Efficient Transmissions & Recovery in Wireless LANs

by

Anand Padmanabha Iyer,

Thesis

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Arts

The University of Texas at Austin

May 2008

Efficient Transmissions & Recovery in Wireless LANs

Approved by
Supervising Committee:

_____

_____

Dedicated to Appa, Amma & Arun

Acknowledgments

This thesis is the result of my joint work with my adviser Dr. Lili Qiu and my colleagues Gaurav Deshpande, Eric Rozner, Yogita Mehta and Mansoor Jafry. I could not express enough gratitude to Dr. Lili who has always been a source of encouragement. I am grateful to her for providing me the opportunity to work in the Laboratory for Advanced Systems Research (LASR) group, and helping me realize that research is what I want to pursue in my life.

I am greatly indebted to my colleagues Gaurav, Eric, Yogita and Mansoor. Working with them made research fun. We have had many fruitful discussions which helped us improve many parts of this work.

I would also like to thank Dr. Yin Zhang for taking the time to review my thesis.

Finally, I thank my parents and my brother whose unfaltering support made this possible.

Anand Padmanabha Iyer

The University of Texas at Austin

May 2008

Efficient Transmissions & Recovery in Wireless LANs

Anand Padmanabha Iyer, M.A.
The University of Texas at Austin, 2008

Supervisor: Lili Qiu

Wireless LANs (WLANs) have seen exponential growth in the recent past with ever-increasing deployments at university campuses, office buildings, airports, hotels, and malls. With such phenomenal growth, it is increasingly important to deliver content efficiently and reliably over wireless links. However, limited wireless spectrum, inherent lossy wireless medium, and imperfect packet scheduling reduce the efficiency and reliability of wireless networks. In this work, we take a holistic approach to optimizing wireless performance and resilience. Specifically, we propose *Fast Resilient Jumbo Frames (FRJ)* as a technique for achieving resiliency, and *Efficient Retransmission Scheme (ER)* as a technique for smarter and efficient retransmissions in WLANs. Our proposal is based on the observation that retransmissions and resiliency are two key issues tightly interleaved with wireless performance.

The idea of FRJ is nurtured from the fact that much of the existing work on resiliency focuses on individual design space. Consequently, FRJ exploits the synergy between three important design approaches: (i) frame size selection, (ii)

partial packet recovery, and (iii) rate adaptation. FRJ uses jumbo frames to boost network throughput under good channel conditions. Additionally, it uses partial packet recovery to efficiently recover packet losses under bad channel conditions. It further jointly optimizes rate adaptation and partial packet recovery to achieve high transmission rates and protect against wireless losses. Our contributions in this work are two-fold: (i) we present a mechanism to support jumbo frames by making them resilient to wireless losses, and (ii) we propose a rate adaptation scheme which combines partial recovery and rate selection. Using real implementation and testbed experiments, we demonstrate that FRJ achieves efficient and resilient performance in wireless LANs. Our results indicate that FRJ consistently out-performs regular sized frames under different channel conditions and multiple flow scenarios. In the more typical WLAN deployment with multiple flows, FRJ achieves up to 69% improvement over existing rate adaptation schemes.

While FRJ tries to achieve resiliency, ER leverages intelligent network coding techniques to achieve efficient retransmissions. Instead of retransmitting the lost packets in their original forms, ER codes packets lost at different destinations and uses a single retransmission to potentially recover multiple packet losses. We develop a simple and practical protocol to realize the idea and implement it in both simulation and testbed, and our results demonstrate the effectiveness of this approach. Since ER techniques can be applied to a wide variety of scenarios, higher gains can be achieved by the application of ER to FRJ.

Contents

Acknowledgments **v**

Abstract **vi**

List of Tables **xi**

List of Figures **xii**

List of Algorithms **xiv**

Chapter 1 Introduction **1**

1.1 Motivation 1

1.2 FRJ 2

1.3 ER 3

1.4 Contributions 5

1.5 Outline 6

Chapter 2 Related Work **7**

2.1 Packet Recovery 8

2.1.1 Retransmission 8

2.1.2 FEC 8

2.1.3 Partial Packet Recovery 9

2.2 Channel Reservation 10
2.3 Rate Adaptation 11
2.4 Jumbo Frames 11
2.5 Network Coding 12

Chapter 3 FRJ: Fast Resilient Jumbo Frames in WLANs 13

3.1 Approach 13
3.1.1 Resilient Jumbo Frames 14
3.1.2 Partial Recovery Aware Rate Adaptation 16
3.2 Implementation 21
3.2.1 Overview 21
3.2.2 Frame Formats 22
3.3 Evaluation 25
3.3.1 Evaluation Methodology 25
3.3.2 Evaluation Results 25

Chapter 4 ER: Efficient Retransmission Scheme for WLANs 36

4.1 Approach 36
4.1.1 Overview 36
4.1.2 Receiver Feedback 38
4.1.3 Scheduling Algorithm 39
4.1.4 Coding Problem and Algorithms 40
4.2 Simulation 45
4.2.1 Simulation Methodology 45
4.2.2 Simulation Results 46
4.3 Testbed Experiments 55
4.3.1 Implementation 56
4.3.2 Experiment Methodology 56

4.3.3 Experiment Results 57

Chapter 5 Conclusion **61**

Bibliography **63**

Vita **68**

List of Tables

3.1 MAC layer retry counts for different frames. 24
3.2 Throughput of one flow under no wireless losses with fixed rate (Mbps). 26
3.3 Performance of two flows. 28

List of Figures

3.1 Pseudo-code for partial packet recovery 15
3.2 Frame formats. 22
3.3 Average throughput under single flow in four different link conditions. 27
3.4 Average Total Throughput vs #flows under fixed rate. 29
3.5 Average Total Throughput vs #flows under auto rate. 30
3.6 Jain's fairness index vs #flows under fixed rate. 30
3.7 Fairness vs #flows under auto rate. 31
3.8 CDF of throughput under fixed rate with 10 flows. 31
3.9 CDF of throughput under autorate with 10 flows. 32
3.10 Average throughput 10 flows. 33
3.11 Time series plot of rate adaptation. 34

4.1 Multicast comparison under a varying number of receivers with homogeneous Bernoulli losses. 47
4.2 Multicast comparison under a varying number of receivers with homogeneous Gilbert losses. 47
4.3 Multicast comparison under a varying loss rate with homogeneous Bernoulli losses. 49
4.4 Multicast comparison under a varying batch size with 10 receivers and 20% homogeneous Bernoulli loss rates. 50

4.5 Multicast comparison under a varying number of receivers with heterogeneous Bernoulli losses. 51
4.6 Multicast comparison under a varying loss bound with heterogeneous Bernoulli losses. 51
4.7 Unicast comparison under a varying number of receivers with homogeneous Bernoulli losses. 52
4.8 Unicast comparison under a varying loss rate with homogeneous Bernoulli losses. 53
4.9 Unicast comparison under a varying batch size with 10 receivers and 20% homogeneous Bernoulli loss rates. 54
4.10 Unicast comparison under a varying number of receivers with heterogeneous Bernoulli losses. 54
4.11 Unicast comparison under a varying loss bound with heterogeneous Bernoulli losses. 55
4.12 Multicast experiment results. 57
4.13 Compare throughput against the basic scheme for multicast under a varying loss rate. 58
4.14 Compare retransmission mechanisms for unicast under a varying loss rate. 59
4.15 Compare retransmission mechanisms for unicast under a varying loss rate, where only packets destined to the receivers are dropped. . . . 59
4.16 Unicast experiment results under a varying number of clients. 60
4.17 Compare throughput against the basic scheme for unicast under a varying loss rate. 60

List of Algorithms

1 RTO calculation in FRJ 16

2 Optimal segment size calculation 18

3 Rate adaptation in FRJ 20

4 RTO calculation in ER 40

Chapter 1

Introduction

The popularity of wireless networks has grown at a phenomenal rate. The proliferation of lightweight hand-held devices with built-in high-speed WiFi network cards and the significant benefit of any-where any-time Internet access has spurred the deployment of wireless local-area networks (WLANs) [4, 5]. Yet limited wireless spectrum, inherent lossy wireless medium, and imperfect packet scheduling reduce the efficiency and reliability of wireless networks. Emerging trends such as rapidly growing densities and increasing traffic volumes only exacerbate this problem.

1.1 Motivation

Performance improvement in wireless networks has attained significant research attention. Many novel techniques have been recently proposed in enhancing performance in wireless networks, ranging from designing packet recovery schemes to developing rate adaptation algorithms [6, 14, 26, 29, 43]. While each of these techniques addresses specific areas and are hence useful individually, none of them present a comprehensive solution.

Our motivation for this work is based on the observation that many of these

techniques could be combined to work in synchrony to provide substantial performance benefits. This also allows us to offset the negative effects of the applied techniques in many cases. For instance, consider a simple technique for increasing network throughput - sending larger frames. Larger frames result in MAC overhead reduction, and thus increases throughput. However, larger frames are subject to higher losses. As a result, larger frames do not perform well under lossy conditions. Hence, achieving *resiliency* is important to extracting performance. With recent advancements in the areas of packet recovery, we find that satisfactory levels of resiliency could be achieved by clever use of packet recovery techniques in conjunction with larger frame sizes. Additionally, larger frames offset the overhead incurred by packet recovery techniques.

However, it is not possible to achieve 100% resiliency in wireless networks due to it's inherent lossy nature. Thus, it is necessary to provide certain levels of *reliability* in the network. Common techniques to achieve reliable communication in wireless networks involve retransmissions. We observe that technique combination could work here too.

Based on our observations, we take a holistic approach to optimizing wireless performance and resilience in this work. Specifically, we propose techniques that address two key issues in WLAN performance – *resiliency* and *reliability*. Our first proposal, *Fast Resilient Jumbo frames (FRJ)*, tries to achieve resiliency by exploiting synergy between orthogonal techniques; and our second proposal, *Efficient Retransmission scheme (ER)*, builds on existing network coding work to provide reliable unicast, multicast and broadcast communication in WLANs.

1.2 FRJ

FRJ is built on our observation that strong synergy exists between three seemingly orthogonal design approaches for achieving resiliency: (i) supporting large frames,

(ii) partial packet recovery, and (iii) rate adaptation.

First, a natural way to boost network throughput is to use large frame size. However, the loss rate of a frame tends to increase with frame size. As a result, even though sending a jumbo frame boosts network throughput under no-loss scenarios, its performance would suffer significantly under either collision losses or inherent wireless medium losses. We find that effective partial packet recovery schemes allow the use of jumbo frames without incurring performance penalty due to lossy wireless medium.

Second, the use of jumbo frames reduces the cost of RTS/CTS, which helps to significantly reduce collision losses. Reduced collisions in turn help improve the effectiveness of packet recovery schemes. This is because most existing partial recovery techniques work best if the number of erroneous bits in a frame is small, whereas collisions tend to result in large erroneous bits.

Third, effective packet recovery reduces the frame loss rate. This allows use of higher transmission rates. Increased transmission rate reduces the medium occupancy duration and thus helps reduce contention. Further, it also improves chances of partial packet recovery.

FRJ uses jumbo frames to boost network throughput under good channel conditions. Additionally, it uses partial packet recovery to efficiently recover packet losses under bad channel conditions. It further jointly optimizes rate adaptation and partial packet recovery to achieve high transmission rates and protect against wireless losses.

1.3 ER

The main impetus behind ER is the observation that the average loss rate in deployments could be as high as 20-40% [2, 37]. ER is an efficient retransmission mechanism to support reliable unicast, broadcast, and multicast in WLANs. The

design of ER can also easily be extended to multi-hop wireless networks. We illustrate the idea of ER using the following two simple examples.

Consider two clients $C1$ and $C2$ associated with an access point (AP). The AP has two packets to send: $p1$ destined to $C1$ and $p2$ destined to $C2$. The links $AP - C1$ and $AP - C2$ both have 50% loss rates. Using the traditional unicast in IEEE 802.11 [28], on average 4 transmissions are required to successfully send packets to both clients. Due to the broadcast nature of wireless medium, $C1$ may lose $p1$ but receive $p2$; similarly, $C2$ may lose $p2$ but receive $p1$. Whenever this case occurs, ER reduces the number of transmissions by letting AP retransmit $p1 + p2$, which is $p1$ XOR-ed with $p2$, instead of sending $p1$ and $p2$ separately. Then $C1$ can extract $p1$ by xoring $p2$ with $p1 + p2$, and similarly $C2$ can extract $p2$ by xoring $p1$ with $p1 + p2$. In this way, AP reduces the number of transmissions (including the original transmissions) from 4 to 3 to successfully deliver both packets.

Now consider a multicast example. Since broadcast is a special case of multicast, in this paper we consider multicast without loss of generality. Suppose the AP wants to send both packets $p1$ and $p2$ to the clients $C1$ and $C2$. If both clients only receive one packet and the packets they receive are different, then AP can retransmit $p1 + p2$ instead of retransmitting them separately, thereby using one transmission to recover two packet losses.

In both unicast and multicast examples, ER takes advantage of wireless broadcast medium and minimizes the number of transmissions by effectively combining packets. As we will show later, the coding benefit further increases with the number of clients and/or the number of packets.

The design of ER is inspired by several recent works on network coding, in particular, COPE [16]. It complements the previous work in several important ways. First, the existing network coding approaches target multi-hop wireless networks and there are no coding opportunities for single hop paths. Instead we

show that the coding benefit also exists in widely-used single-hop WLANs. Second, the existing coding schemes, such as COPE, maximize efficiency by coding the original transmissions destined to different receivers, but relies on MAC-layer retransmissions to recover lost packets. In comparison, ER improves the efficiency of MAC-layer retransmissions by reducing the number of retransmissions required to recover the losses. Therefore ER can be applied to wireless LANs and multi-hop wireless networks to achieve efficient retransmission. When combined with COPE, it helps achieve high efficiency in both original transmissions and retransmissions of lost packets. Third, the coding opportunities in the previous work are determined by traffic demands. For example, coding opportunities arise in COPE when traffic heading towards different directions meet at the same intermediate router. Instead the coding opportunities in ER are determined by the loss patterns – more coding opportunities arise when different receivers lose different sets of packets. So understanding the coding benefits under realistic packet loss characteristics is an interesting and open question.

1.4 Contributions

Our work makes the following key contributions:

1. In Fast Resilient Jumbo frames (FRJ),

    (a) We present a mechanism to support jumbo frames by making them resilient to wireless losses. We believe ours is the first work to prototype resilient jumbo frames in wireless networks.

    (b) We propose a rate adaptation scheme which combines partial recovery and rate selection.

2. In Efficient Retransmission scheme (ER),

(a) We develop an efficient retransmission mechanism (ER) for unicast, broadcast, and multicast communications in WLANs. ER can potentially be applied to multi-hop wireless networks to further improve the efficiency of existing network coding, such as COPE.

