ER: Efficient Retransmission Scheme for Wireless LANs [*]

Eric Rozner, Anand Padmanabha Iyer, Yogita Mehta, Lili Qiu, and Mansoor Jafry
{erozner,aiyer,yamehta,lili,mansoor}@cs.utexas.edu
University of Texas at Austin

ABSTRACT

Wireless LANs (WLANs) have been deployed at a remarkable rate at university campuses, office buildings, airports, hotels, and malls. Providing efficient and reliable wireless communications is challenging due to inherent lossy wireless medium and imperfect packet scheduling that results in packet collisions. In this paper, we develop an efficient retransmission scheme (*ER*) for wireless LANs. 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.

1. INTRODUCTION

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]. Unicast traffic contributes to the majority of wireless traffic today, but with increasing popularity of multicast applications, such as streaming video and file sharing, wireless multicast will become increasingly important. Providing efficient and reliable communication, however, is challenging due to inherent lossy wireless medium and imperfect packet scheduling that results in packet collisions. The average loss rate in some deployments is as high as 20-40% [2, 25].

This paper presents *ER*, an efficient retransmission mechanism to support reliable unicast, broadcast, and multicast in WLANs. The design of ER can also easily be extended to multihop 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 [1], 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, the 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 [11]. ER complements the previous work in several important ways. First, the existing network coding approaches target multihop 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 re-

[*]This research is sponsored in part by National Science Foundation grants CNS-0546755 and CNS-0627020.

transmissions required to recover the losses. Therefore ER can be applied to wireless LANs and multihop 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.

In this paper, we develop and implement ER to provide reliable unicast, broadcast, and multicast in WLANs. An important component in the design of ER, as well as other network coding schemes, such as COPE [11] and broadcast coding [15], is which set of packets should be coded together to minimize the number of required transmissions. We formally study the problem, and show it is NP-hard. We describe several practical heuristics and use empirical evaluation to study their effectiveness. Our extensive simulation and testbed experiments show that ER significantly reduces the number of retransmissions compared to the existing retransmission scheme, which retransmits the lost packets by themselves.

The rest of the paper is organized as follows. In Section 2, we review the existing work on providing reliable communication in WLANs. In Section 3, we present our approach. We describe our simulation methodology and results in Section 4, and present the implementation and experimental results in Section 5. We conclude in Section 6.

2. RELATED WORK

The existing work of supporting reliable communications uses one or a combination of the following techniques: (i) retransmissions, (ii) forward error correction (FEC), (iii) network coding, and (iv) channel reservation for reducing collision losses.

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) [1, 13]. IEEE 802.11 [1] 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 failed transmission 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.

FEC: FEC has been used to provide reliable unicast and multicast communication in both wireless networks (*e.g.*, [24, 29, 17, 20, 14, 30]) and wireline networks (*e.g.*, [6, 22, 23, 26]). For example, in [17], McKinley et al. dynamically adjusts the level of FEC redundancy based on observed channel quality. [14] points out that many existing FEC-based works 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, [30] analyzes the tradeoff between improving multicast throughput and minimizing power consumption when using FEC techniques.

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.*, [12, 16, 11, 15]). In particular, COPE [11] develops a practical network coding scheme for unicast in multi-hop wireless networks and [15] 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 [15] does not consider loss recovery. Built on existing network coding work, ER applies the coding concept to provide efficient MAC-layer retransmissions and is complementary to the existing work.

Channel reservation: One of the major sources of packet losses in wireless networks comes from packet collisions. 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 [1] 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) [27], Broadcast Medium Window (BMW) [28], Batch Mode Multicast MAC protocol (BMMM) [9], and Leader based Priority Ring Multicast Protocol (LPRMP) [8]. 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).

3. OUR 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.

3.1 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 into 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 together 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 both encrypted and unencrypted data packets. 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 wireless 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? (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.

