The Application of Network Coding to Multicast Routing in Wireless Networks

by

Michael Jennings

B.A., Cornell University (2005)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2007

© Massachusetts Institute of Technology 2007. All rights reserved.

Author . . . .
Department of Electrical Engineering and Computer Science
May 29, 2007

Certified by . . . .
Dina Katabi
Assistant Professor
Thesis Supervisor

Accepted by . . . .
[illegible]h
Chairman, Department Committee on Graduate Students

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
AUG 16 2007
LIBRARIES
BARKER

The Application of Network Coding to Multicast Routing in Wireless Networks

by

Michael Jennings

Submitted to the Department of Electrical Engineering and Computer Science
on May 29, 2007, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering and Computer Science

Abstract

This thesis considers the application of network coding and opportunistic routing to improve the performance of multicast flows in wireless networks. Network coding allows routers to randomly mix packets before forwarding them. This randomness ensures that routers that hear the same transmission are unlikely to forward the same packets, which permits routers to exploit wireless opportunism with minimal coordination. By mixing packets, network coding is able to reduce the number of transmissions necessary to convey packets to multiple receivers, which can lead to a large increase in throughput for multicast traffic.

We discuss the design of a multicast enabled variant of MORE, a network coding based protocol for file transfer in wireless mesh networks, and evaluate this extension, which we call MORE-M, in a 20-node indoor wireless testbed. We compare MORE-M to a wireless multicast protocol that takes an approach similar to that of wired multicast by using the ETX metric to build unicast routing trees. We also compare MORE-M to a multicast enabled variant of the ExOR routing protocol. Experiments show that MORE-M's gains increase with the number of destinations, and are 35-200% greater than that of ExOR.

We then consider the problem of video streaming in a wireless local area network for applications such as video conferencing. A network coding based protocol that uses opportunistic receptions at clients is proposed. We evaluate the design in our testbed and demonstrate that the use of network coding and, in particular, the use of wireless opportunism increase the quality of the video stream.

Thesis Supervisor: Dina Katabi
Title: Assistant Professor

Acknowledgments

This thesis is the result of joint work with Szymon Chachulski and Dina Katabi. I owe tremendous thanks to both Szymon Chachulski and Dina Katabi and I owe a great deal of thanks to: Micah Brodsky, Sachin Katti, Asfandyar Qureshi, and Hariharan Rahul. Lastly, I'm thankful for the many open source tools that supported this work, particularly Click, Evalvid, and Madwifi. This research was performed while on appointment as a U.S. Department of Homeland Security (DHS) Fellow under the DHS Scholarship and Fellowship Program, a program administered by the Oak Ridge Institute for Science and Education (ORISE) for DHS through an interagency agreement with the U.S. Department of Energy (DOE). ORISE is managed by Oak Ridge Associated Universities under DOE contract number DE-AC05-00OR22750. All opinions expressed in this paper are the author's and do not necessarily reflect the policies and views of DHS, DOE, or ORISE.

Disclaimer

Many of the ideas in this thesis are the result of close interaction with Szymon Chachulski. Together we worked on the development of MORE, an opportunistic routing protocol that exploits network coding. Szymon's work has focused on the design of MORE, while mine has focused on extending the protocol to support multicast.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Wireless multicast is an important problem with applications to file distribution, delay tolerant networks, home entertainment, and video conferencing. As more wireless networks are deployed on city-wide scales, and as mobile wireless devices continue to replace their immobile wired counterparts, this importance will increase.

This thesis explores the use of network coding and opportunistic routing to improve the throughput of wireless multicast. We focus on two types of applications for wireless networks. We first consider file sharing, for which we propose MORE-M. MORE-M extends the recent opportunistic routing protocol, MORE [9], for multicast traffic. In MORE-M, forwarders randomly mix packets before forwarding them. This randomness limits spurious transmissions among forwarders without the use of coordination. Like MORE, MORE-M is able to exploit wireless opportunism without the cross-layer scheduling of the ExOR protocol [7], which is of key importance, as such scheduling becomes even less feasible in a multicast setting. MORE-M's use of network coding leads to additional throughput gains in that, unlike other multicast routing protocols that use wireless opportunism, MORE-M does not have to perform a retransmission for every distinct packet loss experienced by a destination in the multicast group. We elaborate on these two points with detailed examples below.

The second application we consider is video streaming in wireless local area networks, as, for example, in wireless video conferencing. Video streaming differs from file sharing in that it does not require complete reliability, it is more sensitive to delay, and packets are not homogeneous, but rather are of varying importance and exhibit dependencies according to the video compression scheme used and the type of compressed frame the packets belong to. We design a protocol to address the needs of wireless video applications. In particular, we consider packet inter-dependencies within the network coding based scheme we present, we use application-layer feedback to deal with losses, and we exploit the broadcast nature of the wireless medium to improve throughput.

We discuss the implementation of both designs and evaluate them in a wireless testbed. Our experiments reveal the following findings.

- For file transfer, MORE-M's throughput gains increase with the number of destinations. For 2-4 destinations, MORE-M's throughput is 35-200% higher than ExOR's. In comparison to multicast using traditional routing, the gain can be as high as 3x.
- For video streaming, our protocol's use of opportunistic receptions at the clients increases throughput roughly by a factor equal to the number of clients. Its use of network coding increases throughput by as much as 10% among 2 clients experiencing packet losses with relatively modest correlation in bandwidth-limited scenarios. With 6 clients and more correlated losses, the gains are slight. We believe this small improvement is due to the fact that our protocol does not use coding to achieve reliability, but further analysis is needed.

In the next section we discuss the benefits of network coding most relevant to this work. Chapter 2 presents a survey of related work. In Chapter 3 we give an overview of MORE. Chapter 4 describes the design of MORE-M, the performance of which is evaluated in Chapter 5. We discuss a network coding based solution for streaming video in wireless local area networks in Chapter 6. An analysis of this protocol's performance is given in Chapter 7. Chapter 8 concludes the thesis with a brief summary.

1.1 The Benefits of Network Coding

Network coding refers to the method of allowing routers inside the network to combine packets before transmitting them and has two important benefits relevant to multicast routing. First, it decreases the number of transmissions necessary to route packets to multiple

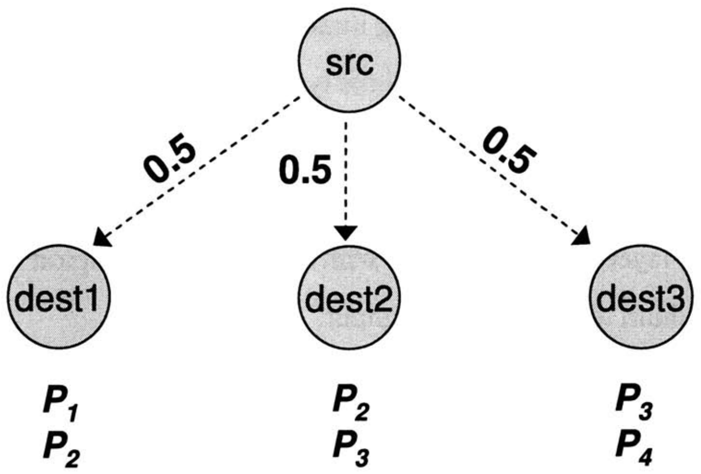

Figure 1-1—Network Coding Reduces Routing Overhead. Instead of retransmitting all four packets, the source can transmit two linear combinations, e.g., $p_1+p_2+p_3+p_4$ and $p_1+2p_2+3p_3+4p_4$. These two coded packets allow all three destinations to retrieve the four original packets, saving the source 2 transmissions.

receivers for both multi-hop and single-hop routing. Second, it reduces the need for coordination among routers in multi-hop routing. More generally, network coding allows for a natural and efficient means of loss recovery in the face of low-quality wireless links and provides for economical path diversity, which is particularly important for multicast traffic in the unstable and lossy environments characteristic of wireless networks.

1.1.1 Network Coding Reduces Routing Overhead

How can network coding increase the throughput of multicast flows in particular? Put informally, statically packetizing the information to be conveyed to multiple receivers using a lossy broadcast medium increases routing overhead. Nodes receive different sets of packets and routers are forced to perform many transmissions in order to fill the distinct holes at the receivers. Network coding reduces this overhead by allowing routers to combine packets so that a single transmission can fill different holes at multiple receivers. We refer to this decrease in routing overhead as **coding gain** and illustrate how it can increase throughput in the following example.

In Fig. 1-1, the source multicasts 4 packets to three destinations. Wireless receptions at different nodes have been observed to be independent in prior work [39, 36]. Assume that each destination receives the packets indicated in the figure–i.e., the first destination receives p_1 and p_2, the second destination receives p_2 and p_3, and the last destination receives p_3 and p_4. Note that each of the four packets is lost by some destination.

Without coding, the sender has to retransmit the union of all lost packets, i.e., the sender needs to retransmit all four packets. In contrast, with network coding, it is sufficient to transmit 2 randomly coded packets. For example, the sender may send $p'_1 = p_1+p_2+p_3+p_4$ and $p'_2 = p_1 + 2p_2 + 3p_3 + 4p_4$. Despite the fact that they lost different packets, all three destinations can retrieve the four original packets using these two coded packets. For example, the first destination, which has received p'_1, p'_2 and p_1, p_2, retrieves all four original packets by inverting the matrix of coefficients, and multiplying it with the packets it received, as follows:

$$\begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}^{-1} \begin{pmatrix} p'_1 \\ p'_2 \\ p_1 \\ p_2 \end{pmatrix}.$$

Thus, in this simple example, network coding has reduced the needed retransmissions from 4 packets to 2, improving the overall throughput.

1.1.2 Network Coding does not Require Coordination

Consider the scenario in Fig. 1-2. Traditional routing predetermines the path before transmission. It sends traffic along the path "$src \rightarrow R \rightarrow dest$", which has the highest delivery probability. However, wireless is a broadcast medium. When a node transmits, there is always a chance that a node closer than the chosen nexthop to the destination overhears the packet. For example, assume the source sends 2 packets, p_1 and p_2. The nexthop, R, receives both, and the destination happens to overhear p_1. It would be a waste to have node R forward p_1 again to the destination. This observation has been noted in [7] and used to develop ExOR, an opportunistic routing protocol for wireless mesh networks.

ExOR, however, requires node coordination, which is hard to achieve in a large network. Consider again the example in the previous paragraph. R should forward only

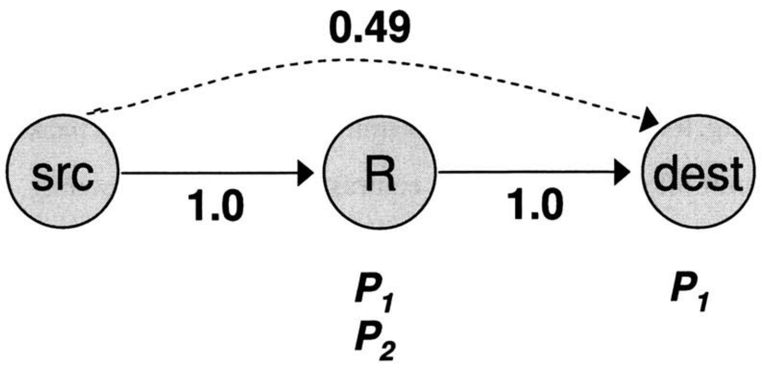

Figure 1-2—Network Coding does not Require Coordination. The source sends 2 packets. The destination overhears p_1, while R receives both. R needs to forward just one packet but, without node coordination, it may forward p_1, which is already known to the destination. With network coding, however, R does not need to know which packet the destination misses. R just sends the sum of the 2 packets $p_1 + p_2$. This coded packet allows the destination to retrieve the packet it misses independently of its identity. Once the destination receives the whole transfer (p_1 and p_2), it acks the transfer causing R to stop transmitting.

packet p_2 because the first packet has already been received by the destination; but, without consulting with the destination, R has no way of knowing which packet to transmit. The problem becomes harder in larger networks, where many nodes hear a transmitted packet. Opportunistic routing allows these nodes to participate in forwarding the heard packets. Without coordination, however, multiple nodes may unnecessarily forward the same packets, creating spurious transmissions. To deal with this issue, ExOR imposes a special scheduler on top of 802.11. The scheduler goes in rounds and reserves the medium for a single forwarder at any one time. The rest of the nodes listen to learn the packets overheard by each node. Due to this strict schedule, nodes farther away from the destination (which could potentially have transmitted at the same time as nodes close to the destination due to spatial reuse), cannot, since they have to wait for the nodes close to the destination to finish transmitting. Hence the scheduler has the side effect of preventing a flow from exploiting spatial reuse.

