Inverting Wireless Collision Avoidance

Technical Report SING-09-00

Jung Il Choi, Mayank Jain, Maria A. Kazandjieva, and Philip Levis
Computer Systems Laboratory, Stanford University, Stanford, CA

Abstract

We describe grant-to-send, a novel collision avoidance algorithm for wireless mesh networks. Rather than announce packets it intends to send, a node using grant-to-send announces packets it expects to hear *others* send.

We present evidence that inverting collision avoidance in this way greatly improves wireless mesh performance. Evaluating four protocols from 802.11 meshes and 802.15.4 sensor networks, we find that grant-to-send matches or outperforms CSMA and RTS/CTS in all cases. For example, in a 4-hop UDP flow, grant-to-send can achieve 96% of the theoretical maximum throughput while maintaining a 99.9% packet delivery ratio. We also find that grant-to-send is general enough to replace protocol-specific collision avoidance mechanisms common to sensor network protocols.

Grant-to-send is simple. Incorporating it into 802.11 requires only 11 lines of driver code and no hardware changes. Furthermore, as it reuses existing 802.11 mechanisms, grant-to-send interoperates with current networks and can be incrementally deployed.

1. INTRODUCTION

Collisions are a significant design challenge for wireless mesh protocols. Traditionally, wireless MAC layers improve collision avoidance by trading off throughput. Simple schemes such as CSMA/CA introduce very little throughput overhead, but exhibit significant collisions under load. More complex schemes, such as RTS/CTS, avoid collisions better but reduce throughput when there is no contention. In practice, this tradeoff has led mesh protocol designers to choose CSMA [15, 25] and deal with the challenges collisions introduce, such as highly variable and severe packet loss [9].

More recently, network coding at the physical layer has emerged as a way to sidestep the tradeoff between collision avoidance and throughput. Approaches such as analog network coding [23] or ZigZag decoding [20] have shown that nodes can recover collided packets using simple signal processing and redundant information. The downside of these approaches is that they require new chipsets and hardware: they cannot be easily deployed in existing networks.

Can we improve collision avoidance without sacrificing throughput or requiring new hardware? Given the maturity of research in collision avoidance, a positive answer might seem unlikely. The sheer number of CSMA/CA backoff schemes [10, 13, 14, 16, 31] and RTS/CTS variations [7, 8, 11, 21, 32, 35] implies that the problem has been put to rest, suggesting the only way forward is through better signal processing, cross-layer optimizations, and network coding.

This paper presents evidence to the contrary. It shows that inverting collision avoidance's information flow can significantly reduce collisions without lowering throughput. More precisely, this paper proposes grant-to-send (GTS), a novel collision avoidance primitive. Grant-to-send "inverts" collision avoidance because a grant announces what a node expects to hear other nodes transmit. It embeds its collision avoidance information in data packets, and so uses no control packets.

When sending a packet, a node may specify an interval for which it "grants" its local channel to the recipient. The granter and all overhearing nodes remain silent for this interval. A grant allows the recipient to transmit without causing collisions at the granting node. Grants are a suppression mechanism: they are not a precondition for transmission. If all grants are zero, grant-to-send behaves identically to CSMA.

Grants do not protect the packet they are in: they avoid collisions between future packets sent by other nodes.

Long grants avoid collisions but reduce throughput through channel idleness. Short grants do not waste the channel but suffer from more collisions. So how long should a grant be? To answer this question, we derive an analytical expression of grant-to-send's behavior. The analysis shows that a grant should be as long as node expects the recipient to use the channel. Simulation and testbed results support this analysis. For example, in the case of a UDP or TCP flow, the optimal grant is a packet time, long enough for the next hop to forward. We examine how complications such as broadcast protocols, variable bit rates, and link-layer retransmissions affect this rule. In cases where the optimal grant is not known, we present simple conservative heuristics that select the optimal grant 98.8% of the time.

We evaluate grant-to-send by examining four different protocols from two network regimes, 802.11 meshes and 802.15.4 sensor networks. We measure the performance benefits over traditional approaches such as CSMA/CA and RTS/CTS. This paper makes five research contributions:

- It presents the design of grant-to-send, a novel collision avoidance mechanism. Using simulation and testbed experiments as a guide, it derives an analytical formulation of grant-to-send's behavior and performance.
- Grant-to-send matches or outperforms the throughput and delivery ratio of CSMA and RTS/CTS for all protocols in all testbed and simulation cases. For example, in a 4-hop route, grant-to-send increases UDP throughput by up to 23%, achieving 96% of the maximum possible throughput, while simultaneously reducing end-to-end losses by >95%. In 4 hop routes, grant-to-send increases TCP throughput by 48%.
- Contrary to common wisdom in existing literature [15, 17, 25], CSMA is not always superior to RTS/CTS: RTS/CTS's UDP throughput is up to 38% higher for flows longer than 3 hops.
- Self-interference can be a bottleneck to TCP performance. Reducing interference with grant-to-send and other mechanisms brings TCP's throughput on a 4-hop route to within 86% of UDP's.
- Grant-to-send is general enough to implement and replace existing collision avoidance mechanisms in sensor network protocols with no loss of performance.

Grant-to-send can reuse existing 802.11 MAC protocol mechanisms, such that it is completely interoperable with existing CSMA and RTS/CTS networks. This interoperability enables grant-to-send nodes to be incrementally deployed with an 11-line change to existing 802.11 drivers.

The next section provides background on wireless collision avoidance. Section 3 presents grant-to-send and details two implementations (802.11 and 802.15.4). Section 4 analyzes grant-to-send's behavior and provides guidance for how long grants should be. Sections 5–8 explore and evaluate grant-to-send for a variety of network protocols. Section 9 discusses limitations of the mechanism. Section 10 presents prior related work and Section 11 concludes.

2. WIRELESS COLLISIONS

A wireless collision occurs when a node receiving a packet hears one or more additional transmissions strong enough to corrupt the current reception. Collisions introduce two major problems. First, they are a waste of the wireless channel and harm throughput. Second, as the collision rate depends on the traffic load, they introduce load-based dynamics into observed link qualities, such that increasing load reduces throughput as well as end-to-end delivery.

This section describes CSMA, the standard MAC protocol used in wireless meshes, as well as the hidden terminal problem, a major cause of collisions. It revisits prior observations that a flow can exhibit significant self-collisions due to the hidden terminal, and discusses why existing mesh

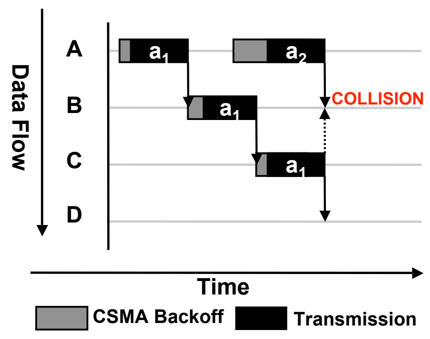


Figure 1: The hidden terminal problem in a simple flow. Node A must wait for node C to forward a_1 before transmitting a_2, or both will collide at node B.

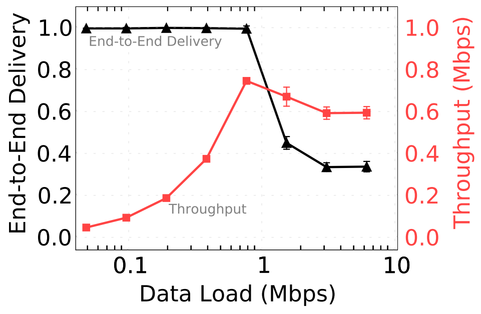


Figure 2: The effect of the hidden terminal problem within a single CSMA flow. When load passes the threshold the path can sustain, self-interference becomes significant and the delivery ratio drops accordingly. Furthermore, throughput also drops: sending more packets causes fewer packets to arrive.

protocols typically use CSMA rather than another collision avoidance scheme, RTS/CTS.

2.1 CSMA/CA and Hidden Terminals

CSMA collision avoidance is weak but inexpensive. In a CSMA/CA layer, a transmitter decides when to send by sensing the state of its channel. If its local channel is busy, it assumes the channel is busy at the receiver as well and does not transmit. CSMA/CA is simple to implement and has little overhead: nodes sense the channel and transmit.

This leads to the hidden terminal problem. The hidden terminal problem happens when two nodes that cannot hear each other (are "hidden") transmit at the same time. A third node hearing both transmissions receives neither because they collide. The hidden terminal problem is common in real-world wireless meshes and is a dominant source of packet losses especially with CSMA [18].

Figure 1 shows a basic example along a route in a multihop network, where packets self-collide due to the hidden terminal problem. Node A and node C cannot hear each other, so their transmissions collide at node B. This behavior

Bitrate	CSMA	RTS/CTS	Overhead
1 Mbps	0.79	0.76	4.0%
2 Mbps	1.44	1.35	6.6%
5.5 Mbps	3.36	2.89	14.1%
11 Mbps	5.89	4.42	25.1%

Table 1: Single-hop throughput (Mbps) on a high quality 802.11b link. RTS/CTS overhead ranges from 4-25%.

is well-known, and in the best case bounds a flow's throughput to one third of the single-hop throughput [39], as a node must wait for a packet to progress out of interference range before transmitting the next one.