3.2 Receiver Feedback

Our receiver feedback scheme is built on COPE [11], 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 the $start+i$-th packet is received. Our implementation differs from COPE in the following ways. First, to increase the 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 small cost increase is 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 destined to itself and the latter acknowledge the receipt of packets destined 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 destined; and only the nodes that are specified as destinations will send feedback. In this way, we can reduce the receiver feedback especially when the multicast group is large but only a small number of nodes need the packet.

3.3 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?

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

Figure 1: Estimation of RTO.

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 Figure 1. RTO is initialized based on the MAC data rate. Our evaluation uses $K = 4$, $\alpha = 1/8$, and $\beta = 1/4$ as in TCP [21].

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.

3.4 Coding Problem and Algorithms

Another important design issue is how to code packets 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.

3.4.1 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 [11, 15]. 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 vertex in the coding graph. For any two packets that can be coded together, we draw an edge between their corresponding vertices. 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 vertices such that there is an edge between every pair of the vertices. 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 [10], 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, which is known to be NP-hard, 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 need it – $N(i)$, and (iii) for each packet i which clients have it – $H(i)$. For each vertex i in G of the minimum partition problem, we create a corresponding packet P_i in the coding problem. Each packet P_i is needed 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.

3.4.2 *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 and Section 5, 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.

Sort by time: The heuristic described in COPE [11] 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).

Sort by utility: We find the order in which packets are examined for potential coding is important. The previous heuristic codes the packet in the order of their arrival time. In the sort-by-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 we examine 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 this algorithm is useful for broadcast and multicast. In unicast, each packet is needed by one client, and has the same utility of 1. So it is equivalent to the sort-by-time heuristic under unicast.

Maximum clique: As shown in Section 3.4.1, the coding problem can be cast as finding a minimum clique partition in a coding graph. Therefore another approach 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 vertex of highest degree and iteratively adds additional vertices to the clique as long as they maintain the clique property – there is an edge between every pair of vertices in a clique.

Exhaustive search: 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 every 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 that node are the set of packets to transmit that minimizes the number of transmissions.

4. SIMULATION METHODOLOGY AND RESULTS

In this section, we first describe our simulation methodology and then present performance results.

4.1 Simulation Methodology

To evaluate the performance of various retransmission schemes presented in Section 3.4, 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 the same, and are varied from 10% to 90%. In heterogeneous 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 [18, 19], 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.

The high-level simulation evaluates a simplified scheduling algorithm, where a sender sends a constant-sized batch of packets at a time before starting retransmissions. Unless otherwise specified, the batch size is 20. In addition, we also evaluate the impact of varying batch sizes. Note that the batch-based scheduling tries to approximate the effect of the scheduling algorithm presented in Section 3.3. The simulation does not directly evaluate the latter scheduling, because it requires modeling timing dynamics, which the high-level simulation does not model. The testbed evaluation will directly evaluate the scheduling algorithm in Section 3.3.

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. We calculate the retransmission ratio for every 200 new packets that sender sends to each of the clients. Then we compute the average and standard deviation of retransmission ratios over 10 runs. Under all cases, the standard deviation of retransmission ratios is low – typically around 0.02 and no more than 0.08 over all the runs. So in the interest of space and clarity, we only present the average retransmission ratios in the following evaluation results.

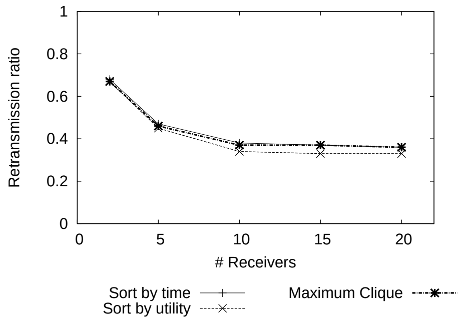
To ensure the same level of reliability, all retransmission schemes use an unlimited number of retransmissions so that they all achieve 100% delivery rate. Our results of bounded retransmissions are qualitatively similar. The only difference is that under extremely high loss rates (*e.g.*, 90%), the retransmission ratio under bounded retry count approaches 1 because the numbers 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 Simulation Results

