

VIDEO MULTICAST OVER WIRELESS LOCAL AREA NETWORKS

BY SHIVESH MAKHARIA

A thesis submitted to the
Graduate School—New Brunswick
Rutgers, The State University of New Jersey
in partial fulfillment of the requirements
for the degree of
Master of Science
Graduate Program in Electrical and Computer Engineering

Written under the direction of
Professor D. Raychaudhuri
and approved by

New Brunswick, New Jersey

October, 2007

ABSTRACT OF THE THESIS

Video Multicast over Wireless Local Area Networks

by Shivesh Makharia

Thesis Director: Professor D. Raychaudhuri

Video multicast services over wireless media are expected to grow in importance over the next few years. Multicasting over wireless networks is complicated by the fact that wireless links are error-prone and time varying. In multicast scenarios using the 802.11 wireless LAN protocol, multiple receivers experience widely varying channel conditions and the link layer protocol does not retransmit erroneous or lost packets, potentially resulting in poor video quality. Therefore, it is a key requirement to support quality of service for all the receivers of the multicast video in the desired coverage area while efficiently utilizing the available wireless LAN resources.

In this thesis, we investigate some of the aspects of reliable video multicast over WLANs. We design, implement and evaluate multi-group hybrid ARQ (*MHARQ*), a new and improved adaptive system for reliable video multicast. *MHARQ* combines the advantages of receiver-driven staggered FEC and hybrid ARQ schemes to compensate the large dynamic range of WLAN channels and to achieve high reliability, scalability and wireless bandwidth efficiency for video multicast. The FEC packets generated by a cross-packet FEC code are divided into multiple streams according to the pre-configured overhead and are transmitted in different multiple IP multicast groups. Certain FEC streams are delayed from the original video stream. The receivers dynamically join/leave the FEC multicast groups based on the channel conditions. For

efficient utilization of WLAN bandwidth, FEC data for a multicast group would not be transmitted by the APs in wireless networks if no receiver joins this group. The time shift between the video stream and the FEC streams introduces temporal diversity and compensates for the client join delay and handoff interruption. In addition, when delayed FEC packets are not enough to recover the lost packets, the receivers can send a hybrid ARQ request to the video server. We design a channel estimation algorithm for a receiver to dynamically determine the delayed FEC multicast groups to join and/or send ARQ NACK to request for retransmission.

Using the *ORBIT* radio grid testbed, we have investigated the performance of the proposed *MHARQ* system with various numbers of users per AP and different numbers of APs per video server. It is demonstrated via real system implementation on *ORBIT* that *MHARQ* improves wireless bandwidth efficiency and scalability for reliable video multicast, compared with existing reliable multicast schemes. The experience and insight obtained from implementation are discussed as well.

Acknowledgements

This thesis is a result of a little more than a year's work at Thomson Corporate Research, Princeton and WINLAB, Rutgers University and an acknowledgement to those who have made this work feasible is long overdue.

I would like to thank Dr. Hang Liu, adjunct faculty at WINLAB and Senior Researcher at Thomson for having advised me in this thesis. This work is partly a brain-child of his ideas. He has really shown me a glimpse of "the art to problem solving" and "decision making with design choices". Dr. Hang Liu has influenced me in more ways than one. Apart from intellectual support for this thesis, he also gave me an opportunity to visit Thomson CR facilities in Princeton on various internships. This helped me sustain my stay here at Rutgers University. Thank you, Dr. Hang !

The other most instrumental person in helping me in this work is my advisor and mentor Dr. Dipankar Raychaudhuri. I distinctly remember entering Prof. Raychaudhuri's office at WINLAB in the summer of 2006 and informing him of the opportunity to work on a collaborated project between WINLAB and Thomson CR, and requesting him to advise me for my Master's thesis. Professor Raychaudhuri was very enthusiastic and encouraging about the idea. His constant guidance on how to build a reliable system and ideas on how to test solutions on ORBIT have really helped me all throughout my thesis journey. Thanks Dr. Ray!

Another person who has been very helpful is Dr. Mingquan Wu from Thomson CR. Dr. Mingquan has guided me with a lot of implementation details on this thesis and some of the discussions with him have been very intellectually stimulating.

Lastly, I would like to thank all my friends and colleagues from the ORBIT team, WINLAB and Thomson CR who have very patiently answered any amateur queries I had with my work. Special thanks to Dr. Dekai Li, Mr. Sachin Ganu, Mr. Zibhin

Whu, Ms. Liquao Han and Ms. Shruti Singhal.

Shivesh Makharia

23 August, 2007

Dedication

To

my mother Mrs. Shashi Prabha Makharia who believed in me
my father Mr. Rajendra Makharia who taught me how to use a computer

Table of Contents

Abstract	ii
Acknowledgements	iv
Dedication	vi
List of Tables	x
List of Figures	xi
List of Abbreviations	xiii
1. Introduction	1
1.1. Thesis Organisation	5
2. Related Work and Scope of this Research	6
2.1. Scope of this Work	8
3. Receiver Driven Adaptive FEC and Hybrid ARQ	10
3.1. Receiver Driven Adaptive FEC	10
3.2. Hybrid ARQ	13
4. Multi Group Hybrid Automatic Repeat Request	15
4.1. Overview of MHARQ and Server Architecture	15
4.2. RS FEC Encoding	19
4.3. Media and Time Driven FEC Encoding at the Video server	20
4.4. ARQ Server	21
4.5. IGMP at Access Point	22
4.6. Client Side Architecture	23
4.6.1. Client Proxy	24

4.6.2.	Client Function Modules	25
4.6.3.	Client Data Structures	26
5.	Algorithms and Packet Formats	29
5.1.	Receiver Driven Adaptive FEC Algorithm	29
5.2.	ARQ Suppression Algorithm	31
5.3.	FEC Packet Encoding	32
5.4.	FEC Packet Header	33
5.4.1.	RTP Header in FEC Packet	34
5.4.2.	FEC Header of FEC Packet	34
5.5.	FEC Protection Operation	34
5.6.	ARQ Message Format	36
6.	Experimental Setup and Results	37
6.1.	Experimental Setup	37
6.1.1.	Introduction to ORBIT Facility	37
6.1.2.	Test Setup on the GRID at ORBIT	39
6.2.	Results and Analysis	39
6.2.1.	Visual Results	40
6.2.2.	Real-Time Network Monitor	41
6.2.3.	Group Join/Leave Monitor	42
6.2.4.	Comparison of Different Multicast Error Recovery Schemes	42
6.2.5.	Scalability Tests on ORBIT	46
	Residual Packet Losses	46
	Downlink Overhead	48
	Uplink Overhead	49
	IGMP Join/Leave Overhead	49
	ARQ NACK Overhead	50
6.2.6.	Overall Gain As Experienced by an End User	51

7. Conclusion	53
7.1. Future Work	53
References	55

List of Tables

6.1. System Parameters for a Typical Visual Result Experiment	40
6.2. Multicast Group Overhead for the Different Schemes for Video Delivery	44
6.3. System Parameters for the Scalability Tests on ORBIT	47

List of Figures

3.1. Receiver Driven Forward Error Correction	10
3.2. Staggering of Video and the FEC Stream	11
3.3. Hybrid Automatic Repeat Request System	14
4.1. The MHARQ System in a Typical Deployment Scenario	16
4.2. Visual Representation of FEC/ARQ Multicast Groups	17
4.3. The Server Side Architecture	18
4.4. The Client Side Architecture	24
4.5. Client Function Modules	25
4.6. Client Data Structures	26
5.1. Encoding of the FEC Packets per Block	33
5.2. FEC Packet Format	34
5.3. ARQ Message Format	36
6.1. ORBIT Testbed at WINLAB, Rutgers University	37
6.2. Schematic of ORBIT Radio Node	38
6.3. Typical Top Level Test Setup for Experimentation on ORBIT	39
6.4. Video Quality Results on Quick Time Player by Apple Inc.	40
6.5. Real Time Packet Trace Monitor	41
6.6. Multicast Group Join/Leave Monitor	42
6.7. Experiment setup for the comparison of SAFEC, Hybrid ARQ and MHARQ	43
6.8. Overhead Incurred by Client on AP1	44
6.9. Overhead Incurred by Client on AP2	45
6.10. ORBIT Setup on 400 Node Grid	46
6.11. Packet Loss (a) Without MHARQ and (b) Residual Packet Loss with MHARQ used	47

6.12. Overhead incurred to Join Additional Multicast Groups	48
6.13. IGMP Join/Leave Overhead per Block	49
6.14. ARQ Supression Results	50
6.15. Comparison of Video Quality using MHARQ	51
6.16. Overhead incurred Using MHARQ	51

List of Abbreviations

ACK	Acknowledgement
NACK	Negative Acknowledgement
AP	Access Point
WLANs	Wireless Local Area Networks
FEC	Forward Error Correction
ARQ	Automatic Repeat Request
hARQ	Hybrid Automatic Repeat Request
MHARQ	Multi Group Hybrid Automatic Repeat Request
SAFEC	Staggered Adaptive FEC Scheme
ORBIT	Open-Access Research Testbed
MAC	Medium Access Control
RTP	Real Time Protocol
IGMP	Internet Group Management Protocol

Chapter 1

Introduction

With the advent of IEEE 802.11 technologies, supporting multimedia applications over wireless local area networks (WLANs) has been a topic of research for quite some time [4] [17] [16]. Although most of research focuses on unicast, interests and demand for multicast applications are growing recently. Video multicasting over WLANs enables the distribution of live or pre-recorded programs to many receivers efficiently. An example application is to redistribute TV programs or location-specific information at hot spots such as airports. Users can watch their favorite TV programs on their mobile devices while browsing the Internet. For enterprise applications, an example is to multicast video of a lecture or training session over WLANs. Other examples include movie previews outside cinemas, replay of the most important scenes in a football match, advertisement in shopping malls, etc. However, wireless video multicast faces several challenges, some of which include:

- Digital video delivery requires high reliability and bounded delay. However the underlying wireless channel is error prone and time-varying due to fading, channel interference, mobility, etc.
- For multicast, the 802.11 link layer does not perform retransmission of lost packets and link rate adaptation. A data frame is discarded at the receiving MAC in the event of an uncorrectable error. Hence users with poor channel conditions may experience very high packet loss rates.
- In video multicast, individual users may experience heterogeneous channel conditions at the same time due to their different locations and users may join or leave the service during a session so that the user topology can change in time.

- Scalability in terms of the increasing number of APs and the increasing number of users may be an issue in video multicast over WLANs.

Therefore it is a key and challenging task to support quality of services (QoS) for all the receivers of the multicast video in the desired serving area while efficiently utilizing the available wireless LAN resources. As mentioned before, no link layer retransmission or link rate adaptation is performed for multicast/broadcast in current 802.11 standards and implementation. This motivates the design of transport layer or application layer mechanisms for improving the quality of delivered video in multicast 802.11 WLAN scenarios.

Application layer Forward Error Correction (FEC) [1], Automatic Repeat Request (ARQ) [11] and Hybrid Automatic Repeat Request (hARQ) [5] are techniques to reduce channel errors and improve receiver video quality.

- Application layer FEC - Application layer FEC mechanism works on packet level such as RTP or UDP packets. Different from physical layer FEC, it applies FEC codes across multiple data packets to generate redundant parity packets on the sender/server side. The sender sends out both data packets and FEC packets to the target receivers. On receiver side, the FEC decoder tries to reconstruct missing packets by using received data packets and FEC packets if some packets were lost during transmission. The reason for encoding across multiple packets is that if redundancy is added within a single packet at the application layer, the erroneous packet is discarded by the receiving MAC and will not be available for error correction at the application layer. Application layer FEC provides a scalable way to protect packets from the noticeable losses at multiple receivers. Static FEC has a low latency and constant overhead regardless of channel quality. But static FEC cannot adapt to the time varying channel conditions and user topology. Generally a strong FEC has to be used for the worst case scenarios all the time, which may result in a high overhead. The system reliability fails as the channel loss is over FEC capability. FEC code rate can be adapted based on channel conditions. However, the challenge is how to estimate the channel conditions

of multiple receivers in multicast and adapt the FEC code to the varying receiver topology and the varying channel conditions of multiple receivers efficiently and reliably.

- ARQ - In this case, the receivers notify the sender if packets are not received correctly and the sender retransmits those packets. ARQ naturally adapts to the channel errors of a receiver and can provide a high reliability under different channel quality. In unicast, it is more efficient than pure FEC in terms of bandwidth utilization. However, the tradeoff is longer delay due to packet retransmission. One approach is to limit the maximum number of retransmissions to meet the delay constraint (best effort delivery model in the given time constraint). For multicast, ARQ is not that efficient and it does not scale well. If different receivers lost different packets in multicast, the sender needs to send all the lost packets. Furthermore all the receivers may send ACK/NACKs to the sender, which cause NACK implosion problem.
- Hybrid ARQ - Hybrid ARQ combines FEC and ARQ and can offer better performance than either scheme alone for multicast in time varying wireless links. In this approach, the sender transmits additional FEC redundant packets to the receivers when the receivers notify the sender that certain packets have not been received correctly. In multicast, different receivers can use parity packets to recover different packet losses. As long as a receiver receives enough number of data packets and parity packets, it can reconstruct the lost packets. Hybrid ARQ is more efficient than pure ARQ. However when different receivers in multicast experience heterogeneous channel conditions, they require different number of retransmitted packets. If retransmissions are simply based on the worse receiver, the receivers with good channel conditions receive unnecessary retransmitted packets, which waste network resource. Furthermore NACK implosion may also result in scalability issue. It is a challenging task to design an efficient multicast retransmission algorithm which can achieve high reliability and scalability.

