Efficiency through Eavesdropping: Link-layer Packet Caching

Mikhail Afanasyev, David G. Andersen[†], and Alex C. Snoeren
University of California, San Diego and [†]Carnegie Mellon University
{mafanasyev,snoeren}@cs.ucsd.edu, dga@cs.cmu.edu

Abstract

The broadcast nature of wireless networks is the source of both their utility and much of their complexity. To turn what would otherwise be unwanted interference into an advantage, this paper examines an entirely backwards-compatible extension to the 802.11 link-layer protocol for making use of overheard packets, called RTS-id. RTS-id operates by augmenting the standard 802.11 RTS/CTS process with a packet ID check, so that if the receiver of an RTS message has already received the packet in question, it can inform the sender and bypass the data transmission entirely.

We present the design, implementation, and evaluation of RTS-id on a real hardware platform that provides a DSP-programmable 802.11 radio. While limited in scale due to restricted availability of the CalRadio platform, our testbed experiments demonstrate that RTS-id can reduce air time usage by 25.2% in simple 802.11b infrastructure deployments on real hardware, even when taking into account all of the protocol overhead. Additionally, we present trace-based simulations demonstrating the potential savings on the MIT Roofnet mesh network. In particular, RTS-id provides a 12% decrease in the number of expected data transmissions for a median path, and over 25% reduction for more than 10% of Roofnet paths.

1 Introduction

Multi-hop wireless networks are becoming a popular mechanism for providing Internet access, both in urban areas [6] and in rural and developing settings [18]. By reducing the need for a fixed wired infrastructure, they offer the hope of providing cheaper connectivity and faster deployment. These networks, however, face a number of challenges not shared by their wired counterparts: interference, multi-path losses, and rapidly changing, unpredictable connectivity patterns. Wireless networks are by nature *broadcast*: a transmission from one node may interfere with or be received by multiple other nodes. The broadcast nature of these networks and the requirement that nodes forward traffic on behalf of one another is one of the primary scaling limitations of multi-hop wireless networks [17].

The question we ask in this paper is: how can we turn the broadcast nature of wireless to our advantage, instead of leaving it purely an interference-causing liability. Past approaches to doing so include caching opportunistically overheard objects (e.g., in satellite-based distribution systems [2]); by modifying ad hoc routing protocols to enable them to acknowledge packets received later in the forwarding chain [7]; by using network coding on bidirectional traffic streams [15]; and, most recently, by using network coding to achieve similar benefits without explicit coordination [10]. We examine instead a simple per-hop link-layer modification, that we call *RTS-id*, that takes advantage of overheard packets in a protocol and topology-independent manner that requires only the cooperation of adjacent nodes in a path.

While previous systems like ExOR and MORE have demonstrated dramatic performance improvements, ranging up to throughput factors of 2–10 [7, 10], it is important to note that these improvements do *not* apply to interactive and highly asymmetric TCP connections common in access networks. Even the most recent opportunistic routing techniques in MORE require batching packets together for transmission, which interacts poorly with TCP's congestion control. Furthermore, to realize these gains, the entire network must be upgraded to support the enhanced routing and forwarding architecture. In contrast, RTS-id is backwards compatible with existing 802.11 hardware: individual nodes can be upgraded by replacing the 802.11 driver and/or firmware, yet they will continue to interoperate with legacy nodes. We verify that the RTS-id extensions are ignored by hardware that does not support it with no ill effects. Furthermore, while substantially more modest than the bulk transfer improvements demonstrated by other systems, the gains we report are independent of transport-layer protocol: they are equally applicable to UDP and TCP. Finally, we demonstrate that significant gains can be had in typical infrastructure deployments as well.

The key property that we build upon is that wireless delivery is probabilistic. Two nearby nodes may hear 90% of each others' transmissions, and two nodes farther away may hear only 20%. This highly variable, unreliable packet reception is the critical challenge to the design of practical ad hoc routing protocols; to minimize metrics such as the expected transmission count [12] or expected transmission time [5] protocols will often select paths in which there is a non-zero chance that packets could be overheard farther along the forwarding chain.

Like previous schemes [15], we argue that all nodes should optimistically cache recently received packets regardless of their destination. Rather than require additional coordination, however, we propose to extend the normal 802.11 RTS/CTS exchange to include a packet ID. If a receiver already has a packet cached, it can respond to an RTS directly with an ACK. Our RTS-id mechanism has the potential to elegantly optimize at least three distinct common occurrences in wireless 802.11 networks:

- **"Node-to-node via access point"**. When two nodes on the same wireless LAN communicate, they *must* do so through the AP(s) to which they are associated, even if they can directly hear each others communication. RTS-id allows the AP to avoid relaying packets to the receiver if the receiver heard the initial transmission.

- **"Hop-over" transmission**. More generally, when sending packets through a series of nodes $A \rightarrow B \rightarrow C$, the probability of C hearing A's initial broadcast is often non-zero. When B then attempts to relay the packet to C, RTS-id allows it to immediately respond "I already have this packet."

- **Packet retransmissions.** If a link-layer ACK is lost, RTS-id prevents spurious retransmissions by short-circuiting the retransmission.

The key contributions of the current work are the design of RTS-id and a backwards-compatible implementation on a real 802.11 hardware platform, CalRadio. We show that RTS-id can decrease air time usage by 25.2% when two nodes communicate to each other through an access point in our testbed. Moreover, we demonstrate via simulation the potential performance improvement in a large-scale multi-hop deployment. Using publicly available data from the MIT Roofnet community network, we show that RTS-id can decrease by 12% the expected number of data packet transmissions for the median route when compared to Roofnet's existing routing protocol, and by over 40% for the most-improved routes. Even if a network does not normally use RTS/CTS, RTS-id decreases by up to 15% the air time for fully optimized Roofnet routes after accounting for the additional overhead (and without considering any potential benefits from avoided hidden terminal collisions). Perhaps most importantly, we show that RTS-id can significantly enhance the performance of much simpler routing algorithms—so much so that they out-perform Roofnet's complex routing algorithm that requires maintaining accurate, up-to-date loss information about all node pairs.

The remainder of this paper is organized as follows. We begin in Section 2 with an overview of previous schemes that leverage overhearing. Section 3 presents the design of RTS-id, and we describe our prototype CalRadio implementation in Section 4. We systematically evaluate the potential performance improvement in both infrastructure networks (Section 5) and the MIT Roofnet mesh network (Section 6) before discussing security concerns (Section 7) and future work in Section 8.

2 Related work

The work philosophically most related to ours is ExOR [7, 8] and the more recent MORE [10]. Both protocols essentially define new, bulk-transfer transport protocols to increase efficiency. ExOR aggressively batches packet transmissions together to take advantage of overheard transmissions. It substantially increases packet throughput, but to do so it requires a redesigned ad hoc routing protocol, and its large batch sizes render ExOR functionally incompatible with traditional transport protocols like TCP and latency-sensitive applications. (The aggressive batching can increase latency by up to 3.5 seconds [7]; its authors acknowledge that "ExOR's batches are likely to interact poorly with TCP's window mechanism" [8].) MORE's operation is similar, but it uses random network coding to avoid the need for ExOR's scheduler. Mostly by increasing opportunities for spatial reuse, MORE achieves unicast throughput 22–45% higher than ExOR's.

In contrast, RTS-id targets the operation of legacy routing and transport protocols, sacrificing some of their impressive performance gains as the price of broader applicability. RTS-id can reduce the number of transmissions required by any existing transport protocol; no changes to applications or operating systems are required.