Figure 2 shows this effect experimentally in a single flow in an 802.11b mesh testbed. Section 4.1 provides greater details on the experimental setup, but in summary, one node sends UDP traffic along a static 4-hop route with a fixed bitrate of 5.5Mbps. As the data rate surpasses the path's capacity, the end-to-end delivery ratio and throughput drop due to collisions. This experiment validates earlier simulation results by Li et al. [29] and Vyas et al. [36] that pushing a path beyond what it can support increases collisions and reduces performance. The plots flatten at 3.0 Mbps because link layer queuing prevents sending faster.

2.2 RTS/CTS

RTS/CTS avoids collisions through a control packet exchange before each data packet. The transmitter sends a request-to-send (RTS) to the intended receiver, describing how long it wishes to transmit for. The receiver replies with a clear-to-send (CTS) packet if it thinks its channel will be clear. Nodes around the receiver hearing the clear-to-send remain quiet, avoiding the hidden terminal problem. However, as RTS/CTS avoids control packet collisions with CSMA, CTS packets can be lost through collisions with other control packets. These losses cause RTS/CTS to have hidden terminals, albeit fewer than CSMA [34].

Pushing collisions to the control exchange improves the data delivery ratio. This improvement has a cost: a data packet requires a control packet exchange, reducing throughput. In practice, many protocol designers have found that RTS/CTS's costs outweigh its benefits [17, 25], and AP vendors suggest disabling it [2, 4].

To quantify this cost, we measure UDP throughput between two nearby 802.11b nodes. In this experimental setup, the packet drop rate and collision rate are very low. The RTS/CTS exchange is pure overhead. Table 1 shows the results. RTS/CTS overhead is 4-25%. The overhead increases with the bitrate because data packets at higher bitrates are faster, but the fixed-duration control packets sent at 1 or 2 Mbps consume a larger portion of the packet exchange time.

2.3 Summary

Intra-flow collisions from the hidden terminal problem cause UDP throughput to drop under high load, as CSMA has no knowledge about future transmissions and so cannot avoid them. RTS/CTS, in contrast, avoids more collisions, but is costly when collisions are rare. Protocols today therefore trade off between two goals:

Throughput: when collision opportunities are rare, the network should be able to use the complete link throughput as effectively as CSMA.

Collision avoidance: layer 3 protocol metrics, such as throughput and end-to-end delivery, should not decrease when collision opportunities are common.

Is it possible to get the best of both worlds? Such a protocol would avoid collisions when there is contention, yet impose no overhead when collisions are rare. One answer might be to use a hybrid approach, switching between CSMA and RTS/CTS based on circumstances. But such an approach would add further complexity to networks that are already poorly understood and hard to troubleshoot. The next section proposes an alternative approach: grant-to-send, a simple protocol that simultaneously achieves both goals as well as or better than RTS/CTS and CSMA.

3. GRANT-TO-SEND

Give every man thy ear, but few thy voice.
Hamlet, Act I, Scene iii

This section provides an intuitive and formal description of grant-to-send. Through a simple example of a flow, it illustrates how grant-to-send avoids collisions. Later sections examine more complex protocol interactions. The section concludes with details on the two implementations (802.11 and 802.15.4) we use in the rest of the paper.

3.1 Intuitive Description

Grant-to-send's primitive is simple. When a node sends a packet, it can tell nodes around it to be quiet so they do not collide with the recipient's future transmission. For unicast routing, that transmission is to forward the received packet. Given an estimate of what a recipient will do in response to a transmission, a node shares this information with neighbors to help them avoid collisions.

Grant-to-send sits on top of a CSMA/CA layer providing local node fairness and basic single-hop collision avoidance.

3.2 Formal Description

Each node i maintains a local quiet time q_i, which states the point in time when channel grants end and it may send a packet. The variable t_i refers to the current time on i's clock. When node i overhears or transmits a grant-to-send packet with a grant of length g, it extends its quiet time to $max(q_i, t_i + g)$. A packet's recipient considers g to be zero. While $q_i > t_i$, a node assumes the channel is busy. If $q_i \leq t_i$, then a node transmits using the underlying CSMA layer. A grant recipient may be unable to transmit immediately:

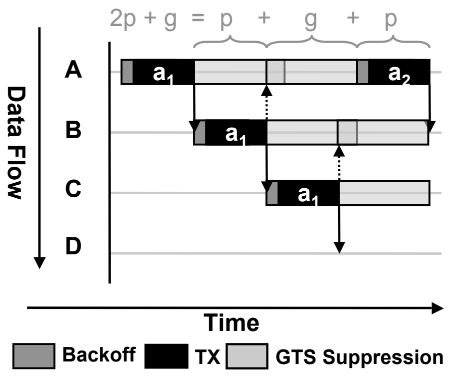


Figure 3: A packet flow using grants slightly longer than a packet time. The grants avoid intra-flow collisions by forcing nodes to wait until a forwarded packet clears the channel of their next hop.

outstanding grants to other nodes can make $q_i > t_i$.

For a transmitter, t_i is when the last bit is sent. For a receiver, t_i is when the last bit is received. As propagation time is typically below 1μs, the difference in timing between the two is irrelevant in practice.

If all packets have $g = 0$, then it is always the case that $q_i \leq t_i$: grant-to-send does not affect packet scheduling or timing and behaves identically to CSMA.

3.3 Avoiding Collisions

Figure 1 shows how the hidden terminal problem causes a packet flow to self-interfere. Figure 3 shows the same flow using grant-to-send where grant durations are slightly longer than one packet time. There are no intra-flow collisions.

The example begins with Node A granting B its channel. This grant prevents A from transmitting for a single packet time, and so B does not have to compete with A for channel access. B forwards the packet collision-free to C. This packet, in turn, grants B's channel to C. A hears the grant from B to C and extends its quiet time. C forwards the packet, granting its channel to D. As B overhears this grant, it extends its quiet time. Node A's quiet time, however, has expired, so it can now transmit to B.

Every time A transmits a packet to node B, it waits just over two packet times before transmitting again: the first from its own grant and B's transmission, the second from B's grant. Grant-to-send enforces the basic rate limiting (one third) needed in multihop flows. The last hop in a flow has a grant of zero, as it does not expect a retransmission, so grant-to-send does not force idleness on shorter flows.

This example makes many simplifying assumptions: it assumes that the interference range is the same as the transmit range, that there is a single transmitter in a unicast flow, and that a granter somehow knows the correct grant duration for the next hop. We relax these assumptions in later sections by examining how grant-to-send affects four different protocols on testbeds with two different link layers. Before we delve into these results, however, the next two subsections describe the practical considerations in the two implementations we use to evaluate grant-to-send : 802.11 and 802.15.4.

3.4 Grant-to-send in 802.11

The 802.11 duration header field states the expected length of the current packet exchange between two nodes in terms of microseconds. For example, an 802.11 data packet's duration is the length of an ACK response, while an 802.11 request-to-send packet states the expected length of the CTS-DATA-ACK exchange. Nodes can assume the channel is busy during duration intervals without sensing the channel.

The 802.11 duration field is similar to a grant, but differs in mechanism and use. As a mechanism, the duration field does not suppress a transmitter, so that a node can transmit data after an RTS. Grant-to-send, in contrast, suppresses transmitters. In terms of use, 802.11 uses the duration field to state the duration of future transmissions *between the communicating pair*, while in grant-to-send it to state the duration of future transmissions *to any destination*.

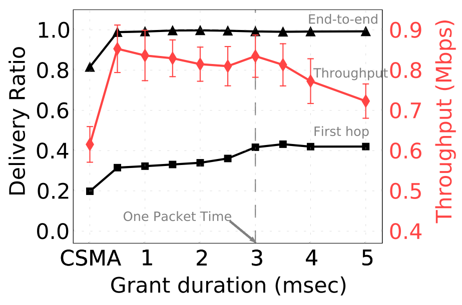
Implementing grant-to-send requires two modifications to existing 802.11 drivers. First, the driver must place grant intervals into the duration field. Second, the driver must suppress packet transmitters. We modified current MadWifi [3] and ath9k [1] drivers to provide grant-to-send on Atheros-based 802.11b/g and 802.11b/g/n cards. All of the experiments in Section 5 and Section 7 use the MadWifi driver; modifying the ath9k driver was similarly easy.

