UNIVERSITY OF CALIFORNIA, SAN DIEGO

**Overhearing in 802.11 Mesh Networks**

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Computer Science

by

Mikhail Afanasyev

Committee in charge:

Professor Alex C. Snoeren, Chair
Professor Rajesh Gupta
Professor Tara Javidi
Professor Stefan Savage
Professor Geoffrey M. Voelker

2009

The dissertation of Mikhail Afanasyev is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____

_____

Chair

University of California, San Diego

2009

DEDICATION

To my wife, Raisa Karasik.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

Most of all, I am greatly indebted to my adviser, Alex Snoeren. Our frequent interaction over my graduate career provided me with direction and guided me to the results that I was able to produce. His advice was very valuable, inspired and motivated me when I had difficulties with my research. He also taught me how to interpret results and set up meaningful experiments. I would like to thank him for many editing rounds of this dissertation – even when he was very busy, he still found time to give feedback.

I also would like to thank Professors Stefan Savage and Geoff Voelker. Their advice and attention during our weekly group meetings provided valuable insights and perspective to my projects.

I would also like to thank the wonderful people in the wireless group. I was able to formulate my thoughts and develop my ideas during many discussions with Patrick Verkaik and Danny Turner. Yu-Chung Cheng taught me a lot about wireless monitoring and development.

I would like to give special thanks to Tsu-wei Chen from Google, Inc, for an opportunity to work with city-wide networks. He was very helpful, and it was because of him that I was able to write my IMC 2008 paper.

In addition, I would like to thank all the people in the department who made me feel at home and kept me company during our electronic projects: Kirill Levchenko, Michael Vrable, and John McCullough. Special thanks goes to Brian Kantor — he was always there to offer a cup of tea and listen to my stories.

Finally, I would like to thank my wife, Raisa Karasik, for the motivation and support which allowed me to complete my dissertation in such a short interval of time.

Chapters 3 and 4, in part, are a reprint of the material as it appears in the proceedings of the ACM/USENIX Symposium on Networked Systems Design and Implementation, 2008, Afanasyev, Mikhail; Andersen, David G.; Snoeren, Alex C.. The dissertation author was the primary investigator and author of this paper.

Chapters 3, 5 and 6, in part, are preprints of material that has been accepted for publication in the proceedings of the ACM SIGCOMM Conference on Internet Measurement, 2009, Afanasyev, Mikhail; Snoeren, Alex C. The dissertation author was the primary investigator and author of this paper.

Chapter 3, in part, is a reprint of the material as it appears in the proceedings for the ACM SIGCOMM Conference on Internet Measurement, 2008, Afanasyev, Mikhail; Chen, Tsuwei; Voelker, Geoffrey M.; Snoeren, Alex C. The dissertation author was the primary investigator and author of this paper.

## VITA

| | |
|---|---|
| 2004 | Bachelor of Science in Engineering<br>University of California, Berkeley<br>Berkeley, CA, USA |
| 2007 | Master of Science in Computer Science<br>University of California, San Diego<br>San Diego, CA, USA |
| 2009 | Doctor of Philosophy in Computer Science<br>University of California, San Diego<br>San Diego, CA, USA |

## PUBLICATIONS

"The Importance of Being Overheard: Throughput Gains in Wireless Mesh Networks."
Mikhail Afanasyev and Alex C. Snoeren.
Accepted to the *9th ACM SIGCOMM Conference on Internet Measurement (IMC 09)*, Chicago, IL, November 2009

"Usage Patterns in an Urban WiFi Network."
Mikhail Afanasyev, Tsuwei Chen, Geoffrey M. Voelker, and Alex C. Snoeren
In review for the *IEEE/ACM Transactions on Networking*.

"Analysis of a Mixed-Use Urban WiFi Network: When Metropolitan becomes Neapolitan."
Mikhail Afanasyev, Tsuwei Chen, Geoffrey M. Voelker, and Alex C. Snoeren.
*Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement (IMC 08)*, Vouliagmeni, Greece, October 2008

"Efficiency through Eavesdropping: Link-layer Packet Caching."
Mikhail Afanasyev, David G. Andersen, and Alex C. Snoeren
*Proceedings of the 5th ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI 08)*, San Francisco, CA, April 2008.

"Automating Cross-Layer Diagnosis of Enterprise Wireless Networks."
Yu-Chung Cheng, Mikhail Afanasyev, Patrick Verkaik, Peter Benko, Jennifer Chiang, Alex C. Snoeren, Geoffrey M. Voelker, and Stefan Savage.
*Proceedings of the ACM SIGCOMM Conference (SIGCOMM 07)*, Kyoto, Japan, August 2007

ABSTRACT OF THE DISSERTATION

**Overhearing in 802.11 Mesh Networks**

by

Mikhail Afanasyev

Doctor of Philosophy in Computer Science

University of California, San Diego, 2009

Professor Alex C. Snoeren, Chair

802.11-based mesh networks provide a useful and practical alternative to regular infrastructure-based wireless networks, but they have an intrinsic scaling limit due to their less efficient airtime utilization. Mesh networks forward packets multiple times, increasing airtime utilization and decreasing path throughput and useful channel capacity available to the clients. In this dissertation, we explore ways to optimize forwarding in order to decrease the number of packet transmissions and increase path throughput.

The main technique that we explore in this dissertation is a phenomenon called 'overhearing'. Traditional mesh networks use only 'good' links with low packet loss rates in order to forward packets; overhearing allows utilization of the links with high losses in order to reduce the number of transmissions where possible. This dissertation proposes two methods that allow mesh networks to take full advantage of overhearing: 'RTS-id' is a backwards-compatible link-layer modification that allows adding overhearing support to traditional mesh networks without requiring changes to hardware or transport protocols. 'Modrate' is a new rate selection algorithm that can increase the amount of overhearing in bulk transfer systems that are already taking advantage of overhearing opportunities.

In order to verify the operation of RTS-id, we implement the algorithm on a

software-defined radio. We verify that RTS-id is compatible with existing, unmodified radios. We then develop a probabilistic transmission simulator and use it to quantify the potential gains from deploying RTS-id on existing large-scale wireless mesh networks.

In order to verify the operation of modrate, we set up two wireless testbeds: a large, building-wide testbed operating in the 2.4-GHz range, and a smaller testbed operating in the 5-GHz range. We apply modrate to two existing overhearing-aware routing protocols, ExOR and MORE, and use our testbeds to measure the improvement provided by modrate in those systems.

Finally, motivated by the somewhat unimpressive performance of modrate, we study the specific reasons for performance improvements in the ExOR and MORE protocols. We measure the performance of each protocol with various pieces of functionality disabled, and come to surprising conclusions: while systems such as ExOR and MORE have significantly better performance than traditional systems, a large fraction of these performance gains is caused not by overhearing, but by simpler protocol aspects like flow control and group acknowledgments.

# Chapter 1

# Introduction

Wireless networks based on the 802.11 family of standards have become ubiquitous. Most of the computers sold today have a built-in wireless interface, and wireless Internet access is available in many places. Existing wireless networks span a wide range of sizes: from small networks that share Internet within a single apartment to very large networks that cover entire cities.

Most existing networks run in so-called 'infrastructure mode'. This mode requires a special dedicated device, known as an 'access point', which is connected to the Internet via a wired link. Each access point has a finite range, and therefore provides wireless Internet access to a limited area. If a larger area needs to be covered, it is possible to install multiple access points, each with its own wired Internet uplink. This procedure is trivial for small (apartment-size) networks, but gets complicated quickly as network size grows. A modest-sized building might have dozens of access points, requiring extensive wiring and network equipment for proper installation.

An alternative approach to infrastructure networks is multi-hop wireless networking, also known as mesh networking. This approach simplifies the deployment of larger wireless networks by forwarding packets from source (client station) to destination (access point) via intermediate stations. This way, only a few stations need to be connected to uplink nodes, and the total amount of fixed wiring is greatly reduced. Multi-hop networks are becoming a popular mechanism for providing Internet access, both in urban areas [BM03] and in rural and developing settings [RC04]. By reducing the need for a fixed wired infrastructure, they offer the hope of providing cheaper connectivity and

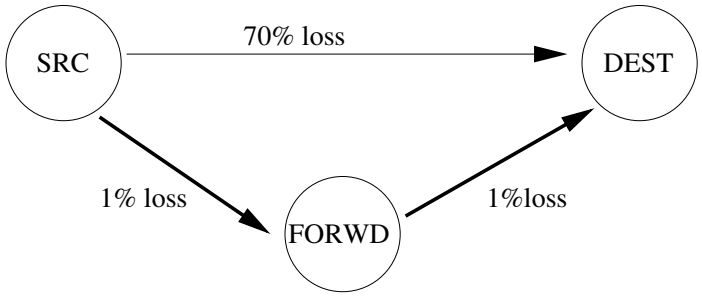



Figure 1.1: A simple overhearing scenario

faster deployment.

One of the biggest disadvantages of multi-hop networks is an intrinsic limitation of raw channel capacity [LBD+01]: physical properties of radio transmissions limit the total transfer rate of the data from all stations. Since multi-hop networks transmit each data packet multiple times, a large fraction of the channel capacity is used for forwarding packets between the nodes, further reducing the maximum data throughput available to each client. 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 limits the channel capacity, and together with requirement that nodes forward traffic on behalf of one another, constitutes one of the primary scaling limitations of multi-hop wireless networks. In this dissertation, we explore the ways to optimize forwarding in large mesh networks in order to increase path throughput.

## 1.1 Overhearing

The main technique that we explore in this dissertation is to leverage phenomenon called overhearing. A number of research projects attempt to turn the broadcast nature of wireless networks into an advantage, instead of considering it purely an interference-causing liability. Traditional multi-hop wireless networks use only 'good' links with low loss rates for data transmission; however, frequently there are other links with higher loss rates which could be used to transfer some of the packets. Thus, it is possible that a node closer to the destination than the intended forwarder will *overhear* a packet transmission before the forwarder node has sent the packet.

1.1 shows a trivial example of overhearing. There is a multi-hop wireless network consisting of three nodes: source (SRC), forwarder (FORWD) and destination (DEST). When the source transmits a packet, there is a 99% chance that the forwarder will receive it, but there is also 30% chance that the destination will receive it directly. A traditional mesh network will always forward the packet through the FORWD node. Assuming that each node retries the transmission until the packet is acknowledged, and that acknowledgment are not lost, it will take on average 1.01 transmissions to get packet from SRC to FORWD, and 1.01 transmissions to eventually deliver every packet from FORWD to DEST, for 2.02 transmissions total. However, 30% of packets from SRC are overheard by DEST, and thus do not have to be forwarded. An overhearing-aware algorithm could take $0.30 + 0.70 * 2.02 = 1.7$ transmissions in expectation, an improvement of 15%.

There have been a number of proposed approaches to take advantage of overhearing: by caching opportunistically overheard objects (e.g., in satellite-based distribution systems [AL04]); by modifying ad hoc routing protocols to enable them to acknowledge packets received later in the forwarding chain (ExOR[BM05]); by using network coding on bi-directional traffic streams [KRH$^+$06]; and, most recently, by using network coding to achieve similar benefits without explicit coordination (MORE[CJKK07]). In this dissertation, we approach the problem from two directions: first, we consider the ways to add overhearing to existing systems, and second, we explore ways to increase overhearing in systems that already take advantage of it. Our thesis is that it is possible to further improve the performance of mesh networks, including large-scale wireless networks, by taking full advantage of overhearing.

## 1.2 Contributions

This dissertation has four main contributions. First, we perform a detailed study of existing overhearing systems on multiple testbeds, in both simulation and experiment, to discover the similarities between the different kinds of networks. Second, we present 'RTS-id', a new method to take advantage of overhearing which is designed to be applied to non-overhearing-aware systems, requires few modifications to 802.11 nodes, and

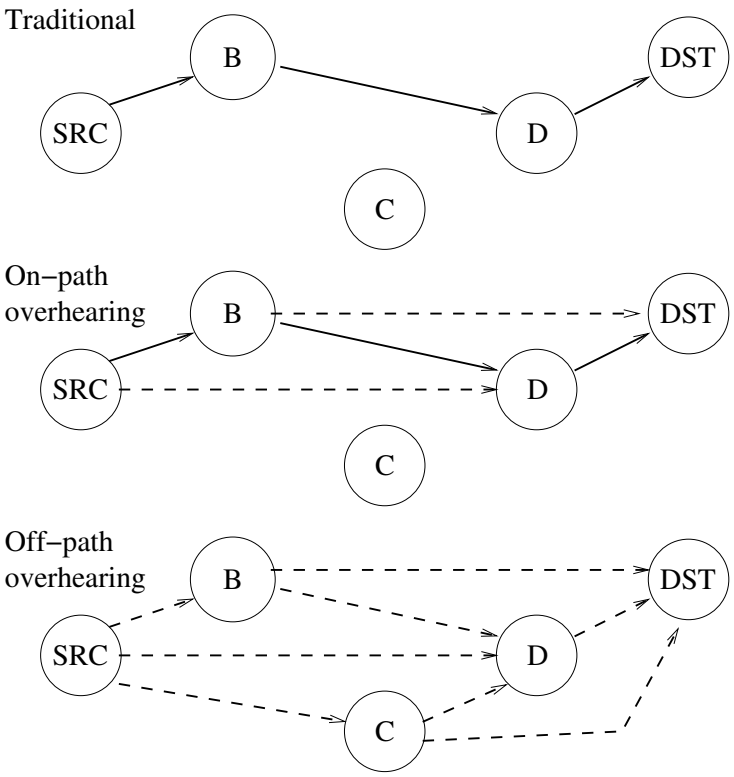



Figure 1.2: Types of overhearing. Solid lines represent primary links, while dotted lines represent overhearing links

is fully backwards compatible. Third, we present 'modrate', a way to improve the performance of existing overhearing-aware systems by increasing the total amount of overhearing. Finally, we closely examine the performance of overhearing-aware systems which show great improvements over non-overhearing-aware systems, and discover that the true cause of those performance improvements is not overhearing, but rather general protocol modifications.

### 1.2.1 Study of previous solutions

There has been a number of research projects that study overhearing, but the experimental verification was limited. Each paper typically run the experiments on a single testbed, at a single channel, power setting and frequency. Moreover, each paper uses its own testbed, so a direct comparisons of results was not possible.

To facilitate head-to-head comparition, we set up two test-beds: a large, building-

wide testbed, and a smaller, 10-node testbed. By varying the power level and by taking a subset of nodes, it is possible to simulate a variety of the real-world conditions. Furthermore, we develop a simulator for existing wireless mesh protocols and study the amount of overhearing under different experimental conditions.

In order to illustrate the various types of overhearing, we show a sample network 1.2. The first part of this picture shows a path that a packet routed using a traditional routing algorithm would take.

### 1.2.2 On-path overhearing

The simplest way to take advantage of overhearing is to introduce overhearing support to a traditional mesh routing algorithm. This is illustrated in a second picture in 1.2. In this case, when overhearing is not present, the packets are routed using existing mesh algorithms, and if the packet is overheard, then some of the transmissions are not made. For example, if a packet sent from *B* to node *D* was overheard by *DST*, then *D* does not need to send packet to *DST*, thus saving a transmission and increasing overall performance. This conservative optimization routes packets using existing paths, and we call this type of overhearing 'on-path overhearing'.

We introduce 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 requiring only the cooperation of adjacent nodes in a path. We implement and test this modification using a software-defined radio. Additionally, we use simulation to estimate the speed gain on a larger scale.

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 inter-operate with legacy nodes. We verify that the RTS-id extensions are ignored by hardware that does not support it with no ill effects. While substantially more modest than the bulk transfer improvements demonstrated by overhearing-aware bulk transfer systems, the system has a number of advantages in its compatibility: it works well with TCP and UDP protocols, thus it does not require application rewrites; and it could be deployed incrementally, with nodes automatically taking advantage of the system as it becomes available.

### 1.2.3 Off-path overhearing

Some of the more advanced networks take advantage of the overhearing in a more complex way – instead of using a traditional mesh protocol as a base, they completely change the routing structure. Instead of being confined to a traditional paths, packets have a large number of alternative paths, each packet chooses the most optimal route. Since there is no longer an single optimal path for the packet, we call this kind of overhearing 'off-path overhearing'.

Off-path overhearing works by changing the core routing idea. In traditional routing, there is a fixed forwarder list, with each node having a 'next' node which is an expected recipient of each packet sent. In off-path overhearing systems, each node has a metric which indicates closeness to the source. When a node transmits a packet, all nodes listen, and any recipient which is closed to destination than transmitter may forward a packet. An example of this kind of overhearing is illustrated in third part of 1.2: each packet is forwarded by the station closest to the *DST*. For instance, if *SRC* sends a packet which is received by *B* and *C*, then station *C* will forward it. If this transmission is not be received by *DST*, then some other station that heard it, such as *D*, will forward it again. Off-path overhearing is used by protocols such as ExOR and MORE.

An important part of the routing process is selection of 'bitrate', or transmission rate – the packets which are sent at slower rate are more likely to be received by far-away stations, but they take a longer time to transmit. The problem of rate selection has been surprisingly little studied. We introduce 'modrate' – a new rate-selection algorithm that seeks to jointly optimize individual link bitrate selections with network-wide overhearing opportunities. In particular, as opposed to selecting bitrates in a link-local fashion based only upon a packet's next hop, modrate selects the bitrate that minimizes a packet's expected number of transmissions along a path to its eventual destination assuming that any overhearing can be profitably exploited. We integrate modrate with ExOR and MORE protocols, and deploy it on our testbeds. We demonstrate that modrate provides modest, but noticeable improvement to those networks.

### 1.2.4 Extensive study of overhearing

Our testbeds present ample opportunities for overhearing, and its prevalence varies noticeably with the particular bitrates employed. We therefore expected modrate to provide significant further throughput improvement on ExOR and MORE. While modrate is able to increase performance in some instances, the boost is surprisingly modest in many cases. A detailed evaluation of the cause leads to the final contribution of this dissertation: we show that while proposed opportunistic algorithms—ExOR and MORE in particular—can provide tremendous performance improvement, in our environment at least, the vast majority of their gains come *not* from leveraging overhearing, but instead from a number of other substantial changes to the transfer protocols in their implementations.

Motivated by this observation we present a careful analysis of a spectrum of potential protocols on our testbeds, starting with Srcr, a state-of-the art traditional routing protocol that does not leverage overhearing [BABM05], and incrementally applying changes to arrive at ExOR with modrate. Previous studies have compared only two points in this spectrum, typically traditional routing and their proposed protocol. By considering each modification individually, we discover that in many circumstances ExOR gains more from the relatively prosaic step of eliminating individual per-packet acknowledgments than from taking advantage of overhearing. This discrepancy is especially pronounced in networks with lossy links.

While considerable work remains to be done to determine the generality of our findings, we believe the results may have significant implications. In particular, many researchers—ourselves included—may have overestimated the ability of existing systems to effectively exploit overhearing in mesh networks. Conversely, it appears significant gains can be extracted from far more banal protocol changes.

## 1.3 Organization of the dissertation

This dissertation is organized as follows. In Chapter 2, we provide an introduction to relevant technologies: an overview of parts of the 802.11 standard important to overhearing, and various routing protocols used in wireless networks. We present the

wireless networks used in our study in detail in Chapter 3, including our simulator and the testbeds that serve as a source of input data for simulator. Chapter 4 describes RTS-id, a new way to take advantage of overhearing in regular wireless networks. Chapter 5 describes 'modrate', an improvement to wireless networks that are already taking advantage of overhearing. Chapter 6 provides important insights, confirmed by simulations and experiments, on the importance of overhearing in the wireless mesh protocols. Finally, Chapter 7 summarizes the conclusions of this dissertation.

# Chapter 2

# Background

This chapter covers background information about the technologies discussed in the rest of the dissertation. It provides a basic overview of the 802.11 wireless protocol (WiFi): physical layer, media access layer, as well as management procedure. We also provide an overview of existing mesh routing protocols.

## 2.1 The 802.11 wireless protocols

The 802.11 standard defines a family of WLAN (wireless local area network) protocols, designed for the transmission of data in the medium range (about 20 meters). It operates in the 2.4-GHz or 5-GHz unlicensed band, and is extremely widely used today.

The most popular frequency band today is 2.4-GHz. Protocols which operate in this range are described in 802.11g, and its predecessor, 802.11b. The available bandwidth is split into 5-MHz wide channels, and the transmissions are 22-MHz wide. Thus, two stations transmitting on frequencies less than 5 channels apart will interfere with each other. The number of available channels varies by country. In the US, there are 11 available channels, and only 3 orthogonal ones. The small number of available orthogonal channels has a profound impact on the operation and design of wireless networks: any technology that uses wireless transmissions has to share the channel with other devices, and the overall channel capacity limits the maximum attainable bandwidth.

The alternative band, which is used by the 802.11a standard, is 5-GHz. The amount of allocated bandwidth is much larger than in the 2.4-GHz range, and, thus, the

5-GHz band has up to 12 independent channels (8 in US). However, this band has its own disadvantages. The higher frequency is more easily absorbed by solid objects, such as walls and doors, and, as a result, the indoor range of 5-GHz devices is smaller than 2.4-GHz devices.

The upcoming 802.11n standard improves on 802.11a and 802.11g to add features such as MIMO antennae for better interference reduction, and channel bonding for higher bandwidth. It has not been officially released yet, but a number of devices based on a draft of the standard are already available. While 802.11n is certainly an interesting research topic, we do not study it in this dissertation.

### 2.1.1 Physical layer

Historically, the 802.11 standard and all of its additions have described four modulation standards: FHSS (Frequency-hopping spread spectrum), infrared transmission, DSSS (directly sequence spread spectrum), and OFDM (orthogonal frequency division multiplexing). However, the first two are present only in the very first version of the protocol, and have been subsequently depreciated and removed. In the next section, we describe the two protocols used today: DSSS and OFDM.

#### Direct Sequence Spread Spectrum

Direct Sequence Spread Spectrum (DSSS) is the original modulation standard, initially defined in the 802.11 document in 1997, with its final definition published in 802.11b at the end of 1999. It allows for four modulation rates: 1, 2, 5.5 and 11 Mbps (MBit/second), and uses the 2.4-GHz band. It has since been superseded by more modern standards such as 802.11g, but it has seen wide adoption and plays important role in existing networking research: papers published as late as 2007 use 802.11b-only testbeds for verification [CJKK07].

An 802.11b packet consists of multiple parts (see 2.1). It starts with a ‘preamble’, a bit pattern which is used to wake up the receiver and synchronize the decoder. The preamble is followed by a header, which is always transmitted at the lowest speed, 1 Mbps, and the contains packet duration and modulation rate for the rest of the packet. After the header, there is a small pause to allow the receiver and transmitter to change

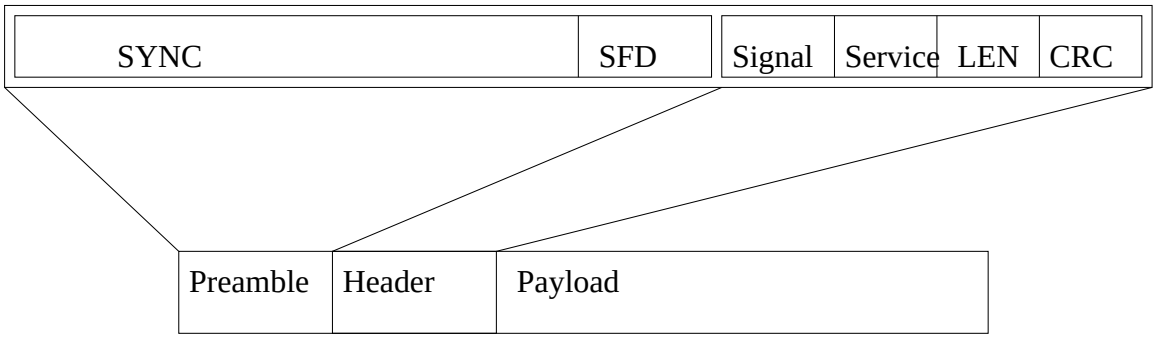



Figure 2.1: 802.11b packet format

modulation rates to the new setting, followed by the payload, encoded at the specified data rate.

The physical modulation used depends upon the bit rate. The lower rates (1 and 2 Mbps) transmit data at 1 or 2 bits per symbol, using Barker encoding and binary phase shift keying (BPSK)/quadrature phase-shift keying (QPSK) sequencing. The higher data rates use QPSK encoding directly in order to transmit 4 or 8 bits per symbol. There is a per-packet checksum, but no inter-symbol error correction, so a burst of interference which is longer than a symbol will cause bit errors and packet loss.

**Orthogonal Frequency Division Multiplexing**

Orthogonal Frequency Division Multiplexing (OFDM) is the newer modulation scheme. It was originally introduced for use in the 5-GHz band in 802.11a standard in 1999, but adoption was slow. However, in 2003, 802.11g standard introduced OFDM to the 2.4-GHz band. Adoption of 802.11g has been quite rapid, and today the majority of devices are 802.11g-compatible. OFDM modulation used in 802.11a and 802.11g standards provides rates of 6, 9, 12, 18, 24, 36, 48 and 54 Mbps.

Similarly to the DSSS 802.11b case, an 802.11a packet in the 5-GHz band also contains a preamble, header and payload. However, the header is transmitted using OFDM. The preamble is thus transmitted using OFDM, and is used to lock in the frequency and set up gain control on the receiver. The header contains the same information that the DSSS header contains, and is sent at the lowest speed, 6 Mbps.

In the 2.4-GHz band, the situation is more complicated. When there are no

Table 2.1: 802.11a modulation parameters

| Rate (Mbps) | Data bits per symbol | Coding rate | Coded bits per symbol | Subcarrier modulation |
|---|---|---|---|---|
| 6 | 24 | 1/2 | 48 | BPSK |
| 9 | 36 | 3/4 | 48 | BPSK |
| 12 | 48 | 1/2 | 96 | QPSK |
| 18 | 72 | 3/4 | 96 | QPSK |
| 24 | 96 | 1/2 | 192 | 16-QAM |
| 36 | 144 | 3/4 | 192 | 16-QAM |
| 48 | 192 | 2/3 | 288 | 64-QAM |
| 54 | 216 | 3/4 | 288 | 64-QAM |

802.11b devices around, the OFDM preamble and header from 5-GHz band are used. However, if there are 802.11b devices present, they can cause compatibility problems. OFDM spreads the power much more evenly over the available bandwidth, and therefore the carrier detect function of DSSS might not detect the presence of OFDM signal. In order to prevent that, the 802.11g standard supports an '802.11b protection mode', in which the preamble and header are sent at 1 Mbps using Barker encoding. This way, old 802.11b-compatible stations decode the header and reserve the channel for the specified duration. Protection mode is required when there are any old stations present, but it adds significantly to the per-packet overhead — when it is enabled, the packet must contain two separate preambles: an 802.11b-compatible preamble for headers, and OFDM preamble for the payload.

802.11a/g transmits using large symbols, from 24 to 216 data bits. Each symbol is transmitted using 52 separate subcarriers. The data bits are scrambled, convolutionally coded, and sent as Quadrature Amplitude Modulation (QAM) with 2 to 64 constellations. As a result, the proper symbol value can be recovered even if a significant number of subcarriers is unreadable. There is no inter-symbol checksum, but the symbols are much longer then 802.11b symbols, and, as a result, a short interference bursts which might have completely damaged 802.11b packet, can be survived by 802.11g packet. 2.1 summarizes the rates and coding overhead in 802.11a and 802.11g.

An important property of 802.11a/g modulation, the one we leverage in this work, is the wide variety of modulation methods and coding rates.

### 2.1.2 Media access control (MAC)

Wireless transmissions use a shared channel, and if two stations transmit simultaneously, they might interfere with each other, resulting in damage to both packets. In order to prevent that, a media access control (MAC) scheme must be used to mediate access to the channel. 802.11 uses carrier sense, multiple access with collision detection (CSMA/CD). The MAC operates in two steps: first, it senses the power on the channel (carrier sense) to make sure that no one is transmitting. If the channel has been empty for some time, the station transmits a packet. The intended recipient should receive the packet and send an acknowledgment packet (ACK). If the sender receives the acknowledgment, it marks the packet as sent and transmissions ends. If the sender does not receive the acknowledgment, for example because of insufficient signal strength or because of interference, the packet is presumed lost. The sending station backs off for an exponentially increasing time, and re-tries the transmission once the channel has been idle again. If the packet is not transmitted after multiple retries (the exact number is not specified, and is configured by the user), the packet is marked as not sent, and transmission is aborted.

Multicast and broadcast packets do not have a pre-defined destination, and do not use acknowledgments. These packets are transmitted only once, as there is no way for the transmitter to detect if the packet was successfully received by all destinations.

The basic 802.11 MAC, while simple, has a number of important properties. First, for successful reception, the channel must be free at the destination, yet the sender only checks the channel at the source. This scheme causes two potential problems. A ‘hidden terminal’ is a station whose transmissions can be detected by the destination, but not by the source. When a hidden terminal transmits, its transmission does not have enough energy to trigger the carrier sense at the source and defer the transmission, but it does have enough energy to jam the signal at the receiver. This condition has the potential to degrade multi-hop transmissions under some conditions by increasing the packet loss rate.

The converse problem is known as an ‘exposed terminal’. In this case, the exposed station can be heard by the sender, but not by the receiver. Thus, the exposed terminal might cause un-necessary backoff – the sender will detect the channel as being busy,

while a transmission would be possible. Exposed terminals do not typically cause packet loss, and therefore have less impact than hidden terminals.

Finally, the symmetric nature of link-layer transmission (data forward, ACKs in reverse) requires connectivity in both directions on the link. This requirement can cause a problem, and excessive data retransmissions, when there is poor connectivity in one direction only.

### 2.1.3 Avoiding hidden terminals

In order to prevent a problem with hidden terminals, a 802.11 specification defines a specific method which uses two special packets, 'Ready to Send' (RTS) and 'Clear To Send' (CTS). These packets do not contain any data payload, and only have two fields: the address of the recipient and a duration. When a device needs to transmit a packet, it waits for the channel to clear in the usual manner, and then transmits an RTS packet, which contains expected duration of the frame exchange (including both data packet and ACK transmissions). The destination station receives the RTS and replies with CTS, which contains the remaining duration. Once the source receives the CTS, the data transfer proceeds as usual: the source sends data packet, and then waits for acknowledgment packet. All other stations always listen for the RTS/CTS packets. If any of them receive the CTS, then their internal timer will automatically postpone any station transmission until the original transaction is over.

RTS/CTS is effective against hidden terminal problems, but it slows down the transmission by introducing additional overheads. The main overhead is the extra time, attached to every packet, which is required to send RTS and receive CTS. Traditionally, the CTS packet is transmitted at lower speeds in order to maximize the effective coverage area. As a result, the RTS/CTS overhead while transmitting a short packet (such as a TCP acknowledgment) could be over 90% of the total packet duration. A second problem is caused by additional loss rate during RTS or CTS transmission: RTS/CTS losses in highly lossy channel would be frequent, and more retransmissions would be required. Practically, most networks do not see a speed advantage from RTS/CTS exchanges, and therefore keep this method disabled.