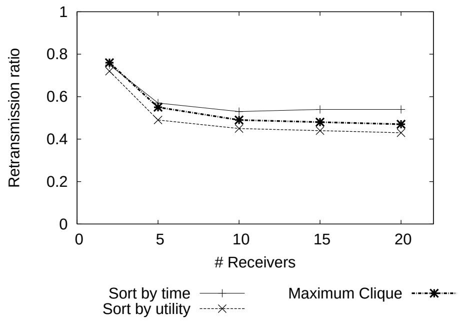
First we present the simulation results of multicast by varying the number of receivers, loss rates, and batch sizes. Then we present the unicast performance results.

4.2.1 Multicast Results under Homogeneous loss rates

Varying the number of receivers: Figure 2 and Figure 3 show retransmission ratios with a varying number of clients under Bernoulli and Gilbert loss models, respectively. We make the following observations.

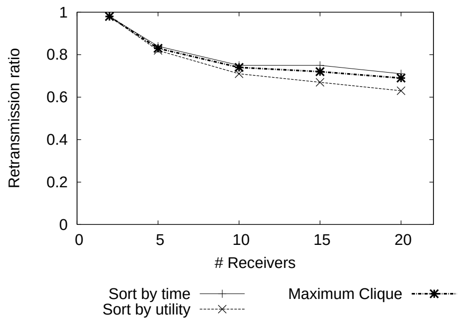


(a) 20% loss rate to each receiver

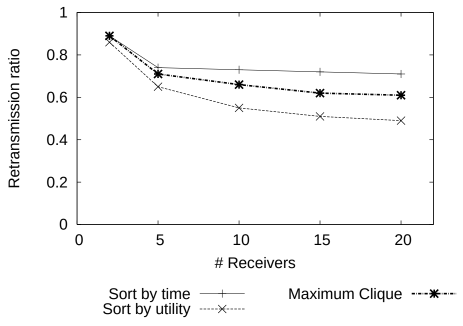


(b) 50% loss rate to each receiver

Figure 2: 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 3: Multicast comparison under a varying number of receivers with homogeneous Gilbert losses.

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%.

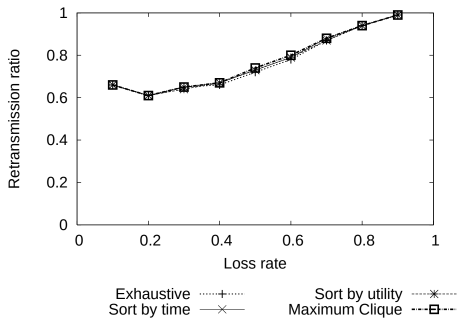
Second, the retransmission ratios decrease with the number of receivers, which suggests that the benefit of coding-based retransmissions increases with the number of receivers. This is because a larger number of receivers makes it easier to find receivers that lose different packets and create coding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

Third, comparing the three different coding algorithms, we observe the sort-by-utility algorithm out-performs the maximum clique, which out-performs the sort-by-time. Their performance difference is larger under the Gilbert loss model than under the Bernoulli loss model. The good performance of the sort-by-utility algorithm is likely because packets 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 with since 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 sort-by-utility algorithm makes a larger difference. In the interest of brevity, below we present the results under the Bernoulli loss model, and comment on the the Gilbert results whenever their difference is significant.