(b) We formulate the problem of optimizing the sets of packets to code, and show it is NP-hard to optimize or even approximate. We also develop several algorithms for solving the problem, and show their effectiveness over the existing ones [16, 22]. Note that these algorithms are applicable to both WLANs and multi-hop wireless networks.

Our implementation of FRJ and ER is based on Madwifi driver [24] and Click toolkit [8]. Through extensive simulation and testbed experiments, we demonstrate the effectiveness of our proposals. Specifically, we observe that FRJ achieves up to 69% improvement over existing rate adaptation schemes, and that ER significantly reduces the number of retransmissions compared to the existing retransmission scheme, which retransmits the lost packets by themselves.

1.5 Outline

The rest of the thesis is organized as follows. In Chapter 2, we review the existing work on providing resiliency and reliable communication in wireless networks. In Chapter 3, we present the FRJ's design aspects, implementation details and experimental results. Chapter 4 describes ER's design details, followed by simulation and testbed results. Finally, we conclude and discuss future work in Chapter 5.

Chapter 2

Related Work

Performance improvement in wireless networks has always been an active area of research. Extracting performance from wireless networks is challenging because various reasons. First, the inherent lossy nature of wireless medium affects network throughput significantly. Second, limited wireless spectrum invites many new problems given the growth in deployment densities of wireless networks. Third, imperfect packet scheduling reduces the overall efficiency and reliability of wireless networks.

A wide variety of techniques have been proposed to enhance wireless performance by achieving resiliency and improving reliability: Packet recovery and channel scheduling reduces packet losses, thus resulting in increased throughput and better reliability. Rate adaptation techniques take advantage of the multi-rate capabilities provided by the physical layer to dynamically change transmission rate, and thus attain maximum transmission speeds. Techniques such as supporting larger sized frames builds on the fact that large frames result in lower transmission overhead. Recent proposals such as network coding is also driven by the same reason. In our work, we extend network coding techniques to retransmissions.

2.1 Packet Recovery

Of these techniques, a vast majority has focused on supporting reliable communication due to the fact that wireless medium is inherently lossy. Recovering packets that result in error is one intuitive way to combat against losses. Packet recovery can be done at various levels, as described below:

2.1.1 Retransmission

Retransmission is the most commonly used approach to recover packet errors and losses. Retransmissions require feedback from the receivers, specifying which packets are required for retransmissions. The feedback can be either ACKs or negative ACKs (NACKs) [28, 19]. IEEE 802.11 [28] uses a retransmission mechanism to improve the reliability of unicast traffic, but provides no reliability support to broadcast and multicast traffic. In 802.11 unicast, a station transmits the packet and waits for an ACK. If the receiver successfully receives the packet, it waits for a short inter-frame spacing time (SIFS) and then transmits an ACK frame. If the sender does not receive an ACK (e.g., due to a collision or poor channel condition), it retransmits the packet using binary exponential back-off, where its contention window is doubled every time after a retransmission until it reaches its maximum value, denoted as CW_{max}. In 802.11, the packet is retransmitted in its original form. When an AP unicasts to multiple clients via lossy links, the retransmission mechanism in IEEE 802.11 is not efficient, as illustrated in Section 1. This inefficiency has led the research community to explore intuitive ways to recover packets - such as adding redundancy or recovering parts of a corrupted packet.

2.1.2 FEC

FEC has been used to provide reliable unicast and multicast communication in both wireless networks (*e.g.*, [36, 44, 25, 31, 21, 45]) and wireline networks (*e.g.*,

[7, 34, 35, 39]). For example, in [25], McKinley et al. dynamically adjusts the level of FEC redundancy based on observed channel quality. [21] points out that many existing FEC-based work incorrectly assume independent packet losses, and studies the impact of spatial and temporal correlation of packet losses on FEC schemes. In addition to network performance, [45] analyzes the tradeoff between improving

2.1.3 Partial Packet Recovery

Miu et al. [26] develop MRD, which leverages multiple receivers to recover corrupted packets. When the same packet received at multiple receivers differs in one or more blocks, MRD exhaustively searches over all possible block combinations to find the one that passes the packet checksum. As MRD, SOFT proposed by Woo et al. [43] also takes advantage of multiple receivers for packet recovery. Different from MRD, SOFT also exploits the physical layer information and develops an efficient combining strategy that maximizes the likelihood of recovering a packet. Kyle et al. [14] describe PPR scheme in which a single receiver performs partial packet recovery. In this scheme, the receiver leverages the confidence information at the physical layer to identify bits with high uncertainty and requests retransmission of these bits. By using PHY information, PPR out-performs fragment-based partial recovery, proposed by Ganti et al. [11], however its performance improvement is usually around 25%.

In addition to lossy channels, packet collision is another major source of packet losses in wireless networks. Packet collisions occur when two or more nodes transmit at the same time. Such a scenario occurs when these nodes are not within the carrier sensing range of each other. In such scenarios, effective channel reservation schemes plays a vital role.

2.2 Channel Reservation

For unicast traffic, binary exponential back-off and RTS/CTS are used to reduce collision losses and avoid hidden terminals. Due to expensive feedback, neither schemes are applicable to multicast/broadcast traffic [28] and the collision losses of multicast/broadcast traffic can be quite high. Motivated by this observation, several channel reservation schemes have been proposed to reduce collision losses for multicast traffic, such as Broadcast Support Multiple Access (BSMA) [40], Broadcast Medium Window (BMW) [41], Batch Mode Multicast MAC protocol (BMMM) [13], and Leader based Priority Ring Multicast Protocol (LPRMP) [10]. These protocols are complementary to loss recovery schemes using retransmissions with or without source/network coding, and can be used in combination with ER. In particular, the channel reservation schemes reduce collision losses, while the retransmissions can recover both collision and other wireless medium related losses (*e.g.*, those due to fading and low SNR).

All of these techniques have had significant impact on reducing packet losses and enhancing reliability in wireless networks. However, achieving reliability alone is not sufficient for increasing performance gains. Consequently, efforts have been made to explore other arenas for improving wireless performance. Maximizing transmissions speeds helps attain higher network throughput. A number of techniques have been proposed to optimize transmission speed, such as rate adaptation, transmitting large frames, and network coding. Since the PHY layer provides multi-rate capabilities, dynamically adapting transmission rate depending on channel conditions is a promising way to improve wireless performance by selecting the best data rate.

2.3 Rate Adaptation

Rate adaptation has received significant research attention and various rate adaptation algorithms have been developed [12, 38, 33, 20, 6, 42, 17, 29]. For example, [29] is the rate adaptation algorithm used in MadWiFi driver. It uses long-term loss rate estimation and threshold to determine rate changes. More recently, Wong et al. [42] identify the limitations of existing design guidelines for rate adaptation. Based on their observations, they develop Robust Rate Adaptation Algorithm (RRAA), which uses short-term loss ratio to opportunistically change rate and incorporates an adaptive RTS filter to prevent collision losses from causing rate decrease.

While rate adaptation focuses on selecting optimal rate for a packet, another way to achieve higher transmission speed is to reduce the overhead associated with packet transmission. Thus, transmitting jumbo packets is a natural way to improve throughput.

2.4 Jumbo Frames

To our knowledge there exists no previous work that explores the use of jumbo frames in wireless networks. IEEE 802.11 standard supports a maximum MTU size 2300 bytes (approx). Thus, we believe ours is the first work that explores the use of jumbo frames in wireless networks. Atheros' Super G [1] includes an optimization called fast frames whereby two consecutive data frames are combined and transmitted as a single frame. This could result into large frame sizes.

Finally, information theory techniques such as network coding has proved to be useful for extracting performance in multicast scenarios by reducing the number of transmissions.

2.5 Network Coding

The pioneering work by Ahlswede et al. [3] shows that allowing relay nodes to encode and decode traffic can achieve maximum multicast rate, and this is generally more efficient than only allowing the relay nodes to forward traffic. Since then, lots of progress has been made in applying network coding to wireless and wireline networks (*e.g.*, [18, 23, 16, 22]). In particular, COPE [16] develops a practical network coding scheme for unicast in multi-hop wireless networks and [22] further extends the idea to broadcast. Both works focus on multihop wireless networks, and use network coding for the initial transmissions. COPE relies on MAC-layer retransmissions to recover packet losses, while [22] does not consider loss recovery.

Our work differs from the existing ones in the following ways:

- FRJ aims to export a larger link layer MTU to the upper layers in contrast to fast frames. Further, to handle wireless losses FRJ combines the idea of jumbo frames with partial packet recovery.
- Although ER is built on existing network coding work, it applies the coding concept to provide efficient MAC-layer retransmissions and is complementary to the existing work.

Chapter 3

FRJ: Fast Resilient Jumbo Frames in WLANs

This chapter describes the design, implementation and evaluation details of *FRJ: Fast Resilient Jumbo frames in WLANs*.

3.1 Approach

Our approach intends to find synergies between three different techniques for performance improvement in 802.11 networks - (i) jumbo frames (ii) partial packet recovery and (iii) rate adaptation.

Transmission of large data (jumbo) frames helps achieve higher throughput by avoiding MAC overhead. However, since wireless medium is inherently lossy, jumbo frames are susceptible to higher loss rates. To overcome this, jumbo frames need to be combined with partial recovery, thus making them more resilient.

It has been proven that use of rate adaptation can translate into significant throughput improvement. However, most of the current rate adaptive schemes depend on frame loss rate to estimate a new rate. In contrast, we propose a new rate

adaptation algorithm which takes advantage of partial recovery.

In the following subsections, we describe how our work combines these techniques:

3.1.1 Resilient Jumbo Frames

As mentioned in section 1, using jumbo frames allows us to: (i) effectively lower MAC overhead, and (ii) reduce the cost of using RTS/CTS. On the other hand, jumbo frames are subject to higher losses in wireless networks. Partial packet recovery techniques can be applied to make them more resilient to such losses.

We adopt a segment based partial packet recovery scheme described by Ganti et. al in [11], as shown in figure 3.2:

Basically, each upper layer datagram is fragmented into segments, and each segment is appended with a segment-CRC. Next these segments with segment-CRC are assembled into a frame which is handed over to MAC layer for transmission. The receiver assembles all correctly received segments and requests retransmission of corrupted segments via 2.5 layer ACKs periodically.

We distinguish ourself from the scheme in [11] by the use of variable segment sizes, the motive behind which is described in the following sections. In addition, we also introduce a few modifications as optimizations (Since the partial packet recovery protocol acts as a thin layer just above MAC layer, we henceforth refer to it as 2.5 layer protocol):

Receiver feedback: For partial recovery, we use 2.5 layer cumulative ACKs. These ACKs convey status of received segments to the sender. These ACKs are useful in the following cases: (i) when 802.11 MAC ACK is lost, or (ii) when some segments are received in error. On receipt of 2.5 layer ACKs, sender retransmits only segments which were received in error. To minimize the overhead of such ACKs, a 2.5-layer ACK is sent only when a threshold number (N) of new frames have been

```
SENDER

1  Divide upper layer datagram into segments
2  For each segment, add segment-CRC. Pack
   into frame, add header, hand over to MAC
   layer for transmission
4  Buffer transmitted segments
5  If ACK is not received within timeout
6      sender retransmits frame segments
7  else if completely ACKed
8     remove frame segments from buffer
9  else if partially ACKed
10     retransmit requested segments

RECEIVER

1  On receipt of a frame:
2  If all segments received correctly
3      reassemble and propagate up
4  else
5      buffer correct segments
6  On receipt of N frames or after
   timeout send 2.5 layer ACK
```

Figure 3.1: Pseudo-code for partial packet recovery

received or after a timeout.

In addition to 2.5 layer ACKs, we also use default 802.11 MAC ACKs. MAC ACKs indicate the receipt of the complete frame without any corruption. The sender uses these to remove the buffered segments for the ACKed frame.

Retransmission timeout: When a frame is not acknowledged by either MAC ACK or 2.5-layer ACK, it requires retransmission. We use a standard approach to estimate retransmission timeout (RTO), similar to TCP [32]. Specifically, for every frame that has not been retransmitted, a node measures the time difference

between when the frame was transmitted and when the corresponding ACK was received. Let T denote the measured round-trip time of the current frame. Then the node updates its RTO based on smoothened RTT and RTT variance as shown in algorithm 1. RTO is initialized based on the MAC data rate. Our evaluation uses $K = 8$, $\alpha = 1/8$, and $\beta = 1/4$.

if *(T is the first RTT measurement)* **then**
  SRTT = T;
  RTTVAR = T/2;
  $RTO = SRTT + K \times RTTVAR$;
else
  $RTTVAR = (1 - \beta) \times RTTVAR + \beta \times |SRTT - T|$;
  $SRTT = (1 - \alpha) \times SRTT + \alpha \times T$;
  $RTO = SRTT + K \times RTTVAR$;
end

Algorithm 1: RTO calculation in FRJ

3.1.2 Partial Recovery Aware Rate Adaptation

Rate adaptation is critical to the performance of IEEE 802.11 networks. Several rate adaptation schemes have been proposed in the past and it has been shown that tremendous throughput improvements can be achieved with careful rate adaptation. However, the existing rate adaptation schemes identify the optimal rate based on the frame loss rate. Our intuition behind a new rate adaptation scheme is the additional freedom provided by partial packet recovery: since it is now possible to adjust segment sizes in a frame, a rate adaptation scheme is not restricted on choosing rate based on frame loss rate. Instead, out partial recovery aware rate adaptation scheme computes optimal rate and segment size that maximizes throughput for a given channel condition.

A new rate adaptation scheme invites several challenges. Some immediate questions that need to be answered are: (i) how to estimate the channel conditions

accurately?, (ii) how to select an optimal segment size for a given rate? and (iii) how to choose the rate and segment size combination that maximizes throughput? We address these challenges in the sections below:

Estimation of channel conditions

Our rate adaptation scheme intends to find the optimal segment size. In order to find the loss rate for a given segment size, we estimate the channel condition using Bit Error Ratio (BER) and header loss rate (HL). HL is defined as the ratio of frames that are lost due to header or preamble corruption. Given BER and HL, it is easy to compute segment delivery ratio for various segment sizes. For instance, the delivery probability of a segment with size SS is given by equation 3.1:

$$(1 - HL)(1 - BER)^{SS} \tag{3.1}$$

We use probing to estimate BER and HL. A sender broadcasts probe packets periodically at different data rates to measure channel quality. The probe packet contains a known payload so that the receiver can easily identify corrupted bits. The receiver computes BER as the ratio of the number of corrupted bits and the total number of bits received. HL is the ratio of the total number of probe packets that were lost to the total number of expected probe packets in a given probing interval. This BER and HL estimate is then communicated to the sender in a probe response packet.

To limit the probing overhead, a sender transmits probes only at three data rates: (i) its current data rate ($CurrRate_r$), (ii) one data rate below its current data rate ($CurrRate_r^-$), and (iii) one data rate above its current data rate ($CurrRate_r^+$). Further, the sender sends probes at the rate of 2 probes per second per rate, and receiver sends probe responses every 5 seconds. We found that these values are sufficient to estimate the correct BER and HL values in our testbed.