### 2.1.4 Rate adaptation

As discussed previously, 802.11 physical layers support multiple bit rates, ranging from 1–11 Mbps for 802.11b, 6–54 Mbps for 802.11a, and 1–54 Mbps for 802.11g. Because channel characteristics vary across space and time, an effective 802.11 sender will periodically reconsider the bitrate it employs. A large number of rate-adaptation techniques have been proposed in the literature [Bic05, HVB01] [KM97, SKSK05, WYLB06] including several [Bic05, KM97] which have been deployed in commercial products. Each seeks the same goal, however: to optimize the goodput of the wireless link between sender and receiver.

Because the basic 802.11 standard does not provide for explicit feedback about channel quality at the receiver, senders are forced to estimate the optimal transmission rate through indirect means. The mechanism first deployed commercially, Auto Rate Fallback (ARF) [KM97], defaults to the highest bitrate and falls back to slower speeds if it fails to receive a link-layer acknowledgment for a transmitted frame. ARF speeds back up after a string of successive successful packet transmissions. Researchers have observed, however, that 802.11's link-layer retransmission mechanism may mask frame losses, causing ARF to over-estimate the optimal bit rate.

As an alternative, Receiver-Based Auto Rate (RBAR) [HVB01] proposes to have the receiver report received channel quality in RTS packets, allowing the sender to dynamically adjust transmission rates according to current channel conditions. This presumes that CTS signal-to-noise ratios are effective predictors of frame-exchange success, however, which Bicket found was not always the case [Bic05]. Instead, he proposes to send periodic probe packets at speeds higher than the one currently employed and keep track of their relative success rates in a protocol he calls SampleRate, which has been widely deployed in the MadWifi driver and employed by follow-on research projects [BABM05, BM05, CJKK07]. Recent results, however, have shown that SampleRate can be too conservative in certain cases; indeed its poor performance has lead to its deprecation within the MadWifi driver. Instead, Starsky *et al.* have proposed combining feedback from the RTS/CTS exchange with loss-rate information gathered at the current rate into a system they call Robust Rate Adaptation Algorithm (RRAA) [WYLB06].

## 2.2 Routing

Routing is essential for the operation of wireless networks. The 802.11 standard describes a management layer, which is used for basic routing. However, the management layer does not cover the case of large wireless networks, either limiting the connectivity to the devices in the range of each other or requiring multiple dedicated stations hard-wired to main router.

A number of research projects attempt to enable large wireless networks without wired links. Those systems usually work by adding a routing layer which can route the packet over the multiple wireless stations before reaching the final destination. They bypass the management layer, and insert themselves between layers 3 (Network) and 4 (Transport). A routing daemon is responsible for processing and forwarding packets.

### 2.2.1 802.11 Management Layer

The 802.11 standard specifies a management layer that allows devices to discover and form networks. The main component of an 802.11 wireless network is the Basic Service Set (BSS), a set of stations which can talk to each other. The BSS ID is a 48-bit value that is stored in the header of each data packet transmitted, and stations ignore packets with BSS values different than their own.

There are two main methods of BSS formation: Independent BSS, also known as ad-hoc networking, and Extended BSS, also known as 'Access Point' (AP) or infrastructure mode.

In ad-hoc mode, all stations have equal status, and a BSS ID is chosen to be the MAC address of one of the stations. A set of management packets is used to set up a network, choose a main station and select a channel. This mode was intended to be used by peer-to-peer communication without any centralized control.

In the 'Access Point' mode, there are special, dedicated stations called access points, and every BSS has exactly one access point. All other stations are clients, and the only possible communications are between clients and access points. In order to support mobile stations, multiple BSS can be united into an Extended Service Set, all sharing the same network name. Management packets are used to authorize and associate a client

with an access point, or switch from one access point to another. It is assumed that all access points are physically connected to the same network via a wired backbone.

Both of those modes require an association procedure, which takes some time, and which often causes problems in the low signal-strength conditions. As a result, some card manufacturers have implemented more basic modes which do not require management packets. One example, implemented by some Atheros cards, is 'Ad-hoc demo' mode, which is very similar to ad-hoc mode, but requires the user to manually specify a channel and BSS ID. The most generic solution is 'monitor mode', in which the whole management layer is bypassed: stations receive every packet on the same channel, and they can send packets with any BSS ID values in packet headers. Originally designed for debugging and troubleshooting, this mode is widely used by custom routing layers.

### 2.2.2 Traditional mesh protocols

The routing problem is significantly more complicated in wireless network as compared to wired ones. Unlike wired networks, wireless networks often have a large number of possible links of varying quality. Due to various radio propagation effects, such as fading, interference, and client mobility, the quality of the links is constantly changing. Thus, a wireless routing algorithm must be able to frequently evaluate link quality, and update and propagate new routing tables based on any changes.

There is a number of routing protocols currently used for wireless transmissions. They can be classified by the way the routes are updated (proactive or reactive), as well as by the way the packets are routed (routing table vs. source routing).

Proactive protocols continuously monitor link quality. The monitoring is usually achieved by continuously sending special probe packets, and monitoring the loss rate of incoming probe packets from neighbors. Sometimes, these probe packets contain routing information. The frequency of such packets presents a tradeoff: a higher frequency provides more precise delivery probabilities, and can react faster to route changes, but it creates more overhead for the network, especially when they are dense.

Alternatively, reactive protocols only maintain routes which are actively in use. The first packet in a flow requires a route discovery step, for example by flooding the network. Once a path to the destination is found, the remaining packets are sent via that

established path.

Another division is the way the packets are routed. In protocols based on routing table, each node maintains a route to every other node (or to every active destination, in case of reactive protocols). When a station needs to send a packet, it consults its routing table for the next-hop information, and then sends the packet there. A next-hop node will consult its own routing table to see where to forward the packet. This model is similar to the operation of wired networks (for example RIP works the same way), but it may suffer from loops due to out-of-date node state. There are multiple ways to prevent loops. For example, the Destination Sequenced Distance Vector (DSDV [PB94]) protocol uses route sequence numbers to prevent loops.

An alternative approach to distance vector algorithms is source-routing. Unlike protocols which use per-hop routing table, protocols which use source routing have to have the complete and up-to-date routing information during each packet transmission. The complete path to the destination is then computed and stored in the packet header, and all that the intermediate node has to do is to forward packet as specified in the header. Source routing algorithms are naturally loop-free, but they require a much bigger state maintained at each node. An example of a source-routed protocol is Srcr [BABM05].

**Routing metrics**

An important step in operation of mesh network protocols is computing routes between two nodes. In general, routing protocols attempt to compute paths that minimize some cost metric. The most natural metric, commonly used in wired networks, is hop count. While straightforward to compute, hop count favors paths consisting of fewer, longer hops, which tend to be less reliable than shorter hops.

Instead, the Roofnet urban mesh network introduced ETX, or expected transmission count, which accounts for the retransmissions that are likely to be required on less-reliable links [CABM03]. This metric has been shown to outperform previous routing metrics [DPZ04]. It is defined as a sum of expected number of transmissions at each link. Assuming that the packet will be deliver with probability $P_{data}$, and the transmission will be retried until the packet is delivered successfully, the expected number of transmissions is simply $1/P_{data}$. If the protocol uses acknowledgment packets, then the

metric needs to account for that, too: assuming that probability that the acknowledgment is delivered is $P_{ACK}$, the metric is defined as $1/P_{data}P_{ACK}$. Thus, only data packets are counted, as they take most of the time in the packet transmission.

Yet, if a particular link employs a lower bitrate which is more likely to succeed, it is also more likely to be included on a path despite other, potentially higher-throughput alternatives. To address this deficiency, Roofnet replaced ETX with ETT, or expected transmission time, that incorporates link rates in addition to retry attempts into the link cost [BABM05]. ETT equals to sum of expected transmission time at each hop, where expected transmission time is simply time to transmit packet once multiplied by expected number of transmissions. Time to transmit once includes all parts required to send a packet – time to check that the channel is free, transmitting data packet itself, transmitting ACK, CTS and RTS as required per protocol.

In the original definition of ETT, $P_{ACK}$ is based upon the measured delivery rate of 60-byte ACK packets transmitted at 1 Mbps. In most surveys, we do not have the specific information about the loss rate for the short packets, and thus we calculate $P_{ACK}$ based on measured delivery rate at 1500 bytes.

### 2.2.3 Multi-path routing

An alternative to traditional, single-path mesh routing systems is multi-path routing. In this case, instead of having a single, pre-defined next hop node, each transmission has a list of potential forwarders, each of them might forward a subset of the packets. This method has a number of advantages: it naturally takes advantage of any overhearing, and it requires a less accurate estimate of the network state.

In particular, both ExOR [BM05] and the more recent MORE [CJKK07] define new, bulk-transfer transport protocols that leverage multi-path routing to dramatically increase network goodput. While effective at achieving high throughput, both systems are unfortunately incompatible with traditional transport protocols like TCP and latency-sensitive applications.

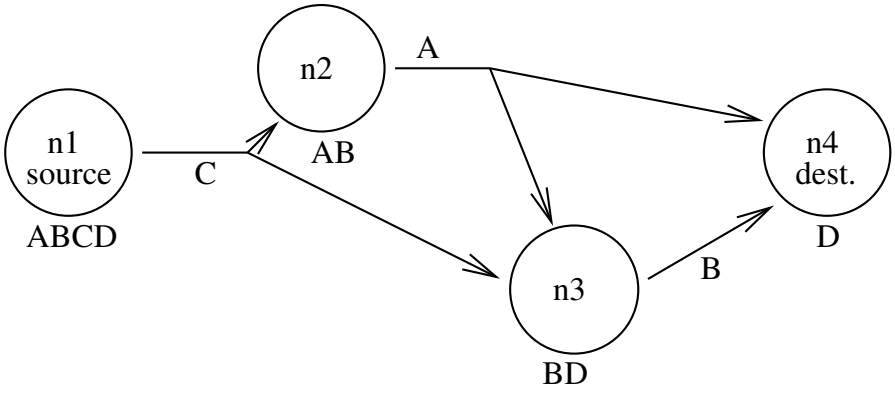



Figure 2.2: Operation of EXOR system

**ExOR**

ExOR is a bulk-data protocol: rather than transmitting individual packets, it transfers 'batches' of packets. The source gathers together the set of packets destined for a particular destination and transmits them all at once, along with a precomputed 'forwarder list' enumerating any likely intermediate nodes between the source and destination (in order to keep the list size manageable in dense networks, ExOR prunes nodes expected to overhear less than 10% of the packets.) The source prioritizes the forwarder list based upon its estimation of their proximity to the destination (computed using the ETX metric, described above). Any nodes contained within the forwarding list that successfully receive packets transmitted by the sender buffer them until the batch is completed.

Once the sender has finished sending the batch, the receiving node with highest priority begins forwarding any packets it has buffered. The node annotates this so-called batch 'fragment' with its estimation of the highest-priority node to have received each packet in the batch, called a 'batch map'. Subsequently, each node in the forwarding list takes its turn sending any packets not previously acknowledged in another's batch map until the destination has received at least 90% of of the packets in the batch. The remainder of the packets are forwarded using traditional routing.

One of the most challenging aspects of implementing ExOR is ensuring each forwarder transmits its batch fragment at the appropriate time. If transmissions are uncoordinated, fragments will collide, eliminating any potential benefits. Moreover, until first packet is transmitted by a higher priority node, the lower priority nodes do not have

the complete batch map. Thus, if nodes start transmitting in the wrong order, they will transmit extraneous packets, discarding any benefits. For those reasons, the ExOR design requires a sophisticated scheduling system which keeps a transmission timer, as well as records the fragment numbers being transmitted to estimate the effective channel rate and predict when individual forwarders will complete their fragment transmissions.

The operation of ExOR protocol is illustrated in 2.2. There are four packets, A, B, C, and D and four stations, *n1* to *n4*. Each station is annotated with the batch map. The solid lines show the packet propagation, and annotated with the packets that the station would send when it is scheduled to transmit. Note that *n2* will not send the B packet because it knows that *n3* already has the packet, and *n3* is closer to the destination than *n2*.

Rate selection for the ExOR protocol is still an open question. The original paper ran all stations at a fixed data transmission rate (11 Mbps), while the code available for download uses standard rate selection algorithms such as SampleRate to select the transmission rate on hop-by-hop basis. Since ExOR is based on the ETX metric, which includes the transmission rate, another obvious solution would be to utilize the rate used in the ranking algorithm. However, these rate selection strategies are not aware of the multiple possible paths: SampleRate optimizes the rate to the next hop, while ETX optimizes the rate along the single best route to the destination. In this dissertation, we explore a more advanced method of rate selection that is aware of the overhearing.

**MORE**

MORE is another routing protocol, proposed in [CJKK07]. MORE's operation is similar to ExOR, but it uses 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 [CJKK07].

Linear network coding breaks data into native blocks, and then transmits the linear combination of those blocks. This has a useful property: if the original data was broken into $n$ blocks, one can generate a very large number of coded blocks – each coded block being a linear combination of all native blocks. Each coded block is annotated with code vector, which describes the coefficients of the native blocks. Once any $n$ coded

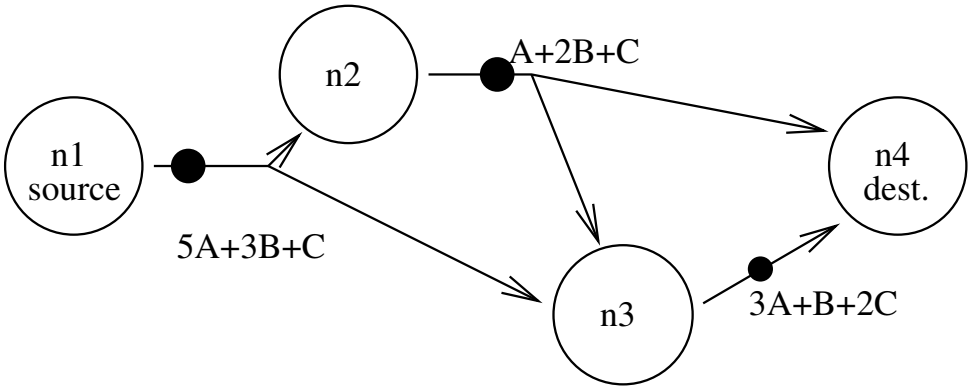



Figure 2.3: Operation of MORE system

blocks with independent code vectors have been received, all native blocks, and thus the original data, could be recovered using simple matrix operations.

MORE uses finite-field linear network coding, which ensures that coded blocks have the same size as native blocks, and that code vectors are small. A source generates and transmits coded blocks with random code vectors. Forwarder nodes listen to all received coded blocks and checks if they are 'innovative' – they contain new data that could not be calculated from stored blocks. If a coded block is innovative, the forwarder stores it. Additionally, when a block is received from a node which is farther from the destination than forwarder, the forwarder calculates a random linear combination of the blocks it has, thus generating a new coded block, and transmits it. The destination simply listens to all coded blocks, and once enough coded blocks to reconstruct original data are received, stops the process.

The operation of algorithm is illustrated in 2.3. There are three native packets, A, B and C, and four stations, *n1* to *n4*. Each packet that a station could sent is illustrated by the solid black dot, and annotated with corresponding code vector.

MORE does not have a strict scheduler which ExOR has; instead, each node makes a local decision based on pre-calculated flow amounts. Before each transfer, the control system calculates the expected number of inbound and outbound packets for each node, corrected for the measured link loss rate. The source transmits continuously (subject to MAC restrictions), and the nodes transmit enough to maintain the calculated ratio. For example, if calculations showed that for a 100-block batch, some node would

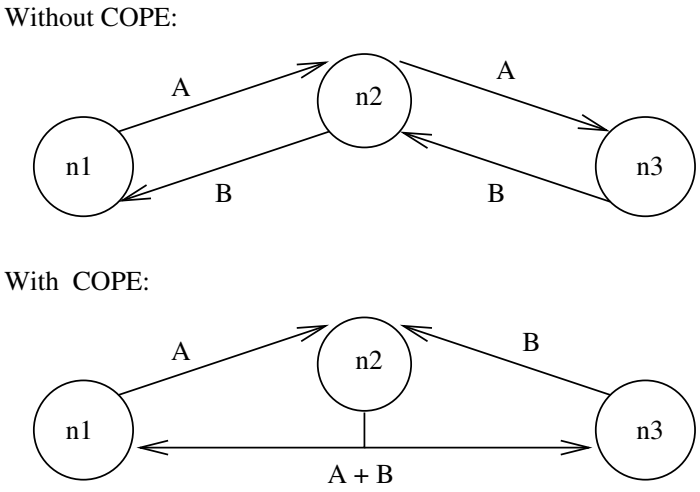



Figure 2.4: Operation of COPE system

have to receive 90 coded blocks and transmit 110 coded blocks, then during the actual data transfer, this node would send out the packets in order to maintain this ratio. Since each transmitted block is a different linear combination of stored blocks, the nodes can transmit more than they receive, and still generate useful information. The lack of strict scheduler simplifies the system, but at the same time potentially increases the hidden terminal problems.

The original MORE paper does not address the issue of rate selection, and performs all experiments at a fixed rate. We explore the question of rate selection in MORE in Chapter 5.

### COPE

Another notable approach based on network coding, COPE [KRH+06], does not target opportunistic overhearing in quite the same fashion as the schemes described previously. Instead, 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. This is illustrated in 2.4: the system contains two data streams, from *n1* to *n3*, and from *n3* to *n1*. There is no direct connectivity between *n1* and *n3*, so *n2* acts as forwarder. Without COPE, four packets transmissions are needed to transfer one packet from each stream.

With COPE, *n2* sends a sum of two packets. Each station can retrieve the other packet by subtracting the packet that it sent.

While we believe that there is room to optimize the performance of COPE, we leave this to future work.

## 2.3 Summary

While infrastructure-based wireless networks are widely used and well known, there is a lot of research in the area of mesh networking. A variety of existing projects separately study various aspects of mesh: routing, rate selection, and overhearing. However, there are few studies that try to combine these various methods of link optimization. In the rest of this dissertation, we explore two scenarios related to overhearing: taking advantage of overhearing in regular (single-path) mesh networks, as well as increasing overhearing in multi-path networks.

# Chapter 3

# Networks under test

Before proposing methods to take advantage of overhearing, we seek to understand how frequently it occurs, and which possible benefits it can bring. An efficient way to do so is to simulate networks with and without overhearing and compare the results. This chapter describes our simulation efforts. It describes the the design of our simulator, the datasets we use, and finally provides the initial results. We also describe the testbeds that we have set up, and that we will use for the experimental evaluation of our protocol improvements.

## 3.1 Metrics of interest

Our high-level goal is to increase the efficiency of mesh networks. There are a number of metrics that attempt to evaluate efficiency; in this work, we focus on total airtime utilization. The total airtime utilization determines the total duration of all transmissions that are required to transfer a single packet from source to destination. The main limitation in mesh networks is network capacity, thus, for a network with large number of users, the total airtime utilization is going to directly affect the goodput of all users.

As a simplification, we sometimes refer to a different metric, the total number of packets transmitted. If all nodes transmit at the same rate, this metric is loosely proportional to total airtime, neglecting overheads like acknowledgment packets.

As a starting point, we seek to study the potential benefits of overhearing, so we

simulate simplified, abstract protocols. In the latter parts of this dissertation, we work with specific protocols, and we modify the simulator to provide specific details for those protocols.

## 3.2 Datasets

In order to properly understand the phenomena of overhearing, we need extensive measurements of real networks, namely the transmission probabilities between each station pair at all rates. We call such a set of measurements 'dataset'. In this work, we use four datasets: one existing set of measurements, two taken from testbeds we have set up, and one set of measurements we captured ourselves from an existing network. Later, we use these datasets to quantify the potential improvement from overhearing using a simulator that we have developed.

### 3.2.1 Roofnet

Our initial analysis is based on the Roofnet mesh network dataset. Roofnet [ABB+04] is a mesh network deployed in Cambridge, MA. It consists of 38 nodes, spreading over four square kilometers. Nodes are are located at apartments of volunteers, which were not selected in any particular way (except to provide basic radio connectivity). The location of nodes is shown in 3.1.

Each node has a single wireless network card based on Intersil Prism 2.5 chipset, which supports 802.11b. The cards are set up to use channel 3 of the 2.4-GHz band, and transmit at +23 dBm (200 mW). Each node has an antenna mounted outside, on the roof of the building. Antennas are omni-directional, and provide 8 dBi of gain, with a 20-degree -3 dB vertical beam width. The cables and lightning arrestors introduce additional an 6 to 10 dB of loss.

The dataset was collected on this 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. [ABB+04]

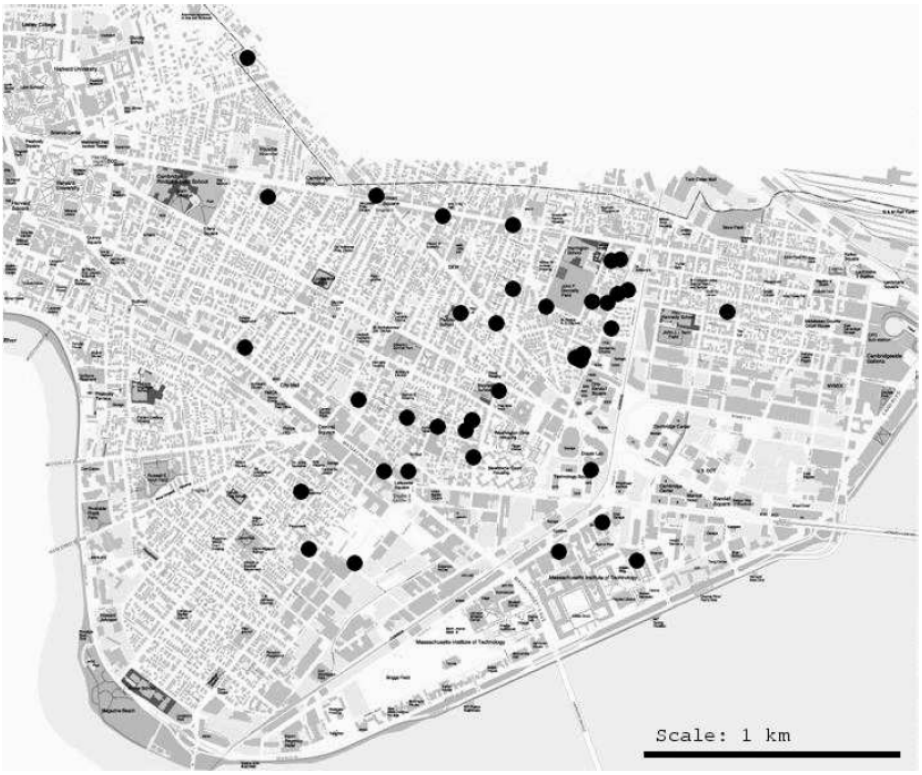



Figure 3.1: The Roofnet mesh network. Reproduced from [ABB+04].

The reception rates were measured while 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 is. Unfortunately, there are no published Roofnet datasets under such conditions.

The available dataset contains the complete log of every packet being sent or received. From this raw data, we calculate a direct distribution, which specifies the probability that a packet transmitted at one node is received at some other node. In other words, this distribution specifies $\Pr_r[A \rightarrow B] \; \forall \; A, B \in G, r \in \{1, 2, 5.5, 11\}$ Mbps.

Using the direct distribution, however, requires us to assume that the transmission probabilities are mutually independent. We lift this assumption by calculating a joint distribution – that is, the probability that a packet sent by one node is received by *subset* of a nodes: $\Pr_r[A \rightarrow S] \; \forall \; A \in G, S \subset G, r \in \{1, 2, 5.5, 11\}$ Mbps.

Unfortunately, the Roofnet 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 1 Mbps, but are forced to use the rate for 1500-byte packets, which is likely to be lower.

### 3.2.2 Jigsaw testbed

While the Roofnet dataset is well studied, it is based on a single survey, and does not cover variations like different transmit powers, different channels or 802.11a / 802.11g standards. Thus, we replicate the measurements ourselves on a testbed based on the Jigsaw wireless analysis engine at the University of California, San Diego [CAV+07, CBB+06].

The Jigsaw system is deployed within the Computer Science and Engineering building and spans four and a half floors covering approximately 150,000 square feet of floor space and one million cubic feet of volume. 3.2 shows the layout of the sniffer nodes. In addition to human inhabitants, the building contains thousands of workstations and a large variety of electronics operating in the same 2.4 and 5-GHz unlicensed frequency bands as 802.11, resulting in highly variable channel quality in different portions of the building and during different times of the day [CBB+06]. There are 39 ‘pods’, each

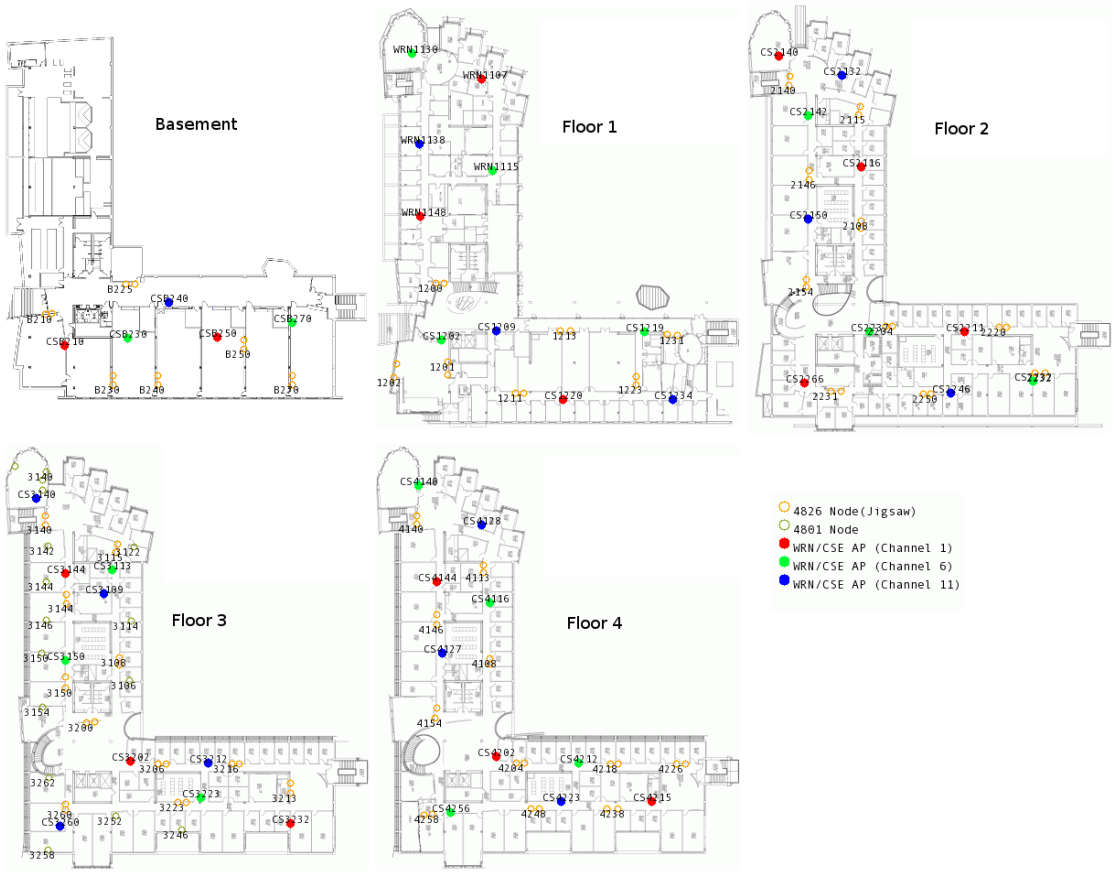



Figure 3.2: The UC San Diego Jigsaw wireless testbed. For our work, we use the Jigsaw mesh nodes (depicted as circles); production 802.11 access in the building is provided by traditional infrastructure-mode access points (represented as triangles).

consisting of two sniffer nodes, plus a number of additional nodes installed later to provide coverage for a total of 92 sniffers. All nodes are connected to a central server with a wired Ethernet control connection. The central server saves all received data and runs an analysis engine to synchronize the data.

Sniffer nodes are based on a modified Soekris Engineering net4826 embedded computer, and contain a 266-MHz AMD Geode CPU, 128 MB of DRAM, 64 MB of Flash RAM, a 100-Mbps Ethernet interface, and two Wistron CM9 miniPCI 802.11a/b/g interfaces based on the Atheros 5004 chipset. Each wireless interface is connected, via shielded cable, to a separate external omnidirectional'rubber duck' antenna mounted six inches apart on an aluminum enclosure. The antennas provide a signal gain of 2-3 dBi at 2.4-GHz. Each monitor receives wired connectivity and power through a port on an HP 2626-PWR switch. There are seven such switches.

Sniffer nodes have a fixed channel assignment: out of a total of four cards in a pod, three cover the common 802.11b channels (1, 6 and 11), and the fourth one is either disabled, or monitors some other channel, such as channel 3 or 8. The wireless cards on the sniffers are permanently placed in monitor mode, and special dumping software continuously collects all received packets and sends them to a central server via the control connection.

The Jigsaw analysis engine is particularly appropriate for our needs, since it is able to tightly time-synchronize traces collected at multiple nodes and precisely determine which received frames are actually identical—i.e., a single transmission that was successfully decoded at multiple receivers—and which are duplicates (such as those that result from link-layer retransmission). The infrastructure also automatically records the RSSI and any associated hardware errors reported along with each received frame.

For our tests, we do not want to disrupt the normal operation of the Jigsaw system, and thus we use the existing channel assignment. I.e., when we run experiments on a channel 11, we select only those sniffers which have a radio sniffing a channel 11, and convert them into a testbed node by running our custom software. This way, our full testbed effectively has 48-50 nodes, with only one radio used in most pods.

To support a comprehensive evaluation, it would be desirable to have multiple testbeds with varying physical properties. It would be difficult to create a number of

separate testbeds, however, so we instead vary the subset of nodes we select or vary the transmit power of the nodes in use. By employing different subsets of the nodes, we can change the set of possible routes: the whole testbed spans multiple floors, naturally providing multiple possible paths, especially when source and destination are not on the same floor. Also, by limiting our experiments to a single floor, we can construct a 'flat' topology with a smaller number of paths which should be closer to other 'flat' topologies like Roofnet. Another way to vary the testbed is to alter the transmission power. Reduced power will cause decreased reception range, roughly equivalent to increasing inter-node distance. This will cause route length to increase, thus presenting more opportunities for overhearing.

To measure the performance of the testbed, we conduct a network-wide link survey simular to that conducted by Roofnet researchers. During each survey, we fix all nodes to the same 802.11 channel (11 in these experiments), set them to listen in monitor mode, and then perform the test. We conduct our experiments during the night to reduce the level of interference.