Network coding offers an elegant solution to the above problem. In our example, the destination has overheard one of the transmitted packets, p_1, but node R is unaware of this fortunate reception. With network coding, node R naturally forwards linear combinations of the received packets. For example, R can send the sum $p_1 + p_2$. The destination retrieves the packet p_2 it misses by subtracting from the sum and acks the whole transfer. Thus, R

need not know which packet the destination has overheard.

Indeed, the above works if R sends any random linear combination of the two packets instead of the sum. Thus, one can generalize the above approach. The source broadcasts its packets. Routers create random linear combinations of the packets they hear (i.e., $c_1p_1 + \ldots + c_np_n$, where c_i is a random coefficient). The destination sends an ack along the reverse path once it receives the whole transfer. This approach does not require node coordination and preserves spatial reuse.

Chapter 2

Related Work

This chapter provides a brief survey of work in wireless multicast, considers practical network coding in the context of wireless networking, and concludes with a discussion of video streaming.

2.1 Wireless Multicast

Considerable effort has been put into designing routing protocols for wireless multicast in MANETs [41, 37, 29], the On Demand Multicast Routing Protocol (ODMRP) being perhaps the most notable example. This work was motivated by the emergence of ad hoc networks for applications relevant to the military and first responders in disaster settings. Such protocols are most concerned with maintaining path availability in the face of random node movement and focus on reducing the resources required for routing packets. They use minimum-hop-count as the routing metric in an effort to take advantage of the wireless multicast advantage (WMA) [50]. WMA refers to the fact that in a wireless network where nodes use omnidirectional antennas, the wireless medium is shared and a single transmission can reach more than one party. The design of a sophisticated routing metric able to quantify potential WMA usage along a specific route remains open. WMA is strongly related to the notion of opportunistic routing [7], which refers to the objective of employing fortuitous receptions at routers that overhear local transmissions to improve routing efficiency, rather than depending on a static predetermined route. A substantial amount of recent work in wireless networking is built around these two fundamental concepts.

Research in multicast for wireless mesh networks is more recent, as the technology fueling it is younger. This research is concerned with designing routing protocols to decrease delay, decrease the total number of transmissions, and/or increase throughput. WMNs are static and node failure is not a concern, so performance is of great importance. For example, some recent work has considered the important practical question of how to define high throughput routing metrics for protocols that use link layer broadcast instead of 802.11 DCF with ARQ in order to leverage WMA more effectively [40], but has limited itself to adapting MANET multicast routing protocols to the WMN setting. On the other hand, much work in designing these protocols is theoretical, for example [42] is concerned with constructing multicast trees that require very low bandwidth consumption. Other work in the field has been driven by more application-specific goals in mind (e.g. resiliency [51]).

In Chapter 4 we present a multicast enabled variant of MORE [9], called MORE-M. MORE-M is designed to be a practical, general-purpose multicast routing protocol for WMNs that naturally utilizes WMA and opportunistic routing without incurring some of the negative side effects that previously proposed protocols have. MORE-M preserves the architectural abstraction between the routing and MAC layers and is able to exploit spatial reuse, unlike the ExOR protocol [7], for example.

2.2 Network Coding

Work on network coding began with the pioneering paper by Ahlswede et al. that establishes the value of coding in the routers and provides theoretical bounds on the capacity of such networks [3]. The combination of [32, 27, 19] shows that, for multicast traffic, linear codes achieve the maximum capacity bounds, and coding and decoding can be done in polynomial time. Additionally, Ho et al. show that the above is true even when the routers pick random coefficients [18]. Researchers have extended the above results to a variety of areas including content distribution [16], secrecy [8, 20], and distributed storage [21].

Of particular relevance to this work is recent research on wireless network coding [34, 24, 25, 9]. This work can be divided into three classes. The first is theoretical; it extends known information theory bounds from wired to wireless networks [34, 20]. The second is simulation-based; it designs and evaluates network coding protocols using simulations [38, 49]. The third is implementation-based; it uses implementation and testbed experiments to demonstrate achievable throughput gains for sensors and mesh networks [25, 22, 9].

In this thesis we design, implement, and evaluate network coding based wireless multicast routing protocols in order to understand how network coding can improve multicast throughput in practice. Our work and [9], upon which it is based, is distinct from prior research in network coding in that it is the first implementation-based work to consider wireless multicast.

2.3 Video Streaming

Multicasting video in a WLAN has become important to applications such as video conferencing and streaming multimedia. Here we provide some background material in video streaming that is relevant to the thesis and discuss related work.

2.3.1 Background

Video Compression

Compression is very important to video transport. An HDTV video signal, for example, would require 1.3 gbps of bandwidth in uncompressed form, but typical HDTV channel bandwidth is only 20 mbps [4]. Thus the signal must be compressed considerably without sacrificing perceived quality.

Under the MPEG-4 standard, video compression is accomplished via intraframe and interframe compression. Intraframe compression uses I frames, which compress the image of the original frame independent of preceding or succeeding frames in the stream by exploiting spatial redundancy within the pixels of the original frame. Interframe compression, on the other hand, exploits temporal redundancy in the video stream by using P and B frames. A P frame uses intraframe compression and also references previous I or P frames in the encoded video frame sequence. A B frame references both preceding and succeeding

I or P frames and does not use intraframe compression. B frames are never referenced by other frames. Complete decoding of a P or B frame requires that the frames on which they depend be available. Since I frames are coded independently of other frames, they tend to be much larger than P frames, though this depends on the nature of the video encoded - it is true for the types of videos seen in the applications we would like to support, such as video conferencing. Compressed video is grouped into a series of group of pictures (GOPs). Each GOP begins with an I frame, followed by successive P frames and optionally B frames. The size of the GOP of a video encoding is a fixed parameter, as is the number of B frames used in each GOP. Thus the number of I, P and B frames of all GOPs is fixed and determined by the encoding parameters.

Video Streaming

In this work, we use an MPEG-4 part 2 standard-compliant [1] codec to encode YUV formatted video files. The MPEG-4 standard uses object-based compression. Individual objects within a scene are tracked separately and compressed together to create an MPEG4 file. Separate objects in a scene are separately encoded and decoded. This allows identification and selective decoding of particular objects of interest and hence provides for interactivity and manipulation of content by allowing developers to control objects independently in a scene and makes MPEG-4 a good fit for multimedia compression and low bitrate transmission. An MPEG4 encoded file is converted into an MP4 container file with a hint track, resulting in an ISO compliant MPEG-4 stream. Hint tracks describe how the encoded frames should be packetized to support a streaming server or RTP and the converter takes parameters, such as the network MTU, to perform this conversion according to the network constraints (relying on IP fragmentation, for example, would lead to poor streaming performance). See [12] for a more complete discussion of this process.

The Challenges of Video Streaming

In this section, following the discussion in [4], we consider three main challenges that a transport protocol for video streaming faces. These challenges are not specific to wireless, but do become more difficult to address in a wireless environment. We also discuss the means used for responding to these challenges.

- **Bandwidth Constraints:**

  Bandwidth is limited and variable in a wireless network, making some form of rate control necessary. The sender can provide for rate control by estimating the available bandwidth via probes or modeling and adapting the send rate accordingly. For example, [10] uses multiple TCP friendly rate control connections between the client and server in order to more fully utilize the wireless bandwidth while providing end-to-end rate control. Exploiting the broadcast nature of the medium increases the informational capacity of the network, and is also important in addressing this challenge.
- **Delay Jitter:**

  Since the links in a wireless network generally experience a high rate of loss, the end-to-end packet delay is variable. However, packets in a video stream have a fixed playback time indicating when they must be decoded for the corresponding frame to be displayed. A late packet is therefore useless, and, because of the nature of the encoding, can propagate errors to other frames, even if those frames are received by their deadline. Additionally, although frames must be received at a constant rate, they are generally not all the same size, for example, I frames tend to be much large than P frames. This burstiness introduces an additional variability to end-to-end packet delay. Clients use a playback buffer to address this problem. Frames are collected until the buffer is full, at which point playback begins. The buffer effectively adds an offset to the playback time of each frame. The trade off in designing a buffering scheme is to balance the desire to limit the consequences of packet lateness with the desire to avoid an overly long initial buffering delay, and is explored in [45]. One technique used to reduce initial buffering delay is adaptive media playout [47], which involves decreasing the playback rate if the playback buffer has less frames than expected. The reduction in playback rate is generally small enough so as to be unnoticeable.
- **Packet Loss:**

  Besides creating variable delay, poor quality wireless links result in considerable packet loss. To deal with this, one can use FEC, link layer retransmissions, or source coding. FEC has the advantage of a low delay if matched well to the channel, but will require high overhead for a highly variable channel with bursty losses. Retransmission schemes can be designed to optimize a particular metric or application-specific goal. For example, a RaDiO [11] streaming system selects the next packet to retransmit based on expected distortion reduction (assuming reception), the packet's deadline, channel statistics, and feedback information. The advantage of using a retransmission-based scheme is that it is adaptable and tunable, but the disadvantage is the necessary RTT latency. Lastly, source coding can be used to conceal errors in the decoding process or in the use of error resilient video coding. See [4] for a discussion of those methods.

2.3.2 Related Work on Wireless Video Streaming

Delivering content in a wireless network with strict timing constraints is a difficult problem. Very low and variable bandwidth, high loss rates, contention among senders, packet collisions, background interference, and multipath fading make it very difficult to provide any performance guarantees and aggravate the problems discussed above. In this section we analyze the current techniques used for wireless video streaming.

[35] was the first to address the particular case of streaming over WLANs. They proposed a simple hybrid ARQ algorithm that combined FEC and ARQ by having the client send an acknowledgment after decoding a batch of k packets to which FEC had been applied. Other work has looked closely into particular aspects of video streaming in wireless. For example, [12] shows that the choice of encoding parameters is very important in a wireless setting. In particular, they show that halving the hint track MTU values increases the medium access requirements by 20% on account of the increase in header overhead and the increase in the number of packets to transmit. They also show that there is little benefit to decreasing I frame frequency, as this leads to a marginal bandwidth gain at the cost of an encoding that is much more sensitive to packet loss.

The general transport mechanisms for wireless video streaming that have been developed fall into two categories: application layer based and cross layer based. On the application layer side, packet scheduling is considered in [23]. They show that an earliest deadline first scheduling policy performs worse than a policy that takes into account the relative importance of a frame within its GOP. Their frame-based scheduling approach assigns a delay threshold to each frame which gives priority to frames that occur earlier in a GOP, since later frames in a GOP have a coding dependency on those frames. [11], mentioned above, describes a rate-distortion (R-D) optimized streaming system where a Lagrangian cost function is minimized by dividing time and bandwidth resources among packets. [46] proposes a simple frame dropping policy of discarding B frames when the network conditions become challenged, since B frames are the least important in a GOP, as no frame ever depends on them for its own decoding. [30] shows that adaptive sizing of the MAC layer frame in the presence of varying channel noise has a considerable effect on throughput and can be used to improve energy efficiency for the application's desired level of throughput. Path diversity is discussed in [48], where the focus is on the use of multiple description coding (MDC) and layered coding (LC). MDC is shown to be more effective when the application has specific delay constraints and long RTTs, while LC works well when a retransmission policy can be used along one of the paths.

Other solutions propose cross layer design (CLD) architectures. A timestamp-based content-aware adaptive retry (CAR) mechanism is proposed in [33]. A CAR-aided MAC judges whether to transmit a packet based on its retransmission deadline. This deadline is dependent on the packets temporal relationship and encoding dependencies with respect to other video packets in its GOP. CAR improves upon the design from [31] in that the retransmission decision is made after the completion of the most recent transmission, and thus in consideration of the playback schedule of the actual retransmission, which avoids late arrivals. A general CLD framework is outlined in [44]. They propose a layered CLD architecture and identify the parameters to be passed to the different network layers. Adaptive link layer techniques that adjust packet size, symbol rate, and constellation size according to channel conditions are used to improve throughput. At the MAC and network layers, the joint allocation of capacity optimizes throughput, while scheduling at the transport layer protects the video stream from packet losses while avoiding congestion.

The solution proposed in this thesis is at the application layer and is similar to many other application layer strategies in that it takes into account a frames relative importance within its GOP. The contribution of our solution is that it is the first network coding implementation-based approach to the problem. Additionally, we evaluate the design, and analyze the performance seen in our implementation.

Chapter 3

An Overview of the MORE Architecture

This chapter gives and overview of MORE, an opportunistic routing protocol that exploits network coding. In Chapter 4, we extend MORE to support multicast.

MORE is a routing protocol for stationary wireless meshes, such as Roofnet [2] and community wireless networks [43, 5]. Nodes in these networks are PCs with ample CPU and memory.

MORE sits below IP and above the 802.11 MAC. It provides *reliable* file transfer. It is particularly suitable for delivering files of medium to large size (i.e., 8 or more packets). For shorter files or control packets, standard best path routing (e.g., Srcr [6]) can be used.