An alternative technique proposed to harness broadcast is network coding, e.g., COPE [15]. Network coding does not target opportunistic overhearing; rather, it takes advantage of the fact that a sender in the middle of a three-node chain can be heard by both of the nearby nodes during a single transmission, allowing bidirectional traffic to be sent using three transmissions instead of four. Because COPE exploits a different property of broadcast, we believe that its benefits are orthogonal to—and quite possibly compatible with—those from RTS-id. We defer an analysis of this complex interaction for future work. We note, however, that many coding-based approaches

System	Coding	Batch size (packets)	Per-packet overhead	Median improvement
ExOR [7]	none	32	44–114 bytes	60%
MORE [10]	intra-flow	32	<70 bytes	95%
COPE [15]	inter-flow	variable	variable	25–70%
RTS-id	none	1	4 bytes (in RTS)	15–25%*

Table 1: A comparison of mechanisms that leverage the broadcast nature of the wireless channel. Improvement values for ExOR and MORE are taken from the three-node topologies extracted from [10, Figure 6]. COPE numbers reflect TCP and UDP, but *require* bi-directional communication in this simplistic topology. More modest performance gains are possible for uni-directional flows in more complex topologies. *Due to hardware issues, we report savings in terms of air time usage as opposed to throughput; we improve UDP throughput by 26.1% compared to a network with RTS/CTS enabled.

also provide relatively modest gains for the legacy traffic that RTS-id targets, performing optimally only with symmetric UDP traffic. Depending on the degree of asymmetry, the performance improvement may be as low as 5–15%—for UDP. The best case TCP improvement for COPE was 38%, but this is only for networks that do not require RTS/CTS (i.e., do not have any hidden terminals).

Table 1 attempts to summarize the various features of each of these systems. Recognizing that comparing performance claims across implementations and testbeds is problematic, we attempt to give a flavor of the order of magnitude of performance gains offered by each system. We use Roofnet's routing protocol, Srcr, as a baseline, and include the median performance gain over Srcr reported by each system's authors in a three-hop topology. While only MORE and COPE incur coding overhead, the per-packet overhead can be substantial in many cases, rendering the techniques inappropriate for small flows. We were unable to determine the COPE packet header size from [15], but it appears to be of similar order to ExOR and MORE.

A key distinction of RTS-id is its independence from the routing protocol. In particular, we show it can improve the performance of *any* routing protocol—not just Srcr. Considerable previous work has examined techniques for selecting routes to leverage opportunistic forwarding opportunities in multi-hop networks [11, 16]. Some prior protocols may be easier to implement than Srcr in certain networks; others may be so computationally expensive that it could be more efficient to use simple routes and allow RTS-id to optimize them on-the-fly.

In some ways, RTS-id's packet cache is reminiscent of the duplicate-suppression cache used in the original DARPA packet radio network [14]. That mechanism lacked RTS-id's query mechanism, however, only enabling receivers to avoid re-sending packets if they were incorrectly retransmitted or looped.

More generally, Santos and Wetherall proposed a compression mechanism for suppressing long-range packet duplication on wired links [19], later extended to sub-packet duplication by Spring and Wetherall [21]. Unfortunately, these mechanisms do not directly translate to opportunistic wireless: they rely on a near-perfect synchronization of the sender and receivers' "dictionaries," exactly the knowledge that does *not* exist in our target environment. However, these techniques suggest promising ways (e.g., combining header compression with RTS-id) for RTS-id to exploit cross-flow and longer-term overhearing. We plan to examine such extensions in future work.

3 RTS-id

Our proposed technique, RTS-id, adds a small exchange *before* packet transmission to ask the receiver if it already has the packet in question. Receivers maintain a small cache of recently observed packets that they check during this exchange. To reduce overhead and ensure backwards-compatibility, RTS-id piggy-backs this query on the existing 802.11 request-to-send (RTS) frames.

RTS/CTS is normally used to reserve the medium for a packet transmission to prevent hidden terminal problems [3]. It operates by having senders broadcast a "request to send" (to a particular receiver) specifying the expected duration of the frame exchange. In accordance with the 802.11 standard, if the receiver determines the channel to be idle, it will reply with a "clear to send" (CTS) frame containing the expected remaining duration of the frame exchange, permitting the sender to begin transmission and informing nearby nodes to remain silent. RTS-id adds a second possibility: the receiver can directly "ACK" the packet with a special CTS-ACK frame.

This section first examines the roles of senders and receivers in RTS-id, then discusses the design alternatives to identify packets. Finally, because RTS-id increases the size of RTS frames (or necessitates their use in a system that does not use them), we discuss how senders and receivers can dynamically enable RTS-id based upon an on-line determination of whether it would benefit them.

3.1 Sender and receiver operation

RTS-id senders operate as shown in Figure 1: they first decide whether to use RTS-id for a packet. If so, they

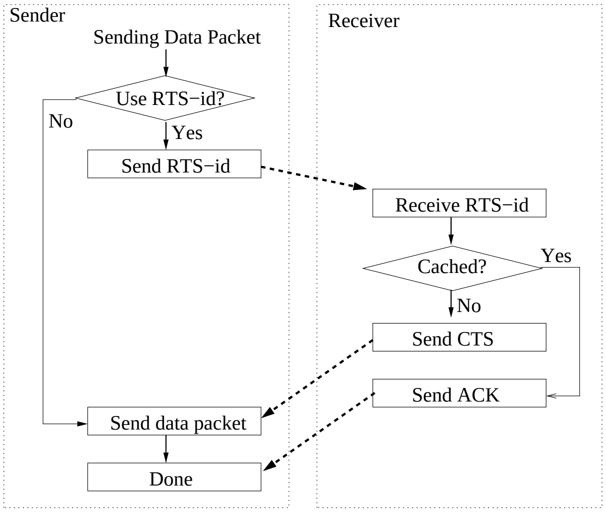


Figure 1: RTS-id operation. For clarity, this figure assumes that the sender does not fall back to normal RTS/CTS use.

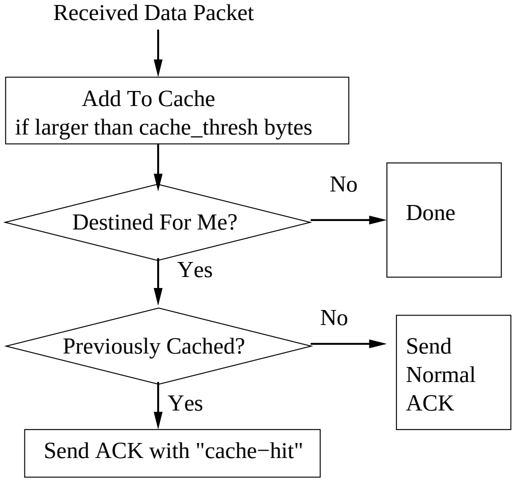


Figure 2: Received packet processing.

transmit an RTS-id frame to the receiver, and expect to receive either a CTS-ACK (the receiver has the packet already) or a normal CTS (the receiver does not have the packet; the sender must transmit). An RTS-id frame is simply a standard RTS frame extended to include a packet ID. With RTS-id, however, rather than setting the duration field to the standard value, it sets it only to the time in microseconds required to transmit the CTS frame and one SIFS (short inter-frame space) interval. This way, nodes overhearing the RTS will only consider the channel reserved for the RTS-id/CTS exchange at this point.

Upon reception of an RTS-id frame, the receiver checks its local packet cache for a packet whose ID matches that in the RTS frame. If present, the receiver sends a CTS-ACK and processes the frame as if it had been received normally. A CTS-ACK is simply a normal CTS frame with the remaining duration field set to zero. This both signals to the sender that the packet was already received, and resets the network allocation vector (NAV) for other stations in the contention domain. If the packet was not found, the receiver sets the CTS duration field to be the same value that would have been used in response to a normal RTS frame, reserving the channel for the time expected to transmit the pending frame, plus one ACK frame and two SIFS intervals.

When a node receives a normal data frame, it operates according to the flowchart in Figure 2. It inserts into a small FIFO cache all received packets larger than `cache_thresh` bytes, regardless of the packet's source or destination. The `cache_thresh` avoids wasting cache entries on small packets such as TCP ACKs. If the packet was previously cached, the receiver informs the sender that the transmission could have been avoided, which enables the adaptive enabling scheme below.

3.2 Choice of hash and collisions

RTS-id uses a 32-bit hash of the IP packet contents—*not* the link-layer frame—as the packet ID. Such a small hash is acceptable if it provides three properties:

Low drop and duplication rate: A hash collision results in both a drop (of the transmitted packet) and a duplication (of the cached packet it collides with). A 32-bit hash with a 64-packet cache will drop about 1 in 67 million packets due to hash collisions. This rate is much lower than typical end-to-end loss on wireless networks.