In our initial experiments, we implemented the Roofnet procedure, iterating through each of the nodes in the network in the following fashion. At each node, we transmit 1,000 maximum length (1,500-byte) packets back-to-back at a particular bitrate. We cycle through each of the 12 available 802.11g bitrates in order before moving on to the next node. The entire process takes roughly 10 minutes. Once the transmission phase is complete, we submit the traces to the Jigsaw analysis engine to determine how many stations received each individual frame and calculate the probability matrices. The analysis takes an additional 10 minutes, during which the infrastructure cannot conduct further probe experiments.

We discovered, however, that this measurement technique can be highly inaccurate in our environment. In particular, a moderate-length burst of broadband interference can completely distort measurements for one or more links. Thus, we alter our survey procedure to split transmissions into groups: We divide the 1,000 packets that each node transmits into 10 groups. Then, at each node, we transmit 100 1,500-byte packets back-to-back at a particular bitrate. We cycle through every node in the system and then move to next bitrate. Once all bitrates are done, we repeat the whole process 10 times


Figure 3.3: ALIX network map. All nodes are located on the third floor.

until every station has transmitted 1,000 packets at each bitrate. In addition to spreading the impact of broadband interference, this method also allows us to estimate short-term variations in link quality. We calculate the standard deviation of the reception rate of each link. In our experience, the deviation is an important parameter of the link, staying relatively consistent over each measurement. For a typical link with 0.70 reception probability, we see deviations that range from 0.01 to 0.30.

### 3.2.3 ALIX testbed

While the wireless network cards in Jigsaw sniffer nodes network support the 5-GHz band, some of them require a separate antenna. This antenna is not installed in the existing nodes, making them effectively incapable of supporting 802.11a transmissions. Additionally, testing shows that a higher frequency signal has significantly less penetration through the walls, and that more dense a testbed is needed to obtain a

well-connected network. Therefore, we deploy an additional, separate testbed to evaluate 5-GHz performance.

The second testbed is based on ALIX 3d2 embedded computers from PC Engines. These nodes contain a 500-MHz AMD Geode LX800 CPU, 256 MB of DRAM, 1 GB of Flash RAM on CompactFlash card, a 100-Mbps Ethernet interface, and one or two Wistron CM9 miniPCI 802.11a/b/g interfaces based on the Atheros 5004 chipset. These nodes do not support PoE, so each node is powered by an AC adapter.

We densely deploy the ALIX nodes around the third floor of the building as shown in 3.3. Unlike Jigsaw nodes which are mounted primarily in the hallways near the ceiling, the ALIX nodes are placed on the tables in people's offices. The nodes are connected to the same central server that was used for the Jigsaw testbed, but we do not run the Jigsaw sniffing software on these nodes. In all other aspects, the ALIX nodes are running the same software, and performed the measurements in the same fashion as nodes in the Jigsaw testbed.

At the time of our experiments, there were no access points or clients operating at 5-GHz band in the building. Thus, the ALIX testbed provides measurements that are more repeatable and consistent than measurements on the Jigsaw testbed. For that reason, we use it as the primary network in the analysis.

### 3.2.4 Google network

Finally, we obtained limited access to the Google WiFi network – a free, outdoor wireless Internet service deployed in Mountain View, California. The network has been continuously operational since August 16, 2006, and provides public access to anyone who signs up for an account.

The network consists of over 500 Tropos MetroMesh pole-top access points. Each Tropos node has a distinct identifier and a well-known geographic location; 3.4 shows the approximate location of the Tropos nodes. Each Tropos node serves as an access point (AP) for client devices, as well as a relay node in a wide-area backhaul mesh that provides connectivity to the wired gateways. The topology of the Tropos mesh network is constructed dynamically through a proprietary Tropos routing algorithm. A pure mesh network of this scale exhibits significant traffic congestion at nodes close

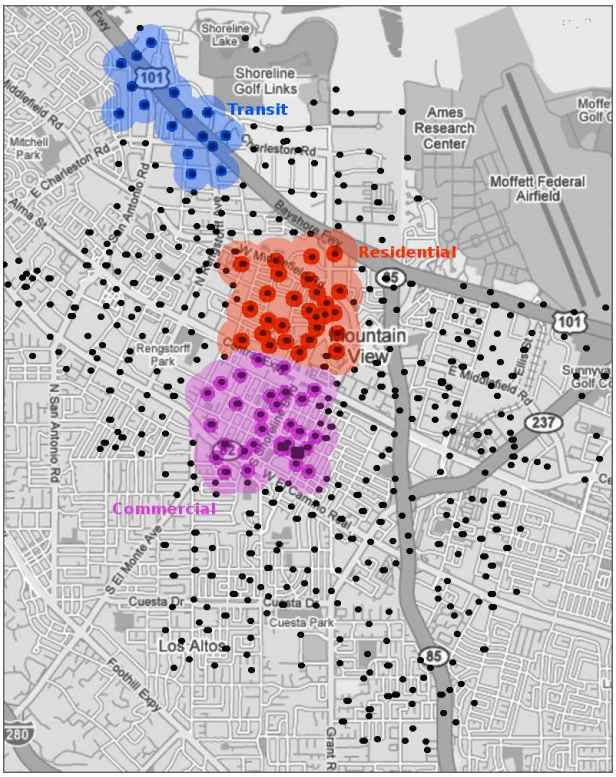



Figure 3.4: Google WiFi network

to the gateway router, however. To alleviate the congestion, the Google WiFi network is hierarchically clustered around approximately 70 point-to-point radio uplinks that serve as a fixed long-haul backbone for the mesh network. This effectively partitions the Google network into multiple separate partitions, each containing exactly one uplink node. The channel assignment mechanism attempts to select different channels for each partition, and a same channel for all nodes within that partition. As a result, the network is split into a large number of small sub-networks, with up to 14 nodes each.

We were not able to run our experimental code on the Google network, thus we are limited in the measurements we were able to obtain. Still, we feel that it presents an important data point, being an existing, commercially deployed mesh network.

## 3.3 Network characteristics

In this section, we present basic measurement results from different networks. These measurements show general channel characteristics, and allow us to estimate the frequency of overhearing and effects of the link rates on the amount of overhearing.

### 3.3.1 Number of recipients

We examine the the average number of recipients of a transmission. While those results do not measure overhearing-related factors directly, they can provide a hint of possible overhearing – a small number causes the network to have few possible paths with little opportunities for overhearing, but a large number implies short paths with reduced opportunities for overhearing.

We plot a CDF over each of the nodes in the testbeds. The horizontal axis shows expected number of recipients per single packet transmission – thus, a value of 2 can mean either two links with 100% transmission rate, or four links with 50% transmission rate. We expect every value to be at least two for a well-connected network.

3.5 shows the information for the Roofnet testbed. The Roofnet network is not well connected – about 10% of nodes have less than one expected recipient, even at the lowest bitrate, meaning that no matter which path and bitrate is chosen, multiple transmissions will be likely required. We discard the three least connected nodes from all

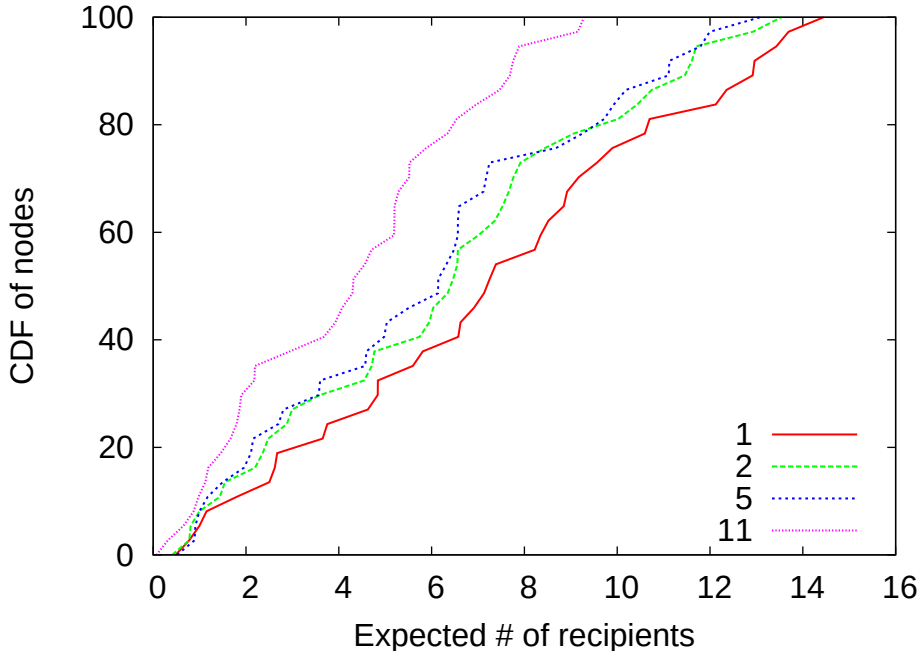



Figure 3.5: CDF of the number of the average number of recipients per packet as a function of bitrate. Roofnet testbed.

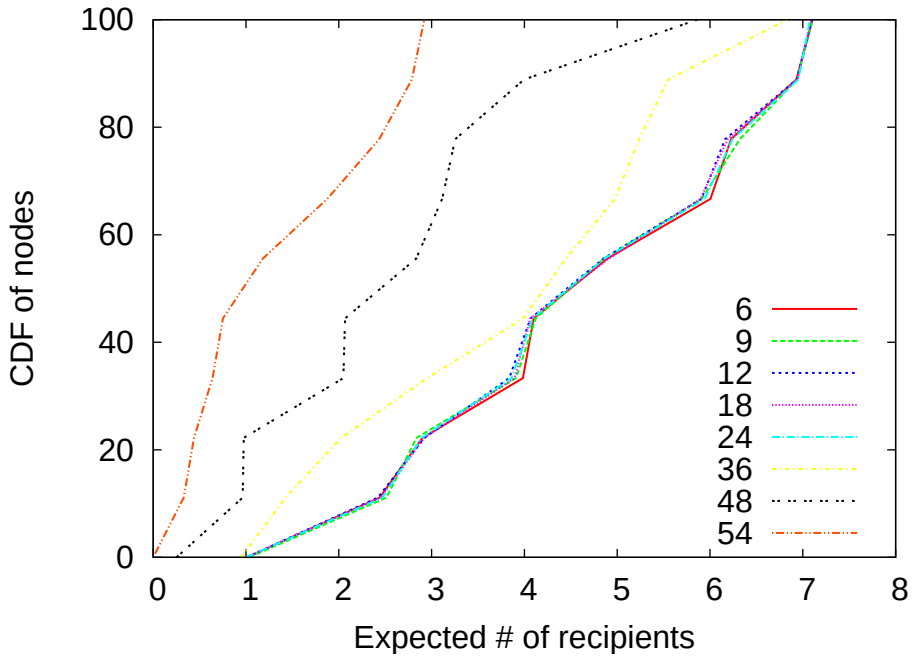
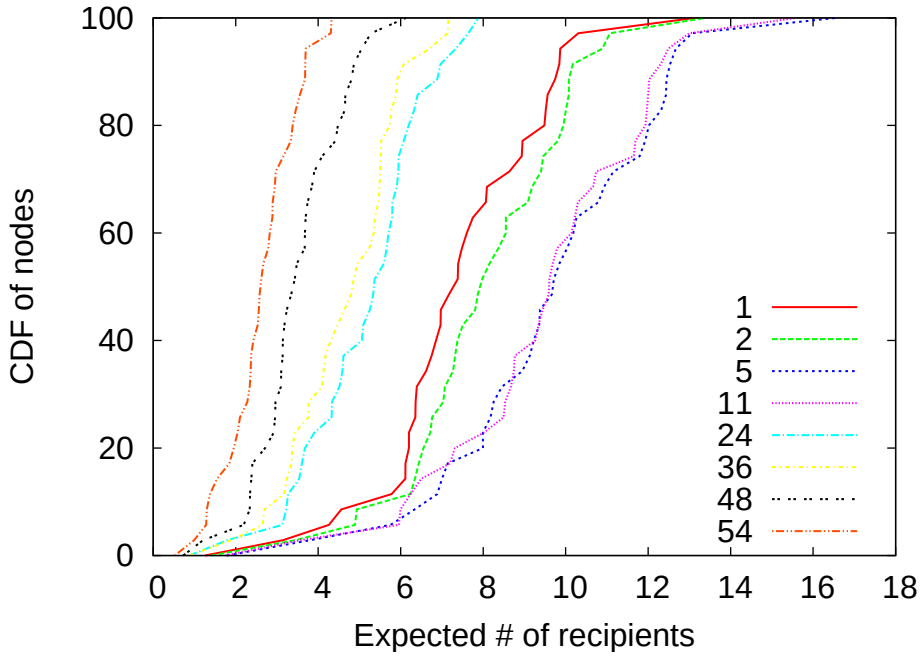
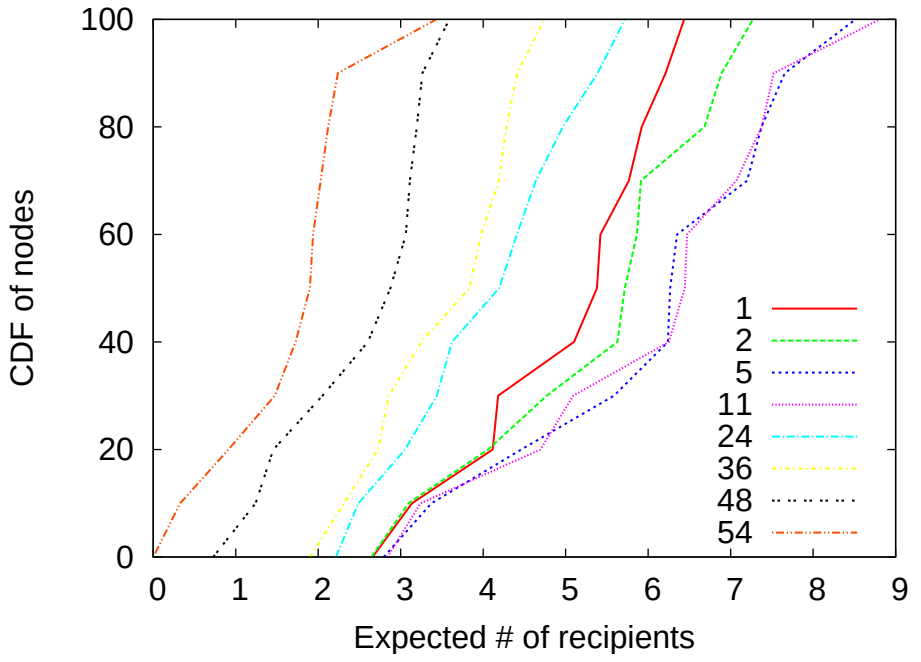



Figure 3.6: CDF of the number of the average number of recipients per packet as a function of bitrate. ALIX testbed, maximum transmission power.




(a) Whole building




(b) 2nd floor

Figure 3.7: CDF of the number of the average number of recipients per packet as a function of bitrate. Maximum transmission power, Jigsaw network

further analysis.

The network supports only 802.11b, so only four bitrates are available. The higher bitrates show lower reception probabilities: while at a minimum bitrate (1 Mbps) a transmission from the median station is heard by 7.1 stations on average, at maximum bitrate (11 Mbps) this number decreases to 4 stations. Since Roofnet is an outdoor network, we expect that most of the packet loss is caused by signal attenuation, thus more sensitive encoding methods produce a better range.

3.7(a) shows the results for the Jigsaw testbed. This testbed is quite different – since the Jigsaw testbed is indoors, interference has a significantly higher impact. As a result, Barker-based modulation (1 Mbps and 2 Mbps) has smaller reception range than CCK-based modulation (5.5 Mbps and 11 Mbps). The bitrate has little impact for 802.11b CCK-based encodings (5.5 and 11 Mbps): we see that the curves for each of these speeds are very similar. In contrast, 802.11g encodings show markedly smaller reception ranges in general, and significantly different reception rates at the high end (i.e., 54, 48 and 36 Mbps). For example, switching between 54 and 48 Mbps adds one additional recipient to the median node, while dropping all the way down to 24 Mbps adds 4 additional receivers on average, and up to 10 in the best case. The lower rates, on the other hand, perform almost identically to each other.

Yet, this network is still not well connected – some nodes have less than one expected recipient even at the slowest rates. We discard the poorly connected nodes when running the experiments on the whole testbed.

Since the results are very similar for the lower 802.11g rates, we limit the subset of rates that we study in the remainder of this paper. We consider only 1, 2, 5.5, 11 Mbps in the 11b range, and rates 24, 36, 48, 54 in the 11g range.

3.7(b) shows the results for subset of the Jigsaw nodes located on the second floor. While similar to the whole network, the subset has significantly stronger reception – in particular, at speeds of 36 Mbps and below, every node has at least two expected recipients. This effect is caused by the lack of ‘difficult’ inter-floor links. We use this subset of the testbed as a sample well-connected network for some of our routing experiments.

Finally, 3.6 shows the results for the ALIX testbed. The results are qualitatively

similar to the second-floor results, although the absolute number of nodes receiving a single transmission is lower due to the smaller size of the testbed, and there is far greater separation between the three high 802.11a speeds and the remaining ones.

### 3.3.2 Power control

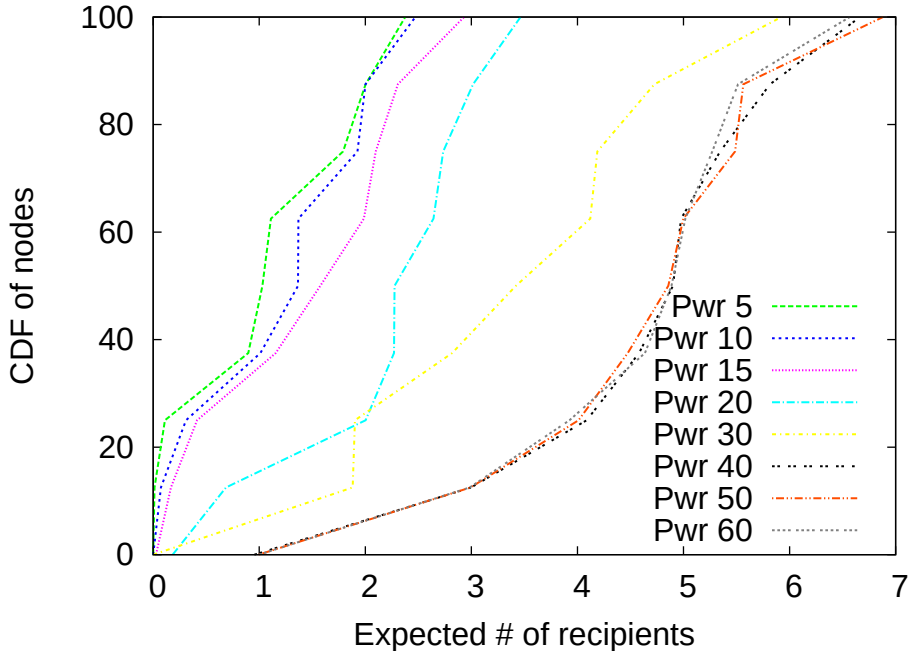
We can vary the connectivity of the testbeds by adjust nodes' transmit power. 3.8(a) and 3.8(b) compare the transmission range of 24 and 54 Mbps link rates on an ALIX testbed across a range of channel powers. These graphs show that power control is effective in changing the network density – depending on the transmit power, node connectivity varies across a wide range. Some of the experiments in the latter part of this thesis will use experiments at power level of 30.

### 3.3.3 Google network measurements

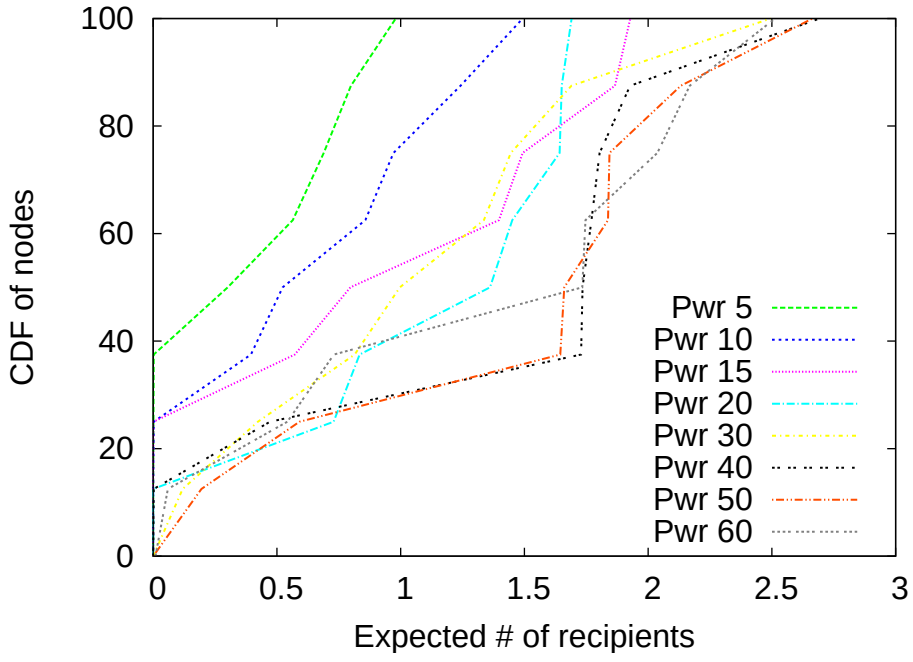
While we could not conduct explicit measurements on the Google network, we are able to collect basic information about the mesh topology and link properties through an administrative interface exported by the Tropos nodes.

Mesh degree is the number of recipients capable of receiving transmissions. Those values can be compared to average number of recipients graphs in the section 3.3.1. However, unlike number of recipients graph, the mesh degree does not indicate the link quality, and any link which could be a considered neighbor adds 1 to the graph. 3.9 shows three distinct ways to measure the average Mesh degree.

The relatively dense deployment of APs provides significant path diversity. When considering neighbors which provide acceptable link quality (SNR ratio of 14 dBm or better [SR07]), only 5% of APs have a unique neighbor; the median AP can communicate with at least 4 neighboring APs, and the most well-connected 10% have more than 8 potential next hops. We observe, however, that very few of these potential links are used in practice. The 'active link' curve plots only the links which are being used by routes in the network. Most access points use just one link. (There is a very small fraction of nodes with zero mesh links—these are nodes with a point-to-point uplink, but no neighbors in the mesh.)




(a) 24 Mbps




(b) 54 Mbps

Figure 3.8: Expected number of recipients as a function of transmit power, ALIX network.

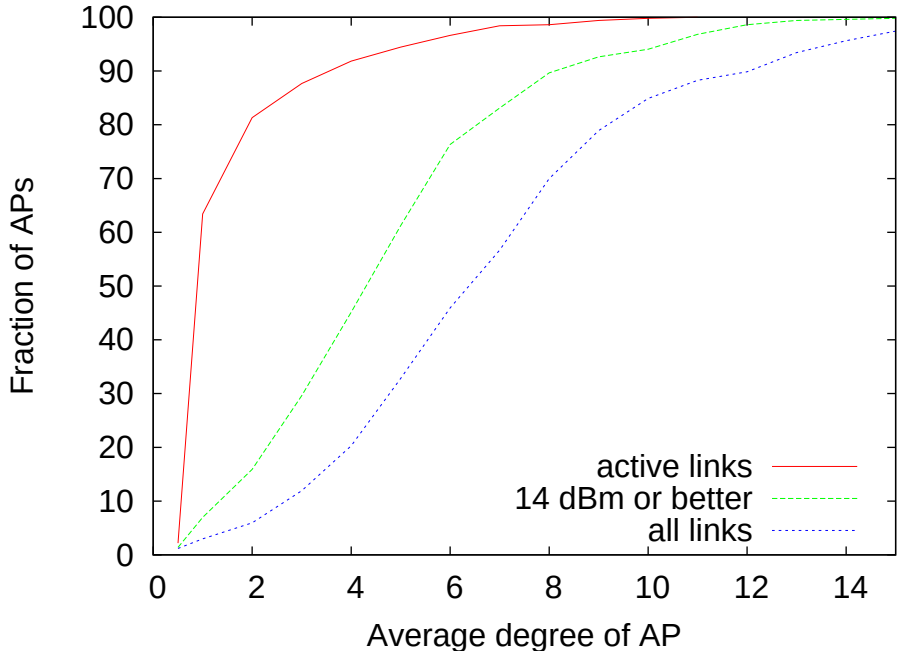



Figure 3.9: Average mesh degree of each Tropos AP.

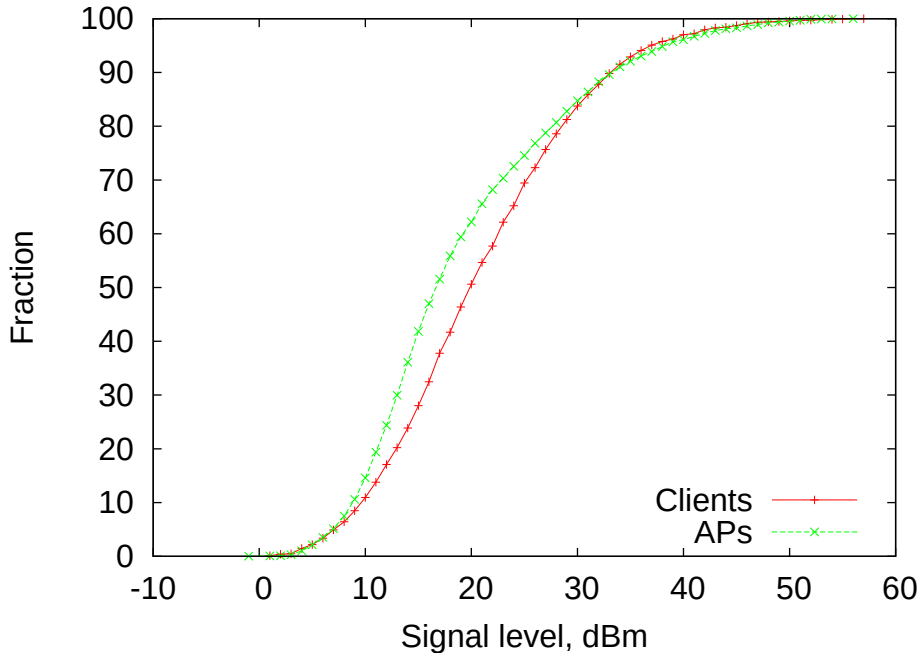



Figure 3.10: Received SNR levels from other Tropos nodes and clients.

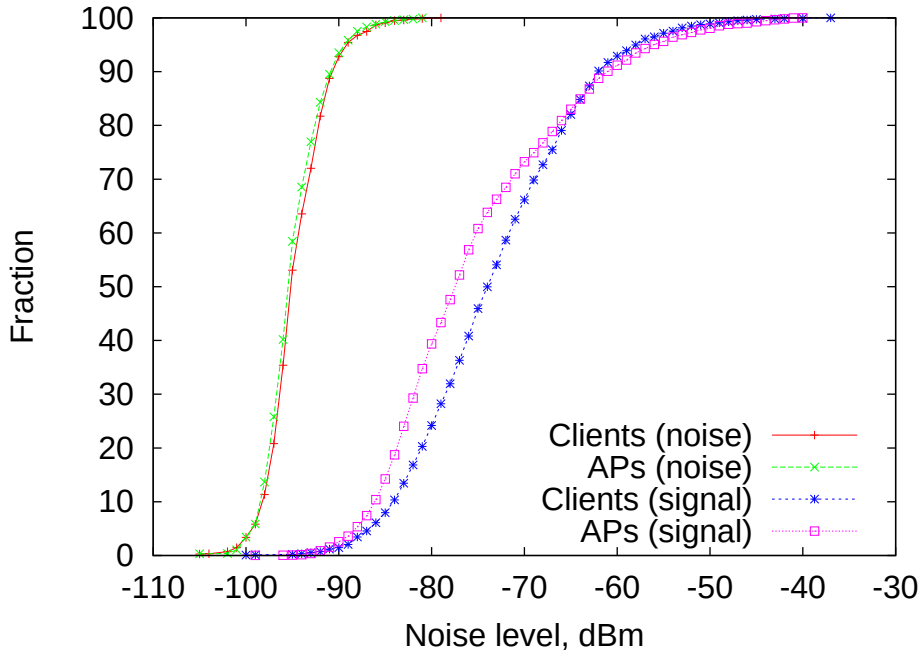



Figure 3.11: Signal and noise levels for both Tropos nodes and clients.

One of the main advantages of overhearing-aware systems is ability to make use of low quality links which are avoided by regular routing. Thus, the quality of each link is important. In an attempt to quantify the quality of the mesh backbone links, we collect signal strength and noise measurements at the Tropos nodes as reported by the Tropos administrative interface. (To the best of our knowledge, however, the Tropos routing software does *not* use link quality measurements to establish routes, instead preferring reception probabilities—which we do not have the facilities to report.) 3.10 shows a CDF of the signal to noise level measured at each Tropos nodes for links to both other Tropos mesh nodes and individual network clients. The AP curve plots *all* possible links (c.f. 3.9), including adequate links which are not currently in use and those with SNRs of less than 14 dBm. While qualitatively similar, APs appear to enjoy a slightly higher SNR than clients. This phenomena is explained in 3.11 by observing that the noise levels are essentially identical for both types of nodes, but the received signal strength of other Tropos nodes is generally higher than clients, likely due to the line-of-sight provisioning of the pole-top Tropos nodes.

It is well known that SNR levels do not correlate directly with link quality,

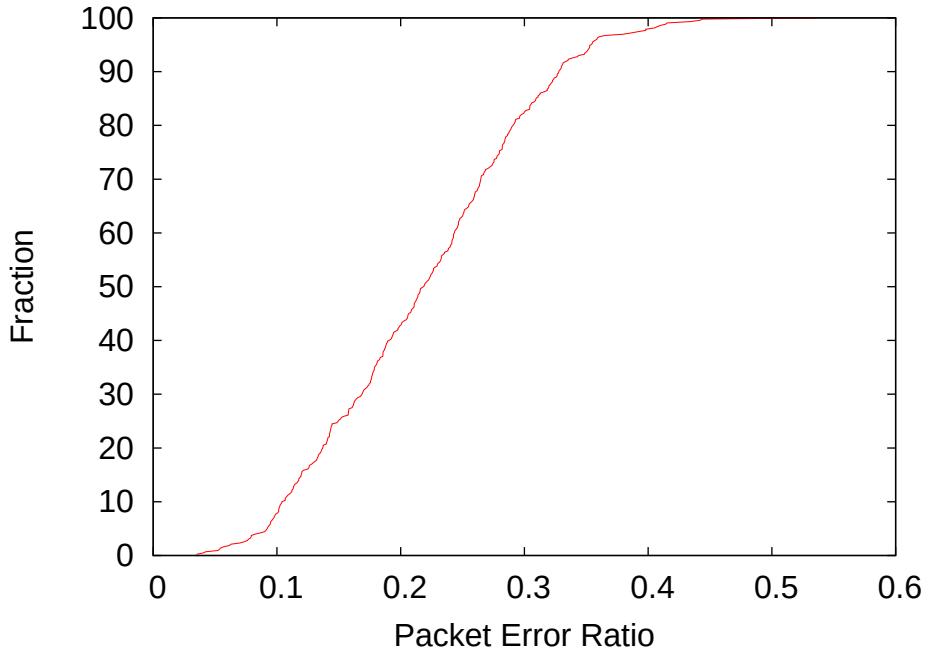



Figure 3.12: Packet error ratios for mesh-backbone communications

however. 3.12 plots the packet error ratio (PER) of mesh backbone links, i.e., links between Tropos nodes, as reported by the transmitter. Concretely, it is the average number of frame transmissions required to successfully transmit a packet measured over the same interval as the signal and noise data reported previously. We see that the median link needs to retransmit frames more than 20% of the time. We have not studied whether any particular aspects of Tropos node placement leads to better or worse PER values.

