protocol packet is not rescheduled.

As expected, pure fair scheduling achieves the highest fairness: 1.0000. However, it comes with a 36% reduction in throughput, from 109.2 to 69.4 packets per second. Fair cancellation achieves a slightly lower fairness of 0.9998, but it mitigates the throughput drop and induces only 13% reduction in the goodput.

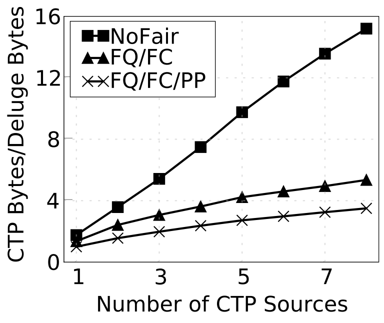
6.3 Single-hop Collection and Deluge

Next, we examine a more complex scenario where CTP and Deluge operate concurrently. CTP uses 86 byte payloads, and Deluge disseminates a binary of 250 pages, where each page is 125 packets of a 20-byte payload. We examine how changing the number of nodes sending CTP traffic affects fairness. All nodes run Deluge, but only one node has a new Deluge image, so there is only one Deluge data transmitter.

In this experiment, we measure the performance of each protocol using a protocol-specific metric. In the case of CTP, we measure packets per second, as in the previous section. For Deluge, we measure the number of pages it disseminates per minute. These represent the received goodput of each protocol, the performance that higher layers see. To quantify fairness, we measure the ratio of transmitted bytes per second between the two, as the isolation layer applies fair-



(a) Protocols' performances



(b) Ratio of transmitted bytes

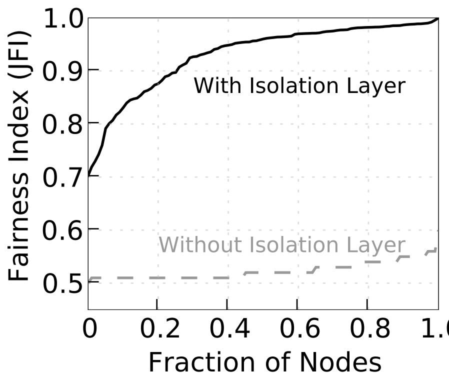
Figure 12. Application-level performances and transmission ratio between Deluge and CTP. The isolation layer improves fairness both when there is a single CTP transmitter and as the number of CTP transmitters increases.

ness based on channel occupancy, not application-level performance.

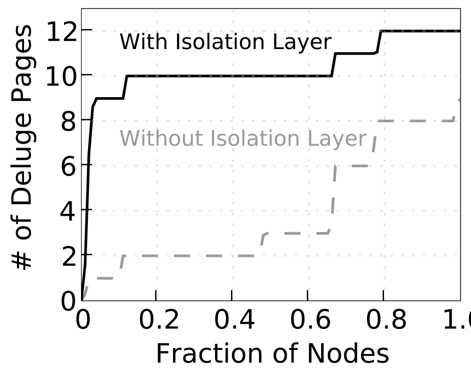
Figure 12(a) shows application-level performance. Without the isolation layer, increasing the number of CTP transmitters causes CTP to get a much larger share of the channel and begin to starve Deluge. Deluge's application-level performance drops from 23 pages/minute to 4 pages/minute, an 82% decrease. At the same time, CTP's goodput goes from 17.9pps to 71.9pps, a 318% increase.

Using the isolation layer, performance still changes as the number of CTP transmitters increases, but this change is much smaller, as the fair schemes strive to give each protocol a fair share of the channel. Deluge's performance drops by 35% and CTP's increases by 320%. This latter number is a bit troubling: it suggests that transmission fairness for CTP degrades just as quickly with the isolation layer as without. However, a closer examination of Figure 12(a) shows otherwise. Without the isolation layer, CTP reaches 72pps with only 5 transmitters, then flattens as it saturates the channel. With the isolation layer, it takes 8 CTP transmitters to have the same relative increase in channel use. As Deluge receives a more fair share of the channel, it is starved less, and CTP has more room to grow.