Table 3.1 defines the terms we use throughout this thesis.

Term	**Definition**
Native Packet	Uncoded packet
Coded Packet	Random linear combination of native or coded packets
Code Vector of a Coded Packet	The vector of coefficients that describes how to derive the coded packet from the native packets. For a coded packet $p' = \sum c_i p_i$, where p_i's are the native packets, the code vector is $\vec{c} = (c_1, c_2, \ldots, c_K)$.
Innovative Packet	A packet is innovative to a node if it is linearly independent from its previously received packets.
Closer to destination	Node X is closer than node Y to the destination, if the best path from X to the destination has a lower ETX metric than that from Y.

Table 3.1—Definitions

3.1 Source

The source breaks up the file into batches of K packets, where K may vary from one batch to another. These K uncoded packets are called *native packets*. When the 802.11 MAC is ready to send, the source creates a random linear combination of the K native packets in the current batch and broadcasts the coded packet. In MORE, data packets are always coded. A *coded packet* is $p' = \sum_i c_i p_i$, where the c_i's are random coefficients picked by the node, and the p_i's are native packets from the same batch. We call $\vec{c} = (c_1, \ldots, c_i, \ldots, c_K)$ the packet's *code vector*. Thus, the code vector describes how to generate the coded packet from the native packets.

The sender attaches a MORE header to each data packet. The header reports the packet's code vector (which will be used in decoding), the batch ID, the source and destination IP addresses, and the list of nodes that could participate in forwarding the packet. ETX [13] is used to compute the forwarder list. Specifically, nodes periodically ping each other and estimate the delivery probability on each link. They use these probabilities to compute the ETX distance to the destination, which is the expected number of transmissions to deliver a packet from each node to the destination. The sender includes in the forwarder list nodes that are closer (in ETX metric) to the destination than itself, ordered according to their proximity to the destination.

The sender keeps transmitting coded packets from the current batch until the batch is acked by the destination, at which time, the sender proceeds to the next batch.

3.2 Forwarders

Nodes listen to all transmissions. When a node hears a packet, it checks whether it is in the packet's forwarder list. If so, the node checks whether the packet contains new information, in which case it is called an *innovative packet*. Technically speaking, a packet is innovative if it is linearly independent from the packets the node has previously received from this batch. Checking for independence can be done using Gaussian elimination. The node ignores non-innovative packets, and stores the innovative packets it receives from the current batch.

If the node is in the forwarder list, the arrival of this new packet triggers the node to broadcast a coded packet. To do so the node creates a random linear combination of the coded packets it has heard from the same batch and broadcasts it. Note that *a linear combination of coded packets is also a linear combination of the corresponding native packets.* In particular, assume that the forwarder has heard coded packets of the form $p'_j = \sum_i c_{ji} p_i$, where p_i is a native packet. It linearly combines these coded packets to create more coded packets as follows: $p'' = \sum_j r_j p'_j$, where r_j's are random numbers. The resulting coded packet p'' can be expressed in terms of the native packets as follows $p'' = \sum_j (r_j \sum_i c_{ji} p_i) = \sum_i (\sum_j r_j c_{ji}) p_i$; thus, it is a linear combination of the native packets themselves.

3.3 Destination

For each packet it receives, the destination checks whether the packet is innovative, i.e., it is linearly independent from previously received packets. The destination discards non-innovative packets because they do not contain new information. Once the destination receives K innovative packets, it decodes the whole batch (i.e., it obtains the native packets) using simple matrix inversion:

$$\begin{pmatrix} p_1 \\ \vdots \\ p_K \end{pmatrix} = \begin{pmatrix} c_{11} & \dots & c_{1K} \\ \vdots & \ddots & \\ c_{K1} & \dots & c_{KK} \end{pmatrix}^{-1} \begin{pmatrix} p'_1 \\ \vdots \\ p'_K \end{pmatrix},$$

where, p_i is a native packet, and p'_i is a coded packet whose code vector is $\vec{c_i} = c_{i1}, \dots, c_{iK}$. As soon as the destination decodes the batch, it sends an acknowledgment to the source to allow it to move to the next batch. Acks are sent using best path routing, which is possible because MORE uses standard 802.11 and co-exists with shortest path routing. Acks are also given priority over data packets at every node.

Chapter 4

MORE-M: MORE for Multicast Traffic

4.1 The Design of MORE-M

The basic design of MORE-M is very similar to that of MORE. The source transmits random linear combinations from a batch of K native packets and the forwarders transmit random linear combinations of the innovative packets they receive. Traffic is source-driven and forwarder transmissions are triggered by the reception of packets from upstream. A batch is completed when all destinations in the multicast group have delivered acks to the source.

4.1.1 Selecting the Forwarders for a Multicast Session

Opportunistic routing does not use a fixed path to route packets. A set of forwarders and an ordering among the forwarders is all that is required; the actual path of the flow varies dynamically with the underlying conditions of the network. In MORE, the source considers any node in the network that has a lower ETX to the destination than itself as a potential forwarder for unicast traffic. ETX gives a natural ordering among the forwarders that MORE inherits in order to determine which forwarder is upstream of another. MORE uses a heuristic to determine the expected number of transmissions that forwarders will need to make for every source transmission and then prunes forwarders from the forwarder set if they make less than 10% of the total expected number of transmissions required to route a batch of K packets to the destination. This pruning is done in order to limit the inefficiencies that would result from excessive contention.

MORE-M builds upon MORE's approach to determine the forwarder set. For each destination in the multicast group, MORE-M uses MORE's technique to obtain a forwarder set for the unicast flow from the source to that destination. The set of forwarders used by MORE-M for multicast traffic is the union of all forwarders obtained via this process. Forwarders are totally ordered using the ETX from the source to each forwarder. If the source has a lower ETX to forwarder i than to forwarder j, then forwarder i is considered upstream of forwarder j.

4.1.2 MORE-M's Transmission Strategy

Any routing protocol has to address two fundamental questions after determining a group of forwarders: When should forwarders transmit packets, and how many packets should be transmitted? The hop-by-hop retransmission method used by traditional routing cannot be applied in the context of MORE. This is because in opportunistic routing a next hop is not chosen until after transmission, and thus can vary across transmissions. MORE addresses these questions in two steps. First, MORE estimates the number of transmissions a forwarder must make for each source transmission in order to guarantee that a forwarder closer to the destination than the transmitting forwarder will receive the packet being forwarded. Second MORE legislates that forwarders transmit packets in proportion to this number according to a reception-driven transmission policy.

Let N be the number of nodes in the network. For any two nodes, i and j, let $i < j$ denote that i has a smaller ETX than j. Let p_{ij} denote the loss probability in sending a packet from i to j. Let z_i be the expected number of transmissions that forwarder i must make to route one packet from the source, s, to the destination, d. We assume that wireless receptions at different nodes are independent, an assumption that is supported by prior measurements [39, 36].

Let us calculate the number of packets that a forwarder j must forward to deliver one packet from source, s to destination, d, or, in other words, the load, L_j, on forwarder j. The expected number of packets that j receives from nodes with higher ETX is $\sum_{i>j} z_i(1-p_{ij})$. For each packet j receives, j should forward it only if no node with lower ETX metric hears the packet. This happens with probability $\prod_{k<j} p_{ik}$. Thus, in expectation, the number of packets that j must forward, denoted by L_j, is:

$$L_j = \sum_{i>j}(z_i(1-p_{ij})\prod_{k<j} p_{ik}). \tag{4.1}$$

Note that $L_s = 1$ because the source generates the packet.

Now, consider the expected number of transmissions a node j must make. j should transmit each packet until a node with lower ETX has received it. Thus, the number of transmissions that j makes for each packet it forwards is a geometric random variable with success probability $(1-\prod_{k<j} p_{jk})$. This is the probability that some node with ETX lower than j receives the packet. Knowing the number of packets that j has to forward from Eq. 4.1, the expected number of transmissions that j must make is:

$$z_j = \frac{L_j}{(1-\prod_{k<j} p_{jk})}. \tag{4.2}$$

From this we know the number of transmissions, z_i, that forwarder i should make for every packet sent by the source. Intuitively, we think of z_i as the amount of `credit` forwarder i is given for each source transmission. MORE legislates a reception-driven transmission policy for the forwarders, as most forwarders will not be able to detect when a packet was transmitted by the source. Define the `TX_credit` of a node to be the number of transmissions that a node should make for every packet it receives from upstream. For each packet sent from source to destination, node i receives $\sum_{j>i}(1-p_{ji})z_j$, where z_j is the number of transmissions made by node j and $(1-p_{ji})$ is the delivery probability from j to i. Thus, the TX_credit of node i is:

$$TX\_credit_i = \frac{z_i}{\sum_{j>i} z_j(1-p_{ji})}. \tag{4.3}$$

In MORE-M forwarder i is assigned the maximum z_i credit that it would receive in any of the unicast flows to a destination in the multicast group. TX_credit_i is then computed by considering all upstream nodes in the multicast forwarder set using Eq. 4.3. Assigning the maximum credit helps to ensure that the forwarder will have enough credit to forward the load it receives from upstream to each destination that it is a forwarder for. Forwarder i keeps a `credit counter` that is incremented by TX_credit_i when a packet is received from upstream. When forwarder i has the opportunity to transmit a packet, it checks the credit counter. If it is positive, the forwarder transmits the packet and decrements the credit counter by 1. If it is not positive, the forwarder does nothing.

4.1.3 Batch Completion

Once a destination in the multicast group is able to decode the current batch, an ack is sent to the source. An ack is given priority over data and local retransmissions are used at each hop to reliably deliver an ack along the minimum ETX path from the destination to the source. The source does not move onto the next batch until all destinations in the multicast group have acked the current batch.

As destinations ack the batch, the forwarders that were included in the forwarder set become superfluous and will generate spurious transmissions as they receive packets for destinations in the multicast group for which they are not forwarders. These spurious transmissions will generate even more spurious transmissions. In order to address this problem, whenever a destination acknowledges the current batch, the source recomputes the credit, z_i, for each forwarder i, as the maximum credit taken over only those hypothetical unicast flows to the destinations that have not yet acked the batch. For each forwarder i, TX_credit_i is then computed over the active upstream forwarders in the multicast group forwarder set. The source places this updated TX_credit_i assignment in the header of packets transmitted thereafter. The forwarders that receive these packets update their state accordingly.

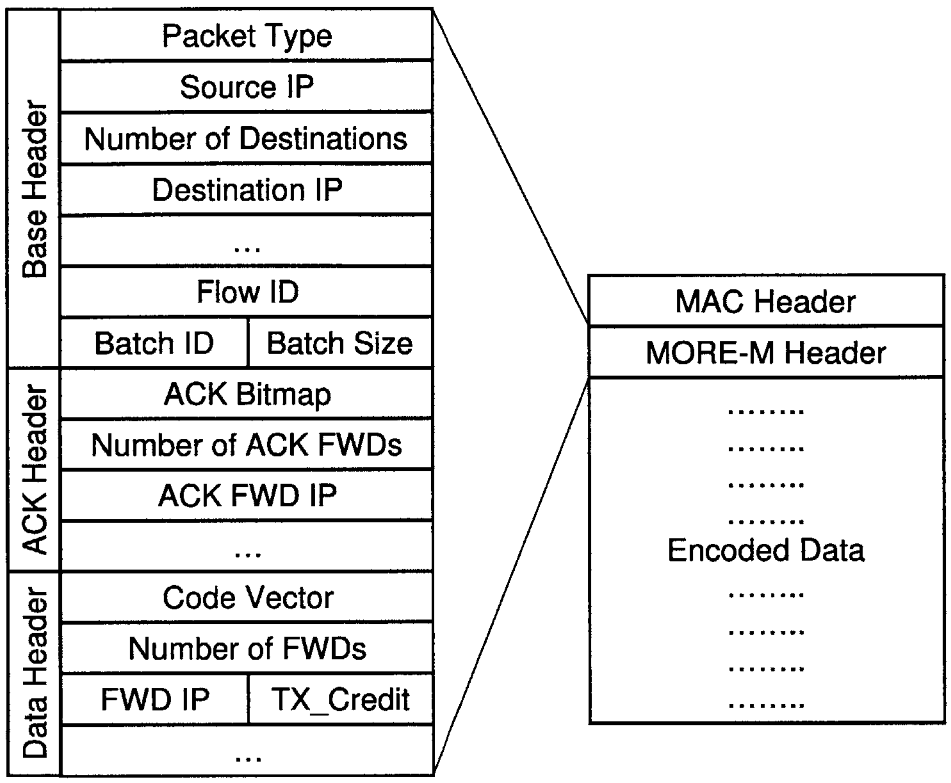

Figure 4-1—MORE-M Header.