## 3.4 Simulator design

The traditional way to simulate wireless network transmission is a per-packet network simulator, such as ns2 [ns]. Such simulators typically employ a Ricean or Rayleigh fading model to simulate the physical transmissions of each individual packet. These simulators can provide a good approximation of real-life behavior if the model parameters are set adequately. However, such simulators have disadvantages: they require many hard-to-determine parameters in order to properly simulate an existing network; they are not very good in simulating indoor networks with large amounts of fading and

interference; and event-based simulations are extremely time consuming, requiring long experiments to provide precise results.

In this dissertation, we focus on the overall performance at a given path between source and destination, rather than on an tracing each individual packet, so we chose a different approach. Rather than use time-based simulators, we create a statistical network simulator which uses Bayes Theorem and Markov chains to calculate the expected number of total transmissions for each node per unique source packet. The output of the simulator is total airtime utilization. Our simulator will cover three cases:

1. *Traditional case* has no overhearing at all. This will be used as a baseline in all our comparisons.

2. *On-path overhearing* is the simplest form of overhearing. It assumes a network with a structure similar to traditional, but with an additional property that any packets overheard by nodes closer to the destination do not have to be transmitted to that node.

3. *Off-path overhearing* assumes a complete re-design of the protocol in order to take full advantage of overhearing. Such networks no longer have a single, default route and instead have a large number of forwarder nodes, each of them is ready to send data to the destination.

### 3.4.1 Traditional case

The baseline case is a traditional mesh routing protocol like Srcr. For the purpose of the simulation, we assume that the route information is up-to-date on all nodes – there are no routing loops or transient effects. In this case, the total transmission time is the same as ETT route metric[BABM05], which is the route metric used by Srcr.

### 3.4.2 On-path overhearing

On-path overhearing refers to the case when packet sent to a node is also received by a node further along the path. For the purpose of this section, we assume that there is some scheme that can take advantage of that fact – for example, there exists some

additional channel which can notify a sender that the receiver has already heard the packet. In this case, every overheard packet will cause one less packet transmission, thus saving airtime. One implementation of such a scheme will be given in the next chapter.

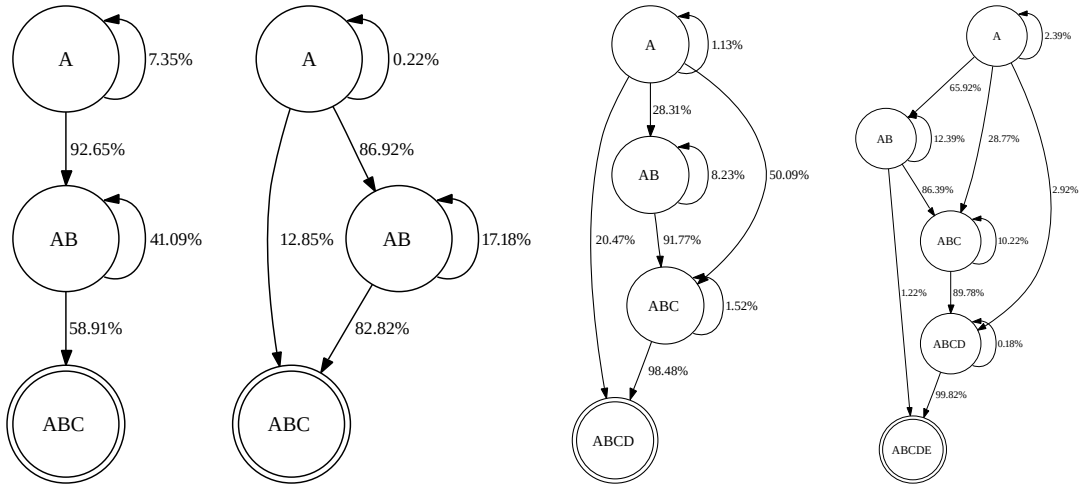
To evaluate whether on-path overhearing can increase performance by avoiding transmissions, we construct a statistical model to estimate the expected number of transmissions along each path. We examine each source/destination pair individually, and, for each pair:

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 both normal 802.11 and using on-path overhearing. Initially, we 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 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. For the overhearing-aware case, we ignore the ACK delivery rate, 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 overhearing transitions and probabilities, and once using only the on-path $A \rightarrow AB$ and $AB \rightarrow ABC$ base-case 802.11 transitions (in the latter case, the results are identical to the route metric).

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




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

Figure 3.13: Actual overhearing scenarios. Self-loops represent complete packet loss events. All probabilities are based upon a 1-Mbps transmission rate.

interesting. 3.13 shows four state machines corresponding to actual paths in our datasets. 3.13(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. 3.13(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. 3.13(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. An overhearing-aware system, on the other hand, benefits from this overhearing because it does not need to ACK directly from $C$ to $A$. Finally, 3.13(d) shows three distinct overhearing paths from $A$ to $E$: $A \rightarrow B \rightarrow E$, $A \rightarrow D \rightarrow E$, and $A \rightarrow C \rightarrow D \rightarrow E$.

### 3.4.3 Off-path overhearing

Off-path overhearing is more complex, and implemented by protocols like ExOR. Unlike on-path overhearing, there is no default 'path' for packets to follow. Any node that has received a packet has a chance to forward it, as long as it involves 'making progress'. The 'making progress' step is important, as it will ensure that the whole transfer will eventually terminate. ExOR assigns a rank to each node the same way Srcr rank is assigned, from highest to lowest, and only forward packets which came from a node with higher rank. Thus, any node will forward a packet from the source, since the source has the highest rank. On the other hand, nodes which have high probability of delivering packets directly to the destination will have a very small rank, and their packets will not be forwarded needlessly.

From the simulation standpoint, the situation is very similar to on-path overhearing. We simulate an ExOR-like protocol. The differences from the on-path case are:

1. Set of input nodes: instead of routing over the small set of nodes located on the shortest path from source to destination, there is a much larger set of nodes which are closer to destination than sources, according to the routing metric.

2. Routing metric: Since ExOR does not need per-packet acknowledgments, the quality of reverse link does not affect the metric. In our simulator, we assume that the batch maps are always propagated.

### 3.4.4 Potential benefits

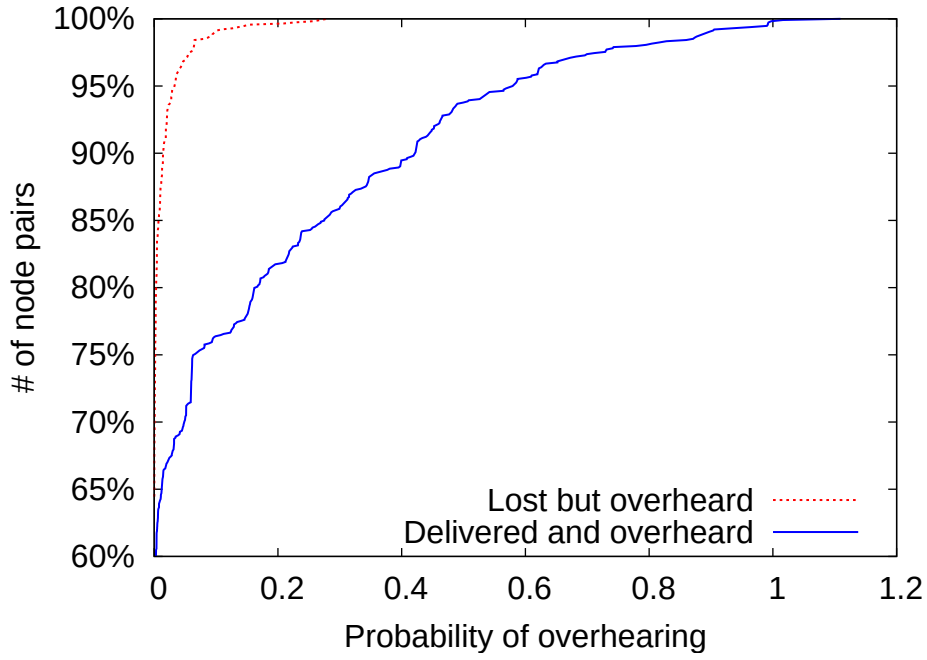
For the testbed networks, we can compute a simplified estimation of overhearing, which attempts to measure the amount of the overhearing in the intuitive way. We compute the probability of overhearing by all node pairs that occur together on some valid source-destination route in the topology. To do 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.

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. The first case, where the packet is overheard but not delivered to its intended next hop, requires a significantly more sophistication from overhearing-detection algorithm – now $X_j$ has somehow communicate to $X_{i+1}$ or $X_i$ that it has overheard a packet and that $X_i$ does not need to retransmit. 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 practice. 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. 3.14 shows the CDF of the overhearing probabilities for paths between each pair of nodes in the network for Roofnet network. 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.

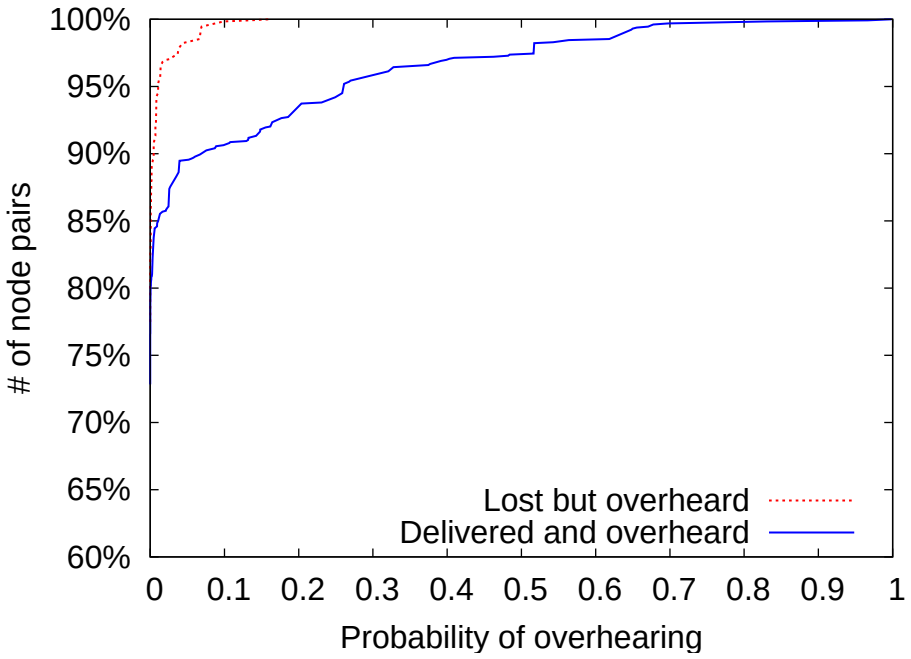
We also plot a number of transmissions results from our simulator. We do not plot a total airtime, as it depends on the specific details of wireless protocol, however the number of data packets transmitted is a good estimator that should be constant across all protocols. We show simulator results for Roofnet network.

3.15(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 overhearing, each path requires at least as many transmissions as there are hops, sometimes many more due to failed transmissions. Overhearing is able to significantly decrease the number of transmissions required, often quite dramatically. To clearly illustrate the performance improvement, 3.16 plots both the relative performance improvement for various path lengths at 1 Mbps and 11 Mbps.

At 1 Mbps, we are able to save over 20% of path transmissions for the median

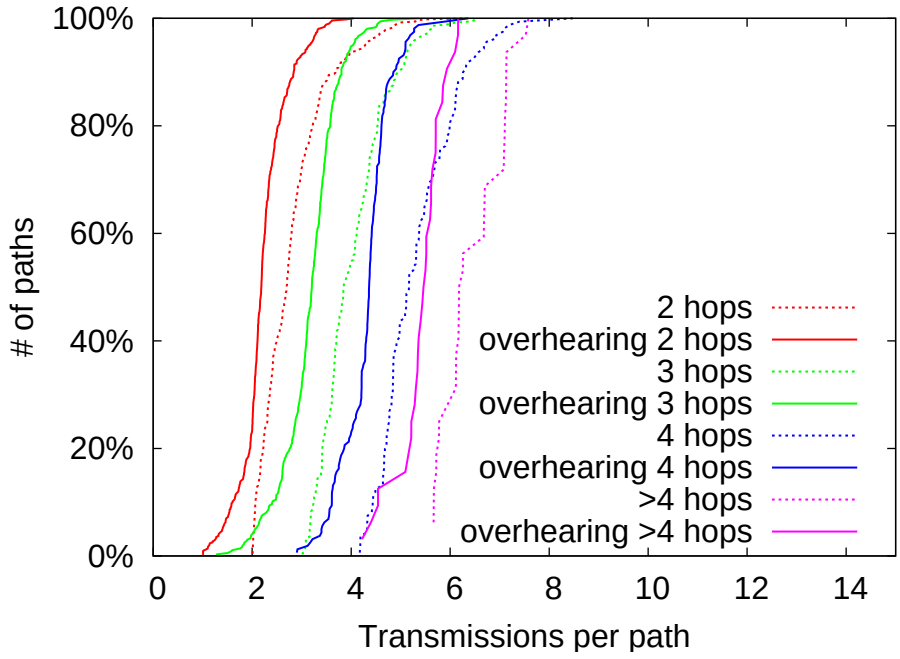



(a) 1 Mbps

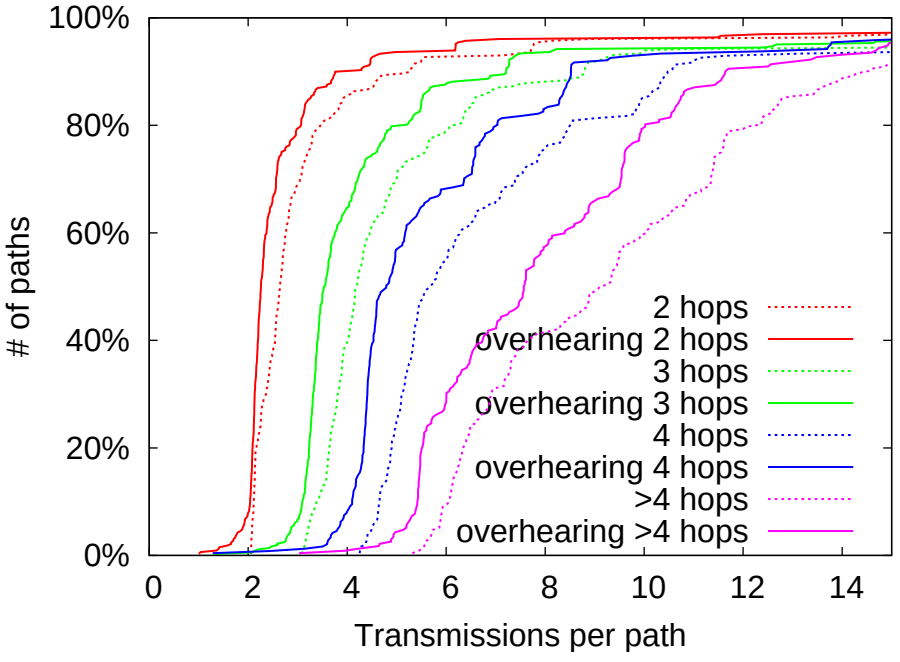



(b) 11 Mbps

Figure 3.14: 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%.

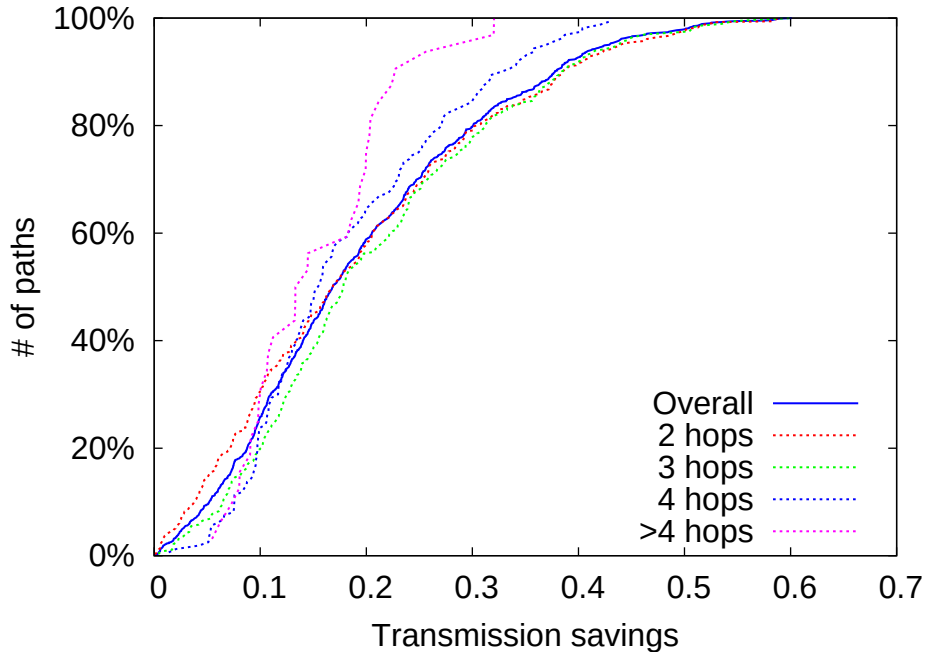



(a) 1 Mbps

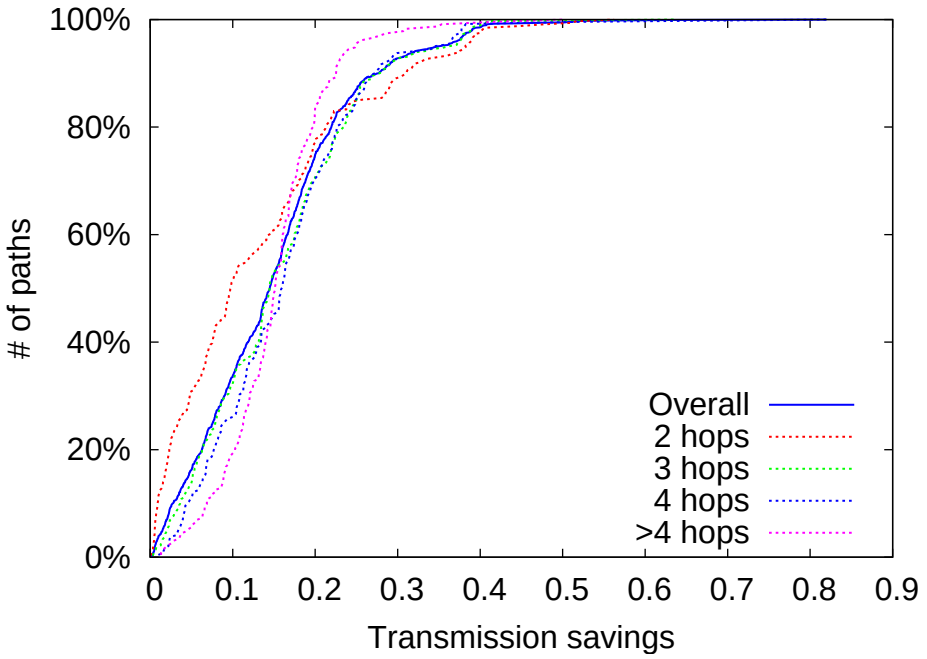



(b) 11 Mbps

Figure 3.15: The expected number of packet transmissions per ETT path with and without on-path overhearing




(a) 1 Mbps




(b) 11 Mbps

Figure 3.16: On-path overhearing 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.

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, algorithm provides at least 20% savings for only a quarter of all paths.

3.17, 3.18 and 3.19 shows the simulator results for off-path overhearing for all testbeds. There are two lines on each graph, which show the CDF of expected total transmission time per packet for the traditional routing (no overhearing) and for ExOR system. In all cases, ExOR shows a reduced total airtime. While these results can not be directly translated into an increase in goodput, they show that while the absolute transmission time is different for each testbed, one can expect to see some improvement from overhearing in all cases.

## 3.5 Taking advantage of overhearing

We have shown that many mesh networks have a potential for improvement if they can take advantage of overhearing. However, a practical implementation of an overhearing-aware systems needs to answer an three important questions: which packets to send, at what speed and when. These question is trivial in the traditional routing schemes, as they assume a presence of reverse channel that could be used to deliver an acknowledgment packets, determine optimal speed, and schedule forwarding time. However, overhearing-aware systems operate over several lossy links simultaneously, and thus do not have the luxury of well-defined return path.

We explore various ways of solving this problem in the remainder of the dissertation. In the next section, we introduce a simple on-path overhearing optimization that works by explicitly asking the receiver if the given packet was overheard. In the Chapters 5 and 6, we explore the performance of existing off-path optimization systems.

Chapter 3, in part, is a reprint of the material as it appears in the proceedings of the ACM/USENIX Symposium on Networked Systems Design and Implementation, 2008, Afanasyev, Mikhail; Andersen, David G.; Snoeren, Alex C.. The dissertation author was the primary investigator and author of this paper.

Chapter 3, in part, is a preprint of material that has been accepted for publication in the proceedings of the ACM SIGCOMM Conference on Internet Measurement,

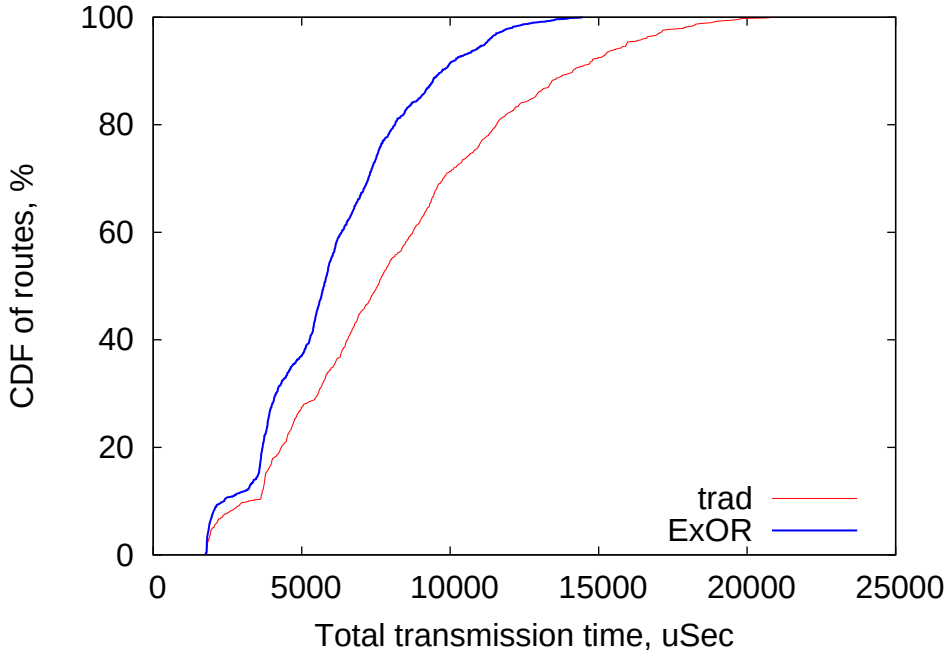



Figure 3.17: CDF of the total transmission time per path. Roofnet testbed

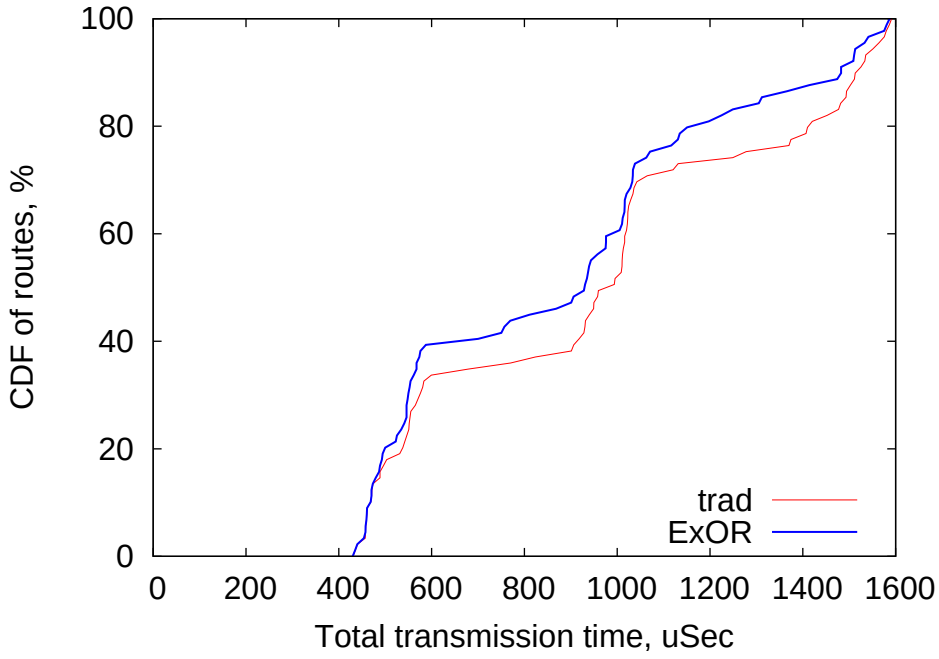
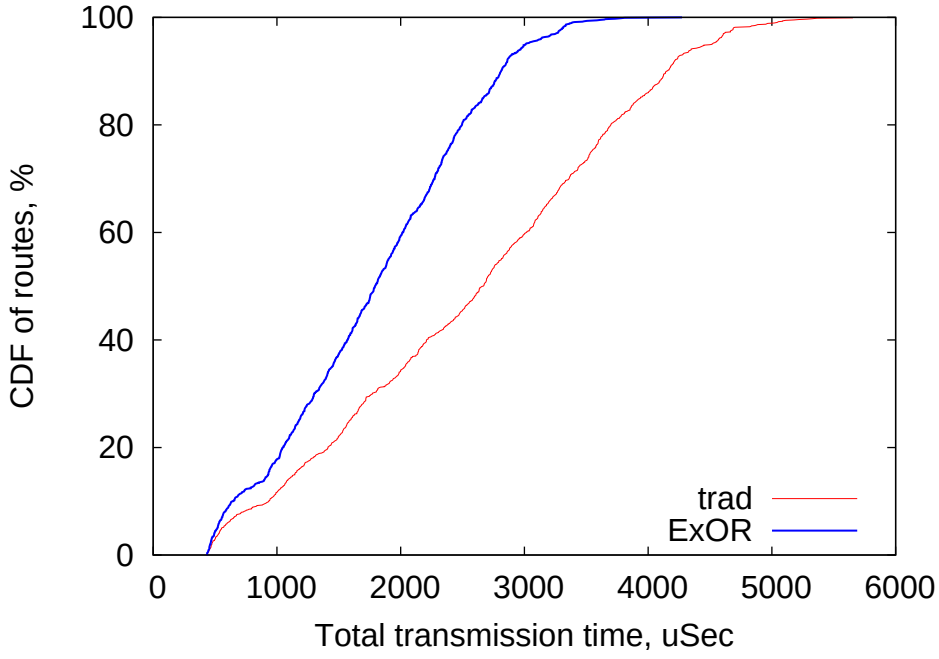
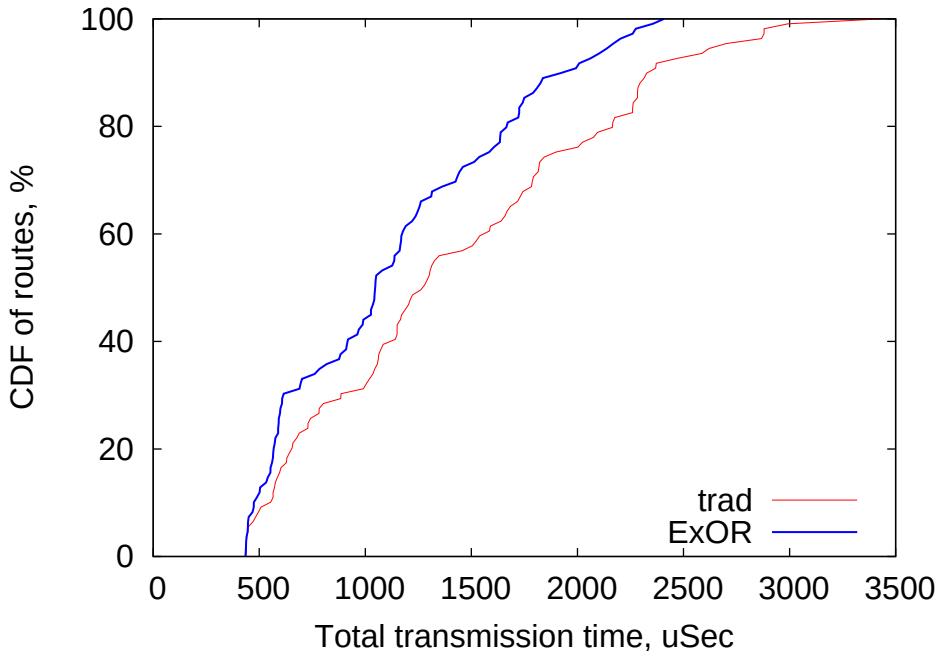



Figure 3.18: CDF of the total transmission time per path. ALIX network, full power




(a) Jigsaw testbed, whole building




(b) Jigsaw testbed, 2nd floor

Figure 3.19: CDF of the total transmission time per path. Jigsaw network, maximum transmission power.

2009, Afanasyev, Mikhail; Snoeren, Alex C. The dissertation author was the primary investigator and author of this paper.

Chapter 3, in part, is a reprint of the material as it appears in the proceedings for the ACM SIGCOMM Conference on Internet Measurement, 2008, Afanasyev, Mikhail; Chen, Tsuwei; Voelker, Geoffrey M.; Snoeren, Alex C. The dissertation author was the primary investigator and author of this paper.

# Chapter 4

# On-path overhearing

In this chapter, we propose a system that takes advantage of on-path overhearing. This system, which we call 'RTS-id', is a simple per-hop link-layer modification that takes advantage of overheard packets in a protocol and topology-independent manner that requires only the cooperation of adjacent nodes in a path.

The main advantage of RTS-id is the fact that it introduces few modifications to the regular 802.11 routing stack, and is backwards compatible with existing 802.11 hardware. This means that individual nodes can be upgraded by replacing the 802.11 driver and/or firmware, yet they will continue to inter-operate with legacy nodes. We verify that the RTS-id extension is ignored by hardware that does not support it with no ill effects. Furthermore, while substantially more modest than the bulk transfer improvements obtained by the overhearing-aware systems discussed in the next chapter, the gains we report are independent of transport-layer protocol: they are equally applicable to UDP and TCP.

