4

Smart Network Caches:

Localized Content and Application Negotiated Recovery Mechanisms for Multicast Media Distribution

by
Roger George Kermode

B.E. (hons) Electrical Engineering,
University of Melbourne, November 1989

B.Sc. Computer Science,
University of Melbourne, November 1990

S.M. Media Arts and Sciences
Massachusetts Institute of Technology, June 1994

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
in partial fulfillment of the requirements of the degree of

DOCTOR OF PHILOSOPHY IN MEDIA ARTS AND SCIENCES
at the
Massachusetts Institute of Technology
June 1998

Signature of Author: [illegible]

Program in Media Arts and Sciences
April 13, 1998

Certified By: [illegible]

Dr. Andrew B. Lippman
Associate Director, MIT Media Laboratory
Thesis Supervisor

Accepted By:

Dr. Stephen A. Benton
Chairperson
Departmental Committee on Graduate Students
Program in Media Arts and Sciences

Smart Network Caches:

Localized Content and Application Negotiated Recovery Mechanisms for Multicast Media Distribution

by
Roger George Kermode

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
on April 13, 1998
in partial fulfillment of the requirements of the degree of

DOCTOR OF PHILOSOPHY IN MEDIA ARTS AND SCIENCES

Abstract

In the past fifteen to twenty years, two digital revolutions have been quietly occurring. The first is an effect of Moore's law of computing power; microprocessor power has grown to the point where inexpensive home computers can encode, decode, and display video. This new found ability to cheaply process video heralds the impending realization of *digital convergence* at the source and consumer, the end points of the distribution chain. Between the source and consumer lies the distribution channel. The second revolution—realizing digital convergence in the distribution channel—is only just beginning since the only widespread communication paradigm is that of point-to-point. Point-to-multipoint and multipoint-to-multipoint paradigms are proving to be much more difficult to deploy with the same flexibility, timeliness, and reliability that is associated with the point-to-point paradigm.

In this dissertation, a new model called *content and application negotiated delivery* (CANDY) is developed to realize the point-to-multipoint and multipoint-to-multipoint communication paradigms. Two ideas are combined in this model: (1) every member of the session can be a cache or a server for every other member, and (2) the characteristics of the data and needs of the receiving application are used to assist in the delivery of data between session members. These two ideas have been explored individually at both the packet and stream levels, but they have not been explored simultaneously as the basis for a distribution system.

By developing these ideas together, several new pieces of technology have been invented that advance the state of the art in terms of realizing digital convergence in the communications channel. A new receiver-driven multicast transport protocol is developed for the delivery of individual single streams of packets. This new protocol scales significantly better than existing ones. The protocol is further extended, applying the lessons at the packet level to the stream level, to develop a scalable set of methods that increase the ability for on demand access through "late joins" or catch ups. Finally, a number of integrated caching tools are developed to exploit, to full effect, the power afforded by the new transport protocol.

Thesis Supervisor: Andrew B. Lippman
Title: Associate Director, MIT Media Laboratory

This work was supported by the members of the Singapore Program in Digital Media and Digital Life Consortium, AT&T, and Motorola.

Smart Network Caches:

Localized Content and Application Negotiated Recovery Mechanisms for Multicast Media Distribution

by

Roger George Kermode

The following people served as readers for this dissertation:

Reader: ______________________________

Dr. David D. Clark
Senior Research Scientist
MIT Laboratory for Computer Science

Reader: ______________________________

Mr. Carl Malamud
Visiting Professor
MIT Media Arts and Sciences Program

Acknowledgments

When I first arrived at MIT in September of 1992, I had no idea of what to expect. Never having travelled to the United States before, I had only heard rumors about how I was going to work harder than I had ever worked in my life. The rumors turned out to be true. Fortunately, the huge amounts of effort I expended were more than amply rewarded in what I learned during my stay. This benefit was largely derived from having worked with Andy Lippman, who took every opportunity to challenge my thinking and acquiesced to my numerous requests for travel to conferences and meetings. Andy's keen insight and experience were instrumental in tying together the threads of this dissertation.

Many many thanks must also go to the other members of my committee: Dave Clark and Carl Malamud. Dave Clark's patience and probing questions on the technical aspects of my work helped me resolve countless confusions pertaining to multicast transport. Likewise, Carl Malamud—despite a hectic schedule involving multiple moves around the world to the Far East and Europe—always took the time to review my work and call my attention to questionable claims. His frank assessments and willingness to share real world experiences in writing and the deployment of multicast applications was greatly appreciated.

I would be remiss if I did not thank Nicholas Negroponte for establishing the Media Lab. The lab is truly a Valhalla for technophiles—a fact easily forgotten when you're head-down, working flat-out like a lizard drinking. It is difficult to imagine how the lab could have been brought into being and operate without his tireless efforts to seek continued funding for it.

I have been fortunate to receive the assistance of many people while performing the work described in this dissertation. Without the following people's help this dissertation would not exist.

> Henry Holtzman, what can I say? Without your help in maintaining the various machines I used and keen perspective—on issues both technical and political—I doubt I would have survived the program.
>
> Scott Smith, I can't thank you enough for porting my software to NT, and also for developing the distributed CD Juke Box. Your patience and keen programming skills were essential in debugging the server; without you it would not have worked.
>
> Stefan Agamanolis and Christian Baekkelund, your perseverance and feedback during the augmentation of your Reflection of Presence application was similarly indispensable in debugging the server.
>
> Jonathan Salz, there's no way the user-centered simulation results would have been generated in time without your speedy implementation of the region based network simulator and facile grasp of the on-demand media delivery system developed in this dissertation.
>
> Dan Gruhl, thank you for answering countless questions on threads and programming in general.
>
> Jane Wojcik, Michail Bletsas, and the rest of the NecSys crew, thank your for your quick response and assistance in tweaking the Media Lab's networks to support multicast.

Many other Media Labbers and MIT people also helped me along the way by asking great questions that helped me clarify my thoughts: Guillaume Boissier, Mike Bove, Chris Metcalfe, John Watlington, Jon Wroclawski.

Thanks also to the members of the Internet Research Task Force Reliable Multicast Research Group for many conversations, insights into current practice, and clarifications—in particular Jon Crowcroft, Jim Gemmel, Mark Handley, and Brian Whetten.

Much thanks must go to the Media Lab administrative staff who somehow managed to cope with my Aussie accent and somewhat direct style—in particular Gillian Galloway, Linda Peterson, Santina Tonelli, and Deborah Widener.

Special thanks must go to Linda Peterson for her invaluable help, perspective in times of stress, and willingness to read rough drafts of this and my Master's thesis.

A big Aussie "Thanks mate!" goes to Mark Kortekaas who also provided invaluable feedback on multiple drafts of this document and whose friendship helped me more than a few difficult times during my stay.

Another big Aussie "Thanks mate!" goes to Jon "O Dog" Orwant who not only plays a mean game of softball but also patiently applied his extensive editing skills to several papers and chapters of this dissertation, and in doing so finally taught me how to write.

Thanks to Dave Morgan and Robert Wallace of Motorola. Dave, your grace, dignity, and wisdom are inspirational. I can't thank you enough for all your help. Bob, thanks for your encouragement and for providing me with the cable modem performance figures I needed.

Last but by no means least, I have been very fortunate to have the friendship, love, and support, of many people during the nearly six years I have spent at MIT At the risk of missing someone out, I'd like to thank the following people whom I've not already thanked: Dave Alexander, Nicole Atkins, Bill Butera, Freedom Baird, Katy Brown, Vanessa Chan, Pascal Chenais, Peter Dodds, Cris Dolan, Matthew Dyer, Michelle Evard, Paul Huibers, Tanya Kanigan, Simon Karecki, Pat Kreidl, Karen Lauchlan, Megan Morris, Wendy Plesniak, Matthew Reagan, Tom Slowe, Phil Soo, Peter Trapa, John Underkoffler, Alex Westner, Karen Willcox, and Caleb Winder.

Roger Kermode,

March 27, 1998

Author

Roger Kermode earned undergraduate degrees in Electrical Engineering and Computer Science from the University of Melbourne, Australia, in 1989 and 1990. He was awarded a Telecom Australia Education Fellowship in 1990 for his final year of undergraduate education. After completing his undergraduate studies, he joined the Telstra (Australia) Research Laboratories as a Research Engineer. There he investigated cell loss recover techniques for packetized digital video and implemented an ISDN protocol stack for a secure voice, fax, and data terminal. In 1992 he left Australia to commence study at the MIT Media Lab, where he subsequently earned his Master's degree in 1994. During the summers of 1993 and 1994 he worked for Silicon Graphics Incorporated on MPEG video coding technology and was also the President of the MIT Graduate Student Council for the 1994/1995 academic year. He is a Fulbright Scholar and has received fellowships from AT&T and Motorola. His research interests include scalable media distribution architectures, multicast networking, video coding, object-oriented representations for multimedia, and computational media.

To my Family;

George, Fairlie, Ivy May, and Meredith,

whose love, support, and understanding throughout the many years of study

have made this thesis possible.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The computer has forever changed the way in which we create, manipulate, and store information. All manner of media—from text and pictures to audio and video—is now operated on in digital form, and the old analog forms that use atoms instead of bits [10] are finding less and less use. However, this digital revolution is only half complete. While the tools exist for the creation, manipulation, storage, and display of digital media, an efficient means for distributing it in the ways traditionally used by analog media does not.

The problem lies in the distribution channel. Today the vast majority of computer traffic is carried over the Internet using unicast—or point-to-point—protocols, with only two parties involved in any conversation. While telephone calls over the Internet will work using these protocols, mass media applications such as radio, television, or news will not. Neither will distributed group applications such as video conferencing. These applications are inherently point-to-multipoint and multipoint-to-multipoint respectively—they involve multiple receivers and potentially multiple senders.

As a result, increasingly sophisticated applications that lend themselves to multiparty conversational models are using the point-to-point protocols to emulate multipoint ones. For example, every time someone wants to listen to a music clip stored on a Web server, that server must retransmit the audio data separately for that client. This traffic replication not only increases the probability of server overload, but also increases the likelihood of congestion on network links used by the server.

Multicast [94]—or multipoint-to-multipoint—distribution mechanisms remove this replication, but at the cost of also eliminating on-demand access—all receivers receive the same data at roughly the same time. Thus, the challenge of providing a general digital delivery network is to devise a reliable multipoint distribution mechanism that affords the efficiency of multicast and the flexibility of unicast. The essence of this dissertation is that such a network can be realized within the framework of the Internet. Since both the number of people using the Internet and the amount of bandwidth applications use will continue to increase, this distribution mechanism must be developed with an eye towards continued growth. If this does not occur, the high bandwidth multipoint applications that people are expecting (such as true on-demand pay-per-view movies and multiparty telepresence) will not work.

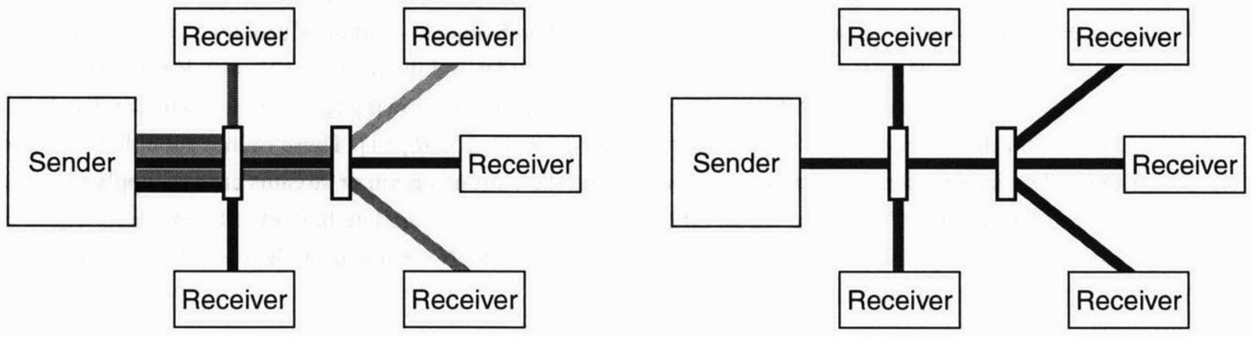


Figure 1-1 Point-to-point versus Point-to-multipoint mass media distribution

1.1 The problem: multicast is unreliable and implies synchronized receivers

Creating a point-to-multipoint or multipoint-to-multipoint mechanism for media delivery that is reliable and also scales is not a trivial task; one must allow for a number of complicating factors.

A solution should not make assumptions about the application, network type, network bandwidth, or receiver capability. While not mandatory, the ability to provide media on-demand appears to be an important feature that should be supported wherever possible. Any solution incapable of supporting new applications as they are developed will quickly become obsolete. Solutions that only work for certain network types or above certain bandwidths are similarly inefficient. Finally, the ability to simultaneously support different receivers is essential. Not everyone will be able to afford the biggest and best receiver or network connection, nor is it reasonable to expect everyone to upgrade their equipment at the same time when new receiver software releases occur.