4.2 Implementation Details

4.2.1 Packet Header

MORE-M inserts a variable length header in each packet, as shown in Fig. 4-1. The type field distinguishes data packets from acks. The source IP address, the IP address of each destination, flow ID, batch ID, and batch size follow. For ack packets, an ack bitmap is included in the header which indicates which of the destinations has acked the batch. Ack headers also contain the list of forwarders to use ordered according to the minimum ETX path to the source. For data packets, the code vector follows and identifies the coefficients that generate the coded packet from the native packets in the batch. Data headers also contain the list of forwarder IP addresses ordered according to their proximity to the source. For each forwarder i, the data header contains TX_credit_i as well.

4.2.2 Flow State

Each MORE-M node maintains per-flow state created on reception of the first packet from a flow that has the node's ID in the list of forwarders. This soft state is timed out after 5 minutes of no activity. The state includes:

- The `batch buffer` stores the received innovative packets. Note that the number of innovative packets in a batch is bounded by the batch size K.
- The `current batch` variable identifies the most recent batch from the flow.
- The `forwarder list` contains the list of forwarders and their corresponding TX_credits, ordered according to their proximity to the source.
- The `credit counter` tracks the transmission credit. For each packet arrival from a node with a higher ETX, the forwarder increments the counter by its corresponding TX_credit, and decrements it by 1 for each transmission. A forwarder transmits only when the counter is positive.

4.2.3 MORE-M Operation in Detail

Figure 4-2 shows the MORE-M operation in detail. A forwarder prepares a pre-coded packet for flows that have positive credit counters in order to avoid adding unnecessary delay to transmissions. The forwarder chooses among flows in round-robin order. A new packet is pre-coded for the currently selected flow and stored for future use immediately after transmission and the flow's credit counter is decremented by 1.

The handling of receptions depends on whether the node is the source, a forwarder, or a destination. The node first determines the type of the packet. The handling of acks is described below. The source discards data packet receptions. A forwarder checks the batch ID of the data packet. If it is higher than the forwarder's current batch, the forwarder updates its current batch and flushes packets from older batches from the batch buffer. If the packet was transmitted from upstream, the forwarder checks first whether its TX_credit must be updated and then increments the flows credit counter by its updated TX_credit. Innovative packets are added to the batch buffer while non-innovative packets are discarded. The forwarder's pre-coded packet for this flow is updated by adding the recent packet

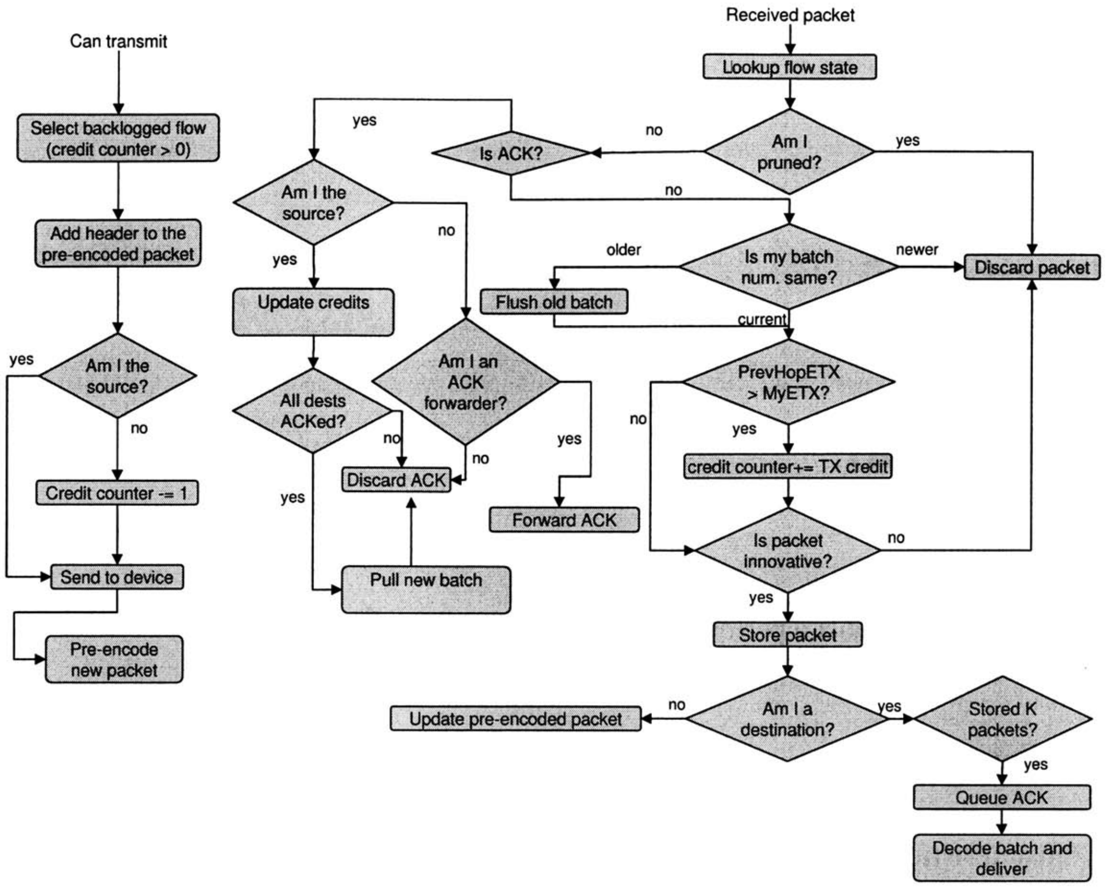

Figure 4-2—MORE-M Operation in Detail. The figure gives a detailed presentation of how MORE-M works.

multiplied by a random coefficient. A destination checks if the reception completes the batch, and queues an ack if so.

As noted above, ack packets are routed to the source along the shortest ETX path. Acks are prioritized over data packets and transferred reliably using link layer retransmissions. Source routing is used by the destination and the list of ack forwarders is placed in the MORE-M header so that the multicast forwarders know whether to forwarder the ack onward. All forwarders in the multicast forwarder set that overhear an ack update their current batch variable and flush packets from the acked batch from their batch buffer. Once the source receives an ack, it checks to see if all destinations have acked the batch. If so, a new batch is pulled. Otherwise, the source recomputes the TX_credits to take into account only the active destinations, as describe above, and uses the updated TX_credits in future transmissions.

Chapter 5

Evaluation

In this chapter we conduct a performance evaluation of MORE-M.

5.1 Testbed

(a) Characteristics: We have a 20-node wireless testbed that spans three floors in our building connected via open lounges. The nodes of the testbed are distributed in several offices, passages, and lounges. Fig. 5-1 shows the locations of the nodes on one of the floors. Paths between nodes are 1–5 hops in length, and the loss rates of links on these paths vary between 0 and 60%, and averages to 27%.

(b) Hardware: Each node in the testbed is a PC equipped with a NETGEAR WAG311 wireless card attached to an omni-directional antenna. They transmit at a power level of 18 dBm, and operate in the 802.11 ad hoc mode, with RTS/CTS disabled.

(c) Software: Nodes in the testbed run Linux, the Click toolkit [28] and the Roofnet software package [2]. Our implementation runs as a user space daemon on Linux. It sends and receives raw 802.11 frames from the wireless device using a libpcap-like interface.

5.2 Performance of MORE-M

We use measurements from a 20-node wireless testbed to evaluate MORE-M and compare it with a variant of ExOR extended for multicast traffic. We also compare MORE-M to

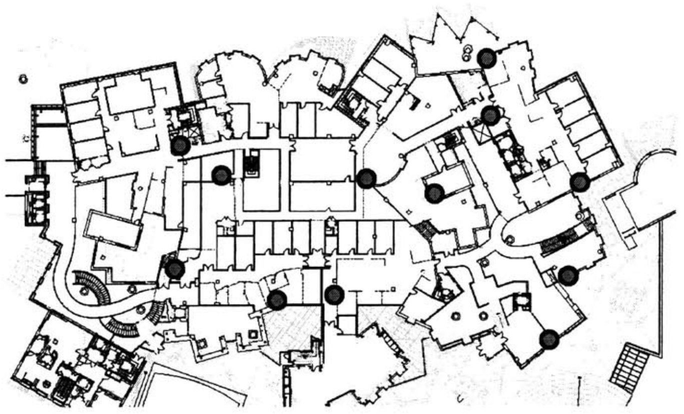

Figure 5-1—One Floor of our Testbed. Nodes' location on one floor of our 3-floor testbed.

a traditional best path routing protocol that adopts the same approach as wired multicast to form an MST among nodes in the multicast group using the ETX metric. We note the following results:

- MORE-M's throughput gain increases with the number of destinations. For 2-4 destinations, MORE-M's throughput is 35-200% larger than ExOR's. In comparison to traditional multicast routing, the gain can be as high as $3x$.
- In terms of overhead, MORE-M stores the current batch from each flow. Our implementation supports up to 44 mbps on low-end machines with Celeron 800MHz CPU and 128 kilobytes of cache. Thus, the overhead is reasonable for the environment it is designed for, namely stationary wireless meshes, such as Roofnet [2] and community wireless networks [43, 5]. See [9] for a discussion of this result.

5.2.1 Compared Protocols

We compare the following three protocols:

- *MORE-M* as described in Chapter 4.
- *ExOR* [7], the current opportunistic routing protocol. ExOR does not have multicast extensions. Thus, we must define how it deals with multicast. The extension must exploit WMA and opportunistic receptions. The protocol works by determining the ExOR forwarders for each destination. The per-destination forwarders use the ExOR protocol to access the medium and coordinate their transmissions. In contrast to unicast

ExOR, if the forwarders toward destination X opportunistically hears a packet by a forwarder in the forwarder list of destination Y, it exploits that opportunistic reception. Said differently, we allow opportunistic receptions across the forwarders of various destinations.

- *Srcr* [6] which is a state-of-the-art best path routing protocol for wireless mesh networks. It uses Dijkstra's shortest path algorithm where link weights are assigned based on the ETX metric [13] and uses the wired multicast approach to form an MST among nodes in the multicast group.

5.2.2 Setup

In each experiment, we run Srcr, MORE-M, and ExOR in sequence between the same source destination pairs. Each run transfers a 5 megabyte file. We leverage the ETX implementation provided with the Roofnet Software to measure link delivery probabilities. Before running an experiment, we run the ETX measurement module for 10 minutes to compute pair-wise delivery probabilities and the corresponding ETX metric. These measurements are then fed to all three protocols, Srcr, MORE-M, and ExOR, and used for route selection.

Unless stated otherwise, the batch size for both MORE-M and ExOR is set to $K = 32$ packets. The packet size for all three protocols is 1500 bytes. The queue size at Srcr routers is 50 packets. In contrast, MORE-M and ExOR do not use queues; they buffer active batches. The experiments are performed over 802.11b with a bit-rate of 5.5 mbps.

5.2.3 Results

Our results show that MORE-M's throughput is significantly higher than both ExOR and Srcr. In particular, we experiment with the simple topology in Fig. 5-2, where the source multicasts a file to a varying number of destinations. Fig. 5-3 shows the average throughput as a function of the number of destinations. The average is computed over 40 different instantiations of the topology in Fig 5-2, using nodes in our testbed. As expected, the per-destination average throughput decreases with increased number of destinations.

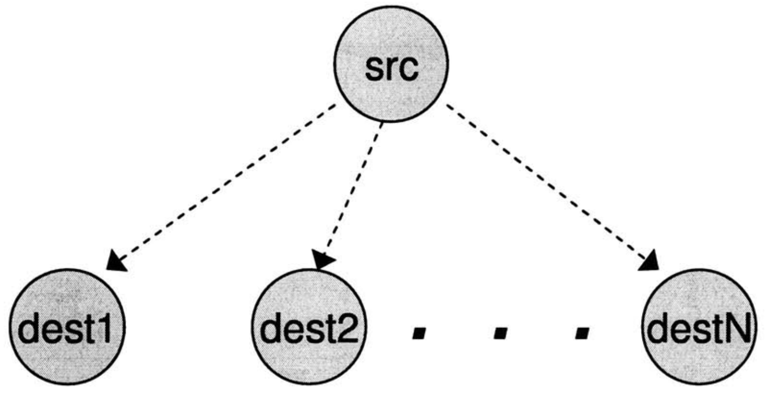

Figure 5-2—Fig. 5-3 Topology.