## 4.1 Algorithm design

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.

As described in section 2.1.3, the 802.11 standard defines RTS/CTS as a mechanism which reduces the hidden terminal problems. It operates by having senders broadcast a 'request to send' (to a particular receiver) specifying the expected duration of the frame exchange. Original specification allowed only one possible answer: if the receiver received the packet, it replies with a 'clear to send' (CTS) frame containing the expected remaining duration of the frame exchange. RTS-id changes the algorithm in two small ways: first, the RTS packet now has short ID number which identifies the data packet associated with a given RTS/CTS request. Second, RTS-id adds an additional possible response to an RTS packet: if the data packet has been overheard, then the receiver can directly acknowledge the packet with a special 'CTS-ACK' frame. This frame will prevent the transmission of the data packet, but at the same time, it will signal the host that the transmission was successful, thus ensuring that the host will not attempt to retransmit the packet.

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.

### 4.1.1 Sender and receiver operation

RTS-id is an improvement which is intended to be used together with the existing mesh routing systems such as SRCR. Thus, every node may act as a sender and a receiver simultaneously. We study each of these roles separately.

RTS-id senders operate as shown in 4.1: they first decide whether to use RTS-id for a packet, based on the packet size and whether the use of RTS-id will benefit the performance. If so, they 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,

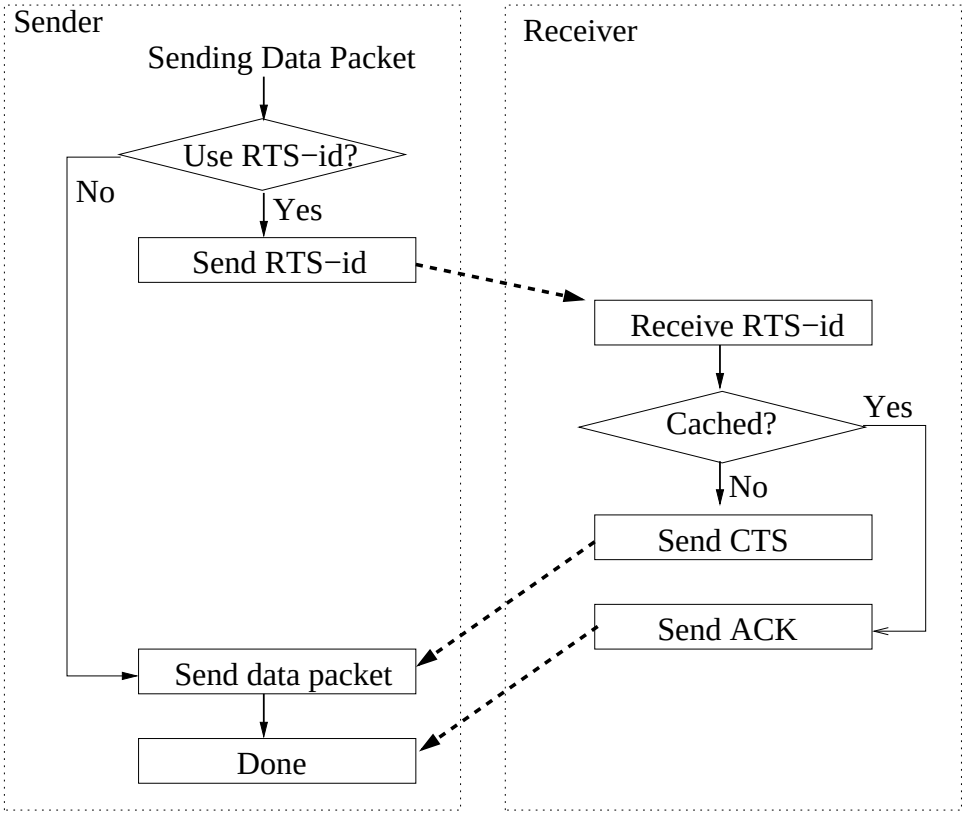



Figure 4.1: RTS-id operation. For clarity, this figure assumes that the sender does not fall back to normal RTS/CTS use.

nodes overhearing the RTS will only consider the channel reserved for the RTS-id/CTS exchange at this point.

When a node receives a normal data frame, it operates according to the flowchart in 4.2. 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.

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

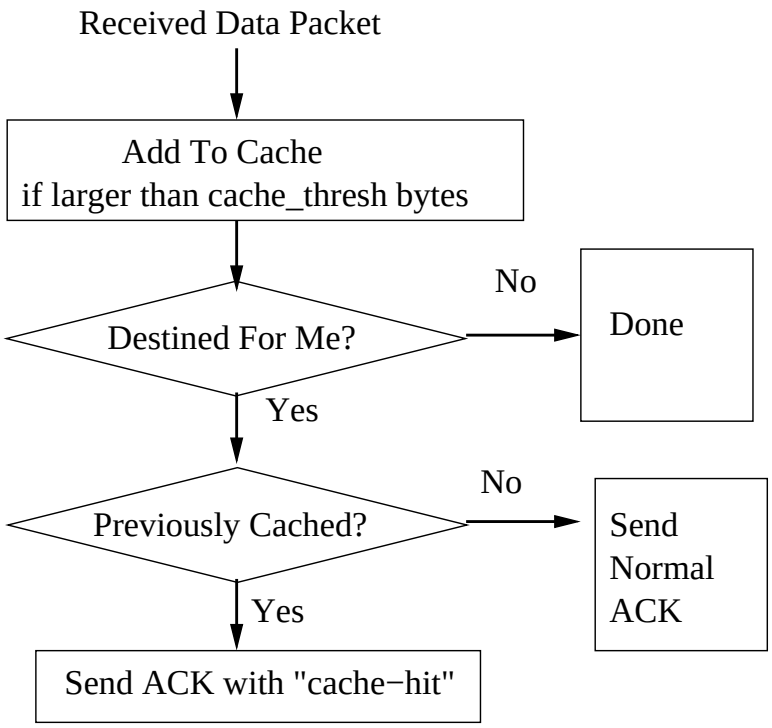



Figure 4.2: Received packet processing.

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.

### 4.1.2 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.

### 4.1.3 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 a low rate, (1 Mbps for 802.11b or 802.11g networks, and 6 or 12 Mbps for 802.11a networks), 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 (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).

Before transmitting 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.

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

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 if it is not currently caching packets.

## 4.2 Implementation

In order to verify the practicality of the solution, we implement RTS-id on real hardware and verify its operation. The main complexity of the implementation is the extremely short time that a station has to send a CTS or CTS-ACK, which is smaller than interrupt latency, and rules out the use of regular 802.11 wireless cards for our prototype without access to their firmware. At the time of writing, no commercial 802.11 cards provide access to their firmware.

In order to work around the high latency of regular 802.11 cards, we implement RTS-id using the CalRadio 1.0 platform designed and manufactured by CalIT2 [Cal], shown in 4.3. The CalRadio is a relatively low-cost software radio platform consisting of an ARM processor, an on-board Texas Instruments DSP, and a Prism 802.11b baseband


Figure 4.3: The CalRadio 1.0 platform.

| | | | | | | |
|---|---|---|---|---|---|---|
| RTS: | FC (2) | Dur (2) | Source (6) | Dest (6) | FCS (4) | |
| RTS-id: | FC (2) | Dur (2) | Source (6) | Dest (6) | FCS (4) | ID (8) |

Figure 4.4: RTS and RTS-id packet formats.

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 ($\mu$CLinux [Emb] 2.4.19) and user-level programs run on the ARM.

### 4.2.1 Packet details

The RTS-id packet is a simple extension to the standard 802.11 RTS packet as shown in 4.4. 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. We have verified that existing 802.11 devices (specifically, the Atheros chip sets), properly decode the RTS-id frame as an RTS. In the worst case, non-compliant stations will simply discard the seemingly mal-formed RTS-id frame with no ill effects. 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 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 wastes 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.

### 4.2.2 Packet caching and RTS-id

According to the 802.11 specification, a station must respond within 10 microseconds to an RTS request. To inter-operate 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 [SPS+02] 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.

### 4.2.3 Test-bed deployment

Our current test-bed consists of three CalRadio devices. Because of the small size of the test-bed, and the limited functionality of the CalRadio devices, we do not perform extensive measurements of this testbed at the different rates, and do not include it in the list of our data sets.

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

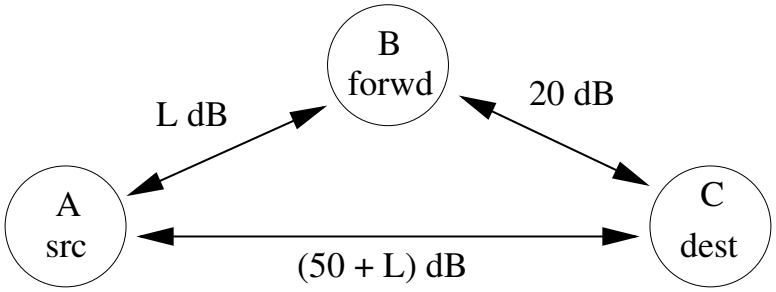



Figure 4.5: Testbed system for RTS-id

impacts only the RTS/CTS exchange, as the successful completion of such an exchange will reserve the channel for data transmission.

### 4.2.4 Experimental Setup

The small size of our CalRadio testbed limits the scenarios which could be verified. We attempt to simulate a typical three-station overhearing case, shown in 4.5. We set 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 forwarder 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 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

Table 4.1: Experimental results from the CalRadio test-bed.

| **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** | | | | |

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.

### 4.2.5 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.

4.2.4 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.

### 4.2.6 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 4.1.3). 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 4.2.4, 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, 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 $C$. 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.

## 4.3 Other networks

Due to the limited availability of CalRadios, we use our simulator to evaluate the effectiveness of RTS-id in a multi-hop mesh network. Since our implementation supports

802.11b only, we use Roofnet, an 802.11b network, as the dataset for our simulator.

In this scenario, the benefits of RTS-id 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 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 [ABB$^+$04]. 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.

RTS-id exactly implements the on-path overhearing as described in previous chapter, so 3.15(a), which plots the expected number of transmissions for all-pairs paths of length greater than one, applies to RTS-id. 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 decrease the number of transmissions required, often quite dramatically. To clearly illustrate the performance improvement of RTS-id, 3.16 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.

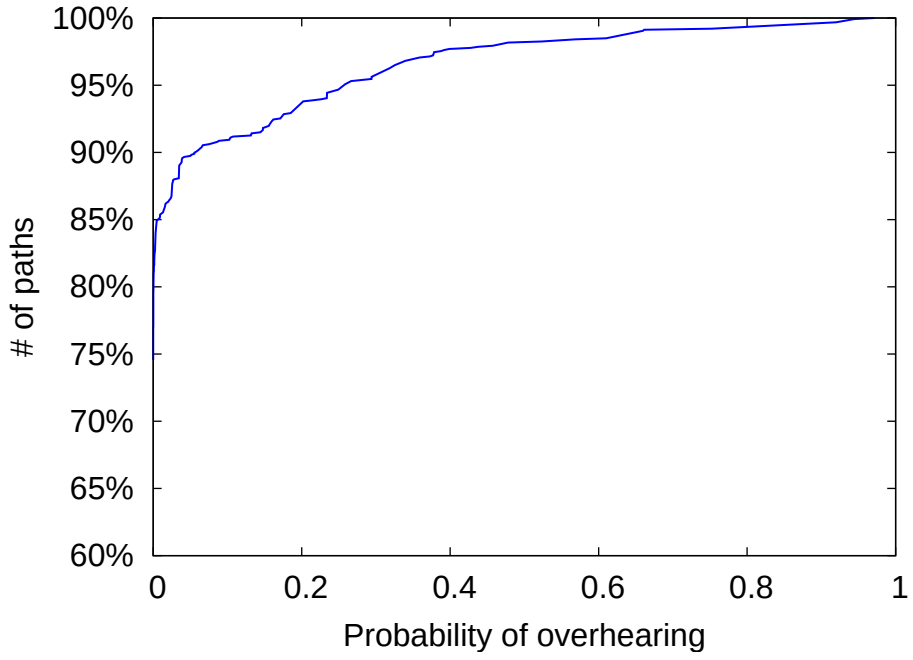
### Rate adaptation

As the previous results show, 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 [Bic05, BABM05]. 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.

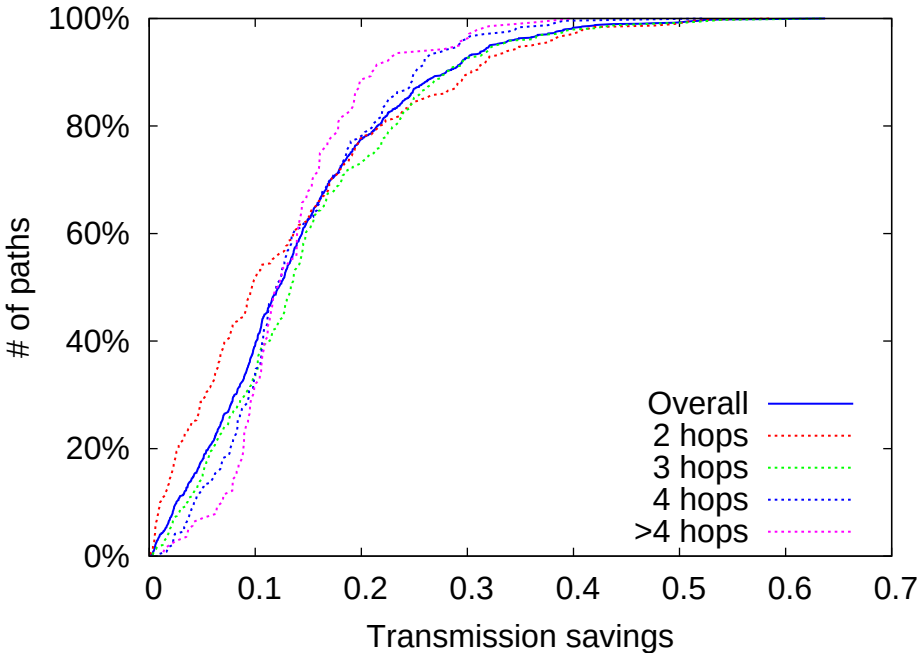
4.6 plots both the overhearing prevalence (c.f. 3.14) and the relative performance improvement versus ETT (c.f. 3.16) with dynamic rate adaptation. It turns out that most links in our dataset select the 11 Mbps transmit rate, so the overhearing is closer to that observed with a constant 11-Mbps transmit rate than a 1-Mbps transmit rate, resulting in similar savings (interestingly, its designers note that Roofnet generally transmits at 5.5 Mbps in practice [BABM05], so we are likely understating the potential savings.) In particular, RTS-id provides more than 20% savings for one quarter of all routes, and over 35% savings for the most-improved 5%.

### 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. 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. 4.7 shows the same data as 4.6(b), except that it contains only




(a) Overhearing frequency




(b) Relative performance improvement

Figure 4.6: The impact of rate adaptation. The first graph shows the overhearing prevalence (c.f. 3.14), and the second shows the relative performance improvement versus ETT.

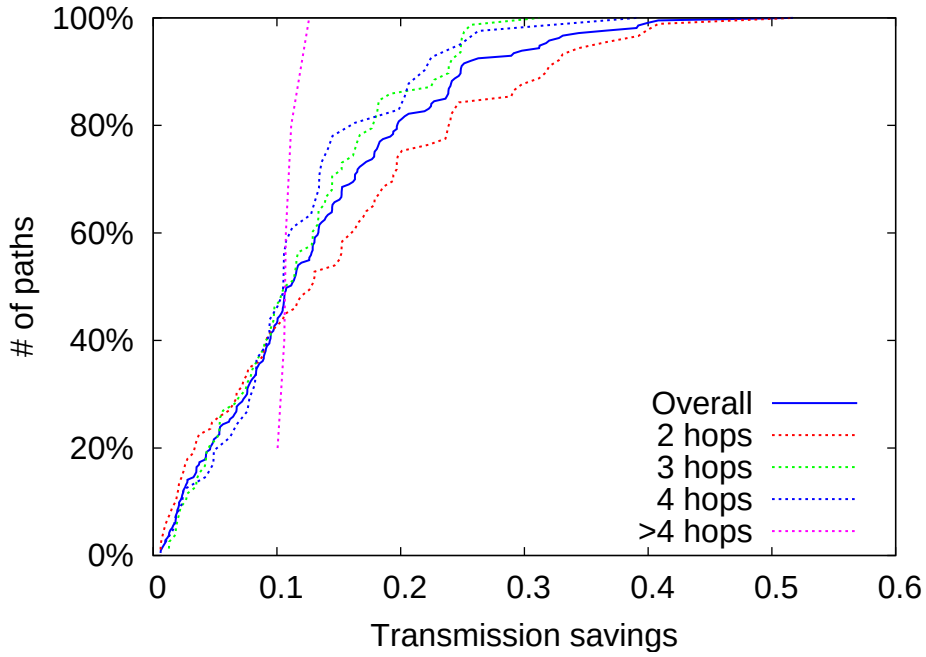



Figure 4.7: The relative performance improvement versus ETT for paths leading to or from a Roofnet gateway.

gateway routes. The overall distribution of savings is roughly unchanged.

### 4.3.1 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.

For example, the Roofnet network computes its metrics using only ten 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,

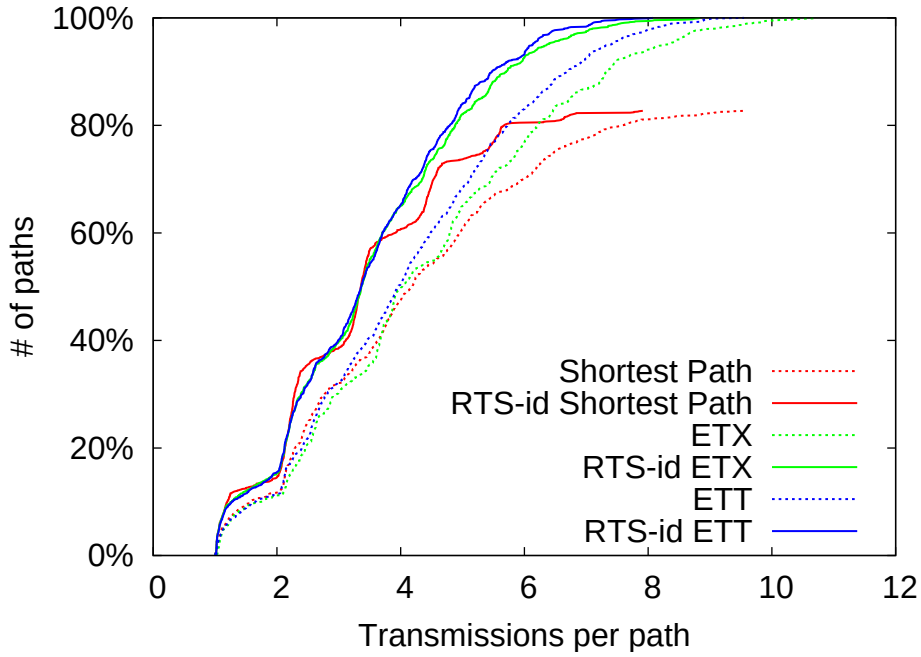



Figure 4.8: A variety of routing protocols with and without RTS-id, all using SampleRate to select link transmission rates.

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.

4.8 shows the performance of three routing protocols, ETT (c.f. 4.6), 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.

### 4.3.2 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 [BABM05],

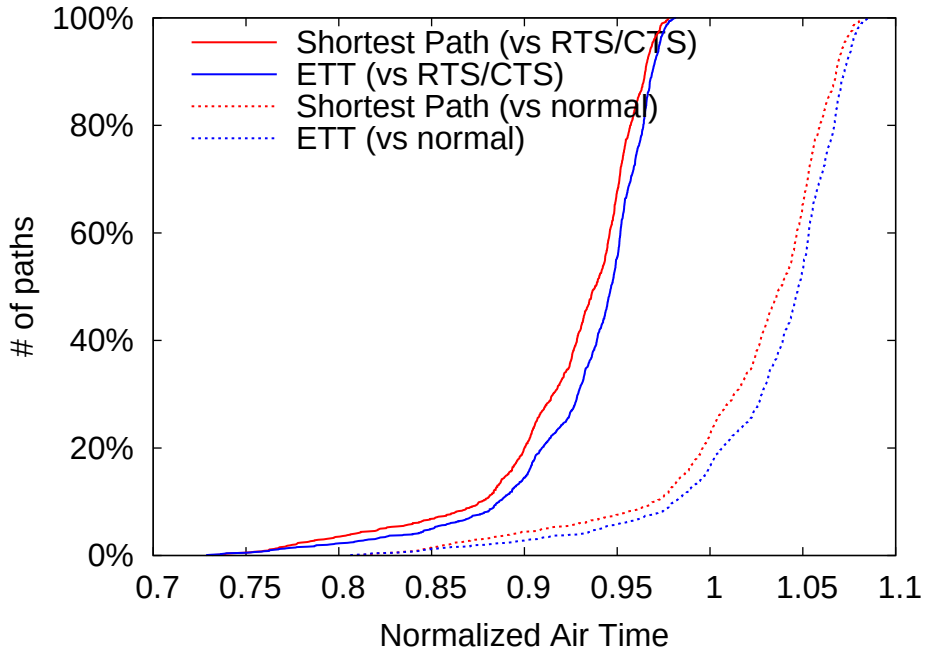



Figure 4.9: The normalized total path transmission time for RTS-id, with and without RTS/CTS.

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.

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. 4.9 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 4.8.)

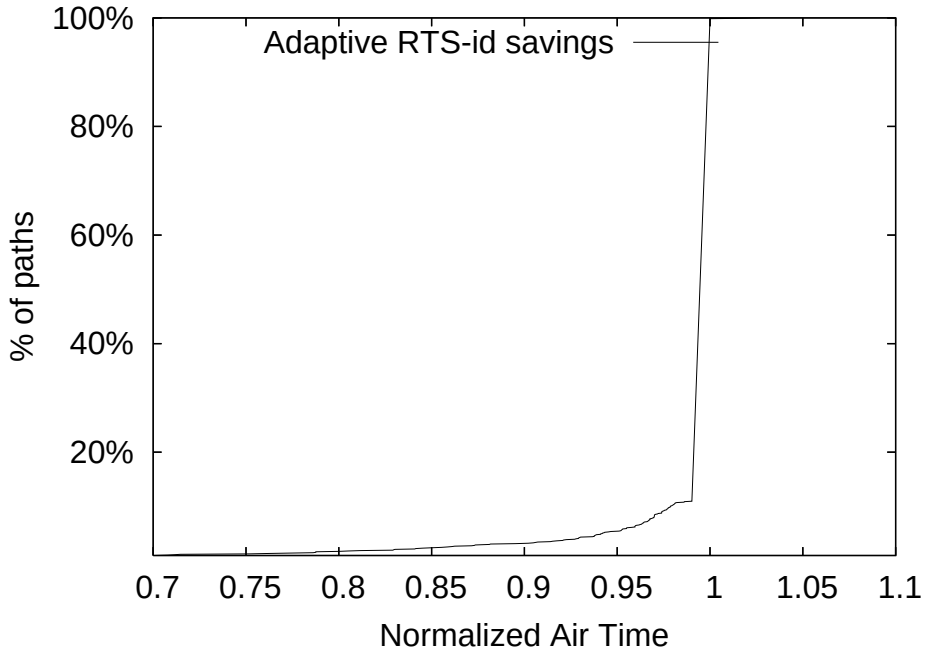



Figure 4.10: The normalized air time of adaptive RTS-id vs. a network that does not natively use RTS/CTS.

Adaptively enabling RTS-id as described in Section 4.1.3 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. 4.10 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 4.9, adaptive RTS-id never *harms* transmission time.

## 4.4 Summary

In this section, we presented RTS-id, a system that directly implements on-path overhearing, while remaining compatible with existing 802.11 devices. We show the operation of the algorithm and implement it in the physical hardware. Due to tight timing

constraints, we used a CalRadio, a software radio platform, in our implementation. Using a synthetic testbed consisting of three CalRadio nodes, we verified that RTS-id can save transmissions when the overhearing is present, and that it is compatible with existing 802.11 devices.

Due to the small size of the CalRadio testbed, we used the simulator developed in Section 3.4 in order to estimate the potential gain from the RTS-id system. The simulation of the Roofnet dataset shows that a reduction of more than 20% in the number of transmissions is possible for one quarter of the routes. An interesting property of the RTS-id is that it increases tolerance of sub-optimal routing algorithm – even a simple protocol like 'shortest path', when combined with RTS-id, performs as well as ETT. However, RTS-id introduces an additional overhead by required the stations to be using RTS/CTS. While the algorithm is adaptively enabled, thus never increasing transmission time, the resulting overhead results in overall improvements of 10% in airtime for about 5% of the paths.

While the general premise of RTS-id works, the resulting time improvements are significantly smaller than the reduction in number of transmitted packets, suggesting that the overhead introduced by per-hop queries is significant. Thus, in the next chapter we turn to off-path overhearing networks, which do not need to be compatible with existing networks, and can use more efficient ways to collect information about overheard packets.

Chapters 4, in part, is a reprint of the material as it appears in the proceedings of the ACM/USENIX Symposium on Networked Systems Design and Implementation, 2008, Afanasyev, Mikhail; Andersen, David G.; Snoeren, Alex C.. The dissertation author was the primary investigator and author of this paper.

# Chapter 5

# Off-path overhearing

In the previous chapter, we have shown how to exploit on-path overhearing in traditional mesh routing systems. However, such systems do not take full advantage of the overhearing present in the system. In order to maximize performance, we must use systems that implement off-path overhearing, such as ExOR and MORE. In this chapter, we investigate a way to optimize off-path overhearing systems. Specifically, we focus on the selection of the transmission rates: we present a rate-selecting algorithm, and implement it on both ExOR and MORE systems.

## 5.1 Rate selection

The selection of transmission rates in overhearing-aware systems has been little studied. Existing off-path overhearing algorithms use a single, fixed rate for all nodes in the network — 1 Mbps in the original ExOR work, and 11 Mbps for MORE (although Chachulski *et al.* also publish results for ExOR at 11 Mbps)—and defer issues of bitrate selection to future work [BM05, CJKK07]. Other systems, like our RTS-id work, or ExOR extensions in Chachulski's master thesis [Cha07] leveraged existing rate-adaptation techniques designed for non-overhearing aware systems to determine appropriate bitrate.

In either case, the resulting transmissions are likely to be sub-optimal in the global sense. As we have seen in 3.3.1, the transmission range, and therefore number of recipients, varies greatly depending on the rate, thus we would expect that the selection of

transmission rate would impact overall performance. We consider how one might select more efficient bitrates to improve throughput when possible. We begin by considering the case of an 802.11b network, as used in previous work, and then present 'modrate', an approach better suited for modern, 802.11a/b/g networks.

### 5.1.1 Fixed range

From the section 3.3.1, we see that all of the 802.11b rates provide approximately the same range in our testbeds, so if we consider an 802.11b-only transmitter, it likely suffices to select the bitrate for each node independently—as in traditional routing algorithms [Bic05]—since the transmitting node's choice is unlikely to have a significant impact on the set of forwarder nodes that will receive the batch fragment. Indeed, it appears even for 802.11g the same can be said for most speeds—all but the highest three, in fact, when transmitting at the highest power in our test bed. In other words, each transmitter can disregard the presence (or absence) of overhearing, and focus on the natural goal of selecting the bitrate that minimizes the remaining expected transmission time (ETT) of the batch fragment to its ultimate destination.

Happily, this is the same goal in traditional routing: Roofnet's Srcr routing protocol [BABM05] selects a shortest path in terms of ETT presuming each node transmits to the next hop at its optimal bitrate. In fact, ExOR uses ETT to determine the priority order of the forwarding list, so it will automatically incorporate any improvements due to bitrate selection into its forwarding algorithm. Extending the notation of Chachulski *et al.* [CJKK07], let $\epsilon_{ij}^r$ denote the the expected loss probability when node $i$ transmits to note $j$ at rate $r$. If we denote the time taken to transmit a packet at rate $r$ as $T(r)$ (a constant value regardless of the nodes in question), we can write

$$ETT_{ij}^r = \frac{T(r)}{1 - \epsilon_{ij}^r}.$$

Because ExOR only transmits packets for a single destination in any given batch, a node can consider each batch fragment transmission independently. In particular, for a batch fragment originating at $s$ destined to node $d$, forwarding node $i$ selects the bitrate as follows. Assume node $j$ is the next hop on the optimal Srcr-computed route from $i$ to $d$. (Note that, due to overhearing, $i$ may not have been on the original Srcr route from $s$

to $d$.) Then, $i$ selects a bitrate $r$ that minimizes $ETT_{ij}^r$:

$$r(i,j) = \arg\min_{r \in Rates_i} \left( \frac{T(r)}{1 - \epsilon_{ij}^r} \right) \tag{5.1}$$

where $Rates_j$ is the set of bitrates available at node $i$.

### 5.1.2 Modrate

Modern systems typically use 802.11a/b/g radios, however, which have a direct correlation between transmission rate and average reception range. Thus, it is important to consider the potential impact of decreased overhearing opportunities when choosing an appropriate bitrate. We propose a rate-selection algorithm called 'modrate' that jointly optimizes next-hop throughput and overhearing prevalence. Said another way, instead of trying to optimize for the expected single (Srcr) path as above, the rate instead is selected to minimize the expected transmission time over all useful paths including those that arise from overhearing.

In ExOR, a packet could be received at multiple destinations, but will be processed first by the destination with lowest $ETT$ to the destination; to ease discussion we order all nodes in terms of their ETT to $d$, $s \geq i \geq j \geq d = 0$, Now, rather than adjusting the bitrate in view of just the next Srcr hop, we seek to consider the bitrate in view of the furthest (i.e., closest to $d$) recipient. If we define $\rho_{ij}^r$ as probability that the furthest recipient of the packet sent from $i$ at rate $r$ will be $j$, we can compute the expected total transmission time for the packets which are received by $j$ as:

$$(T(r) + ETT_{jd}) \cdot \rho_{ij}^r \tag{5.2}$$

This formula is valid even for the transmissions which were not received by any other nodes: in this case, we assume that $i = j$, or that the sender always hears its own packet. Given that, we can determine the optimal bitrate $r^*$ as

$$r^*(i) = \arg\min_{r \in Rates_i} \left( \sum_{j \leq i} \left[ (T(r) + ETT_{jd}) \cdot \rho_{ij}^r \right] \right) \tag{5.3}$$

How could we calculate $\rho_{ij}^r$? One way is to assume that all transmission probabilities are independent, a frequent assumption in the literature [CJKK07, RMRW06].