Estimation of optimal segment size for a given rate

A smaller segment size reduces the loss rate per segment, but increases the header overhead (due to segment-CRC). On the other hand, a larger segment size results in an increased loss rate per segment. To strike a good balance between minimizing segment loss rate and header overhead, we use a simple search algorithm, shown in Figure 2 to optimize segment size. More specifically, given frame size FS, segment size SS, segment header size SH, header loss rate HL, and bit-error-rate BER for a given rate $rate$, we can compute throughput (*i.e.*, the receiving rate of correct bits excluding frame and segment headers) as

$$rate * (1 - HL) * (1 - BER)^{SS} * \frac{numSegs * (SS - SH)}{FS} \tag{3.2}$$

where $numSegs$ is the number of segments in the frame given by (FS / SS).

We identify the optimal segment size by searching over all possible values of segment size as given in Algorithm 2.

Data: $rate$, BER, HL
Result: Optimal segment size $bestSS$
Initialize $bestThroughput$, $bestSS$;
for *i=1* **to** *FS/MinSegmentSize* **do**
  $SS = FS/i$;
  $throughput = (rate)(1 - HL)(1 - BER)^{SS}(\frac{i \times (SS-SH)}{FS})$;
  if $throughput > bestThroughput$ **then**
    $bestThroughput = throughput$;
    $bestSS = SS$;
  end
end

Algorithm 2: Optimal segment size calculation

For fixed rate scenarios, we can use the above optimal segment size calculation to obtain higher throughput. In the following section, we augment the above algorithm to a rate adaptation scheme.

Rate adaptation

As discussed in subsection 3.1.2, we obtain BER and HL at three different rates: $CurrRate_r$, $CurrRate_r^-$, and $CurrRate_r^+$. For each of these three rates, we use the algorithm in Figure 2 to calculate the $bestThroughput$ and $bestSS$. The rate that maximizes $bestThroughput$ is chosen as the new rate, and the corresponding $bestSS$ as the new segment size.

Our rate adaptation scheme relies on the reliable channel estimation through probing. However, probes and probe responses are also subjected to losses. A robust rate adaptation scheme needs to account for such adverse conditions. We next discuss how our rate adaptation scheme adapts to various wireless link conditions:

Case 0 - Good reverse link: When the reverse link is good, the probe responses will provide BER and HL at three different rates. In this case our rate adaptation will work as designed. See Figure 3 for complete algorithm of our rate adaptation scheme.

Case 1 - Bad reverse link: When the reverse link is bad it is possible that we do not receive any probe responses. However we may receive occasional 2.5 layer ACKs since they are sent more frequently when compared to probe responses. In this case, our rate adaptation scheme will estimate the BER at current rate $CurrRate_r$ using segment delivery ratio (SDR) as shown in equation 3.3.

$$SDR = (1 - BER)^{SS} \tag{3.3}$$

where SDR is computed over the last probe interval as:

$$SDR = \frac{\#segments\ acked}{\#segments\ sent} \tag{3.4}$$

Since BER is lower for a lower data rate, we conservatively assume BER=0 for $CurrRate_r^-$ using regular frame size (1500 bytes). We then use algorithm 2 for

```
Data: BER and HL at CurrRate_r,CurrRate_r^-,CurrRate_r^+
Result: newRate, newSegSize
Initialize CurrRate_r, maxThroughput, CurrSegSize;
Broadcast probes at CurrRate_r,CurrRate_r^-, and CurrRate_r^+;
if probe response received within timeout t_timeout then
    for i=(CurrRate_r,CurrRate_r^-,CurrRate_r^+) do
        compute bestThroughput_i, bestSS_i using algorithm 2;
        if bestThroughput_i > maxThroughput then
            newRate = i;
            newSegSize = bestSS_i;
        end
    end
else
    if 2.5 layer ACKs received within timeout t_timeout then
        compute BER at CurrRate_r using equation 3.3;
        assume BER=0 for CurrRate_r^-;
        for i=(CurrRate_r,CurrRate_r^-) do
            compute bestThroughput_i, bestSS_i using algorithm 2;
            if bestThroughput_i > maxThroughput then
                newRate = i;
                newSegSize = bestSS_i;
            end
        end
    else
        if CurrRate_r is lowest then
            newSegSize = CurrSegSize / 2;
        else
            newRate = CurrRate_r^-;
        end
    end
end
CurrRate_r = newRate;
CurrSegSize = newSegSize;
```

Algorithm 3: Rate adaptation in FRJ

each of these two rates $CurrRate_r$ and $CurrRate_r^-$. We select the rate that gives the higher throughput.

Case 2 - No reverse link: When there is no reverse link, both probe responses and 2.5 layer ACKs are lost. In such case, the rate adaptation scheme

will not get any estimation of the channel condition. To handle such such scenarios, our scheme will step down the rate to the next lower rate after a timeout period. If the current rate is already the lowest rate, the segment size is reduced to half.

The complete rate adaptation algorithm is presented in Algorithm 3.

3.2 Implementation

3.2.1 Overview

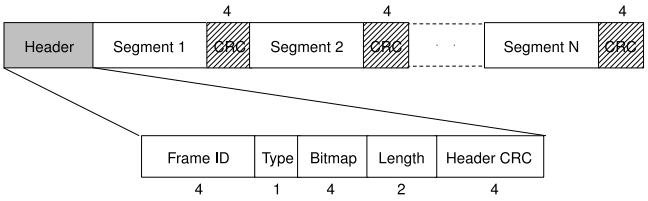
We implement FRJ on PCs running Linux and equipped with 802.11 a/b/g NetGear WAG511 wireless cards using Atheros chipset. To support jumbo frames we modified Madwifi-old driver [24]. The partial recovery and rate adaptation algorithms have been implemented in Click modular router [8]. We use Click in userlevel mode.

We use Madwifi driver in monitor mode since it allows us to prepend radiotap headers in Click to configure the rate and retry count for every outgoing packet. We modified the driver in two ways: (i) support jumbo frames of size up to 3000 bytes, and (ii) to disable MAC layer CRC check for 802.11 frames. The first modification is required to export a larger link layer MTU size to support jumbo frames. Although our current setup supports jumbo frames of size up to 3000 bytes, our scheme is applicable for even larger frame sizes. Since the Madwifi driver does not support jumbo frames, we were unable to implement frames larger than 3000 bytes. The second modification is required for our partial packet recovery scheme. By disabling the MAC layer CRC check in the driver, packets with errors are propagated to Click. These packets are then processed to recover segments that are not in error.

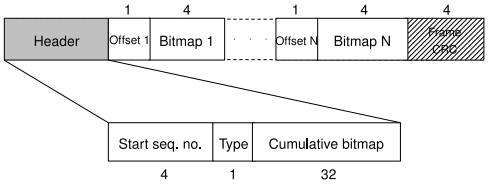
Our implementation uses four different frame formats: (i) Data frame (ii) 2.5 layer ACK frame (iii) Probe frame and (iv) Probe response frame.

3.2.2 Frame Formats

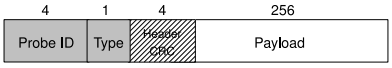
The different frame formats are as shown in Figure 3.2. Each frame type is identified using a unique frame ID. The *frametype* field is used to identify different frame types.



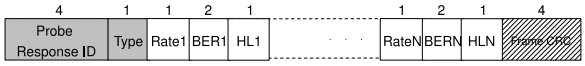
(a) Data frame



(b) 2.5 layer ACK



(c) Probe



(d) Probe response

Figure 3.2: Frame formats.

Data Frame

A data frame consists of a 15-byte frame header, followed by frame segments. A 32-bit segment-CRC follows each frame segment. The frame header consists of frame ID, frame type, bitmap, length and header CRC. The bitmap is a 32-bit field which indicates the segments present in the current frame. A bit value of 1 at position i implies $segment_i$ is present in the current frame. For instance, if the frame contains 32 segments, the bitmap contains a string of 1's. The bitmap for a retransmitted frame will indicate the segments that are being retransmitted. The receiver can then combine these segments with already received segments of the same frame. Header CRC field protects the entire header. In order to allow partial packet recovery, we disable MAC layer retransmission of data frames by setting the retry count to 0.

2.5 layer ACK Frame

FRJ uses cumulative ACKs (2.5 layer ACKs) to reduce the ACK overhead and minimize the impact of ACK losses. These ACKs convey the status of the received segments to the sender using a bitmap field. The 2.5 layer ACK frame contains: (i) a starting sequence number ($start$) in the header followed by a cumulative bitmap $Bitmap_c$ (128 bits), and (ii) payload consisting of a sequence of two tuple ($offset$, $Bitmap_F$).

$start$ and $Bitmap_c$ together determine the window of frame IDs for which the 2.5 layer ACK contains a status. All the packets up to $start$ are assumed to be received completely. $start$ is the first frame for which the ACK contains a status. For every frame from $start$ to ($start$ + 127), the corresponding bit in $Bitmap_c$ is 0 or 1. A 1 bit indicates that the corresponding frame was received completely. A 0 bit indicates that the corresponding frame F_i was either (i) not received, or (ii) received partially. In this case, the payload of the 2.5 layer ACK will contain the two tuple ($offset_i$, $Bitmap_{F_i}$) for the corresponding frame. The $offset_i$ is the offet

Frame Type	Retry Count
Data frame	0
2.5 layer ACK frame	16
Probe frame	0
Probe response frame	16

Table 3.1: MAC layer retry counts for different frames.

of the bit corresponding to F_i in $Bitmap_c$. $Bitmap_{F_i}$ is the frame bitmap computed by the receiver which indicates which segments were received correctly in F_i.

We update *start* so that the largest received packet is no more than $start + 128$. This implies that it is possible that a node may not have received all packets up to *start* even though it assumes so. The likelihood of such occurrence is low since the bit-map size of 128 bits is generally large enough. Moreover, FRJ is designed to provide best-effort reliability and leaves the upper-layer to ensure full reliability if needed. To make the 2.5 layer ACK reliable, we use a retry count of 16.

Probe Frame

A probe frame consists of Probe ID, frame type, header CRC, and payload. The payload contains known pattern that is used to identify corrupted bits and compute BER as described in Section 3.1.2. The probe ID field is auto-incremented for each probe. Gaps in probe IDs are accounted towards HL. To reduce overhead, we broadcast probes.

Probe Response Frame

A probe response frame includes probe response ID, type, payload and frame CRC. The payload consists of a three tuple ($Rate$, BER, HL) for every rate at which probes were received. Since the whole probe response needs to be accurate, the complete frame is protected by a CRC. Probe responses also have retry count set to 16.

Table 3.1 lists the retry counts for various frame types.

3.3 Evaluation

In this section, we first describe our testbed and evaluation methodology, and then present evaluation results under a variety of scenarios.

3.3.1 Evaluation Methodology

We evaluate FRJ in our testbed, which consists of several DELL dimensions 1100 PCs, located on two adjacent floors of an office building. Each machine has a 2.66 GHz Intel Celeron D Processor 330 with 512 MB of memory and is equipped with a 802.11 a/b/g NetGear WAG511 wireless card. For all our experiments (both controlled and testbed), we use 802.11a to avoid interference from our campus networks that use 802.11b/g. We evaluate the performance benefit of FRJ using a combination of controlled experiments and 24-node testbed experiments. We use the controlled experiments to evaluate the performance gain in single flow and two flow cases. In addition, we use a 24-node testbed to evaluate our scheme in a more general setting. Throughout the evaluation, we use throughput (*i.e.*, total number of correctly received non-duplicate bits per second) as the performance metric. We compare FRJ against regular frames. We consider both fixed rate and auto rate scenarios. Our rate adaptation scheme is compared against SampleRate [6].

3.3.2 Evaluation Results

In this section, we present the performance results from both controlled experiments and a 24-node testbed.

Under no wireless losses

First we evaluate the performance of FRJ under no wireless losses using fixed rate. This evaluation projects the benefits of using jumbo frames. Here, the main benefit of FRJ comes from reduction in MAC-layer overhead. As shown in Table 3.2, FRJ

Type	6Mbps	54Mbps
FRJ (3000B)	5.6	32.18
Regular (1500B)	5.43	28.60
Regular (1000B)	5.22	25.15
Regular (576B)	4.75	16.76

Table 3.2: Throughput of one flow under no wireless losses with fixed rate (Mbps).

with payload of 3000 bytes out-performs regular sized frames using payloads of 1500, 1000, and 576-bytes by 3-12%, 7-28%, and 18-92%, respectively. As we would expect, the benefit of FRJ is larger when compared with regular frames using a smaller sized payload. In addition, its benefit increases with the data rate, because MAC-layer overhead is larger under higher data rate.

Single Flow Results

To evaluate how FRJ behaves under different link conditions, we evaluate the performance of FRJ under four cases with fixed rate. These four cases indicate different link conditions. We classify a directional link as good if the delivery ratio is more than 80% and bad otherwise. Thus, in Figure 3.3, Good-Bad indicates a good forward link and a bad reverse link. This experiment aims to show the advantage of jumbo frames and partial recovery under different link conditions. We compare FRJ with regular frames of 1500B size, and repeat the experiment with and without RTS.

Figure 3.3 indicates that FRJ performs better than regular frames in all four link conditions. Under Good-Good link condition, most of the frames are delivered completely. Hence, the chances of partial recovery are extremely less. Thus, the benefit from partial recovery in this case is almost nil. Additionally, the overhead due to partial recovery reduces the gain. The relatively small performance gain of FRJ with respect to the regular frames indicate this fact. However, as the forward link condition degrades, the benefit of partial recovery becomes more evident. The gain from partial recovery is most evident in Bad-Bad link condition. Here, FRJ

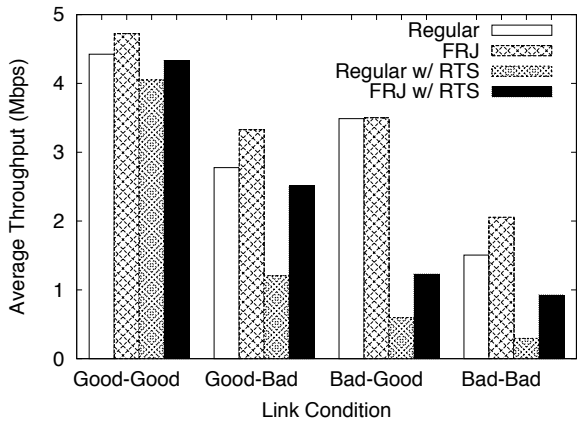


Figure 3.3: Average throughput under single flow in four different link conditions.

outperforms regular frames by as much as 37% without RTS/CTS and up to 217% with RTS/CTS. Since the experiments are conducted for a single flow, turning on RTS/CTS degrades the performance due to the additional overhead it incurs.