In this thesis, we investigate video multicast over WLANs. We design, implement and evaluate multi-group hybrid ARQ (*MHARQ*), a new and improved adaptive system for reliable video multicast. *MHARQ* combines the advantages of receiver-driven staggered adaptive FEC and hybrid ARQ schemes to compensate the large dynamic range of WLAN channels and to achieve high reliability, scalability and wireless bandwidth efficiency for video multicast over WLANs. The FEC packets generated by a cross-packet FEC code are divided into multiple streams according to the pre-configured overhead and are transmitted in multiple different IP multicast groups. Certain FEC multicast streams are delayed from the original video stream. The receivers dynamically join/leave the FEC multicast groups based on the channel conditions by sending request to the WLAN APs. For efficient utilization of WLAN bandwidth, FEC data for a multicast group would not be transmitted by the APs in wireless networks if no receiver joins this group. The time shift between the video stream and the FEC streams not only introduce temporal diversity but also compensate client join delay and handoff interruption. In addition, when delayed FEC packets are not enough to recover the lost packets, the receivers can send a hybrid ARQ request to the video server. The video server retransmits the extra FEC packets and/or missing video packets in another multicast group based on the ARQ requests from multiple receivers. The *MHARQ* adapts to the varying receiver topology and the varying channel conditions of multiple receivers by sending FEC data on-demand, and offers flexibility to support different deployment scenarios with various numbers of users per AP and various numbers of APs per server. It can optimize the performance by configuring the system operation parameters according to the application scenario. Furthermore, it can fall back to either the receiver driven adaptive FEC solution or the hybrid ARQ solution by simply modifying the configuration files based on the deployment scenarios. Furthermore we design a channel estimation algorithm for a receiver to dynamically determine the delayed FEC multicast groups to join and/or send ARQ NACK to request for retransmission. A NACK suppression scheme is also developed to reduce the feedback overhead. In our design, even if multiple video packets are completely lost in a burst such as handoffs, the lost packets can be recovered from the corresponding FEC parity packets alone.

We have implemented the *MHARQ* video multicast system including the streaming server and client proxy. Flexible software architecture is designed to integrate the error protection functionality in the clients with no requirement for changing the existing video player. The video and FEC streams are transmitted in a backward compatible manner. Using the *ORBIT* testbed, we investigated the the performance of the proposed *MHARQ* system under various number of users per AP and different numbers of APs per video server. It is demonstrated in real system implementation that *MHARQ* improves wireless bandwidth efficiency and scalability for reliable video multicast, compared with existing reliable multicast schemes. The experience and insight obtained from implementation are discussed as well.

1.1 Thesis Organisation

The rest of this thesis is organized as follows. Chapter 2 surveys related work and discuss our contributions. In chapter 3, we discuss our previous two implementations for reliable video multicast, the staggered adaptive FEC scheme and the hybrid ARQ scheme. In Chapter 4 describe the proposed *MHARQ* system and the software architecture of server and client. Chapter 5 presents the the implementation details of client and server algorithms. In chapter 6, we discuss the *ORBIT* and present our experimental results performed on *ORBIT* testbed . Finally, in chapter 7, we conclude by making inferences from the results and also throw some light on possible future work.

Chapter 2

Related Work and Scope of this Research

In recent years, plentiful work has been done on reliable video streaming over wired Internet and wireless networks. While a lot of research focuses on unicast, reliable video multicast using various adaptive FEC, hybrid ARQ and scalable video algorithms have been proposed.

- M. van der Schaar et al. [15] proposed a cross-layer protection strategy for video unicast over WLAN by adapting the number of MAC retransmissions, packet size and application layer FEC. They didn't adapt PHY modulation and channel coding.
- P. Chou [1] and W. Tan [13] proposed a receiver-driven adaptive FEC system for layered video multicast in wired Internet, in which layered video data for the same group of pictures are transmitted with different multicast addresses in the same time slot and multiple FEC layers are transmitted with other multicast addresses at later time slots. However they focused on theoretical analysis of algorithms to select the video and FEC layers in an Internet setting to optimize the video quality of a receiver given its available bandwidth and packet loss probability. The difference between our scheme and what they proposed include: They didn't consider retransmission from server. The shared nature of the wireless medium was not addressed by them. One client joins a multicast group, the other clients associated with the same AP would receive the data for this multicast group no matter whether it joins this multicast group or not. They did not consider how to reduce IGMP traffic in shared networks such as WLANs. They did not consider burst error recovery, for example, how to recover if all the video packets in a FEC

coding block are lost in an error burst. They also didn't address implementation issues, for example, the backward compatibility with non-FEC capable players.

- Majumdar et al. [10] have proposed an ACK-based hybrid ARQ algorithm for unicast video transmission and progressive video coding with FEC (MDFEC) for multicast video transmission over WLANs. They didn't consider hybrid ARQ for multicast with multiple receivers and NACK suppression. In addition, their studies involved a single access point. Handoffs between access points were not considered.
- Towsley et al. [14] have given a quantitative analysis on performance of hybrid ARQ. However they do not discuss any implementation details for an actual system.
- L. Han [3] in her thesis gives an introduction on the video multicast for wireless channels considering the receiver driven FEC mechanism. They also propose to stagger the different levels of FEC protection in different multicast groups, similar to our work. However, she does not implement an adaptive algorithm at the receiver for joining different FEC groups. Neither is hybrid ARQ-based retransmission from server considered. Nevertheless the work in my thesis is actually a continuation and improvement of her work.
- Simulcasting the content at different rates and qualities in different multicast groups to handle receiver heterogeneity in video multicast over wireless networks has been proposed. The low quality video is transmitted with more robust modulation, channel coding or application layer protection. Receivers subscribe to multicast groups according to their channel conditions. This approach requires a lot more bandwidth. Different receivers does not have unified video quality and the video quality of a receiver may change when its channel conditions vary. In our design, we try to satisfy the video quality for all the receivers in the service area by efficiently adapting the FEC overhead to the varying channel conditions of multiple receivers and receiver topology.

- Scalable video coding and unequal protection have been suggested and extensively studied in [2] to tackle the problem of heterogeneous channel conditions of different end users in a multicast scenario. In this approach, video is coded in multiple layers; there is a base layer and one or more enhancement layers that are transmitted in different multicast group. A base layer is transmitted with more robust modulation, channel coding and/or application layer protection. All receivers subscribe to the base layer to get a certain minimum quality of video and receivers with good channels additionally subscribe to one or more enhancement layers. Although requiring less bandwidth, scalable video requires more complex video codecs and introduces coding overhead. Similar to simulcasting, different receivers does not have unified video quality and the video quality of a receiver may change when its channel conditions vary.
- To overcome packet losses in WLAN video transmission, solutions targeted at MAC layers have been proposed in [8] [9], including the selection of optimal physical layer mode and MAC layer retransmission. However these solutions are more applicable to unicast.

To the best of our knowledge, wireless video multicast applications, especially scalability issue, have not been studied extensively for IEEE 802.11 WLANs.

2.1 Scope of this Work

In the above publications except Han's work [3], only results obtained from simulations were given. Actually although there has been a lot of research and theoretical analysis/simulations on various application layer forward error correction (FEC) and automatic repeat request (ARQ) algorithms to recover from packet loss and to improve transmission reliability in wireless networks, little real implementation (freeware or commercial system) and experiment investigation has been done in this area. Our contributions in this thesis includes:

- Proposing multi-group hybrid ARQ *MHARQ*, a new and improved adaptive system for reliable video multicast over wireless LANs. It combines the advantages

of receiver-driven staggered adaptive FEC and hybrid ARQ schemes. In the design, *MHARQ* considers wireless multicast characteristics. For example, wired backhaul has more bandwidth than wireless links so delayed FEC will always be transmitted in wired backhaul but only transmitted on-demand in wireless networks. Wireless medium is shared, if one user joins a FEC group, other users under the same AP will receive the FEC data no matter they join the multicast group or not. As demonstrated later, *MHARQ* offers flexibility to support different deployment scenarios and can improve scalability and wireless bandwidth efficiency compared with the existing reliable video multicast system.

- Designing flexible client and server software architecture. A client software architecture allows integrating the FEC functionality without changing the existing video player code.
- Solving implementation details and implementing the real system.
- Addressing backward compatibility issues.
- Using experiments to investigate reliability, scalability and bandwidth efficiency of the proposed *MHARQ* and comparing its performance with the existing video multicast system to demonstrate the efficacy of *MHARQ* in a real system.

Chapter 3

Receiver Driven Adaptive FEC and Hybrid ARQ

In this chapter, we describe two related schemes and point out their pros and cons. We also discuss possible improvements by combining the advantages of two schemes, which motivated our *MHARQ* system design.

3.1 Receiver Driven Adaptive FEC

In [3] a receiver driven staggered adaptive FEC scheme (SAFEC) had been proposed. Our work is a continuation and improvement of this scheme. We first briefly describe this scheme and provide solutions which would make it more efficient.

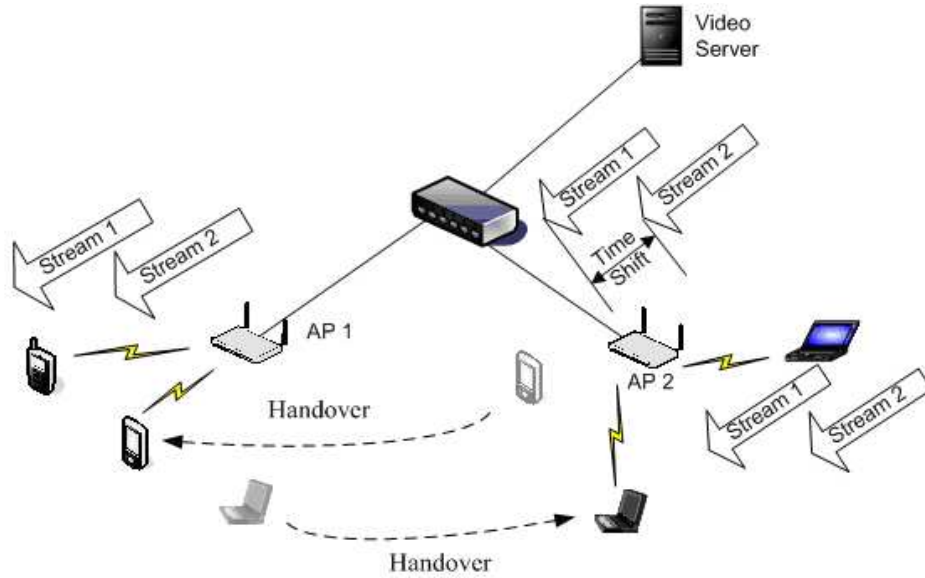


Figure 3.1: Receiver Driven Forward Error Correction

As shown in Figure 3.1 the SAFEC system proposed in [3] consists of a video streaming server that is connected to multiple APs through high speed Ethernet LANs.

In the staggered FEC scheme, cross packet systematic FEC code is applied to video packets to generate parity packets. The original video stream and the additional parity stream are transmitted to all APs in different IP multicast groups with different multicast addresses. The FEC streams are delayed from the original video group, i.e. staggercasting the video stream and parity stream. A receiver dynamically joins the FEC group if it suffers video packet loss. If the join time is less than the time shift between the video stream and FEC stream, the corresponding FEC parity packets will be received to recover the lost packets. After the packet loss is corrected, the receiver may leave the FEC group. Data from the multicast group would not be transmitted by the AP in the wireless network if no receiver joins the group, i.e. FEC packets are sent on demand over the wireless LANs. The normal video stream and the time shifted parity stream provide not only temporal diversity to improve the robustness of video multicast but also compensate the join delay. If a (N, K) Reed Solomon code is applied across K video packets to produce $(N - K)$ parity packets, the N video and parity packets form a FEC code block. As long as at least K packets (no matter video or parity packets) in each block are received correctly, the original K video packets can be recovered.

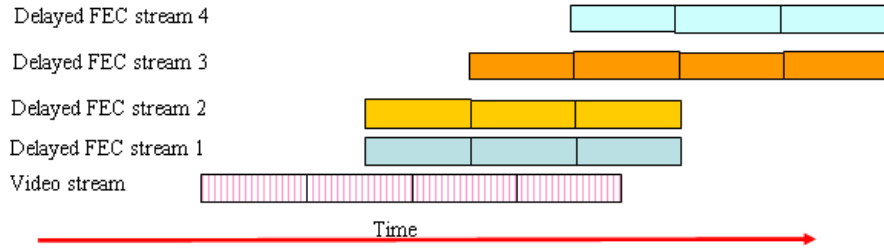


Figure 3.2: Staggering of Video and the FEC Stream

This system proposed is error resilient but not very efficient and scalable.

- First, although multiple FEC groups are proposed, only one FEC group is used in the implementation. If one receiver loses any video packet, it joins the FEC group. All the FEC packets in the FEC group are transmitted over link by the AP until all the receivers under this AP leave the FEC group. In other words, there is no finer granularity on the amount of FEC overhead incurred by the

receivers to receive FEC packets. This scheme can correct burst loss very well, such as during handoffs, but it is not that efficient for random errors. In this thesis, we improve the staggered adaptive FEC schemes by implementing the capability in software to support multiple delayed FEC protection streams in different multicast groups, as shown in Figure 3.1. The number of FEC groups, the amount of FEC and the time delay of each FEC group are configurable. Since each receiver may experience different channel conditions in a wireless scenario, it would be beneficial and flexible to provide more granularity of redundant packet overhead in the different FEC multicast groups. Furthermore we implement an adaptation algorithm at the receiver so the receiver can dynamically choose the FEC multicast groups to join/leave different FEC multicast groups based on its estimated packet loss conditions.

- The system proposed in [3] modifies the video packets format to carry the FEC protection information. The FEC information is carried in the padding field to meet the backward compatibility. It requires the video player to support padding process specified in RFC 2733. In our scheme, we design a new FEC packet format, which carries all the FEC protection information for the receiver to recover the lost packet. In our scheme, the original video packet is not modified at all, so our system is completely backward compatible. Clients which do not understand FEC can just join the video multicast group and receive the original video packets. We'll describe our FEC packet format in Chapter 5.