Figure 12(b) shows more clearly how the isolation layer improves fairness. With one transmitter, the isolation layer has a CTP-to-Deluge ratio of 1.0 (fairness of 1). Standard CSMA, in contrast, has a ratio of 1.76 (fairness of 0.92). With 8 transmitters, the isolation layer has a ratio of 3.5,



(a) CDF of the fairness between the channel occupancy times



(b) CDF of the number of delivered Deluge pages.

Figure 13. CDF of channel fairness around each node as well as Deluge dissemination speed when CTP is sending as fast as possible. The isolation layer greatly improves the fairness while providing more bandwidth to Deluge.

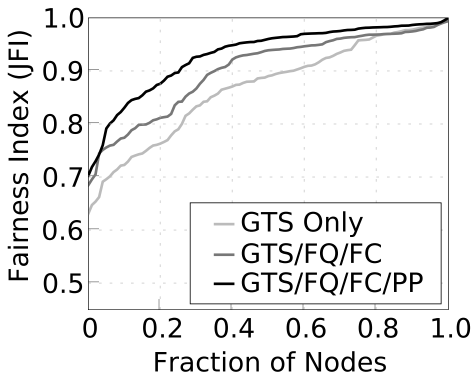
while CSMA has a ratio of 15.2: the isolation layer reduces the transmit ratio by 76%.

6.4 Multihop Networks

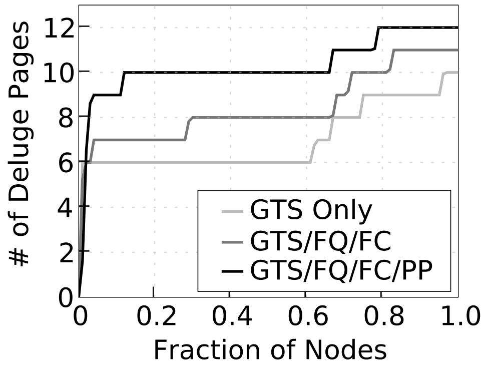
This section extends the CTP-Deluge scenario into a multihop network of 138 nodes in the Motelab testbed [38]. In these experiments, CTP uses 90 byte payloads and Deluge pages are 125 20-byte packets. CTP generates packets as fast as possible. This is a harsh scenario for Deluge; CTP spams the channel on all nodes. Thus, the focus of the scenario is how Deluge survives in this environment. Each experimental run is 40 minutes long, where the first 20 minutes are used to allow CTP to stabilize; performance is measured for the second 20 minutes.

We evaluate the effect of the isolation layer by comparing two network setups – one implements all proposed mechanisms, GTS and fairness schemes, and the other uses the default TinyOS MAC. In the case without the isolation layer, CTP has its transmit timer; in the second case, the isolation layer replaces it with GTS and allows Deluge to include grants in its request packets.

Figure 13(a) presents the cumulative distribution of the



(a) CDF of fairness index



(b) CDF of the number of delivered pages

Figure 14. Evaluation of CTP and Deluge running concurrently in a multihop network. The breakdown of isolation layer mechanisms shows how much each one contributes to the overall improvement.

fairness index for the two networks, with and without the isolation layer. To calculate fairness, we use channel occupancy time, including grant suppression durations. Actual packet durations are not counted when the suppression for the protocol is in effect, because the channel already belongs to that protocol. When nodes hear CTP's grants while they are suppressed by Deluge's grants, or vice versa, channel occupancy time is shared between the two protocols.

Without the isolation layer, 100% of the nodes have a JFI of 0.6 or lower. Employing the isolation and fairness mechanisms raises the overall fairness; the 50^{th} percentile is at 0.96. About 6% of the nodes have fairness lower than 0.8.