Two Flow Results

To estimate the fairness of FRJ, we conduct three different two-flow experiments as follows: (i) *BothRegular*: both flows use regular frames, (ii) $BothFRJ$: both flows use FRJ, and (iii) $Mixed$: the first flow uses FRJ and the second uses regular frames. We perform the experiment for both fixed rate and autorate. The results are as indicated in table 3.3 (a) and (b).

Fixed rate results: Table 3.3 (a) indicates that FRJ is fairer than regular frames. When both the flows use FRJ, both achieve similar throughputs. This is as expected. In $Mixed$ flow, we note that the FRJ flow contributes to 66% of the total throughput. This is expected since the payload for FRJ (3000B) is twice that of regular (1500B).

Auto rate results: As seen in fixed rate case, when both the flows are FRJ, they achieve similar throughput. In this case, $BothFRJ$ outperforms *BothRegular* by

Type	Flow 1	Flow 2	Total
BothRegular	3.25	2.25	5.5
BothFRJ	2.99	2.65	5.64
Mixed	3.72 (FRJ)	1.86 (Regular)	5.58

(a) Under fixed rate.

Type	Flow 1	Flow 2	Total
BothRegular	10.78	12.75	23.53
BothFRJ	13.03	14.92	27.95
Mixed	17.28 (FRJ)	5.69 (Regular)	22.97

(b) Under auto rate.

Table 3.3: Performance of two flows.

19%. In *Mixed* case, we observe that FRJ's contribution is higher than twice that by regular flow. This could be attributed to the fact that the overhead for regular frames is higher at higher rates compared to FRJ.

From the results, it is evident that FRJ can co-exist with regular frames without throttling the regular flow.

Multiple Flow Results

To evaluate the performance of FRJ under multiple flow setting, we run experiments with different number of simultaneous flows: 1, 2, 4, 8 and 10 flows. For each number of flows, we repeat the experiment 10 times, selecting random pairs every time. We use average total throughput, Jain's fairness index, and throughput distribution as our evaluation metrics.

Average total throughput: For each number of flows, we report the average of total throughput of all flows. For instance, in 10 flow scenario, the total throughput is the sum of the individual throughputs of all the 10 flows. We report the average

of 10 such runs.

The results for fixed rate are shown in Figure 3.4. We note that FRJ w/ RTS/CTS consistently outperforms both regular w/ and w/o RTS/CTS. Further, the improvement increases with the number of flows. The improvement achieved by FRJ in this case is in the range 14-19%. With increase in number of flows, FRJ can make optimal use of RTS/CTS without incurring huge MAC overhead.

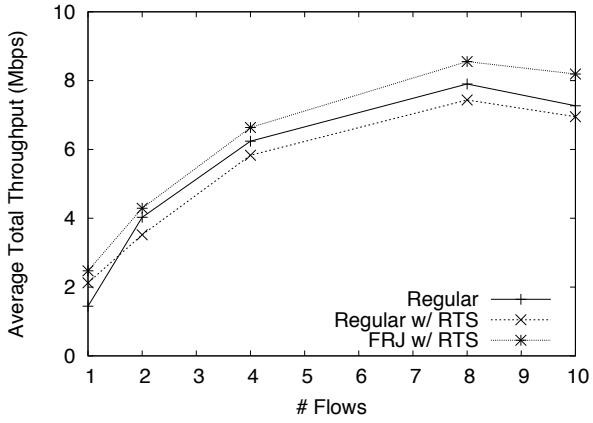


Figure 3.4: Average Total Throughput vs #flows under fixed rate.

Autorate results are shown in Figure 3.5. The improvement achieved by FRJ in this case is 69% over SampleRate w/ RTS and 223% over SampleRate w/o RTS. With a single flow, the RTS/CTS overhead incurred by regular frames is high. Hence, when the number of flows is one, we see that turning on RTS/CTS infact degrades the performance of regular frames. However, as the number of flows increase, we note the benefits of using RTS/CTS. These are most evident for FRJ, since the overhead incurred by FRJ, even with the use of RTS/CTS, is low compared to regular frames. In addition, FRJ also makes use of partial recovery in combination with rate adaptation which results in increased gains.

These results indicate the effectiveness of the combination of partial recovery and rate adaptation in jumbo frames. More specifically, the auto rate results indicate

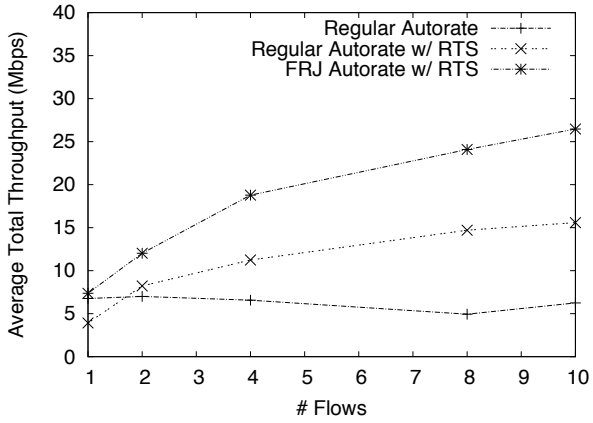


Figure 3.5: Average Total Throughput vs #flows under auto rate.

the importance of choosing an optimal segment size and rate.

Fairness: To evaluate if the nodes are receiving a fair share of the channel and to check how FRJ compares with the regular frame, we calculate Jain's fairness index. Figure 3.6 shows the result for fixed rate.

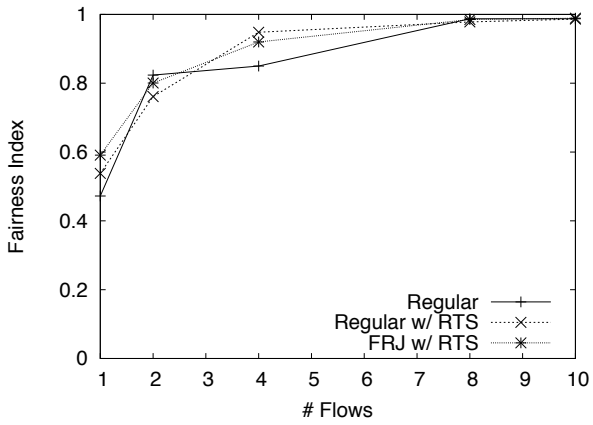


Figure 3.6: Jain's fairness index vs #flows under fixed rate.

In fixed rate case, we observe that FRJ is as fair as regular frames.

Figure 3.7 shows the result for auto rate case. In this case, we note that FRJ is fairer compared to regular frames, especially when the number of flows increases.

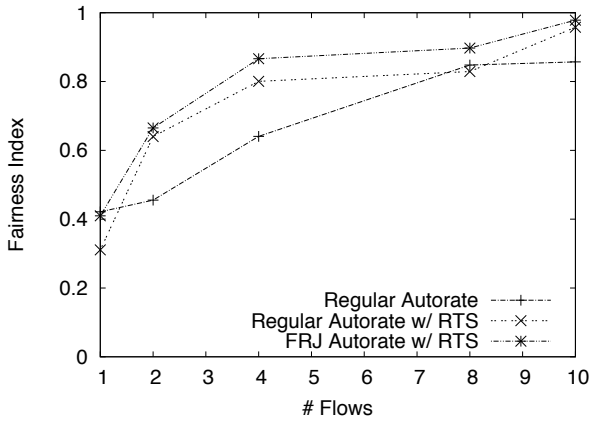


Figure 3.7: Fairness vs #flows under auto rate.

Throughput distribution: In this section, we analyze the CDF of throughput for 10 simultaneous flows. Figure 3.8 shows the plot for fixed rate.

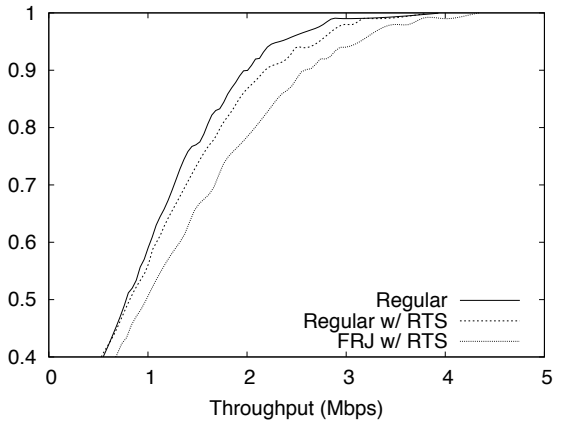


Figure 3.8: CDF of throughput under fixed rate with 10 flows.

FRJ w/ RTS has a higher probability of achieving higher throughput compared to regular frames w/ and w/o RTS. For instance, with 80% probability, FRJ achieves a throughput of 2Mbps while regular frames achieve 1.5Mbps and 1.3Mbps respectively w/ and w/o RTS.

Figure 3.9 shows the plot for auto rate.

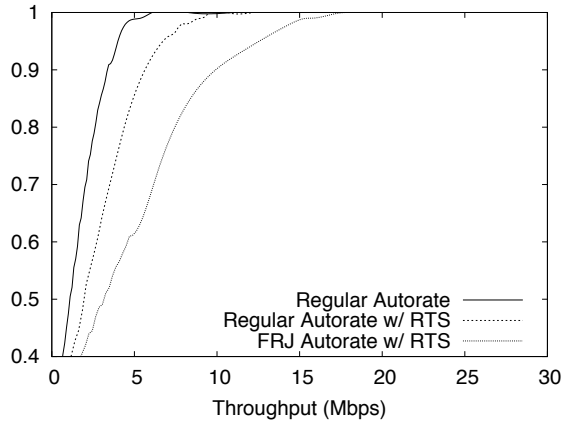


Figure 3.9: CDF of throughput under autorate with 10 flows.

For autorate, we see similar trends as seen in fixed rate scenario. Here, the magnitude of improvement achieved by FRJ is significantly higher. For instance, with 90% probability, the throughputs for the three cases are: 10Mbps, 5Mbps and 2.5Mbps.

Understanding performance gain in FRJ

The previous sections show that FRJ significantly out-performs regular frames. In order to understand where the benefits in FRJ come from, we systematically remove one component from FRJ at a time and compares the resulting scheme with both FRJ and regular frames. More specifically, we conduct experiments with: (i) disabling partial packet recovery, and (ii) using 1500 byte payload instead of jumbo frame. Figure 3.10 shows the total throughput under 10 simultaneous flows averaged over 5 random runs. We make the following observations. The two variants of FRJ perform only comparable with regular frames, and considerably worse than FRJ. This demonstrates that each technique alone is insufficient. FRJ effectively

exploits the synergy between them to maximize the throughput. In this case, we note that FRJ has 24% improvement over SampleRate with RTS and 81% over SampleRate without RTS.

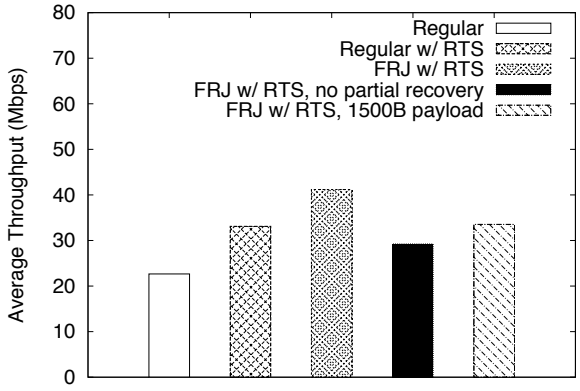


Figure 3.10: Average throughput 10 flows.

Understanding rate adaptation in FRJ

The maximum benefit of FRJ is achieved with our rate adaptation scheme. To understand how it performs better than SampleRate, we present time-series plot of FRJ and SampleRate in Figure 3.11. We log the rate used by both these rate adaptation schemes over a period of 60 seconds. Additionally, for our rate adaptation scheme, we also note the segments size used. We observe that FRJ is able to maintain higher transmission rates by choosing lower segment size when the link conditions are adverse. Both algorithms start at 24Mbps. To be conservative, FRJ chooses an initial segment size of 1500 bytes. After 5 seconds, FRJ receives probe response and based on it decides to move to a higher rate. After 10 seconds until the end of 15 seconds, FRJ is able to maintain a higher rate by reducing segment size to 100 and 120 bytes. SampleRate in this case remains at 24Mbps. It is to be noted that SampleRate never chooses 54Mbps as the transmission rate due to the link

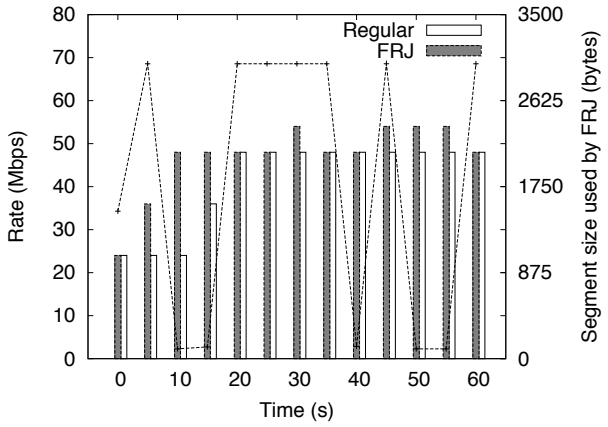


Figure 3.11: Time series plot of rate adaptation.

conditions. FRJ on the other hand chooses to transmit at the highest rate 4 times during the 60 second interval, each time carefully selecting a segment size based on BER and HL estimation using probing.

This experiment signifies the need for a partial recovery aware rate adaptation in the design of FRJ.

In this particular experiment, we recorded a throughput of 25Mbps for FRJ and 14Mbps for regular frames which translates to a 78% improvement.

Summary

Based on our evaluation, we make the following observations:

- Under single-flow, FRJ improves throughput by up to 92% under good channel condition by reducing MAC overhead. As the channel condition degrades, the benefit of FRJ becomes even larger due to partial packet recovery.

- Under multiple flows, FRJ out-performs regular frames by 69-223% (Figure 3.5) under autorate. The improvement increases as the number of flows increase. This improvement can be attributed to the optimal rate and segment

size selection. Such multiple flow scenarios are common in today's WLAN deployments.

FRJ achieves higher throughput compared to regular frames under different channel conditions and multiple flow scenarios. As evident, each of the individual components in FRJ namely jumbo frames, partial packet recovery and rate adaptation scheme has their own merits. However, our evaluation underlines the fact that the real power of FRJ lies in the combination of all components.

Chapter 4

ER: Efficient Retransmission Scheme for WLANs

This chapter details the approach, design and evaluation of *ER: Efficient Retransmission Scheme for WLANs*.

4.1 Approach

ER can be applied to single-hop WLANs and multihop wireless networks in the same way. In the following description, a sender refers to an AP in a wireless LAN, or refers to a traffic source or an intermediate router in a multihop wireless network.