To summarize, the resulting solution must be:

- **Scalable**: it should be capable of supporting very large numbers of receivers.
- **Incremental**: it should be deployable in a piecemeal fashion and not require widespread changes to the network.
- **Flexible**: it should work over heterogeneous networks and allow for heterogeneous receivers.
- **Extensible**: it must allow for new data and applications when they are appear.

This dissertation addresses the creation of solution to this problem for the Internet and the Internet Protocol [95, 103]. The Internet Protocol provides a best-effort delivery service—one that is fast but does not guarantee reliable delivery. Therefore, the solution developed must provide its own reliability mechanisms.

Many attempts have been made to add reliability to IP multicast [11, 14, 27, 29, 31, 32, 33, 44, 50, 57, 59, 63, 64, 74, 78]. Unfortunately, this seemingly simple task has proved to be one of the most difficult problems in networking. The basic problem is that receivers lose packets at different times. They are then needlessly subjected to listening to the subsequent repairs intended for others. This additional repair traffic can—and frequently does—cause congestion and hence losses in previously uncongested portions of the network. It is therefore very important that any reliability mechanism must not further exacerbate the congestion that could have caused the packet to be dropped in the first place. Solutions that require the network to be rebuilt before they can work are impractical.

Multicast implies that a single copy of the data is sent to all receivers listening to a particular stream. In order to provide the illusion of on-demand access, each receiver must be equipped with a sizeable cache. Previous caching work has concentrated on the creation of caches for entire small web objects—either as proxies [23, 83] or as part of the routers [18]—or the use of intermediate caches between the sender and the receiver [26, 66]. Recent work in the delivery of streaming media objects has taken a more aggressive stance in allowing receivers to cache the data locally [5] or to negotiate for the delivery of data cached inside their peers [1, 25, 46, 51]. These caches should also be capable of retrieving data independently when people join late and the current stream or streams cannot supply the necessary data in time. The ability to "catch up" in an efficient manner when joining late has yet to be explored within the context of multicast delivery and is essential, since it is this ability that affords the user choice and frees him or her from having to view the same material as everyone else.

1.2 The solution: distribute application-aware caches throughout the network

This dissertation proposes a set of methods to realize a unified digital data delivery service. This service is realized by simultaneously developing several pieces of technology:

- A scalable reliable multicast transport mechanism.
- A cache, potentially within every receiver.
- A scalable mechanism for providing on-demand delivery.

Aspects of these three technologies have been studied in great depth individually:

- providing reliability mechanisms for a single multicasted stream [11, 14, 27, 29, 31, 32, 33, 44, 50, 57, 59, 63, 64, 74, 78].
- adding Quality-of-Service (QoS) and reliability mechanisms to the network itself [47, 90, 92].
- supporting synchronized heterogeneous clients [52].
- creating mechanisms for reducing the start-up latency for single media objects delivered by broadcast from a central source [6, 21, 22, 34, 72].
- caching of media objects in their entirety [15].
- optimizing disk layouts for local play back [4].

However, no work has yet been published that deliberately undertakes the design of a smart network cache by considering all three technologies simultaneously. The smart network caches developed here are designed in this manner. This a key contribution of this dissertation. A new principle for media delivery is developed to assist in this process: *Content and Application Negotiated Delivery* (CANDY). The CANDY principle embodies two ideas: every session member can be a cache or server for every other session member, and session members use hints that describe the data, the manner in which it is transmitted, and their needs to negotiate for missing data.

The ideas contained within the CANDY principle stem largely from two others: Application Level Framing and Localized Recovery. Application Level Framing (ALF)—as proposed by Clarke and Tennenhouse [19]—states that data to be transmitted over a network should be packetized and tagged by the application, since it knows how to perform this operation most efficiently. This breaks the often referenced but seldom used concept of protocol layering espoused within the OSI layered protocol stack [65, 69]. Localized Recovery—as put forward by Floyd *et al.* [27]—states that when a session member requests data, the closest possible receiver capable of doing so should be the one to supply it.

CANDY augments ALF and localized recovery by stating that hints about the data, its method of transmission, and receiver needs should be supplied to smart network caches to assist in data delivery with minimal network impact. Smart network caches use these hints to determine what data can be gleaned from existing streams, and to negotiate with other caches for the retrieval of any remaining data from the nearest cache that contains the missing data. The use of the CANDY principle by smart network caches implies a new application of the ALF principle to cache as well as network operations.

1.3 The approach: solve the basics first

The realization of a unified digital data delivery service is a difficult multifaceted problem composed of a number of parts. Some of these parts—such as multicast address allocation [109, 110, 112, 113, 115, 116], object discovery [108, 114, 117], and security [85, 86, 87]—are relatively discrete issues and are the subject of ongoing independent research. Therefore, this dissertation concentrates on the aspects associated with scalable reliable delivery. Specifically the work describes:

- a flexible naming scheme that can be used both inside a cache and as part of a transport protocol.
- a preliminary sample of hints that could be used to convey information about an object, its method of delivery, and a receiver's needs for its data.
- a scalable reliable transport mechanism for delivering a single multicast stream.
- a new transport protocol that not only incorporates all of the above but also includes mechanisms for effecting late joins.
- a number of different late-join mechanisms and their relative performance.

1.4 Dissertation Overview

The remainder of this dissertation develops the design for a smart network cache, concentrating on the design aspects listed in 1.3. Where appropriate, simulation results are reported to support claims of scalability. Chapters 2, 3, and 4 develop three essential pieces of technology associated with scalable reliable delivery required by a smart network cache: Data naming, Content and Application Hints, and Reliable Multicast Data Delivery. They set the stage for Chapter 5, which merges them into a single scalable reliable transport protocol. Chapter 6 shows how the newly developed protocol can be used to deliver media on demand—specifically streaming media in the form of movies. Chapter 7 describes two test applications that were implemented as proof that the protocol developed in Chapter 5 can be realized in the real world, while Chapter 8 summarizes the work and points towards

- Chapter 2, "Data Naming" develops a naming scheme for use both within a network transport protocol and also a local-object cache. The fact that a single naming scheme is used by both the cache and the network eliminates the need to translate data after reception and thereby provides the means for efficient data exchange between caches in different parts of the network. Unlike previous naming schemes that use fixed formats, the scheme described in this chapter uses formats completely determined by the sending application. Senders convey the data format for each data type used within an object to the local smart cache, which creates a meta-description to inform other caches of how to interpret the object data. The naming scheme's structure is also specifically designed to support layered encodings, and late joins.
- Chapter 3, "Content and Application Hints" describes the development of a basic set of hints that sending and receiving applications can use to expedite data delivery between each other. These hints assist the smart network caches in making intelligent decisions about how to reduce their use of network bandwidth. In effect, the hints are the means by which the network cache becomes smart. They affect caching policy decisions regarding whether or not an object can be cached, how long it can be cached for, permission to write to an object, and the decision as to which session members can take part in effecting the repair of others' caches.

- Chapter 4, "Reliable Multicast Data Delivery" examines existing methods used to reliably multicast a single stream of data, and then introduces a new method that has the potential to scale to the numbers required for mass media distribution. The new method fully exploits the existence of object data in other caches and is the first to combine Automatic Repeat Request and Forward Error Correction methods with localized recovery through administrative scoping. Simulations reported in this chapter show that the new method drastically reduces traffic in the core of the network and localizes repair traffic to regions where it is needed most.
- Chapter 5, "The Hierarchically Partitioned Reliable Transport Protocol (HPRTP)" draws together the threads of work presented in Chapters 2, 3, and 4, and interweaves them to create a new transport protocol—HPRTP—that can be driven directly by a smart network cache. HPRTP reuses the administratively scoped zones created for the single multicast stream delivery method in the previous chapter to further reduce the scope of late-join traffic.
- Chapter 6, "Partitioning Schemes" shows how the HPRTP protocol can be used in a number of different schemes to scalably deliver streaming object data. Both data-centered (broadcast) and user-centered (on-request) methods are explored. A new data-centered scheme for providing infinitely scalable on-demand movie distribution is developed in this chapter. This chapter also presents simulations that show how HPRTP localizes late-join traffic within a hypothetical cable television network.
- Chapter 7 "Test Applications" describes two test applications—Reflection of Presence, a novel video conferencing system; and the Distributed CD Juke Box, a distributed network CD player application—that use smart network caches to assist in data delivery. These two applications validate the HPRTP protocol and the ability of smart network caches to autonomously retrieve data from previously unknown locations to present a seamless presentation.
- Chapter 8, "Conclusions, Deployment, and Future Work" summarizes the findings and technologies developed, briefly examines possible deployment scenarios, and lists areas for further study.

Chapter 2

Data Naming

The concepts of multiple interlinked media objects, the selection among various objects, and the synchronization and scheduling of multiple objects during playback, as identified by Werner and Wolf, are major factors that affect the user's perceived quality of performance and interactivity [73]. All three concepts are determined by the granularity with which an object's data is partitioned for storage and transmission. The storage and transmission granularity may not be the same, as Rada states in his book "Interactive Media" [61]. Furthermore, the implications of how one chooses to partition data affects the client's ability to schedule the successful presentation of composite media objects. The memory and communications resources available to a client are not infinite. Therefore, careful scheduling of data movement among the various storage elements is required to ensure that the presentation occurs smoothly, without jumps or breaks, should the demand for these resources become high.

An application that attempts to address these issues is the MPEG systems stream [39] that provides two methods for interleaving MPEG-2 video[40] and MPEG-2 audio[41] streams into a single stream. The first method defines the Program Stream to be used in situations where the data can be delivered without loss to the decoder; it is used primarily to decode MPEG-2 data that is stored locally on a hard disk or CD-ROM. The second stream, the Transport Stream, is designed to be used in situations where the data must first be carried over a possibly unreliable network before being decoded. While the MPEG-2 systems stream provides synchronized management of multiple data streams, its application is limited to MPEG-2 audio and video data. Once one expands the types of data that can be carried to include all kinds of media in many different formats, it becomes apparent that a more generic mechanism for managing the delivery and synchronization of multiple media objects of disparate types is needed.

2.1 Packet Identification

One of the most important tasks of any transport protocol is that of distinguishing one packet from another. The choice of packet identification method affects the protocol's operation at a fundamental level. Error correction methods, data packaging, and the ease with which a client can join are all determined by packet identification methods. Since smart network caches are expected to cache the contents of packets, packet identification also affects the operation of the local RAM and disk caches. Clark and Tennenhouse's ALF principle [19] addresses this problem.

The ALF principle dictates that the task of packet identification should be left to the application, since it knows how best to segment its data for transmission and how to recover from errors when they occur. Applications that use ALF tag packets with Application Data Unit Identifiers (ADUIs) to give real meaning to the data they contain. This enables their clients to join multicast sessions late and to synchronize instantly. ADUIs also encourage repair mechanisms that use receivers to supply repairs in addition to the originating source.

Packet identifiers should be constructed from two pieces of information: a fixed-format Data Source Identifier that specifies the data's original source, and a variable-format Application Data Unit Identifier.

2.1.1 Data Source Identifier (DSI)

Traditional delivery protocols such as TCP [67, 68, 76] and UDP [67, 76] use a combination of the sender's IP address and a port number to determine a packet's source. While this approach works well for unicast applications, it does not scale well for multicast applications, since repairs must originate from the original source. Scalable Reliable Multicast (SRM) [27] and the Real Time Transport Protocol (RTP) [105, 106], solve this problem by tagging each packet with a unique number that identifies the data's original source instead of the host sending the packet.

Several benefits follow from this additional freedom:

- Reliability and robustness increase, since multiple hosts can act as data sources for the same object.
- Additional hosts capable of responding to a repair might be closer to the host needing the repair, thus repair times might be lower in those cases where repairs can only originate from the source.
- Repair requests and repairs can be localized to regions of the network immediately adjacent to the receiver.

Having determined that Data Source Identifiers provide a superior means for differentiating a packet's source, one must decide how the DSI should be chosen, and what form it should take. One simple solution is to use the host's IP address; however, this leads to problems when one considers DSIs in a global context. DSIs must be globally unique, and while in theory IP addresses are globally unique, in practice many companies hide duplicated addresses behind firewalls and gateways. They do this to partition their local networks from the global internet. Another problem that arises from using IP addresses occurs as a result of the impending shift from the current version of IP (IPv4) [67, 76, 103]—which uses 32 bit addresses—to the next generation of IP (IPnG or IPv6) [9, 93, 95]—which uses 128 bit addresses. IPv4's 32 bits represent a total of 4.3×10^9 sources, moving to IPv6's 128 bits addresses that represent a total of 3.4×10^{38} sources is an over-kill that increases bandwidth requirements.