Independent collisions for transport-layer retransmissions: If the drop probability is non-negligible, then a collision that prevented a particular frame exchange must not cause the end-to-end retransmission of that packet to also be dropped with high probability. This property is provided as long as 1) the hash of the retransmitted packet is different from that of the original; or 2) the contents of the cache differ during the retransmission. Fortunately, both conditions are likely to hold, as several fields in the packet typically change when a packet is retransmitted at the transport or application layers, such as the IP ID, TCP timestamps, DNS query IDs, etc.

Resistant to attacks: The hash should ensure that a non-local attacker cannot guess the ID of a packet and that no attacker can easily craft a packet that will collide with a target packet. We assume that an attacker who transmits on the order of 2^{32} packets over the course of a few seconds has at his command a more effective way of denying service than causing packet collisions. We discuss the security implications of RTS-id, with and without strong hash functions, in Section 7.

3.3 Adaptively enabling RTS-id

RTS-id adds 32 bits of overhead to the small RTS packets. On links in which RTS-id does not provide benefit, this cost may loom large, because 802.11 transmits RTS/CTS packets at the lowest possible rate, 1 Mbps, while the data may be sent at higher rates. Moreover, for networks that would not otherwise use RTS/CTS, the insertion of an entirely new frame exchange comes at considerable cost. Each sender therefore dynamically determines whether or not to use RTS-id when communicating with a particular receiver, based on its past history of cache hits and the size of the packet it is about to transmit.

First, RTS-id processing only considers packets larger than `cache_thresh` $\approx$ 500 bytes. Smaller packets are transmitted directly (they may, however use normal RTS/CTS depending on the station configuration). For large packets, every participating receiver maintains an RTS-id cache, regardless of whether senders choose to use it. On receiving a packet, the receiver checks its cache to see if the packet had already been heard. If it had, the receiver sets a bit in the ACK packet it sends in response to the packet arrival. Otherwise, it leaves this bit unset. The sender thus is able to determine which packets *would have* resulted in a cache hit had it used RTS-id.[1]

On each packet, the sender calculates the (possibly negative) time saved, T_s, by using RTS-id. In the calculation that follows, T_{rtscts} is the time required for a normal RTS-CTS exchange, or zero if RTS-CTS is not enabled.

$$\begin{aligned}
B_s &= \text{The bytes saved} \\
&= \begin{cases} 0 & \text{if no cache hit} \\ \textit{Packet size} & \text{if cache hit} \end{cases} \\
T_s &= \frac{B_s}{rate_{tx}} - (T_{rtsid} - T_{rtscts}).
\end{aligned}$$

The sender maintains for each (link-level) receiver an exponential weighted moving average with parameter $w \approx \frac{1}{200}$ of the time saved for each packet:

$$savings = (1-w) \cdot savings + w \cdot T_s.$$

If the estimated time savings for a particular receiver is large enough, the sender will enable RTS-id. It is not necessary to explicitly enable RTS-id on the receiver: it can promiscuously cache packets whenever sufficient memory and power are available, and may always respond to an RTS-id packet with a normal CTS frame.

4 Implementation

We have implemented RTS-id in a backwards-compatible fashion. RTS-id stations inter-operate seamlessly with non-RTS-id stations, enabling enhanced performance between adjacent RTS-id capable stations. Our implementation uses the CalRadio 1.0 platform designed and manufactured by CalIT2 [9]. The CalRadio is a low-cost software radio platform consisting of an ARM processor, an on-board Texas Instruments DSP, and a Prism 802.11b baseband processor. The salient feature of the CalRadio for our purposes is that the MAC protocol is implemented almost entirely in C, which allows us to change the format and contents of the RTS and CTS packets. The ARM has access to 4 MB of flash ROM, 2 MB of static RAM and 16 MB of SDRAM, while the DSP operates with 512 KB of RAM. The 802.11 MAC protocol is implemented on the DSP, while the operating system (μCLinux 2.4.19) and user-level programs run on the ARM.

RTS:	FC (2)	Dur (2)	Source (6)	Dest (6)	FCS (4)	
RTS-id:	FC (2)	Dur (2)	Source (6)	Dest (6)	FCS (4)	ID (4)

Figure 3: RTS and RTS-id packet formats.

4.1 Packet details

The RTS-id packet is a simple extension to the standard 802.11 RTS packet as shown in Figure 3. Note that the new ID field is sent *after* the normal RTS frame fields, including the frame check sequence (FCS). Furthermore, when transmitting the RTS-id frame, the length field of the PLCP header is set to the length of the standard RTS frame, *not including* the new ID field. Hence, spec-compliant 802.11 stations that do not support RTS-id will not even decode the hash field, and the frame will look like a normal, well-formed RTS frame.[2] RTS-id capable stations, however, expecting an RTS-id frame, will know to decode the additional field.

It is important to note that the use of RTS-id does not interfere with the normal ability of RTS/CTS to prevent hidden terminals. The duration specified by the sender's RTS-id frame will reserve the channel until the end of the RTS-id/CTS exchange. If the data frame is eventually sent, its duration field will update the NAV for all stations in range of the sender. Nodes that hear only the CTS frame will obey its duration field. Because, however, we insert a different value into the RTS-id duration field, the receiver no longer knows how long the pending packet will take to transmit, and is unable to accurately fill out the duration field in the corresponding CTS frame.

To resolve this problem, stations sending a CTS can estimate the appropriate duration based upon a packet size

[1] To avoid the need to redefine the ACK packet in practice, we overload the "retry" bit. In our experience, current 802.11 devices do not set the retry bit on ACK frames.

[2] The Atheros chip sets we have tested properly decode the RTS-id frame as an RTS. Due to time constraints, we have not yet conducted an exhaustive test of other 802.11 devices. In the worst case, non-compliant stations will simply discard the seemingly mal-formed RTS-id frame with no ill effects.

of cache_thresh (smaller packets would not have instigated an RTS-id exchange) and the previous transmission speed used by the sender. (Over-estimating the size prevents hidden terminal problems, but potentially waste air time. Under-estimating creates a small window where a collision may occur that normal RTS/CTS would have prevented.) If greater accuracy is needed, the low-order bits of the RTS-id duration field can be used to encode the approximate size of the pending data packet. Our prototype, however, does not yet implement this extension.

While the ID field is not covered by the FCS (in order to preserve backwards compatibility), a corrupt ID field has little effect. All nodes in our implementation recompute the ID of received packets before insertion into the cache or local delivery, so there is no danger of cache or data corruption. Hence, there are only two issues of concern: First, an ID that should hit in an overhearer's cache is corrupted so that it misses. In this case, an avoidable transmission occurs, resulting in a slight performance decrease. The second, somewhat more expensive case occurs when an ID is corrupted so that it collides with that of a previously overheard case. This situation is no different than a normal hash collision, and occurs (assuming a binary symmetric channel) with equal probability. Such a collision results in a drop (of the corrupted packet) and retransmission (of the packet the ID collided with), impairing performance but not correctness.

4.2 Packet caching and RTS-id

According to the 802.11 specification, a station must respond within 10 microseconds to an RTS request. To interoperate with legacy stations, RTS-id nodes should conform to this response time requirement for both CTS and CTS-ACK packets. We therefore implement the packet cache on the DSP. Due to the tight cycle budget, our implementation uses the CRC32 checksum of invariant [20] packet contents (including the transport layer header and a portion of the payload) as its ID. This choice is obviously deficient with respect to attack resilience; a future implementation will use the low-order 32 bits of a strong cryptographic hash.

4.3 Test-bed deployment

Our current test-bed consists of three CalRadio devices. While CalIT2 distributes CalRadio with basic 802.11b PHY code, the publicly available MAC code is far from complete. We have extended the provided code base to support the core of the 802.11b MAC protocol, including data, ACK, RTS/CTS, and RTS-id/CTS-ACK frames as well as link-layer retransmission and collision avoidance. Due to a hardware defect with the CalRadio platform, however, we are not able to faithfully implement carrier sense. Our implementation is sufficient to exchange packets both between CalRadios and with other, Atheros-based 802.11b devices in our lab, but suffers from an unusually high loss rate due to lack of carrier sense. We have attempted to ameliorate this issue by introducing a fixed, per-station delay after the completion of a previous transmission to avoid frequent collisions. While this slotting mechanism does not interfere with the operation of RTS-id, it has the unfortunate effect of decreasing the effective channel utilization. When RTS(-id)/CTS is enabled, however, this limitation impacts only the RTS/CTS exchange, as the successful completion of such an exchange will reserve the channel for data transmission.