4.1.1 Overview

First, we consider unicast transmissions. In ER, a sender maintains two packet queues: one for new packets and the other for retransmission packets. In the new packet transmission mode, the sender sends packets from its new packet queue, following 802.11's contention mechanism. Since ER is a replacement of the MAC-layer retransmission in 802.11, the sender disables the default MAC-layer retransmission

by setting the MAC retry count (*i.e.*, the maximum number of retransmissions at MAC-layer) to 0. ER retransmits the packet above the MAC-layer until its receiver acknowledges the packet or the retry count in ER is reached. To provide the same level of reliability, the retry count in ER is set to the original MAC retry count.

The receivers periodically send feedback of which packets are received successfully. Based on the feedback, the sender puts the packets that require retransmissions to the retransmission queue. In the retransmission mode, the sender examines all the packets in its retransmission queue to determine which sets of packets to code in order to minimize the number of retransmissions. The sender uses MAC-layer unicast to send both new and retransmitted packets, while all the other nodes use promiscuous mode monitoring so that they can receive packets destined to other nodes, which is necessary to create coding opportunities. Unicast is used in this case because its binary exponential backoff can help reduce collision losses under high load and it also allows the use of RTS/CTS to avoid hidden terminals if needed.

ER can be applied to multicast traffic in a similar manner. As in unicast, the sender also maintains two queues and switches between them for sending new and retransmitted packets. The receivers report to the sender the set of packets that they receive, and a packet is retransmitted until all its receivers acknowledge the packet or the retry count in ER is reached. Multicast packets can be sent either using MAC-layer multicast or MAC-layer unicast with promiscuous monitoring. Our implementation uses the latter approach: packets are unicast to one of the receivers in the multicast group, and the other receivers in the group use promiscuous mode monitoring to extract their data. We choose this implementation because it unifies the unicast and multicast implementation and also allows potential use of exponential backoff and RTS/CTS. However this choice is not fundamental, and ER can also be built on top of MAC-layer multicast.

Note that ER can be applied to data with and without encryption. When

encryption (*e.g.*, WPA) is used, the sender xors encrypted packets and adds ER's header in plain text to specify which packets are combined, and the receiver uses the ER's header information to extract the new packet and then decrypt its content. Therefore the benefit of ER extends to corporate networks using WPA.

Several important design issues should be addressed in order to realize ER.

- First, how should the receivers give timely feedback to the sender without incurring much overhead?
- Second, when to retransmit data? This question involves two parts: (i) how should the sender determine that a packet requires a retransmission? and (ii) when the medium is available for the sender to transmit, which packet to send – a new packet or a lost packet? The answers to these questions affect the retransmission delay, the number of unnecessary retransmissions, and the potential coding benefit.
- Third, which set of packets should be coded together to minimize the number of retransmissions?

To address the above issues, ER consists of the following three components: (i) a light-weight receiver feedback scheme, (ii) a scheduling algorithm to determine which packets need retransmissions and when to transmit a new or lost packet, and (iii) a coding algorithm to optimize which set of packets to be coded together.

4.1.2 Receiver Feedback

Our receiver feedback scheme is built on COPE [16], where a node sends reception reports to inform which set of packets it has recently received. As in COPE, we use selective/cumulative ACKs to minimize the impact of ACK losses. Specifically, the report contains two fields: (i) the starting sequence number of the out of order ACKs (*start*) and (ii) a bit-map of out of order ACKs. All the packets up to *start* are

assumed to be received, and i-th position in the bitmap is 1 if and only if $start+i$-th packet is received. Our implementation differs from COPE in the following ways. First, to increase reliability of feedback, we send feedback using MAC-layer unicast, which will automatically retransmit lost feedback. Second, the length of bitmap increases from 1 byte in COPE to 8 bytes in ER so that it is more resilient to high ACK losses at a cost of a small increase in ACK overhead. We find this increase to be worthwhile since its benefits under ACK losses is significant. Third, COPE has separate ACKs and reception reports, where the former acknowledge the receipt of packets destinated to itself and the latter acknowledge the receipt of packets destinated to other nodes; furthermore these two types of reports are sent in different time scales. For the purpose of ER, the difference between ACK and reception reports is no longer necessary because in order to determine which packets to retransmit and how to code them, the sender needs both ACKs and reception reports. Therefore, our implementation unifies ACK and reception reports. Finally, when retransmitting a multicast packet, the sender specifies the nodes to which multicast packets are destinated; and only the nodes that are specified as destinations will send feedback. This way, we can reduce the receiver feedback especially when the multicast group is large but the packet is needed only at a small number of nodes.

4.1.3 Scheduling Algorithm

Next we need to decide (i) when a packet needs a retransmission and (ii) when the medium is available for the sender to transmit, which packet should the sender transmit – a new packet or a lost packet?

To address the first question, we use a standard approach to estimate retransmission timeout (RTO), similar to TCP. Specifically, for every packet that has not been retransmitted, a node measures the time difference between when the packet is transmitted and when the corresponding ACK in ER is received. Let T denote

the measured round-trip time of the current packet. Then the node updates its RTO based on smoothed RTT and RTT variance as shown in Algorithm 4. RTO is initialized based on the MAC data rate. Our evaluation uses $K = 4$, $\alpha = 1/8$, and $\beta = 1/4$ as in TCP [32].

if *(T is the first RTT measurement)* **then**
  $SRTT = T$;
  $RTTVAR = T/2$;
  $RTO = SRTT + K \times RTTVAR$;
else
  $RTTVAR = (1 - \beta) \times RTTVAR + \beta \times |SRTT - T|$;
  $SRTT = (1 - \alpha) \times SRTT + \alpha \times T$;
  $RTO = SRTT + K \times RTTVAR$;
end

Algorithm 4: RTO calculation in ER

To answer the second question, we make the following observation. If the sender retransmits a packet whenever the retransmission queue is non-empty, it achieves lowest retransmission delay. On the other hand, such aggressive retransmission would reduce or even eliminate coding opportunities. In the extreme, there is only one packet in the retransmission queue, and the packet has to be sent by itself and results in 0 coding gain. To strike a good balance between low delay and high coding gain, we use the following heuristic: retransmit the packet when the retransmission queue reaches a certain threshold or the packets in the retransmission queue timeout. The first condition increases the coding gain, and the second condition bounds retransmission delay. Our evaluation uses 25 as the threshold for the retransmission queue, and uses 250 msec as the timeout.

4.1.4 Coding Problem and Algorithms

Another important design issue is that given a set of packets to retransmit, how to code them together to minimize the number of transmissions. In this section, we

first formally study the coding problem and show that it is NP-hard to solve. Then we describe several practical coding algorithms.

Problem Specification

First, we introduce some notation. Let $N(i)$ denote the set of nodes that need packet i, and $H(i)$ denote the set of nodes that have packet i. A sender only codes packets together if the coded packet can be decoded right after its reception. This condition is commonly used in existing coding algorithms to simplify decoding algorithms [16, 22]. Under the above condition, two packets i and j can be coded if and only if $N(i) \subseteq H(j)$ and $N(j) \subseteq H(i)$, which we call *coding condition*. Essentially it means that i and j can be coded if and only if any nodes that need j have i, and any nodes that need i have j. To show the forward direction holds, i and j can be coded means that any node in $N(i)$ can decode the packet $P_i + P_j$ immediately after its reception; since nodes in $N(i)$ do not have P_i, the only way for decoding to succeed is that $N(i)$ have P_j so that they can xor $P_i + P_j$ with P_j. Similarly for N(j). To show the reverse direction holds, since $N(i) \subseteq H(j)$, every node in $N(i)$ has P_j. Then after receiving the coded packet $P_i + P_j$, it can extract P_i by xoring $P_i + P_j$ with P_j. Similarly for $N(j)$.

Based on the coding condition, we construct the following coding graph. Each packet is denoted by a node in the coding graph. For any two packets that can be coded together, we draw an edge between their corresponding nodes. It is not difficult to see that a transmission can be decoded if and only if the transmission only involves packets corresponding to a clique in the coding graph, where a clique is a set of nodes such that every pair of the nodes are connected. This is a simple generalization of the coding condition from 2 packets to N packets. Therefore the coding problem, *i.e.*, transmitting a given set of packets using a minimum number of transmissions, is essentially finding a minimum clique partition [15], which is stated

as follows. Given a graph $G = (V, E)$, where V are vertices and E are edges in G, partition V into a minimum number of disjoint subsets V_1, V_2, ..., V_k such that the subgraph induced by V_i is a complete graph. Next we show the coding problem is NP-hard. We prove this by reducing the minimum clique partition problem to the coding problem.

Given a graph $G = (V, E)$ for a minimum clique partition problem, we construct the coding problem that consists of three types of input: (i) a set of packets, (ii) for each packet i which clients require it – $N(i)$, and (iii) for each packet i which clients have it – $H(i)$. For each node i in G of the minimum partition problem, we create a corresponding packet P_i in the coding problem. Each packet P_i is required by a distinct receiver R_i, *i.e.*, $N(i) = \{R_i\}$. Based on the coding condition, we assign $H(i)$ as follows:

$$H(i) = \{R_j | \forall j \ s.t. (i, j) \in E\}$$

To show the above assignment of $H(i)$ satisfies the coding condition, we need to show that (i) any two adjacent nodes i and j satisfy $N(i) \subseteq H(j)$ and $N(j) \subseteq H(i)$, and (ii) any nodes that satisfy $N(i) \subseteq H(j)$ and $N(j) \subseteq H(i)$ are adjacent. The former holds because for $\forall\ (i, j) \in E$, $N(i) = \{R_i\} \subseteq \{R_k | \forall k \ s.t.\ (k, j) \in E\} = H(j)$, similarly for $N(j) \subseteq H(i)$. The latter holds because for $\forall N(i) = \{R_i\} \subseteq H(j) = \{R_k | \forall k \ s.t.\ (k, j) \in E\}$, we have $(i, j) \in E$. Therefore with the above construction, finding a minimum clique partition is essentially finding the optimal solution to the coding problem. Hence the coding problem is NP-hard.

Coding Algorithms

We describe three practical heuristics to solve the coding problem. Given the NP-hard nature of the problem, these heuristics are not guaranteed to yield optimal results. However as we will show in Section 4.2 and Section 4.3, they work well in practice. In addition, we also present an exhaustive search algorithm. While the

algorithm is guaranteed to give an optimal solution, it is computational very expensive and can only run on small-sized problems. So it just serves as an interesting baseline comparison.

COPE coding heuristic: The heuristic described in COPE [16] can be directly applied here. This heuristic is greedy in nature. Packets are sorted according to their arrival time with the first packet being the one that arrives the earliest. Every time the sender starts with the first packet in the queue, and iteratively combines with subsequent packets in the queue as long as the combined packet can be decoded (*i.e.*, all the receivers of the combined packet already have all but one packets in the combined packet).

Maximum utility: We find the order in which packets are examined for potential coding is important. The COPE heuristic codes the packet in the order of their arrival time. In the maximum utility heuristic, each packet is assigned a utility, defined as the number of receivers that need the packet. Intuitively, the packet that is required by more receivers is more important, and should be transmitted earlier. Therefore the maximum utility algorithm examines the packet in the non-increasing order of utility and using arrival time for tie-break. Specifically, the sender starts with the packet having the highest utility, and iteratively codes subsequent packets as long as the combined packet can be decoded. Note that maximum utility algorithm is useful for broadcast and multicast. In unicast, each packet is needed by one client, and has the same utility of 1. Therefore the maximum utility is equivalent to the COPE heuristic under unicast.

Maximum clique: As shown in Section 4.1.4, the coding problem can be cast as finding a minimum clique partition in a coding graph. Therefore another approach to solving the coding problem is to employ heuristics for minimum clique partition. One of the commonly used heuristics to minimum clique partition is to first find a maximum clique in the graph; then remove the clique from the graph and find

another maximum clique, and iterate. Note that the maximum clique problem itself is NP-hard, but has a simple heuristic, which starts with the node of highest degree and iteratively adds additional nodes to the clique as long as they maintain the clique property – all nodes in a clique are connected.

Exhaustive search via dynamic programming: We develop an exhaustive search algorithm to minimize the number of retransmissions. This algorithm is computationally very expensive and is not for practical use. Instead it serves as an interesting baseline comparison to quantify the effectiveness of the other coding algorithms.

First, we introduce a few notations. Let M denote the number of packets required for retransmissions. Let S denote a state, indicating for each packet i which nodes need it and which nodes have it, namely $(N(i), H(i))$. The exhaustive search algorithm first generates all possible packet combinations. There are 2^M packet combinations, since each packet either belongs to a packet combination or not. The goal is to find a smallest number of packet combinations that converts the current state to the state where every node gets the packets it needs (*i.e.*, $N(i) = \{\}$ for all i). To identify a minimum set of packet combinations, we build the following coding tree. The root of the tree is the current state. Starting from the root, we try every packet combination. A packet combination is considered useful if it allows at least one receiver to get a packet it needs if there is no loss. For each useful packet combination, we add a child node to the root; we also label the edge of to the child with the packet combination and label the node with the state after all nodes receive the packet combination. Packets combinations that are not useful are simply ignored. After going through all the packet combinations, we then repeat the process – for each of the child nodes we identify the useful packet combinations and add them to the next level of the tree. The process continues until we reach a state where every node gets the packet that it needs. The depth of the tree at that

node is the minimum number of transmissions required (assuming the depth of a root is 0). Moreover, the packet combinations marked along the path from the root to the current node are the set of packets to transmit that minimizes the number of transmissions.

4.2 Simulation

To evaluate our proposal of ER and to check the feasibility of the idea in the testbed, we conduct a number of simulation experiments. In this chapter, we first describe our simulation methodology and then present performance results.

4.2.1 Simulation Methodology

To evaluate the performance of various retransmission schemes presented in Section 4.1.4, we simulate the behavior of the algorithms under both unicast and multicast using a variety of network topologies. In our simulation, we generate network topologies consisting of a sender and a varying number of receivers with varying loss rates. We consider both homogeneous and heterogeneous loss rate assignments between a sender and each of its receivers. In homogeneous cases, the loss rates between the sender and all its receivers are roughly the same, and are varied from 10% to 90%. In hetergeneous loss cases, we assign the loss rates between the sender and its receivers randomly chosen between 0 and an upperbound, where the upperbound is varied from 10% to 90%. Therefore some receivers may see loss rate as low as 0, while other receivers may see loss rates close to the upperbound. For both homogeneous and heterogeneous cases, we generate losses using Bernoulli and Gilbert models. In the Bernoulli model, each packet is dropped with a fixed probability determined by the loss rate of the link. In the Gilbert model, the link moves between a good state and a bad state, where no packets are dropped at the good state and all packets are dropped at the bad state. Following [27, 30], we use 35%

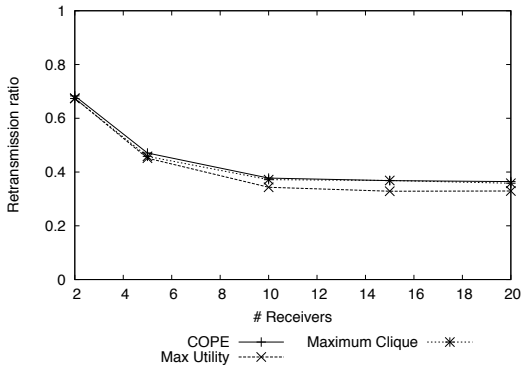
as the probability of remaining in the bad state. The other state-transition probabilities are determined to match the average loss rate with the loss rate assigned to the link. Unless otherwise specified, the batch size used is 20 (*i.e.*, a sender sends 20 packets at a time before starting retransmissions), and the results are averaged over 100 batches. In addition, we also evaluate the impact of batch sizes.