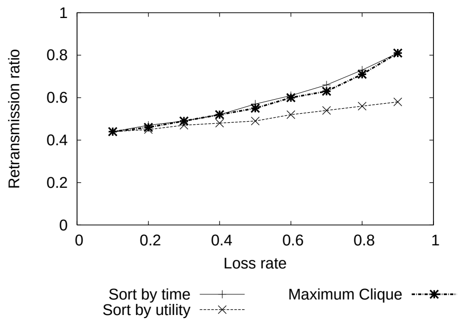
Varying loss rates: Next we evaluate the performance by varying loss rates. Figure 4 summarizes the results under 3, 5, and 10 receivers. For 3 receivers, we also plot the results of the exhaustive search; the results of the exhaustive search under a higher number of receivers are not available due to its high computational complexity. Under 3 receivers, the practical coding schemes perform 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%. The sort-by-utility continues to perform the best. In all cases, the ratios are lowest 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 5, 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 saving. The good news is that only a small batch (or delay) is needed to achieve significant saving.

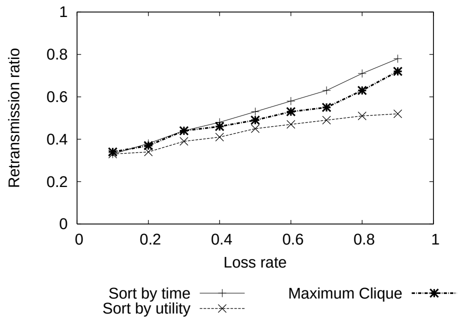
4.2.2 Multicast Results under Heterogeneous Losses



(a) 3 receivers



(b) 5 receivers



(c) 10 receivers

Figure 4: Multicast comparison under a varying loss rate with homogeneous Bernoulli losses.

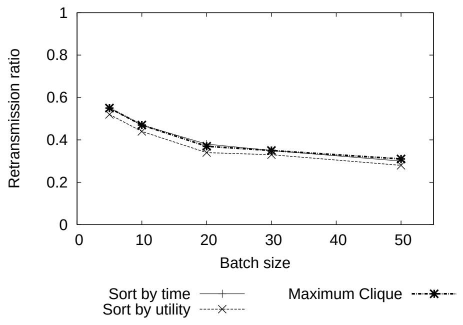
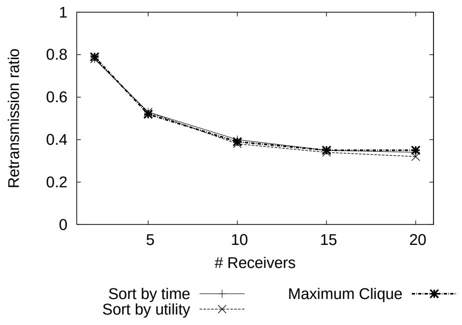


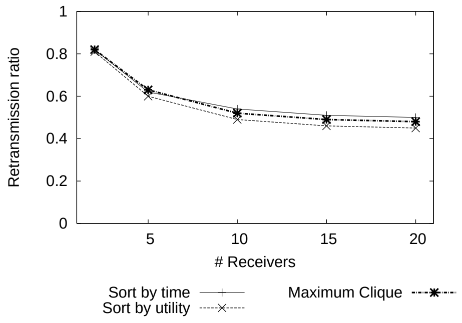
Figure 5: Multicast comparison under a varying batch size with 10 receivers and 20% homogeneous Bernoulli loss rates.

So far we consider 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 6(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 sort-by-utility out-performing the sort-by-time by up to 25%.



(a) 20% loss bound



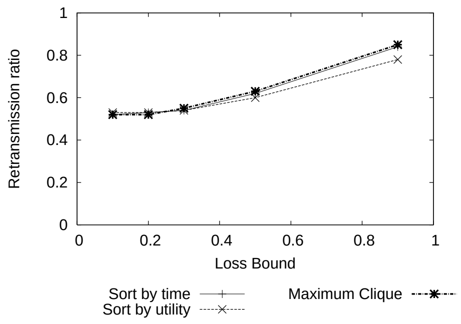
(b) 50% loss bound

Figure 6: Multicast comparison under a varying number of receivers with heterogeneous Bernoulli losses.