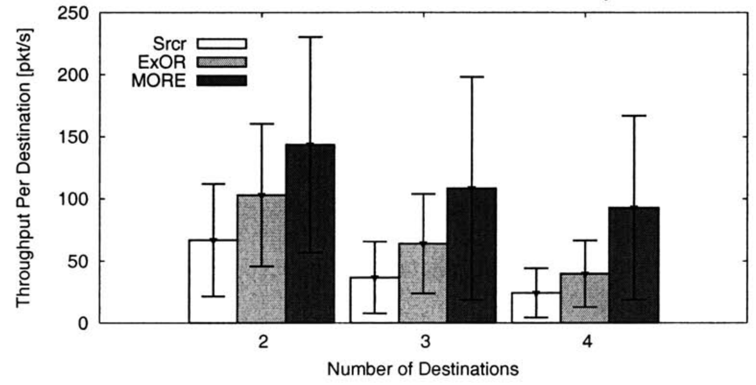

Figure 5-3—Throughput as a Function of the Number of Destinations for the Topology in Fig. 5-2. The figure shows the per-destination throughput of MORE-M, ExOR, and Srcr. The thick bars show the average per-destination throughput taken over 40 runs with different nodes. The lines show the standard deviation.

Interestingly however, the figure shows that MORE-M's throughput gain increases with increased number of destinations. MORE-M has 35-200% throughput gain over ExOR and 100-300% gain over Srcr.

Next, we run multicast over random topologies and multihop paths. We pick a source and 3 destinations randomly from the nodes in the testbed. We make the source multicast a file to the three destinations, using MORE-M, ExOR, and Srcr. We repeat the experiment for 40 different instantiations of the nodes, and plot the CDFs of the throughput. Fig. 5-4 confirms our prior results showing significant gain for MORE-M over both ExOR and Srcr. In this figure however the difference between MORE-M and ExOR is less pronounced than in Fig. 5-3. This is because the CDF uses random topologies with all nodes in the testbed

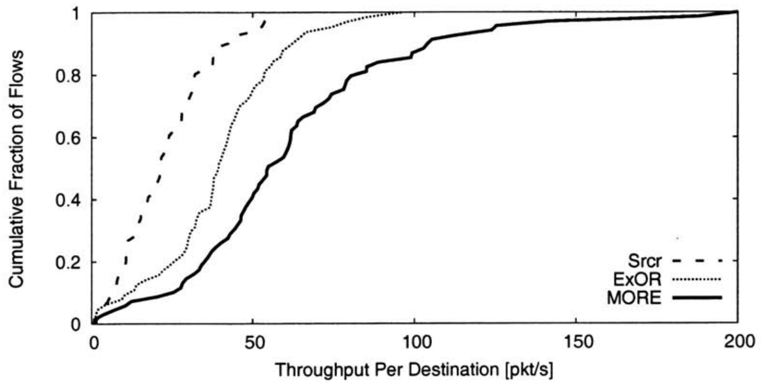

Figure 5-4—CDF of Throughput for 3 Destinations in a Random Topology. The figure shows the CDF of the per-destination throughput of MORE-M, ExOR, and Srcr. For each run, a source and 3 destinations are picked randomly from among the nodes in the testbed.

potentially acting as forwarders. This increases the potential for opportunistic receptions and thus makes the relative gain from network coding look less apparent.

Chapter 6

Streaming Video in a WLAN

In this chapter we design an opportunistic routing protocol that is suitable for video streaming over a WLAN. Video streaming differs from file transfer in that it does not require complete reliability, it is more sensitive to delay, and packets are not homogeneous. Our protocol addresses these differences as well as the particular challenges described in Chapter 2.

6.1 Design Overview

The server broadcasts packets in the stream to the clients. The clients play the stream in synch one playback buffering interval behind the stream at the server. The playback buffer itself is equal to the time required to play nx GOPs, where $n \geq 2$ and $x \geq 1$, and a GOP consists of one I frame followed by 4 or more P frames. The clients send periodic nacks reliably using link layer retransmissions every x GOPs (see fig. 6-1). These nacks indicate which packets in the $(n-1)x$ GOPs that have most recently been streamed by the server must be retransmitted. The server performs retransmissions when spare bandwidth is available.

The server retransmits packets in order of earliest deadline. Within a GOP, the server uses random linear codes to retransmit the lost I frame packets. Rather than nacking particular I frame losses, the clients simply nack the first k packets of an I frame, where k is the number of innovative packets necessary to achieve full rank for decoding. In order to retransmit the lost I frame packets, the server determines the maximum number of innovative packets needed by a client for decoding and broadcasts that number of random linear combinations of I frame packets. Random linear codes are used for I frame packet retransmissions in order to reduce the number of transmissions required to fill the holes at the clients. Recall from Chapter 1 that we called this decrease in overhead **coding gain**. Since losses are experimentally observed to be independent [39, 36], clients will lose different I frame packets. The size of the union of losses among clients will be larger than the number of innovative packets required by any particular client. Since I frames span many packets, the use of random linear codes leads to a more efficient use of bandwidth.

After broadcasting coded I frame packets, the server retransmits lost P frame packets in order of earliest deadline. Packets coded with random linear codes generally cannot be decoded until full rank is attained among the code vectors of the packets. For the encoded videos of the applications we'd like to support there are not enough packets in a P frame to consider applying linear codes to just the packets of the frame, while coding over multiple P frames is not ideal because, if full rank is not achieved, one would be unable to decode multiple P frames. For these reasons, random linear codes are not used for retransmitting packets belonging to P frames. However, to improve efficiency, the server uses a greedy coding algorithm to decrease the total number of retransmissions of P frame packets. The server considers the next P frame packet p_i to retransmit and determines which clients have not nacked p_i, and then searches for the packet p_j of a P frame with earliest deadline that was nacked by at least one of those clients, but not by the clients that nacked p_i. If such a packet is found, it is XORed with p_i. Note that p_j can be taken from among $(n-1)x$ GOPs if the server has streamed all packets in the buffering interval. The server repeats this coding process until no such packet can be found or until the coded packet p_i consists of packets such that every client has nacked exactly one of them and every client can decode the transmission based on its current state. See Algorithm 1 for a more formal presentation of the server's retransmission algorithm.

Algorithm 1 The server's GOP retransmission algorithm

```
MAX_T = 0
for each client C do
    Determine number of innovative packets, T, requested by C.
    if MAX_T < T then
        MAX_T ← T
Broadcast random linear combinations of the I frame packets MAX_T times.
for each nacked P frame packet $p_i$, in order of earliest deadline do
    Add clients that nacked $p_i$ to the set of covered clients (covered clients are clients for whom
    the transmission will be innovative)
    while there are clients not covered do
        Search for the P frame packet with earliest deadline that covers any of those clients but not
        the clients covered by the currently coded packet $p_i$
        if such a packet exists then
            XOR that packet with $p_i$ and add those clients to the set of covered clients
        else
            break
    Broadcast the coded packet and remove the nack associated with each coded packet for those
    clients that can decode the transmission based on their current state.
```

6.2 Solutions to the Three Challenges

Our protocol exploits both the broadcast nature of the medium and uses network coding to achieve better performance. It addresses the three challenges that a streaming protocol must face as described below.

- **Bandwidth Constraints:**

  To improve bandwidth utilization, the server broadcasts a single stream to the clients, while the clients play the video in synch with the stream. This use of wireless opportunism reduces the overhead at the server and allows the server to use a network coding based retransmission scheme, which is second source of more efficient utilization.

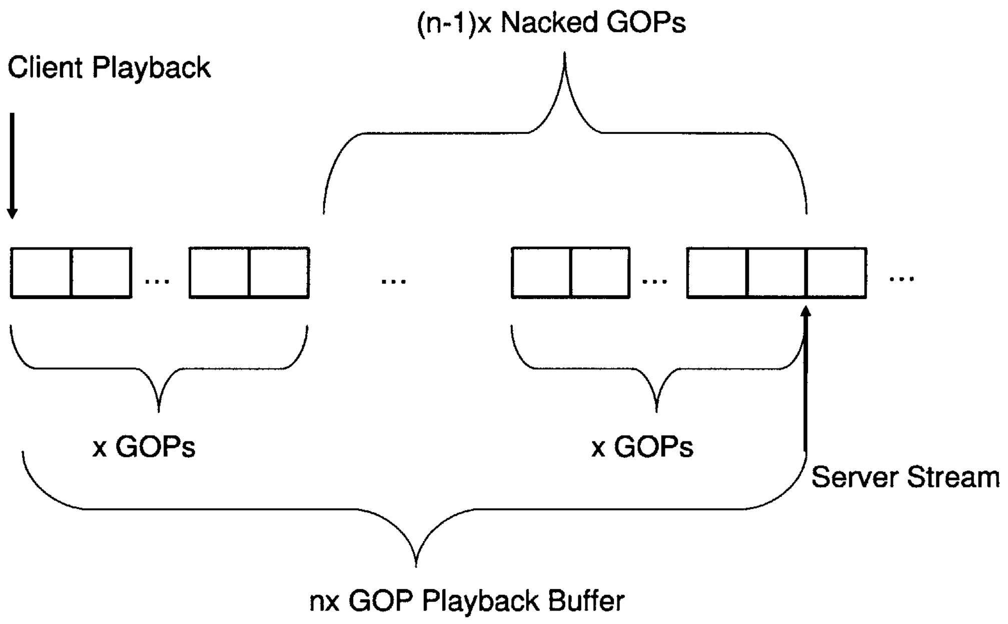

Figure 6-1—WLAN Video Streaming Protocol Playback Buffer and Nacking Scheme. This figure shows the playback buffer used in the protocol. The client playback is nx GOPs behind the stream at the server. Every x GOPs, the client sends a nack reliably to the server for the (n-1)x GOPs the server has most recently streamed.

- **Delay Jitter:**

  As noted above, each client uses a playback buffer equivalent to the time required to play all frames in nx GOPs, where $n \geq 2$ and $x \geq 1$, and a GOP consists of one I frame followed by 4 or more P frames. This buffer is used to deal with delay jitter. Playback at a client starts once the the first frame in the $(nx + 1)$th GOP is delivered by the application to the server for streaming. We assume that the clients have memory to buffer packets that may arrive for the next playback buffering interval so that such packets are not dropped. The total memory capacity required is equal to the amount of buffering for $(n + 1)x$ GOPs. Since the clients remain synchronized in playback with the server's input stream, the server will not transmit packets from more than nx distinct GOPs, while the clients consider at most x GOPs distinct from those the server is streaming.

- **Packet Loss:**

  The packets transmitted by the server are unlikely to reach all clients most of the time. Therefore, the clients periodically send nacks reliably every x GOPs. These nacks are for packets in the $(n-1)x$ GOPs that have most recently been sent by the server. Fig. 6-1 provides an illustration of the clients' playback buffer and nacking scheme. Each packet can be nacked up to $(n-1)$ times by a client. Nacks are sent reliably using link layer retransmissions, and jitter is added to each nack transmission to reduce the potential for collisions at the server.

6.3 Design Considerations

6.3.1 Client Feedback

In order to avoid the overhead of m separate streams for m separate clients, a broadcast-based scheme with application layer retransmissions is used. The clients need to get feedback to the server for losses to be detected. It is not obvious what event should trigger feedback. This observation led to the use of periodic feedback. Note that a larger x means less nack overhead at the expense of less feedback. Using a larger x may also introduce more P frame packet coding opportunities. A larger n increases feedback, at the expense of an increase in the initial buffering delay at the clients. Generally n should be adjusted according to the link qualities seen in the network relative to the burstiness of the stream. A larger nx product also leads to a larger space of packets that can be nacked, yielding larger nack packets.

6.3.2 Coding

The streams we consider have GOPs consisting of a large I frame, spanning many MTU-sized packets, followed by small P frames, often spanning only one packet. We use random linear codes to retransmit lost I frame packets. The number of transmissions required to deliver the I frame to the clients is thereby reduced from the size of the union of lost I frame packets to maximum number of I frame losses at a particular client because of coding gain, as discussed in Chapter 1. If receptions at the clients are independent and the clients' links to the server are roughly equivalent, then the union of lost I frame packets will be larger than the number of losses at any particular client. As mentioned previously, based on the

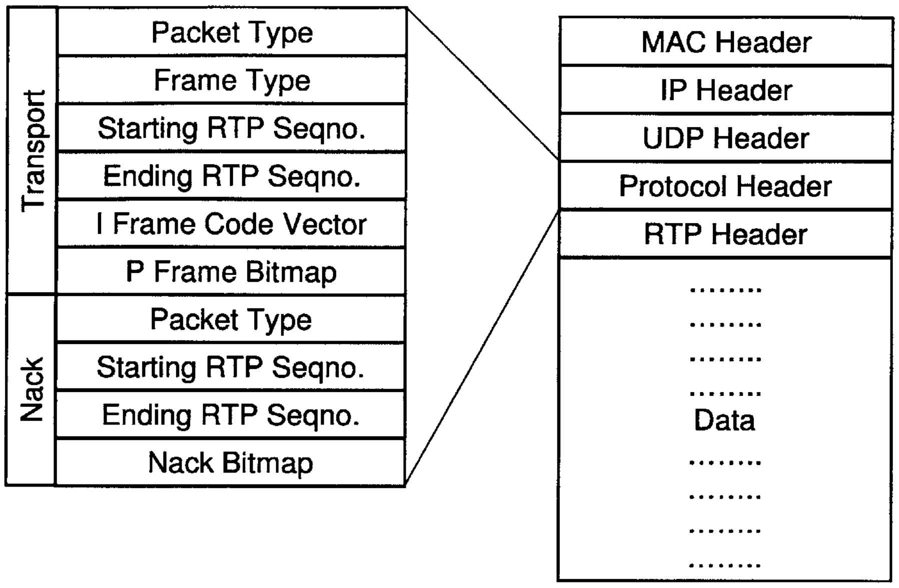

Figure 6-2—WLAN Video Streaming Protocol Header.

experimental results of [39, 36], we expect receptions among clients to be independent. Thus the use of random linear codes can reduce the overhead of retransmitting the lost I frame packets, meaning more bandwidth will be made available to the protocol, which will lead to performance gains. The greedy strategy of coding P frame packets is used with the same goal in mind - to reduce the number of retransmissions necessary to stream the media to clients and increase throughput.