5 Infrastructure networks

We use our testbed to show RTS-id's ability to optimize "node-to-node" transmissions between nodes communicating through the same access point, finding that RTS-id reduces the number of data frame transmissions by 50.7% and improves bulk UDP throughput by 26.1% in our testbed configuration. These results translate into a 25–46% reduction (depending on data rate) in air time usage compared to a network that does not use RTS/CTS.

5.1 Node-to-node transmission

When two nodes on the same infrastructure-based wireless network communicate with each other, they must relay their packets through an AP with which they are associated, even if they are within transmission range of each other. RTS-id provides a natural mechanism to optimize this communication by allowing the AP to short-circuit its retransmission of the packet.

We are not aware of empirical data quantifying the overhearing prevalence in typical access point deployments. Hence, we attempt to emulate realistic situations such as meetings or office collaborations by setting up three nodes in a controlled topology. We physically connect three nodes together through a series of splitters and variable attenuators so that the path loss between A and B is L dB, B and C is 20 dB, and the loss between A and C is $(50+L)$ dB. We have found that our CalRadios can tolerate path loss of approximately 100 dB in our noise-free configuration, so we can control the prevalence of overhearing by adjusting the value of L.

Node A is configured to use node B as its first-hop router. Node B plays the role of an access point by forwarding A's packets on to node C. We use the ttcp application to send 1100-byte UDP packets and report our results both in terms of individual frame exchanges and path throughput. To reduce the impact of external nodes, we set the CalRadios to channel nine, a relatively quiet

Node	Tx success	CTS-ACK	CTS	ACK
A	99.3%	0%	56.6%	99.9%
B	98.6%	0%	45.0%	99.9%
110-dB path loss : **2.05** data frames per packet, **29.13 KBps**				
A	99.7%	0.1%	96.6%	99.8%
B	99.9%	97.6%	1.1%	100%
100-dB path loss: **1.01** data frames per packet, **36.74 KBps**				

Table 2: Experimental results from the CalRadio test-bed.

channel in our building. All three nodes support RTS-id. Node *A* first sends the packet to node *B*. The Linux networking stack on node *B* then forwards the packet to node *C*. Meanwhile, node *C* is promiscuously listening to all packets; since all three nodes are in close physical proximity, *C* frequently overhears *A*'s transmissions to *B*. In such cases, it caches the packet and records the packet ID. When *B* subsequently sends an RTS-id frame to *C* requesting to transmit a packet with an ID that *C* just overheard, *C* delivers the cached copy to the Linux kernel and responds with a CTS-ACK preventing the transmission of the data frame. If *C* did not overhear the original transmission, it sends a CTS, and *B* transmits the data frame to *C*, which acknowledges its receipt and delivers the packet to the application.

5.2 Transmission reduction

To demonstrate the effectiveness of RTS-id, we conduct two separate experiments with drastically different overhearing rates. In the first, we set the variable attenuator to $L = 60$ dB, resulting in a path loss from *A* to *C* of 110 dB, effectively preventing overhearing. In the second, we adjust the attenuator to 50 dB, giving an effective path loss of 100 dB which results in significant overhearing. Both experiments attempt to transmit a train of UDP packets from *A* to *C* at 1 Mbps with RTS-id enabled. We set the link-layer retransmission count to ten, meaning a sender will attempt the RTS-id/CTS/data/ACK frame exchange at most ten times for each packet.

Table 2 presents the results of these experiments. For each node, we show the fraction of attempted packet transmissions successfully completed by that node, as well as the fraction of RTS attempts that were met with either a CTS-ACK (and therefore avoided) or a regular CTS (and therefore transmitted). Finally, we show the percentage of transmitted data frames that were successfully acknowledged by the receiver.

Due to lack of carrier sense, RTS/CTS exchanges fail relatively frequently in our experiment, especially without overhearing. Recall that the frame exchange will be attempted up to ten times for each packet, so the overall transmission success rate is still quite high. In contrast, almost no data frames are dropped. The stark difference in RTS/CTS success rates between the two experiments is due to the fact that node *B* rarely needs to transmit data frames in the overhearing case, so there is far less contention for the channel.

As expected, node *C* overhears a large fraction of the transmissions from *A* to *B* when $L = 50$ dB; hence, it is able to prevent all but 1.1% of the packets from being forwarded by *B*. Comparing the overhearing case with the non-overhearing case, RTS-id provides dramatic savings, reducing the number of data frames transmitted per successfully delivered packet from just over 2.05 (recall that 2.0 is the best case without overhearing if there is no data frame loss) to 1.01, a 50.7% reduction in transmission rate, which resulted in a 26.1% improvement in end-to-end bandwidth in our testbed configuration.

5.3 RTS/CTS overhead

Most infrastructure deployments do *not* enable RTS/CTS by default; as a result, using our adaptive algorithm an AP will only enable RTS-id if the expected savings outweigh the additional overhead (Section 3.3). Due to the lack of carrier sense, we are unable to effectively measure the performance improvement in this scenario. Using statistics collected from the experiments depicted in Table 2, however, we can calculate the air time usage for a non-RTS-id network from the non-overhearing case by simply summing the amount of air time used by the data transmissions (DIFS + data + SIFS + ACK), as RTS/CTS frames would not be used in this case. Conversely, we can calculate the total air time usage for an adaptive RTS-id deployment by summing the air time used by the data transmissions from *A* to *B* in the overhearing case and combining that with the data transmissions and RTS-id/CTS frames from *B* to *C* (DIFS + RTS-id + SIFS + CTS-ACK) or (DIFS + RTS-id + SIFS + CTS + SIFS + data + SIFS + ACK). Considering the 1100-byte packets transmitted at 1 Mbps in the previous experiment, an RTS-id enabled network would use 46.1% less air time than one not using RTS/CTS at all. The savings reduce to 25.2% if one considers MTU-sized packets at 11 Mbps.

6 Mesh networks

Due to the limited availability of CalRadios, we use trace-based simulation to evaluate the effectiveness of RTS-id in a multi-hop mesh network. Its benefits in this scenario range from a 20% savings for the median route at 1 Mbps to a 12% savings for the median route at 11 Mbps. In general, we find that RTS-id benefits even highly optimized routing mechanisms, but that its benefit is somewhat inversely proportional to how optimal the route choice and—more significantly—rate and power selection is. This

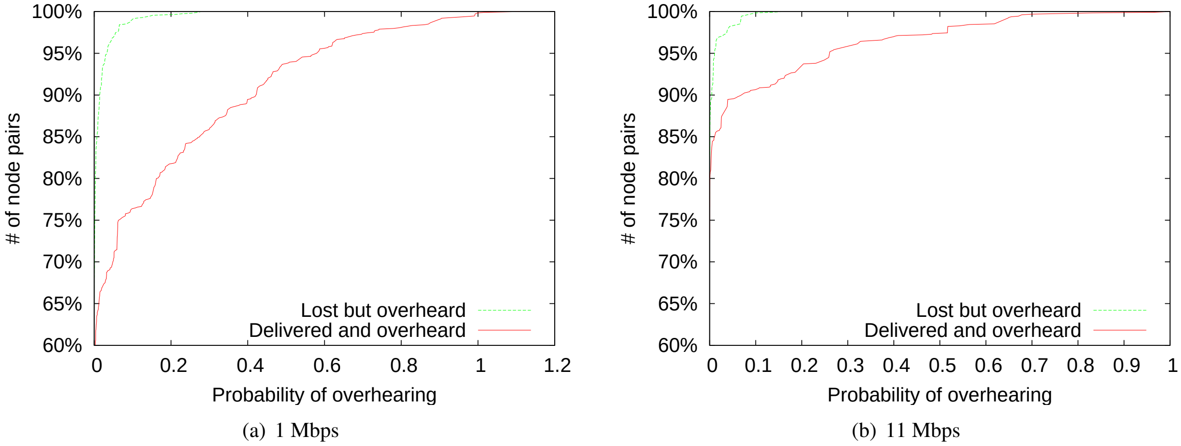


Figure 4: Overhearing in Roofnet. We plot the probability that any transmission along an ETT path is overheard by a node *further* along the path. We plot two mutually exclusive cases: when intended destination does and does not also receive the packet. Both y axes start at 60%.

follows intuitively: a large amount of overhearing along a transmission path is a possible signal that the sender is transmitting "too well" to reach the receiver, and so could perhaps spend that extra signal/noise ratio by using a faster transmission rate or lower power.