RTP's designers recognized these potential pitfalls and chose to use a single 32 bit number for RTP's DSI equivalent, the Synchronization Source (SSRC). SSRCs are "randomly chosen to be globally unique" [105] using the Message Digest 5 algorithm, MD5, [104] and as such there is a finite probability that two clients within a session will choose the same SSRC value. Session members detect SSRC collisions by using a table that maps SSRC values to the host addresses for all the senders within a session. When a message containing a new SSRC is received, it is checked against the table's entries. If an entry is found that matches the SSRC, but not the address, then a collision is signalled and steps are taken to inform the offending sender that it should generate a new SSRC.[1]

1. As an aside, RTP also uses SSRCs in regular informational transmissions from both senders and receivers. These Sender Reports (SRs) and Receiver Reports (RRs) are used to inform the receivers of the current sender state and also allow the receivers to indicate when they are listening and how good their reception is. As the number of receivers increases, so does the number of RRs. RTP solves this problem by requiring receivers to throttle back the sending rate of RRs as the number of receivers increases. Ideally, RTP's RRs allow receivers to control the senders' operation; however, in practice, this ability diminishes quickly as the number of receivers—and hence the time between consecutive reports from individual receivers—increases.

2.1.2 Application Data Unit Identifier (ADUI)

The central tenet of the ALF principle is that the packet identifier format is determined on a per-application basis. The ALF principle shifts the responsibility for packetization out of the network layer and into the application. This shift affords many benefits, but it also has one drawback: one can no longer determine the meaning of an ADUI format from observing a single packet.

This problem can be solved in two ways. First, the ADUI format can be selected from a small number of payload-specific fixed-format ADUIs. Sending applications choose one format, and then use a payload identifier to inform receivers of which one to expect prior to data transmission. Each payload identifier is associated with a particular data type—such as MPEG2, JPEG, or mu-law audio—and determines how the common skeleton format is fleshed out on a per-data-type basis. This method is used by RTP and, while efficient, presupposes that all possible ADUI formats can be predetermined prior to their being needed. This restriction makes it difficult for new data formats to find wide use.

The second solution assumes that the receiver and sender do not use a code book of ADUI formats. ADUI format information is conveyed from the sender to the receivers using a meta-description language. This method is far more flexible and allows ADUI formats to be more finely tuned to the data being transmitted. Multidimensional descriptors can easily be constructed without shoe-horning the desired ADUI into a pre-existing format that may not be as efficient or meaningful. Consider the following three sample applications:

- whiteboard (*wb*)
- a multithreaded narrative, movie or novel
- a large relational database

U.C. Berkeley's whiteboard application, *wb*, [84] was one of the first applications in wide use that utilized the ALF principle. *wb* provides a means for viewing and modifying the contents of an electronic whiteboard by participants who are geographically dispersed around the globe. A commonly used tool during Internet Engineering Task Force (IETF) meetings, *wb* allows participants to annotate drawings and create new pages with the changes propagating to all other *wb* instances in the session. *wb* deals with some fairly tricky synchronization issues, such as "How does one distinguish between changes made at the same time to the same page by different people?" *wb*'s solution involves the use of a three-part ADUI that consists of the participant's SSRC, a page id, and a drawing operation counter. While this ADUI structure works well for the short-lived sessions of the IETF, one can envisage others where one may wish to segregate the page space further into groups, say by topic. In these situations, a fourth dimension could be added to the ADUI for this purpose.

In the second example, the multithreaded narrative, consumers navigate their way through a story space that is composed of multiple paths, where each path corresponds to a different version of a piece in the narrative. Theoretically, pieces could be differentiated from one another using any number of metrics from a simple rating—like those currently used for movies (and more recently television shows)—to a character's point of view (POV). Each scene in the movie, TV drama, news report, or documentary could be annotated with a number of metrics that are checked against a profile stored locally on the consumer's computer or set top box. Thus, one can imagine several multidimensional ADUIs for the delivery of multithreaded narratives, as shown in Figure 2-1.

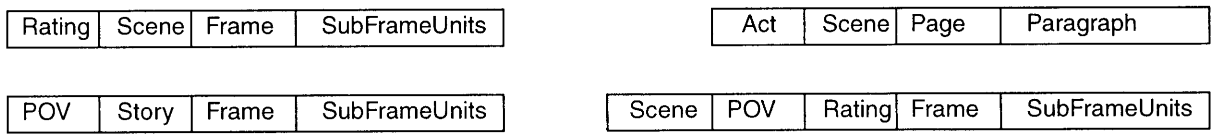

Rating | Scene | Frame | SubFrameUnits
Act | Scene | Page | Paragraph
POV | Story | Frame | SubFrameUnits
Scene | POV | Rating | Frame | SubFrameUnits

Figure 2-1 Example Multithreaded Narrative ADUIs

In the final and most general example, a large relational database, clients connect to one of a number of servers containing data. They then issue requests for small subsets of that data by specifying restrictions on the values of one or more fields associated with each entry. Matching entries are returned to the client. In this situation, the ADUI naturally forms from a concatenation of the fields used to describe an entry. Given that there are likely to be nearly as many different field combinations as there are database instances, it makes sense to use a variable format ADUI that is advertised prior to the client making requests.

A review of these three examples shows that the first two are specific instances of the third. Consider the *wb* application. Its database entries are described by an ADUI that consisting of three fields: an SSRC identifier, a page id, and a drawing operation number. All data within the *wb* space can be described by sub-matrix operations on this three-dimensional space. Likewise, the ADUI in the interactive narrative application is constructed from a number of fields that when combined allow for the unique specification of any data element or sample needed to make a presentation.

One must devise a mechanism that allows the sender to inform the receivers of the ADUI's format. One mechanism is to use a meta description for ADUIs that describes the number of fields, their length, and how they change over time during the course of single-speed playback. Listed in order from slowest varying fields to fastest varying, this information allows the ADUI format to be varied at will for the most efficient description. Fields common to different media types within a presentation are listed first with media-specific fields listed last. This allows the synchronization points between different temporal media to be made explicit. In addition, the process of describing subsections of the media object is simplified since a single description can simultaneously apply to multiple media. An example of how this restriction could be applied is shown in Figure 2-2.

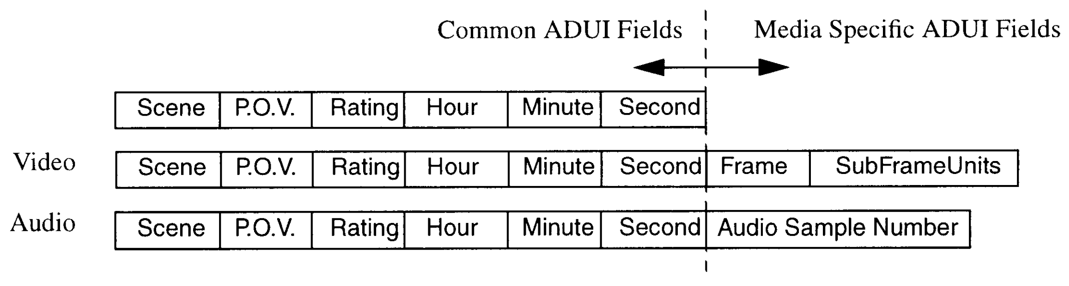

Common ADUI Fields
Media Specific ADUI Fields
Scene | P.O.V. | Rating | Hour | Minute | Second
Video
Scene | P.O.V. | Rating | Hour | Minute | Second | Frame | SubFrameUnits
Audio
Scene | P.O.V. | Rating | Hour | Minute | Second | Audio Sample Number

Figure 2-2 Common vs. Media Specific ADUI Fields

2.2 Data Scoping Through Hierarchical Partitioning

A key feature missing from most delivery methods is the ability to limit the amount of traffic that a receiver handles by separating traffic into separate channels according to its type. Hierarchical Partitioning builds upon the foundation provided by layered video coding and generalizes the concept of layering by resolution and time introduced by RLM [53, 54]and LVRM[49]. Object data is split into a number of multiple parts according to application metrics. Examples of metrics for streaming media are data type, encoding method, spatial resolution, temporal resolution, and playback offset. Higher level metrics for movies such as rating, act number, or scene number, and distributed games such as position or radar type are also possible.

The remainder of this section describes how one can construct a hierarchical framework of multicast groups for the transportation of an object's multiple parts. The hierarchy is designed with the lessons learned from the previous section in mind, and lays the foundation for the development of a protocol that provides support for:

- Arbitrary format ADUIs
- Separate independent mechanisms for short-burst and late-join repair traffic
- Localized recovery for both short-bursts and late-joins

2.2.1 Logical Data Streams (LDSs)

A Logical Data Stream is part of an object or presentation in which the ADUs all have the same format and level of meaning. The use of Logical Data Streams (LDSs) allows one to construct a three-level hierarchy for data delivery that minimizes the volume of unwanted traffic heard by any given receiver. Some examples of Logical Data Streams are:

- The MPEG encoded representation of an audio stream.
- A raw-format representation of an audio stream.
- The MPEG encoded representation of the simulcast French audio track for a movie.
- The lowest level in a spatially layered encoding of a video stream.
- The whiteboard portion of an IETF session over the multicast backbone (mbone).
- The radar information associated with aircraft within a certain volume of virtual airspace.

It is possible for an object to have several LDSs that are syntactically identical but differ in semantics (e.g. two different layers of layered MPEG video). It is illegal, however, for an object to have two LDSs that carry syntactically and semantically identical ADUs. Were this allowed, receivers would be unable to determine to which LDS they should subscribe.

A simple example of how one might decompose a multiple format, annotated, multilingual television broadcast of an Olympic sporting event is shown in Figure 2-3. In this example, the object's data is decomposed into nine LDSs. Each data stream is logically distinct from the others. Some require significant bandwidth, (e.g., the Highest Spatial Resolution layer of the MPEG-2 layered video stream), while others are require very little bandwidths (e.g., the closed captioning tracks in English and Spanish.)

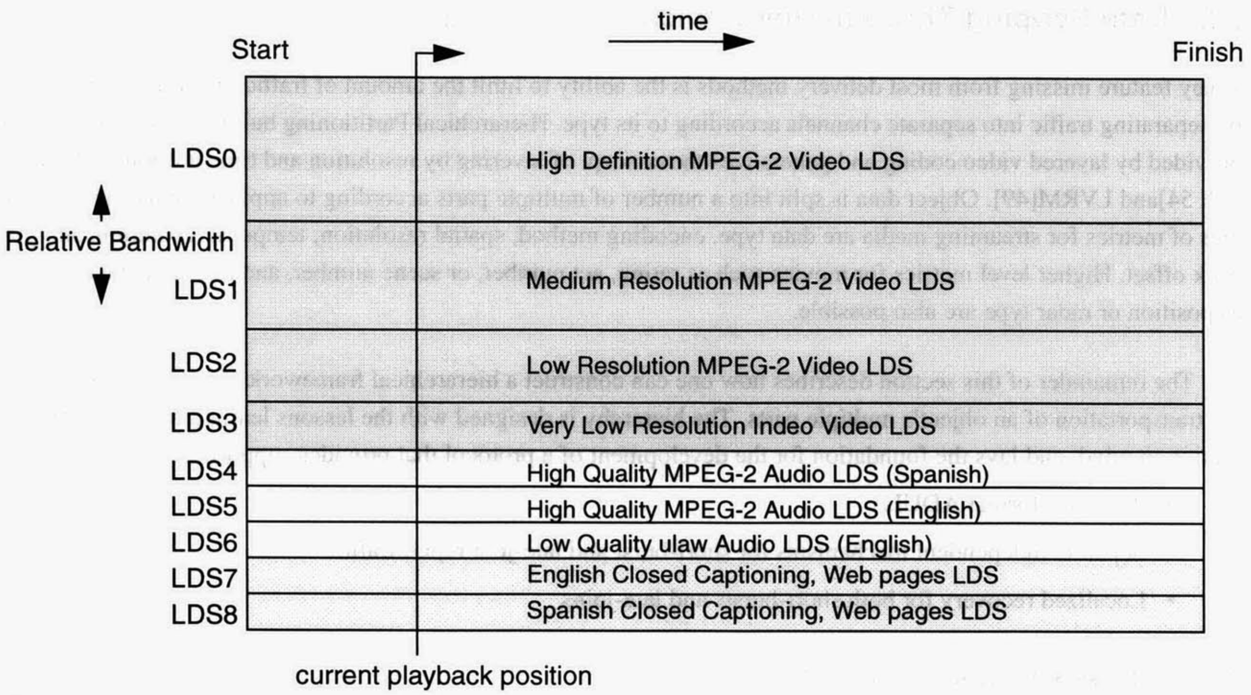

time
Start
Finish
LDS0
High Definition MPEG-2 Video LDS
Relative Bandwidth
LDS1
Medium Resolution MPEG-2 Video LDS
LDS2
Low Resolution MPEG-2 Video LDS
LDS3
Very Low Resolution Indeo Video LDS
LDS4
High Quality MPEG-2 Audio LDS (Spanish)
LDS5
High Quality MPEG-2 Audio LDS (English)
LDS6
Low Quality ulaw Audio LDS (English)
LDS7
English Closed Captioning, Web pages LDS
LDS8
Spanish Closed Captioning, Web pages LDS
current playback position

Figure 2-3 Logical Data Stream Decomposition of a multi-format annotated multi-lingual TV broadcast.