6.4 Implementation Details

In this section we consider the details behind our implementation.

6.4.1 Packet Format

The protocol header sits above the RTP header in the networking stack. The clients can determine the frame that a packet belongs to by using the packet's RTP header. Our implementation uses two packet types: transport packets and nack packets. The first byte in the header is used to distinguish the packet type. Fig. 6-2 provides an illustration of the protocol header and its place in the networking stack.

Transport Header

The next byte in the header of a transport packet indicates whether the packet(s) are from P frames or I frames. For I frames, the next 16 bits in the header give the sequence number of the first packet in the I frame. The sequence number of the last packet in the I frame is given by the succeeding 16 bits. This is followed by a code vector of bytes giving the coefficient applied to each I frame packet. For P frames, the next 16 bits in the header give the sequence number of the first packet coded. The next 16 bits give the sequence number of the last packet coded. A bitmap follows which indicates which packets in the sequence number space given by the first and last sequence numbers, excluding those sequence numbers, have been XORed.

Nack Header

The next byte of the nack header is unused. The next 16 bits indicate the RTP sequence number of the first packet nacked. The next 16 bits indicate the RTP sequence number of the last packet nacked. A bitmap for each packet in the sequence number space, excluding the first and last sequence numbers, follows. A marked bit in the bitmap indicates a nack for the packet with the corresponding sequence number. Nacks for I frame packets are slightly different. Rather than nacking particular packets, the client determines the number, k, of innovative packets required to decode the I frame, and nacks the first k packets in the I frame in order to convey this information to the server. Thus, nacks for I frame packets merely convey the rank of the buffered coded packets at the client. Note that the RTP header is not actually included in a nack packet, as it is unnecessary.

6.4.2 Server Retransmission

The server keeps a bitmap for each packet in the current buffering interval indicating which client has nacked the packet. Additionally, each packet has a retransmission time associated with it. In order to avoid redundant retransmissions, the server ignores a nack for a packet if the nack is received within the expected amount of time it would take to broadcast the packet and then receive a reliable nack since the server last retransmitted the packet.

The retransmission time of a packet is set to the current time whenever the packet is retransmitted.

The basic retransmission algorithm is implemented as follows. We define the **retransmission interval** to be the $(n-1)x$ GOPs the server can retransmit packets from. We define the **starting GOP** to be the first GOP of the retransmission interval. The server keeps track of the next packet it must stream in the current buffering interval. If spare bandwidth is available, the server retransmits nacked packets.

Every I frame has a retransmission counter that indicates the number of coded transmissions the server needs to perform. This counter is updated when the server receives a nack for the I frame and when a coded retransmission from among the I frames packets is performed. If a nack is received, the server determines the number, T, of innovative packets the client needs. The I frame's transmission counter is then set to T if T is greater than the current transmission counter value. If the nack is received within the expected amount of time it would take to broadcast a coded packet and then receive a reliable nack since the server last transmitted a linear combination from the I frame packets, 1 is subtracted from T before the server attempts to update the I frame's transmission counter.

To perform a retransmission, the server begins at the starting GOP within the retransmission interval and determines the first packet nacked by at least one client. If the packet belongs to an I frame, the server decrements the I frame's retransmission counter. If the retransmission counter is zero after the decrement, all nacks on the I frame packets bitmaps are cleared. The server then transmits a random linear combination of the I frame packets. If the packet is a P frame, the nack bitmap of the P frame packet is cleared and the greedy coding algorithm described above is used. The relevant bits on each coded P frame packet are cleared for those clients who nacked those packets and who can decode the transmission. The coded P frame packet is then transmitted.

6.4.3 Client Decoding

The clients buffer innovative coded I frame packets they receive if their playback time has not passed. When a new I frame packet arrives, the client attempts to decode the I frame and updates the number of innovative packets required to decode it if it cannot decode it at that point. If it can decode the I frame, the client pushes the frame to the video application. I frame packets that are not innovative are discarded. Clients separately buffer P frame packets in the playback buffer in order to use them to decode coded P frame packet retransmissions. Coded P frame packets that cannot be decoded immediately are discarded.

Chapter 7

Evaluation

In this chapter we evaluate the WLAN video streaming protocol. We focus on determining the performance gained through the use or wireless opportunism and network coding.

7.1 Testbed

The nodes and software from the WMN experiments are used in our experiments, but we use a different topology. Fig. 7-1 shows the locations of the nodes used in our experiments involving 6 clients. We designate one of the testbed nodes as the server AP. The server has bidirectional links to each client. The loss rates of the links vary between 0-50% in both directions, depending on the current network conditions, with an average of about 30%. Unless noted otherwise, in our experiments we use 802.11b with a bitrate of 2 mbps and channel 3, which does experience some background interference with other traffic. Generally the nodes have the same quality links in both directions to the server and were positioned with that goal.

7.2 Performance of the WLAN Video Streaming Protocol

We evaluate our design by examining how using opportunistic receptions and network coding affect performance as we limit the bandwidth available to the server. From our experiments we are able to draw the following conclusions.

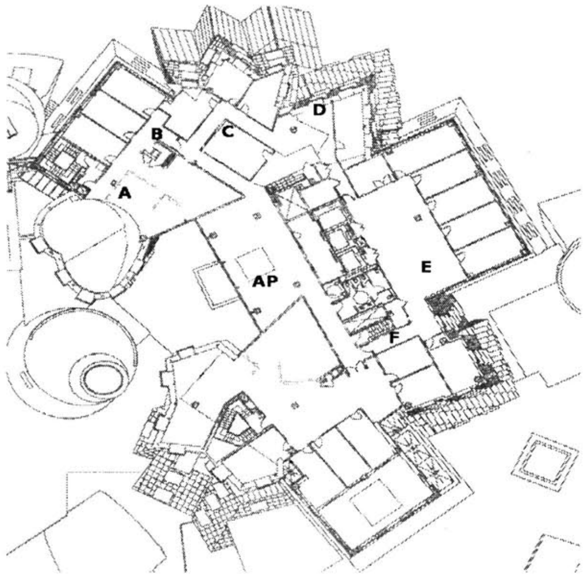

Figure 7-1—The WLAN Topology.

- Exploiting opportunistic receptions at the clients increases throughput roughly by a factor equal to the number of clients.
- The use of network coding increases throughput by as much as 10% among 2 clients experiencing packet losses with relatively modest correlation. This increase in throughput translates to higher quality video playback according to the objective metric we use. We see a slight increase in performance with network coding when we stream video to 6 clients. We believe this modest improvement is due to the fact that our protocol does not use coding to achieve reliability, but further analysis is needed.

7.2.1 PSNR: An Objective Video Quality Metric

Peak signal-to-noise ratio (PSNR) is used throughout the literature to measure the performance of video streaming protocols and we use it in evaluating our design. The PSNR between the luminance component of a source frame S and a destination frame D is defined as [26]:

$$20 \log_{10}\left(\frac{V_{peak}}{\sqrt{\frac{1}{N_{col}N_{row}} \sum_{i=0}^{N_{col}} \sum_{j=0}^{N_{row}} (Y_S(i,j) - Y_D(i,j))^2}}\right) \tag{7.1}$$

Where $V_{peak} = 2^k - 1$, k being the number of bits used to represent the luminance component of a pixel, and $Y(i, j)$ is the luminance component of the particular pixel in column i and row j of the frame.

Performance of a video transport protocol can be measured by considering the average, over all frames, of the PSNR of the decoded transported video relative to the original unencoded video to the PSNR of of the decoded video relative to the original unencoded video. This gives an objective measurement of how much quality has been lost in the network. This objective measurement can be roughly translated to a subjective human quality impression on a scale of 1 to 5, where 5 is the score for the highest quality impression. This scale is called the Mean Opinion Score (MOS). [26] gives a conversion table and provides some additional discussion.

7.2.2 Compared Protocols

We compare the following three approaches in order to determine the performance gain from coding and wireless opportunism:

- The design described in Chapter 6, which we refer to as OPCODE.
- The Chapter 6 design, but where the server performs retransmissions without the use of coding. Nacked packets are retransmitted in order of earliest achievable deadline. In particular, the server retransmits all nacked I frame packets. We refer to this protocol as OP.
- A scheme where the server uses a separate unicast stream for each client. Clients ignore packets that are not specifically destined for them. We refer to this scheme as BASELINE.

7.2.3 Setup

We use the Akiyo_cif.yuv video from the Evalvid web site [14]. This is a video of a newscast, and is a good fit for the video conferencing application we would like to support. The video is looped 5 times for a total of 1500 frames. We use ffmpeg [15] to encode and decode into MPEG-4 compressed video, and use MP4Box from the GPAC project [17] to import the encoded video into a *.mp4 container and to generate hint tracks for the RTP protocol

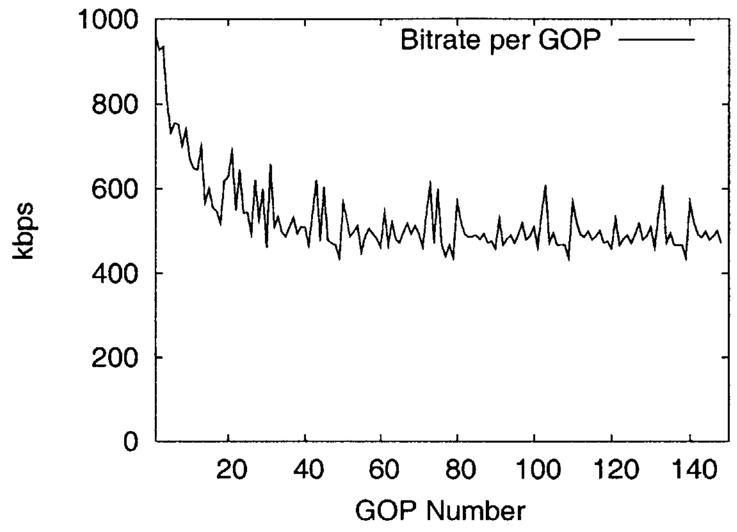

Figure 7-2—Streaming Bitrate of Akiyo_cif.yuv Encoded Video as a Function of GOP

for streaming. The Evalvid [26] toolkit is used to measure PSNR. In our experiments we encode the video at 30 fps with a bit rate of 500 kbps, using a GOP of 10, and do not use B frames. The reference PSNR for this encoding is 43.13. Fig. 7-2 shows the streaming bitrate in kbps as a function of GOP. This plot shows that ffmpeg does not encode the video at 500 kbps for the initial GOPs. One of the beginning GOPs has an output bitrate of 962 kbps, for example.

In order to evaluate the protocols we artificially limit the bandwidth available to the server. The server is given an initial credit equal to the number of bytes that can be transmitted at the chosen bandwidth in a given time interval. The credit is updated with this initial credit at the conclusion of every time interval. We used a time interval of 50 milliseconds in our experiments. The server cannot transmit a packet until the credit exceeds the packet's byte length, and the credit is decremented by the byte length of the packet when a transmission is made.