Then, we just need to calculate the probability that a transmission would be received by $j$ and not by any $k < j$:

$$\rho_{ij}^r = (1 - \epsilon_{ij}^r) \prod_{k<j} \epsilon_{ik}^r.$$

An alternative method is to not rely on independence, and instead measure probability of reception for all possible sets of receivers. We adopt the latter approach in our evaluation. During the measurements, we calculate and store probability that a packet sent by $i$ will be received by some subset of nodes $A$ for every subset of nodes: $Pr\left[i \rightarrow A\right], A \subseteq \mathcal{P}(nodes)$. Then, $\rho_{ij}^r$ can be computed directly:

$$\rho_{ij}^r = Pr\left[i \rightarrow A \mid (j \in A) \wedge (k \notin A \forall k < j)\right]$$

## 5.2 Experimental setup

In order to verify the advantages of the modrate, we implement both ExOR and MORE routing protocols and evaluate them on our testbeds.

### 5.2.1 Testbeds

We conduct our tests on both ALIX and Jigsaw testbeds. We drive the experiments with a centralized controller that has wired connectivity to each node in the network. We begin experiments by performing the network-wide link survey as described in Section 3.2.2 in order to produce appropriate routing and speed information, which we calculate using our simulator described in Section 3.4. This information is then communicated to the stations; thus, the stations themselves do not run any routing code, ensuring that all protocols operate with the same routes.

When conducting experiments comparing various protocols, we run all the protocols under test in sequence at each pair of source/destination nodes, before moving to the next pair. By doing so, we roughly equalize any impact of out-of-date delivery probabilities. Additionally, for long-running experiments, we update our estimates of the transmission probabilities and re-calculate routes every twenty minutes.

### 5.2.2 Traditional routing

As a baseline, we measure the throughput of traditional, single-path routing that employs both link-layer and end-to-end acknowledgments to ensure reliable delivery. In order to evaluate the most prevalent scenario in today's wireless networks—TCP data being sent over a single, rate-adapting path—we implement a simple Srcr forwarder. Our Srcr forwarder uses the link probabilities calculated by our measurement procedure and selects routes using a modified ETT metric that accounts for asymmetric links. We assume that ACKs are always sent at the lowest speed for the 802.11 protocol in use (1 or 6 Mbps). This is a Click-based system which forwards all packets between two hosts along a predefined path, as provided by the experiment controller. We use the regular Linux 2.4 kernel TCP stack without modifications, and the `ttcp` application to measure the time it takes to transfer 1 megabyte of data. We refer to this mechanism in all of our results as 'trad-TCP.'

### 5.2.3 ExOR implementation

We were unable to obtain the original ExOR implementation, so were forced to reimplement it. Because we are unsure whether we were able to faithfully replicate the exact behavior of the transmission timer, we instead implement a scheduling "oracle" within the control server: Once a forwarding node is done transmitting a batch fragment, it notifies the control server over the wired network. The server then notifies the next node in the batch's forwarding list to begin transmission. Should that node not have any remaining packets to send, it may send a set of empty packets to propagate the batch map; regardless, it notifies the server when finished. Communication with the scheduling oracle takes time, so each station keeps track of how long it spent transmitting the batch fragment. Once the batch is successfully received at the destination, all round times are added together to get the actual transmission time without oracle communication overhead.

### 5.2.4 MORE module

We implement the MORE algorithm using the publicly available MORE source code. The implementation is completely separate from our ExOR code base, so by default, it uses its own code to calculate routes. To increase consistency in route selection, we use our simulator to calculate the list of forwarder nodes and their transmission speeds with and without modrate. We then instrument the public MORE code to fix the forwarder list and forwarder rates to the specified list. This way, we still use the original MORE code for MORE-specific calculations, such as expected input/output ratio, and we use our simulator for the task of selecting off-path overhearing routes, with or without modrate support.

It is worth noting that MORE is substantially more CPU-intensive than any of the other protocols we evaluate. Even with data calculations disabled (i.e., only verification of innovativeness was done), it still requires more CPU power than the Jigsaw testbed can provide. Thus, we report MORE results only for the ALIX testbed.

As originally described, MORE uses a single, fixed link speed for all nodes in the network. The publicly available implementation selects a link-local optimal speed based upon the ETX metric in a manner similar to Equation 5.1 [Cha07]. While it would be possible to use this implementation, we instead elect to use our simulator to select speeds, as we believe it provides a more consistent comparisons between MORE and ExOR.

According to the original paper [CJKK07], MORE prunes the potential forwarder nodes in a simular fashion to ExOR: nodes which are expected to transmit less than 10% of the total number of packets are removed. The available implementation, however, implements a more advanced algorithm, which involves pruning on the basis of both total packets received, as well as on the received/transmitted packet ratio. We find out that the selection of pruning algorithm parameters can have a large effect on the performance. For more consistent comparisons, we consider the pruning algorithm described in the paper. A rigorous exploration of the pruning algorithms is interesting, but beyond the scope of this dissertation.

## 5.3 Results

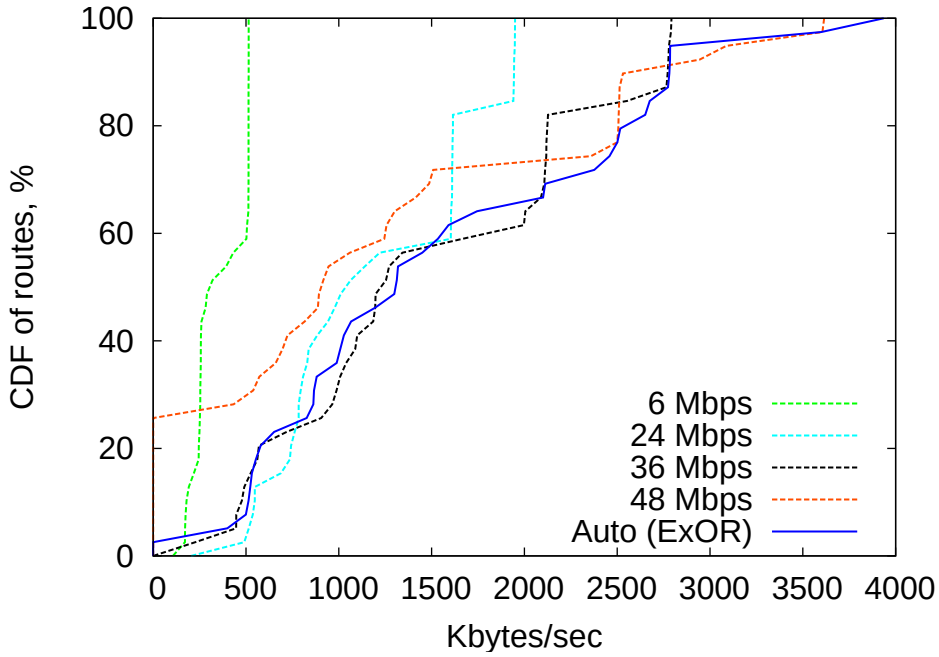
In order to facilitate direct comparisons, our experimental methodology largely follows those of the original ExOR [BM05] and MORE [CJKK07] papers, although with several slight differences. For each source/destination pair, we transfer a 1.5-megabyte file, consisting of 10 batches of 100 packets, each containing 1,500 bytes (c.f. 1,024 in the original ExOR paper) of payload. As is customary, we do not implement traditional routing of the final 10%; instead, we stop and report the throughput when the destination has received 90% of the packets in each batch. Thus, our experiments result in ten separate transmission times, each corresponding to the successful reception of at least 90% of a 150-KB chunk of the original file.

### 5.3.1 Overhearing-oblivious rate selection

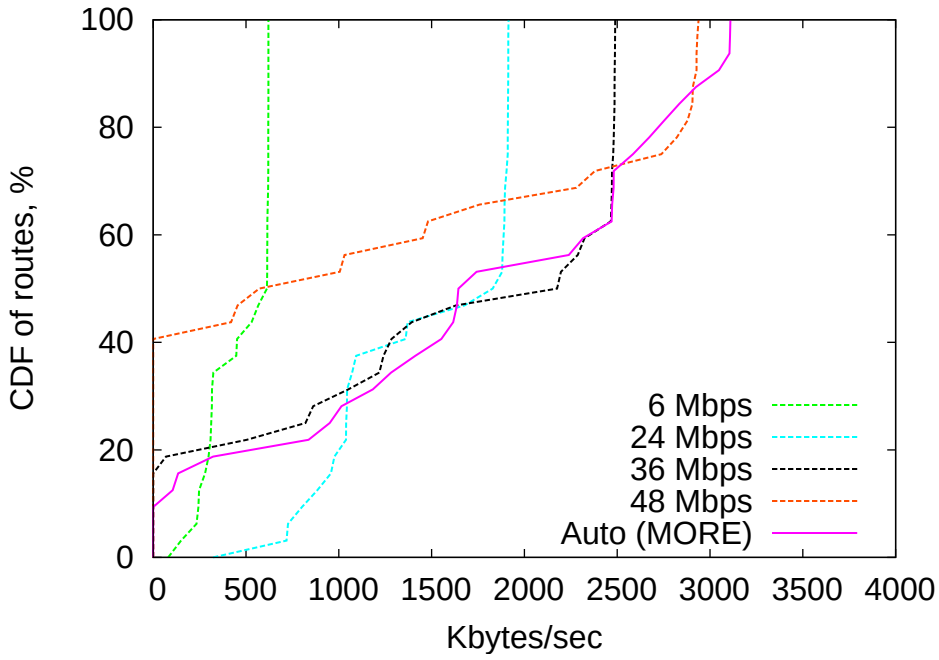
To begin, we consider the performance of ExOR as originally described by Biswas and Morris [BM05]. In particular, we assume that all nodes in the network use a single, fixed speed. 3.6 suggests that the performance in the ALIX testbed with bit rates less than 24 Mbps are likely to be gated by link speeds rather than reception rates. Increasing speed beyond 24 Mbps, however, seems likely to markedly decrease the degree of connectivity in the network, potentially harming performance.

5.1(a) plots the performance of four fixed speeds, 6, 24, 36, and 48 Mbps on the ALIX testbed when all nodes transmit at maximum (60) power, roughly equivalent to 18 dBm. Recall that throughput is the total number of bytes delivered over 10 independent batches divided by the cumulative time required. For all of the graphs in this section, we report on the performance of 40 randomly selected node pairs among the 100 possible combinations. We bias the 40 routes to include longer-hop paths if possible, as one-hop paths tend to be uninteresting. None of our paths are longer than four hops. To select the 40 random paths, we first select up to 10 paths of each length—four, three, two, and one hops—and then fill in the remainder with randomly selected paths if we do not have enough of a particular length.

While performance generally improves with higher links speeds, the network becomes disconnected at 48 Mbps and no route exists for 11 of the selected route pairs;




(a) ExOR




(b) MORE

Figure 5.1: Throughput of ExOR and MORE with automatic and various fixed rate selections. ALIX network with full power.

this phenomenon is even more pronounced at 54 Mbps. The globally optimal rate will obviously vary from network to network, and likely even over time. Instead, we see that an automatic rate assignment that selects the locally optimal speed for each link (neglecting overhearing potential) as specified in Equation 5.1 generally outperforms any fixed speed selection. We refer to this automatic-rate-assignment ExOR implementation as 'ExOR' in all subsequent graphs.

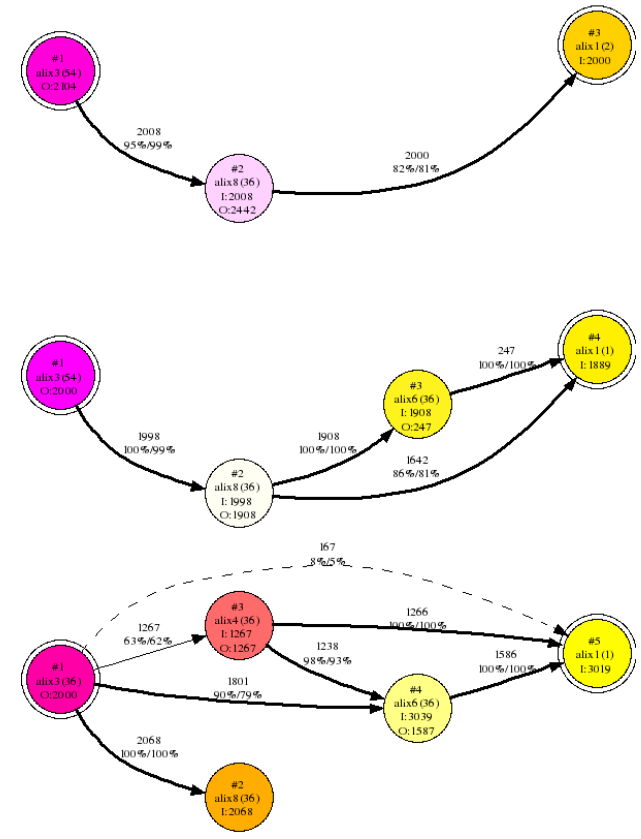
### 5.3.2 ExOR evaluation

We now consider the additional performance gains from considering the impact of link rates on overhearing opportunities. In particular, we enhance ExOR with the modrate algorithm described in Equation 5.3 and conduct a second experiment on the ALIX testbed at a highest power level.

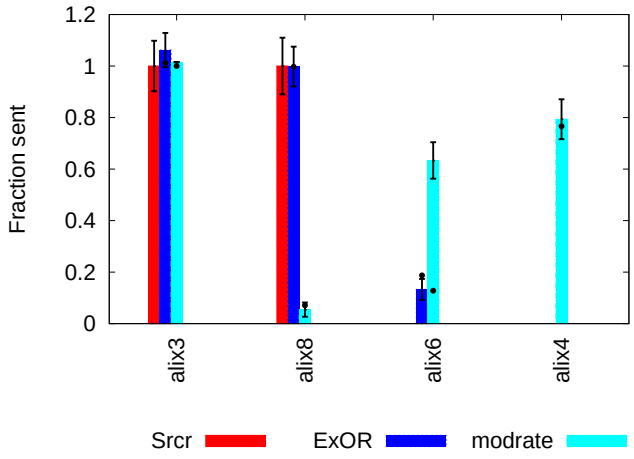
#### An example route

5.2(a) diagrams one particular route where modrate dramatically changes the forwarding behavior. The top portion shows a two-hop Srcr route from *alix3* to *alix1* that uses *alix8* as an intermediary when nodes transmit at full power; *alix3* transmits at 54 Mbps, while *alix8* selects 36 Mbps. In addition to link speed, each node is annotated with the number of packets it transmits (O) and receives (I). The links are labeled with both the number of packets successfully transferred as well as the experimental and predicted (by the survey) reception rate. The middle portion shows how ExOR uses the route, leveraging overhearing by node *alix6* to assist with packets on the second hop. In one particular batch, *alix6* overhears all of the packets transmitted by *alix8*, and is able to deliver 247 of them to *alix1*, saving retransmissions.

Finally, the bottom portion of the figure shows how modrate enhances overhearing by decreasing the transmission speeds of *alix3* (from 54 to 36 Mbps). By doing so, it introduces three new overhearing opportunities: First, the destination is able to directly receive packets approximately 8% of the time. Second, *alix4* and *alix6*, which are closer to the destination than *alix8*, are now able to overhear transmissions. In fact, between the two of these nodes, they are able to forward all of the packets to the destination, freeing




(a)




(b)

Figure 5.2: An example route using different algorithms.

the original intermediate hop, *alix8*, from forwarding any packets at all in this particular batch.

The same effect is presented in another format in 5.2(b), which depicts the same transfer from *alix1* to *alix3*. The $y$ axis indicates the number of packets a station transmits normalized to the total number of packets transmitted along the route (although we do not plot link-layer transmissions used by the 'hop-by-hop' protocol). Node *alix3* is the source, and therefore has to transmit all packets at least once. The four transmitting nodes are laid out along the $x$ axis, ordered in increasing proximity to the destination. Bars correspond to the average among all 10 batches, while the dots indicate the performance predicted by the measured reception rates.

### 5.3.3 MORE evaluation

We have enhanced MORE with the modrate algorithm, and run the set of experiments on the ALIX testbed at the highest power. 5.1(b) shows the performance of the MORE protocol when all nodes run at the same, fixed speed.

Our immediate observation on the MORE algorithm is the lack of any rate control – the source, for example, transmits packets any time the air is free, thus increasing the probability of packet collisions and exacerbating any hidden-terminal problems that the system might have. We find that practically speaking, this causes serious performance degradation on some paths.

This effect is already visible on 5.1, if one compares ExOR and MORE at, for example, a 36 Mbps transmission rate. The median throughput is much higher for MORE – 2.2 megabytes per second for MORE versus 1.2 megabytes per second for ExOR, but the median variation is also much higher: the number of routes that transfer less than 0.1 megabytes per second is 2% for ExOR and 18% for MORE.

### 5.3.4 Network-wide performance

5.3 plots throughput in the same fashion as 5.1, comparing ExOR, ExOR with modrate, MORE, MORE with modrate, and traditional routing. Despite some significant changes in speed selections, the overall difference in performance between ExOR and

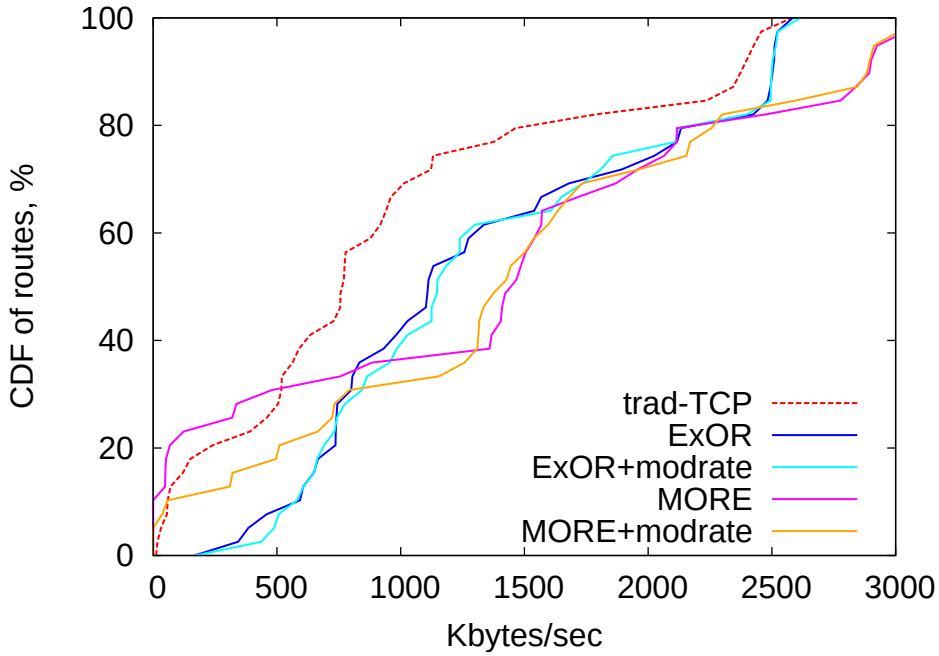



Figure 5.3: CDF of path throughputs for ExOR, MORE, and traditional routing. ALIX network, full power.

ExOR+modrate are slight. 5.4 accentuates the differences by plotting the per-route throughput normalized to that of ExOR; a positive difference means that the performance is better than ExOR, and negative implies less throughput.

The differences between MORE and MORE+modrate lines are bigger, but some of them are caused by the degradation from hidden terminals. In order to understand the real impact of modrate on MORE, we plot the per-route relative differences against traditional routing in 5.5. The negative speed differences are likely to indicate a hidden terminal problem. The positive speed differences show that modrate works well with MORE.

In theory, the performance of the modrate-enabled algorithm should be strictly better than the regular one, but some variance is to be expected in practice due to time-varying delivery probabilities, and has been reported many times in the literature [CJKK07, WYLB06]. In this experiment, modrate+ExOR manages to equal or best the link-local scheme on all but 10% of the routes, and is rarely more than 10% worse. In the ExOR configuration, modrate provides limited benefit for the vast majority of routes,

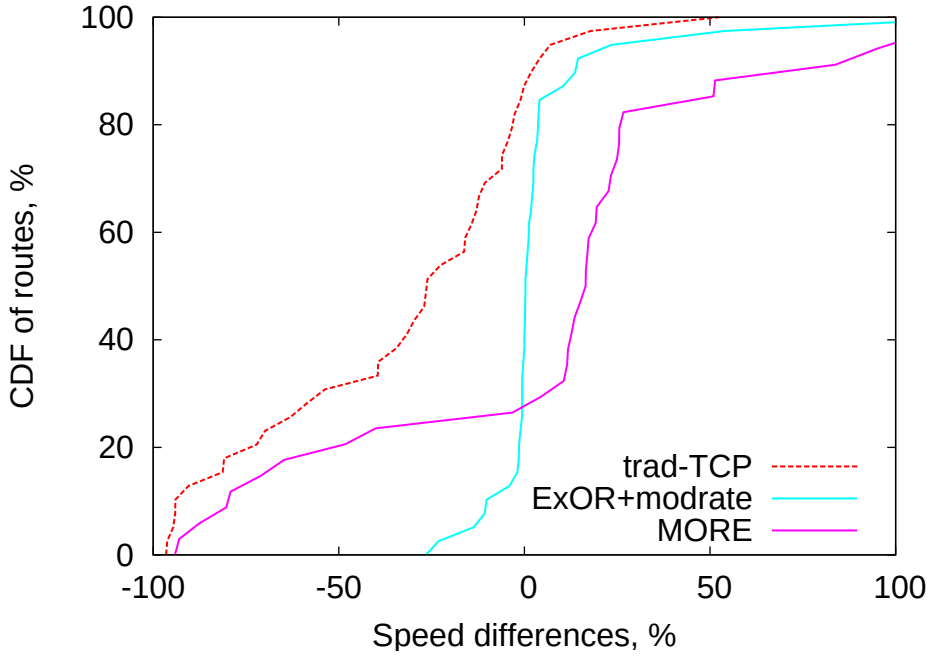



Figure 5.4: Per-path throughput relative to ExOR without modrate for ExOR with modrate, MORE, and traditional routing. ALIX network, full power.

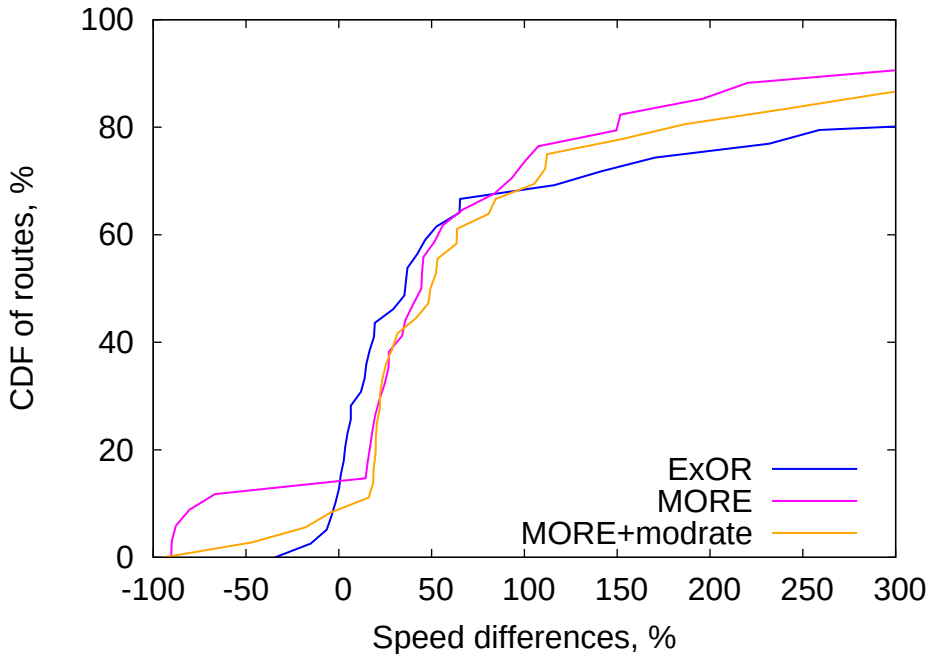



Figure 5.5: Per-path throughput relative to traditional routing for modrate, ExOR, and MORE. ALIX network, full power.

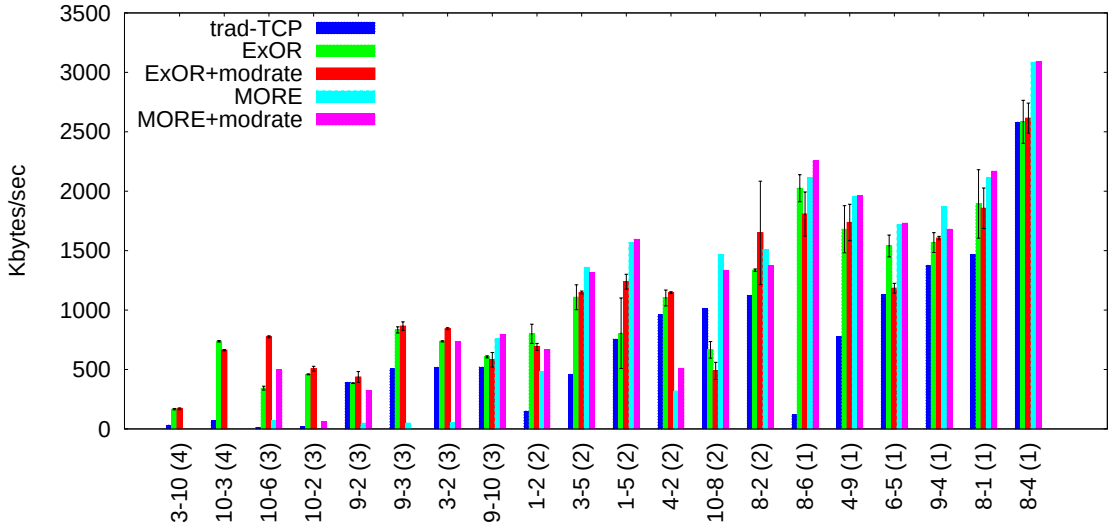


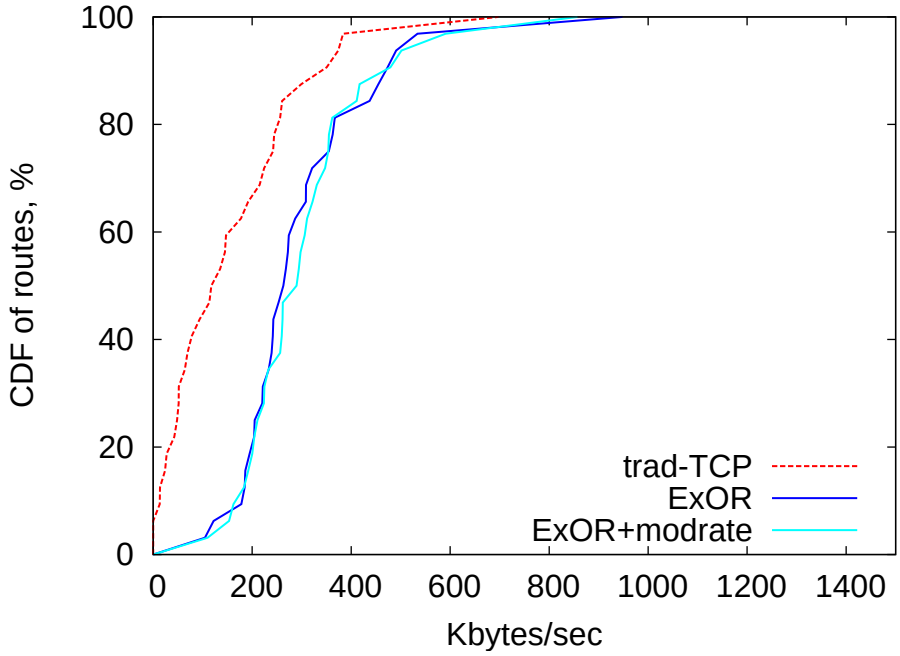

Figure 5.6: Path throughput for 15 representative routes. ALIX network, max power.

but brings significant improvement in around 15% of paths. This is easily explained by observing that modrate degenerates to the link-local scheme in the case of one-hop routes; even for longer routes, modrate select identical rates 62% of the time. It is impossible to tell from the CDFs, however, precisely which routes are seeing improvement. 5.6 presents 15 representative routes sorted according to their length and performance under traditional routing. Error bars report the standard deviation of the 10 constituent batches. We see that modrate provides performance increases in many of the two- and three-hop cases, but—as expected—none of the one-hop paths.

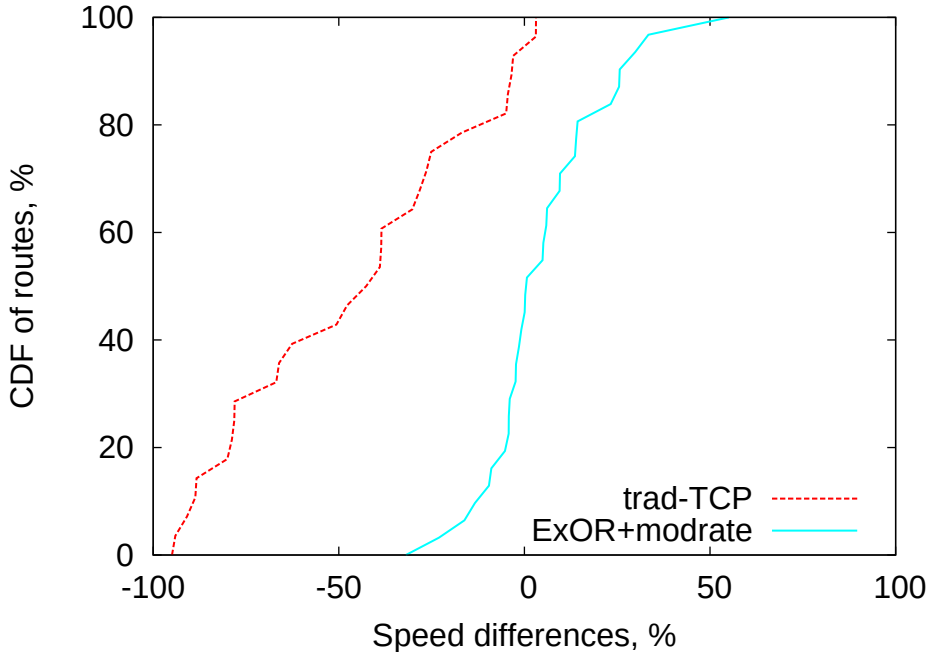
### 5.3.5 Building-wide performance

5.7 shows the performance of various schemes on the top three (2nd through 4th) floors of the Jigsaw testbed. These experiments use 802.11g so they were conducted late at night in an attempt to reduce the interference from the production 802.11g network. The Jigsaw nodes are significantly less powerful than those in the ALIX testbed, and turn out to be CPU-limited when using the MORE protocol, so we do not report results for MORE.