Putting grants into the duration field requires 4 lines of code to calculate a grant duration and place it in the duration field. Suppressing transmitters requires modifying the interrupt handler for transmission completion. Wireless chipsets store how long they should remain quiet based on duration fields in a variable called the network allocation vector (NAV). The modified interrupt handler writes the grant duration to the NAV register when an acknowledged transmission completes, causing a transmitter to observe the grant. Changing the interrupt handler requires 7 lines of code, for a total of 11 changed lines in each driver. Because grant-to-send uses the duration field and NAV, existing 802.11 nodes respect the grants that they hear and grant-to-send respects RTS/CTS exchanges. We defer discussing how a routing layer addresses 802.11's variable bitrates to Section 5.

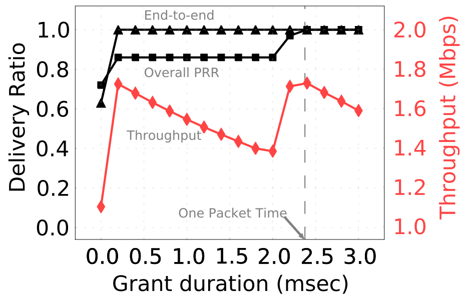
3.5 Grant-to-send in 802.15.4

802.15.4 is a low-power link layer that operates in the same 2.4GHz band as 802.11b/g. The maximum transfer unit, including header, is 127 bytes, with a bitrate of 250Kbps. The packet header has no analogue to 802.11's duration field. Supporting grant-to-send requires inserting a one-byte header between layer 2 and layer 3.

We implemented grant-to-send on the Telos [33] and MicaZ sensor nodes running TinyOS [28] version 2.0.2. Both platforms have a TI/ChipCon 2420 (CC2420). Because CC2420 hardware acknowledgments preclude overhearing packets des-



(a) A 5-node linear topology in a testbed.



(b) A 7-node linear topology in ns-2.

Figure 4: The effect of grant duration on UDP throughput, link delivery and end-to-end delivery. In the testbed, "first hop" is the link frame delivery ratio for the first hop. Both exhibit two peaks, at the minimum grant interval and at one packet time.

tined to other nodes, the implementation uses software acknowledgments. The underlying CSMA layer is the standard TinyOS 2.0.2 CSMA layer. All experiments use the Intel Mirage testbed [19], which has MicaZ nodes. MicaZ nodes have a faster processor (7.3MHz vs 1MHz) than Telos and can sustain high I/O rates to the radio [28].

Implementing grant-to-send requires adding a single-byte header that specifies a grant duration in milliseconds. One byte header limits grant durations to 255ms. Grant-to-send itself is approximately 50 lines of TinyOS code. It requires nine bytes of state for a timer that marks when q_i expires.

3.6 Summary

Grant-to-send reverses the traditional information flow in collision avoidance. When a node sends a data packet, it can help others avoid collisions by granting its local channel to the recipient. As grant-to-send requires no control packets, it does not impose the overhead typically observed with RTS/CTS. Using a combination of testbed experiments and simulation, the next section examines the effect of grant durations on a UDP flow and derives an analytical formulation of the effect of grant durations on throughput.

4. GRANT DURATION

Longer grants improve collision avoidance but shorter grants have higher channel utilization. This raises the question: how long is *long enough* for a grant? To answer this question, we examine the most basic case of a multihop protocol, a single UDP flow with a static bitrate.

4.1 Small Testbed

To experimentally determine the best grant duration, we deploy a seven-node 802.11b/g testbed in our building's hallways. All 802.11 nodes in this paper are PCEngines ALIX.3C boards, with an AMD Geode CPU, 128MB RAM and a 2GB CF card. They have Compex WLM54GP23 802.11b/g cards with Atheros chipsets and dual antennas. The nodes run OpenWRT, a build of Linux 2.6.25-17. The 802.11 stack has a maximum link retransmission count of 4.

In this experiment, each node uses a fixed 5.5Mbps bitrate and runs on channel 1, which was cleared for the purposes of the experiment. A few wireless APs in nearby buildings within reception range remain on channel 1. We use 5.5 Mbps because it is the highest bitrate that our logging facility permits. The routing layer uses the standard Roofnet distribution of Srcr [5], implemented using Click [27]. The source has a reasonably stable but not static route to the gateway that is typically 4 hops. We run iperf for 90 seconds with a payload of 1470 bytes to measure UDP's performance. Because the last hop does not expect the destination to transmit wirelessly, it always specifies a grant of zero.

We use three metrics to evaluate the effect of grant-to-send. Throughput is the rate at which UDP delivers data. End-to-end is the percentage of transport-layer segments that arrived at the destination. This metric is important as many higher layer protocols, such as TCP, respond to end-to-end loss. First hop is percentage of link-layer frames that arrived successfully at the first hop. In this case retransmissions via ARQ are considered separate frames and end-to-end delivery is higher than first hop due to ARQ. The first-hop metric is important because of its effect on mesh routing protocols; packet losses imply self-interference.

Figure 4(a) shows the results. CSMA sustains a throughput of 0.62 Mbps while having an end-to-end delivery ratio of 81%. A small grant of 500 μs increases throughput to 0.85 Mbps, a 37% improvement. End-to-end delivery ratio increases 98.8%, reducing losses by 94%.

Increasing the grant past 500 μs decreases throughput until it reaches 3 ms, at which point throughput jumps to 0.84Mbps, well within the error of the throughput at 500 μs. Furthermore, a grant of 3 ms has an end-to-end delivery ratio of $>$99%, a $>$95% reduction in losses over CSMA and a 31% reduction over a 500 μs grant.

3 ms is the expected transmit time of a 5.5 Mbps packet, including CSMA backoff. A small grant greatly improves throughput and end-to-end delivery; a grant of a single packet time has the same throughput but even better end-to-end delivery. Intermediate values, while better than CSMA, are inferior to these two grant durations.

Unlike throughput, first-hop delivery increases steadily with

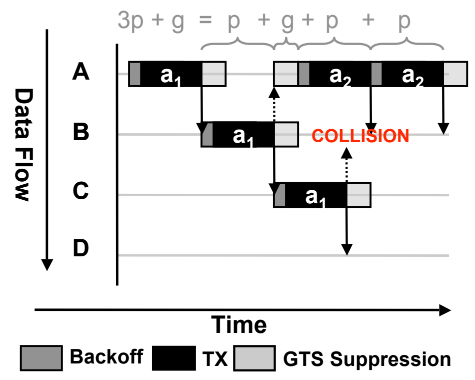


Figure 5: A UDP flow using a short grant. A short grant gives a forwarder CSMA priority over the first transmitter, avoiding simultaneous transmissions. However, the source still encounters the hidden terminal problem at the first hop, and so has to transmit each packet twice.

larger grant durations. While CSMA has a first-hop delivery ratio of 20%, a small grant boosts this to 32%, and a full packet grant boosts it further to 42%. Increasing the grant past 3 ms does not improve the chance that the first hop will successfully receive a link-layer frame.

These results are from a single route in a somewhat controlled wireless network. Blindly generalizing them to all networks is dangerous. Instead, we turn to the repeatability, control and visibility of simulation to understand the cause of these peaks, and to see if they are fundamental or an artifact of the experimental setup. Simulation removes uncontrollable variables, such as external 2.4GHz interference.

4.2 Simulation

We simulate a 7-node chain topology in ns-2 with the 802.11Ext MAC layer at 6Mbps and a link MTU of 1500 bytes. The physical layer model consists of logical links, where nodes can communicate perfectly with the two adjacent nodes. The interference range is the same as the transmit range: packet reception fails only when two adjacent nodes transmit at the same time. In these experiments, we look at the overall PRR of all link layer segments, because in ns2's simplification of the wireless channel, where the increased forwarding load of grant-to-send causes all collisions to occur on the first link, unlike in real networks.

Figure 4(b) shows UDP's throughput and end-to-end network delivery ratio as a function of grant duration. While the values are different than in Figure 4(a) and the peaks are steeper, the simulation shows the same trends. The second peak is at 2.4 ms rather than 3ms because of the differences in preamble length and bitrate.

At the minimum grant, 200μs, both throughput and delivery are significantly higher than CSMA. From Figure 3, we know that a grant longer than a packet time avoids intra-flow collisions. Why does a short grant help?

Examining the simulation logs, we find that a small grant's benefit comes from delaying a transmitter slightly, so the forwarder is likely to win CSMA. Essentially, a small grant gives CSMA scheduling priority to the forwarding node. Figure 5 shows this behavior. This priority avoids the case where two nodes both enter carrier sense at the same time, sense an idle channel, and transmit.

Small grants do not avoid hidden terminal collisions. Packet a_2 still collides in Figure 5, as node A believes it can transmit at the same time as node C. Furthermore, grants longer than the CSMA backoff window harm throughput. A's first transmission of a_2 starts after B's grant concludes, but collides at B. A retransmits a_2, and this second transmission succeeds. A's first transmission will always fail as long as B's grant is shorter than a packet time, and it delays when A's second, successful, transmission occurs.