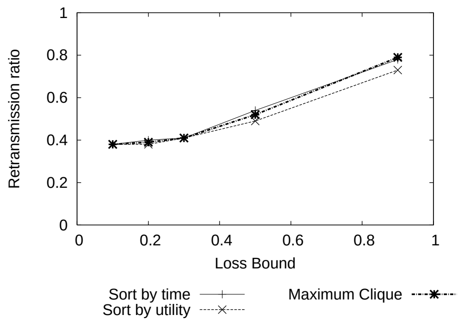
Varying loss bounds: We further evaluate the heterogeneous cases by varying the loss bound. Figure 7 summarizes the results under 5 and 10 receivers. As the loss bound increases, loss heterogeneity increases, which increases the retransmission ratio and decreases the coding benefit. This is expected 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.

4.2.3 *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 sort-by-utility and sort-by-time algorithms are equivalent under unicast, we only compare the sort-by-time and maximum clique with the basic retransmission.

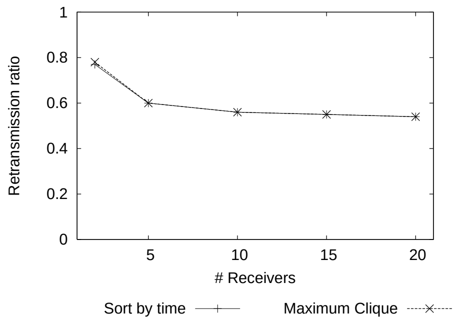


(a) 5 receivers

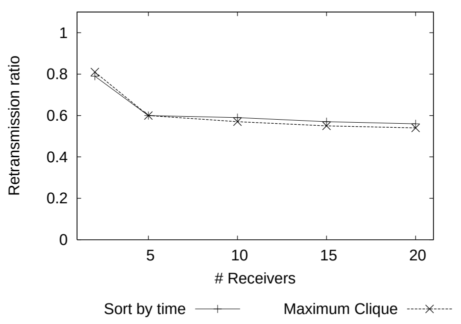


(b) 10 receivers

Figure 7: Multicast comparison under a varying loss bound with heterogeneous Bernoulli losses.



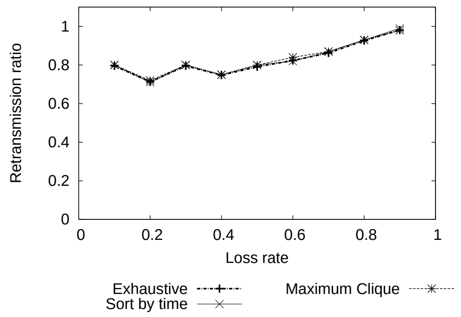
(a) 20% loss rate to each receiver



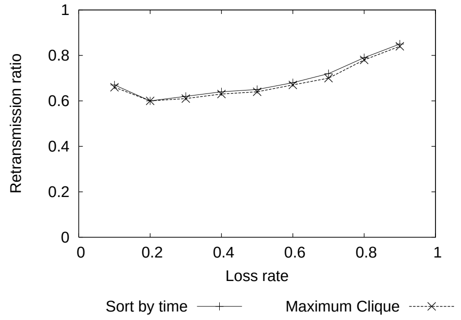
(b) 50% loss rate to each receiver

Figure 8: Unicast comparison under a varying number of receivers with homogeneous Bernoulli losses.

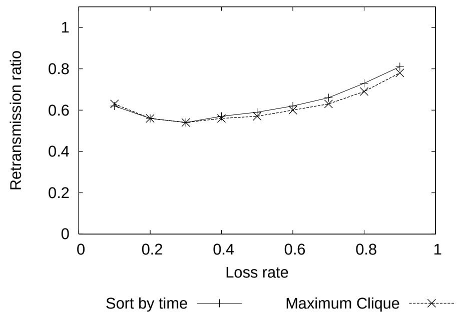
Varying the number of receivers: Figure 8(a) and (b) show the results under a varying number of receivers when the loss rate to each receiver is 20% and 50%, respectively. In both cases, the coding-based schemes achieve retransmission ratios between 0.6 and 0.8. As in multicast cases, with an increasing number of receivers, the retransmission ratio decreases and the coding benefit increases.