Due to the disparate layouts of the floors, there is significant variation between the connectivity of individual floors. Hence, we also plot the performance one floor at

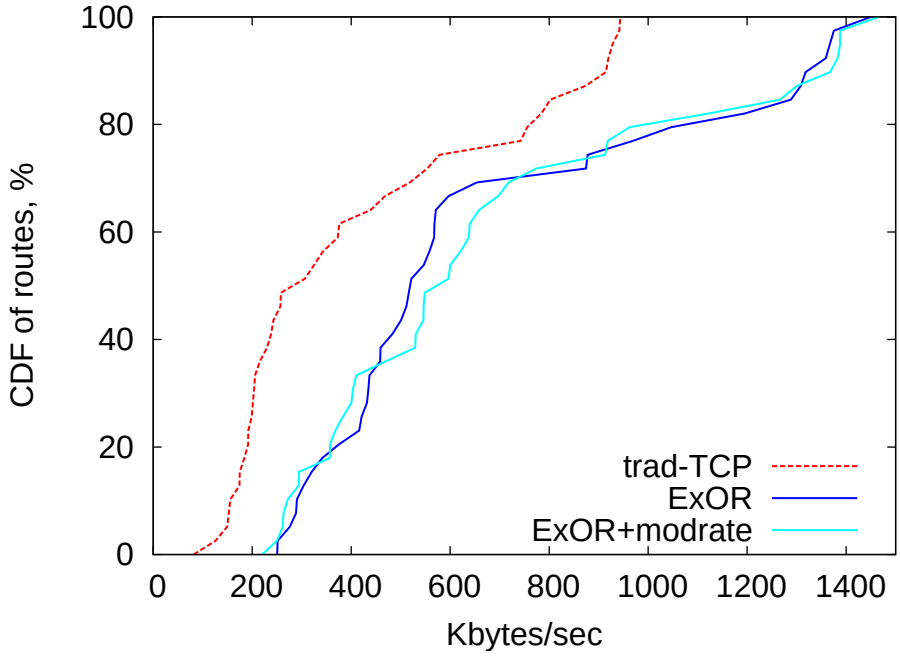



(a) CDF of path throughputs.

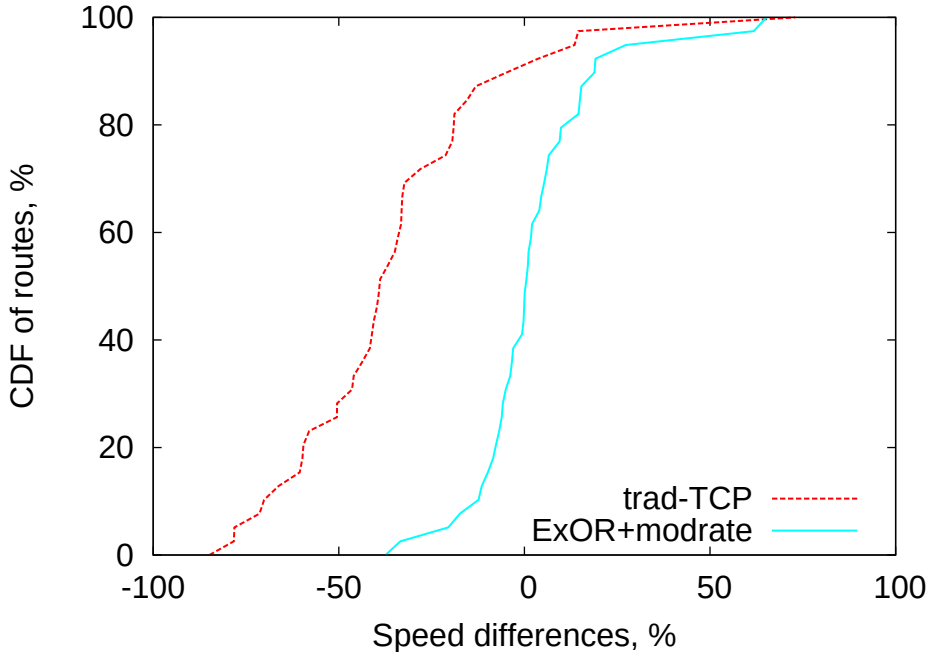



(b) Per-path throughput relative to ExOR for modrate

Figure 5.7: Jigsaw network, full power.

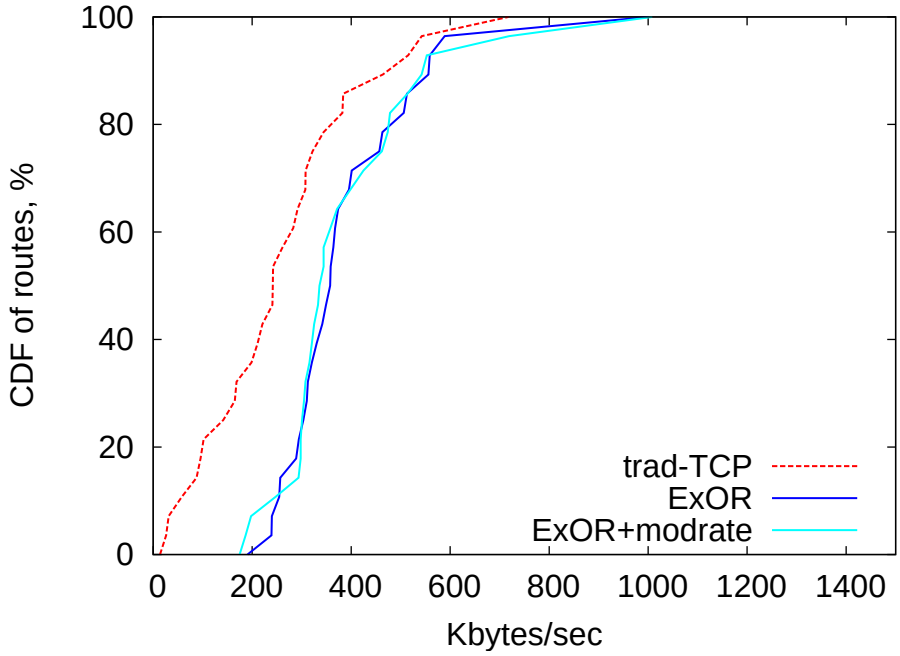



(a) CDF of path throughputs.

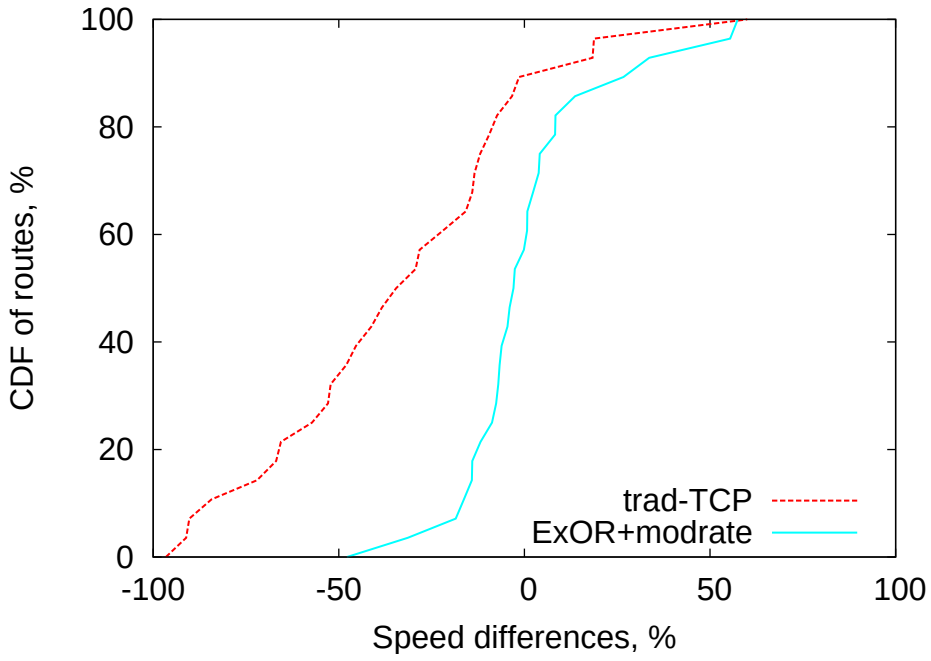



(b) Per-path throughput relative to ExOR for modrate

Figure 5.8: Jigsaw network, 2nd floor only; full power.




(a) CDF of path throughputs.




(b) Per-path throughput relative to ExOR for modrate

Figure 5.9: Jigsaw network, 3rd floor only; full power.

a time. 5.8 and 5.9 show the results for floors two and three, respectively. Each floor has a similar area and density of nodes, so comparing floors reveals differences between similar networks in different physical locations.

While the differences between traditional routing and on-path overhearing varies between each floor, the difference between non-modrate and modrate-enabled version of ExOR stays approximately the same for all testbeds. We explore this phenomena in more details in next chapter.

## 5.4 Conclusion

In this chapter, we have presented modrate, an algorithm for selecting optimal transmission rates in off-path overhearing systems in order to optimize throughput by increasing the prevalence of overhearing. We have also implemented the system on two of our testbeds using the ExOR protocol, and on one testbed using the MORE protocol. Our results show that the ExOR with modrate support outperforms regular ExOR on most routes in the Jigsaw testbed, but provides on average 20% improvement on 15% of the routes in the ALIX testbed. MORE with modrate support also outperforms regular MORE in many cases, but the exact results are unclear because of the high variability of MORE's performance. There is often detectable improvement, but it is still much smaller than the improvement introduced by enabling off-path overhearing, and smaller than the differences in the reception range as a function of modulation rate may suggest. In the next section, we take a closer look at the operation of the ExOR and MORE in order to determine the main factors which affect performance.

Chapter 5, in part, is a preprint of material that has been accepted for publication in the proceedings of the ACM SIGCOMM Conference on Internet Measurement, 2009, Afanasyev, Mikhail; Snoeren, Alex C. The dissertation author was the primary investigator and author of this paper.

# Chapter 6

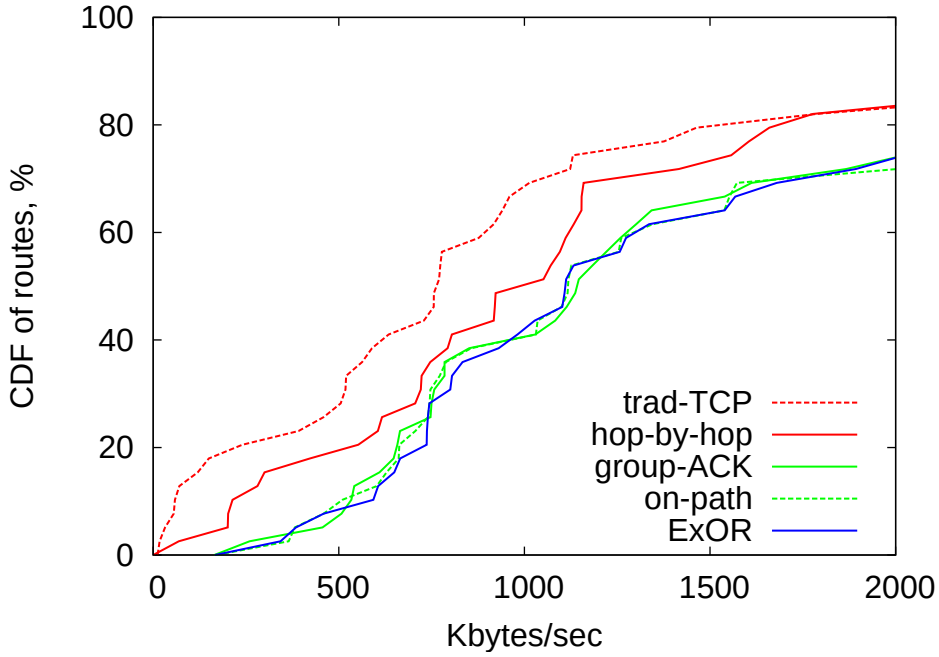
# The importance of overhearing

While modrate functions as expected, we were initially surprised by its modest gains given the dramatic differences in reception ranges shown in 3.7. In particular, modrate is often able to significantly increase overhearing opportunities as shown in 5.2, yet throughput gains are limited. Attempting to "debug" this situation leads to the last major contribution of our dissertation, namely uncovering the reasons behind ExOR and MORE's performance. Our experiments indicate that, for our testbeds at least, overhearing plays a relatively minor role.

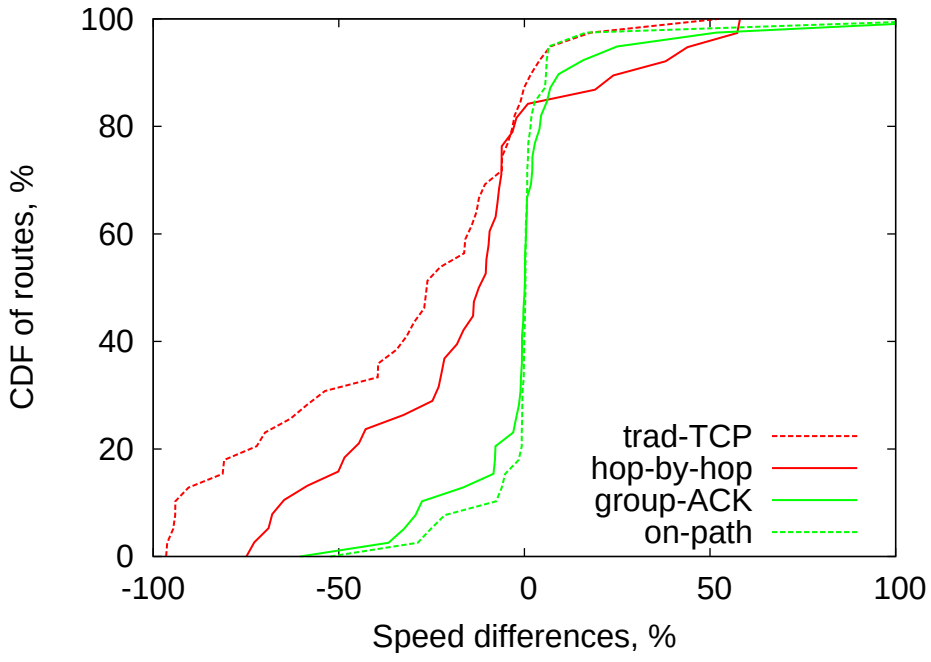
## 6.1 Analysis of ExOR

In this section, we take a close look at the ExOR protocol. In order to attribute performance gains to various aspects of the ExOR implementation, we have expanded the number of cases that we are testing – in addition to the two existing protocols, traditional and off-path overhearing, we have implemented three intermediate protocols, derived from ExOR, each with a subset of features disabled.

We replot the results of our previous experiments on the ALIX network in 6.1. Note that these graphs use the same dataset as 5.3 and 5.4, and that both 'trad-TCP' and 'ExOR' lines are unchanged. We describe the additional protocol versions and their relative performance in increasing levels of sophistication (and performance) below.




(a) CDF of path throughputs.




(b) Per-path throughput relative to ExOR.

Figure 6.1: The performance of various subsets of the ExOR algorithm. ALIX network, maximum power.

### 6.1.1 Traditional routing

Traditional routing is used to represent a no-overhearing case. It uses TCP for flow control and standard 802.11 ACK and retry mechanisms (as described in 2.1.2) are used to ensure the reliable delivery of packets at each hop. We employ our Srcr forwarder, described in more detail in Section 5.2.2.

### 6.1.2 Bulk transport

Perhaps the most fundamental aspect of practical implementations of off-path overhearing protocols such as ExOR and MORE a their batch structure. Rather than transmitting packets as a stream (or windowed stream as in TCP), these protocols uses an explicit batch construct. In ExOR, each node transmits an entire batch at a time before pausing to allow downstream nodes to forward them. We implement this functionality on top of traditional routing with link-level acknowledgments. In this mode, the ExOR's batch map is not used (but for better comparison, it is still included as an overhead). Instead, each station transmits all packets it has once. Packets are sent in 802.11 unicast mode (as opposed to off-path protocols' usual broadcast), so link-level retransmissions may occur on lossy links, up to 10 times in our configuration. We note that this—not our 'trad-TCP' line—is what the ExOR paper [BM05] calls 'Srcr'.

It is frequently observed that TCP's back-off behavior is not ideal in wireless mesh networks. Hence, one might expect that bulk transfer, even operating on exactly the same routes at the same speeds, would perform better. Indeed, the simple bulk-transfer variant, labeled 'hop-by-hop' in the graphs, significantly out-performs 'trad-TCP', on average constituting more than 50% of ExOR's improvement. Interestingly, in almost 20% of cases, it actually out-performs ExOR itself.

### 6.1.3 Group acknowledgments

For a protocol transmitting batches at a time, it is natural to consider getting rid of individual packet acknowledgments in favor of bulk or group acknowledgments. In particular, instead of waiting for a link-level ACK after every frame, a node can send a single, combined transport-layer ACK at the end of transfer. Indeed, this is precisely

what ExOR does with its batch maps. Group acknowledgments increase the latency of retransmissions, but latency is not a figure of merit for ExOR or the other protocols we study.

We have implemented a group acknowledgment scheme by simply disabling overhearing in ExOR. In particular, a node will only accept packets transmitted by the previous hop according to the underlying Srcr route. This algorithm is labeled 'group-ACK' in the graphs. We observe that 'group-ACK' is likely to perform well on low-loss links—because no time is wasted on superfluous link-level ACKs—and asymmetric links with lossy ACK channels. Given the significant improvement over the 'hop-by-hop' line in this configuration, we conjecture one or both of these instances occur frequently. We ascribe the small number of routes where 'hop-by-hop' outperforms 'group-ACK' to experimental variation.

### 6.1.4 On-path overhearing

While ExOR takes advantage of off-path overhearing, on-path overhearing is easier to build into existing protocols (as we showed in Chapter 4). We evaluate the effectiveness of strictly on-path overhearing by restricting the forwarder list to include nodes only on the Srcr path—as opposed to any node that is predicted to overhear at least 10% of the transmissions. Note that unlike RTS-id, which operates with traditional transfer protocols, we leverage a fully batched bulk transfer protocol. Since the system already includes group acknowledgments, this method of implementing on-path overhearing does not require additional per-packet transfers, and thus does not add significant overhead like RTS-id does.

Forwarding with this restricted form of overhearing is labeled 'on-path' in the graphs. In our implementation, there can be no overhead with respect to group acknowledgments (any deviations are once again attributable to experimental noise). In this configuration, however, there is also no significant benefit. Theoretically, however, on-path overhearing can add value when there is no single high-quality link for a particular hop in a route, but the combination of reception rates at the the next hop and down-line forwarders combine to provide efficient performance.

### 6.1.5 Off-path overhearing

The final addition to arrive at ExOR is to enable off-path overhearing; namely, to include the full set of potential forwarders in the forwarder list. In this case, there are multiple possible paths, and packets choose the best path dynamically. We expect that extra nodes will improve the performance when when routing information is unreliable or out of date, as the extra nodes might suddenly become valuable.

We observe, however, that ExOR is not always the most efficient. In particular, these extra nodes can actually add overhead due to scheduling: it takes time to communicate the longer forwarder list and start and stop a round. Also, if the additional nodes have poor reception, they may not receive batch maps, and keep transmitting the same data over and over again. We find that ExOR works best when routes are generally poor, but there are many of them.

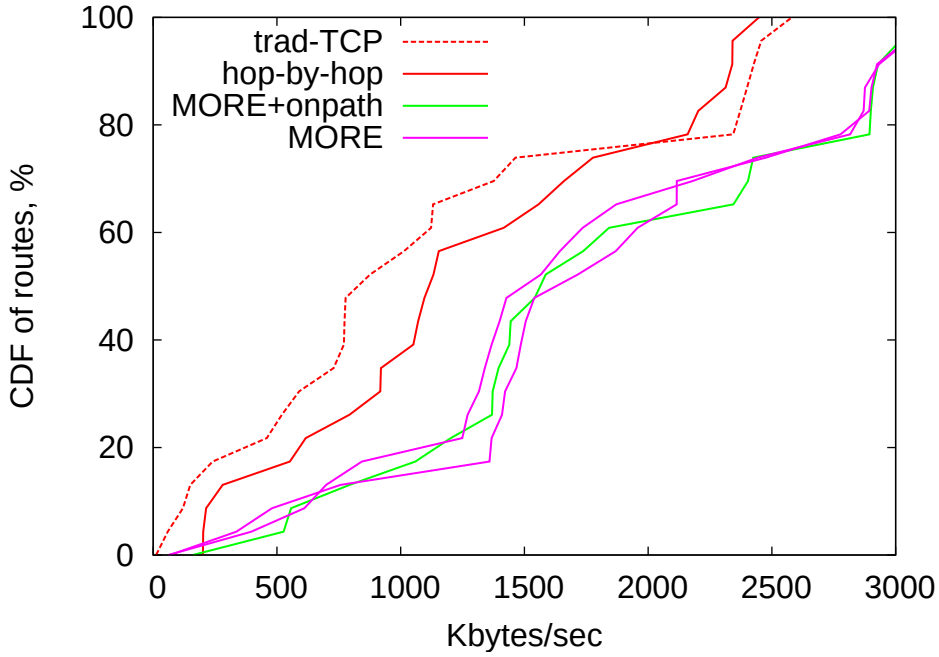
## 6.2 MORE analysis

In this section, we turn our attention to the MORE protocol. Again, in order to attribute the performance gains, we have disabled some functionality of the MORE algorithm. In practice, we find that MORE's performance varies dramatically over time. In order to decrease the impact of this variability on our evaluation, we follow the following procedure:

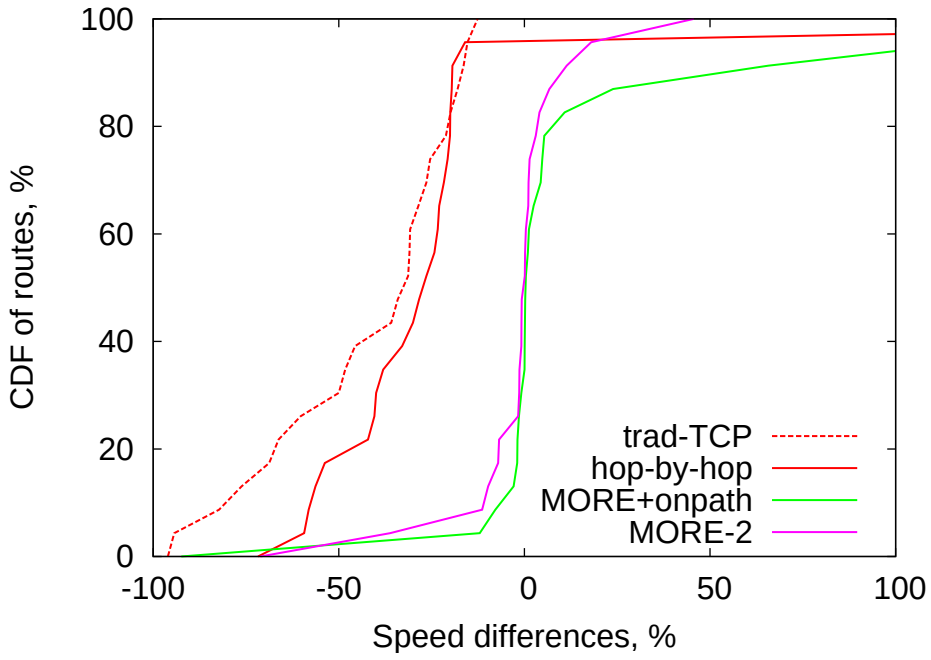
1. In order to estimate the variability of the algorithm, we run each experiment twice over an approximately 3-minute interval. The resulting graphs have two lines with the same color and label – those lines correspond to the two consecutive runs of the algorithm, and provide an appreciation of how stable the results are.

2. We have removed all data for paths where one of the MORE runs shows worse performance than trad-TCP. About 35% of the routes were removed this way.

6.2(a) shows individual CDFs for each algorithm, and 6.2(b) shows the per-route performance relative to first execution of MORE.




(a) CDF of path throughputs.




(b) Per-path throughput relative to MORE

Figure 6.2: The detailed performance of the MORE algorithm. ALIX network, maximum power.

### 6.2.1 Traditional protocol

The MORE publication does not explicitly describe the details of the implementation of their Srcr protocol. The code that the authors make publicly available on their website includes a simplified version of Srcr, which is internally called 'SPP'. This is a bulk transport protocol that transfers packets one at a time in unicast mode, using 802.11 link-level acknowledgments in order to ensure the reliability of each link.

We attempted to use SPP in our comparisons, but found that the protocol is unreliable on our testbeds, frequently failing to transfer any significant amount data. A closer study of the supplied code reveals that SPP does not use any form of flow control at all, instead transmitting packets from the source at the maximum rate that channel utilization allows. This configuration causes overflow of packet queues at the intermediate nodes and significant packet loss. Because of these problems, we instead use the same 'hop-by-hop' and 'trad-TCP' protocols that we considered in ExOR case, shown in 6.1. The plots are based on the same data, but they look different, since only those paths that do not exhibit hidden terminals are displayed.

### 6.2.2 Group acknowledgments

The MORE protocol uses network coding and does not implement any acknowledgments in the traditional sense. While using coding in order to avoid retransmissions in the link-layer is a known technique, it is normally used only on one independent link.

It is possible to disable overhearing in MORE, but time constraints force us to leave this to the future work.

### 6.2.3 On-path overhearing

We implement on-path-only overhearing in MORE by limiting the set of nodes which can forward packets to the nodes that are on the Srcr path. The restricted version of the protocol appears on the graph as 'MORE-onpath', and shows better performance than regular 'MORE'. This is not surprising, given that one of the biggest problems in MORE is caused by large number of stations transmitting simulatenously. On-path overhearing uses fewer stations, and thus produces better results on our testbed.

### 6.2.4 Summary

The main difference between the MORE and ExOR algorithms is the lack of any kind of acknowledgments or scheduling mechanisms. While those properties help MORE provide a significantly higher median throughput, they also cause severe hidden terminal problems on some paths. Thus, it is hard to evaluate the impact of overhearing on MORE. The difference between on-path overhearing and off-path overhearing is small, and comparable to the same difference for the ExOR mechanism. Thus, we belive that most of the performance advantages of MORE relative to ExOR come from very aggressive flow control, and the ability of all nodes to transmit simultaneously.

## 6.3 Power variations

Given the small contribution that overhearing—either on-path or off-path—makes to ExOR's and MORE's performance in the testbed configuration studied so far, in retrospect it is not at all surprising that modrate would have relatively modest gains. In particular, intuitively, modrate provides larger gains when a protocol runs all links at high speed (so there is room for modrate to decrease them), but reception rates are similar across a range of intermediate hops (so the best path is just one of a number of alternatives).

In order to evaluate the potential for modrate to improve performance when these conditions arise, we attempt to modify the average transmission rates selected by the algorithm by changing the connectivity of the network. Rather than modify the topology—which would make it hard to compare results across runs—we adjust the network-wide power level. As observed in 3.6, different power levels have dramatically different reception ranges in the ALIX testbed. Due to similarity of our findings in the previous sections, we believe that the results would be similar for both ExOR and MORE. However, due to the high variance of MORE's performance, we will consider only ExOR going forward. We re-run the previous ExOR experiments at three additional power levels—40, 50, and 60 (full power)—in addition to the level 30 results previously reported. As previously noted, modrate frequently chooses the same rate as ExOR. Hence, we restrict our attention to those routes where modrate selects different link

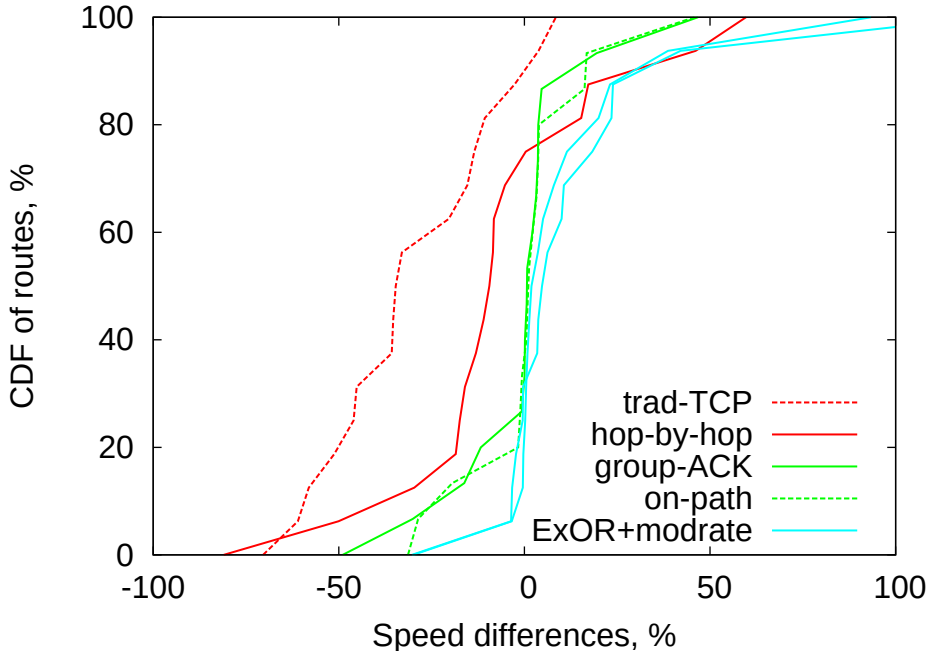



Figure 6.3: Protocol break-down for ExOR on the ALIX network at power levels 30.

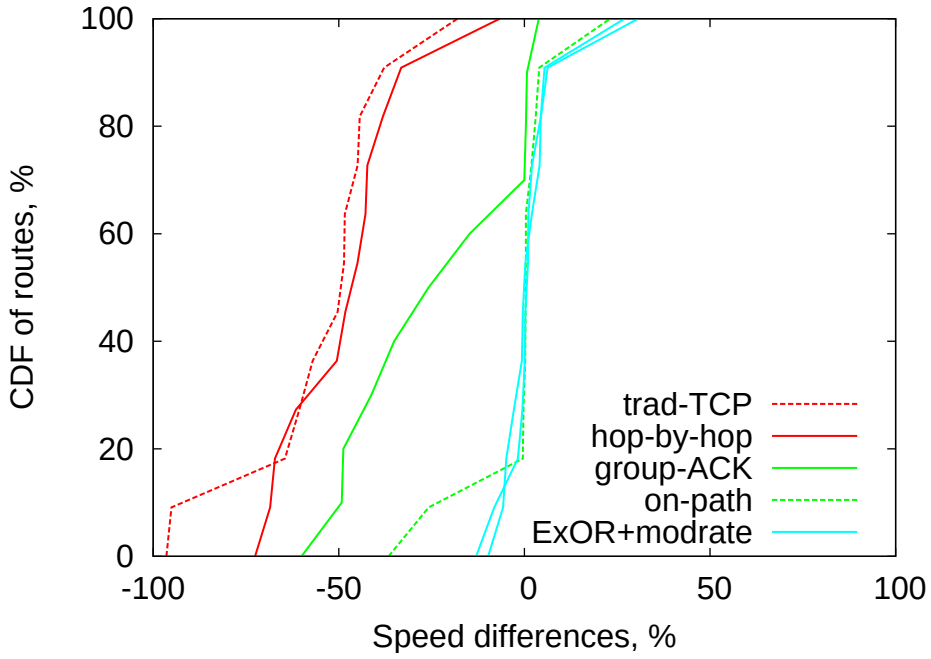



Figure 6.4: Protocol break-down for ExOR on the ALIX network at power levels 40.