We use *retransmission ratio* to quantify the performance of different retransmission schemes. The retransmission ratio is defined as the total number of retransmissions using the current scheme divided by the total number of retransmissions using a basic retransmission scheme, which retransmits each lost packet by itself without coding and corresponds to the retransmission scheme in IEEE 802.11. A lower retransmission ratio indicates fewer retransmissions, and hence is preferred.

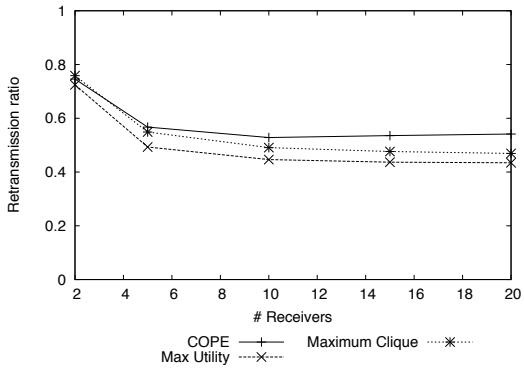
To ensure the same level of reliability, all retransmission schemes use unlimited number of retransmissions so that they all achieve 100% delivery rate. Our results of bounded retransmissions are qualitatively similar. The only difference is that under extremely high loss rates (*e.g.*, 90%), the retransmission ratio under bounded retry count approaches 1 because the number of retransmissions under both the basic and coding algorithms are determined by the retry count. In such cases, the coding based retransmission schemes deliver more packets successfully. Therefore in the interest of brevity, we will focus on the performance of unbounded retry count in this section.

4.2.2 Simulation Results

First we present the simulation results of multicast by varying the number of receivers, batch size, and loss rates. Then we present the unicast performance results.

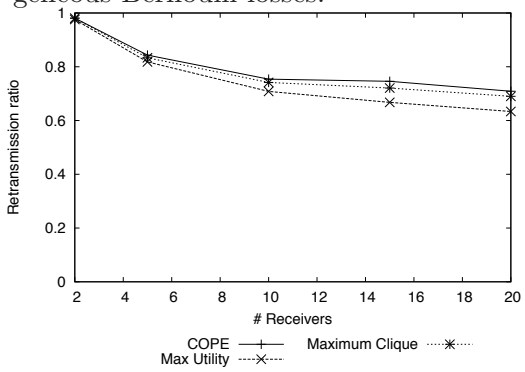


(a) 20% loss rate to each receiver

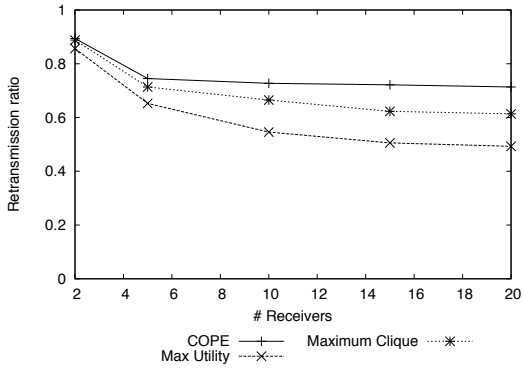


(b) 50% loss rate to each receiver

Figure 4.1: Multicast comparison under a varying number of receivers with homogeneous Bernoulli losses.



(a) 20% loss rate to each receiver



(b) 50% loss rate to each receiver

Figure 4.2: Multicast comparison under a varying number of receivers with homogeneous Gilbert losses.

Multicast Results under Homogeneous loss rates

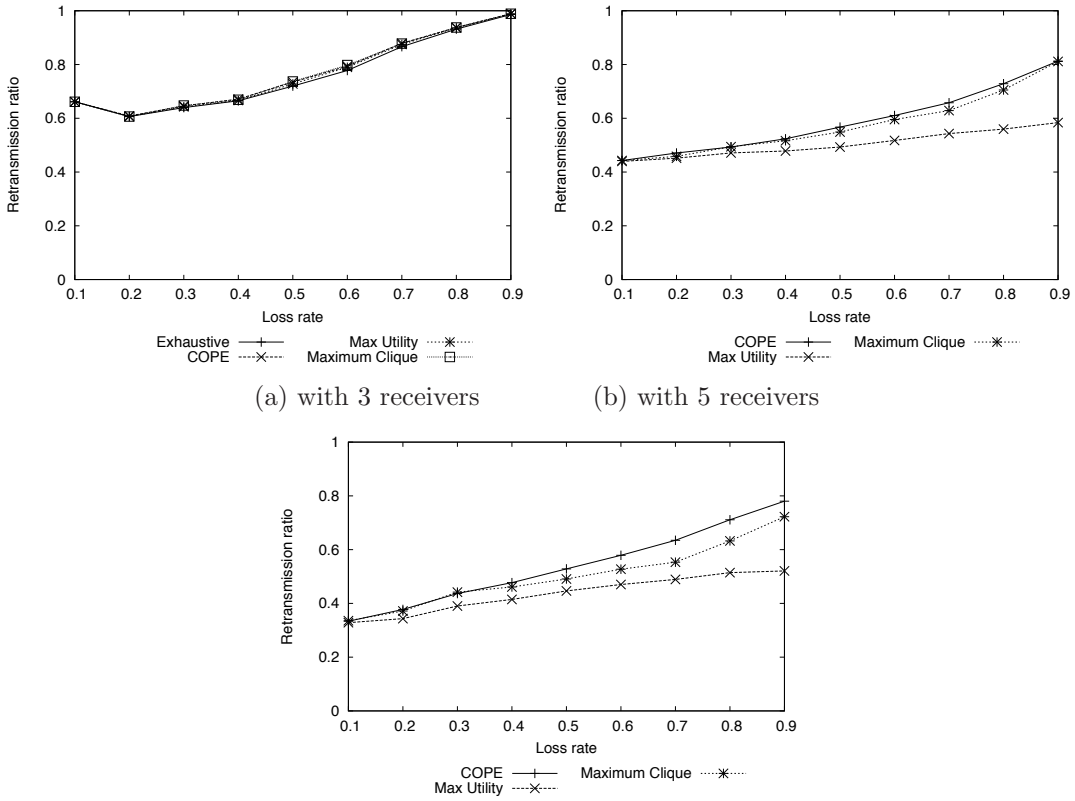
Varying the number of receivers: Figure 4.1 and Figure 4.2 show retransmission ratios with a varying number of clients under Bernoulli and Gilbert loss models, respectively. We make the following observations.

First, in all cases the coding-based retransmissions yield retransmission ratios below 1. This indicates that the coding-based retransmissions is more efficient than the basic retransmission. The lowest ratios achieved are around 0.4, reducing the total number of retransmissions by 60%.

Second, the retransmission ratios decrease with the number of receivers, which suggests that the benefit of coding-based retransmissions increases with the number of receivers. This occurs because a larger number of receivers makes it easier to find receivers that lose different packets and create coding opportunities.

Third, comparing the three different coding algorithms, we observe they perform similarly under the Bernoulli loss model, and their difference enlarges under the Gilbert loss model, with maximum utility out-performing maximum clique, which out-performs COPE. The good performance of the maximum utility algorithm is likely because packets that are lost at many nodes are harder to find other packets to code with (in the extreme, the packets lost at all nodes have to be retransmitted by itself); sending them earlier makes it easier to find packets to code as there are more candidates to choose from. In addition, sending them earlier helps to create coding opportunities for future retransmissions as coding opportunities arise after enough packets are received. The larger benefit under the Gilbert loss model is likely because utility distribution is more skewed under the Gilbert loss model and the maximum utility algorithm makes more difference. In the interest of brevity, below we present the results under the Bernoulli loss model, and comment on the the Gilbert results whenever the difference is significant.

Varying loss rates: Next we evaluate the performance by varying loss rates. Figure 4.3 summarizes the results under 3, 5, and 10 receivers. For 3 receivers, we also plot the results of the exhaustive search; the results of the exhaustive search under a higher number of receivers are not available due to its high computational complexity. Under 3 receivers, the practical coding schemes performs almost the same as the exhaustive search, with all curves overlapping with each other. This further confirms the effectiveness of the coding heuristics. Under 5 and 10 receivers, the retransmission ratios of three coding heuristics are between 0.35 and 0.8, cutting the number of retransmissions by 20% to 65%. In all cases, the ratios are lowest



(a) with 3 receivers

(b) with 5 receivers

(c) with 10 receivers

Figure 4.3: Multicast comparison under a varying loss rate with homogeneous Bernoulli losses.

under low packet loss rates because the packets lost at different receivers are more likely to be different under low loss rates and create more coding opportunities.

Varying batch sizes: We further evaluate the impact of batch sizes. As shown in Figure 4.4, with an increasing batch size, the retransmission ratio decreases and coding benefit increases. When the batch size is 5 packets, the coding-based retransmission schemes already achieve the ratio below 0.6. When the batch size increases to 50, the ratios are as low as 0.3. This shows that there is a tradeoff between packet delay and bandwidth savings. The good news is that only a small batch (or delay) is needed to achieve significant bandwidth savings.

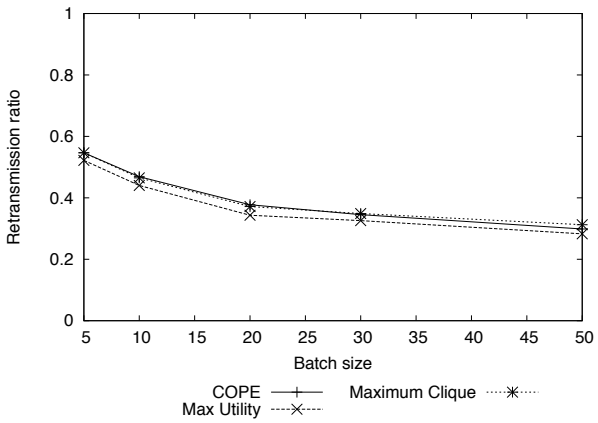


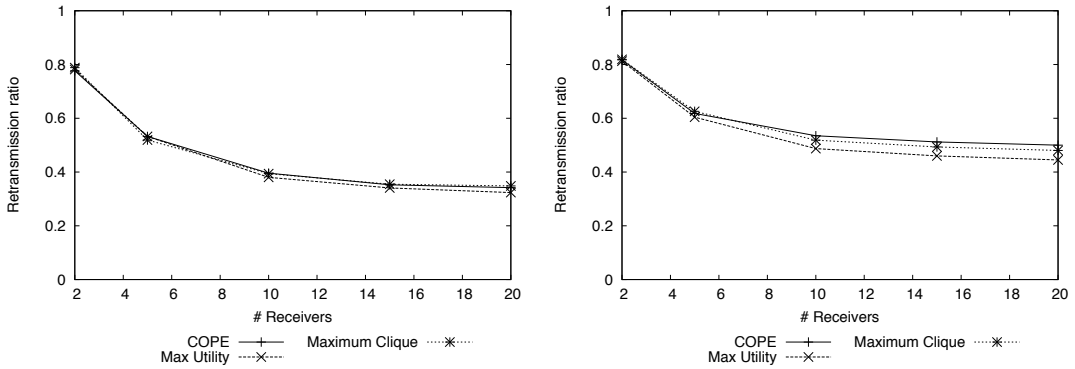
Figure 4.4: Multicast comparison under a varying batch size with 10 receivers and 20% homogeneous Bernoulli loss rates.

Multicast Results under Heterogeneous Losses

So far we consider the similar loss rates between the sender and all its receivers. In the following evaluation, we consider heterogeneous loss rates. We assign the average loss rate to each client randomly, chosen between 0 and the loss bound. In this case, the difference between loss rates across different clients is up to the loss bound.

Varying the number of receivers: Figure 4.5(a) and (b) show the results under 20% and 50% loss bounds, respectively, where the number of receivers varies from 2 to 20. As we can see, the coding-based retransmission schemes significantly out-perform the basic retransmission, with retransmission ratios ranging between 0.4 and 0.8. However, the difference across different coding algorithms is small under Bernoulli losses. The difference under the Gilbert loss model (not shown) is larger, with the same ranking as before and the maximum utility out-performing COPE by up to 25%.

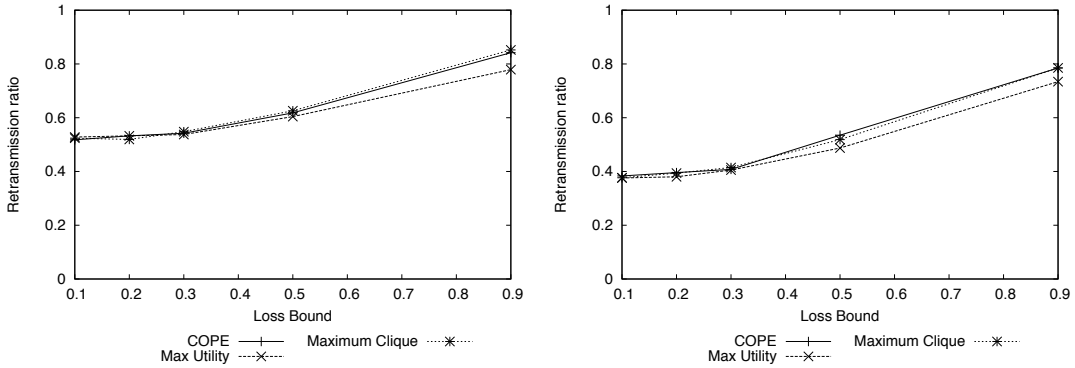
Varying loss bounds: We further evaluate the heterogeneous cases by varying the loss bound. Figure 4.6 summarizes the results under 5 and 10 receivers. As the loss bound increases, loss heterogeneity increases, which increases the retransmission



(a) 20% loss bound (b) 50% loss bound

Figure 4.5: Multicast comparison under a varying number of receivers with heterogeneous Bernoulli losses.

ratio and decreases the coding benefit. This is expected because under higher loss heterogeneity most of the retransmissions are sent to one or few receivers and such imbalanced retransmission load makes it hard to find coding opportunities.