The staggered adaptive FEC can have good scalability because it does not depend on the number of APs in the network served by a video server. Each AP controls which FEC groups to transmit over wireless links according to the receivers' request. However, it may have FEC overhead granularity issue. It is possible that a receiver will have to join a FEC multicast group even if it does not require all the redundant packets in that group. For example consider a configuration which has redundant FEC packets in each multicast group as: 10% 20% 30% 40%. Assume a packet loss of 32%. In such a case the receiver would have to receive 60% (1st group + 2nd group + 3rd group) overhead

packets to correct its packet losses. Note that in our solution we do not join group 1 + group 3 only because that would incur a lot of feedback traffic (IGMP Join/Leave) every time we vary our overhead requirement according to channel conditions. A simple solution to this approach might be to increase the number of FEC multicast groups with finer granularity on overhead in each group. However, the number of IGMP Join/Leave would drastically increase as the number of multicast groups increase.

3.2 Hybrid ARQ

A hybrid ARQ [8] system consists of an FEC subsystem contained in an ARQ system. Hybrid ARQ is more efficient than FEC and pure ARQ, especially for multicast because each retransmitted FEC parity packet can recover any one of lost source media packets (assuming RS encoding). Figure 3.3 shows the Sender/Server Driven Hybrid ARQ algorithm. The server transmits only the source packets. One or more receivers send a NACK for the missing media packets to recover the lost media after receiving the last packet of a FEC coding block. If the last packet of a coding block is lost, the receiver sends a NACK as soon as it receives a packet for next coding block. The server transmits i parity packets in multicast in an ARQ multicast group, where i is the maximum number of parity packets required by all the receivers under an AP belonging to a particular WLAN. The server delays the retransmission of FEC packets and analyzes the NACKs to find out the maximum number of parity packets required to be retransmitted for a FEC coding block. The server continues transmitting the source/parity packets from other coding blocks during this delay period. If a receiver still needs more packets, it sends a request for retransmission again. The server then transmits more parity packets. This procedure continues until the receiver has recovered all media packets or reaches the deadline of retransmission.

Hybrid ARQ is more bandwidth efficient if there is only one AP. If there are multiple APs, it may have scalability issue. For example, a receiver under AP 1 requires a lot of retransmitted FEC packets, but the receivers under AP 2 only need a small number of retransmitted FEC packets. The video server must retransmit a lot of FEC packets in ARQ multicast group to meet the requirement of AP 1. Then AP 2 also relays

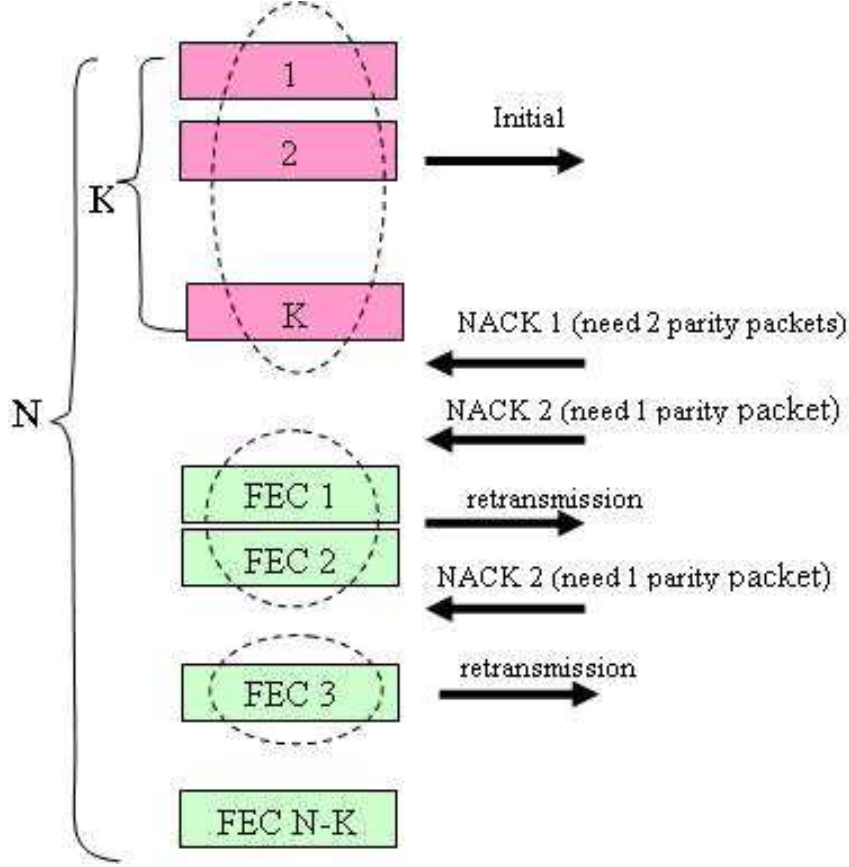


Figure 3.3: Hybrid Automatic Repeat Request System

these FEC packets over its wireless link, which result in the waste of AP 2 wireless resource. This is because the retransmission in hybrid ARQ is controlled by means of a central controller, the video server that serves multiple APs (the transmission of FEC packets in staggered adaptive FEC is controlled by individual AP). Of course, an ARQ proxy can be implemented in each AP. However it introduce complexity in the AP. The AP must be modified to support application layer retransmission functionality. Another method is that each AP has its own ARQ multicast group. However when a receiver hand offs from one AP to another AP, it needs to know the address of the new AP's ARQ group. This requires extra signaling between the receiver and video server or other mechanisms. In addition, the required number of multicast groups and the retransmitted FEC packets increase with the number of APs served by the video server. In this thesis, we propose *MHARQ* to combine the staggered adaptive FEC and hybrid ARQ to achieve better scalability and bandwidth efficiency.

Chapter 4

Multi Group Hybrid Automatic Repeat Request

As seen from the previous chapter, both receiver-driven staggered adaptive FEC (SAFEC) scheme and Hybrid ARQ scheme have their own advantages and disadvantages. In this chapter, we propose to integrate the advantages of the hybrid ARQ scheme and SAFEC. We call our combined solution *MHARQ*. It improves scalability and bandwidth efficiency for reliable video multicast over WLANs. The *MHARQ* is designed to recover random and burst packet loss, and achieve seamless mobile handoff.

MHARQ supports different deployment scenarios with various numbers of users per AP and various numbers of APs per server. It can also optimize the performance by configuring the system operation parameters according to the application scenario. It can also fall back to either the receiver driven adaptive FEC solution or hybrid ARQ solution by simply modifying the configuration files based on the deployment scenarios.

Figure 4.1 shows a wireless LAN video multicast system. The video servers are connected to one or more wireless access points through a high-speed Ethernet LAN. The video server multicasts one or more video programs over the high-speed wired network to the wireless access points. The access points distribute the video to the wireless devices in multicast over the wireless links. The users with the wireless devices can view one or more video programs and simultaneously access the Internet.

4.1 Overview of MHARQ and Server Architecture

With the proposed *MHARQ* scheme, at the streaming server, application layer FEC is applied on a block of source video/audio packets to generate the FEC parity packets. The FEC packets for a source video/audio stream are divided into different layers similar to staggered FEC scheme. Each layer is transmitted in a different multicast group. As

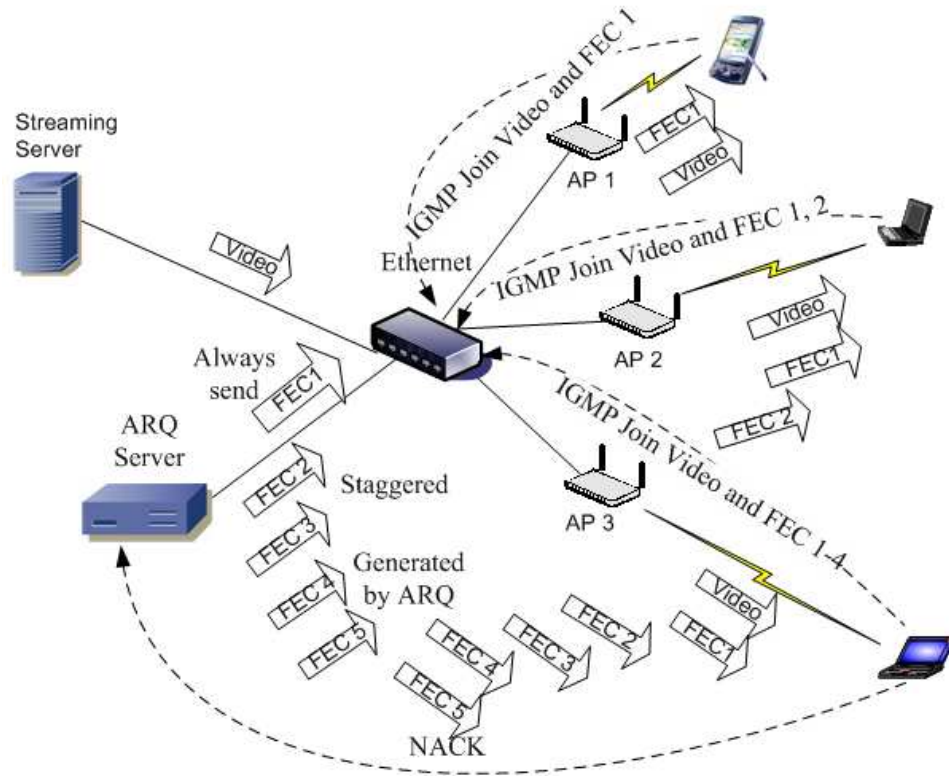


Figure 4.1: The MHARQ System in a Typical Deployment Scenario

shown in Figure 4.2, there are three types of FEC groups for a source stream.

- Non-delayed FEC groups: The FEC packets in these groups are not delayed from the source, i.e. they are sent as soon as they are generated. For random loss recovery, a client joins one or more non-delayed FEC groups based on "long-term" channel conditions. In normal case, the client does not need to join delayed FEC groups or send ARQ (reduce feedback traffic)
- Delayed FEC groups: The FEC packets in delayed FEC groups are transmitted by the server over wired high-speed Ethernet to the APs. The transmission of FEC packets in these groups are delayed from the media packets by a configurable time. If the non-delayed FEC is not enough to recover from packet loss, a client dynamically joins/leaves the delayed FEC groups using IGMP (Internet Group Management Protocol). If the join time is less than the time shift between video

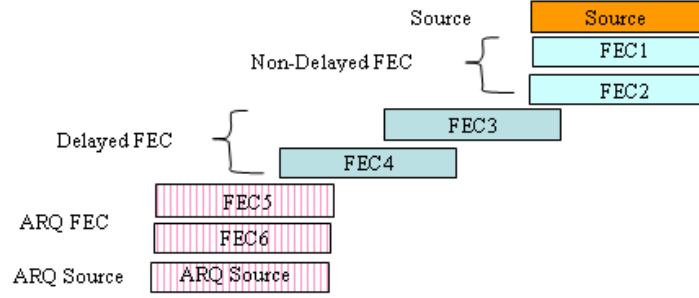


Figure 4.2: Visual Representation of FEC/ARQ Multicast Groups

and FEC, the corresponding FEC parity packets will be received to recover lost video packets. To save wireless bandwidth, FEC data for a multicast group would not be transmitted by the AP/router in wireless network if no clients join this group.

- **ARQ FEC Groups:** The FEC packets in the ARQ FEC groups are sent to the ARQ server. The FEC packets in the ARQ FEC groups are transmitted by the ARQ server according to NACK from the clients. A client sends the NACK to the ARQ server to request retransmission of extra FEC or video packets and joins the corresponding ARQ FEC and retransmitted video groups by sending the IGMP message to the AP for receiving retransmitted packets if the delayed FEC is not enough to recover the loss. In our implementation, the ARQ server can be co-located with the video streaming server or runs on a separate machine.
- **ARQ Retransmitted Video groups:** If the number of lost packets is over a threshold, the ARQ server may retransmit the original block of media packets in a multicast group. The reason is that FEC either recovers all the lost packets or none of packets. When there are a lot of lost media packets, the receiver is at least able to receive some of retransmitted media packets.

Figure 4.3 shows the server side design for our video multicast system using the *MHARQ*. The RTP/UDP/IP protocol stack is used for video multicast. The FEC encoder is placed after the packetization, but before the UDP layer. The compressed video is packetized by the packetizer and the RTP packet header is added. H.264 video

coding is used in our implementation due to its compression efficiency. Systematic FEC codes are applied across video packets at the FEC encoding module to generate parity packets. We use Reed-Solomon (RS) codes constructed with Vandermonde matrix. To make decoding possible at the receiver, a FEC header is added in the FEC packets containing FEC information. In our implementation, FEC coding parameters, including FEC overhead for non-delayed FEC groups, delayed FEC groups, and ARQ FEC group, the time delay for each delayed FEC group can be configured.

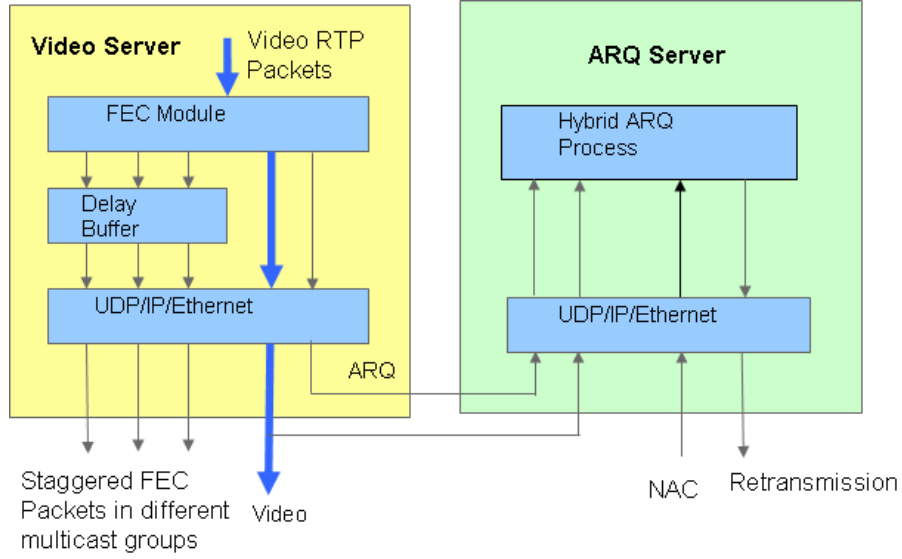


Figure 4.3: The Server Side Architecture

The video packets are transmitted in an IP multicast group (video multicast group) through the UDP/IP stack and Ethernet interface to the APs. Non-delayed FEC parity packets are transmitted in a different multicast group without time shift to correct the random packet loss according to the desired coverage range of the AP. Certain FEC parity packets are stored in the delay buffer for different time offset. Delayed FEC parity packets are transmitted after the delay in multiple multicast groups to correct burst packet loss. FEC encoding, delayed and non-delayed FEC transmission is integrated in the video server. Certain ARQ FEC packets and video packets are also sent to the ARQ server. The ARQ server is responsible for handling NACK request from clients and retransmits the ARQ FEC parity packets and video packets according to the client

request.