Given this set of nine LDSs, a variety of presentations tailored to the receiver type, network connectivity, and location can be derived. Some example presentations are

- High-bandwidth, best-quality home theater, Spanish audio: LDS0 LDS1 LDS2, LDS4
- Low-to-medium bandwidth PC "screen saver" at work (audio muted): LDS3, LDS7
- Low-bandwidth English audio only: LDS6

Logical Data Streams provide layering at the coarsest level. They assume that all receivers are synchronized and display each LDS at the same single point in time. In other words, it is not possible for a person to arrive home, turn on the receiver, and start playing the various streams from a point ten minutes in the past.

2.2.2 Logical Data Stream Partitions

The decomposition of an object's data into separate Logical Data Streams only partially reduces the amount of traffic on the network. Additional on-demand accesses require traffic duplication, and eventually either the network capacity must be increased, or the number of distinct Logical Data Stream sets must be restricted. Thus, the number of receivers that can be served is limited. Logical Data Stream Partitioning solves this dilemma by using the following observations

- Since multicast distribution is used, receivers can receive data from more than one set of LDSs.
- Given sufficient knowledge about the playback characteristics of the various LDS sets, receivers can selectively subscribe to LDS sets that are ahead of the current playback position and cache the data they carry for later use.
- If receivers are likely to pre-load data from multiple LDS sets, individual LDS sets need not playback the entire object.

Logical Data Stream Partitions extend Logical Data Streams by allowing multiple playback points. The previous example, Figure 2-3, uses a single playback point. Each LDS is transmitted over its own multicast group, but with the restriction that the playback points of each LDS were synchronized to that playback point. In the example below (Figure 2-4), there are four playback points, or partitions. (N.B., since the position of a playback point is the same across all LDSs, only one partition is shown for clarity.) These partitions are equally spaced and traverse the entire media object before looping back to the beginning in a manner similar to that used for Pay-Per-View (PPV) movies on analog cable TV systems.

This partitioning scheme affords several benefits. The receiver can subscribe to one or more partitions, thereby allowing the movie to be downloaded into the local cache in less time than it takes to watch it. Once the movie is in the cache, the receiver can perform random accesses to any scene without further network accesses. Furthermore, late joins can be made using the partition closest to the desired playback point, not just the original leading partition. Finally, network congestion is reduced through the following mechanisms:

- Receiver traffic is aggregated into a few partitions that correspond to different object regions, thereby reducing the redundant traffic volume.
- Receivers can drop partitions whose data they have already cached, potentially further reducing the amount of redundant traffic.
- Receivers can drop non-essential partitions or entire logical data streams when congestion occurs instead of waiting and continuing to request data.

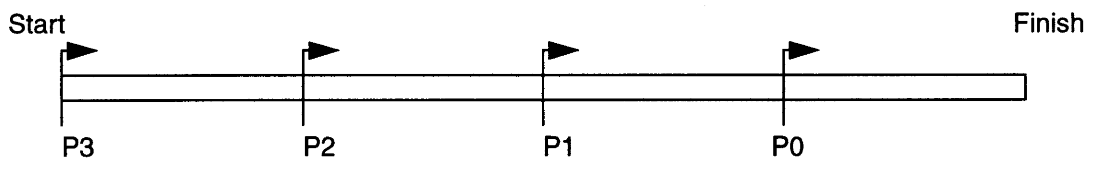


Figure 2-4 Simple Partitioning example containing four equally spaced partitions

2.3 Application Specific Media ADU and Partitioning Examples

Partitioning can be applied in any number of application-specific ways, a few examples are listed below. Some of these example applications use prerecorded data, others live, and most tend to deal with streaming video and audio. The final example shows how partitioning could be used with discrete non-streaming data in the form of game events.

2.3.1 Weekly Serials

Unlike movies, which tend to be at least an hour or more in length, weekly TV serials and sitcoms usually contain 20 to 22 minutes of content for a half-hour show, or 40 to 44 minutes of content for a 1 hour show. The difference is made up with commercial advertising. Typically, TV serials and sitcoms are composed of many short prerecorded scenes viewed in a sequential manner and—like movies—are normally viewed from beginning to end. However, unlike movies and sitcoms—which are mostly self-contained—TV serials tend to be self-referential and build up a history and continuity that encourages viewers to develop the habit of viewing each and every episode. Thus it makes sense that weekly serials be represented as two LDSs with the following format.

Episode #	Scene #	Seconds	Frame #	Byte Offset

Episode #	Scene #	Seconds	Sample #

Figure 2-5 Sample ADU Format for Weekly Serials

The use of partitioning in the delivery of weekly TV serials opens the door for a number of new viewing paradigms. For example, episodes could be created that contain multiple versions of each scene, thereby providing a means for creating and delivering the multithreaded narratives. Viewers could use intelligent receivers to quickly reference the content of previous episodes so they could understand the current episode. Intelligent receivers could keep track of and cache the first several minutes of particularly high-rating shows to reduce access latency. Finally, advertisers could insert personalized advertisements at a discount to the viewer, or remove them entirely for a premium.

2.3.2 Custom News

Perhaps one of the biggest opportunities that partitioning makes possible is an efficient personalized news service that allows story segments to be cached throughout the day. Once cached locally, segments could then be composited together to form a custom presentation according to the needs of the viewer. This forward, store, and post-edit approach would eliminate practically all of the repetition currently seen in today's broadcast cable news services. The freed bandwidth could then be used for other valued-added services to be delivered. For example,

- developing stories could be filed one segment at a time, allowing a news service to deliver the most up-to-date footage possible in a breaking story.
- Stories could be constructed from only a few segments for a brief precis, or from many to provide in-depth coverage. The choice as to how much depth should be presented could be made directly by the viewer, or indirectly by a client-side agent that attempts to predict the viewer's mood according to the day of the week, time, date, location, or other user-defined metrics.
- News presentations could be constructed from multiple sources that include local, national, and international news services.

An example of a set of ADUI formats that would accommodate all of the above services is given below. A third ADUI format has been included in addition to those normally provided for video and audio Logical Data Streams. This third format would be used by an LDS that carries time-synchronized textual information in the form of closed captions or hypertext. The partitioning mechanism would then be used to provide separate delivery channels for different stories.

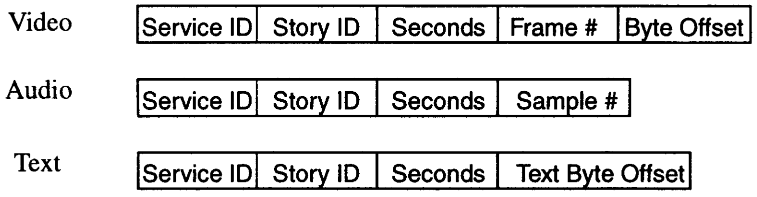


Figure 2-6 Sample ADU Format for a Custom News service

2.3.3 Live Sports Events

Unlike the previous two examples, live sporting events would be constructed "on the fly" thereby disallowing forward prefetching. Viewers watching the presentation would do so in a much more synchronized fashion; that is, the majority of viewers would be watching the event live as it happened. Consequently, the need to provide late-joins would be greatly reduced and more effort could be spent providing the live presentation in a reliable manner. As an aside, it should be noted that as sporting events are typically delivered in a broadcast fashion, receivers could be configured to deliberately introduce a small delay between reception and delay to allow for repairs to be made without significantly impairing the presentation.

The ability to simultaneously manage and receive multiple streams affords the ability to move the post production step out of the studio and into the home. Multiple camera views could be delivered to and cached by the set top box with the selection as to which camera angle to view being made locally, which could now also supply on demand instant replays. This ability to post produce in the home would be particularly useful for avid sports fans intent on analyzing plays, or for the Olympics during which too many events occur simultaneously to be covered to every viewers satisfaction.

It should be noted that the Olympics have already been a scene for introducing new media technologies. One notable experiment was NBC's ambitious triple-cast of the 1992 summer games and the subsequent backlash of "plausibly live" replays in the 1996 summer games. While "plausibly live" replays gave the illusion of events occurring live and for the most part worked, the triple-cast experiment of broadcasting simultaneously over three analog broadcast channels was by most accounts a failure, and did not attract significant market share. One could argue that a major reason for the respective success and failure of these two delivery paradigms lies in the level of effort required to keep track of the events. Plausibly live delivery constructs a single viewing experience integrated by sports personalties, while the triple-cast method puts a significant load on the viewer, who has to remember what event is being shown when, on which channel it is showing, and to structure the presentation accordingly. The shift from analog to digital delivery using partitions combined with intelligent filtering software to selectively choose among channels and create instant replays would reduce this load, while at the same time providing greater choice than before.

2.3.4 Global "Niche TV"

Another application that is made feasible by combining partitioning, caching, and increased scaling capabilities is global "Niche TV." People could create and distribute their own video programming at minimal expense compared to today's broadcasters. Unlike today's TV stations—which deliver content for most of every day—the creators of these programs would create at most a couple of hours of content per week. As such, their programs could fulfill niche markets where there are only a few viewers, or where receivers are so geographically dispersed that they are inaccessible distribution via traditional broadcast means.

2.3.5 Infinite-Membership Multi-User Networked Games

While the partitioning mechanism was designed primarily to facilitate late-joins, it could also be used as a traffic-limiting mechanism in networked games. Instead of being used to separate traffic on the basis of time, partitions could be used to limit traffic on the basis of space.

Consider, for example, a simulation game, such as Netrek [80], in which players roam around a virtual space pursuing each other or some goal. Traditionally, games like Netrek have been implemented using the hub-and-spoke model in which clients send events to a central server that performs synchronization before relaying them back to the client's peers. The use of multicast removes the need to perform synchronization at the central server and places it with each client. Data sent among clients would therefore be tagged with a global timestamp indicating when the event took place. Clients would introduce a small delay to allow for propagation skew for traffic from different receivers. This delay would enable them to sort events received out of order. An example of a serverless multicast game currently played over the Internet that uses these ideas is INRIA's MiMaze game [28].

As it is unlikely that the entire ADUI space described by an n-dimensional ADUI will be entirely populated, the meta description should also include mechanisms for describing which ADUIs are valid. Furthermore, in the case where they are played back over time, the mechanisms should also provide a means of describing the order of transmission. Finally, allowances must be made for the fact that a single ADU may contain too much data for a single packet. Sender's should be able to inform receivers that a field other than the fastest varying one is the nominal sample size. Thus, an MPEG video [38, 40] sequence's frames may be further segmented into smaller pieces such as slices or macroblocks for delivery, while still retaining the notion that the nominal playback is thirty frames per second.

2.4 Summary

The naming scheme developed in this chapter is flexible, extensible, and general. It builds upon the Application Level Framing (ALF) principle and layered coding to create a new data naming scheme based on arbitrary-length multidimensional bit-fields. This new scheme extends the ALF principle's use into the cache thereby providing—for the first time—a single general unified naming mechanism that can be used both locally on individual machines and over the networks that connect them. Furthermore, the new scheme's support for layered coding—through Logical Data Streams—and multiple playback points—through Logical Data Stream Partitions—sets the stage for efficient point-to-multipoint and multipoint-to-multipoint delivery mechanisms in which receivers selectively subscribe to a subset of the channels associated with a particular object.

Chapter 3

Content and Application Cache Hints

Traditional transport mechanisms deliver data blindly from one application to another. Data is considered opaque and no effort is made to advance, delay, or filter its reception—it is simply delivered. Should application designers desire the ability to schedule data for delivery within their applications, they must provide it themselves. Applications designed with the CANDY principle take a different view: since multicast and caching are used for delivery, the smart network cache should schedule the retrieval of data on behalf of the application. The fact that this responsibility shifts from the application and into the local smart network cache allows the cache to schedule network and disk access more efficiently between multiple applications on the same host in a more equitable fashion.

This chapter describes two broad classes of hints that applications may supply to the local smart network cache to assist in scheduling data retrieval. The first hint class—content hints—are provided by the object's sender. Content hints describe the data and the way it is delivered. The second hint class—application hints—are provided by the receiving application and describe its needs for the object's data. Content and application hints are merged within the local smart network cache to determine which multicast channels should be joined, when the join should occur, and for how long the cache should listen. The list of hints described in this chapter is not intended to be exhaustive; a full-scale deployment of smart-network caches and applications that use them may identify additional hints.