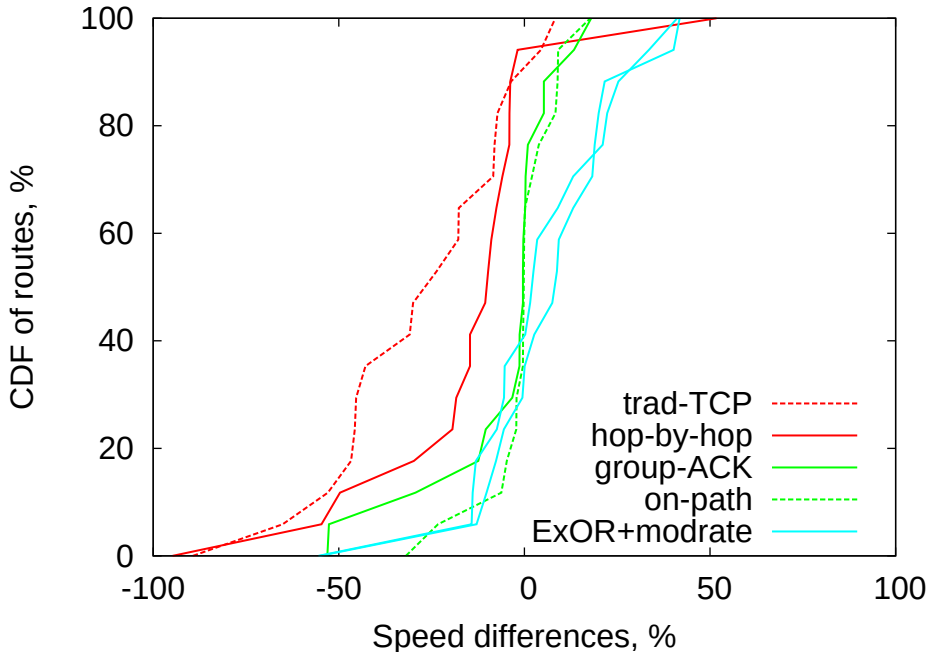



Figure 6.5: Protocol break-down for ExOR on the ALIX network at power level 50.

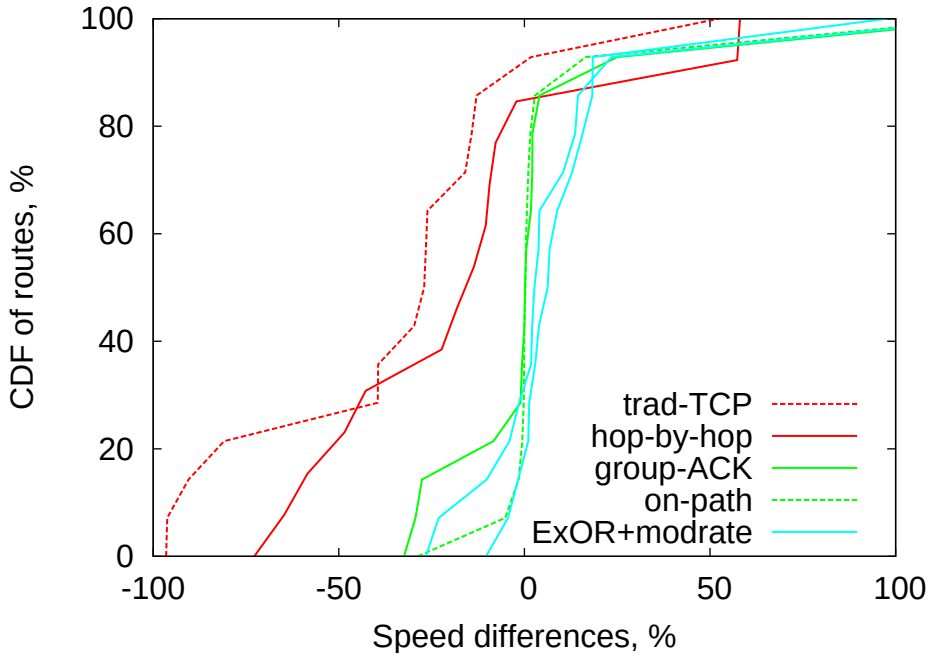



Figure 6.6: Protocol break-down for ExOR on the ALIX network at power level 60 (maximum power).

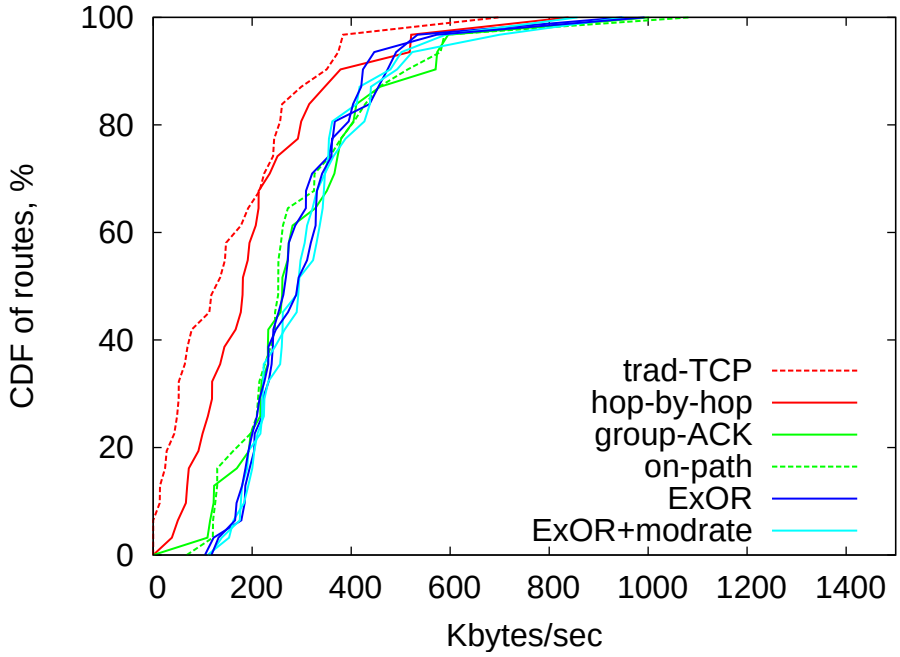
speeds—approximately 7 to 10% of all possible routes in the ALIX testbed, depending on the power level employed. We present the results for all four speeds in 6.3 thru 6.6. In an effort to account for experimental noise, we run both ExOR and the ExOR+modrate algorithm twice, so the graphs have two lines with identical labels corresponding to the two consecutive runs. In addition, the plots showing relative performance have a line labeled 'ExOR-2', which directly indicates the experimental noise between the two ExOR runs.

Not only does the contribution of modrate change with power level (peaking at power level 50 when connectivity is high, but still more variable than at full power), but the various components of ExOR do as well. Notably, the contribution of group acknowledgments decreases at power level 40, presumably because ExOR has selected unreliable links. Bulk acknowledgments are similarly of limited utility in the presence of lossy links. Several overall observations can be made as well: none of the techniques provide much improvement at low or full power, as poorly connected network generally has only one path made of of low quality links, while, conversely, a well-connected network with short paths does just fine with traditional routing. Networks with a range of connectivity provide the most fertile ground for all of the enhancements, but the relative importance of each can vary.

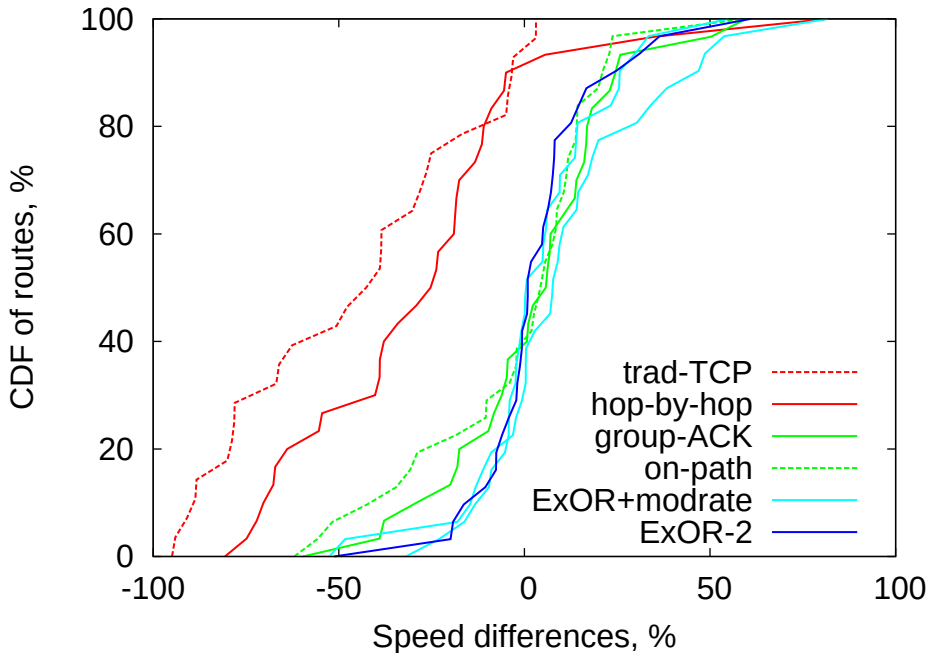
## 6.4 Jigsaw testbed

In order to verify that the results that we obtain on the ALIX testbed are not anomalous, we repeat the experiment on the Jigsaw testbed. Since MORE is too CPU-intensive for Jigsaw nodes, we only repeat ExOR experiments. The results are shown in 6.7 thru 6.9 as overlays to 5.7 thru 5.9 — the 'trad-TCP', 'ExOR' and 'ExOR+modrate' lines are unchanged, but new lines in the middle provide additional information on the reasons for performance.

5.7 shows the performance of various schemes on the top three (2nd through 4th) floors of the Jigsaw testbed. Because it was originally deployed as a passive sniffer network, the Jigsaw nodes are quite dense. Hence, the overall results most closely resemble those from the maximum-power ALIX experiment: there is little difference

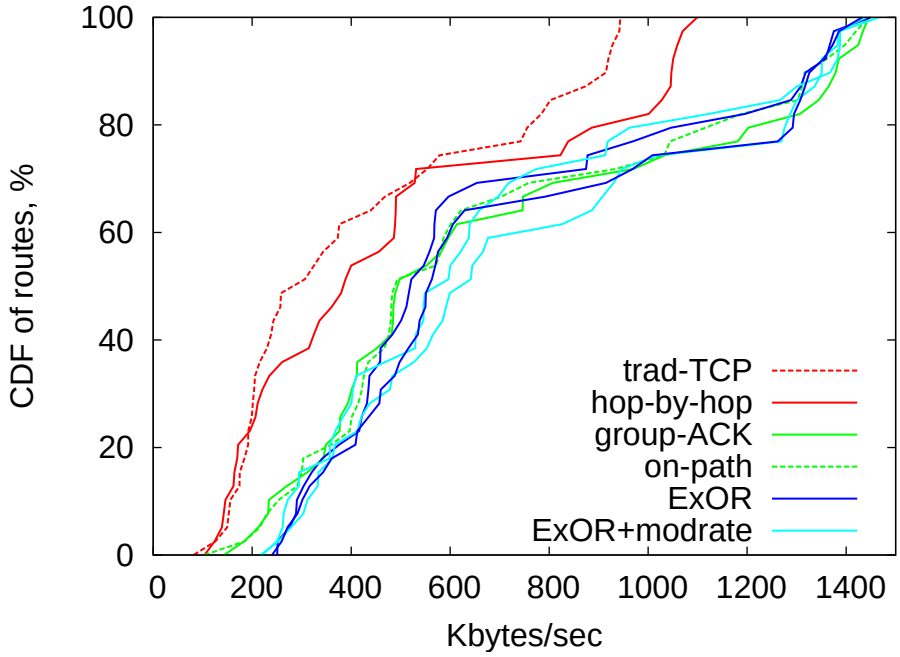



(a) CDF of path throughputs.

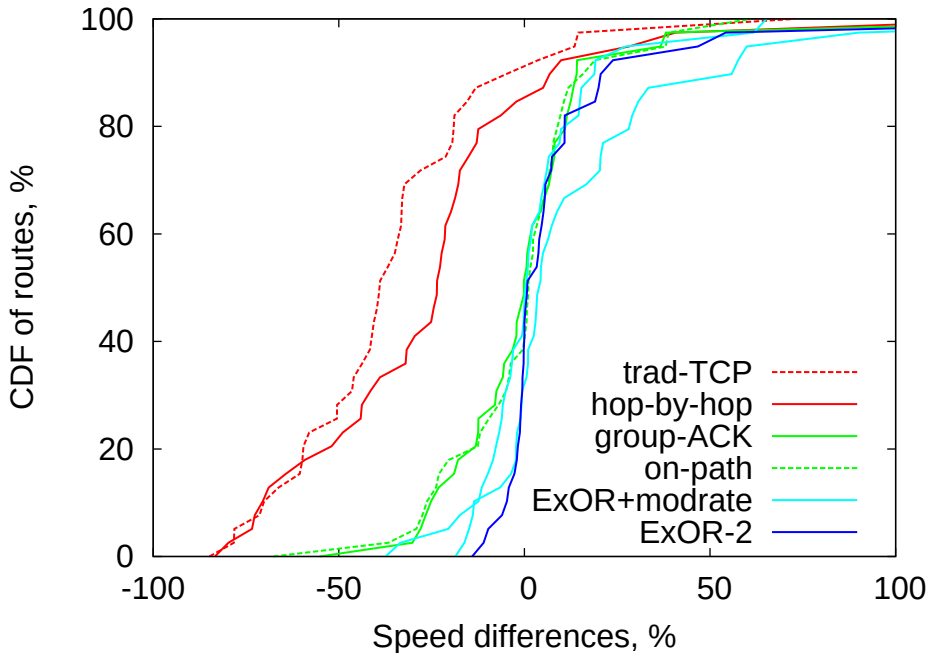



(b) Per-path throughput relative to ExOR for modrate

Figure 6.7: The performance of various subsets of the ExOR algorithm. Jigsaw network, full power.

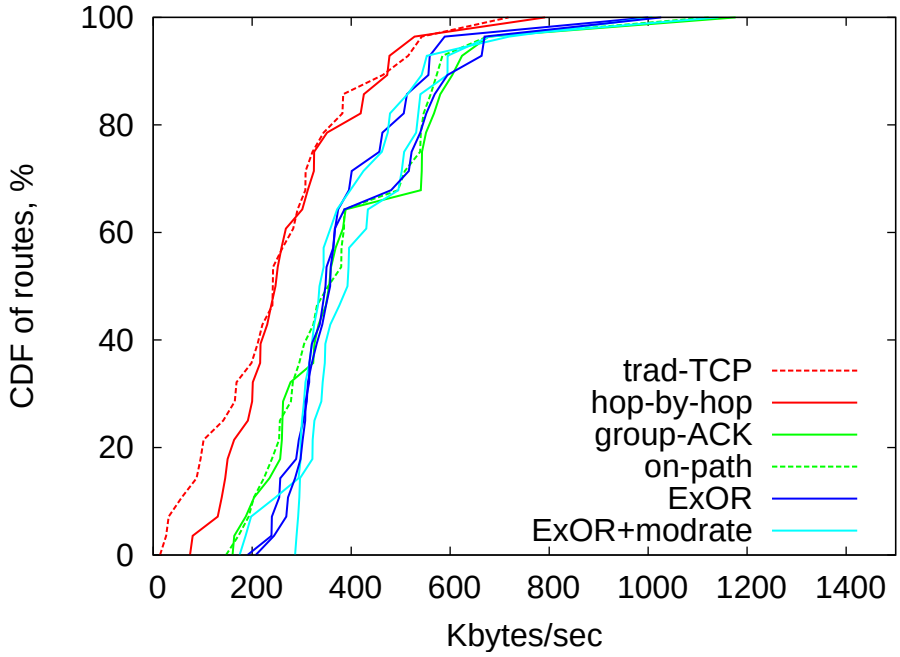



(a) CDF of path throughputs.

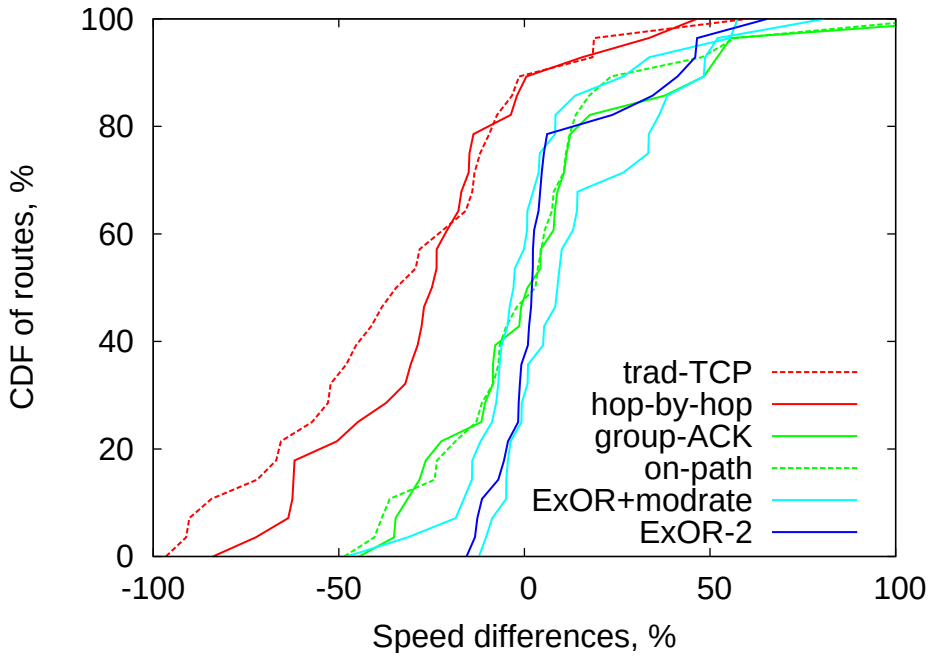



(b) Per-path throughput relative to ExOR for modrate

Figure 6.8: The performance of various subsets of the ExOR algorithm. Jigsaw network, 2nd floor only; full power.




(a) CDF of path throughputs.




(b) Per-path throughput relative to ExOR for modrate

Figure 6.9: The performance of various subsets of the ExOR algorithm. Jigsaw network, 3rd floor only; full power.

between ExOR and any of its variants—including modrate. Due to the disparate layouts of the floors, however, there is significant variation between the connectivity of individual floors. Hence, we also plot the performance one floor at a time.

5.8 and 5.9 show the results for floors two and three, respectively. Each floor has a similar area and density of nodes, so comparing floors reveals differences between similar networks in different physical locations. Floor two, in particular, shows substantial improvement from modrate and bulk transport—similar to power 50 in the ALIX testbed. Yet floor three has some routes with dramatic improvements under modrate, some exceeding 100%, over and above the already dramatic improvements due to group acknowledgments.

## 6.5 Conclusion

In this chapter, we have researched the specific causes of performance improvement in both the ExOR and MORE protocols. We have evaluated the protocols with various subsets of their features disabled, in order to determine how much each feature contributes to overall performance. Our results definitely show that for the ExOR protocol, the major contributors to its performance on our testbeds are the elimination of TCP flow control and removal of per-packet acknowledgments. Both on-path and off-path overhearing contribute much less improvement, relatively speaking.

Our analysis of the MORE protocol was more complicated due to MORE's susceptibility to the hidden terminal problem. While we are unable to definitely separate the contributions of flow control, per-packet acknowledgments and on-path overhearing, we see only modest improvements from off-path overhearing and modrate, which leads us to suspect that the majority of MORE's speed advantages do not come from overhearing either.

Chapters 6, in part, is a preprint of material that has been accepted for publication in the proceedings of the ACM SIGCOMM Conference on Internet Measurement, 2009, Afanasyev, Mikhail; Snoeren, Alex C. The dissertation author was the primary investigator and author of this paper.

# Chapter 7

# Conclusion

Mesh networks provide a useful and practical alternative to traditional infrastructure wireless networks, but they have an intrinsic scaling limit – the raw channel capacity. Wireless networks are broadcast, so a packet sent to one station might interfere with or be received by other nearby stations. Mesh networks forward packets multiple times, increasing airtime utilization and decreasing path throughput and channel capacity available to the clients. In this dissertation, we have explored ways to optimize forwarding in order to increase path throughput.

The main technique that we explored in this dissertation is a phenomenon called 'overhearing'. Traditional mesh networks use only 'good' links with low data losses in order to forward packets; overhearing allows utilization of the links with high losses in order to reduce the number of transmissions where possible. Our thesis is that it is possible to improve the performance of wireless mesh networks by taking full advantage of overhearing.

## 7.1 Summary

We considered two types of overhearing in this dissertation. Previous work introduced opportunistic routing systems that support off-path overhearing. However, such systems require a complete re-design of transport layer protocols, and thus are not compatible with existing protocols like TCP. We have shown that it is possible to achieve some of our goals without a complete re-design of transport protocols. In

order to do this, we use on-path overhearing – an optimization applied to traditional mesh routing systems. On-path overhearing can prevent un-needed retransmissions when a packet is received by a node that is closer to the destination than the intended forwarder. We introduced a per-hop link-layer modification that we call 'RTS-id', that can take advantage of overheard packets in a protocol and topology-independent way. This system is backwards-compatible with existing 802.11 hardware, and thus can be deployed incrementally. Additionally, it works well with existing transport protocols and does not require application rewrites.

In order to verify our RTS-id system, we have implemented it in the hardware using CalRadio, a software radio platform developed at CalIT2. We first implemented a basic 802.11 MAC in order to make CalRadio operational, and then extended the MAC with RTS-id support. Using a small testbed consisting of three CalRadio nodes, we have verified that RTS-id works as designed, and that it is backward-compatible with existing 802.11 radios.

The number of CalRadio devices available to us is limited, so in order to see the results of RTS-id on the large-scale systems, we implemented a statistical, trace-based simulator. We obtained traces from an existing wireless mesh network (Roofnet), and used our simulator to estimate the expected performance gains from large-scale installation of RTS-id system.

The results show that overhearing can save a significant number of transmissions, up to 20% for a quarter of all routes. However, due to the overheads, the transmissions themselves are longer, and as a results, the airtime usage savings are modest: up to 10% reduction in airtime for only 5% of possible paths. One observation that we have is that RTS-id brings the most improvement to systems that do not use advanced routing protocols, and that even a simple routing protocols (such as shortest path), when combined with RTS-id, outperforms the state-of-the-art traditional routing protocols like Srcr. Thus, it appears that, in some cases at least, an optimized link-layer protocol can compensate for less-than-optimal routing protocol, and vice versa.

Motivated by the relatively modest improvements of RTS-id, we decided to look at the systems that support off-path overhearing, opportunistic routing systems like ExOR and MORE. These systems use an alternative routing structure with each packet taking

an individual, most optimal route. We observed, however, that the important question of the rate selection in such systems has not been studied. As our measurements shows, a lower transmission rate increases the range, and thus provides more opportunities for overhearing. At the same time, a higher transmission rate decreases the time it takes to send a packet. We designed a system called 'modrate', which allows selection of optimal transmission rates in opportunistic routing systems. Modrate globally optimizes the rates in order to increase the amount of overhearing in the systems, and minimize the total transmission time.

In order to verify and demonstrate our approach, we set up two new wireless testbeds: an 802.11b/g testbed which is based on the Jigsaw infrastructure and spans four floors in our building, and a smaller, 802.11a ten-node testbed made from ALIX nodes, which spans one wing of the building.

We have implemented ExOR and MORE algorithms with modrate support, and ran a large number of experiments on those testbeds. We discovered that modrate increases probability of overhearing, and improves the end-to-end goodput by 10% to 20% for over 15% of the routes for ExOR protocol. We have also confirmed that modrate increases the performance of the MORE protocol. While these results are better than RTS-id, they are still not as good as improvements seen from adding off-path overhearing – a route might have 200%–400% throughput gain from adding ExOR, and an additional 10%–20% gain from adding modrate. Since we believed that most of the performance gains came from off-path overhearing, we were surprised by those improvements.

Finally, in order to explain our modrate results, we studied the specific causes of performance improvements in the ExOR and MORE protocols. We evaluated the protocols with a subset of features disabled in order to isolate the contribution of each feature to the overall performance. We discovered that most of the performance improvement in ExOR comes not from the overhearing, but from more prosaic things like elimination of per-packet acknowledgments and using a specialized flow-control algorithms instead of TCP. We have also evaluated MORE, and came to the conclusion that most of the performance improvement of MORE comes from the same sources, with an additional improvement coming from the lack of an explicit scheduler. The latter reason, unfortunately, significantly increases the effects of hidden terminals, making

MORE unusable in over 15% of our routes.

## 7.2 Open questions

There are multiple directions of future work for this thesis. First, it would be interesting to determine how typical our situation is. We have run experiments only on indoor testbeds – while our experiences with the Google urban WiFi network lead us to believe results would be similar, the experiments on outdoor testbed would be extremely valuable.

Next, given the relative unimportance of overhearing in existing bulk-transfer protocols, it would be interesting to try to improve the non-overhearing aspects of the existing algorithms. For example, MORE has somewhat trivial scheduler and flow control system, and ExOR does not use spectrum in most efficient way. Since it seems that those factors have an extremely large effect on the algorithm's performance, it seems likely that some optimizations could significantly improve the algorithms' performance.

Finally, it is worth considering whether overhearing is useful at many circumstances. Since most of the performance gains come from bulk acknowledgments and per-flow algorithms, it is quite possible that a simpler algorithm can achieve much of the performance advantage that ExOR and MORE have, but without significant added complexity.

# Bibliography


[ABB+04] Daniel Aguayo, John Bicket, Sanjit Biswas, Glenn Judd, and Robert Morris. Link-level measurements from an 802.11b mesh network. In *Proc. ACM SIGCOMM*, Portland, OR, August 2004.

[AL04] Aner Armon and Hanoch Levy. Cache satellite distribution systems: Modeling, analysis, and efficient operation. *IEEE Journal on Selected Areas in Communication*, 22(2), February 2004.

[BABM05] John Bicket, Daniel Aguayo, Sanjit Biswas, and Robert Morris. Architecture and evaluation of an unplanned 802.11b mesh network. In *Proc. ACM Mobicom*, Cologne, Germany, September 2005.

[Bic05] John C. Bicket. Bit-rate selection in wireless networks. Master's thesis, Massachusetts Institute of Technology, 2005.

[BM03] Sanjit Biswas and Robert Morris. Opportunistic routing in multi-hop wireless networks. In *Proc. 2nd ACM Workshop on Hot Topics in Networks (Hotnets-II)*, Cambridge, MA, November 2003.

[BM05] Sanjit Biswas and Robert Morris. ExOR: opportunistic multi-hop routing for wireless networks. In *Proc. ACM SIGCOMM*, Philadelphia, PA, August 2005.

[CABM03] Douglas S. J. De Couto, Daniel Aguayo, John Bicket, and Robert Morris. A high-throughput path metric for multi-hop wireless routing. In *Proc. ACM Mobicom*, San Diego, CA, September 2003.

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

[CAV+07] Yu-Chung Cheng, Mikhail Afanasyev, Patrick Verkaik, Peter Benko, Jennifer Chiang, Alex C. Snoeren, Geoffrey M. Voelker, and Stefan Savage. Automating cross-layer diagnosis of enterprise wireless networks. In *Proc. ACM SIGCOMM*, Kyoto, Japan, August 2007.

[CBB+06] Yu-Chung Cheng, John Bellardo, Peter Benko, Alex C. Snoeren, Geoffrey M. Voelker, and Stefan Savage. Jigsaw: Solving the puzzle of enterprise 802.11 analysis. In *Proc. ACM SIGCOMM*, Pisa, Italy, August 2006.

[Cha07] Szymon Chachulski. Trading structure for randomness in wireless opportunistic routing. Master's thesis, Massachusetts Institute of Technology, 2007.

[CJKK07] Szymon Chachulski, Michael Jennings, Sachin Katti, and Dina Katabi. Trading structure for randomness in wireless opportunistic routing. In *Proc. ACM SIGCOMM*, Kyoto, Japan, August 2007.

[DPZ04] Richard Draves, Jitendra Padhye, and Brian Zill. Comparison of routing metrics for static multi-hop wireless networks. In *Proc. ACM SIGCOMM*, Portland, OR, August 2004.

[Emb] Embedded Linux/Microcontroller Project. uCLinux. `http://www.uclinux.org/`.

[HVB01] G. Holland, Nitin Vaidya, and P. Bahl. A rate-adaptive MAC protocol for multi-hop wireless networks. In *Proceedings of ACM Mobicom*, July 2001.

[KM97] A. Kamerman and L. Monteban. WaveLAN II: A high-performance wireless LAN for the unlicensed band. *Bell Labs Technical Journal*, pages 118–133, 1997.

[KRH+06] Sachin Katti, Hariharan Rahul, Wenjun Hu, Dina Katabi, Muriel Mèdard, and Jon Crowcroft. XORs in the air: practical wireless network coding. In *Proc. ACM SIGCOMM*, pages 243–254, Pisa, Italy, August 2006.

[LBD+01] Jinyang Li, Charles Blake, Douglas S. J. De Couto, Hu Imm Lee, and Robert Morris. Capacity of ad hoc wireless networks. In *Proc. ACM Mobicom*, pages 61–69, Rome, Italy, July 2001.

[ns] The Network Simulator NS-2. `http://www.isi.edu/nsnam/ns/`.

[PB94] C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. *ACM SIGCOMM Computer Communication Review*, October 1994.

[RC04] Bhaskaran Raman and Kameswari 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, November 2004.

[RMRW06] Charlie Reis, Ratul Mahajan, Maya Rodrig, and David Wetherall. Measurement-based models of delivery and intereference. In *Proceedings of ACM SIGCOMM*, August 2006.

[SKSK05] B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knightly. OAR: An opportunistic auto-rate media access protocol for ad hoc networks. *Wireless Networks*, 11(1–2):39–53, January 2005.

[SPS+02] Alex C. Snoeren, Craig Partridge, Luis A. Sanchez, Christine E. Jones, Fabrice Tchakountio, Beverly Schwartz, Stephen T. Kent, and W. Timothy Strayer. Single-packet IP traceback. *IEEE/ACM Transactions on Networking*, 10(6), December 2002.

[SR07] Sayandeep Sen and Bhaskaran Raman. Long Distance Wireless Mesh Network Planning: Problem Formulation and Solution. In *Proceedings of World Wide Web Conference*, May 2007.

[WYLB06] Starsky H. Y. Wong, Hao Yang, Songwu Lu, and Vaduvur Bharghavan. Robust rate adaptation for 802.11 wireless networks. In *Proceedings of ACM Mobicom*, pages 146–157, September 2006.