4.2 RS FEC Encoding

In the software implementations of RS codecs, for fast encoding and decoding, it is beneficial to choose a code symbol length of 8-bits. This results in an RS code over GF (256) with a block length $N \leq 255$ octets. We use 1-byte code symbol in our implementation. RS codes of shorter block lengths and dimensions can be obtained by puncturing and shortening the mother code with $N = 255$. Our software implementation is based on Vandermonde generator matrix for efficient erasure correction [12]. A RS (N, K) codeword consists of K source symbols and $(N - K)$ parity symbols. During the encoding, the $K \times N$ Vandermonde generator matrix is transformed into its systematic version, where first K columns form an identity matrix, and then the RS codeword is computed by multiplying a vector of K symbols with the systematic generator matrix. Since the code is systematic, the first K coded symbols are exactly the same as the original source symbols. During the decoding, a $K \times K$ submatrix is formed from the K columns of systematic Vandermonde matrix according to the positions of the received K symbols in the codeword. The submatrix is inverted and the original K source symbols are recovered by multiplying the vector of K received symbols with the inverted submatrix.

During the encoding, multiple codewords are computed across the padded packets, each codeword consisting of one symbol from each padded packet. When a RS (N, K) code is used, $(N - K)$ parity packets are generated from K source packets. For block FEC, the FEC encoder waits for enough video/source packets to fill in the coding block, and then generates parity packets. In our implementation, video packets are transmitted out immediately without waiting for filling up the coding block. The FEC encoding module keeps a local copy of these video packets in the coding block buffer. After the coding block is filled, the FEC encoding module generates the parity packets. As described above, the parity packets are divided into multiple non-delayed, delayed, and ARQ FEC groups.

4.3 Media and Time Driven FEC Encoding at the Video server

The FEC parity packets are encoded and transmitted in separate multicast groups. The parity packets for adaptive FEC groups are sent out through the corresponding multicast addresses, and the parity packets for the ARQ multicast group are sent through a unicast address to the ARQ server. An extra header is added to the FEC packets that are sent to the ARQ server so that the ARQ server knows which stream and multicast address the FEC packets belongs to.

Before FEC encode operation, a certain number of media packets (K) have to be buffered (note that as mentioned before, the media packets are sent out immediately and a local copy is buffered to generate FEC packets in our implementation). Different media streams have different bit rates, usually video has a higher bit rate than audio. If we use the same FEC block size for both video and audio we would have audio/video synchronization problems at the receivers.

For example, when the number of video packets in the video buffer is K , FEC encoding is performed on these packets. Then the generated FEC packets are sent out in corresponding multicast groups and the buffer is emptied for the next block. At the same time the number of audio packets in the audio buffer may still have to wait an unspecified time before it reaches K . This will cause synchronization problems as the audio packets will be in the delay buffer for a longer period of time than the video packets. One approach to this problem is to use different block size for both video and audio. However, for a particular block size of video buffer, how to select the block size for the audio buffer is a difficult design problem because the amount of audio data corresponding to a certain video data varies; different content may require different block size pair and even for the same content, different scenarios may have different requirements.

We design a simple solution to solve this problem. The maximum number of packets ($maxK$) that will be used for FEC encoding in a FEC block is set and when the number of video (or audio) packets in the buffer reaches $maxK$, the FEC encoding based on these video (or audio) packets is performed. At the same time, the FEC encoding

based on the corresponding audio (or video) packets is also performed, no matter how many packets are in the audio (or video) buffer. As such, the N and K parameters for Reed-Solomon coding is not static in our implementation. Since both N and K vary for different FEC blocks, the value of N and K for each block must be included in the FEC header for the client to correctly decode the packets.

For time varying bit rate stream, the time to fill a fixed number of media packets in a block may be variant. To further combat the playback jitter that may be incurred because of FEC buffering, a maximum buffer time ($maxT$) maybe set for a FEC block. When the buffering time for the first packet in the FEC buffer reaches $maxT$, the FEC encoding on both video and audio should be performed even if the number of packets in both FEC buffer is smaller than $maxK$.

For real time applications, using variable N and K may cause extra FEC encoding and decoding overhead, because the generating matrix used to construct these codes for different N and K are different. Creating generating matrix is usually very computation expensive. To improve the encoding and decoding efficiency, we also fix both $maxK$ and $maxN$. Now the generating matrix is created based on $maxK$ and $maxN$, and is initialized only once on both server and client side. Based on our encoding scheme, we can guarantee that for all (N, K) codes generated, $K < maxK$ and $N < maxN$, the sub-generating matrix can be obtained using the initial generating matrix based on punctured/shortened codes, which dramatically reduced the encoding and decoding time [12].

4.4 ARQ Server

The ARQ server can be co-located with the streaming server as a separate process or run on a different host. All the media packets and the ARQ FEC packets are sent by the video streaming server to the ARQ server in unicast. We chose to unicast these packets to address scalability concerns. We designed our ARQ server to handle any number of independent program streams. In such a case, opening a port for every stream at the ARQ server to buffer for retransmissions is not a very scalable solution. Hence, to

differentiate packets from one FEC stream from another, we append an extra header on the FEC ARQ packets transferred from the video server to the ARQ server over a single UDP connection. If the ARQ server is collocated with the streaming server, the packets sent to the ARQ server are looped back.

When the ARQ server receives a FEC packet, it extracts the session (program) information from this extra header, strip the extra header from the FEC packet, and store the FEC packet in the buffer cache. These packets are buffered for a constant fixed time which is a design parameter after which these packets are discarded by the ARQ server. This prevents the ARQ server memory to increase indefinitely. On receiving an ARQ request, the ARQ server checks if the program stream for the request is valid and then goes on to transmit these packets from the server in the ARQ multicast group. The server listens for requests on a different multicast group and transmits packets to the clients in a different multicast group. Once an FEC ARQ packet is sent out of the ARQ server, the packet is deleted from the server to avoid resending duplicate FEC packets to the clients.

4.5 IGMP at Access Point

We use IGMP at the access point to efficiently use wireless bandwidth. We illustrate our choice with an example. Assume a content delivery system which does not use IGMP at the APs. There is one video multicast group and assume four staggered FEC multicast groups and one ARQ multicast group. Also assume that the losses in WLAN1 are very low such that the clients need to only join one multicast FEC group. In such a case, even though no client requires FEC packets from multicast FEC groups 2 to 4, they are still present on the wireless channel hogging up bandwidth.

Using IGMP at the APs, we can ensure that packets of a particular FEC multicast group would be available on the wireless channel only if at least one of the clients subscribe to that multicast group. Also, IGMP uses multicast group subscription suppression algorithms to ensure little network uplink overhead.

An alternative approach is that the client sends the IGMP messages to the Ethernet

switch. If no client under an AP to join a multicast group, the Ethernet switch will not send the packets of this multicast group to the AP. If the switch and APs can not understand IGMP signals, we need an IGMP sniffer between the switch and each AP. We have modified an existing package of *Thomson Corporate Research, Princeton* to serve as our IGMP sniffer. The IGMP sniffer intercepts the multicast traffic and if a client joins a multicast group, the multicast packets are transferred to the client. If one or more clients join a multicast group, an entry is added to the EB-Table of the IGMP sniffer, enabling forwarding of the multicast packets in that link. If there are no clients requesting the multicast packets, the entry is deleted from the EB-Table. We have also added the IGMP query at the sniffer to facilitate un-notified leaving of the client. It is important to make sure that the overhead of multicast group join/leave control messages scales well as the number of clients increase. We'll investigate the performance of IGMP in next chapter.

4.6 Client Side Architecture

Available commercial and freeware video players, e.g. *Quicktime*, *Thomson MMAF*, and *VLC* players, don't support FEC. The source code of these players is generally not available. It would be difficult to integrate FEC into every freeware player as well as maintain and update it even if the source code of freeware is available. In our work, we also design novel client proxy architecture as shown in Figure 4.4. It can work with any commercial and freeware video players without making any changes the player code. Our client proxy receives the video and FEC packets from different multicast groups, joins and leaves delayed FEC groups, send ARQ request to the server for retransmission of extra FEC and/or media packet, receives retransmitted packets, recovers the lost video packets and sends the recovered video packets to the player through the loop back interface on the client. The *MHARQ* solution resides on this client proxy. To the player, this information is transparent. It does not have any idea of a proxy residing between the video server and it.

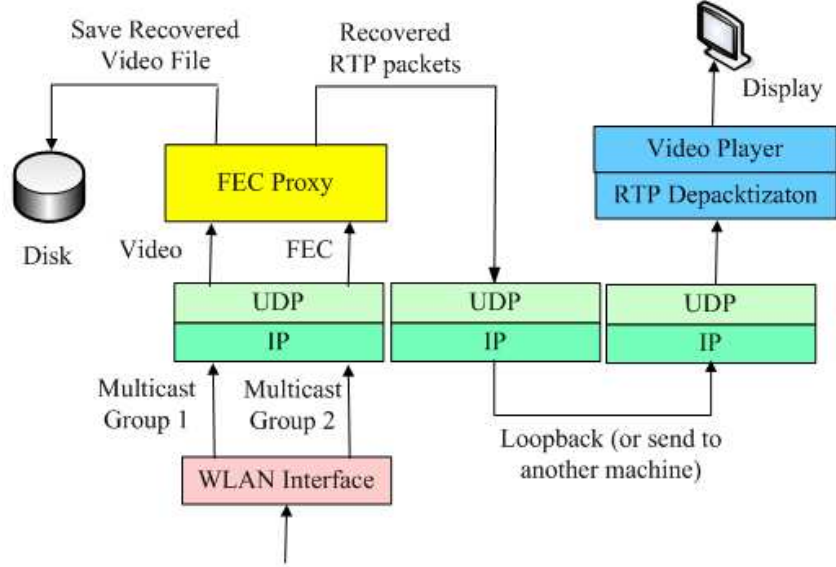


Figure 4.4: The Client Side Architecture

4.6.1 Client Proxy

As described above, a *MHARQ* client is a proxy that receives media and FEC packets from the media streaming server and/or ARQ server, recover the lost media packets, and forward the original and the recovered media packets to the media player. A client can recover lost media packets using different FEC schemes (adaptive FEC, ARQ, or both) according to the configuration. A legacy client that does not support FEC can just join the media multicast group and playback, its function is not affected by the FEC packets. This way, our content distribution system offers a very scalable solution, and also supports backward compatibility.

We describe the architecture of a *MHARQ* client that supports both adaptive FEC and ARQ. Clients will monitor and estimate the channel conditions instantly, and adaptively join/leave delayed FEC groups according to the channel conditions. When a client detects that there is a long burst of packet loss (like just after handoff to another AP), and it can not recover a FEC block even it joins all the delayed FEC groups, it may send a ARQ request to the ARQ server to ask for more FEC packets or/and retransmission of the original media packets. The Adaptive FEC algorithms combined with an ARQ suppression algorithm alleviates the classical problem of feedback explosion inherent in

any ARQ scheme.

4.6.2 Client Function Modules

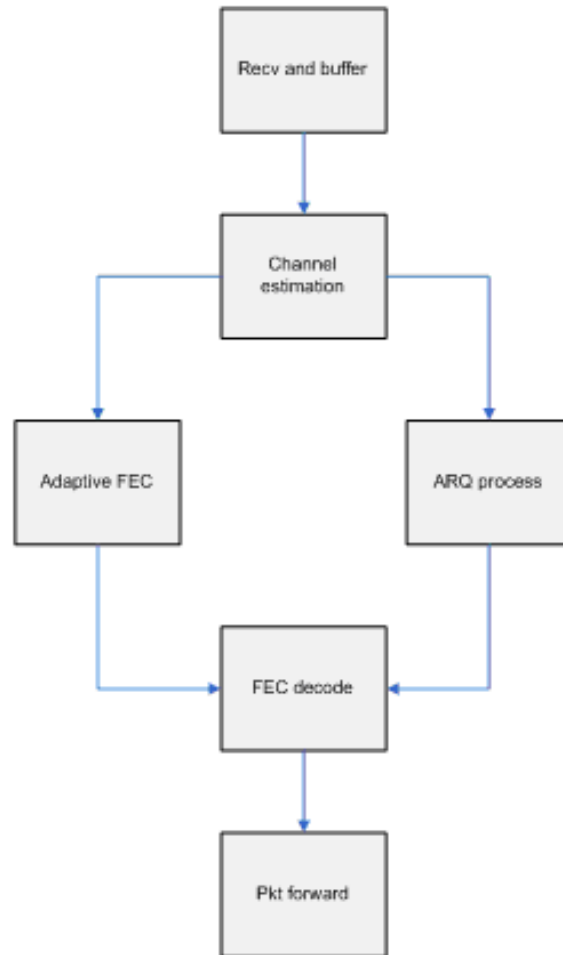


Figure 4.5: Client Function Modules

The functions modules of a client are showed in Figure 4.5. These tasks are performed by three separate threads. The main thread is responsible for:

- Receiving and buffering media/parity packets.
- Receiving and processes ARQ requests from the other clients
- Performs channel estimation, adaptive FEC and FEC decoding

The second and third threads are responsible for ARQ event processing and packet forwarding respectively.