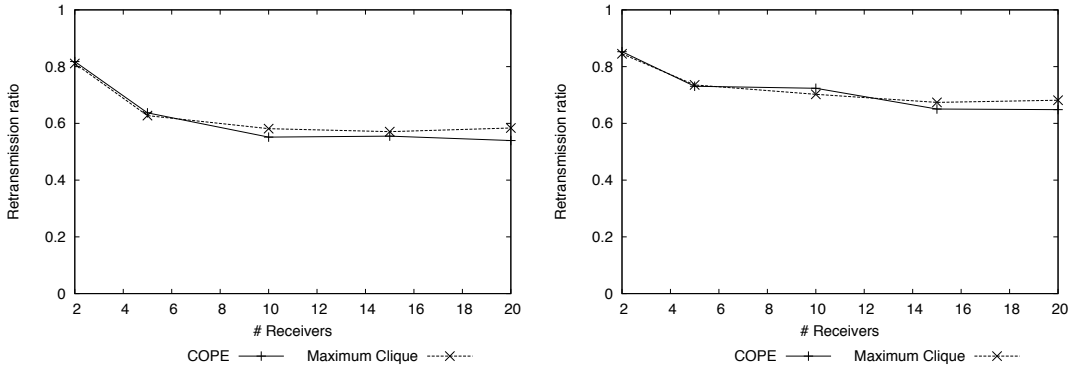
(a) with 5 receivers (b) with 10 receivers

Figure 4.6: Multicast comparison under a varying loss bound with heterogeneous Bernoulli losses.

Unicast Results under Homogeneous Losses

In the following two sections, we evaluate the performance of unicast retransmission schemes under homogeneous and heterogeneous losses. Since the maximum utility is equivalent to COPE under unicast, we only compare COPE and maximum clique

with the basic retransmission.

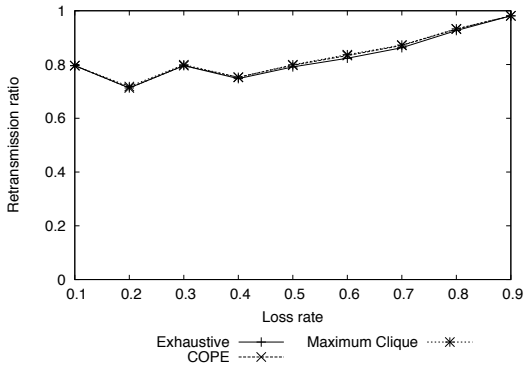


(a) 20% loss rate to each receiver (b) 50% loss rate to each receiver

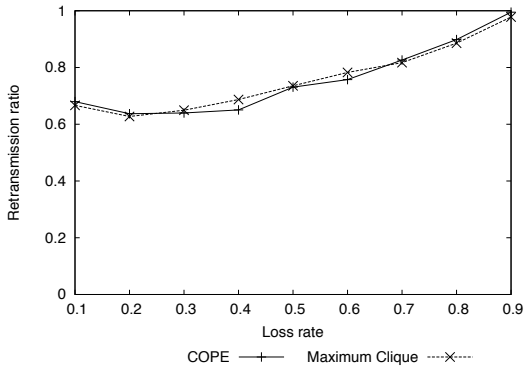
Figure 4.7: Unicast comparison under a varying number of receivers with homogeneous Bernoulli losses.

Varying the number of receivers: Figure 4.7(a) and (b) show the results under a varying number receivers when the loss rate to each receiver is 20% and 50%, respectively. In both cases the coding-based schemes achieve retransmission ratios between 0.6 and 0.8. As in multicast cases, with an increasing number of receivers, the retransmission ratio decreases and the coding benefit increases.

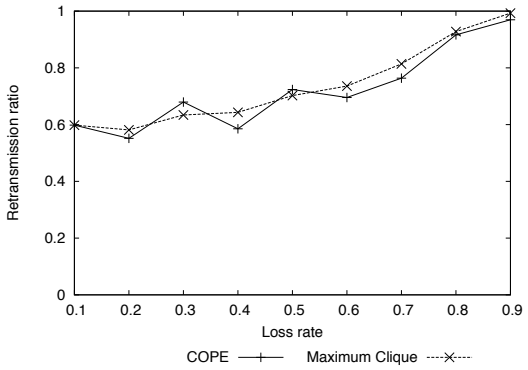
Varying loss rates: Figure 4.8 shows the results under a varying loss rate. Under 3 receivers, the coding heuristics are compared against the exhaustive search, and they all perform similarly, indicating the effectiveness of the heuristics. Under all numbers of receivers, the retransmission ratio tends to increase with the loss rate. This is consistent with the multicast results and due to the same reason. Compared with the multicast performance in Figure 4.3, the coding benefit of unicast retransmissions under the corresponding loss rates are smaller. This is because coding gain in multicast cases arises whenever receivers obtain different sets of packets, whereas coding in unicast not only requires the above condition but also requires that packets that the receivers lose are destined to them (*i.e.*, receivers do not care if they lose packets destined to other nodes). The additional coding constraint reduces the



(a) with 3 receivers



(b) with 5 receivers



(c) with 10 receivers

Figure 4.8: Unicast comparison under a varying loss rate with homogeneous Bernoulli losses.

coding opportunities.

Varying batch sizes: Figure 4.9 shows the result of varying batch size. As the batch size increases, the retransmission ratio decreases and coding benefit increases. This is consistent with multicast results, since a larger batch size has more packet combinations to choose from and increases the coding benefit.

Unicast Results under Heterogeneous Losses

We also evaluate the retransmission schemes under unicast traffic using heterogeneous losses.

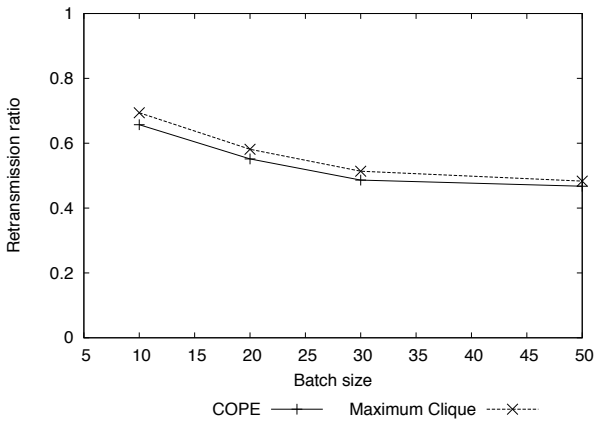
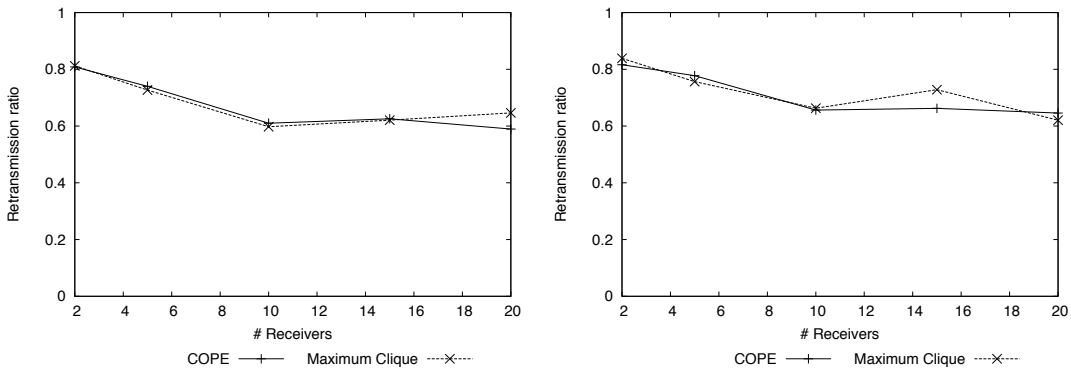


Figure 4.9: Unicast comparison under a varying batch size with 10 receivers and 20% homogeneous Bernoulli loss rates.

Varying the number of receivers: First we vary the number of receivers with the loss bound of either 20% or 50%. As shown in Figure 4.10, in both cases,



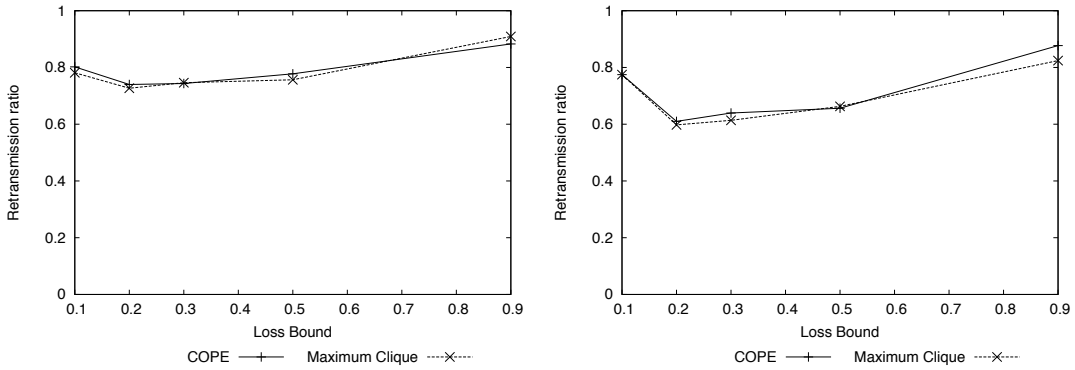
(a) Loss bound = 20% (b) Loss bound = 50%

Figure 4.10: Unicast comparison under a varying number of receivers with heterogeneous Bernoulli losses.

the retransmission ratio is between 0.6 and 0.8. As in multicast cases, the lower retransmission ratios (or higher coding benefit) is achieved under a larger number of receivers due to more coding opportunities.

Varying loss bounds: We further evaluate the performance by varying the loss bounds while setting the number of receivers to 5 or 10. Figure 4.11 shows that

the retransmission ratio initially decreases and then increases with the loss bound. The later increase is due to the same reason as the multicast cases, where under



(a) with 5 receivers (b) with 10 receivers

Figure 4.11: Unicast comparison under a varying loss bound with heterogeneous Bernoulli losses.

higher loss bounds most retransmissions are towards one or few receivers and hard to code them with other packets. The initial decrease is likely because coding unicast retransmissions requires an additional constraint that different nodes miss their own packets, and increasing the loss bound initially helps to increase the likelihood of satisfying this constraint.

4.3 Testbed Experiments

In addition to high-level simulation, we also implement the various retransmission mechanisms in a wireless testbed. Testbed experiments are valuable because they allow us to evaluate the ER protocol under realistic scenarios. In this section, we first describe our testbed implementation and evaluation methodology, and then present the performance results.

4.3.1 Implementation

Our implementation of ER is built on the COPE source code [9], which performs network coding at intermediate nodes in multihop wireless networks. We make the following modifications to support ER. We modify the receiver feedback scheme in COPE as described in Section 4.1.2. We implement the scheduling algorithm described in Section 4.1.3 to determine whether a packet needs a retransmission and when a retransmission should be sent. In addition, we disable MAC-layer retransmissions used in COPE. Instead, we implement three retransmission mechanisms above the MAC-layer: (i) the basic retransmission, which keeps sending a lost packet until all the intended receivers acknowledge it or the maximum retry count is reached, (ii) the coding-based retransmission using the COPE heuristic to determine which set of packets to combine, and (iii) the coding-based retransmission using the maximum utility heuristic. Note that (iii) is identical to (ii) under unicast, so it is skipped for unicast evaluation. The maximum retry count is 7 in both basic retransmission and COPE heuristic to achieve similar level of reliability, and 7 is commonly used retry count in IEEE 802.11.

4.3.2 Experiment Methodology

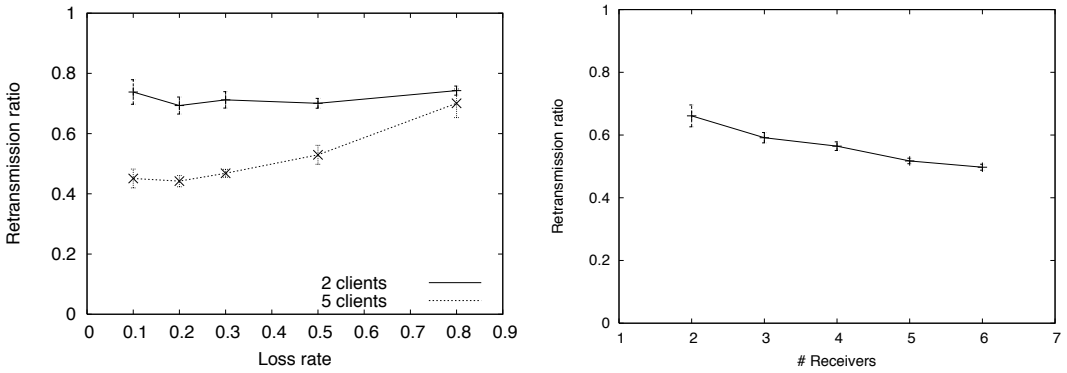
We set up a wireless testbed that consists of 7 DELL Dimension 1100 PCs. The testbed spans one floor of an office building. Each machine has a 2.66 GHz Intel Celeron D Processor, and runs Fedora Core 4 Linux. Each is equipped with 802.11 a/b/g NetGear WAG511 using MadWiFi. RTS/CTS is disabled as in the default setting. Our experiments use 802.11b. To avoid interference with resident wireless networks, we run our experiments during nights and weekends. We use 1 AP as a sender, and use up to 6 clients as receivers. The loss rates between the AP and clients are generated in a controlled manner to evaluate the performance under various loss scenarios. We impose a specific loss rate on each wireless link by artificially

dropping traffic at the receivers, and the dropped packets are not acknowledged by the receiver's feedback in ER. Unless otherwise specified, the packet are dropped using Bernoulli loss model and all clients experience similar loss rates. For each scenario (ie, a given number of receivers and loss rate), we run seven times, where each time we obtain the retransmission ratio (defined in Section 4.2.1) by letting the AP send 1000 packets. We then plot retransmission ratios using errorbars, where the center of an errorbar corresponds to the mean and the length of the errorbar is twice the standard deviation over seven runs.

4.3.3 Experiment Results

Multicast Evaluation

We first evaluate the multicast performance of ER by varying the loss rates. Figure 4.12 (a) summarize the results under 2 and 5 clients. The retransmission ratio is between 0.6 – 0.8 for 2 receivers, and between 0.4 – 0.8 for 5 receivers. These results are consistent with the simulation results, indicating that the benefit of ER extends to real wireless networks.



(a) Under a varying loss rate (b) Under a varying number of clients

Figure 4.12: Multicast experiment results.

We further evaluate the performance using a varying number of receivers while keeping the loss rate to each client around 50%. As shown in Figure 4.12 (b),

the retransmission ratio decreases from 0.7 to 0.5 as the number of receivers varies from 2 to 6. This is consistent to the simulation result, and shows the benefit of ER increases with more receivers.