3.1 Content Hints

3.1.1 Reliability/Latency Delivery Model

Consider a media object's type as it relates to delivery to the display device. Video and audio have temporal aspects that demand timely, low-latency delivery. If a sufficiently large piece of data does not arrive in time, a discontinuity may result that will break the illusion that the object is continuous in time. On the other hand, documents and programs have a strong requirement for reliable delivery, and cannot tolerate any losses no matter how small. Related to both is the size of the packets used to deliver the data. Large packets take longer to deliver and retransmit in the case of packet loss, but offer efficiencies from lower overhead. Smaller packets take less time to deliver but require extra overhead since more packet headers are needed for the transmission of a given volume of data. The location of the packet boundaries is also important. Where possible, ADUs should not be split across multiple packets. This is especially important for Huffman encoded streams where certain codes are used for synchronization. In these cases, the synchronization codes should be aligned with packet boundaries to expedite resynchronization in the event of packet loss.

In general, combinations of varying levels of latency, reliability, and packetization methods are sufficient to categorize a delivery method for most media types. One can construct a taxonomy for the delivery of media based on latency and reliability, as shown in Figure 3-1. The reliability axis in this taxonomy has a discontinuity that indicates a shift from varying reliability (from best-effort to high) to guaranteed delivery, where no losses are allowed. Both the reliability and latency axes have origins that correspond to best-effort service, which reflects the base capability of the Internet today. With regard to reliability, best-effort service means that every effort is made to deliver the data but should congestion occur, packets may be dropped. In an unloaded network, a sufficiently low bandwidth stream of data is delivered without any losses. With regard to latency, best-effort service corresponds to delivering the data as fast as possible, given the constraints of distance, network types, and available bandwidth.

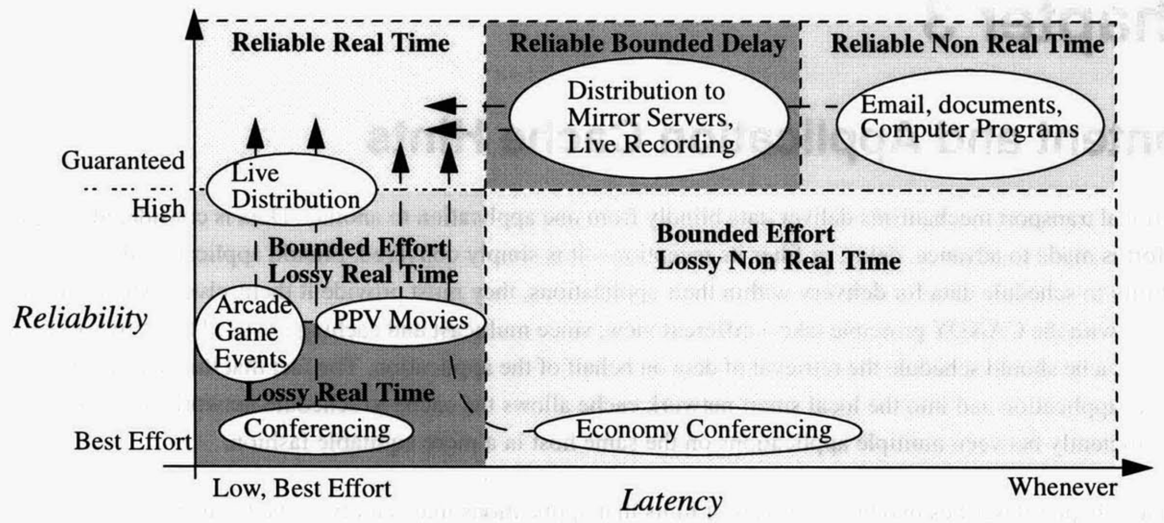


Figure 3-1 Delivery Requirements for Various Media

This Reliability/Latency Model hint specifies the reliability and latency properties used to deliver the LDSs within an object. It specifies the maximum level of service that a sender will attempt to provide when delivering an object. Examination of Figure 3-1 reveals that the reliability/latency space is divided into six regions. Each region corresponds to a certain combination of reliability and latency. Some applications straddle two regions; given sufficient resources, all applications would use the Reliable Real Time delivery model. The six regions and their definitions are as follows:

Reliable Non-Real-Time

The Reliable Non-Real-Time mode best matches the level of service afforded by TCP [67, 68, 76]. It guarantees that the sender will respond indefinitely to requests for missing data and that all the data will eventually be delivered regardless of the amount of time required. Receivers in Reliable Non-Real-Time sessions attempt to minimize bandwidth by using existing partitions whenever possible and also by delaying the creation of dynamic partitions for late-joins for as long as possible.

Reliable Bounded-Delay

The Reliable Bounded-Delay model also guarantees reliable data delivery, but with the added stipulation that data is delivered by a certain time. The time window over which the data is delivered is defined by the sending application, and is usually set to indicate a time after which the data being delivered loses its value. Receivers in Reliable Bounded-Delay session will minimize bandwidth in a manner similar to that employed by Reliable Non-Real-Time sessions, but will schedule dynamic partitions sooner to meet the bounded delay constraint.

Reliable Real-Time

The Reliable Real-Time model is the most resource intensive. Its use implies that data cannot be subjected to losses and must be delivered as soon as possible to all recipients. All members of sessions that use the Reliable Real-Time model are expected to take part in the repair of their peers. This includes both repairs at the packet level and also at the partition level for late-joins. Consequently, the Reliable Real-Time model places timely delivery ahead of constrained bandwidth, and hence has the potential to consume greater bandwidth than the other models.

Bounded-Effort Lossy Real-Time

The Bounded-Effort Lossy Real-Time mode corresponds to the best-effort service provided by UDP [67, 76], with the addition of a limited, time-bound repair service. Reliability is not guaranteed; however, every effort is made to deliver data as quickly possible. Unlike UDP, in which losses are irrecoverable, receivers may provide some repairs. Whether or not a receiver provides this service, and support for dynamic partitions, is a local decision. The provision of these services is neither mandated nor prohibited by the model. Receivers should not expect repair requests made after the bounded-effort delay value to be honored. Bounded-Effort Lossy Real-Time delivery is most likely to be used for the delivery of the most detailed layers in layered schemes sent by resource-bound hosts. In these sessions, repair of the lower layers is more important than repair of the upper layers.

Bounded-Effort Lossy Non-Real Time

The Bounded-Effort Lossy Non-Real-Time model provides the lowest Quality of Service (QoS). Neither reliability nor real-time delivery are strong requirements. As such, its main use may be in economy-conferencing, where the user is unwilling to pay for reliability or timely delivery. Since reliability is not a strict requirement, receivers are not required to provide repair service for other members. They may, however, be configured to do so at the user's discretion.

Lossy Real-Time

Lossy Real-Time delivery is most similar to the Bounded-Effort Lossy Real-Time service. However, it does not provide a repair service of any kind. Thus Lossy Real Time delivery can be thought of as offering service identical to that provided by UDP.

The Reliability/Latency Model hint not only determines the degree of reliability and amount of latency, it also affects the repair and late-join policies used by a session's members. Depending on the model chosen, receiver-based smart network caches may or may not choose to participate in the repair of missing packets and to serve late-joins carried over separate dynamic partitions.

Mode	Reliable	Latency	Packet Repair Policy	Late-Join Serving Policy
Reliable Non-Real-Time	Yes	Possibly indefinite	All members	Defer as long as possible
Reliable Bounded-Delay	Yes	Less than delay bound	All members	Defer up to delay bound
Reliable Real-Time	Yes	Real time	All members	Send as needed
Bounded-Effort Lossy Real-Time	Not guaranteed after bounded effort delay.	Real time	At members discretion	At members discretion
Bounded-Effort Lossy Non-Real-Time	Not guaranteed after bounded effort delay.	Possibly infinite	At members discretion	At members discretion
Lossy Real Time	No	Real time	None to be sent	Not allowed

Table 2-1 Summary of the features of the six delivery models

3.1.2 Logical Data Stream (LDS) hint

The LDS hint supplies meta-information about an LDS. It is used by receiving applications to determine whether or not they should subscribe to particular LDS. The hint consists of several attributes that must be defined by a sending application before it can transmit any of that LDS's data.

Media Payload Types

The Media Payload Type field contains the RTP payload type [106] of the data being carried by an LDS. The inclusion of this attribute enables receiver to quickly determine whether or not they are capable of processing a particular LDS's data. Receivers that determine they are unable to handle an LDS's data should not subscribe to that LDS.

Layering Index

Layered audio and video representations are used to provide scalability and flexibility. The Layering Index field provides a means for discriminating among multiple LDSs that have the same payload type, but represent different levels in a layered-coding scheme. Thus, an MPEG encoder operating in Spatial Scalability mode might generate three bit streams corresponding to the coarse, medium, and fine resolution with the layering index field set to 1, 2, and, 3 respectively.

Layer Drop Order

Given the variable availability of bandwidth in different parts of the network, it makes sense to provide a means for receivers to selectively drop layers from layered representations until the remaining layers are free from loss. The inclusion of this field enables receivers to determine the order in which the sender believes layers should be dropped. As a convention, LDSs with a higher Layer Drop Order value should be dropped first.

3.1.3 Personness

The Personness hint is used by the sending applications to announce whether or not session members can supply their own data to the session. It is based loosely on the human conversational model and can take on one of three values: first, second, or third.

First

Senders that identify their session as First-Person interactive inform other session members that they may originate their own content for a session. The local caches of these other session members accept information from the local instance of the application and retransmit it to the group. Examples of First-Person applications include public meetings, chat rooms, and collaborative work spaces.

Second

Second-Person applications allow session members other than the originating member to contribute, but only under the originating session member's control. An example of a Second-Person application is a moderated public meeting in which the original sender performs floor-control, and selects which senders may transmit at a given time. This task is nontrivial and beyond the scope of this thesis.

Third

Third-Person applications are ones in which only the member who created the session may transmit new information. The local caches of other session members may effect repairs, or create dynamic partitions to retransmit cached data. However, they must not accept new information from applications that attach to them. The majority of applications that publish content for mass distribution will operate in the Third Person to prevent other session members from maliciously overwriting the original content with their own.

3.1.4 Volatility

The volatility hint is used by senders to inform receiving caches as to how stable an object's data is, whether or not advance can be performed, and whether or not it makes sense to cache it.

Recorded

Recorded objects are entirely present at the sender when transmission begins. Hence, receivers may request data that has yet to be transmitted. For example, a user may wish to jump ahead to the halfway point in a movie that he or she was interrupted from viewing several days earlier.

Live

Live objects, unlike recorded ones, are not entirely present in the sender's cache when transmission begins. The object's data is transmitted as soon as it is recorded. Thus, receivers may not request data corresponding to events that have not yet occurred. Receivers may, however, request dynamic partition for data that has already been sent.

Live Delayed

In some cases, a live object may be transmitted subject to a small, finite time delay. In these cases, receivers may request dynamic partitions to jump ahead of the current partition's playback point, but only for the amount the original live stream was delayed.

Static

Static objects are ones for which fetching the ADU will always result in the same data being returned.

Dynamic

Dynamic objects are ones in which there is no guarantee that reading the same ADU multiple times will result in the same data being returned. This can occur when the ADUI format specifies spatial and not time coordinates.

3.1.5 Persistence

The persistence hint informs receiving caches of how long an object's data may be stored in the local cache. This hint takes one of four values: "Transient," "Time-to-live," "This Session Only," or "Indefinite."

Transient

Objects tagged as "Transient" must not be stored in the disk cache. Object data may, however, be stored temporarily in the RAM cache to effect repairs and to provide some buffering for the receiving application. Typically, the several most recent seconds of an object's data will be cached.

Time-to-live

Objects tagged as "Time-to-live" may be stored in the disk cache but must be discarded by the time supplied by the sender. This allows limited repeated playback of an object. The time by which an object's data must be purged can be expressed either as an absolute time or relative to the time the object was opened by a receiving application

This Session Only

Objects tagged as "This Session Only" may be cached for as long as the receiving application that opened the object remains connected to the local smart network cache. Once the receiving application closes the object or disconnects from the local cache, the object's data must be discarded.

Indefinite

Objects tagged as "Indefinite" may be cached forever, subject to local disk cache space availability. Applications may attach to the local smart network cache, open the object, and close the object an indefinite number of times.

3.2 Application Hints

Different applications will have different needs for data. Furthermore, these needs may change over time. Therefore, it makes sense to provide a mechanism for receiving applications to inform the local cache of its particular needs for data. This section describes three hints classes that receiving applications can use to achieve that end.

3.2.1 Intra-Object Locality

Applications use the Intra-Object Locality hint to describe their likely needs for data in the immediate future. Given a well-fitted locality model, the smart network cache can prefetch data for the application on a number of different scales. On a coarse scale, it can check the local disk cache, and fill in holes by scheduling selective reception on pre-existing partitions. Should the existing partitions not be able to deliver the missing data, applications may be able to schedule dynamic partitions that can. In addition, data may be prefetched on a finer scale from the disk cache into the RAM cache in anticipation of future read requests.