To quantify the isolation layer's effect on protocol performance we look at Deluge. Figure 13(b) shows the number of disseminated Deluge pages. Without the isolation layer, nodes receive at most 9 pages, with almost half of the nodes receiving only 2 pages. The non-stop CTP traffic obstructs Deluge traffic from propagating the binary from the nodes around the root to all nodes, inducing the large inconsistency in the disseminated pages. The isolation layer cures the problem – 96% of the nodes receive 9 pages or more.

Figure 14 breaks down the gains from the isolation layer and shows how much each mechanism contributes. The 'GTS Only' lines show the improvement resulting from the introduction of the shared collision avoidance mechanism. Fairness increases dramatically because CTP and Deluge are respecting each others collision avoidance request. GTS suppresses nearby nodes for the duration of the Deluge's data burst, using GTS itself can prevent starvation since Deluge has much longer grant duration than CTP.

The fairness schemes further improve fairness and performance. Fair queueing and cancellation ('GTS/FQ/FC') shift up the curve in Figure 14(a); 50% of the nodes have a JFI of 0.94 or higher, versus 0.89 or higher for the GTS-only case. Implementing protocol penalties introduces further improvement and achieves a fairness index of 0.9602 or above for 50% or the nodes. As a result, 65% of all nodes in the GTS/FQ/FC/PP scheme see fairness of at least 0.95, compared to 35% in GTS only and 0% if no isolation layer exists.

Finally, Figure 14(b) tracks the effectiveness of Deluge dissemination as we add isolation and fairness mechanisms. As a result of GTS, at least 50% of the nodes receive 6 or more Deluge pages, compared to 3 or more without isolation. Fair queueing and fair cancellation bump that to 8 or more pages, and protocol penalty raises it to 10 or more. The fairness mechanisms also improves the consistency of the disseminated Deluge pages.

The performance of CTP with Deluge, even without the isolation layer, is better than what has been reported for MintRoute with Deluge [24]. We believe the intelligent link estimation mechanisms and wait timers in CTP, and environmental differences caused these differences.

7 Related Work

The idea that a network should support concurrent operation for multiple protocols is not new in the Internet. TCP slows down its data generation when it encounters a packet loss. This property is one of the keys to Internet scalability. However, a protocol can disrupt TCP by not slowing down in response to losses: as TCP sources slow down this TCP-unfriendly protocol will saturate the network. To prevent this, non-TCP protocols are expected equip a TCP-friendly feature [16, 23]; the data generation rate must depend on the packet loss rate as TCP does. This property aims to provide a network where multiple protocols can coexist.

This congestion control feature exists at layer 4 because the narrow waist of the Internet is layer 3. That is, there exist multiple layer-4 protocols such as TCP and UDP but most of them use a single layer 3 protocol, IP. Meanwhile, in sensornets, there exist multiple layer 3 protocols while most systems use CSMA for the MAC protocol. Therefore, the isolation layer provides a shared mechanism above layer 2 that layer 3 protocols can share. Unlike congestion control, which operates along an end-to-end path and manages queue occupancy along flows, an isolation layer operates on single-hop wireless communication and manages medium access.

RTS/CTS is another widely studied mechanism that can be used for collision avoidance at the isolation layer. In sensornets, S-MAC [42] utilizes RTS/CTS exchanges for unicast transmissions. These RTS/CTS mechanisms could provide better collision avoidance performance than GTS since they prevent collisions at the current receiver, while GTS prevents collisions at the previous transmitter. In practice, however, RTS/CTS is rarely used. When the interference and communication ranges differ, RTC/CTS is no longer an effective collision avoidance mechanism [41]. In addition, in sensor networks, the control overhead associated with RTS/CTS becomes significant due to small datagram sizes.

More importantly, RTS/CTS cannot easily provide a collision-free environment for the broadcast packets that many wireless protocols depend on. For example, Deluge [17] uses broadcasts for data packets rather than unicasts since one burst of data packets can update all the neighbors of a node. In fact, many smart sensornet protocols exploit this broadcasting nature of the channel. CTP avoids congestion by making nodes overhear their parent's data packets where information on queue depth is piggybacked. Typically, these broadcast packets are sent without control packets to avoid CTS explosions as in S-MAC.