Our mesh evaluation first considers how often a node can overhear transmissions in realistic environments at fixed rates, and how that impacts the number of transmissions required to forward a packet using the popular ETT routing metric [1]. We then evaluate the effect of rate adaptation and alternate traffic patterns. Next, we examine how RTS-id provides greater benefit to less sophisticated route selection metrics, and then evaluate the savings provided by RTS-id in an environment that does not use RTS/CTS by default.

Dataset: Our analysis uses the Roofnet mesh network dataset [1]. The dataset contains the fraction of MTU-sized packets transmitted at each node that are received at every other node. In other words, the dataset specifies $\Pr_r[A \rightarrow B] \;\forall\, A,B \in G, r \in \{1,2,5.5,11\}$ Mbps. The dataset was collected on the 38-node MIT Roofnet network as follows:

> Each node in turn sends 1500-byte 802.11 broadcast packets as fast as it can, while the rest of the nodes passively listen. Each sender sends for 90 seconds at each of the 802.11b bit-rates. The experiment uses 802.11 broadcast packets because they involve no link-level acknowledgments or retransmissions. [1]

The reception rates were measured with only one Roofnet node was transmitting at a time—though there likely were other 802.11 sources during the experiment. It is possible that simultaneous Roofnet transmissions would decrease the rate of overhearing as the load on the network increases, but it unclear how significant this effect would be. Unfortunately, there are no published Roofnet datasets under such conditions.

Route computation: Unless specified otherwise, we compute routes using a modified ETT metric [5], which roughly approximates the expected transmission time (including acknowledgments, retries, and back-offs) required to successfully transmit a packet across a given link. ETT is derived from the expected transmission count (ETX) [12], which has been shown to outperform previous routing metrics [13]. The ETX metric is defined for each pair of nodes at rate r, and is computed as $1/(p_f \cdot p_r)$, where p_f is the transmission success rate in the forward direction (i.e., $\Pr_r[A \rightarrow B]$), and p_r is the success rate in the reverse direction ($\Pr_r[B \rightarrow A]$). A key distinction between traditional ETX and ETT is that, in ETT, p_r is based upon the measured delivery rate of 60-byte ACK packets transmitted at one Mbps. Unfortunately, the 2004 dataset does not include the 60-byte loss data necessary to calculate ETT; hence, we modify ETT slightly to always consider the delivery rate on the reverse channel at one Mbps, but are forced to use the rate for 1500-byte packets, which is likely to be lower. We then use a shortest-path algorithm to find the path between each pair of nodes that minimizes the total ETT metric.

6.1 Overhearing prevalence

Overhearing is common in the Roofnet topology, particularly at lower speeds. We compute the probability of overhearing by all node pairs that occur together on some valid source-destination route in the topology. To do so, we create a superset distribution of packet reception $\Pr_r[A \rightarrow \{B,C\}], \Pr_r[A \rightarrow \{B,C,D\}] \ldots$, the probability that a packet transmitted by A to B at rate r is received by all possible combinations of receivers $\{B,C\},\{B,C,D\}$, etc.

Figure 4 shows the CDF of the overhearing probabilities for paths between each pair of nodes in the network. We consider all ETT paths $P \in G$ longer than one hop, where $P := X_1 \rightarrow X_2 \rightarrow \ldots X_n$, and compute the probability that any transmission along the path is overheard by a node further along the path. That is, X_i's transmission to X_{i+1} is overheard by $X_j, j > i+1$. There are two cases of interest: where X_{i+1} does not and does also receive the transmission. Our current implementation of RTS-id does not immediately assist in the first case where the packet is overheard but not delivered to its intended next hop. The packet will need to be retransmitted by the sender until it has been received at the next-hop—although the subsequent transmission by the next-hop will be avoided. While it may be possible to extend RTS-id to prevent these retransmissions, doing so would require knowledge of the intended route, and the situation is unlikely to occur frequently in practice with reasonable route selection. Indeed, it occurs only rarely in the Roofnet dataset. Hence, for simplicity, we forgo the seemingly minimal potential performance improvement and only act upon packets that are both overheard and successfully received by their intended recipient. Transmissions between a fifth of all node pairs are overheard more than 20% of the time at 1 Mbps. Overhearing is less common at higher speeds. At 11 Mbps, only 5% of node pairs are overheard more than 20% of the time. In an outdoor environment like Roofnet, however, nodes frequently transmit at lower link rates, so ample opportunity exists to exploit overhearing.

6.2 Transmission reduction

To evaluate whether RTS-id can exploit overhearing and ACK loss to avoid transmissions, we construct a statistical model to estimate the expected number of transmissions along each path. We examine each source/destination pair individually, and for each pair:

1. Create a state machine in which each state corresponds to the set of nodes that have heard a given packet. For example, if a route has three hops: $A \rightarrow B \rightarrow C \rightarrow D$, the initial state is A and the final state is $ABCD$.

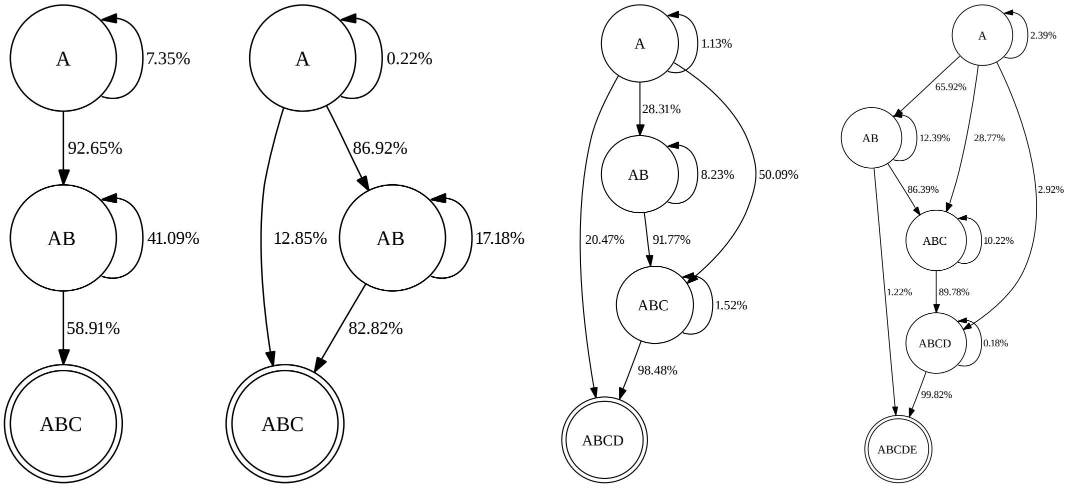
2. Next, calculate the probability for each state transition under normal 802.11 and using RTS-id. Initially, we neglect the RTS/CTS exchange, and consider only data packets and link-layer ACKs. Transitions exist between a node and itself (self-loops due to failed transmissions, regardless of overhearing), adjacent nodes on the path (successful normal transmissions) and, for RTS-id, a node and all subsequent nodes in the path (due to overhearing). For the base 802.11 case, we consider a transmission successful if the packet reaches the receiver and the corresponding ACK reaches the sender; the probabilities are drawn from the Roofnet data set. For RTS-id, we ignore the ACK delivery rate because any spurious retransmission attempts will be bypassed by RTS-id, and compute state transition probabilities based upon the overhearing distribution. For simplicity, in each state we assume that the packet is only transmitted by the node furthest along the path.

3. Finally, calculate the expected number of transitions (i.e., packet transmissions) required to reach the last state (where the destination has received the packet) from the first state. We compute the expected number of transmissions twice, once using the RTS-id transitions and probabilities, and once using only the on-path $A \rightarrow AB$ and $AB \rightarrow ABC$ base-case 802.11 transitions.

Without overhearing, only two transitions leave each state: $AB \rightarrow ABC$ for successful delivery, and $AB \rightarrow AB$ for failure. With overhearing, the picture is far more interesting. Figure 5 shows four state machines corresponding to actual paths in the Roofnet dataset. Figure 5(a) depicts a path with no overhearing; that is, C never overhears A's transmission, therefore the only possible transition is from A to AB, which occurs 92.65% of the time (the other 7.35% of the time the packet is lost and must be resent). The link from B to C is much worse, and succeeds less than 60% of the time. Figure 5(b) shows a simple overhearing scenario, where 12.85% percent of the time A's transmission to B is overheard by C.