The Intra-Object Locality hint codifies this information. Five modes are currently identified; linear, bursty linear, random, application defined, and unknown. The difference between these modes lies in the level of confidence an application has in its ability to predict future needs. In some cases, the application will access data in a completely predictable fashion, while in others its access may be completely determined by the whim of the user. In these cases, the likelihood that prefetched data will actually be used is reduced.

Linear

Linear locality implies that an application will access an object's data in a monotonically increasing fashion. Thus, the cache expects that a receiving application that has just requested ADU(n) will next request ADU(n+1). This hint is particularly useful for streaming-media-player applications and operates at both coarse and fine levels. When invoked this hint contains one or more segments, each of which contains the following information:

- Starting ADU
- Stop ADU
- DSI of the original data source (optional if only one data source)
- Play Rate
- Time that playback will commence for this segment

Bursty Linear

Bursty Linear locality is similar to Linear locality, except that the constraint of knowing all future playback segments in advance is relaxed. Instead, Linear locality is only assumed about the last retrieved ADU. This hint is useful in situations where long-term linearity cannot be guaranteed. An example of such a situation occurs when one uses localized shuttling to align two video scenes for splicing together. As long-term locality constraints are ill-defined or unpredictable for Bursty Linear accesses, specifying a range of ADUs for predicting long term accesses is inappropriate. Consequently, the HPRTP cache will attempt to ensure that a certain minimum amount of data ahead the most recently accessed ADU is preloaded into the disk cache.

Random

If one further relaxes the locality constraint so that the assumption of linearity about the current ADU no longer holds and so that access to any ADU is equally likely, one derives the Random locality mode. When invoked by an application, this mode informs the smart-network cache that there is no point in attempting to prefetch data from the disk cache into the RAM cache. However, since access to a given ADU is as likely as access to another, the use of this mode also informs the smart-network cache that it should attempt to ensure the retrieval of the entire object into the disk cache as soon as possible.

Application Defined

Since not all locality models can be identified in advance, the Application Defined locality mode allows applications to specify exactly which ADUs need to be prefetched. In this case, the application is presumed to know its own immediate needs, and therefore is required to specify the following information:

- Current ADU (data returned)
- Next ADU (data prefetched into RAM cache but not returned)

This locality mode would be useful in a movie editing application that needs to playback at high speed. Given Application Defined locality, the application could skim over scenes and ensure that only the frames needed for display are preloaded from the disk cache to the RAM cache.

Unknown

The Unknown mode for Intra-Object Locality is similar to the Random mode in that initially one is unable to predict which ADU will be fetched next when given the current one. However, unlike the Random mode where this assumption holds, the Unknown mode tells the smart-network cache that there is some structure to the requests. Furthermore, it also informs the smart-network cache that it should attempt to discern what that structure is by observing the relationship among successive reads, and should create its own model for prefetching data from the disk cache to the RAM cache. This mode is for the "lazy" application that is either unable or unwilling to use the Application Defined locality mode.

3.2.2 Inter-Object Locality

Hyperlinks allow references to web objects to be embedded in other objects. Presently, hyperlinks take the form of a Universal Resource Locator (URL) that describes a static reference to an object in terms of the server on which it is stored and its position within the file tree of that server. While the current implementation of hyperlink URLs suffers from a number of well-known problems—the most severe of which is a brittleness that results from the confusion of the address space with the name space—the concept of hyperlinks is a sound and useful one that will continue to find greater use as networked media usage continues to grow.

The Inter-Object Locality hint allows the application to specify how the cache should treat links embedded with objects. Applications set the Inter-Object Locality to one of three values:

- **Greedy**—Open and prefetch all direct-linked objects.
- **Conservative**—Confirm before prefetching linked objects.
- **Paranoid**—Ignore linked objects.

The greedy mode attempts to minimize the delay for subsequent accesses to linked objects and assumes that the host has large amounts of storage and high bandwidth connectivity. The conservative mode requires confirmation from either the application or user before prefetching linked data. This mode is particularly useful for selective filtering of content. Consider a distributed news service with links to third-party content. One could configure an application to load related stories on certain topics and not on others using the conservative mode. The paranoid mode does not allow prefetching and is invoked under the assumption that links are unlikely to be followed.

3.2.3 Previous-Access Mode

In situations where objects are delivered using statically configured partitions, some mismatch between the partitioning scheme used to deliver data and the accesses performed is inevitable. In large sessions, feedback to correct the mismatch while the session is active may be prohibitively expensive. In these situations, receiving applications should observe the users access patterns and send a summary back to the sending application when accesses have been completed.

The sender will then be able to merge the previous-access summaries from multiple receivers and construct a more accurate model of their accesses. This model will then allow the sender to construct partitions that better match the needs of the clients that access the object in the future. This feature is particularly useful for weekly television shows that adhere to a well-defined format.

3.3 Summary

Hints about an object's content, mode of delivery, and the receiver's need for the object's data are the key means by which local smart-network caches determine what data is needed, when it is needed by, and how it can be obtained. They can also propagate information about permissions and about the ability of session members to supply missing data to each other. Caches use the hints to determine which Logical Data Stream they should subscribe to, which partitions can supply missing data, and also when and how long they should subscribe to a given partition. Hints enable the caches to make smart decisions about how to minimize network usage, and therefore are a key mechanism for increasing scalability.

Chapter 4

Reliable Multicast Data Delivery

When multicast service was first introduced to the internet in the form of the MBone [12], reliable delivery was not guaranteed. Since then, adding reliability to IP Multicast has been a much sought-after goal. This task is far from trivial. Initial efforts to map TCP's positive acknowledgment (ACK) model into a multicast framework worked only for sessions containing small numbers of participants. It quickly became apparent that any protocol that required every session member to transmit an acknowledgment back to the source would swamp the source—and the other receivers—with a flood of ACKs.

Several measures were taken to counter ACK flooding. Some reduced the frequency of ACKs by acknowledging multiple packets at once. Others introduced control structures such as token rings [14, 74]. Receivers in the token ring were required to take turns in supplying repairs and to halt transmission of new data until all previously transmitted data had been received. While protocols based on token rings guaranteed delivery, they did so at the expense of scalability, since managing rings containing large numbers of receivers is a very difficult and unwieldy task.

Another approach was to use negative acknowledgments (NACKs) instead of positive ones. Protocols based on this idea shifted the responsibility of error detection to the receivers, which now only sent traffic when packets were lost. While these NACK-based protocols reduced the volume of traffic considerably, they only reduced the frequency of flooding and did not eliminate it entirely. Consequently, they had the unfortunate side effect of increasing traffic during times of congestion, thereby exacerbating it, and causing progressively worse performance as losses increased.

The next tool developed to stem the flow of "back" traffic was that of suppression. Suppression is based upon the idea that receivers should not transmit a NACK for a particular packet if a NACK for that packet is received from another receiver. Scalable Reliable Multicast (SRM) [27] is based upon this idea and—like token-ring protocols—pushes considerable functionality from the sender into the receivers. Since the release of SRM a number of protocols have been released that further attempt to stem traffic through localization techniques based on trees [31, 32, 33, 50, 59, 78].

Protocols that are based on this idea of decentralized responsibility—in which receivers take active responsibility for ensuring delivery of the data they need—are called "Receiver-Driven." Typically group management mechanisms are not explicit in Receiver-Driven protocols and receivers are free to join and leave sessions as they please. This idea is known as Lightweight Sessions (LWS) [43] and further implies the existence of Application-Level Framing (ALF) [19]. LWS states that transport protocols should be extensible entities that implement a minimum level of functionality while providing the means for extensions as needed by various applications.

Merging the findings from these efforts leads to the following set of base requirements for a LWS/ALF skeleton protocol that may be augmented by application designers to yield superior performance:[1]

1. Most of these requirements were presented by Van Jacobson at ACM SIGCOMM94 [43]

- Wherever possible, use multicast over unicast delivery.
- Use NACK-based reliability strategies over ACK-based ones to minimize response implosions.
- Inject randomness to lessen correlated traffic.
- Use data sources in lieu of packet sources for source identification.
- Use Application Data Unit Identifiers (ADUIs) that have real meaning for identifying the contents of a packet.
- To minimize the amount of duplication that results from repairing lost or damaged packets, allow messages to be fragmented over multiple packets.
- Assume simplex communication; in other words, give the receiver the ability to choose how much data it is willing to receive, but don't give it the power to silence the sender.
- Make receivers use dynamic prefetching, repair, and persistence strategies that are based upon the data's type, the application requesting the data, and the default-delivery mode being used by the data's originating source.

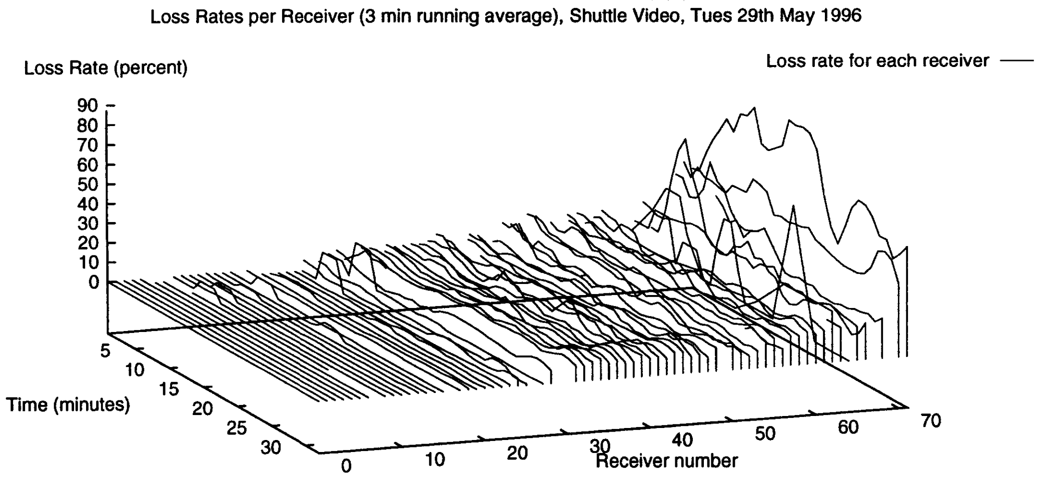
One could ignore the common skeleton described above and design a very finely tuned application-specific protocol for each application. However, solutions derived in this manner would then lose the benefits afforded by a single unified cache and make much more difficult the task of controlling the aggregate network bandwidth used by a host running multiple applications.

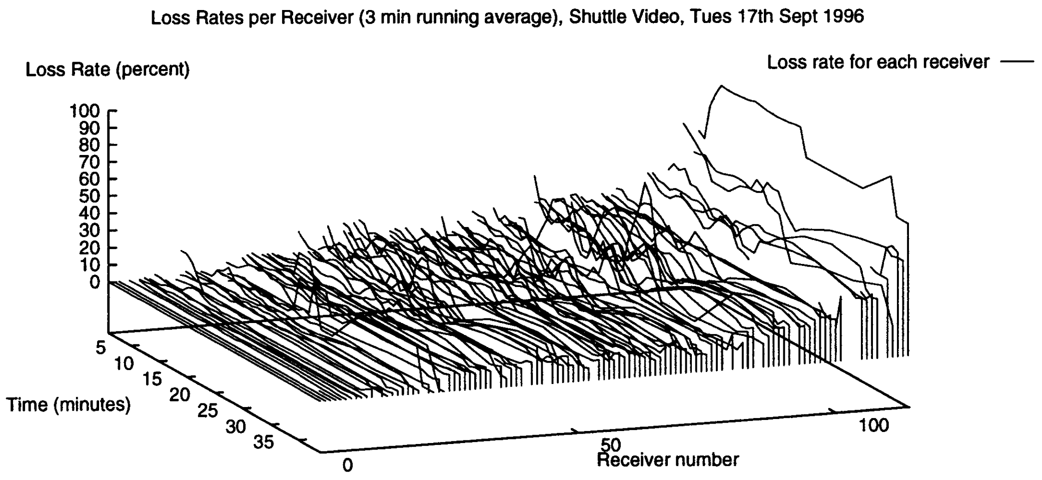
The remainder of this chapter describes the mechanics behind packet-based error recovery and then develops a new reliable multicast transport mechanism for the delivery of a single stream of data. This new mechanism merges the existing methods of Automatic Repeat reQuest (ARQ) and Forward Error Correction (FEC) with additional new localization techniques based on Administrative Scoping. Readers familiar with the concepts of ARQ as implemented within SRM [27] and FEC [29, 56, 57, 63, 64] may proceed directly to 4.2, on page 57.