(a) 3 receivers



(b) 5 receivers



(c) 10 receivers

Figure 9: Unicast comparison under a varying loss rate with homogeneous Bernoulli losses.

Varying loss rates: Figure 9 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. Compared with the multicast performance in Figure 4, 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 coding opportunities.

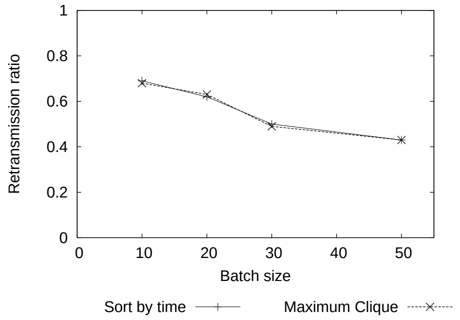


Figure 10: Unicast comparison under a varying batch size with 10 receivers and 20% homogeneous Bernoulli loss rates.

Varying batch sizes: Figure 10 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.

4.2.4 Unicast Results under Heterogeneous Losses

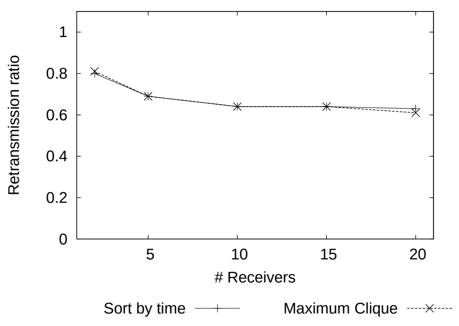
We also evaluate the retransmission schemes under unicast traffic using heterogeneous losses.

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 11, in both cases, 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.

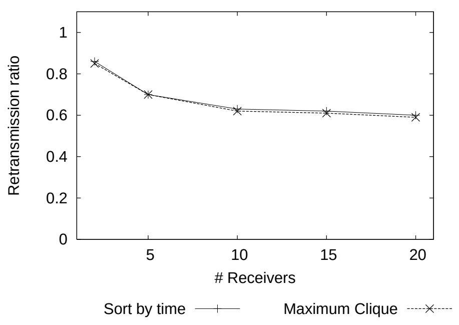
Varying loss bounds: We further evaluate the performance by varying the loss bounds while setting the number of receivers to 5 or 10. Figure 12 shows that the retransmission ratio initially decreases and then increases with the loss bound. The later increase is due to the same reason as in the multicast cases, where under 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 Summary

The simulation results show that coding-based retransmissions are effective in reducing the number of retransmissions required to recover packet losses. Their performance benefit increases with the number of receivers and the batch size. Moreover their performance gain is larger for multicast traffic. Comparing different coding-based heuristics, the sort-by-utility performs the best.

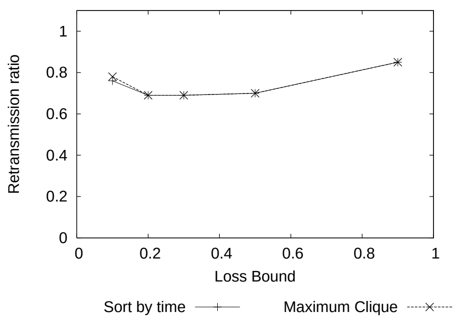


(a) Loss bound = 20%

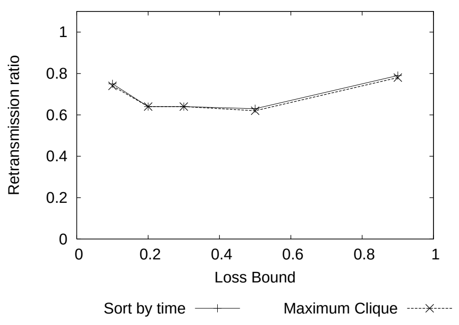


(b) Loss bound = 50%

Figure 11: Unicast comparison under a varying number of receivers with heterogeneous Bernoulli losses.