Recently, ZigZag Decoding [15] has shown that packet can be recovered from collisions using signal samples. Ideally, this can settle the issue of inter-protocol collision, because perceived collisions will become scarce. However, the need for high processing power and large memory requirements makes it hard to be applied to sensornets.

Discussing the whole subject of fairness is beyond the scope of this paper. Rather, we focus on the unique properties of protocol fairness. Demers et al. [10] showed that bit-by-bit round-robin fair queueing can be approximated by a packet-by-packet mechanism in wired network. In wired, because there exist only one transmitter per link, the fair queueing mechanism directly chooses which packet to be transmitted. Meanwhile, in wireless LAN, a distributed mechanism is needed. MACAW [5] proposed a fair allocation scheme that controls the channel access using a modified CSMA backoff algorithm. Vaidya et al. [34] extended this approach by implementing fair scheduling on multiple flows with different priorities. Protocol fairness requires a combination of intra-node fair queueing and inter-node fair scheduling.

The per-flow fairness is further extended to multihop wireless network. Typically in this case, a flow is defined to be a unique source, destination pair in the link layer [25, 28, 33]. However, clearly this does not fit for protocol fairness. That is, it is hard to divide traffic of protocols to flows of every source, destination pair, because flows are correlated with one another. Alternatively, an end-to-end source-destination pair as the definition of a flow is also considered [13, 19]. However, this is also not a feasible solution for protocol fairness because all network protocols have unique performance metrics. Therefore, we define the "flow" for protocol fairness as a single-hop term, allowing multiple sources to be the source nodes for each protocol. As discussed earlier, this caused the inconsistency of the state to be more persistent, requiring an explicit mechanism that can cure the inconsistency.

Our fair queueing scheme differs from common tag-based approaches because it does not update the start-tag when packets are not queued back-to-back. Normally, updating the start-tag serves as a time-window in which the fairness between flows is achieved. However, TinyOS prevents this because the link-layer packet queues are single-depth for each protocol. Thus, although it is possible to enforce immediate re-queueing, this cannot be applied for general protocols. For example, when CTP packets are delayed by the transmit timer, the link layer has no knowledge whether CTP has more packets. Therefore, we adjust the time window using the decay period: frequent decay results in a short-term fairness, and longer decay period improves long-term fairness.

8 Conclusions

Without isolation, interactions between protocols complicate large system design and cause unpredictable performance in deployments.

We introduced a new isolation layer in which collision avoidance information from network protocols is shared through grant-to-send. The isolation layer requires additional functionality: protocol fairness. Since standard fair queueing techniques are not suitable for protocol fairness by themselves, we augment them with two novel mechanisms, channel decay and fair cancellation.

Through a series of testbed experiments, we show that the isolation layer can reduce protocol interactions. Concurrent operation in a multihop network increases the end-to-end delivery cost of CTP and Deluge by 24% and 72%. The isolation layer reduces these cost increases by more than 60%. Furthermore, the isolation layer improves median fairness between CTP and Deluge in a multihop scenario from 0.52 to 0.96.

9 Acknowledgments

This work was supported by generous gifts from Intel Research, DoCoMo Capital, Foundation Capital, the National Science Foundation under grants #0615308 and #0846014, the King Abdullah University of Science and Technology (KAUST), Microsoft Research, scholarships from the Samsung Scholarship Foundation and a Stanford Terman Fellowship. We would like to thank those who maintain Mirage and MoteLab for providing valuable community testbeds. Finally, we would like to thank the reviewers for their comments and our shepherd, John Heidemann, for his help in improving our paper.

10 References

[1] MultihopLQI. http://www.tinyos.net/tinyos-2.x/tos/lib/net/lqi/.