This explains why throughput declines between the two peaks. It also explains why overall PRR increases with a small grant, but does not reach 100% until 2.4 ms. The significant increase at 2.2 ms is because the actual packet time for 6 Mbps is between 2.2 and 2.4 ms. CSMA backoff means that a grant very slightly below a packet duration can avoid most, though not all, collisions.

4.3 Analysis

The simulation results allow us to describe grant-to-send analytically. In this analysis, p is the length of a packet, g is the grant duration in terms of p, and B is the maximum single-hop throughput of the link layer. In real networks g is in absolute time units such as microseconds; here it is in terms of p for simplicity.

When $g = 0$ (CSMA, Figure 1), A and B contend for the channel. Prior work by Vyas et al. shows that such a flow can sustain a throughput of $\frac{B}{3+k}$, where k is in the range of 0.3 to 3, is typically well above 1, and depends on load as well as physical parameters [36].

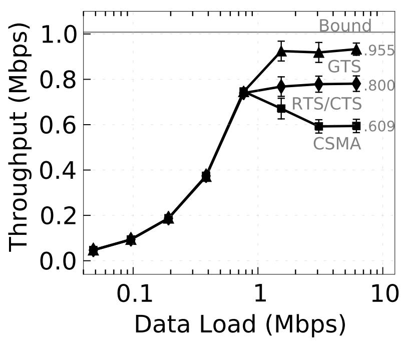
When $g < p$ (Figure 5), A and B do not contend, but A's first transmission is lost due to hidden terminal C. A's inter-packet interval is $3p + g$: B's forwarding, B's grant, A's first transmission, and A's retransmission. The flow can sustain a throughput of $\frac{B}{3+\frac{g}{p}}$.

When $g \geq p$ (Figure 3), A transmits collision-free. A's inter-packet interval is $2p + g$: B's forwarding, B's grant, and A's transmission. The flow throughput is $\frac{B}{2+\frac{g}{p}}$.

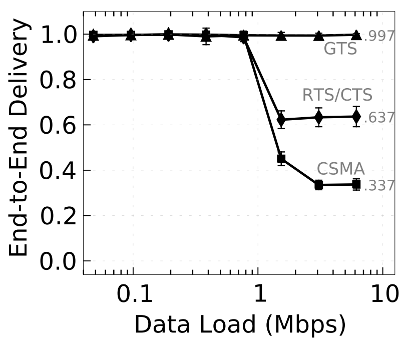
Therefore, the throughput T of a grant g is

$$T(g) = \begin{cases} \frac{B}{3+k} & \text{if } g = 0 \\ \frac{B}{3+\frac{g}{p}} & \text{if } g < p \\ \frac{B}{2+\frac{g}{p}} & \text{if } g \geq p \end{cases}$$

From this analysis, the highest throughput is when $g = p$. This falls under case 3, such that the throughput is $\frac{B}{3}$, which is the maximum achievable throughput in flows longer than 2 hops [39]. As $g \to 0$, case 2 approaches but does not reach $\frac{B}{3}$. In real networks, grants smaller than the CSMA backoff



(a) Throughput. The bound is 1/3 of the throughput of the bottleneck link.



(b) End-to-end delivery.

Figure 6: UDP performance on the small testbed with varying load (log scale). Each data point is averaged over 21 runs. Error bars show the standard deviation.

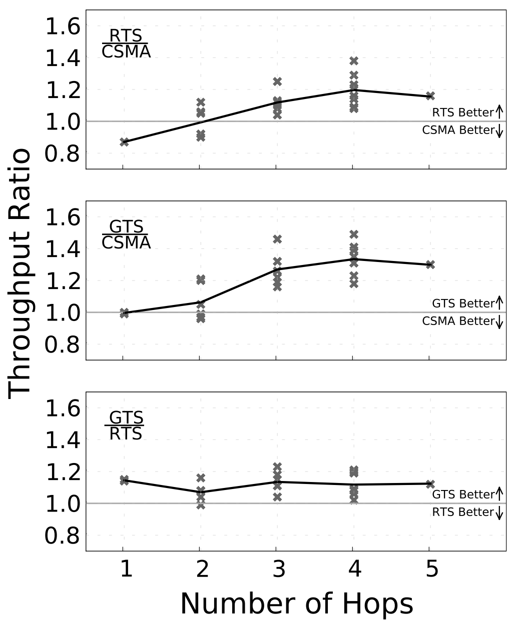


Figure 7: Comparison of UDP throughput for node pairs on a 24-node testbed; the solid line traces the mean. Shorter routes favor CSMA and longer ones favor RTS/CTS. Grant-to-send maintains efficiency of CSMA in shorter routes, and outperforms RTS/CTS in longer routes, showing up to 49% gain over CSMA and 23% over RTS/CTS.

interval cause B to sometimes lose CSMA, so as $g \to 0$ the network starts to behave as a mix of cases 1 and 2.

Figure 4(b) supports this analysis. In this simulation, $p =$ 2.4ms. The two peaks have a throughput of 1.72 Mbps, which is approximately one third of the single hop throughput of 5.5 Mbps. For example, for a 2ms grant (the worst grant in the plot), $3 + \frac{g}{p} = \frac{18}{6} + \frac{5}{6} = \frac{23}{6}$. Therefore $T(2ms) = \frac{5.5Mbps \cdot 6}{23}$, or 1.4Mbps. The results in Figure 4(a) are in a real network with other contenders. They violate assumptions in the analysis so do not match the equation, but show the same trend.

4.4 Summary

Through a controlled experiment, simulation, and analysis, we find that the optimal grant duration in a UDP flow is a packet time. The intuition is that nodes that receive one packet expect to forward one packet. More generally, a grant should announce how long it expects a recipient to use the channel in response to the received packet. We defer an evaluation of this generalization to Section 8, where we examine Deluge, a protocol that uses bursts of broadcasts.

5. UDP IN A LARGE TESTBED

The prior section makes four simplifying assumptions: the transmit and interference range are equal, a node knows the next hop's transmission duration, routes are static, and there are few contenders. Evaluating UDP in a large testbed allows us to validate these controlled results in an uncontrolled environment. It also provides a basis for understanding more complex protocols in later sections.

These experiments use a 24-node testbed in our building. Unlike the controlled experiments in Section 4, the testbed shares the 802.11 spectrum with the building's heavily used wireless network. The 24 nodes are spread across 6 floors. We omit pictures of the physical network topology for anonymity, but report aspects of the logical topology.

5.1 CSMA, RTS/CTS, and Grant-to-send

Figure 6 shows the throughput and end-to-end delivery ratio of CSMA, RTS/CTS, and grant-to-send (GTS) between a single node pair as the offered load increases. The route was typically 4 hops. Nodes used a 5.5Mbps fixed bitrate.

As in Figure 2, CSMA's throughput and delivery degrade after it is pushed past approximately 800Kbps, dropping to 608Kbps and a delivery ratio of 34%. RTS/CTS flattens at 800Kbps, maintaining a delivery ratio of 64%. Grant-to-send is able to sustain a throughput of 956Kbps and a delivery ratio of 99.7%. Under load and in the presence of other interfering transmitters, grant-to-send's throughput is

Hops	# Pairs	GTS	CSMA	RTS/CTS
1	2	2.60	2.60 (0%)	2.27 (15%)
2	6	1.19	1.12 (6%)	1.11 (7%)
3	6	0.90	0.70 (28%)	0.79 (13%)
4	8	0.77	0.57 (35%)	0.69 (15%)
5	1	0.65	0.50 (30%)	0.58 (12%)
Total	23	1.06	0.92 (15%)	0.96 (10%)

Table 2: UDP throughput (Mbps) for 23 node pairs averaged for each hop count. Percentages show grant-to-send's improvement. Grant-to-send has higher or equal throughput in all cases.

20% higher and it reduces end-to-end losses by over 99%.

The bound line Figure 6(a) represents the theoretical throughput bound of the topology. We compute this by starting with the link-layer throughput measurement in Table 1 of 3.36 Mbps. As the route is longer than 2 hops, the bound is one third the link throughput [39]. Furthermore, packet losses reduce throughput: the bottleneck link has a PRR of 90%, cutting the throughput by one tenth, leading to an overall throughput of 1.01 Mbps. Grant-to-send achieves 965Kbps, 96% of this upper bound.

5.2 Effect of Hop Count

In this section, we measure the performance of grant-to-send using the full testbed. We pick one source that has a reasonable distribution of hop counts to other nodes, and measured the throughput of all 23 possible pairs. Each measurement is a 1 minute run of iperf. We measure each pair using CSMA, RTS/CTS, and grant-to-send, repeating this ten times. Each data point is the average of 10 runs. This comprises over ten hours of measurements.

Figure 7 shows the results as throughput ratios between collision avoidance schemes, grouped by hop count. Table 2 presents the raw results, averaged for each hop count. As these results are taken on a dynamic routing topology decided by Srcr, the hop count between two nodes can change over time. We statically sample the hop count between each pair before the experiment. Therefore, the number of hops shown is just for reference, and might not reflect the actual number of hops taken.