4.6.3 Client Data Structures

In our implementation, the FEC protection information is only contained in the FEC packets. The media packets are not changed for backward compatibility. When a client receives a media packet, if the FEC block data structure this packet belongs to has not been allocated yet, the client has no information about which FEC block that this media packet belongs to. A media packet does not include any FEC information and since we do not use fixed N and K in FEC encoding, no information can not be derived from sequence number of RTP packets either. The clients maintain a source buffer (media buffer) for each track of the media stream in a session which will buffer T seconds of media packets. This buffer is implemented using a linked list. Media packets are inserted into the list according to the sequence number. It should be noted that RTCP control packets for a media track are also inserted into the corresponding linked list. As sequence numbers in the RTCP packets are not assigned according to the RTP packets, RTCP packets are inserted to the source buffer according to their receiving time. To meet the delay requirements, if a packet has stayed in the buffer for T or more seconds, the client will send the packet to the media player no matter whether the FEC decoding has been performed on the packets or not.

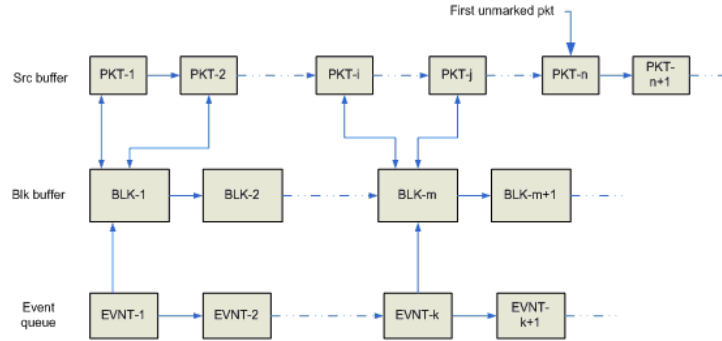


Figure 4.6: Client Data Structures

The client also maintains a linked list of FEC packets. This data structure includes all the information about a FEC block, like the base sequence number, N and K . When a client receives a FEC parity packet, it gets all the information about the FEC block that this parity packet belongs to. The client will first check if a block data structure

has been allocated to the FEC block, if not, a block data structure is allocated. The client will look into the source buffer and mark all the media packets that belongs to this FEC block, and update the information corresponding to this block like how many packets are lost or which packets are lost. The information will be used in the adaptive FEC processing to join delayed FEC groups and ARQ processing for requesting retransmission from the ARQ server. It will also be used in FEC decoding. If a media packet is received after the FEC block this packet belongs to has been allocated, the client would be able to locate this FEC block structure from the link list based on the sequence number of the media packet, the base sequence number and (NK) of the FEC block. The client will then mark this media packet and update the block information. The main data structure at the client side is shown in Figure 4.6.

A FEC block data structure includes an array of pointers that point to all the media packets that belongs to this block. It also includes a decode buffer that is used to store FEC packets for this block. If a FEC block is decodable, FEC decoding is performed and the recovered media packets are inserted into the source buffer. When all the media packets for a FEC block is sent out, the FEC block structure is cleaned and memory is freed.

It is also possible that the parity packets for a certain FEC block are totally lost, for example, in the hand off, and the client may not be able to get any FEC information about these media packets. A client will also keep a pointer to the first unmarked media packet in the source buffer. If the difference between the last unmarked media packet and the first unmarked media packet passes a certain threshold (for example, the $3/2$ the estimated value of K), then a client may send an ARQ request to the server to retransmit the lost packets. Since in our implementation, K is not fixed, a client will keep an estimation on the average value of K and will conclude that all FEC packets for these media packets are lost and activate adaptive FEC algorithm or ARQ request without any FEC information about these media packets.

Each client maintains an event queue data structure. An event can be an ARQ processing event, which includes all the information for a client to send an ARQ request or process ARQ suppression. An event can also be a check event, for an example, after

a client send a ARQ request, it may insert a check event into the event queue for the client to check at a later time to see if it has received the retransmission parity packets and was able to decode a FEC block, if not, it may send another ARQ request.

Chapter 5

Algorithms and Packet Formats

5.1 Receiver Driven Adaptive FEC Algorithm

A multicast session has one media multicast group (in the media streaming server, different media tracks are sent out using the same multicast group address), represented by *media_group*. Each media track in the multicast session also has a set of FEC groups (which may vary from stream to stream). Since we implement the adaptive FEC and ARQ algorithm in each media track separately we will use a multicast session that has only one media track to describe the adaptive FEC and ARQ algorithm.

Assume that a stream has a total of $(m+1)$ FEC groups, $group(0), group(1) \dots group(m)$; $group(0) \dots group(m-1)$ are adaptive FEC groups and $group(m)$ is used for ARQ. The first FEC group is sent immediately after the FEC encoding. A client will always join the media group and the first FEC group. The first non-delayed FEC group is to recover from any packet loss due to random errors which may occur over the wireless channel.

We will use the group number to index the parameters associated to that group. For example, the overhead of $group(i)$ would be $overhead(i)$, and the delay of $group(i)$ would be $delay(i)$. Let the block size of the FEC code be N , the number of media packets in each block would then be:

$$K = \frac{N}{1 + \sum_{i=0}^m overhead(i)} \quad (5.1)$$

The number of parity packets in each group is:

$$num_parity(i) = overhead(i) * K \quad (5.2)$$

A client estimates the average loss rate and its variance for the received FEC blocks, recorded in *avg_loss_rate*, and *avg_variance* respectively.

When a client receives the first FEC packet for a new FEC block, it would usually imply that the client has received (or almost at) the last media packet of the current FEC block. From the FEC header, the client extracts all the information about this FEC block, like the coding parameters N and K . With this information the client has an estimate on how many packets have been lost for this FEC block. This information is used to update the estimation of average loss and variance. At this point, if we have already received enough packets to decode the current FEC block, we do not have to join any new delayed FEC groups or send an ARQ request; however, if we have joined more FEC groups than we require, the client may need to leave those FEC groups. Assume that the client does not receive enough packets to decode the current FEC block. The number of lost media packets is given as:

$$lost_msg = K - recv_msg \quad (5.3)$$

We need to decide the number of FEC groups we need to join. The first approach is to join all the needed groups at one time. The second approach is to join one FEC groups first, after some delay, we then check if we have received enough packets to decode the FEC block, if not, we then join the next FEC group.

For the first approach, the number of packets we are going to need can be estimated as:

$$req_sym = lost_msg * (1 + avg_loss_rate) + \alpha * avg_variance \quad (5.4)$$

In wireless networks, the packet loss variance can be very high, thus in most cases, we might over estimate the parity packets we need. However, there is still a small probability to underestimate the requested symbols, which may result in an undecodable FEC block. The number of groups that we need to join is given as:

$$G = \min \left(j : \sum_{i=1}^j num_parity(i) \geq req_sym \right) \dots j \leq (m - 1) \quad (5.5)$$

When we join a multicast FEC group, we need to specify the duration for which this multicast group needs to be joined. This is specified by the *grp_expire* parameter:

$$grp_expire(i) = current_time + delay(i) + 500ms \dots i \leq (m - 1) \quad (5.6)$$

If the client joins all the adaptive FEC groups but still cannot decode the FEC block, the client will send an ARQ request. To cope with the feedback explosion, an ARQ suppression algorithm is also implemented as described in the next section.

For the second approach, we first join one delayed FEC group and then we set a timer. The timer should expire sometime before the delay time of the next delayed FEC group and if we find out that we can not decode the FEC block, we join the next delayed FEC group. We have tested that the IGMP join latency is about $50ms$ and as such the timer expiration time is specified $100ms$ before the next FEC group delay time:

$$timer_expire(i) = current_time + delay(i + 1) - 100ms \quad (5.7)$$

Again, if the group has already been joined, we just need to update the expiration time of the group, but we still have to set the timer. When the client has joined the last adaptive FEC group but still can not decode the FEC block, the client sends an ARQ request. If the number of lost media packets in a FEC block passed a threshold (for example, 50% of K) it may not be sufficient to send an ARQ to ask only for parity packets. In this case, the client sends an ARQ to ask the retransmission of the original media packets. If we have received enough packets to decode the FEC block, or if we need less parity packets than that provided by the delayed FEC groups we have joined, then we need to check if we should leave one or more multicast groups. Equation 5.5 gives the number of FEC groups that we should join. If the currently joined group number l is bigger than g , then we should leave $groups(g + 1)$ to $group(l)$. For a large number of multicast groups, this can cause a lot of multicast group join/leave overhead. To decrease the number of joins and leaves, we introduce another parameter, let

$$h = \min \left(j : \sum_{i=1}^j overhead(i) \geq avg_loss_rate \right) \quad (5.8)$$

And $t = \max(g, h)$, if $l > t$, then we leave $group(t + 1)$ to $group(l)$.

5.2 ARQ Suppression Algorithm

In a multicast environment, using ARQ/hybrid ARQ, multiple receivers may send NACKs for the same FEC coding block which may result in a NACK implosion. It is

important to suppress the NACKs in such a case. As such, we also implement a NACK suppression algorithm.

After receiving the last packet of a FEC coding block a client delays sending the NACK (assuming it has lost a few packets and needs to send a NACK). This delay is proportional to the number of requested parity packets [14].

The ARQ request is not sent immediately when the client determines it needs to send an ARQ request. The delay for sending ARQ is proportional to the number of requested parity packets. An ARQ request timer is inserted into the event queue. The expiry time of the timer is set between $(K - lost_msg) * T_s$ and $(K - lost_msg + 1) * T_s$. A random value is chosen between this window.

A higher value of *lost_msg* (greater packet loss for a client) would imply a smaller timer value. This way, the client that has lost the most packets will send the ARQ request first. The ARQ request will be multicast to the ARQ request multicast group. Other clients, on receiving this ARQ request, will compare the number of requested parity packets in the received ARQ request to its own requirement. If it is bigger than its own request, the client will suppress its own ARQ request. In most cases it would not be smaller than its own request since otherwise, its timer would have expired before the other clients' timer. In case it is smaller, then that client also sends another request for the remaining number of packets.

5.3 FEC Packet Encoding

In order to prevent error propagation, in our implementation each RTP/UDP/IP packet payload contains a whole video coding unit (a video frame or a slice) so that the packet size varies. To maintain low decoding complexity, it is desirable that the matrix inversion is done only once for each FEC block. Therefore the locations of the received FEC symbols need to be the same for all RS codeword rows in a FEC block. We pad the packets to form each FEC source block.

Before generating FEC packets from a set of media packets, we first find out the length of the largest media packet in a particular set of packets (called a FEC source

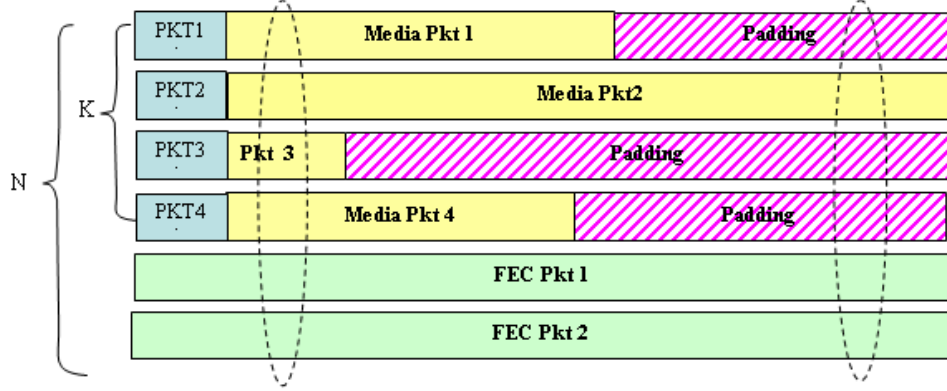


Figure 5.1: Encoding of the FEC Packets per Block

block). Any packets shorter than the largest media packet in that source block is zero padded (see Figure 5.1). This ensures that the length of all the packets in a particular source block is equal. The packet size field and padding bytes are not transmitted because the packet size of a received packet can be obtained from the IP layer of the receiver and padding size can be calculated at the receiver. After FEC encoding, the packet size field and padded bytes are stripped off the source media packet before transmission. When a RS (N, K) code is used, $(N - K)$ parity packets are generated from K source packets.

5.4 FEC Packet Header

A FEC header is added to the FEC packets for the decoding process at the receiver. We now describe our FEC packet header. This FEC packet header is an extension to that defined in *Pro-MPEG Code of Practice #3 Release 2, July 2004(PMCoP2)*. The format of media/source packet is not changed from non-FEC system, i.e. no FEC information is added in the media packets. The FEC-incapable receivers can subscribe to the media multicast group and receive media packets only.

An RTP payload format for FEC parity packets has been defined in the *PMCoP2* to enable error correction of real time media. A FEC header is added in the FEC parity packets between the RTP header and the payload. However the FEC header defined in *PMCoP2* can only support RS code with $(N - K) < 8$ because 3 bits of FEC packet

V=2	P	X	CC	M	Payload type	Sequence Number	
Time Stamp							
Synchronization source (SSRC) identifier							
SN base					Length recovery		
E	PT recovery		Index ext		Reserved		Total_no
TS recovery							
X	D	type	index	Offset		NA	SNBase ext
FEC Payload							

Figure 5.2: FEC Packet Format

index field is used. In our design, the FEC header is extended to support RS code with $(N - K)$ greater than 8 by using some mask bits from the original *PMCoP2* FEC header for extended index field. Before FEC code is applied, the shorter packets are zero-padded so that their length becomes equal to the length of the largest packet in the source coding block(see Figure 5.1).

The FEC packet format is shown in Figure 5.2. Please see *PMCoP2* and *RFC 2733* for more details.

5.4.1 RTP Header in FEC Packet

The RTP header is taken according to the specification of *PMCoP2*. See [9] for details on each field.

5.4.2 FEC Header of FEC Packet

The FEC header is also taken according to the specification of *PMCoP2*. See [9] for details on each field. We also extend the header to suit our implementation.