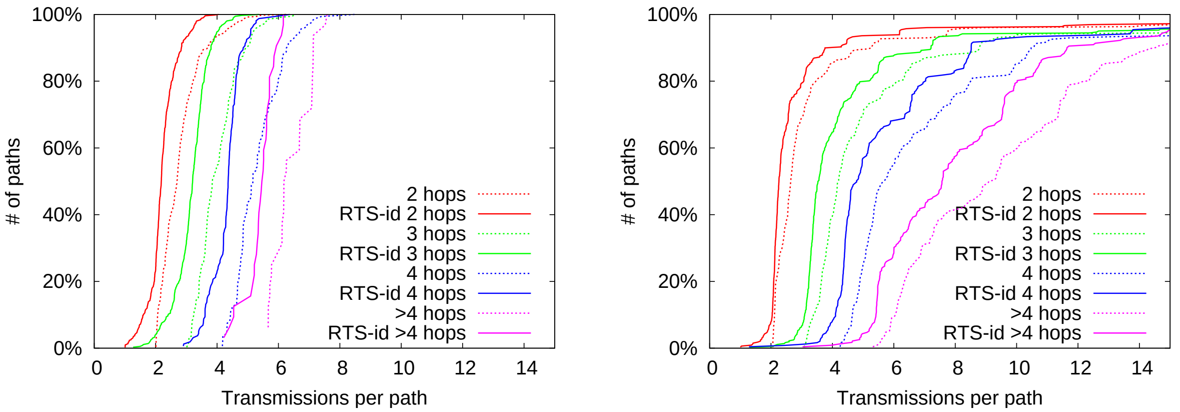
The remaining two examples depict more complicated transitions that occur with longer paths. Figure 5(c) shows a case in which roughly 20% of the time, a packet can be transmitted directly from A to D, obviating the need to forward through B or C. The careful reader may wonder why ETT selected B rather than C as the first hop in the path, as $A \rightarrow C$ appears to have the higher success probability. In this case, the return path (not shown) from C to A is quite lossy, so ETT correctly avoids this hop because the ACKs will be lost. RTS-id, on the other hand, benefits from this overhearing because it does not need to ACK directly from C to A. Finally, Figure 5(d) shows three distinct overhearing paths from A to E: $A \rightarrow B \rightarrow E, A \rightarrow D \rightarrow E$, and $A \rightarrow C \rightarrow D \rightarrow E$.

Figure 6(a) plots the expected number of transmissions for all-pairs paths of length greater than one. We omit the one-hop paths for clarity, although we note that the savings is non-zero due to avoided spurious retransmissions. Without RTS-id, each path requires at least as many transmissions as there are hops, sometimes many more due to failed transmissions. RTS-id is able to significantly de-



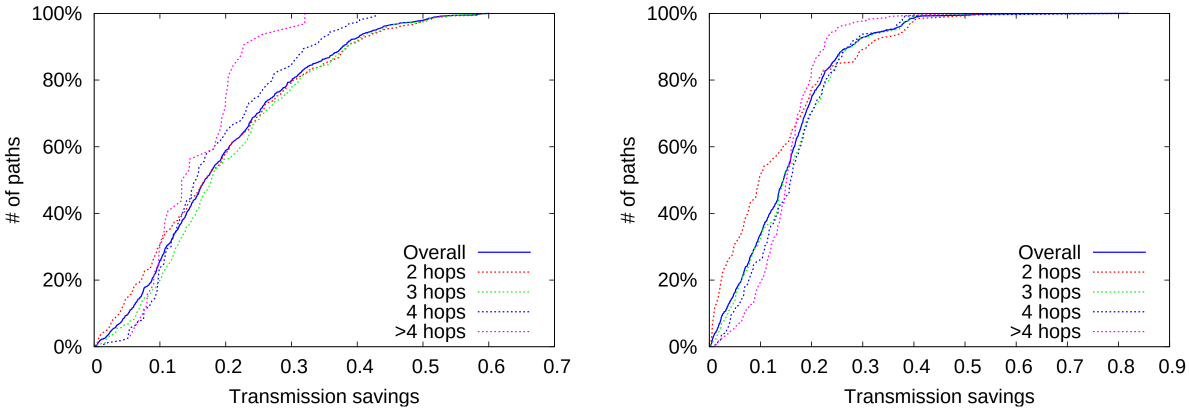
(a) No overhearing (b) Simple overhearing (c) Multi-hop overhearing (d) Complex overhearing

Figure 5: Actual RTS-id scenarios from the Roofnet dataset. Self-loops represent complete packet loss events. All probabilities are based upon a 1-Mbps transmission rate.



(a) 1 Mbps (b) 11 Mbps

Figure 6: The expected number of packet transmissions per ETT path with and without RTS-id.



(a) 1 Mbps (b) 11 Mbps

Figure 7: RTS-id performance improvement versus ETT on the Roofnet dataset. The graphs plot the CDF of the fraction of transmissions saved per path for 1 and 11 Mbps transmission rates.

crease the number of transmissions required, often quite dramatically. To clearly illustrate the performance improvement of RTS-id, Figure 7 plots both the relative performance improvement for various path lengths at 1 Mbps and 11 Mbps. At 1 Mbps, RTS-id is able to save over 20% of path transmissions for the median path, and more than 40% (i.e., turn a 5-hop path into a 3-hop path) for over 10% of the paths. Due to the restricted overhearing at 11 Mbps, however, RTS-id provides at least 20% savings for only a quarter of all paths.

6.2.1 Rate adaptation

As the previous results showed, RTS-id is more effective with lower transmission rates that can reach more nodes. Choosing transmission rates on a per-node basis is complex: higher rates have smaller reception distances, and so may require more hops through the ad hoc network. Here, we model Roofnet's "SampleRate" technique for rate selection [4, 5]. For each link, SampleRate selects the bit-rate with the lowest instantaneous ETT metric. While Roofnet can adjust transmission rates on a per-packet basis, it constructs routes using long-term averages. Hence, we compute an ETT-based path for each source/destination pair as before, except that each hop uses the bit-rate selected by SampleRate. The resulting routes approximate those used by the current version of Roofnet except that we again use the 1500-byte 1-Mbps loss rate for the return channel.

Figure 8 plots both the overhearing prevalence (c.f. Figure 4) and the relative performance improvement versus ETT (c.f. Figure 7) with dynamic rate adaptation. It turns out that most links in our dataset select the 11 Mbps transmit rate, so the overhearing is closer to that observed with a constant 11-Mbps transmit rate than a 1-Mbps transmit rate, resulting in similar savings.[3] In particular, RTS-id provides more than 20% savings for one quarter of all routes, and over 35% savings for the most-improved 5%.

6.2.2 Actual traffic patterns

So far, we have considered all source/destination pairs, which is reasonable for many mesh networks. Some mesh networks (e.g., Roofnet), however, rarely route traffic between internal nodes; instead, they forward traffic to and from a few gateway nodes that transfer packets to the Internet. To confirm that our results are not biased by poorly-performing internal routes, and, instead, are representative of the paths traversed by actual traffic, we restrict ourselves to only those paths connecting each Roofnet node to each of the four Roofnet gateway nodes.

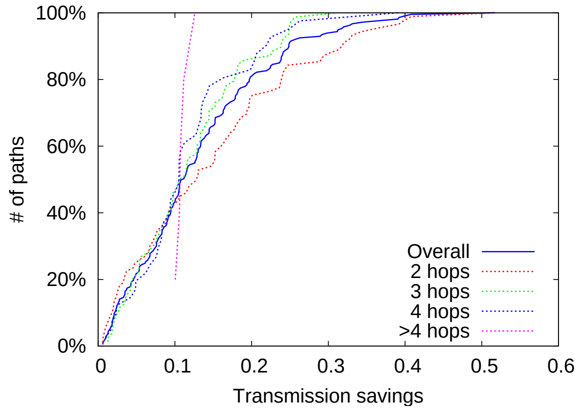


Figure 9: The relative performance improvement versus ETT for paths leading to or from a Roofnet gateway.