The top plot in Figure 7 shows CSMA has higher throughput than RTS/CTS in and 1- and 2-hop routes. This follows from the measurements in Table 1; when collisions are rare, RTS/CTS imposes unnecessary overhead. At 3 hops, the hidden terminal comes into play and RTS/CTS sustains a higher throughput than CSMA – up to 38% – on a 5.5 Mbps link. This contradicts a common belief in the research literature [15, 17, 25] that CSMA is superior to RTS/CTS in wireless meshes; that the primary experimental study, Srcr [15], only examined TCP on routes of up to 3 hops.

The middle plot shows that grant-to-send matches the throughput of CSMA for one-hop networks and is superior – up to 49% – on longer routes. The bottom plot shows that grant-to-send outperforms RTS/CTS for all hop counts.

5.3 Variable Bit Rate

So far, all nodes used the same bitrate, 5.5 Mbps making the forwarder's packet duration predictable. In practice, however, mesh routing layers often use bitrate adaptation to minimize transmit time.

Our modified Srcr implementation includes a grant adaptation scheme. A node maintains a hashtable keyed by the (next hop, destination) pair. The hashtable stores the last bitrate the node heard that entry use. Destination is necessary because it can change what link and bitrate the next hop uses. When selecting a grant duration, grant-to-send assumes the last bitrate heard. If there is no entry, grant-to-send assumes the fastest bitrate: the analysis in Section 4.3 shows that underestimating is better than greatly overestimating.

This approach assumes that bitrate selections are stable. To check the validity of this assumption, we run Srcr with variable bit rates on top of grant-to-send and log whenever a grant does not match the subsequent transmission. This experiment is unfavorable for bitrate stability because nodes go through their startup settling period of finding the best rate. Across the 23 node pairs, 1.2% grants did not match. Of these, 66% were underestimates. Grant-to-send overestimates grants on 0.4% of transmissions. We defer a discussion of throughput with bitrate selection to Section 7.4.

5.4 Summary

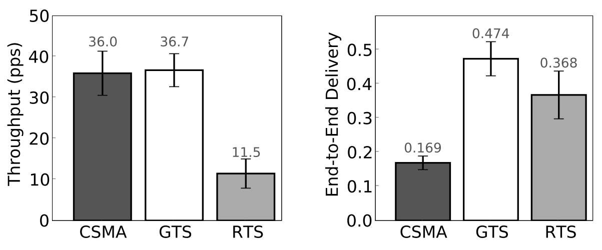
This section examined how grant-to-send affects a UDP flow in an uncontrolled wireless environment. Grant-to-send's throughput is up to 49% higher than CSMA, and up to 23% higher than RTS/CTS.

A single UDP flow, however, is a very simple workload. The next sections examine three wireless protocols of increasing complexity: sensor network collection trees, which consist of multiple unreliable flows converging on a single node, TCP, which has two opposing packet flows, and sensor network data dissemination, which uses bursts of broadcasts to reliably deliver new binaries across a network.

6. COLLECTION TREES

In this section, we evaluate whether grant-to-send can replace a protocol-specific collision avoidance mechanism without loss in performance. In sensor networks, collection protocols construct minimum-cost trees to data sinks. These trees are unidirectional: they do not maintain reverse sink-to-source routes. Like UDP, collection protocols are connectionless and unreliable. Since multiple nodes can report data at once, collection protocols can be viewed as multiple UDP flows converging at a gateway.

Under this abstraction, however, collection protocols use very different mechanisms than 802.11 meshes in order to cope with environmental dynamics and energy constraints. For example, they use distance vector, rather than link-state



(a) Throughput. (b) End-to-end delivery ratio.

Figure 8: Event-driven collection using CTP on CSMA, GTS, and RTS/CTS. GTS delivers throughput comparable to CSMA's, while increasing end-to-end delivery.

algorithms. More generally, the importance of energy efficiency causes most layer 3 sensornet protocols to have custom built-in collision avoidance mechanisms.

6.1 CTP

CTP is the standard collection protocol in TinyOS 2.1 [6]. It uses a transmit timer to avoid self-interference along a route. When a CTP node transmits a data packet, it waits approximately 2 packet times before sending another packet or retransmitting. This timer is local to a node. It does not prevent other nodes from immediately sending to the same next hop and colliding.

Grant-to-send can provide a superior mechanism to CTP's timer: it can avoid collisions with all nearby nodes, not just a single transmitter. We modify CTP by removing its transmit timer and instead having it send all data packets with a grant of one packet time (10ms). As with UDP, data packets to a data sink (the last hop) carry a grant of zero.

6.2 Evaluation

When network traffic is light and there are few collisions, grant-to-send imposes no cost and provides no benefit. On the opposite extreme, if all nodes send data as quickly as possible, the lack of congestion control in CTP causes queues to overflow and the network devolves to single-hop neighbors contending at a sink with grants of zero. Therefore, we evaluate grant-to-send in an event-driven collection scenario, where a subset of nodes detect an event and stream a large data report. Volcanic seismic monitoring is one example of an application that has such a workload [38]. The assumption in this scenario is that the low-power sensor network wakes up for a burst of activity: collision avoidance is not a major concern when the network is asleep.

We run CTP on 64 nodes in the Mirage testbed [19]. A radio packet simulates a triggering event, and 14 nodes that hear the packet begin streaming data to a sink. The source nodes are fixed throughout the experiments to minimize variations between runs. Source nodes stream packets for 15 minutes. Each result is the average of five runs.

Figure 8(a) shows throughput under CSMA, RTS/CTS and grant-to-send. Grant-to-send's throughput is 220% higher than RTS/CTS and within experimental error of CSMA. Grants can replace an existing layer 3 collision avoidance mechanism with no appreciable effect on throughput.

Hops	# Pairs	GTS	CSMA	RTS/CTS
1	2	2.25	2.21 (2%)	1.91 (18%)
2	6	0.77	0.72 (7%)	0.61 (26%)
3	6	0.51	0.44 (16%)	0.24 (113%)
4	8	0.46	0.31 (48%)	0.18 (156%)
5	1	0.50	0.39 (28%)	0.28 (79%)
Total	23	0.71	0.62 (15%)	0.46 (53%)

Table 3: TCP throughput (Mbps) for 23 node pairs averaged for each hop count on the large testbed. Percentages show grant-to-send's improvement.

Figure 8(b) shows CTP's end-to-end delivery ratio. Grant-to-send's delivery ratio is 180% higher than CSMA's, and 29% higher than RTS/CTS. This improvement comes from the natural rate-limiting that grant-to-send provides. For example, while each CSMA node sends at most once every three packet times, the aggregation of this load towards the root causes queue drops. While grant-to-send does not completely solve this problem, it limits load in terms of broadcast regions, rather than individual nodes. Therefore, a group of nodes close to one another impose an aggregate load of at most one transmission every three packet times.

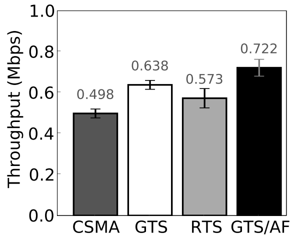
An astute reader will notice that the results in Figure 8 demonstrate a major difference between 802.11 and TinyOS's MAC layer. The throughput benefit of grant-to-send is much smaller in collection than in UDP. This can be explained by two observations. First, the CSMA and RTS/CTS results already include CTP's transmit timer, providing better collision avoidance than 802.11's UDP. Second, the default CSMA backoff interval in the TinyOS MAC layer dominates a packet time. An actual frame transmission takes approximately 1 ms, while backoff is 1-9 ms. Correspondingly, while routes do self-interfere, the interference is much less pronounced than with 802.11.

6.3 Summary

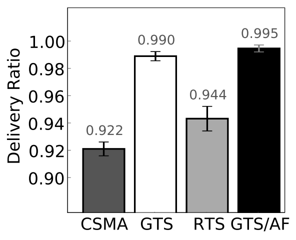
Grant-to-send outperforms both CSMA and RTS/CTS for event-driven workloads in sensor network collection trees. It provides the best of both worlds: the throughput of CSMA and the delivery ratio of RTS/CTS. It replaces a layer 3 mechanism with a more general layer 2 one. Section 8 examines whether the same replacement approach can be applied to a broadcast-based protocol. First, however, the next section examines a slightly more complex flow case: TCP.

7. TCP

Examining TCP, we find that, in contrast to UDP and collection trees, grant-to-send alone offers a small throughput improvement. We identify that collisions between TCP's two flow directions are a significant cause of packet loss. Re-

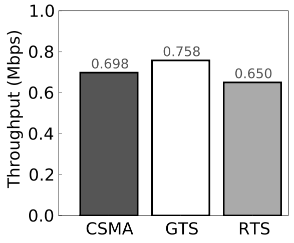


(a) Throughput.