5.5 FEC Protection Operation

The FEC protection operation involves concatenating specific fields from the RTP header of the media packet, appending the payload, padding with zeroes, and then computing the FEC across the resulting bit strings. The resulting bit string is used to

generate the FEC packet. The following procedure may be followed for the FEC operation. For each media packet to be protected, a bit string is generated by concatenating the following fields together in the order specified:

- Padding Bit (1 bit)
- Extension Bit (1 bit)
- CC bits (4 bits)
- Marker bit (1 bit)
- Payload Type (7 bits)
- Timestamp (32 bits)
- Unsigned network-ordered 16 bit representation of the sum of the lengths (in bytes) of the CSRC List, length of the RTP padding, length of the extension, and length of the media payload (16 bits)
- If media packet CC is nonzero, the CSRC List (variable length)
- If media packet X is 1, the Header Extension (variable length)
- The payload (variable length)
- Padding, if present (variable length)

Note that the Padding Bit (first entry above) forms the most significant bit of the bit string. If the lengths of the bit strings are not equal, each bit string that is shorter than the length of the longest, MUST be padded to the length of the longest. "Zeros" may be used for padding. The pad must be added at the end of the bit string. The FEC parity operation is then applied across the bit strings. The result is the bit string used to build the FEC packet. We call this the FEC bit string.

The first (most significant) bit in the FEC bit string is written into the Padding Bit of the FEC packet. The second bit in the FEC bit string is written into the Extension bit of the FEC packet. The next four bits of the FEC bit string are written into the CC

field of the FEC packet. The next bit of the FEC bit string is written into the marker bit of the FEC packet. The next 7 bits of the FEC bit string are written into the PT recovery field in the FEC packet header. The next 32 bits of the FEC bit string are written into the TS recovery field in the packet header. The next 16 bits are written into the length recovery field in the FEC packet header. The remaining bits are set to be the payload of the FEC packet.

5.6 ARQ Message Format

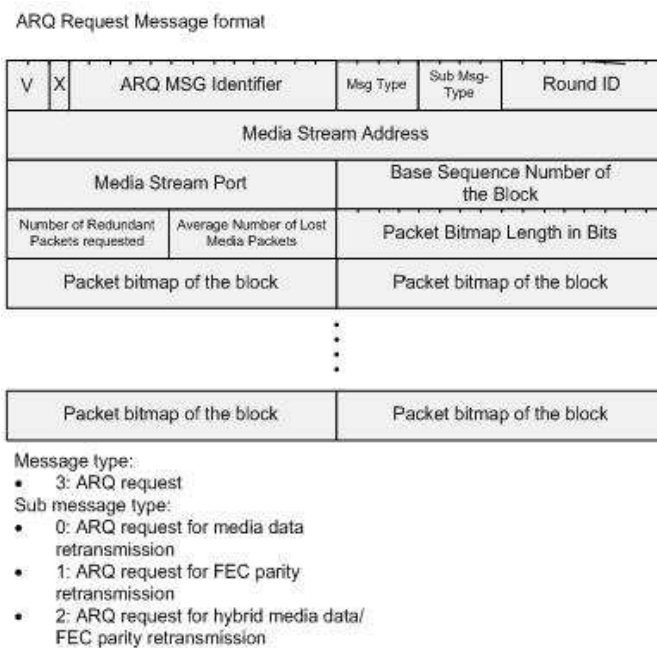


Figure 5.3: ARQ Message Format

The ARQ message format is shown in Figure 5.3. See [9] for details on each field.

Chapter 6

Experimental Setup and Results

6.1 Experimental Setup

After having designed a scalable solution for error resilient video delivery over WLANs, we were motivated to test its effectiveness in a real world scenario. The *ORBIT* (Open-Access Research Testbed for Next-Generation Wireless Networks) developed and maintained by *WINLAB, Rutgers University* provides the necessary experimental platform to conduct experiments typical of a real world setting.



Figure 6.1: ORBIT Testbed at WINLAB, Rutgers University

6.1.1 Introduction to ORBIT Facility

The ORBIT laboratory-based wireless network emulator uses a novel approach involving a large two-dimensional grid of 400 802.11 radio nodes which can be dynamically

interconnected into specified topologies with reproducible wireless channel models. Figure 6.1 shows a snapshot of the main grid at the *ORBIT* facility.

Each *ORBIT* Radio Node is a PC with a 1 GHz VIA C3 processor, 512 MB of RAM, 20 GB of local disk, two 100BaseT Ethernet ports, two 802.11 a/b/g cards and a Chassis Manager to control the node as shown in Figure 6.2. The Chassis Manager has a 10BaseT Ethernet port. The two 100BaseT Ethernet ports are for Data and Control. The Data ports are available to the experimenter. The Control port is used to load and control the *ORBIT* node and collect measurements.

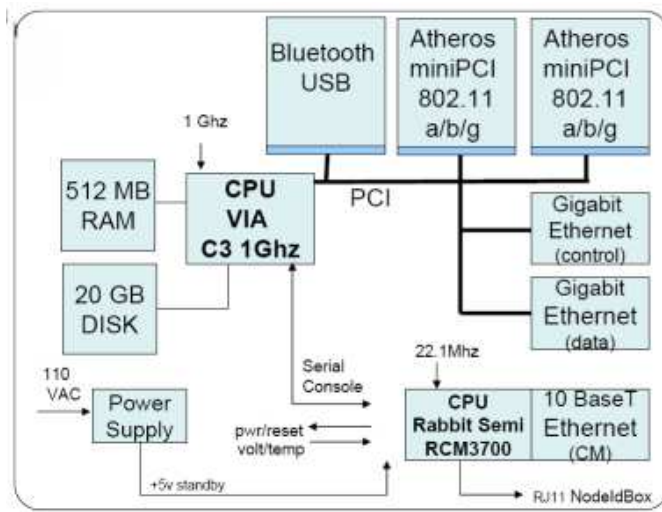


Figure 6.2: Schematic of ORBIT Radio Node

The *ORBIT* support [6]:

- Remote powering on and off of the nodes and permanent check of their status: is a node turned on, what is the processor temperature and similar.
- Imaging of the nodes' hard disks with a custom tailored image before each experiment in a very short time, typically a few minutes.
- Independent traffic monitoring system for post-mortem analysis.
- Controlled radio interference and noise injection to reach targeted SNR on the grid, based on the work presented in [7].
- Real-time visualization of experimental results.

6.1.2 Test Setup on the GRID at ORBIT

To emulate noisy environment we use noise generator to generate additive white Gaussian noise. There are four antennas placed on the four ends of the grid which are used for noise injection. We use a bandwidth of $20MHz$ and vary the noise power to emulate the noisy and lossy environment we need to test for. Figure 6.3 shows a top level design of a typical scenario which we test on *ORBIT*. We use the ARQ server and the Media server on two different nodes. As shown in Figure 6.3, the two nodes are connected via wired network. The two servers are also connected to APs through a wired connection and the IGMP sniffer resides on the node(s) acting as AP.

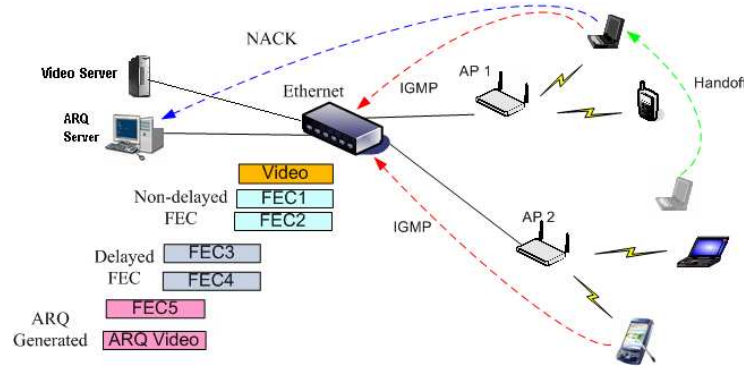


Figure 6.3: Typical Top Level Test Setup for Experimentation on ORBIT

To run our experiments we pick up to 60 nodes to behave as receivers for our video delivery system. It is difficult to go beyond 60 nodes because of some technical issues with the *ORBIT* grid. For the scalability experiments, we randomly pick nodes in increments of 5 or 10 and run our proxy client on them. We output our media packets sequence numbers to a file on each node and write scripts to extract this information from the nodes.

6.2 Results and Analysis

In this section we present the results of some of the experiments carried out with our *MHARQ* system developed. We first discuss the end user experience using our system, then validate the efficiency of our system compared to other traditional methods of

video reception and then discuss the system scalability issues for network using the *MHARQ* solution. Finally we end our discussion with the end user gain with *MHARQ*.

6.2.1 Visual Results

This experiment was carried out at *Thomson Corporate Research, Princeton* using a *LinkSys* Access point operating in 802.11a mode (channel 36). We use a *Microsoft WindowsXP* client using *QuickTime* player to play back the video content. The complete system parameters are described in Table 6.1.

Table 6.1: System Parameters for a Typical Visual Result Experiment

<i>Parameter</i>	<i>Description</i>
(N,K) Reed-Solomon	(255,127)
Viewer	Quick Time Player by Apple Inc.
Video Sequence	Happy Feet
Video Bit Rate	800Kbps
MPEG Encoder	MP4Box
Video Encoding	H.264
Usage Scenario	Typical Office Scenario at Thomson Research Labs, Princeton
Channel Bit Rate	Auto
Packet Error Rate measured	5% per block (Average)

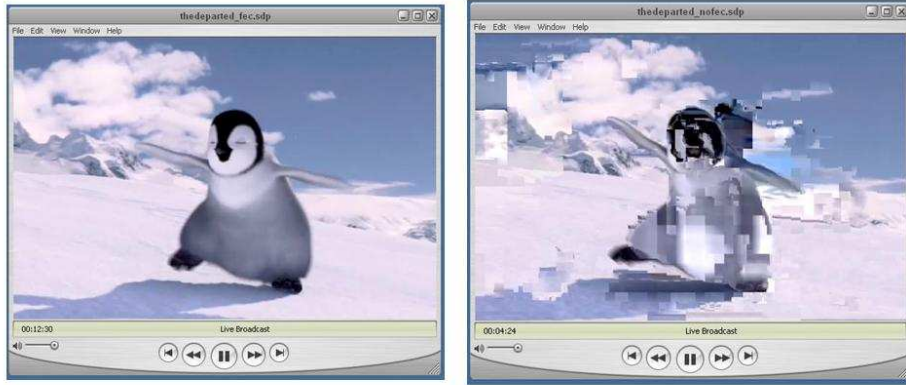


Figure 6.4: Video Quality Results on Quick Time Player by Apple Inc.

Figure 6.4 shows the video quality from the 2 clients. Client 1 (left-hand side) is using our *MHARQ* solution for video reception over the wireless channel. Client 2 (right-hand side) shows the video quality without using our client proxy. Client 2 only

joins the traditional video multicast group while client 1 joins the video multicast group + the FEC/ARQ multicast groups depending upon the link losses experienced by it.

6.2.2 Real-Time Network Monitor

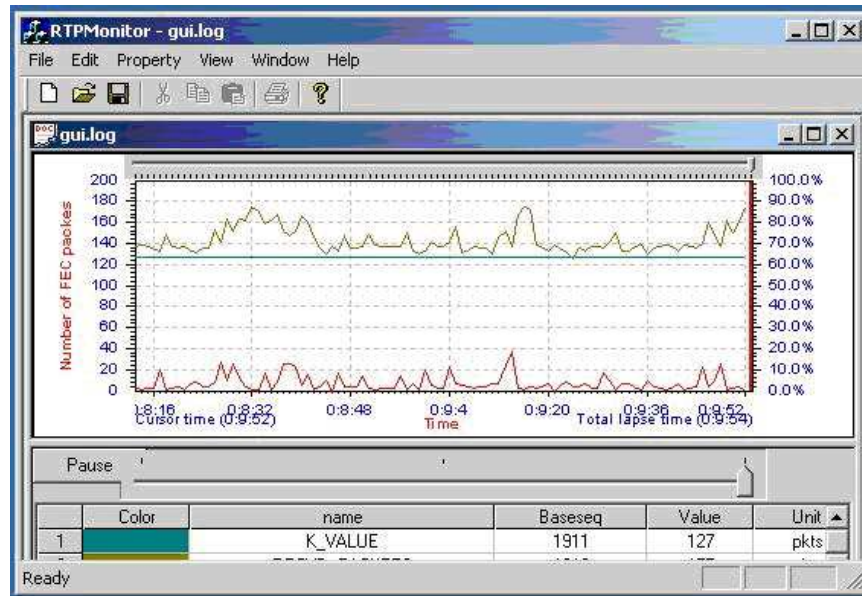


Figure 6.5: Real Time Packet Trace Monitor

We have also developed a few graphical user interfaces (GUI) to help us analyze our network output. This real time monitor in Figure 6.5 shows 3 plots. The bottom most plot (*red*) is that of the losses experienced by a media receiver if no FEC protection (Hybrid ARQ or receiver driven adaptive FEC) is applied to it. The second plot (*green*) is that of the value of the number of media packets chosen for each encoding block. This basically implies the ' K ' for an (N, K) Reed- Solomon code. The top most plot (*yellow*) shows the number of packets received for each block. We can see that the number of packets received for each block closely depends upon the losses experienced per block. This shows that our adaptive FEC algorithm is highly dynamic and aggressive according to channel conditions. Note that if the top most plot falls below the ' K ' plot, it would imply that that block is not decodable. Basically, as long as the total number of packets received per block is greater than ' K ' for each block, we can decode that block. Remember, for a Reed - Solomon code, any ' K ' packets received per block can be used

to recover the lost packets.