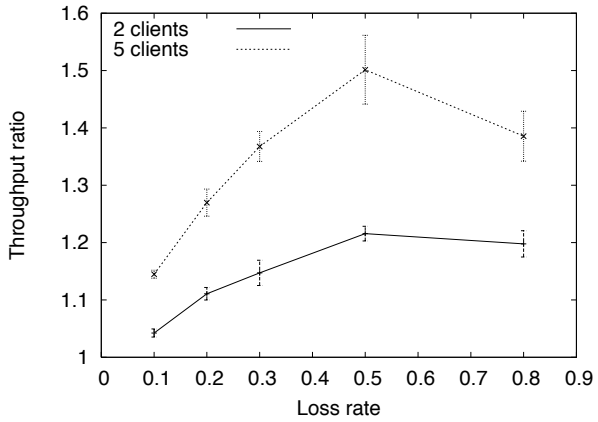


Figure 4.13: Compare throughput against the basic scheme for multicast under a varying loss rate.

Finally we evaluate the performance in terms of throughput when varying the loss rate to each client. We measure the total system throughput for a 30 second transfer for the COPE coding and basic retransmission scheme. Then, we calculate the ratio of the COPE coding scheme against the basic retransmission scheme. Higher values indicate a larger throughput improvement from coding. We present the results in Figure 4.13.

The improvement under COPE can be quite significant, up to 1.21 times the throughput for 2 clients and up to 1.50 times the throughput for 5 clients.

Unicast Evaluation

Next we evaluate the unicast performance of ER. Figure 4.14 shows that the retransmission ratios are between 0.6 and 0.8 as the loss rate varies from 0.1 to 0.8.

To further quantify how the loss patterns affect the coding benefit, we impose a different loss characteristic – only packets destined to the receivers are dropped

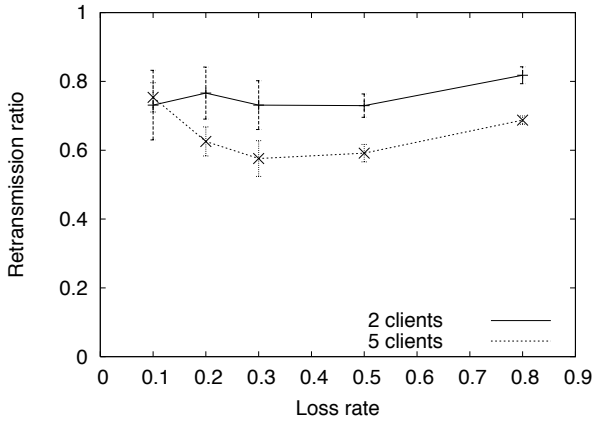


Figure 4.14: Compare retransmission mechanisms for unicast under a varying loss rate.

according to the specified loss rate, while all the other packets incur no artificial losses. In this case, the packets lost at different receivers are guaranteed to be different, and this increases the coding opportunities. Therefore we observe a lower retransmission ratio, between 0.2 and 0.8, as shown in Figure 4.15.

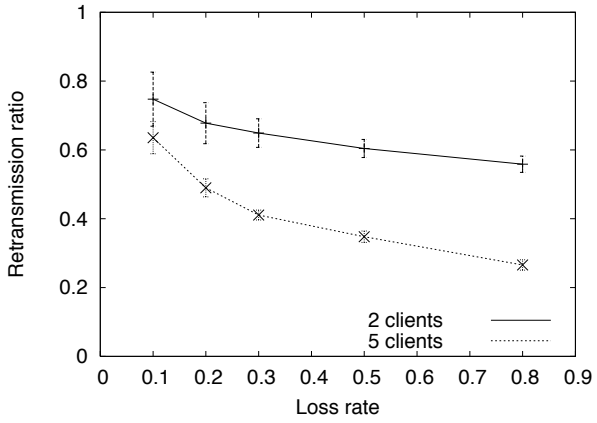


Figure 4.15: Compare retransmission mechanisms for unicast under a varying loss rate, where only packets destined to the receivers are dropped.

We also evaluate the performance by varying the number of clients and keeping the loss rate to each client to be around 0.5. As shown in Figure 4.16, the retransmission ratio is between 0.6 and 0.7. Moreover, the ratio tends to decrease with the number of receivers, as we would expect.

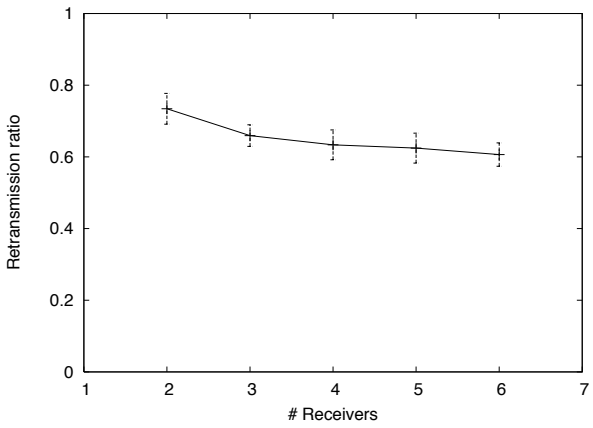


Figure 4.16: Unicast experiment results under a varying number of clients.

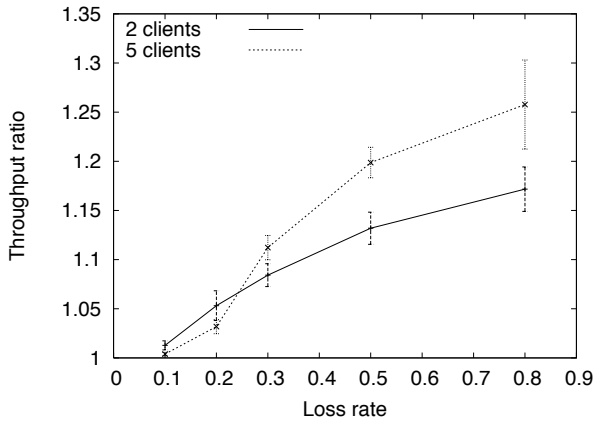


Figure 4.17: Compare throughput against the basic scheme for unicast under a varying loss rate.

Finally we evaluate the performance in terms of throughput ratio when varying the loss rate to each client. We present the results in Figure 4.17. The unicast improvement under COPE is generally less than in the multicast case, as expected. Regardless the throughput improves up to 1.17 times for 2 clients and 1.25 times for 5 clients.

Chapter 5

Conclusion

In this work, we develop Fast Resilient Jumbo frames (FRJ) to enhance the efficiency and reliability of wireless performance, and Efficient Retransmission scheme (ER) to efficiently support retransmissions in wireless networks for both unicast and broadcast/multicast traffic.

ER reduces the number of required transmissions to recover packet losses by coding packets lost at different receivers. Reducing the number of retransmissions vastly improves performance. This attains significance given the observation that common wireless deployments see loss rates as high as 40%. Using simulation and experiments, we show that ER is effective over a wide range of scenarios. Our results indicate that the benefits of using ER is higher under multicast/broadcast traffic rather than with unicast. We also observe that ER achieves highest performance under moderate link losses, and that the benefits increase with number of receivers.

FRJ uses jumbo frames with partial packet recovery to boost network throughput under good channel condition and efficiently recover packet losses under bad channel condition. It further jointly optimizes rate adaptation and partial packet recovery to achieve high transmission rates and protect against wireless losses. Our evaluation based on a real implementation and testbed experiments show that FRJ

consistently out-performs regular frames under different channel and traffic conditions. Also, our evaluation results indicate that combining orthogonal techniques provides substantial benefits over the application of individual techniques. Further, this combination also help mitigate the negative effects of the techniques involved.

As part of our future work, we aim to extend FRJ and ER to multihop wireless networks. We also wish to evaluate TCP performance in FRJ and ER.

Bibliography

[1] Atheros Super G. `http://www.atheros.com/pt/whitepapers/atheros_superg_whitepaper.pdf`.

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

[3] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, Jul. 2000.

[4] J. Ala-Laurila, H. Haverinen, J. Mikkonen, and J. Rinnemaa. Wireless LAN architecture for mobile operators. In *Wireless local area networks: the new wireless revolution*, pages 159–176. John Wiley & Sons, Inc., 2002.

[5] P. Bahl, A. Balachandran, A. Miu, W. Russel, G. M. Voelker, and Y.-M. Wang. PAWNs: Satisfying the need for ubiquitous secure connectivity and location services. In *IEEE Wireless Communications Magazine*, volume 9(1), Feb. 2002.

[6] J. Bicket. Bit-rate selection in wireless networks. In *Masters thesis, Massachusetts Institute of Technology*, Feb. 2005.

[7] Y. Birk and D. Crupnicoff. A multicast transmission schedule for scalable multirate distribution of bulk data using nonscalable erasure correcting codes. In *Proc. of IEEE INFOCOM*, Apr. 2003.

[8] Click. http://pdos.csail.mit.edu/click/.

[9] COPE source code. http://piper.csail.mit.edu/dokuwiki/doku.php?id=cope.

[10] P. Ding, J. Holliday, and A. Celik. MAC layer multicast protocol to increase reliability in WLANs. `http://citeseer.ist.psu.edu/717518.html`.

[11] R. Ganti, P. Jayachandran, H. Luo, and T. Abdelzaher. Datalink streaming in wireless sensor networks. In *Proc. of ACM SenSys*, Nov. 2006.

[12] G. Holland, N. Vaidya, and P. Bahl. A rate-adaptive MAC protocol for multihop wireless networks. In *Proc. of ACM MOBICOM*, Jul. 2001.

[13] L. Huang, A. Arora, and T. Lai. Reliable MAC layer multicast in IEEE 802.11 wireless networks. In *Proc. of ICPP*, Aug. 2002.

[14] K. Jamieson and H. Balakrishnan. PPR: partial packet recovery for wireless networks. In *Proc. of ACM SIGCOMM*, Aug. 2007.

[15] V. Kann. Minimum clique partition. `http://www.csc.kth.se/~viggo/wwwcompendium/node25.html`.

[16] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft. XORs in the air: Practical wireless network coding. In *Proc. of ACM SIGCOMM*, Sept. 2006.

[17] J. Kim, S. Kim, S. Choi, and D. Qiao. CARA: collision-aware rate adaptation for IEEE 802.11 wlans. In *Proc. of IEEE INFOCOM*, Apr. 2006.

[18] R. Koetter and M. Medard. An algebraic approach to network coding. *IEEE/ACM Transactions on Networking*, pages 782–795, Oct. 2003.

[19] J. Kuri and S. Kasera. Reliable multicast in multi-access wireless LANs. *Wireless Networks*, pages 359–369, Apr. 2001.

[20] M. Lacage, M. H. Manshaei, and T. Turletti. IEEE 802.11 rate adaptation: A practical approach. In *Proc. of ACM MSWiM*, Oct. 2004.

[21] J. Lacan and T. Perennou. Evaluation of error control mechanisms for 802.11 b multicast transmissions. In *Proc. of WiNMee*, Apr. 2006.

[22] L. Li, R. Ramjee, M. Buddhikot, and S. Miller. Network-coding based broadcast in mobile ad-hoc networks. In *Proc. of IEEE INFOCOM*, Apr. 2007.

[23] Z. Li, B. Li, and L. C. Lau. On achieving maximum multicast throughput in undirected networks. *IEEE Transactions on Information Theory & IEEE/ACM Transactions on Networking (Special Issue on Networking and Information)*, Jun. 2006.

[24] Madwifi. `http://madwifi.org`.

[25] P. McKinley, C. Tang, and A. Mani. A study of adaptive forward error correction for wireless collaborative computing. *IEEE Transactions on Parallel and Distributed Systems*, pages 936–947, Sept. 2002.

[26] A. Miu, H. Balakrishnan, and C. E. Koksal. Improving loss resilience with multi-radio diversity in wireless networks. In *Proc. of ACM MOBICOM*, Aug. - Sept. 2005.

[27] E. M. Nahum, C. C. Rosu, S. Seshan, and J. Almeida. The effects of wide-area conditions on WWW server performance. In *Proc. of ACM SIGMETRICS*, Jun. 2001.

[28] L. M. S. C. of the IEEE Computer Society. Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. *IEEE Standard 802.11*, 1999.

[29] ONOE rate control. `http://madwifi.org/browser/trunk/athrate/onoe`.

[30] V. N. Padmanbhan, L. Qiu, and H. Wang. Server-based inference of Internet performance. In *In Proc. of IEEE INFOCOM*, Mar. 2003.

[31] R. Patra, S. Nedevschi, S. Surana, A. Sheth, L. Subramanian, and E. Brewer. WiLDNet: design and implementation of high performance WiFi based long distance networks. In *Proc. of NSDI*, Apr. 2007.

[32] V. Paxson and M. Allman. Computing TCP's retransmission timer. *IETF Internet DRAFT*, 2000. http://www3.ietf.org/proceedings/00jul/I-D/paxson-tcp-rto-01.txt.

[33] D. Qiao, S. Choi, and K. Shin. Goodput analysis and link adaptation for IEEE 802.11a wireless LANs. *IEEE TMC*, Oct. 2002.

[34] L. Rizzo. Effective erasure codes for reliable computer communication protocols. *ACM SIGCOMM Computer Communication Review*, pages 24–36, Apr. 1997.

[35] L. Rizzo and L. Vicisano. A reliable multicast data distribution protocol based on software FEC techniques. In *Proc. of HPCS*, Jun. 1997.

[36] L. Rizzo and L. Vicisano. RMDP: an FEC-based reliable multicast protocol for wireless environments. *ACM SIGMOBILE Mobile Computing and Communications Review*, pages 23–31, Apr. 1998.

[37] M. Rodrig, C. Reis, R. Mahajan, D. Wetherall, and J. Zahorjan. Measurement-based characterization of 802.11 in a hotspot setting. In *Proc. of ACM SIGCOMM Workshop on Experimental Approaches to Wireless Network Design and Analysis (E-WIND)*, Aug. 2005.

[38] B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knightly. Opportunistic media access for multirate ad hoc networks. In *Proc. of ACM MOBICOM*, Sept. 2002.

[39] L. Subramanian, I. Stoica, H. Balakrishnan, and R. Katz. OverQos: an overlay based architecture for enhancing internet QoS. In *Proc. of NSDI*, Mar. 2004.

[40] K. Tang and M. Gerla. Random access MAC for efficient broadcast support in ad hoc networks. In *Proc. of Wireless Communications and Networking Conference (WCNC)*, Sept. 2000.

[41] K. Tang and M. Gerla. MAC reliable broadcast in ad hoc networks. In *Proc. of IEEE MILCOM*, Oct. 2001.

[42] S. H. Wong, H. Yang, S. Lu, and V. Bharghavan. Robust rate adaptation in 802.11 wireless networks. In *Proc. of ACM MOBICOM*, Sept. 2006.

[43] G. Woo, P. Kheradpour, and D. Katabi. Beyond the bits: Cooperative packet recovery using PHY information. In *Proc. of ACM MOBICOM*, Sept. 2007.

[44] S. Yuk and D. Cho. Parity-based reliable multicast method for wireless LAN environments. In *Proc. of VTC*, May 1999.

[45] Z. Zhou, P. K. McKinley, and S. M. Sadjadi. On quality-of-service and energy consumption tradeoffs in FEC-encoded wireless audio streaming. In *Proc. of IWQoS*, Jun. 2004.