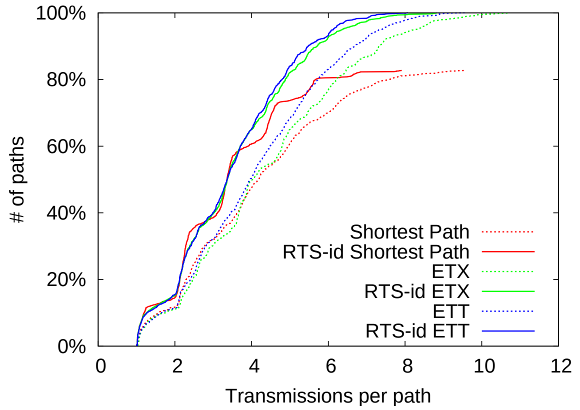


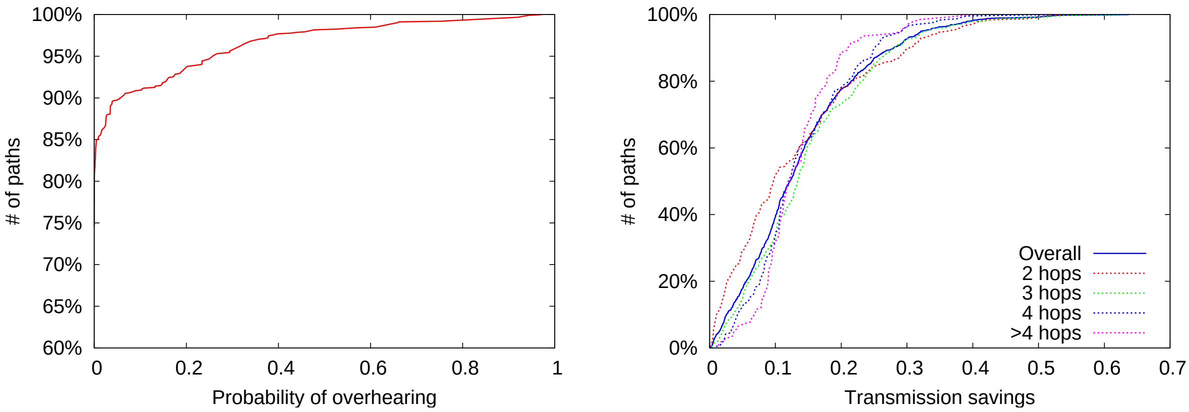
Figure 10: A variety of routing protocols with and without RTS-id, all using SampleRate to select link transmission rates.

Because we do not have a traffic matrix, we consider paths to all four gateways from every node, although only one of them is likely used at any point in time. Figure 9 shows the same data as Figure 8(b), except that it contains only gateway routes. The overall distribution of savings is roughly unchanged.

6.3 Improving other routing protocols

In general, RTS-id improves the performance of routing *more* if those routing protocols do not select routes optimally. Our evaluation of RTS-id using ETT (currently the best-performing routing protocol available for mesh-based networks) gives ETT a large advantage, assuming that ETT has perfect knowledge of link loss rates and that those loss rates are stationary. Our ETT routes are computed as the optimal value over the entire 90-second measurement. In practice, however, networks cannot devote all of their resources to measurement.

[3] Interestingly, its designers note that Roofnet generally transmits at 5.5 Mbps in practice [5], so we are likely understating the potential savings.



(a) Overhearing frequency

(b) Relative performance improvement

Figure 8: The impact of rate adaptation. The first graph shows the overhearing prevalence (c.f. Figure 4), and the second shows the relative performance improvement versus ETT.

For example, the Roofnet network computes its metrics using only 10 measurement packets sent every five minutes, leading to less accurate information for route construction. Furthermore, many networks currently operate with much simpler protocols that do not need to collect such fine-grained loss information. Here, we demonstrate that not only does RTS-id substantially improve the performance of these routing protocols, but that RTS-id, operating only on a local per-link basis, raises the performance of other routing protocols above and beyond ETT's performance.

Figure 10 shows the performance of three routing protocols, ETT (c.f. Figure 8), ETX, and shortest path, where shortest path simply selects the path between source and destination with fewest hops, assuming the link delivery rate is above 80%. (80% is arbitrary, and results are similar for other cut-offs.) Note that not all node pairs are connected by paths consisting entirely of links with greater than 80% delivery rates, so the shortest path algorithm constructs fewer routes. For each routing protocol, we plot the absolute number of expected transmissions per path with and without RTS-id. Note that any routing protocol with RTS-id is generally superior to the best protocol (ETT) without it.

6.4 RTS/CTS overhead

As noted earlier, RTS/CTS is not commonly used in infrastructure deployments (though in some, CTS-to-Self packets are sent for 802.11b/g compatibility). While it was designed for multi-hop scenarios, some mesh networks also eschew its use [5], particularly those with infrequent contention. As in the single AP study, enabling RTS-id in these scenarios also requires an extra RTS/CTS exchange, so we again quantify the transmission time required for all packets in the transmission.

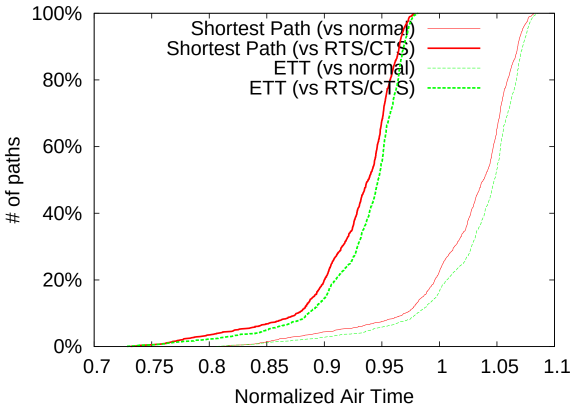


Figure 11: The normalized total path transmission time for RTS-id, with and without RTS/CTS.

We measure this overhead in the Roofnet dataset by examining the path transmission time (the sum of all transmission times along the path). We plot this transmission time normalized against two baselines: a network using no RTS/CTS at all, and a network that already uses RTS/CTS. Note that in this simulation, there is no contending traffic, and so no opportunity for RTS/CTS to provide any benefit. Figure 11 shows the CDF of this normalized transmission time when we do *not* adaptively enable or disable RTS-id and simply leave it enabled on all links. The two lines on the left of the graph show that RTS-id improves transmission times greatly when the network already uses RTS/CTS; the two lines on the right of the graph show the overhead of enabling RTS/CTS and show that in some cases, blindly enabling RTS-id can *reduce* performance over the base network. Some of the paths, however, still benefit from RTS-id, by up to 20%. (The left pair of lines are represent the same data as the ETT and shortest-path lines from Figure 10.)

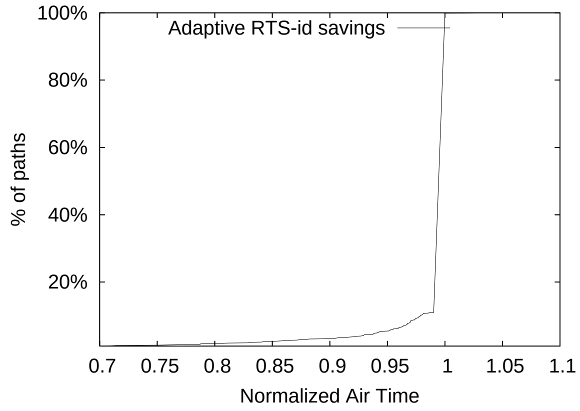


Figure 12: The normalized air time of adaptive RTS-id vs. a network that does not natively use RTS/CTS.

Adaptively enabling RTS-id as described in Section 3.3 avoids the slowdown on links where RTS-id does not provide benefits. To evaluate adaptation, we enable RTS-id only for those link-layer senders who benefit in expectation. Figure 12 shows the fraction of the path transmission time for adaptive RTS-id vs. a network that does not use RTS/CTS at all. The higher overhead of the RTS/CTS exchange means that RTS-id is used on many fewer links than in a network that natively uses RTS/CTS. As a result, its benefits are smaller, but it still provides a 10% reduction in air time for about 5% of the paths, with significantly larger reduction for some paths. Unlike the equivalent lines in Figure 11, adaptive RTS-id never *harms* transmission time.