6.2.3 Group Join/Leave Monitor

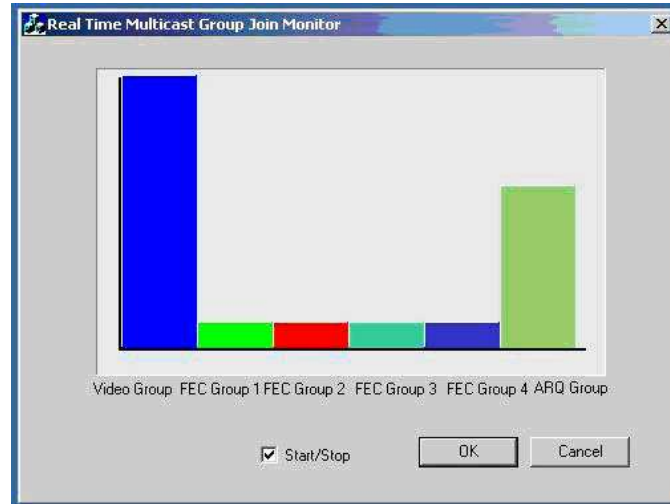


Figure 6.6: Multicast Group Join/Leave Monitor

Another GUI developed is a Multicast Group join/leave Monitor. This GUI is useful to see the number of multicast groups joined by our client proxy in real time. The Y-axis of this GUI shows the overhead associated with each group. Figure 6.6 shows a snapshot of the GUI. The first (blue) bar represents the media group which is always joined. The other 4 bars represent the different FEC groups joined by the client proxy at a particular instant while the last bar indicates the ARQ group joined. In this particular figure, overhead incurred to join each FEC multicast groups was 5% while the maximum overhead incurred to join the ARQ multicast group was 80%

6.2.4 Comparison of Different Multicast Error Recovery Schemes

We have compared 3 error recovery schemes for video delivery:

- Staggered Adaptive Forward Error Correction (SAFEC)
- Hybrid Automatic Repeat Request (HARQ)
- Our proposed scheme, Multi Group Hybrid Automatic Repeat Request (*MHARQ*) or the combined scheme of SAFEC and hARQ

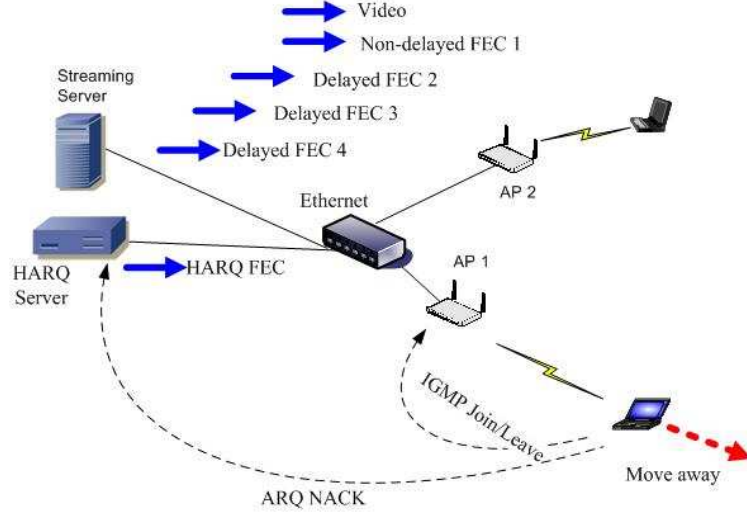


Figure 6.7: Experiment setup for the comparison of SAFEC, Hybrid ARQ and MHARQ

To compare the performance of SAFEC, Hybrid ARQ, and *MHARQ* we set up the experiment on *ORBIT* as shown in Figure 6.7. Assume the Media streaming server and the ARQ server connected through the wired network. Also assume 2 APs (AP1 and AP2) connected to the streaming server and ARQ server over the wired network. The 2 APs represent two different WLANs. AP1 and AP2 work on the two different channels. Client 1 is associated to AP1 and client 2 is associated to AP2. The 2 WLANs share the Video and FEC/ARQ Multicast Group.

We increase the interference of AP1 channel so that the raw packet loss rate of client 1 increases. It is equivalent that the client 1 is moving away from the access point. However, in WLAN2 we keep the packet loss rate of client 2 at around 5%. We use a Gilbert model to emulate such a linear increase in average link loss using packet filtering. Note that we cannot use the *ORBIT* noise generator due to granularity issues to emulate this type of linear increase in packet loss.

Table 6.2 shows the FEC/ARQ multicast group configurations for the 3 different schemes. For the SAFEC scheme we use 5 FEC groups and 1 Video group. The percent overhead incurred to join each group is shown in Table 6.2. The first FEC group in all the 3 schemes is not delayed, while the remaining FEC groups have a time delay of 0.1, 0.2, 0.3, 0.4, 0.5 seconds for FEC group 1, 2, 3, 4, 5 respectively. Also, the first group is

Table 6.2: Multicast Group Overhead for the Different Schemes for Video Delivery

System	FEC 1	FEC 2	FEC 3	FEC 4	FEC 5	ARQ Group
SAFEC	5	5	10	15	65	-
HYBRID	5	-	-	-	-	95
MHARQ	5	5	5	5	-	80

always joined in all the three schemes by the clients to overcome random packet losses.

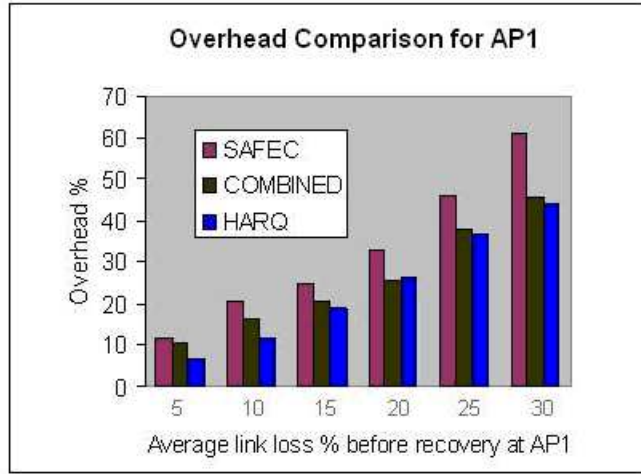


Figure 6.8: Overhead Incurred by Client on AP1

We compare the overhead incurred by the client for the three schemes to recover the lost packets on the wireless channels for AP1 and AP2. Figure 6.8 shows that as the losses increase from 5% to 30% on AP1, the network overhead (only data traffic down-link considered) associated to recover the packets also increases. We define overhead in this context as:

$$Overhead\% = \frac{\text{number of FEC packets received}}{\text{number of media packets received}} \quad (6.1)$$

All of three schemes are able to adapt the FEC overhead to recover the lost packets when the raw packet loss rate changes. Under AP1, the overhead incurred by hybrid ARQ and MHARQ scheme are close. The overhead of SAFEC is higher than hybrid ARQ and MHARQ scheme. This is because the granularity of the FEC in each SAFEC

delayed multicast group is high. With hybrid ARQ the client 1 can request the amount of FEC packets exactly required to recover the lost packet. No additional packets are received. We see that our combined scheme (*MHARQ*) performs close to the HARQ solution.

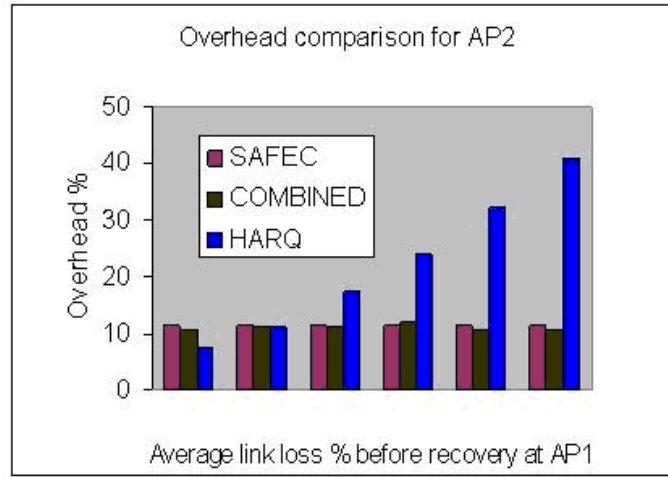


Figure 6.9: Overhead Incurred by Client on AP2

Figure 6.8 would imply that HARQ achieves the lowest overhead and hence should be favorable to use for video multicast over wireless LANs if there is only one AP per video streaming and ARQ server. However, the HARQ solution does not scale well assuming a number of APs each with a different link loss rate. In this scenario for WLAN2, even though the link loss rate is only 5% we see that because of an increase in link losses in WLAN1, the overhead traffic in WLAN2 using HARQ increases (since both WLAN1 and WLAN2 share the Video/FEC/ARQ multicast group). All the clients will receive the extra redundant FEC packets in WLAN2 even though they request a small number of packets. This is because there is only one multicast ARQ group. In these cases, both SAFEC and the *MHARQ* solution scale well since a client in WLAN2 would only join the first FEC group (10% overhead). This is evident from Figure 6.9 where the overhead using SAFEC and *MHARQ* is close to 10% but increased linearly for the case of HARQ. In summary, the *MHARQ* scheme can achieve better performance when there are multiple APs.

6.2.5 Scalability Tests on ORBIT

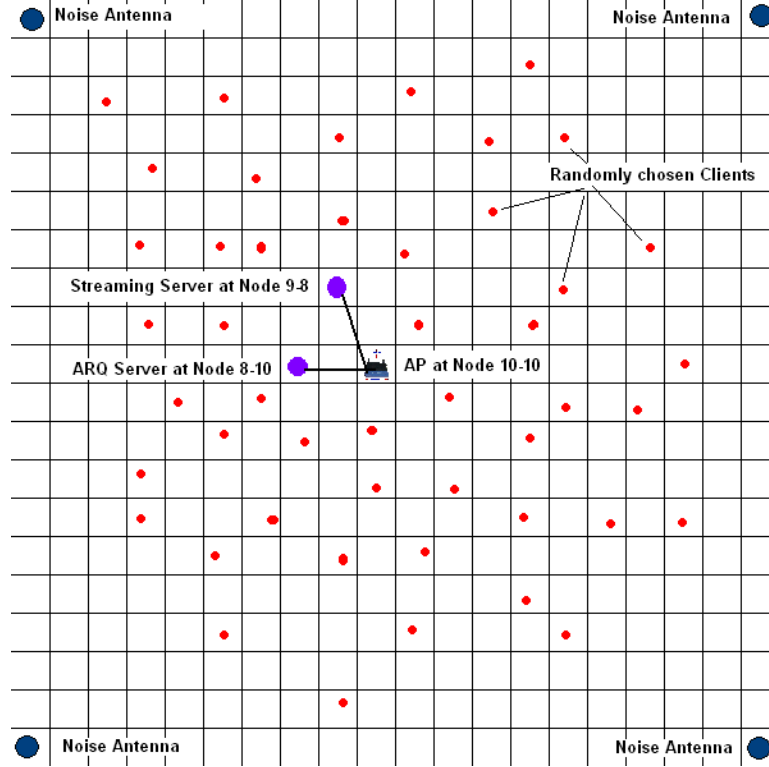


Figure 6.10: ORBIT Setup on 400 Node Grid

We now discuss the scalability issues and results obtained with our solutions. Having implemented a reliable video delivery system, we were also motivated to find out how system and networks parameters would behave under a HOT SPOT environment (typical deployment scenario for our system). In this setup (see Figure 6.10), we randomly choose 60 nodes on the ORBIT grid to behave as clients receiving the video. The AP is placed in the middle of the grid. Noise injection takes place from the four corners of the grid. The noise antennas are placed at the four corners and have the same power. Table 6.3 shows the detailed system configuration for this setup.

Residual Packet Losses

Figure 6.11(a) shows the packet loss experienced at the clients when they do not use any correction schemes. We can see that with an increasing noise level, the packet loss increases at the clients. We do not increase noise beyond -35 dB because after

Table 6.3: System Parameters for the Scalability Tests on ORBIT

<i>Parameter</i>	<i>Description</i>
ESSID	wlancast
Security	Open
Channel	48
Transmission Power	16dBm
Video Source	"Catch Me if You Can"
Video Bit Rate	700 kbps
Noise Injection	AWGN(till 5% loss) and packet filtering (>5% loss rate)
Noise Bandwith	20MHz
Channel Bit Rate	Auto
1st Multicast Group	Video Group
2nd Multicast Group	5% FEC
3rd Multicast Group	5% FEC
4th Multicast Group	5% FEC
5th Multicast Group	5% FEC
6th Multicast Group	ARQ Retransmission Group (max 80% overhead)
7th Multicast Group	ARQ Request Group

this level, certain clients behave very poorly (packet losses in excess of 50% per block) and in such degrading environments, no scheme can be used to recover video quality. This is inherently an issue with the fine calibration of the noise antennas. Instead, to emulate a noise environment with packet losses in excess of (2 3 %) we use a Gilbert model and simulate packet filtering. We drop packets at the receiver to simulate a noisy environment (average 5% packet loss per block)

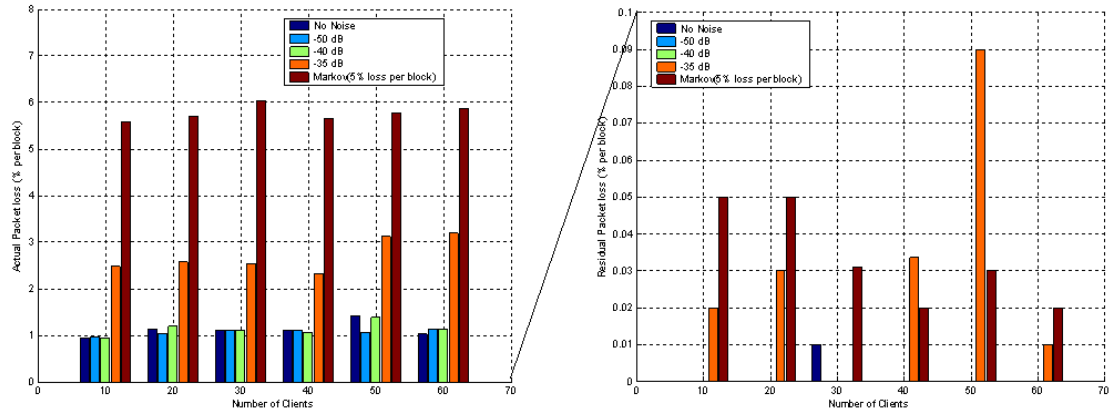


Figure 6.11: Packet Loss (a) Without MHARQ and (b) Residual Packet Loss with MHARQ used

Figure 6.11(b) shows the residual packet loss for the same clients. Note the Y-Axis scale on both the graphs for comparison. We can see that our scheme performs well on an average for around 60 clients. The packet loss is at a maximum of around 0.09% per block which is almost close to no packets lost per block.