[2] TEP 123: Collection Tree Protocol. http://www.tinyos.net/tinyos-2.x/doc/.

[3] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita. A line in the sand: A wireless sensor network for target detection. *Computer Networks (Elsevier)*, 46, 2004.

[4] R. Beckwith, D. Teibel, and P. Bowen. Unwired wine: Sensor networks in vineyards. In *Proceedings of IEEE Sensors*, 2004.

[5] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang. MACAW: Media access protocol for wireless lans. In *Proceedings of the international conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM)*, 1994.

[6] M. Caesar, M. Castro, E. B. Nightingale, G. O'Shea, and A. Rowstron. Virtual ring routing: network routing inspired by dhts. In *Proceedings of the international conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM)*, 2006.

[7] S. Chachulski, M. Jennings, S. Katti, and D. Katabi. Trading structure for randomness in wireless opportunistic routing. In *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM)*, 2007.

[8] J. I. Choi, M. Jain, M. A. Kazandjieva, and P. Levis. Inverting wireless collision avoidance. Technical Report SING-09-00, 2009. http://sing.stanford.edu/pubs/sing-09-00.pdf.

[9] B. Chun, P. Buonadonna, A. AuYoung, C. Ng, D. Parkes, J. Shneidman, A. Snoeren, and A. Vahdat. Mirage: A microeconomic resource allocation system for sensornet testbeds. In *Proceedings of the 2nd IEEE Workshop on Embedded Networked Sensors (EmNets)*, 2005.

[10] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. In *SIGCOMM '89: Symposium proceedings on Communications architectures & protocols*, pages 1–12, New York, NY, USA, 1989. ACM Press.

[11] P. Desnoyers, D. Ganesan, and P. Shenoy. Tsar: a two tier sensor storage architecture using interval skip graphs. In *Proceedings of the Third ACM Conference on Embedded networked sensor systems (SenSys)*, 2005.

[12] R. Fonesca, S. Ratnasamy, J. Zhao, C. T. Ee, D. Culler, S. Shenker, and I. Stoica. Beacon vector routing: Scalable point-to-point routing in wireless sensornets. In *Proceedings of the Second USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2005.

[13] V. Gambiroza, B. Sadeghi, and E. W. Knightly. End-to-end performance and fairness in multihop wireless backhaul networks. In *Proceedings of the 10th annual international conference on Mobile computing and networking (MobiCom)*, 2004.

[14] O. Gnawali, B. Greenstein, K.-Y. Jang, A. Joki, J. Paek, M. Vieira, D. Estrin, R. Govindan, and E. Kohler. The TENET architecture for tiered sensor networks. In *Proceedings of the Fourth ACM Conference on Embedded networked sensor systems (SenSys)*, 2006.

[15] S. Gollakota and D. Katabi. ZigZag decoding: combating hidden terminals in wireless networks. In *Proceedings of the 2008 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM)*, 2008.

[16] M. Handley, S. Floyd, J. Padhye, and J. Widmer. TCP friendly rate control (TFRC): Protocol specification (RFC3448), 2003.

[17] J. W. Hui and D. Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. In *Proceedings of the Second ACM Conference on Embedded networked sensor systems (SenSys)*, 2004.

[18] R. Jain, D. Chiu, and W. Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. Technical Report TR-301, DEC Research, 1984.

[19] V. Kanodia, C. Li, A. Sabharwal, B. Sadeghi, and E. Knightly. Distributed multi-hop scheduling and medium access with delay and throughput constraints. In *Proceedings of the 7th annual international conference on Mobile computing and networking (MobiCom)*, 2001.

[20] B. Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *Proceedings of International Conference on Mobile Computing and Networking (MobiCom)*, 2000.

[21] S. Katti and D. Katabi. Mixit: The network meets the wireless channel. In *Proceedings of ACM Hot Topics in Networks Workshop (Hotnets-VI)*, 2007.