7 Security implications

The deployment of RTS-id would have a number of potential security implications.

Confidentiality: RTS-id may permit an attacker to perform a rough "traceback" to the source of a packet via timing analysis that determines if a packet was already present in a node's cache. The effects of such an attack seem minor, as the attacker would already have to be well-placed in order to conduct the inquiry.

Integrity: RTS-id introduces no new attacks against integrity, but the availability attacks discussed below might permit an attacker to prevent legitimate packets from reaching their destination, enhancing the effectiveness of existing spoofing attacks.

Availability: If an attacker knows the hash of the packet it wishes to stop and can generate a packet that collides with this hash, then the attacker can transmit this packet *before* the real packet, causing the attack packet to take the place of the original packet. This attack may require less power than a jamming attack, and the packet loss would not be detected at link layer, but only end-to-end. A second attack, with similar effect, is that an attacker can spoof a CTS-ACK response to an RTS-id packet, making the sender believe the packet has been transmitted. All of these attacks require that an attacker be able to transmit packets with very tight timing requirements, which today requires programmable hardware such as the CalRadio. While these attacks are somewhat more effective than jamming, an attacker who can mount them is already well-positioned to jam and snoop.

Per-pair keys could help resist these attacks, but their use would require significant modification to the 802.11 protocol: current encryption mechanisms such as WPA only encrypt the data payload, not the packet header.

8 Future work

Our immediate next step is to extend RTS-id to support longer-duration, cross-flow caching. In particular, we would like to integrate Santos and Wetherall-style packet caching with header compression into RTS-id. While existing header compression techniques can compress TCP/IP headers down to a few bytes, they typically rely on tight sender-receiver synchronization; adapting those techniques to the lossy wireless environment poses an interesting challenge. Such extensions could exploit either long-term caching between different flows, or could use small caches to enable efficient simulcast of data over a wireless mesh network without native multicast support.

Our initial evaluation of RTS-id using the Roofnet data leaves several issues unexplored. For instance, how might the performance of RTS-id change in the face of mobility? In particular, the effectiveness of receiver caches may be impacted as the set of overhearing nodes continually changes. Similarly, senders may adjust their transmission power as nodes move, which may increase the need to adaptively enable RTS-id. 802.11 deployments with high levels of mobility, however, may also require higher densities to ensure pervasive connectivity, which could increase overhearing opportunities.

Additionally, our current deployment is restricted to 802.11b. The availability of additional speeds in 802.11g and 802.11a may affect overhearing depending on senders' rate adjustment algorithms. Moreover, it could be possible to improve the performance of rate adaptation algorithms by integrating information from RTS-id. In particular, it may be beneficial to transmit at a lower rate with significantly higher overhearing; conversely, a sender may not want to decrease its send rate even in the face of significant link-layer loss if overhearing is able to compensate for a large enough portion of the losses. We hope to explore these issues with increasing capability and availability of CalRadio.

RTS-id need not operate independent of other advances that leverage wireless broadcast. Like the batching in ExOR, RTS-id might be able to operate more efficiently if it could batch queries *when multiple packets are in its queue*, without increasing end-to-end latency. We believe that there are also interesting possibilities for merging RTS-id's opportunistic overhearing benefits with the exploitation of omni-directional reception by network coding techniques.

9 Conclusion

RTS-id is a simple, backwards-compatible, link layer mechanism for eliminating redundant packet transmissions in wireless networks. Its goal is to turn broadcast reception from a handicap into an advantage, and our evaluation shows that it succeeds at this goal. RTS-id can improve performance by 5% to 30% in existing networks. Perhaps the most important facets of RTS-id, however, is that it boosts the performance of simpler ad hoc routing schemes so that they match the performance of more sophisticated (and complex) schemes, and that it optimizes a common case of same-LAN data transmission.

In addition to our simulation results using the Roofnet data, we have implemented and conducted a preliminary evaluation of RTS-id on real, interoperable 802.11 hardware. While necessarily limited by the limited availability of this hardware and its relative immaturity, our implementation shows that RTS-id can be practically implemented to meet the timing constraints of 802.11 hardware and can reduce redundant packet transmissions in a real-world setting.

Acknowledgments

The authors wish to thank Jeff Pang and Yu-Chung Cheng for comments on earlier drafts. This work is supported in part by Ericsson Research and Qualcomm through the UCSD Center for Networked Systems (CNS) and the UC Discovery program, and by NSF award CNS-0546551.

References

[1] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris. Link-level measurements from an 802.11b mesh network. In *Proc. ACM SIGCOMM*, Portland, OR, Aug. 2004.

[2] A. Armon and H. Levy. Cache satellite distribution systems: Modeling, analysis, and efficient operation. *IEEE JSAC*, 22(2), Feb. 2004.

[3] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang. MACAW: A Media-Access Protocol for Packet Radio. In *Proc. ACM SIGCOMM*, London, England, Aug. 1994.

[4] J. Bicket. Bit-rate selection in wireless networks. Master's thesis, Massachusetts Institute of Technology, Feb. 2005.

[5] J. Bicket, D. Aguayo, S. Biswas, and R. Morris. Architecture and evaluation of an unplanned 802.11b mesh network. In *Proc. ACM Mobicom*, Cologne, Germany, Sept. 2005.

[6] S. Biswas and R. Morris. Opportunistic routing in multihop wireless networks. In *Proc. 2nd ACM Workshop on Hot Topics in Networks (Hotnets-II)*, Cambridge, MA, Nov. 2003.

[7] S. Biswas and R. Morris. ExOR: opportunistic multi-hop routing for wireless networks. In *Proc. ACM SIGCOMM*, Philadelphia, PA, Aug. 2005.

[8] S. Z. Biswas. Opportunistic routing in multi-hop wireless networks. Master's thesis, Massachusetts Institute of Technology, Mar. 2005.

[9] California Institute of Telecommunications and Information Technology (CalIT2). CalRadio 1.0. `http://calradio.calit2.net/calradio1.htm`.

[10] S. Chachulski, M. Jennings, S. Katti, and D. Katabi. Trading structure for randomness in wireless opportunistic routing. In *Proc. ACM SIGCOMM*, Kyoto, Japan, Aug. 2007.

[11] R. R. Choudhury and N. Vaidya. MAC layer anycasting in wireless networks. In *Proc. 2nd ACM Workshop on Hot Topics in Networks (Hotnets-II)*, Cambridge, MA, Nov. 2003.

[12] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *Proc. ACM Mobicom*, San Diego, CA, Sept. 2003.

[13] R. Draves, J. Padhye, and B. Zill. Comparison of routing metrics for static multi-hop wireless networks. In *Proc. ACM SIGCOMM*, Portland, OR, Aug. 2004.

[14] J. Jubin and J. Tarnow. The DARPA Packet Radio Network Protocols. *Proc. of the IEEE*, 75(1):21–32, Jan. 1987.

[15] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Mèdard, and J. Crowcroft. XORs in the air: practical wireless network coding. In *Proc. ACM SIGCOMM*, pages 243–254, Pisa, Italy, Aug. 2006.

[16] P. Larsson. Selection diversity forwarding in a multihop packet radio network with fading channel and capture. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(4):47–54, 2001.

[17] J. Li, C. Blake, D. S. J. De Couto, H. I. Lee, and R. Morris. Capacity of ad hoc wireless networks. In *Proc. ACM Mobicom*, pages 61–69, Rome, Italy, July 2001.

[18] B. Raman and K. Chebrolu. Revisiting MAC design for an 802.11-based mesh network. In *Proc. 3nd ACM Workshop on Hot Topics in Networks (Hotnets-III)*, San Diego, CA, Nov. 2004.

[19] J. Santos and D. Wetherall. Increasing effective link bandwidth by suppressing replicated data. In *Proc. USENIX Annual Technical Conference*, New Orleans, LA, June 1998.

[20] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, B. Schwartz, S. T. Kent, and W. T. Strayer. Single-packet IP traceback. *IEEE/ACM Transactions on Networking*, 10(6), Dec. 2002.

[21] N. T. Spring and D. Wetherall. A protocol-independent technique for eliminating redundant network traffic. In *Proc. ACM SIGCOMM*, Stockholm, Sweden, Sept. 2000.