4.1 Background

Given that IP multicast delivery is by definition unreliable, one must consider the inclusion of mechanisms that allow receivers to recover from lost packets in any transport protocol that uses it. The inclusion of this mechanism becomes increasingly important as redundancy is further removed from the data being delivered. One should also incorporate mechanisms that allow receivers to tune in late and catch up on missing data. These problems cannot be solved with a single mechanism grafted on to the primary delivery channel without severely restricting the scalability of the resulting system.

Consider the case where all the receivers are synchronized; that is, they all play the data as soon as they receive it. In this case, the individual links between the sender and the receivers experience varying degrees of loss that lead to the emergence of loss neighborhoods in which all the receivers behind a particular link share common loss characteristics. As the degree of branching in the routing tree between the sender and the receivers increases, so does the likelihood that at least one receiver will experience the loss of any given packet sent by the sender. Several studies [30, 77] have proven this hypothesis, and some salient observations made by Mark Handley in his Ph.D. dissertation [30] are reproduced with permission in Figures 4-1 and 4-2.





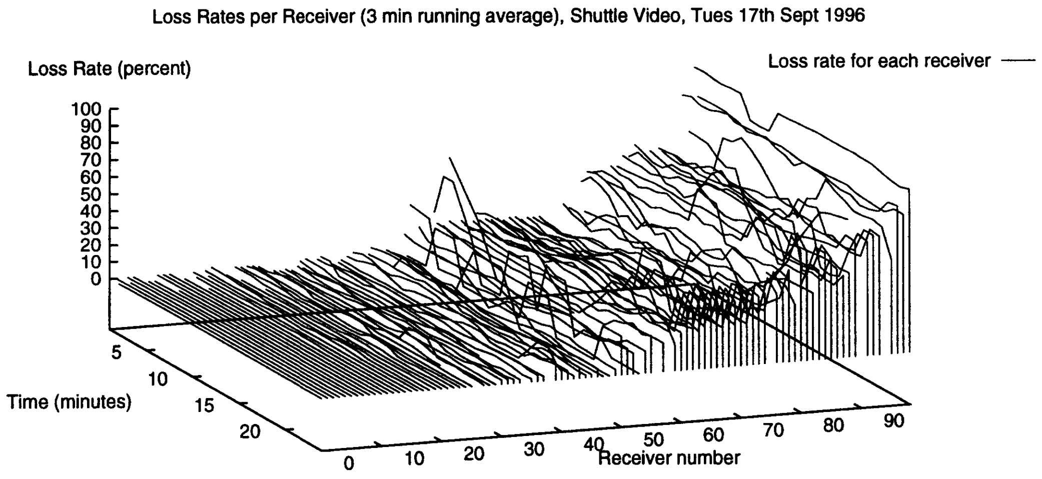
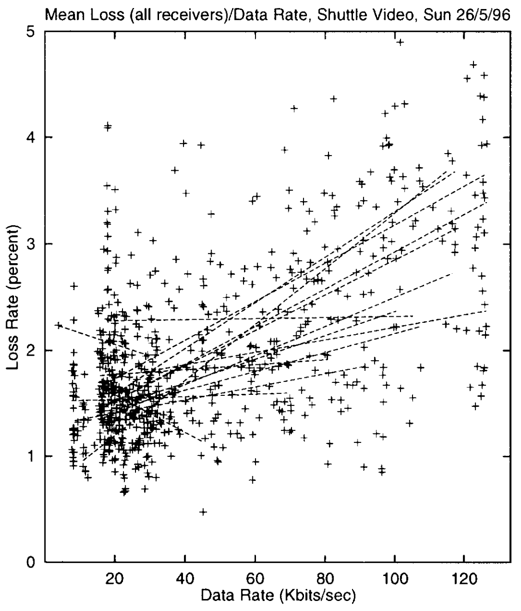


Figure 4-1 Loss rates for MBone broadcast of NASA Shuttle Mission STS-77 and STS-79.[30]



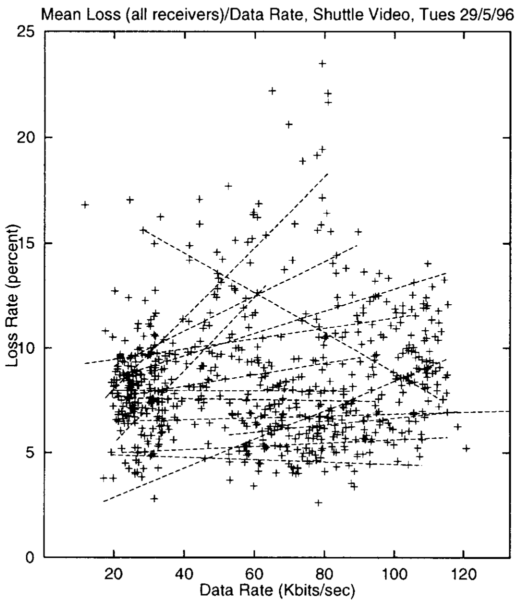


Figure 4-2 Loss versus Data Rate for MBone broadcast of NASA Shuttle Mission STS-77, 1996.[30]

Figure 4-1 shows the loss experienced by some seventy receivers listening to the NASA Shuttle Mission STS-77 during a half hour period on Tuesday 29th May, 1996 and two separate half hour periods for NASA Shuttle Mission STS-79 on Tuesday 17th September, 1996. One observes that some receivers experience no loss while others experience significant loss. While some loss correlation between receivers is evident, many experience small levels of uncorrelated loss. Figure 4-2 shows attempts to perform least squares estimate fits to losses in the STS-77 mission for a number of 50 minute intervals. The degree of loss is largely uncorrelated with the rate at which data is sent. This observation is consistent with the realization that the links measured also carried TCP traffic—which periodically congests as a part of its normal operation. Two classes of repair method are currently under study to effect the repairs in the situation described above: Automatic Retransmit reQuest (ARQ) and Forward Error Correction (FEC).

4.1.1 Automatic Repeat reQuest (ARQ)

ARQ protocols are characterized by the fact that the receivers must explicitly ask the sender—or their peers—for missing data using negative-acknowledgment (NACK) messages. Since this implies a back-channel, possibly all the way to the source, ARQ methods are prone to causing network congestion when multiple receivers send the same NACK, or subsequent repair, from different parts of the network. Congestion is usually worst on links nearest the sender. Even well-tuned ARQ protocols that use suppression subject receivers to the NACK and repair traffic of other receivers. This becomes particularly troublesome in the "crying baby" scenario in which a single receiver repeatedly makes requests for repairs that never arrive. These requests further exacerbate congestion, and cause further losses. In this case, the mechanism for repairing small packet loss bursts should include some means for limiting the number and scope of the repair requests and subsequent repairs. Such mechanisms would minimize replication and ensure that repairs go only where they are needed.

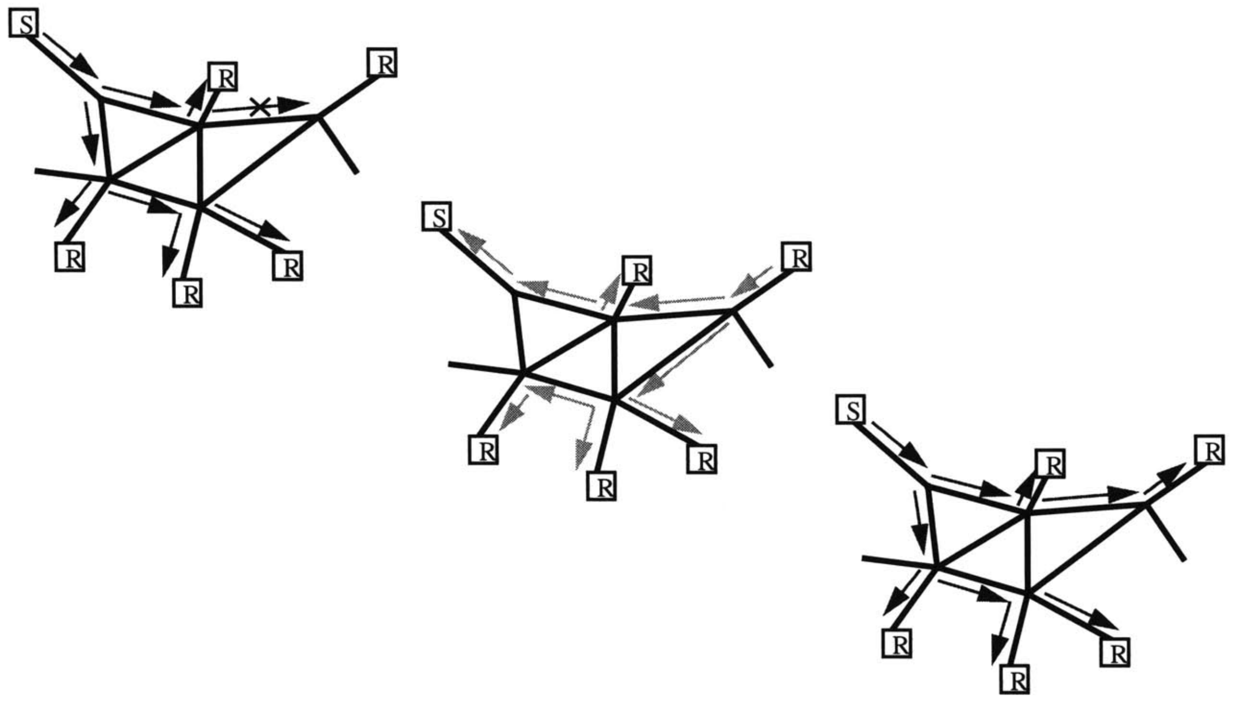


Figure 4-3 Effects of Small-Burst Loss-Recovery when using global ARQ recovery

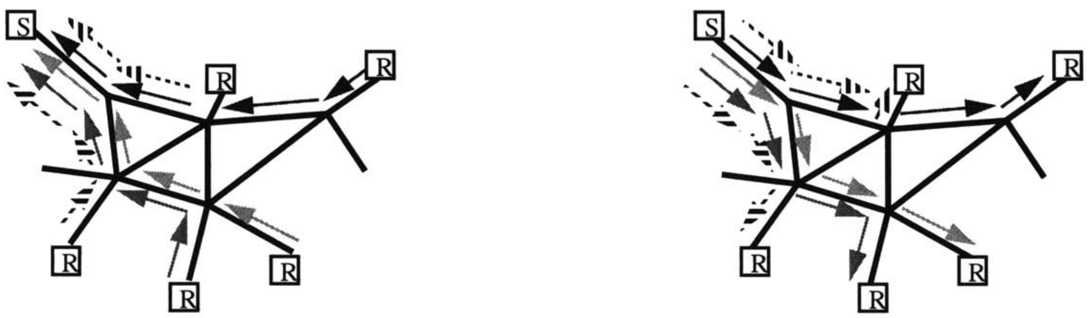


Figure 4-4 Effects of Late-Join Recovery on Server when using Unicast Transport

Much work has been undertaken in the area of reliable multicast for situations where receivers are all receiving data at the same time. Many schemes, with varying degrees of success, have been developed to limit traffic implosion. Popular models for traffic reduction involve the use of structures based on token rings [14, 74], clouds [27], and trees [31, 32, 33, 50, 59, 78]. Token-ring protocols suffer from serious scaling limitations. Cloud-based approaches increase scalability in a general manner, while tree-based approaches increase it still further—but usually require some manner of manual configuration. Many tree-based approaches incorporate suppression mechanisms borrowed from cloud-based approaches.

For that reason, it is worth examining the operation of the cloud-based Scalable Reliable Multicast (SRM)protocol in detail. The analysis of SRM that follows is largely derived from that presented by Floyd *et al.* in their seminal paper describing SRM's operation [27]. This analysis does, however, present new interpretations as to why SRM works better in some topologies than others.

SRM's operation is completely decentralized; there is no distinction between a sender and receivers during packet-loss recovery. All participants in a group are expected to take local responsibility for supplying repair packets to their neighbors in the event of a dropped packet. SRM attempts to minimize traffic implosions by using randomized timers that stagger the transmission times of the initial repair request and the subsequent repair response.

The duration of both the loss detection and reply response timers for Host A in SRM are set from a uniform distribution over a window that has the following general form:

$$2^i[A_1 d_{S,A}, (A_1 + A_2) d_{S,A}] \qquad \text{Eqn 4-1}$$

Where;

A_1, A_2 define the start and length of the window respectively.

$d_{S,A}$ defines Host A 's estimate of the one way delay itself to the Source S

i is the iteration number of the NAK or the response

It is useful to consider how changes in A_1 and A_2 affect the number of duplicate repair requests and repairs for two simple network topologies, chain and star. As reported in [27] the study of these two topologies is useful since they are the building blocks from which more general tree structures are derived.