(a) 5 receivers



(b) 10 receivers

Figure 12: Unicast comparison for unicast under a varying loss bound with heterogeneous Bernoulli losses.

5. IMPLEMENTATION AND TESTBED EXPERIMENTS

In addition to high-level simulation, we also implement different 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.

5.1 Implementation

Our implementation is built on the COPE source code [7], 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 3.2. We implement the scheduling algorithm described in Section 3.3 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 two 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, and (ii) the coding-based retransmission using the sorted by time heuristic. The maximum retry count is 7 in both basic retransmission and the coding-based retransmission to achieve similar level of reliability, and 7 is commonly used retry count in IEEE 802.11. We plan to evaluate the performance of other coding heuristics as part of our future work.

5.2 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 packets are dropped using the Bernoulli loss model and all clients experience similar loss rates. For each scenario (*i.e.*, a given number of receivers and loss rate), we run seven times, where each time we obtain the retransmission ratio (defined in Section 4.1) 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. In addition, we compare the total throughput of ER and the basic retransmission by running a 30-second UDP transfer, and report throughput ratio, defined as the ratio of the ER's throughput against that of the basic retransmission scheme. A higher throughput ratio indicates a larger performance gain from ER.

5.3 Experiment Results

5.3.1 Multicast Evaluation

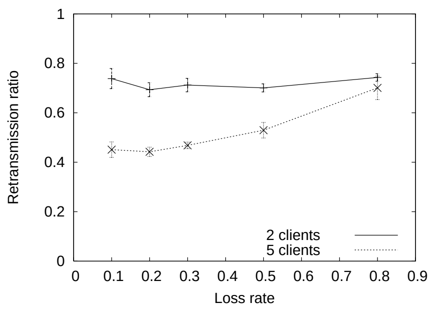


Figure 13: Multicast experiment results under a varying loss rate.

We first evaluate the multicast performance of ER by varying the loss rates. Figure 13 summarizes the results under 2 and 5 clients. The retransmission ratio is between 0.7 – 0.8 for 2 receivers, and between 0.4 – 0.7 for 5 receivers. These results are consistent with the simulation results, indicating that the benefit of ER extends to real wireless networks.

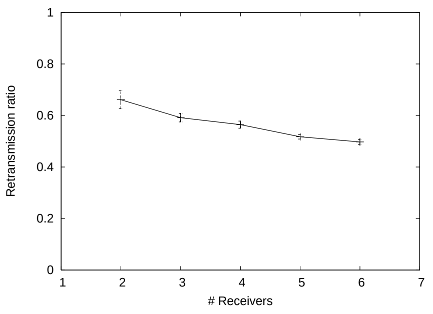


Figure 14: Multicast experiment results under a varying number of clients.

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 14, the retransmission ratio decreases from 0.7 to 0.5 as the number of receivers varies from 2 to 6. This shows that the benefit of ER increases with the number of receivers, which is consistent to the simulation results.

Finally we compare the throughput of ER and basic retransmission by varying the loss rate to each client. As shown in Figure 15, the throughput gain from ER can be quite significant: up to 21% gain for 2 clients, and up to 50% gain for 5 clients. Moreover, the throughput gain tends to increase with loss rate, because ER improves the efficiency of retransmission, which is more important under high loss rates.

5.3.2 Unicast Evaluation

Next we evaluate the unicast performance of ER. Figure 16 shows that the retransmission ratios are between 0.6 and 0.8 as the loss rate varies from 0.1 to 0.8. The retransmission ratio is lower under 5 clients than under 2 clients, as we would expect.