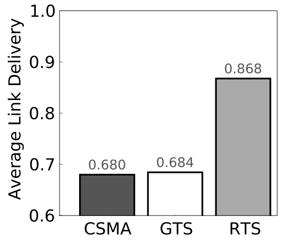


(b) Data end-to-end delivery.

Figure 9: Small testbed TCP performance. Error bars show standard deviation. Grant-to-send significantly improves end-to-end delivery but throughput increase are modest. GTS/AF is GTS with 50% ack filtering.



(a) Throughput



(b) Avg. Link Delivery Ratio

Figure 10: Simulation results on TCP performance. Grant-to-send does not improve throughput. Grant-to-send's lack of link delivery ratio improvement reveals the presence of data-ack collisions.

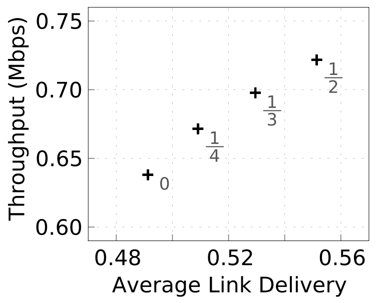


Figure 11: Throughput and average link delivery for different ack filtering rates (fractions shown). Reducing acknowledgments improves TCP performance.

ducing such collisions brings TCP's throughput with grant-to-send up to within 86% of UDP's.

Table 3 shows aggregate results for TCP between the 23 pairs described in Section 5.2. Grant-to-send matches or exceeds the throughput of CSMA and RTS/CTS. CSMA outperforms RTS/CTS for all hop counts, and grant-to-send outperforms CSMA. 3-hop throughput, however, is well below one third of the single-hop throughput. While grant-to-send is improving throughout by avoiding some collisions, others must remain.

7.1 Small Testbed

Figure 9 shows TCP's performance under the same small testbed setup as the single-flow UDP experiments in Section 4.1. External interference is very limited. Grants are 3ms and the last hop uses a grant of zero. The TCP stack is the OpenWRT default, Reno. TCP throughput with grant-to-send is 28% higher than CSMA, and 11% higher than RTS/CTS. Examining the delivery ratio of data packets, grant-to-send reduces their drop rate by ≈85%.

If TCP's throughput is constrained by its congestion window, an 85% reduction in packet losses would lead to a much larger increase in throughput than 28%. 638 Kbps is approximately 60% of the topology's maximum throughput. These results imply that end-to-end losses are not the principal cause of TCP's poor performance. As we did with UDP, we turn to simulation to understand TCP's performance.

7.2 Simulation

We run TCP New Reno in the same ns-2 simulation as Section 5. Figure 10(a) shows similar results to the testbed experiment: grant-to-send's throughput is slightly higher than CSMA's. As in the testbed, grant-to-send in ns2 also has a very high end-to-end delivery ratio (not shown). Grant-to-send reduces end-to-end losses without observing a commensurate higher throughput.

Figure 10(b) shows the limiting factor: the link-layer delivery ratio. Unlike UDP, which observed a >100% improvement in link-layer delivery with grant-to-send (Figure 4(a), 20 to 41%), TCP does not see a significant improvement. This indicates a possible cause of the poor performance. Link losses lead to retransmissions, which cause packets to occupy the channel longer. The fact that a reduction in end-to-end losses does not improve throughput means the medium is already being completely utilized. The bottleneck is not the congestion window, but the actual link throughput.

If poor link performance is the cause, improving the link delivery ratio will increase TCP's throughput. Since links in the simulation only drop packets due to collisions and a one packet time grant greatly avoids intra-flow collisions, collisions must be *between* flows. There are only two flows in this experiment – data and acknowledgments. Collisions between data and acks could be causing TCP to retransmit more at the link layer, reducing the achievable throughput.

7.3 Sending Fewer Acks

RFC 2581 recommends TCP sends at least one ack packet for every two data packets in order to prevent packet bursts. Other transport protocols, such as DCCP [26], have shown that applying congestion control on acknowledgments does not preclude good performance.

To test our hypothesis that TCP's poor performance is in part due to data/ack collisions, we modified our experimental setup to send fewer acknowledgments. To reduce the ack rate without changing the TCP stack, we changed Srcr so

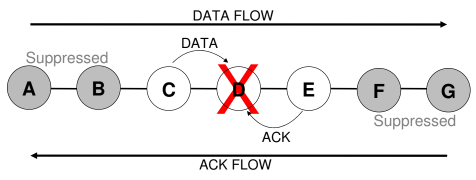


Figure 12: Although grant-to-send successfully avoids intra-flow collisions, it is vulnerable to inter-flow collisions, shown at node D.

that mesh endpoints and gateways filter acknowledgments. Reducing the acknowledgment rate only matters when they are causing collisions, that is, when TCP has a large transmit window. When there is a small window, there are few packets in flight, and lost acknowledgments can more easily lead to timeouts. We set a window threshold, above which Srcr would drop every nth acknowledgment. This filtering clearly would not work in full systems: among other limitations, it assumes that data flows in only one direction. While not a reasonable solution to improve TCP's performance, it tests our hypothesis that data/ack collisions are at fault.

Figure 11 shows TCP's performance using grant-to-send as the acknowledgment filtering rate increases. A combination of grant-to-send and 50% ack filtering leads to a throughput of 722Kbps, a 45% increase over CSMA's 498 Kbps, and 86% of the UDP throughput reported in Section 4.1 while providing TCP's reliability and in-order delivery. The link layer delivery is boosted from 49% to 55%: this represents a 25% improvement in the bottleneck link. Figure 11 validates that grant-to-send's bottleneck was throughput due to the link delivery ratio. Ack filtering decreases throughput with CSMA and RTS/CTS since the clear channel time simply causes them to transmit data packets faster and have more data-data collisions.

Figure 12 illustrates why grant-to-send does not help with data-ack collisions. When C forwards data, both A and B are suppressed by B's prior grant to C. This avoids data-data collisions. Similarly for acknowledgments, F's prior grant to E avoids ack-ack collisions from F and G. However, C and E cannot resolve collisions using grants. This leads to a data-ack collision. Furthermore, this is not an edge case: data and ack packets collide on every hop of the route. TCP's traffic pattern is the worst case scenario for the hidden terminal problem – two flows with highly correlated traffic (data triggers acks) in opposite directions. Quantifying the benefit of ack filtering in the larger testbed is a point of future work.

7.4 Discrepancy

To followers of mesh TCP performance, the results in this section might seem surprising: TCP can, over a 4-hop path, sustain high throughput. This result disagrees with prior studies of Srcr performance [15, 25]. We believe that the cause for Srcr's poor throughput in these studies is its dynamic bitrate selection algorithm. Many links devolve to 1Mbps, choking the maximum throughput.

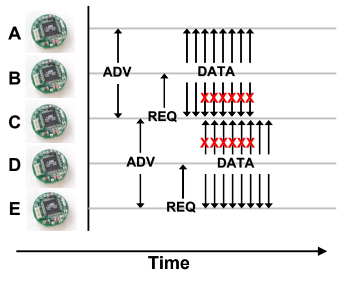
The results in this section use a fixed 5.5Mbps bitrate.Nodes running Srcr on the larger testbed exhibited excellent results, on par with those in Figure 9(a). With bit rate selection, however, Srcr's performance was drastically poorer. For example, UDP running with dynamic bitrate selection reduced throughput by 80% compared to the results in Figure 7. TCP was similarly poor. More recent results, showing good OLSR performance on top of a static bitrate, add further evidence that bitrate selection algorithm causes poor performance [30]. Clearly, future experimentation is needed.

7.5 Summary

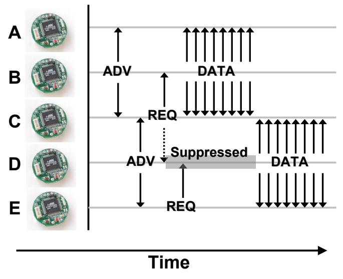
Packet loss is a well known problem for TCP's in wireless networks. To the best of our knowledge, however, there are no experimental studies of *why* packets are lost or what a network could do to avoid them. Applying grant-to-send to TCP provided insight into this typically poor performance: data/ack collisions. Applying grant-to-send to reduce intra-flow collisions and aggregating acks to reduce inter-flow collisions can improve TCP's throughput over CSMA by up to 45%, within 86% of UDP's. The technique we use to show the cost of data/ack collisions is ill-advised for use in real networks, but its effectiveness suggests that more intelligent approaches are worth investigating.

8. DISSEMINATION

In sensor networks, a dissemination protocol reliably a piece of data to every node. This section examines Deluge [22], a dissemination protocol for data items much larger



(a) Standard Deluge.



(b) V-Deluge and GTS-Deluge.

Figure 13: A simple example of Deluge with and without collision avoidance, running on a chain topology.

than node RAM. Deluge's typical use is distributing new node binaries into a network for in-situ reprogramming.