7.2.4 Results

Loss Independence

To attain coding gain on retransmissions of the packets of a particular I frame, the clients should lose different packets belonging to the I frame. That is, the size of the union among

Client 1	**Client 2**	**Mean**	**STD**
A	B	0.6871	0.2742
A	C	0.3570	0.3056
A	D	0.3674	0.2959
A	E	0.3315	0.2556
A	F	0.3064	0.2421
B	C	0.3751	0.3061
B	D	0.3845	0.3050
B	E	0.3190	0.2518
B	F	0.3020	0.2329
C	D	0.5410	0.4041
C	E	0.4932	0.3719
C	F	0.3939	0.3447
D	E	0.5727	0.3731
D	F	0.4648	0.3592
E	F	0.6121	0.3442

Table 7.1—I Frame Loss Correlation. The table shows the mean and standard deviation over all I frames of the normalized difference 7.2. We used 802.11b at 2 mbps and streamed the Akiyo_cif.yuv video with the encoding parameters given above using broadcast. It provides some insight into the correlation among losses of an I frame.

all clients of lost packets of the I frame should be larger than the size of the set of the particular losses of the I frame's packets at any client. In this experiment, the server broadcasts packets in the stream. We look at the pairwise independence of I frame packet losses. For each I frame and each pair of clients C_1, C_2 we consider three loss rates: L_1, the I frame packet loss rate at C_1, L_2, the I frame packet loss rate at C_2, and $L_{1\cap 2}$, the loss rate of I frame packets lost by both C_1 and C_2. For each I frame we consider the the normalized difference:

$$\frac{L_{1\cap 2} - L_1 L_2}{\sqrt{L_1 L_2} - L_1 L_2}. \tag{7.2}$$

This difference is defined to be 0 if either L_1 or L_2 are 0 or if both L_1 and L_2 are 1. We report the mean and standard deviation over all I frames for each client pair in table 7.1. A number closer to 0 indicates less correlated losses, while a number closer to 1 indicates more correlated losses. This table shows that I frame packet losses are correlated to some extent, and the variance is high. If we consider the topology in fig. 7-1, we see that nodes that are physically closer to one another are likely to have more correlated losses.

Streaming Performance

Given accurate link estimates, appropriate choice of parameters, and loss independence, we expect OPCODE to outperform OP in bandwidth-limited scenarios due to coding gain, which, as described in Chapter 1, will reduce the number of transmissions required to stream the video to multiple clients, leading to an increase in throughput when all bandwidth must be utilized. In order to verify this hypothesis, we consider two scenarios. In the first, the server streams video to 2 clients that have very lossy links which appear to experience losses with low correlation, and one in which the server streams video to the 6 clients used in the previous experiment. In the first case we expect to see a distinct performance difference. We expect that trend to continue when we increase the number of clients.

Two Clients with Less Correlated Loss In this experiment we position two nodes farther apart from the server in order to increase the average bidirectional loss rates from 30% to 55%, partly with the goal of finding a setting where losses appear less correlated than seen in the previous experiment. The mean normalized difference 7.2 for this was setup was 0.2621 with standard deviation 0.1772. We set $x = 1$ and $n = 6$, which corresponds to an initial delay of 2 seconds. We justify choosing $n = 6$ by noting that this choice provides up to 5 retransmission opportunities for lost packets, which should be enough for most packets to be successfully received. We choose $x = 1$ because nacks from 2 clients every 333 milliseconds should not create too much overhead. Recall from Chapter 6 that the RTP header is not included in a nack packet. The header overhead for a nack packet consists of 72 bytes for the link-layer, network, and transport headers, and 6 bytes plus the number of bytes in the nack bitmap for our protocol-specific header. The largest number of packets in five consecutive GOPs in our encoding is 158. This puts an upper bound of 100 bytes on the nack packets.

As discussed above, we impose an artificial bandwidth limitation on the server. We vary the bandwidth from 0.60 mbps to 2.0 mbps in increments of 0.1 mbps. We focus on the experiment among 5 trials that saw median PSNR performance increase. We consider the averages over the 2 clients of the percentage of packets received in fig. 7-3, the number and types of packets retransmitted in fig. 7-6, and the bandwidth utilization in fig. 7-7.

From fig. 7-3 we conclude that BASELINE delivers a little over half the packets of OP when bandwidth is limited. BASELINE does not utilize wireless opportunism, and hence performs up to twice as many transmissions as necessary. The server in BASELINE discards many I frame packets without transmitting them because these packets are large, have the same frame timestamp, and occur in bursts. Thus, BASELINE delivers few I frame packets to the clients and uses most of its bandwidth on P frame packets. BASELINE is thus impractical in low bandwidth scenarios, especially when there are a large number of clients.

OPCODE is able to deliver more packets to the clients than OP. In particular, at 0.70 mbps, OPCODE delivers about 10% more packets to the clients than OP. Note that fig. 7-6 shows that OP performs more retransmissions of I frame packets than OPCODE. This is expected, since the size of the union of lost I frame packets at the 2 clients is larger than the number of losses at either client when losses are not strongly correlated. As a result, OPCODE is able to perform more P frame retransmissions and deliver more packets to the clients, since both protocols utilize most of the available bandwidth, as indicated in fig. 7-7. Fig. 7-6 also shows the number of coded P frame retransmissions that OPCODE performs. The figure demonstrates that this is a nontrivial number of total P frame retransmissions when bandwidth is limited. Since OPCODE generally performs at least as many P frame retransmissions as OP, this is a source of coding gain that allows OPCODE to deliver more packets to the clients on average than OP. From this discussion we conclude that, when bandwidth is limited, coding allows OPCODE to perform more P frame retransmissions, and that the coding gain from coding I frame packets and P frame packets increases the number of I and P frame receptions at the clients. This is illustrated in fig. 7-4 and fig. 7-5, which show the percentage of packet receptions according to frame type.

We now consider the average PSNR for each bitrate in fig. 7-8. We see that the increase in receptions using OPCODE does translate to higher PSNR values.

Six Clients with Correlated Loss In this experiment, the server streams video to the 6 clients in the testbed with the topology of fig. 7-1 and mean normalized differences of table 7.1. We set $x = 2$ and $n = 6$, which yields a nack header overhead of at most 120

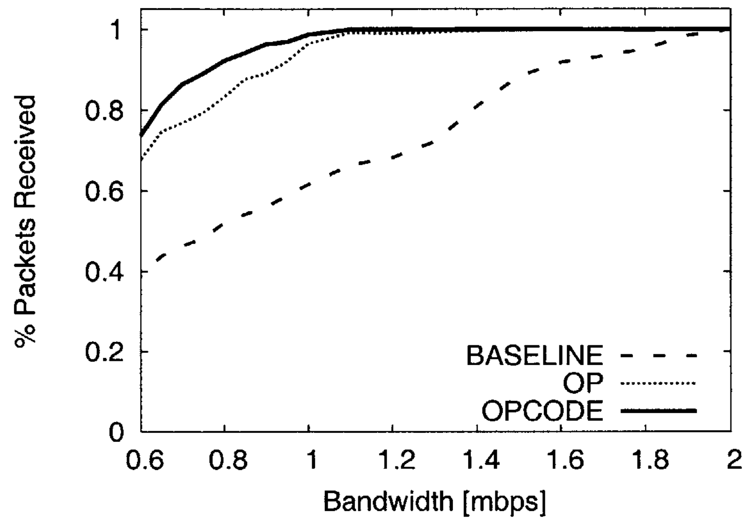

Figure 7-3—Percentage of Packets Received in 2 Client Experiment.

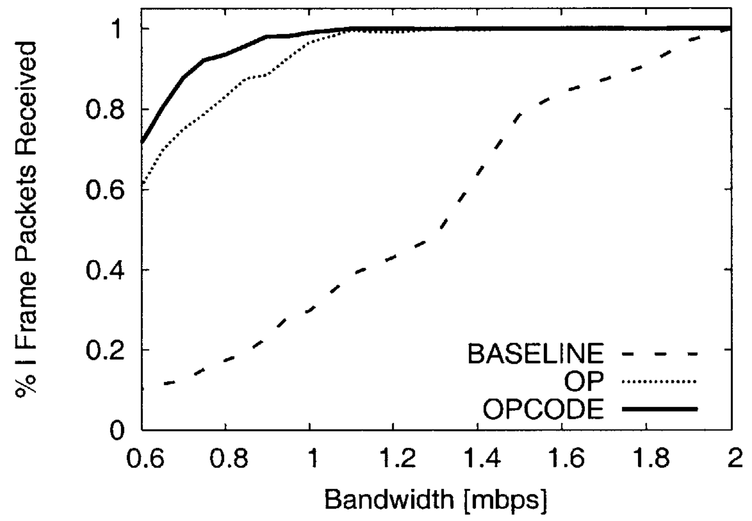

Figure 7-4—Percentage of I Frame Packets Received in 2 Client Experiment.

bytes and corresponds to an initial delay of 4 seconds. Here we have increased x in order to reduce number of nack packets transmitted.

As before, we focus on the experiment among 5 trials that saw median PSNR performance increase and look at the average PSNR for each bitrate fig. 7-9. We conclude that the use of coding modestly increases the PSNR values. The 6 client case is less pronounced for bandwidth limited scenarios because loss rates are lower and losses themselves appear more correlated. In this experiment, BASELINE did not perform well enough to extract

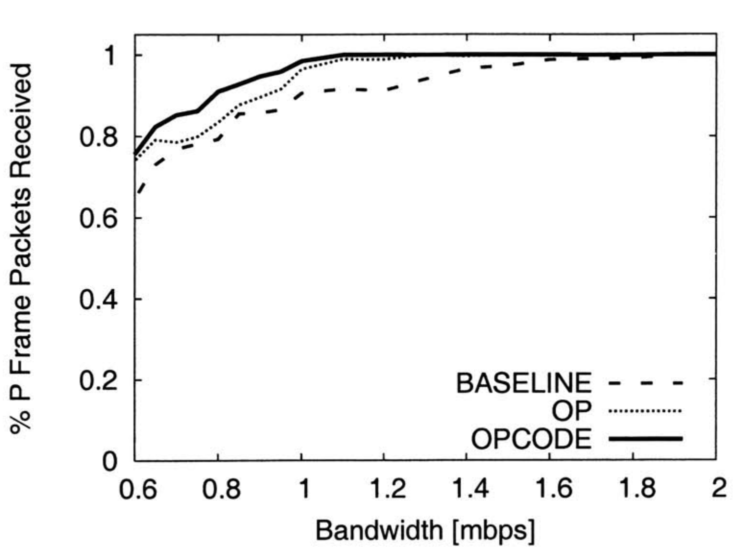

Figure 7-5—Percentage of P Frame Packets Received in 2 Client Experiment.

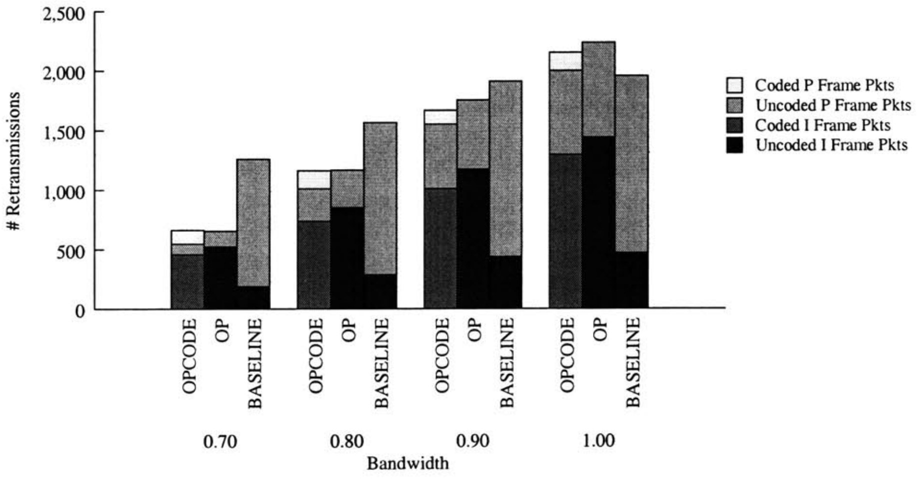

Figure 7-6—Retransmissions in 2 Client Experiment.

PSNR values at any bitrate.

In the evaluation of MORE-M, the use of coding and wireless opportunism resulted in up to a 200% throughput gain over a protocol that used only wireless opportunism. Here we see a minor increase in throughput due to coding. Most of the performance gain is due to the use of wireless opportunism. To partially explain this, recall that MORE-M is a reliable file sharing protocol and that all transmissions driven by the source are coded. The use of random linear codes thus leads to more potential for coding gain than in OPCODE. In OPCODE, for example, all transmissions of I frame packets are wasted if the client does not obtain code vectors that achieve full rank, because OPCODE does not

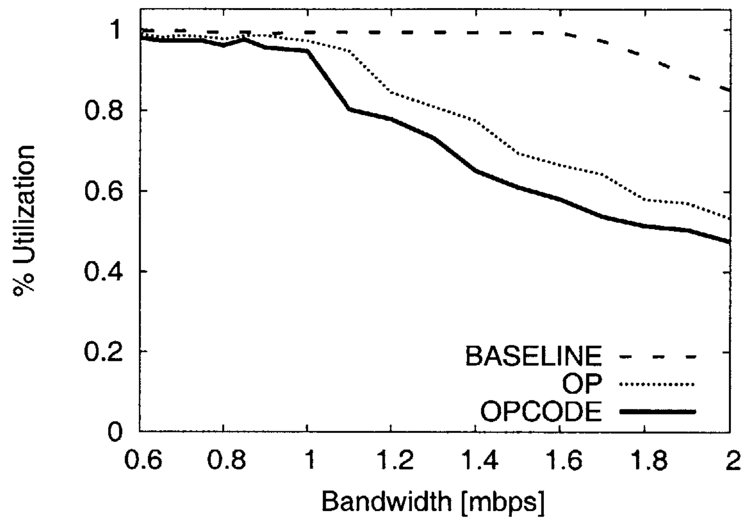

Figure 7-7—Utilization in 2 Client Experiment.