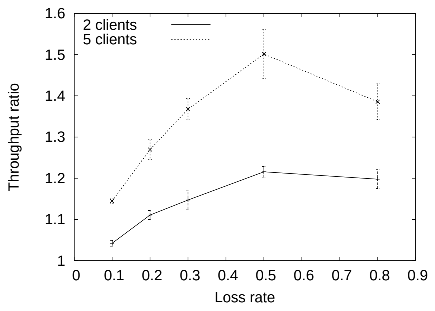


Figure 15: Compare throughput against the basic scheme for multicast under a varying loss rate.

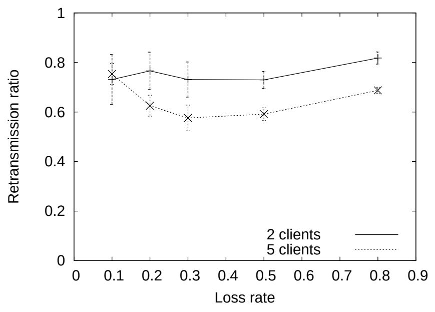


Figure 16: Compare retransmission mechanisms for unicast under a varying loss rate.

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 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 17.

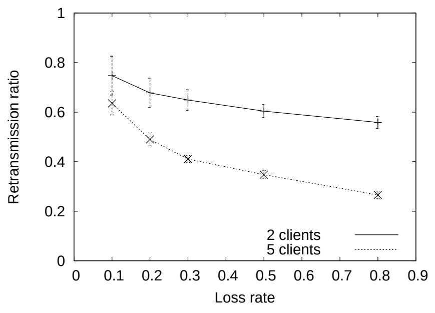


Figure 17: Compare retransmission mechanisms for unicast under a varying loss rate, where only packets destined to the receivers are dropped.

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 18, 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.

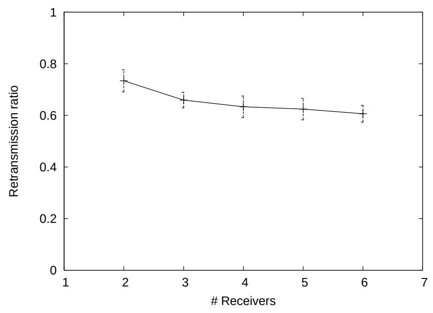


Figure 18: Unicast experiment results under a varying number of clients.

Finally we evaluate the performance in terms of throughput ratio by varying the loss rate to each client. As shown in Figure 19, the improvement of ER under unicast is generally less than under multicast, as we would expect. Nevertheless, we observe that throughput improves by up to 17% for 2 clients and up to 25% for 5 clients. As in multicast, the performance gain of ER under unicast also increases with loss rates, since ER helps to reduce more packet retransmissions under higher loss rates.

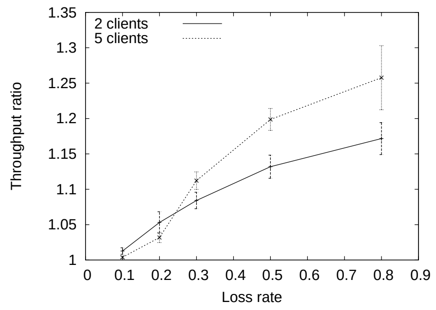


Figure 19: Compare throughput against the basic scheme for unicast under a varying loss rate.

6. CONCLUSION

In this paper, we develop 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. Using simulation and experiments, we show that ER is effective over a wide range of scenarios. In the future, we plan to study the performance of ER in multihop wireless networks.

7. REFERENCES

[1] IEEE standard for wireless LAN medium access control (MAC) and physical layer (PHY) specifications. 1999.

[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] 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.

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

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

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

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

[11] 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.

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

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

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

[15] 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.

[16] 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.

[17] 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.

[18] 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.

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

[20] 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.

[21] 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`.

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

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

[24] 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.

[25] 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.

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

[27] 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.

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

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

[30] 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.