Downlink Overhead

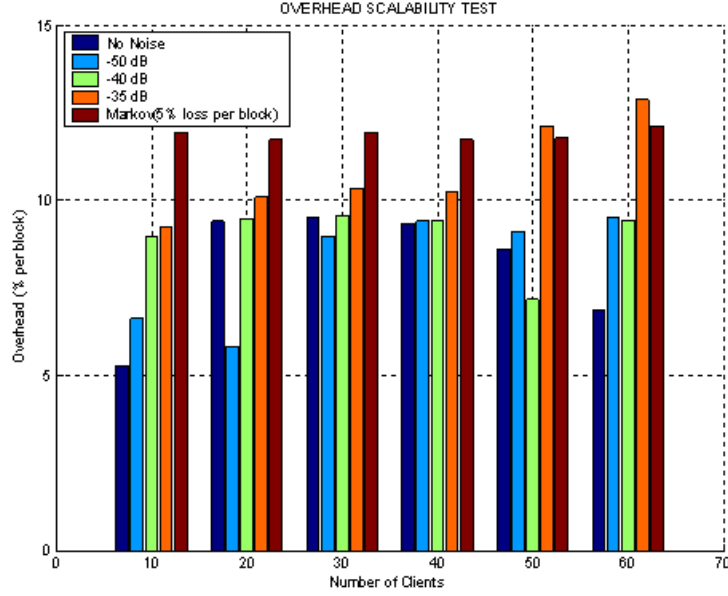


Figure 6.12: Overhead incurred to Join Additional Multicast Groups

Figure 6.12 shows the overhead packets requested by the clients. The actual number of packets received by the clients is lesser than this number because of losses in the FEC groups also. In the no noise case, when the packet loss is around 1%, we see that on average the clients join either 2nd or 3rd multicast groups (total overhead of 5% or 10%). For a packet loss of up to 5-6% (Markov model) we see that the clients at maximum join 2nd, 3rd and 4th multicast groups in some cases (total overhead of 5%, 10% or 15%). The average overall overhead is around 12%. Ideally, with a 5% packet loss we would need a minimum of 7% overhead to correct the losses. This also shows that our receiver driven adaptive algorithm is very aggressive and dynamic. Also, we can see that this solution scales very well for an increasing number of clients.

Uplink Overhead

The above Figures have considered the downlink's scalability when using *MHARQ*. They discuss the data traffic from the server to the clients. It is also important to discuss some scalability results on feedback (uplink traffic).

IGMP Join/Leave Overhead

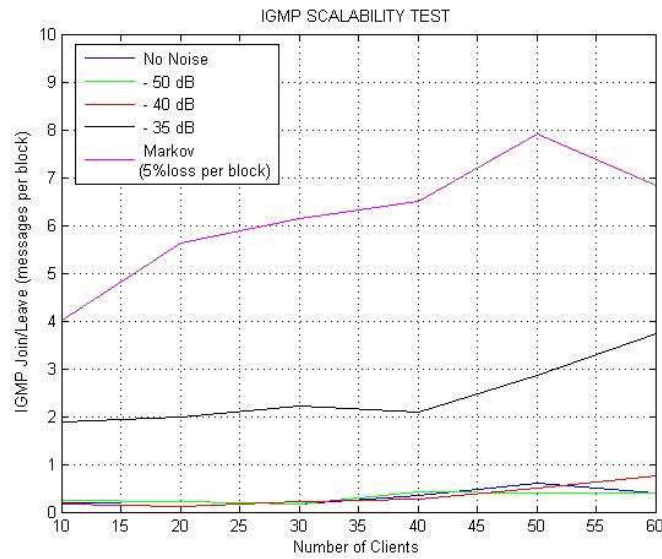


Figure 6.13: IGMP Join/Leave Overhead per Block

We first show how the multicast group subscription overhead behaves. As explained before, we use IGMP for the multicast group subscribe/unsubscribe. To calculate the number of join/leaves we use an IGMP sniffer at the Access Point. We see that (Figure 6.13) with no/very low noise levels, the IGMP overheads are much less than 1 message per block. For around -35 dB, the average number of join/leaves is 3 per block (1 for video, 1 for ARQ request Group and 1 for FEC group). For around 5% packet loss this number scales to around 6-7 messages per block (1 for video, 1 for ARQ request Group, 4 for FEC Groups and 1 for ARQ Retransmission Group). These results imply that the IGMP subscription overhead scales fairly well with around 60 clients deployed.

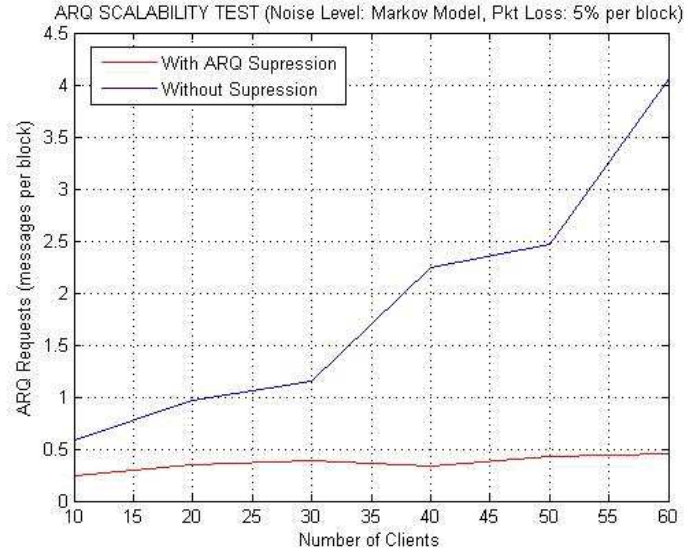


Figure 6.14: ARQ Suppression Results

ARQ NACK Overhead

We also show some results on the ARQ NACK messages send by the clients. We discuss the ARQ NACK message overheads received by the ARQ server for different noise levels. For the Gilbert model (5% packet loss per block) we can see the classical feedback implosion problem shown in Figure 6.14 by the blue plot. The number of ARQ retransmission messages increases drastically as the number of clients increase. However, using our suppression algorithm, we can see that this number is stable (around 0.5 messages per block on average). One important note however in these graphs is that, even though we simulate the losses at around 5%, the actual losses for both these comparisons are not same. Without suppression, the actual packet loss is around 6.5% per block and with suppression the actual packet loss is around 5.8% per block. Since it is very difficult to compare exact packet losses for a wireless environment in the two systems, we have used such an approach. It is important to study the nature of the graphs for these two cases rather than the actual values which may vary slightly because of the difference in packet losses.

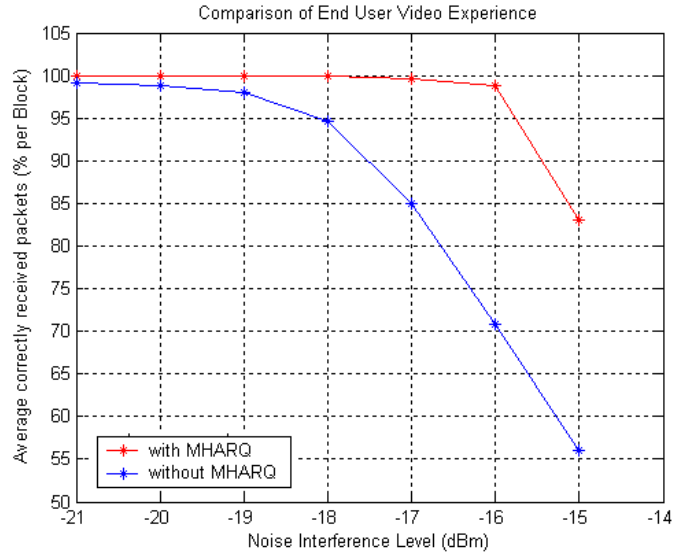


Figure 6.15: Comparison of Video Quality using MHARQ

6.2.6 Overall Gain As Experienced by an End User

In this particular experiment we show the gains achieved through our system as experienced by an end user streaming video over a wireless channel. We have a single user connected to the AP and the video server streaming video to the client associated with the AP. We have our *MHARQ* proxy running on the client.

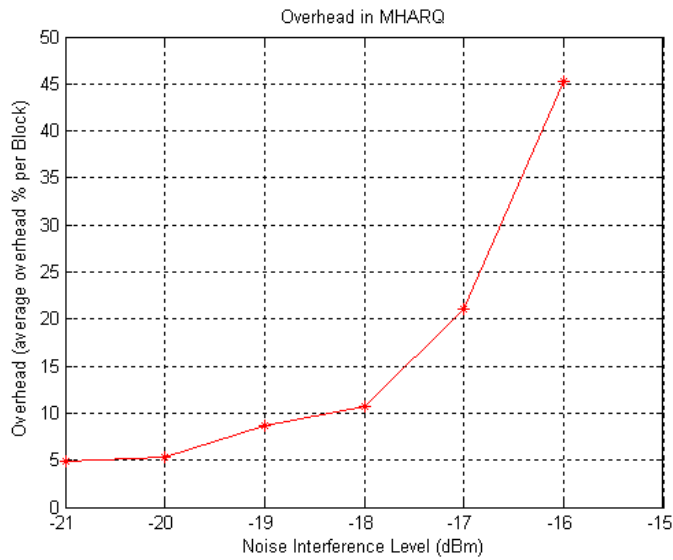


Figure 6.16: Overhead incurred Using MHARQ

Figure 6.15 shows the average correctly received video packets % per FEC block for increasing noise levels. The *blue* plot has no protection and shows that the received video quality degrades as the noise levels increase from -20 dBm to -15dBm. However, using our *MHARQ* solution (*red* plot), the client can receive acceptable video quality upto -16dBm packet loss. However, our system also fails beyond -16dBm. Figure 6.16 shows the overhead incurred by the client to maintain an acceptable video quality for the increasing noise levels.

Chapter 7

Conclusion

In this work, we have proposed and implemented an efficient and an error resilient simple a video distribution system. We have explored the two traditional schemes for video distribution - receiver driven FEC and hybrid ARQ. Our solution combines the advantages of both these schemes with a minimal increase in system complexity. We also propose schemes to reduce the multicast group membership overhead associated when a client joins a FEC/ARQ multicast group. Our system is also backward compatible implying that clients that do not understand FEC can also join the media group and receive video (with a loss in quality since there is no error recovery). Using *ORBIT*, we also investigated the impact of FEC overhead on for a large number of clients. We also show that our system is scalable for a large number of clients in a small area (typical of a WLAN hot spot). The initial results seem favorable and they provide us valuable insight in improving our algorithms and optimizing system operation parameters. The effectiveness of staggered FEC and hybrid ARQ for improving video multicast quality has been demonstrated by real systems on *ORBIT*. We also see that varying the system parameters (amount of FEC overhead in each group) has a considerable effect on the overall network overhead for a large number of clients.

7.1 Future Work

We need to investigate further on how well our system behaves for more than 60 clients. With the advent of HDTV in home multimedia, it is also challenging to find out how very high bit-rate, delay sensitive real time applications can be reliably delivered to mobile clients on a wireless network. We also provide equal FEC protection to all the video frames in our solution. It is well-known that I frames need better protection than

B/P frames in MPEG Video. Hence it may be beneficial to provide more protection to these I-frames rather than B/P frames. Other issues include a detailed video quality assessment as a function of video bit rate and WLAN user parameters like location, service requirement and packet error rate.

References

- [1] P.A. Chou. Fec and pseudo-arq for receiver driven layered multicast of audio and video. *Technical Report MSR-TR-99-86*, 1999.
- [2] Mohammad Mahdi Ghandi. Scalable video coding and transmission. 2005.
- [3] Liqiao Han. Reliable video multicast over 802.11 wlans. *Thesis Submitted to Graduate School, Rutgers University*, 2006.
- [4] Wang Jian-ming. Improving the multimedia traffic performance over wireless lan. *IEEE*, 12:451, 2005.
- [5] Roger G. Kermode. Scoped hybrid automatic repeat request with forward error correction. *Proceedings of the ACM SIGCOMM*, page 278, 1998.
- [6] Haris Kremo. Ieee 802.11 medium access protocol: An experimental case study. *Thesis Submitted to Graduate School, Rutgers University*, 2005.
- [7] J. Lei. Wireless link snr mapping onto an indoor testbed. *First International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities*, 2005.
- [8] Shu Lin. A hybrid arq scheme with parity retransmission for error control of satellite channels. *IEEE Transactions on Communications*, 30, 1982.
- [9] Hang Liu. Design document for mharq solution.
- [10] A. Majumdar. Multicast and unicast real-time video streaming over wireless lans. *IEEE Transactions on Circuits and Systems for Video Technology*, 12.
- [11] Peterson and Davie. Computer networks: A systems approach. 2003.
- [12] Luigi Rizzo. Effective erasure codes for reliable computer communication protocols. *Proceedings of the ACM SIGCOMM*.
- [13] W. Tan. Multicast transmission of scalable video using receiver-driven hierarchical fec. *Visual Communication And Image Processing*, 3:278, 2000.
- [14] Don Towsley. Parity-based loss recovery for reliable multicast transmission. *Technical Report 97-17*, 1997.
- [15] M. van der Schaar. Adaptive cross-layer protection strategies for robust scalable video transmission over 802.11 wlans. *IEEE Journal on Selected Areas in Communications*, 21.
- [16] Hongqiang Zhai. A call admission and rate control scheme for multimedia support over ieee 802.11 wireless lans. 12:451, 2006.

- [17] Qian Zhang. Cross-layer qos support formultimedia delivery over wireless internet. *Journal on Applied Signal Processing*, 2:207, 2005.