[22] S. Kim, R. Fonesca, P. Dutta, A. Tavakoli, D. Culler, P. Levis, S. Shenker, and I. Stoica. Flush: A reliable bulk transport protocol for multihop wireless networks. In *Proceedings of the Fifth ACM Conference on Embedded networked sensor systems (SenSys)*, 2007.

[23] E. Kohler, M. Handley, and S. Floyd. Designing DCCP: congestion control without reliability. *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM)*, 2006.

[24] K. Langendoen, A. Baggio, and O. Visser. Murphy loves potatoes: Experiences from a pilot sensor network deployme nt in precision agriculture. In *Proceedings of the Fourteenth Int. Workshop on Parallel and Distributed Real-Time Systems (WPDRTS)*, 2006.

[25] H. Luo, J. Cheng, and S. Lu. Self-coordinating localized fair queueing in wireless ad hoc networks. *Mobile Computing, IEEE Transactions on*, 3(1):86–98, Jan-Feb 2004.

[26] Y. Mao, F. Wang, L. Qiu, , S. Lam, and J. Smith. S4: Small state and small stretch routing protocol for large wireless sensor networks. In *Proceedings of the Fourth USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2007.

[27] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi. The flooding time synchronization protocol. In *Proceedings of the 2nd international conference on Embedded networked sensor systems (SenSys)*, 2004.

[28] T. Nandagopal, T.-E. Kim, X. Gao, and V. Bharghavan. Achieving mac layer fairness in wireless packet networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking (MobiCom)*, 2000.

[29] S. Rangwala, R. Gummadi, R. Govindan, and K. Psounis. Interference-aware fair rate control in wireless sensor networks. In *Proceedings of the international conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM)*, 2006.

[30] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. GHT: a geographic hash table for data-centric storage. In *Proceedings of the first ACM international workshop on Wireless sensor networks and applications (WSNA)*, pages 78–87. ACM Press, 2002.

[31] R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler. An analysis of a large scale habitat monitoring application. In *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2004.

[32] V. Turau, C. Renner, M. Venzke, S. Waschik, C. Weyer, and M. Witt. The heathland experiment: Results and experiences. In *Proceedings of the Workshop on Real-World Wireless Sensor Networks (REALWSN)*, 2005.

[33] N. Vaidya and P. Bahl. Fair scheduling in broadcast environments. Technical Report MSR-TR-99-61, 1999.

[34] N. H. Vaidya, P. Bahl, and S. Gupta. Distributed fair scheduling in a wireless lan. In *Proceedings of the 6th annual international conference on Mobile computing and networking (MobiCom)*, 2000.

[35] M. Wachs, J. I. Choi, J. W. Lee, K. Srinivasan, Z. Chen, M. Jain, and P. Levis. Visibility: A new metric for protocol design. In *Proceedings of the Fifth ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2007.

[36] L. Wang. MNP: multihop network reprogramming service for sensor networks. In *Proceedings of the Second ACM Conference on Embedded networked sensor systems (SenSys)*, 2004.

[37] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh. Fidelity and yield in a volcano monitoring sensor network. In *Proceedings of the 7th symposium on Operating systems design and implementation (OSDI)*, 2006.

[38] G. Werner-Allen, P. Swieskowski, and M. Welsh. Motelab: a wireless sensor network testbed. In *Proceedings of the 4th international symposium on Information processing in sensor networks (IPSN)*, 2005.

[39] A. Woo and T. Tong. Tinyos mintroute collection protocol. tinyos-1.x/lib/MintRoute.

[40] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of multihop routing in sensor networks. In *Proceedings of the First ACM Conference on Embedded networked sensor systems (SenSys)*, 2003.

[41] K. Xu, M. Gerla, and S. Bae. How effective is the ieee 802.11 rts/cts handshake in ad hoc networks? In *Proceedings of IEEE GLOBECOM'02*, volume 1, pages 17–21, Nov. 2002.

[42] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient mac protocol for wireless sensor networks. In *Proceedings of the 21st International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, New York, NY, June 2002.