Deluge uses a three-way handshake to deliver a burst of data broadcasts. It is an evolution of wireless dissemination protocols such as SPIN [12]. Like CTP, Deluge uses protocol-specific suppression mechanisms at layer 3 to avoid data collisions and deliver data faster. This section examines whether grants are general enough to implement these mechanisms and achieve equivalent or superior performance.

8.1 Deluge and V-Deluge

Deluge periodically broadcasts what binary version a node has. When a node hears a newer version, it sends a unicast request for the data to the advertising node. A node receiving a request broadcasts a flurry of packets in response. When a node has part of the new image, it advertises quickly, such that neighbors with the older version request it.

Prior studies of the original Deluge protocol showed that sending bursts of data packets can cause long periods of high collision, as shown in Figure 13(a). A Deluge variant called Visual Deluge (V-Deluge) solves this problem by having requests suppress other traffic, as shown in Figure 13(b). These suppressions reduce dissemination time by 31% while simultaneously sending 46% fewer packets [37].

This suppression is a grant: when a node sends a request, it grants for how long it expects the flurry of data packets to take. Request packets already state how many data packets they need, so computing a grant duration is trivial. Data packets and advertisements have grants of zero.

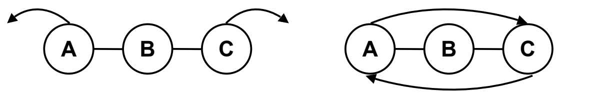
This suppression through requests is not perfect. In Figure 13(b), for example, D cannot respond immediately to E's request, so E's grant will not cover all of D's data transmissions. However, if D is delayed long enough E will re-request and grant again.

8.2 GTS-Deluge

We modify Deluge in TinyOS 2.0.2 to grant as described above: the change involved adding a single parameter to the 3 calls that send a packet (advertisement, request, and data). We call this version of Deluge GTS-Deluge.

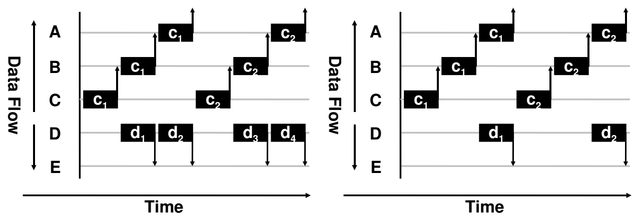
We obtained the V-Deluge source code from its authors. We compare V-Deluge and GTS-Deluge with the same methodology and testbed (Mirage) used to compare standard Deluge and V-Deluge [37]. Each experimental run involves injecting a new binary at one corner of the network. We measure two metrics, dissemination time and packet transmissions, as the average of ten runs. V-Deluge and GTS-Deluge perform equivalently; their latency and transmission counts are within 3%, a variation well within experimental error.

V-Deluge's mechanism can be easily expressed as a grant in GTS-Deluge, and the two have indistinguishable performance. GTS-Deluge requires 3 function call changes to Deluge. In contrast, V-Deluge modifies 247 lines of code, five times the entire implementation of grant-to-send! These results, together with the results from Section 6 provide evidence that grant-to-send is more broadly applicable than unicast flows, and is general enough to describe existing higher-layer collision avoidance mechanisms.



(a) A and C are hidden and grant other nodes. (b) A and C are not hidden and grant each other.

Figure 14: A and C take turns transmitting overlapping grants such that $tq_B > t_B$ casing starvation at B.



(a) CSMA (b) Grant-to-send

Figure 15: An edge case where grant-to-send exacerbates the exposed terminal problem and shows 33% throughput drop over ideal CSMA scheduling. C and D send packets in opposite directions.

9. BEST EFFORT

The previous sections evaluated grant-to-send's benefit and found it outperforms CSMA and RTS/CTS in a wide variety of protocols and scenarios, yet remains general enough to describe existing layer 3 collision avoidance schemes found in sensor network protocols. The philosophy behind grant-to-send is to have a simple and lightweight but generally applicable collision avoidance mechanism. However, as grant-to-send takes a best effort approach, it has limitations and edge cases, which this section examines.

9.1 Starvation

Since grant-to-send suppresses transmissions, a natural concern is whether a node can starve. For example, node B can starve in Figure 14(a) and Figure 14(b). In both cases however, a single packet loss will break the starvation, as B will be able to fairly enter CSMA at the end of the last grant it heard. Needless to say, most wireless networks exhibit significant packet losses. In Figure 14(a) such an event could be due to the hidden terminal problem; in Figure 14(b) a loss could occur due to external interference, a random packet drop, or any of the other vagaries of wireless. These topologies also assume that no other nodes contend for the channel: if either A or C has to compete with neighbors, grants can suppress them long enough for B to transmit.

9.2 Exposed Terminals

The exposed terminal problem is the inverse of the hidden terminal. It represents a situation in which two nodes think they cannot transmit concurrently because they hear each other, but their transmissions would not collide at the intended receivers. In CSMA, the exposed terminal problem is generally irrelevant. Wireless link layers use synchronous acknowledgments so two adjacent nodes transmitting concurrently will interfere with each other's ack reception.

Grant-to-send suffers more from the exposed terminal problem than CSMA, as Figure 15 shows. In this linear topology, C send multihop packets to A through B and D sends single-hop packets to E. Figure 15(a) shows the optimal schedule for CSMA, where B uses two thirds of the link throughput. Figure 15(b) shows how grant-to-send only gives D one third of the link throughput, as C's grant suppresses D. Allowing B and D to transmit concurrently would require knowledge of B's destination: if it is C, D cannot transmit.

While the exposed terminal is an issue in this and similarly constructed examples, its importance and prevalence in mesh workloads is unclear. Nearby nodes in access meshes use the same gateways, making this diverging traffic pattern uncommon. In the presence of other flows, neither C nor D would be able to use the link throughput as shown in Figure 15(a). Additionally, the throughput gains grant-to-send observes is comparable to the 33% reduction seen in Figure 15(b), such that its benefit balance out its costs.

9.3 Imperfect Grants and Retransmissions

Grant durations are the expected duration of the recipient's transmission(s). A recipient cannot guarantee that its transmissions will complete before the grant expires. There are many cases where a transmission takes longer than the grant: CSMA backoff due to external interference, link-layer retransmissions, and outstanding grants to other nodes (such as in Figure 13(b)) can all delay transmission.

Like CSMA and RTS/CTS, grant-to-send *avoids* collisions: it does not *prevent* them. The analysis in Section 4.3 and experimental results in Section 4.1 showed that shorter-than-optimal grants are superior to CSMA. Therefore, while an adaptive scheme to estimate transmission duration could improve performance, small grants are still better than no grants at all. An adaptive scheme that considers expected transmissions (ETX) is a possible future direction.

9.4 Fairness

Because grants can be multiple packet times, they can harm fairness. One node can receive large grants and use more than its fair share of the channel. Furthermore, grant-to-send as described in this paper simply transmits packets in FIFO order. Large grants can give a larger channel share.

Incorporating fair queuing into grant-to-send is our principal of future work. On one hand, grants provide an excellent measurement of channel occupancy for fair queuing calculations. On the other, these grants are distributed in nature, such that each node can have a different perception of channel use. Our initial investigations into this topic have discovered many aberrant edge cases where disagreeing nodes starve flows, and we are actively exploring algorithms to smooth these inconsistencies and restore fairness.

10. RELATED WORK

RTS/CTS as described in Section 2 is only the most basic instance; there is a long history of research and a plethora of variants. A full survey is beyond the scope of this paper so we merely mention a few prominent approaches. One well-understood limitation of RTS/CTS is that it is less effective when interference range is larger than the communication range [40]. BTMA [35] and DBTMA [21] solve this issue by introducing a sideband channel. In multihop wireless networks, collision avoidance is equivalent to the problem of physical layer spatial reuse. POWMAC [32] improves spatial reuse by exchanging signal information with RTS/CTS. MACA-P [8] accumulates multiple RTS/CTS exchanges to reduce unnecessary RTS/CTS suppression.

Grant-to-send has similarities to 802.11 fast-forward scheduling [11], which embeds an RTS in an acknowledgment packet to enable the receiver to forward immediately. Since fast-forward is built on top of RTS/CTS, it inherits the overhead for short routes. Furthermore, as fast-forward RTS packets do not go through CSMA, it sacrifices fairness between contending transmitters. As fast-forward requires different packet formats, it requires new hardware and is not compatible with existing networks.

Network coding between the link and network layers has emerged as a way to increase throughput by having a single frame contain coded packets for multiple destinations. Grant-to-send is complementary to this work. Protocols such as MORE [17], MIXIT [24], and COPE [25] require receiving complete packets or at least uncorrupted packet fragments: reducing collisions boosts their performance. COPE, for example, explicitly notes that hidden terminal collisions prevent it from achieving any coding gain in TCP, and so evaluates TCP in a single-hop, collision-free network with a logical routing topology.