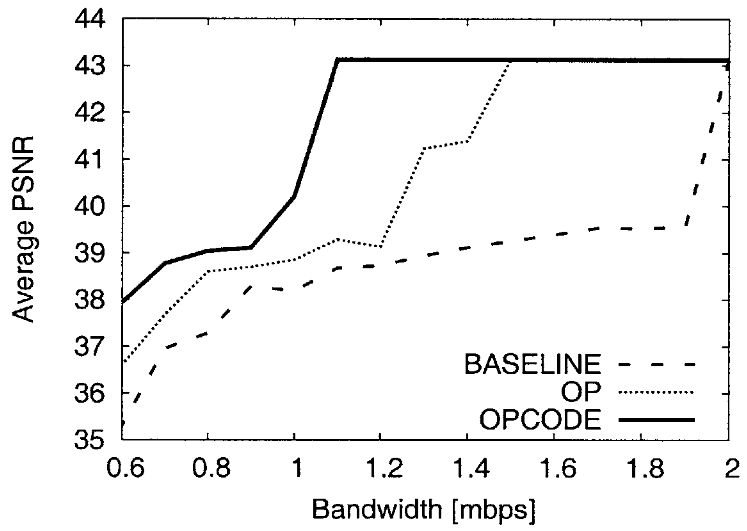

Figure 7-8—Average PSNR for 2 Client Experiment.

guarantee reliability. Additionally, the uncoded P frame retransmissions in OPCODE are less likely to be innovative to multiple destinations than the coded packets of MORE-M, so the coding gain will naturally be smaller. Regarding reliability, in our experiments we set n to high values to allow for many retransmissions, however the variable streaming bitrate in the initial GOPs reduces the number of retransmissions that can be performed when the bandwidth is limited.

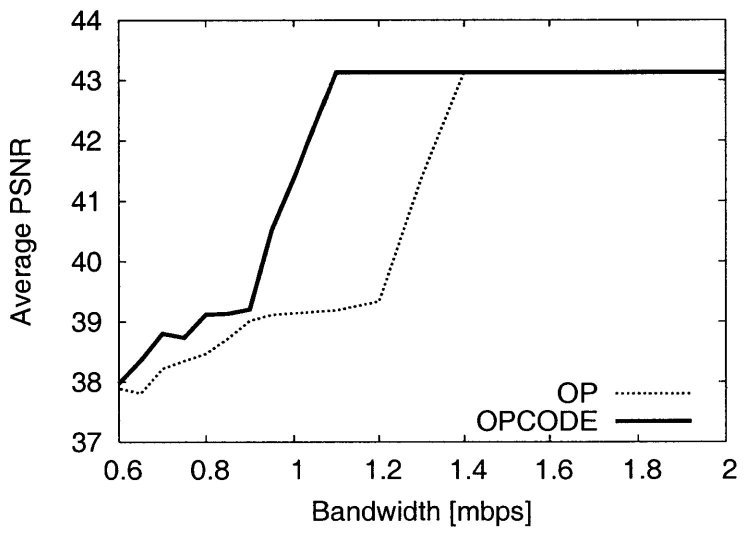

Figure 7-9—Average PSNR for 6 Client Experiment.

Chapter 8

Conclusion

This thesis explored the application of network coding and opportunistic routing to improve the throughput of wireless multicast. We first considered a multicast enabled version of MORE [9], which we called MORE-M, designed for file sharing. We evaluated MORE-M's performance for multicast flows in comparison to other routing protocols in a 20-node indoor wireless testbed. In these experiments, MORE-M attained significantly higher throughput than the other multicast protocols that we considered, particularly as we increased the number of nodes in the multicast group.

We then considered problem of streaming video in a wireless local area network with the goal of supporting video conferencing applications. A network coding based design that utilized opportunistic receptions was presented and evaluated. We showed that the use of coding modestly improved the quality of the video stream, particularly when links experienced higher loss rates and less correlation among losses, and that most of the performance increase seen was due to the use of wireless opportunism. The most likely cause of the slight gain seen with coding is the fact that our protocol did not use coding to achieve reliability.

Bibliography

[1] ISO/IEC 14496. Coding of audio-visual objects. In *International Organization for Standardization (ISO)*, 1999.

[2] Daniel Aguayo, John Bicket, Sanjit Biswas, Glenn Judd, and Robert Morris. Link-level measurements from an 802.11b mesh network. In *SIGCOMM*, 2004.

[3] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung. Network Information Flow. In *IEEE Trans. on Information Theory*, Jul 2000.

[4] John Apostolopoulos, Wai tian Tan, and Susie Wee. Video streaming: Concepts, algorithms, and systems, 2002.

[5] Bay area wireless user group. http://www.bawug.org.

[6] John Bicket, Daniel Aguayo, Sanjit Biswas, and Robert Morris. Architecture and evaluation of an unplanned 802.11b mesh network. In *MOBICOM*, 2005.

[7] Sanjit Biswas and Robert Morris. Opportunistic routing in multi-hop wireless networks. In *SIGCOMM*, 2005.

[8] N. Cai and R. W. Yeung. Secure Network Coding. In *ISIT*, 2002.

[9] S. Chachulski, M. Jennings, S. Katti, and D. Katabi. Trading Structure for Randomness in Wireless Opportunistic Routing. In *SIGCOMM*, 2007.

[10] Minghua Chen and Avideh Zakhor. Rate control for streaming video over wireless. In *IEEE Infocom*, 2004.

[11] P. A. Chou and Z Miao. Rate-distortion optimized streaming of packetized medi. In *IEEE Trans. on Multimedia*, 2001.

[12] Nicola Cranley and Mark Davis. Performanc analysis of network-level qos with encoding configurations for unicast video streaming over ieee 802.11 wlan networks, 2005.

[13] Douglas S. J. De Couto, Daniel Aguayo, John Bicket, and Robert Morris. A high-throughput path metric for multi-hop wireless routing. In *MOBICOM*, 2003.

[14] Evalvid. http://www.tkn.tu-berlin.de/research/evalvid/.

[15] ffmpeg. http://ffmpeg.mplayerhq.hu/.

[16] Christos Gkantsidis and Pablo Rodriguez. Network Coding for Large Scale Content Distribution. In *INFOCOM*, 2005.

[17] Gpac. http://gpac.sourceforge.net/.

[18] T. Ho, M. Médard, J. Shi, M. Effros, and D. Karger. On randomized network coding. In *Allerton*, 2003.

[19] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, and L. Tolhuizen. Polynomial time algorithms for multicast network code construction. *IEEE Trans. on Information Theory*, 2003.

[20] Sid Jaggi, Michael Langberg, Sachin Katti, Tracy Ho, Dina Katabi, and Muriel Médard. Resilient Network Coding In The Presence of Byzantine Adversaries. In *INFOCOM*, 2007.

[21] A. Jiang. Network Coding for Joing Storage and Transmission with Minimum Cost. In *ISIT*, 2006.

[22] Abhinav Kamra, Vishal Misra, Jon Feldman, and Dan Rubenstein. Growth Codes: Maximizing Sensor Network Data Persistence. In *SIGCOMM*, 2006.

[23] S. H. Kang and A. Zakhor. Packet scheduling algorithm for wireless video streaming. In *Packet Video*, 2002.

[24] S. Katti, D. Katabi, W. Hu, H. S. Rahul, and M. Médard. The importance of being opportunistic: Practical network coding for wireless environments. In *Allerton*, 2005.

[25] S. Katti, H. Rahul, D. Katabi, W. Hu M. Médard, and J. Crowcroft. XORs in the Air: Practical Wireless Network Coding. In *SIGCOMM*, 2006.

[26] Jirka Klaue, Berthold Rathke, and Adam Wolisz. Evalvid - a framework for video transmission and quality evaluation. In *13th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, 2003.

[27] Ralf Koetter and Muriel Médard. An algebraic approach to network coding. *IEEE/ACM Trans. on Networking*, 2003.

[28] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M. Frans Kaashoek. The click modular router. *ACM Trans. on Computer Systems*, Aug 2000.

[29] S.-J. Lee, M. Gerla, and C.-C. Chiang. On-demand multicast routing protocol in multihop wireless mobile networks. In *Mob. Netw. Appl.*, 2002.

[30] P. Lettieri and M. B. Srivastava. Adaptive frame length control for improving wireless link throughput, range, and energy efficiency. In *IEEE INFOCOM*, 1998.

[31] Q. Li and M. van der Schaar. Providing adaptive qos to layered video over wireless local area networks through real-time retry limit adaptation. In *IEEE Transactions on Multimedia*, 2004.

[32] S.-Y. R. Li, R. W. Yeung, and N. Cai. Linear network coding. *IEEE Trans. on Information Theory*, Feb 2003.

[33] Mei-Hsuan Lu, Peter Steenkiste, and Tsuhan Chen. Video streaming over 802.11 wlan with content-aware adaptive retry. In *IEEE International Conference on Multimedia and Expo (ICME)*, 2005.

[34] D. S. Lun, M. Médard, and R. Koetter. Efficient operation of wireless packet networks using network coding. In *IWCT*, 2005.

[35] Abhik Majumdar, Daniel Sachs, Igor Kozintsev, Kannan Ramchandran, and Minerva Yeung. Multicast and unicast real-time video streaming over wireless lans, 2002.

[36] Allen K. Miu, Hari Balakrishnan, and Can E. Koksal. Improving loss resilience with multi-radio diversity in wireless networks. In *MOBICOM*, 2005.

[37] C. Guiand P. Mohapatra. Scalable multicasting for mobile ad hoc networks. In *IEEE INFOCOM*, 2004.

[38] J. S. Park, M. Gerla, D. S. Lun, Y. Yi, and M. Médard. Codecast: A network-coding based ad hoc multicast protocol. *IEEE Wireless Comm. Magazine*, 2006.

[39] Charles Reis, Ratul Mahajan, Maya Rodrig, David Wetherall, and John Zahorjan. Measurement-based models of delivery and interference. In *SIGCOMM*, 2006.

[40] Sabyasachi Roy, Dimitrios Koutsonikolas, Saumitra Das, and Y. Charlie Hu. High-throughput multicast routing metrics in wireless mesh networks. In *IEEE ICDCS*, 2006.

[41] E. M. Royer and C. E. Perkins. Multicast operation of the ad-hoc on-demand distance vector routing protocol. In *MobiCom*, 1999.

[42] P. M. Ruiz and A. F. Gomez-Skarmeta. Heuristic algorithms for minimum bandwidth consumption multicast routing in wireless mesh networks. In *ADHOC-NOW*, 2005.

[43] Seattle wireless. http://www.seattlewireless.net.

[44] E. Setton, T. Yoo, X. Zhu, A. Goldsmith, and B. Girod. Cross-layer design of ad hoc networks for real-time video streaming. In *IEEE Advances in Wireless Video*, 2005.

[45] Cormac Sreenan, Jyh-Cheng Chen, Prathima Agrawal, and B. Narendran. Delay reducation techniques for playout buffering. In *IEEE Transactions on Multimedia*, 2000.

[46] K. Sripanidkulchai and T. Chen. Network-adaptive video coding and transmission. In *Visual Communications and Image Processing*, 1999.

[47] E. Steinbach, N. Farber, and B. Girod. Adaptive playout for low latency video streaming. In *ICIP*, 2001.

[48] Y. Wang, S. Panwar, S. Lin, and S. Mao. Wireless video transport using path diversity: multiple description vs layered coding. In *IEEE Int. Conf. on Image Proc.*, 2002.

[49] Jorg Widmer and Jean-Yves Le Boudec. Network Coding for Efficient Communication in Extreme Networks. In *SIGCOMM WDTN*, 2005.

[50] J. E. Wieselthier, G. D. Nguyen, and A. Ephremides. Algorithms for energy-efficient multicasting in static ad hoc wireless networks. In *ACM Mobile Networks and Applications (MONET)*, 2001.

[51] Xin Zhao, Chun Tung Chou, Jun Guo, and Sanjay Jha. Protecting multicast sessions in wireless mesh networks. In *IEEE International Conference on Networking*, 2006.

6195 · 37