Approaches such as ZigZag [20] and analog coding [23] can recover collided packets. ZigZag, for example, is designed for one hop AP-client networks, so APs can mitigate the hidden terminal problem between clients. Grant-to-send, in contrast, addresses the problem of collisions in a multi-hop mesh. A combination of these schemes, where grant-to-send runs on commodity mesh nodes and ZigZag runs at gateways with special hardware, could achieve both benefits.

11. CONCLUSION

Grant-to-send provides a simple and inexpensive way to avoid collisions in multi-hop wireless networks. It is easy to implement and can is effective in many protocols and traffic patterns. It does not have the overhead associated with collision avoidance schemes such as RTS/CTS, yet gives significant throughput gains. For UDP, it achieves >99% end-to-end reliability and can provide up to 96% of the theoretical maximum throughput. While TCP suffers from collisions beyond those grant-to-send can avoid, initial results suggest that improved acknowledgment policies improve TCP's throughput to 86% of UDP's.

Grant-to-send has a very low barrier to adoption. As grant-to-send uses the 802.11 duration field, standard 802.11 nodes respect grants and grant-to-send respects RTS/CTS. Many of our experiments occurred in the midst of a busy 802.11 network during working hours. Furthermore, as grant-to-send reverts to simple CSMA at the last hop, so act identically to standard CSMA devices when talking to an AP.

Grant-to-send is able to achieve these results by completely rethinking the information flow in collision avoidance. Rather than avoid loss of its own packets due to collisions, a grant-to-send node helps others avoid collisions. Networks have traditionally been modeled as individuals that compete, at time selfishly, within certain rules. Grant-to-send's efficacy suggests that, at least in wireless meshes, perhaps these rules should enforce more collaborative relationships.

12. REFERENCES

[1] ath9k : driver for atheros ieee 802.11n wlan based chipsets. `http://linuxwireless.org/en/users/Drivers/ath9k`.

[2] Broadcom wireless LAN adapter user guide.

[3] Madwifi : the linux drivers for wlan cards based on atheros chipsets. `http://madwifi.org`.

[4] Reference manual for the NETGEAR ProSafe 802.11g Wireless AP WG102. `http://kbserver.netgear.com/pdf/wg102_ref_manual_4_0_6.pdf`.

[5] Roofnet mesh routing software using Srcr. `http://read.cs.ucla.edu/click/packages/roofnet`.

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

[7] A. Acharya, S. Ganu, and A. Misra. DCMA: A label switching MAC for efficient packet forwarding in multihop wireless networks. *IEEE Journal on Selected Areas in Communications*, 24(11):1995–2004, Nov. 2006.

[8] A. Acharya, A. Misra, and S. Bansal. MACA-P: a MAC for concurrent transmissions in multi-hop wireless networks. *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, (PerCom)*, pages 505–508, March 2003.

[9] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris. Link-level measurements from an 802.11b mesh network. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, 2004.

[10] B. Bensaou, Y. Wang, and C. C. Ko. Fair medium access in 802.11 based wireless ad-hoc networks. *MobiHOC: First Annual Workshop on Mobile and Ad Hoc Networking and Computing*, pages 99–106, 2000.

[11] D. Berger, Z. Ye, P. Sinha, S. Krishnamurthy, M. Faloutsos, and S. Tripathi. TCP-friendly medium access control for ad-hoc wireless networks: alleviating self-contention. *IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, pages 214–223, Oct. 2004.

[12] B. Bershad, S. Savage, P. Pardyak, E. G. Sirer, D. Becker, M. Fiuczynski, C. Chambers, and S. Eggers. Extensibility, safety and performance in the SPIN operating system. In *Proceedings of the 15th ACM Symposium on Operating Systems Principles (SOSP-15)*, 1995.

[13] G. Bianchi, L. Fratta, and M. Oliveri. Performance evaluation and enhancement of the CSMA/CA MAC protocol for 802.11 wireless LANs. *PIMRC'96: Seventh IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, 2:392–396 vol.2, Oct 1996.

[14] G. Bianchi and I. Tinnirello. Kalman filter estimation of the number of competing terminals in an ieee 802.11 network. *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE*, 2:844–852 vol.2, March-3 April 2003.

[15] J. Bicket, D. Aguayo, S. Biswas, and R. Morris. Architecture and evaluation of an unplanned 802.11b mesh network. In *MobiCom '05: Proceedings of the 11th annual international conference on Mobile computing and networking*, 2005.

[16] F. Cali, M. Conti, and E. Gregori. IEEE 802.11 protocol: design and performance evaluation of an adaptive backoff mechanism. *IEEE Journal on Selected Areas in Communications*, 18(9):1774–1786, Sep 2000.

[17] S. Chachulski, M. Jennings, S. Katti, and D. Katabi. Trading structure for randomness in wireless opportunistic routing. In *SIGCOMM '07: Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, 2007.

[18] Y.-C. Cheng, J. Bellardo, P. Benkö, A. C. Snoeren, G. M. Voelker, and S. Savage. Jigsaw: solving the puzzle of enterprise 802.11 analysis. *SIGCOMM Comput. Commun. Rev.*, 36(4):39–50, 2006.

[19] 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.

[20] S. Gollakota and D. Katabi. ZigZag decoding: combating hidden terminals in wireless networks. In *SIGCOMM '08: Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, pages 159–170, New York, NY, USA, 2008. ACM.

[21] Z. Haas and J. Deng. Dual busy tone multiple access (DBTMA)-a multiple access control scheme for ad hoc networks. *IEEE Transactions on Communications*, 50(6):975–985, Jun 2002.

[22] 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.

[23] S. Katti, S. Gollakota, and D. Katabi. Embracing wireless interference: analog network coding. In *SIGCOMM '07: Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 397–408, New York, NY, USA, 2007. ACM.

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

[25] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft. Xors in the air: practical wireless network coding. In *SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 243–254, New York, NY, USA, 2006. ACM Press.

[26] E. Kohler, M. Handley, and S. Floyd. Designing DCCP: congestion control without reliability. In *SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 27–38, New York, NY, USA, 2006. ACM.

[27] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The Click modular router. *ACM Transactions on Computer Systems*, 18(3):263–297, August 2000.

[28] P. Levis, D. Gay, V. Handziski, J.-H. Hauer, B. Greenstein, M. Turon, J. Hui, K. Klues, R. S. Cory Sharp, J. Polastre, P. Buonadonna, L. Nachman, G. Tolle, D. Culler, and A. Wolisz. T2: A Second Generation OS For Embedded Sensor Networks. Technical Report TKN-05-007, Telecommunication Networks Group, Technische Universitat Berlin, 2005.

[29] J. Li, C. Blake, D. S. D. Couto, H. I. Lee, and R. Morris. Capacity of ad hoc wireless networks. In *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 61–69, New York, NY, USA, 2001. ACM.

[30] M. Li, D. Agrawal, D. Ganesan, A. Venkataramani, and H. Agrawal. Block-switched networks: A new paradigm for wireless transport. Accepted submission to *the Sixth USENIX Symposium on Networked Systems Design and Implementation (NSDI '09)*. Obtained from authors.

[31] K. Medepalli and F. Tobagi. On optimization of CSMA/CA based wireless LANs: Part I: Impact of exponential backoff. *ICC '06: IEEE International Conference on Communications*, 5:2089–2094, June 2006.

[32] A. Muqattash and M. Krunz. POWMAC: a single-channel power-control protocol for throughput enhancement in wireless ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 23(5):1067–1084, May 2005.

[33] J. Polastre, R. Szewczyk, and D. Culler. Telos: enabling ultra-low power wireless research. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, page 48, Piscataway, NJ, USA, 2005. IEEE Press.

[34] S. Ray, J. B. Carruthers, and D. Starobinski. Evaluation of the masked node problem in ad hoc wireless lans. *IEEE Transactions on Mobile Computing*, 4(5):430–442, 2005.

[35] F. Tobagi and L. Kleinrock. Packet switching in radio channels: Part II–the hidden terminal problem in carrier sense multiple-access and the busy-tone solution. *IEEE Transactions on Communications*, 23(12):1417–1433, Dec 1975.

[36] A. Vyas and F. Tobagi. Impact of interference on the throughput of a multihop path in a wireless network. In *Proceedings of the 3rd International Conference on Broadband Communications, Networks, and Systems (BROADNETS)*, 2006.

[37] 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.

[38] 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.

[39] A. Woo and D. Culler. A transmission control scheme for media access in sensor networks. In *Proceedings of International Conference on Mobile Computing and Networking (MobiCom)*, Rome, Italy, July 2001.

[40] K. Xu, M. Gerla, and S. Bae. How effective is the IEEE 802.11 RTS/CTS handshake in ad hoc networks. *Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE*, 1:72–76 vol.1, 17-21 Nov. 2002.