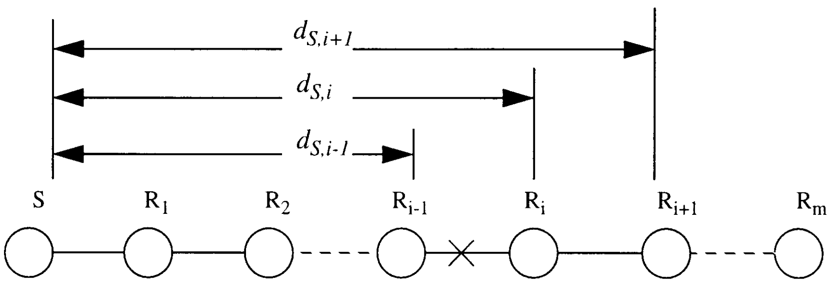


Figure 4-5 SRM in a Chain Topology

The chain topology in Figure 4-5 shows a single source transmitting packets to a number of receivers that are chained together. Packets in transit from the source S to a receiver R_m experience $d_{S,m}$ units of delay. Furthermore, a packet has just been dropped in transit between R_{i-1} and R_i. Let the next packet from S cause R_i to be the first receiver to detect the loss at time t, with receivers to the right of R_i detecting the loss at $t + (d_{S,m} - d_{S,i})$ where $m > i$.

When the A_2 parameter is set to zero for all the receivers, receiver R_i will multicast a NACK at time $t + A_{1,i}d_{S,i}$ that is received by R_{i-1} at $t + A_{1,i}d_{S,i} + (d_{S,i} - d_{S,i-1})$. Receivers to the right of receiver R_i will receive the NACK at $t + A_{1,i}d_{S,i} + (d_{S,m} - d_{S,i})$. For them to suppress their own NACKs, this time should be before the time at which their own NAK timers will expire: $t + (d_{S,m} - d_{S,i}) + A_{1,m}d_{S,m}$. The following inequality can be derived that, when true, guarantees NACK suppression.

$$A_{1,i}d_{S,i} < A_{1,m}d_{S,m} \qquad \text{Eqn 4-2}$$

A trivial solution for this inequality results naturally from the fact that $d_{S,i} < d_{S,m}$ when A_1 is set to a constant value for all receivers. Since setting A_2 to zero in Eqn. 4-2 results in a mechanism that guarantees the suppression of NACKs in receivers downstream of receiver R_i, it is a relatively simple task to show that a similar result can be achieved for repair responses as well. Since the response time is determined solely by the value of A_1, it need not be large if the network topology is indeed a chain. A_1 should be set so that the difference between the $A_1d_{S,m}$ products of successive receivers in the chain is just sufficient to allow for the time taken between receiving the first and last bytes of a NACK packet. The fact that both duplicate NACKs and repairs are always suppressed has led to this mechanism being called *deterministic suppression* when applied to chain topologies.

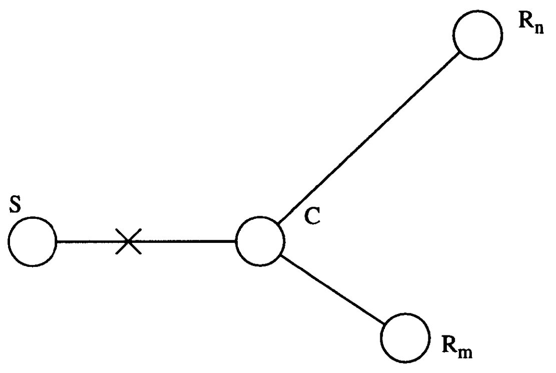


Figure 4-6 SRM in a Star Topology

The second topology examined is the star topology shown in Figure 4-6. In this example, a single source is connected to central node, C, which is not itself a receiver. C does act as a transit point to nodes R_m and R_n, which are receivers. As with the chain topology, a single packet has just been dropped and the receivers are about to initiate their NACK timers.

Assuming that the source sends the next packet at time t, receiver R_i will detect the loss of the previous packet at $t + d_{S,i}$. Receivers equidistant from the central node will detect the loss at the same time, therefore negating any suppression effects that result from A_1. Consequently, A_1 is now set to zero for the rest of this analysis, and NACK suppression is governed solely by the value of A_2—the width of the window over which the NACK timer is randomized.

With these conditions in place, receiver R$_m$'s NACK timer will expire and cause a NACK to be transmitted at some random time between $t + d_{s,m}$ and $t + d_{S,m} + A_{2,m}d_{S,m}$. This NACK will be received by receiver R$_n$ some time between $t + d_{S,m} + (d_{S,m} + d_{S,n} - 2d_{S,C})$ and $t + d_{S,m} + A_{2,m}d_{S,m} + (d_{S,m} + d_{S,n} - 2d_{S,C})$. Now consider the situation in which receiver R$_m$'s NACK timer expires at the earliest possible time, and receiver n 's NACK timer expires at the latest possible time. In this situation, receiver R$_m$ will transmit a NACK at $t + d_{s,m}$ that will be received by receiver R$_n$ at $t + d_{S,m} + (d_{S,m} + d_{S,n} - 2d_{S,C})$. Receiver R$_n$'s own NACK timer will expire at $t + d_{S,n} + A_{2,n}d_{S,n}$. For receiver R$_n$'s NACK to be suppressed, the following must hold true

$$2d_{S,m} - 2d_{S,C} < A_{2,n}d_{S,n} \qquad \text{Eqn 4-3}$$

This equation can also be written as

$$2d_{C,m} < A_{2,n}d_{S,n} \qquad \text{Eqn 4-4}$$

In qualitative terms, this equation states that receiver R$_n$'s NACK timer must be set sufficiently far enough in the future to allow for the transit of the packet that causes the loss detection of its predecessor to travel from C to R$_m$, and for the resulting NAK to travel back to C. It is possible to set A_2 for each receiver so that this inequality is never true. This corresponds to the situation where R$_n$'s NACK timer expires before the reception of NACKs sent by other receivers. Thus, when A_2 is set in this manner for a star with G receivers $(G - 1)$, duplicate NACKs will result [27].

Now suppose the value of A_2 is increased so that the inequality is true. In this case, suppression may occur. One notes that this is not guaranteed since the initial conditions introduced at the beginning of this analysis assumed that receiver R$_m$'s NACK timer expired at the earliest possible time, and the NACK timer of all remaining receivers expired at the latest possible time. The likelihood of suppression increases with A_2. In [27]—where it is further assumed that the delay contributed by each link d is one—the number of NACKs resulting from a loss is shown to be approximately $1 + (G - 2)/A_2$. Furthermore, [27] shows the average delay until the first timer expires is $(2A_2)/G$ for the case where the star consists of G external nodes (receivers plus the sender) for both results. Deterministic suppression is not possible for the star topology. One cannot guarantee that a receiver R$_n$ will not transmit a NACK before receiving the NACK sent by a receiver R$_m$. For this reason, this method of NACK suppression is called *probabilistic suppression.*

In practice, multicast distribution trees are seldom pure chains or stars. Instead, they tend to be a combination of the two. The local topology will vary from receiver to receiver, a condition that results in different emphasis on the relative roles of the A_1 and A_2 parameters according to whether the local topology is more like a chain or a star. The NACK and repair timers should also have different values for A_1 and A_2 to allow for network asymmetries and varying receiver capabilities. For all these reasons, SRM is implemented with separate values for the NACK and repair timers: C_1 and C_2 for the NACKs and D_1 and D_2 for the repairs. These values can be adjusted during data transfer according to the number of duplicate NACKs or repairs received. The basic algorithm for the dynamic adjustment of the timers is essentially the same for both the NACKs and repairs, and is given in [27].

```
After sending a request
   decrease start of request timer interval
Before each new request timer is set:
   if request sent in previous rounds, and any
   duplicate requests were from farther away:
      decrease request timer interval
   else if average duplicate requests is high
      increase request timer interval
   else if average duplicate requests is low
   and average request delay is too high
      decrease request timer interval
```

Figure 4-7 Dynamic adjustment algorithm for request timer interval [27]

```
After a request timer expires or is first reset:
   update ave_req_delay
After sending a request:
   $C_1$ -= 0.1
Before each new request timer is set:
   update ave_dup_req
   if closest_requestor on past requests:
      $C_2$ -=0.1
   else if (ave_dup_req > AveDups):
      $C_1$ += 0.1
      $C_2$ += 0.5
   else if (ave_dup_req < AveDups - $\varepsilon$):
      if (ave_req_delay > AveDelay):
         $C_2$ -= 0.1
      if (ave_dup_req < 1/4):
         $C_1$ -= 0.05
   else $C_1$ += 0.05
```

Figure 4-8 Dynamic adjustment algorithm for request timer parameters [27]

```
Initial Values
   $C_1 = 2$
   $D_1 = log_{10}G$
   $C_2 = 2$
   $D_2 = log_{10}G$
Fixed Parameters
   $MinC_1 = 0.5$; $MaxC_1 = 2$
   $MinC_2 = 1$;   $MaxC_2 = G$
   $MinD_1 = 0.5$; $MaxD_1 = log_{10}G$
   $MinD_2 = 1$;   $MaxD_2 = G$
   AveDups = 1
   AveDelay = 1

where
   G = number of nodes participating in the session
```

Figure 4-9 Parameters for the adaptive algorithms [27]

While SRM largely succeeds in its aim to reduce the number of duplicated NACKs and repairs, it does incur some penalties. First, suppression is not guaranteed; redundant traffic is only reduced to within some probabilistic bound that decreases as the timer constants increase. Second, the use of randomized timers results in an increased recovery delay that is some multiple of the round-trip time (RTT) from a receiver to the originating source. This situation worsens as the depth of, and hence transit delays on, downstream branches of star nodes within the distribution tree increase. In addition, the use of RTTs means that each session member needs to determine the transit delay to every other member. This leads to significant amounts of session traffic and state for large sessions. Finally, all receivers take part in the recovery process for every packet lost by every receiver, regardless of whether a particular receiver is in a part of the distribution tree affected by loss or not.

In response to these shortcomings, several efforts have been made to extend the concepts popularized by SRM using tree structures to provide better scoping of repair traffic. Log Based Receiver Reliable Multicast (LBRM) [33] and the Reliable Multicast Transport Protocol (RMTP) [50, 59] construct a static hierarchy of logging servers or Designated Receivers (DRs) respectively, that handle repairs on a local basis. LBRM achieves reliable delivery through the use of localized NACKs sent to specific logging servers, while RMTP achieves the same through the creation of a shared ACK tree that links the set of DRs. Both LBRM and RMTP rely on all members within the statically configured tree to operate correctly.

This restriction is relaxed by Yatkavar, Griffoen, and Sudan's Tree-based Multicast Protocol (TMTP) [78]. TMTP also arranges receivers in a hierarchy; however, it constructs the hierarchy dynamically. The resulting tree is built via a series of expanding ring searches in which a selected receiver—called a Domain Manager (DM)—in each region attempts to discover a parent DM higher up the tree. Should more than one potential parent DM respond to a new DM's request, the closest parent DM is chosen on the basis of hop count. Each receiver is required to keep the hop count distance to its nearest DM and each DM is required to keep the hop distance to its farthest child. The hop distances are used to limit the scope of subsequent requests and replies for repairs. TMTP improves upon LBRM and RTMP; however, its repair traffic can still be exposed to parts of the tree that need not be included in the repair process. TMTP also constructs the distribution tree using an expanding ring search that can lead to suboptimal trees in which a downstream receiver is chosen as a parent.

Another method that supports dynamic tree construction is described in Hoffman's papers on the "local group concept" [31, 32]. Sessions using this concept are subdivided into groups. Each group contains a Local Group Controller (LGC) responsible for retransmitting repairs to its group. Nodes wishing to join a session search for the nearest LGC and join its group. A number of metrics, such as distance or loss rate, may be used to determine "closeness." The decision as to which metric to use is made by a Group Distance Service (GDS) present in each node. Should a node's search for an LGC fail, it elects itself as an LGC and creates a new group.

Perhaps the most aggressive attempt to limit the scope of NACKs and the subsequent repairs is that of "directed multicast" (also known as subcasting) put forward by Papadopoulos *et al.* in [58]. Directed multicasting is based upon the concept of pivot points that give routers the ability to selectively multicast packets to some parts of the distribution tree and not others. This feature allows receivers in need of a repair to direct their NACKs upstream over the link that experienced the loss to a single receiver known as a replier. Repliers subcast repairs back down over the link. It is important to note that routers only send repairs over links that carry NACKs to the upstream receiver. Repairs are not propagated to other downstream links or upstream toward the source.

Directed multicast is much more efficient in suppressing excess traffic than SRM, since a single receiver sends a single repair that is only seen by those receivers that need it. However, its use does raise a number of issues that have yet to be addressed. The process of determining which receivers are repliers and to which parts of the distribution tree they reply requires an election process, the results of which must be stored as state in the routers. In addition, it is not clear that this extra state will allow "directed multicast" to scale when deployed on a widespread basis. Finally, the fact that "directed multicast" requires modifications to the routers themselves raises questions about its ability to find widespread deployment.

4.1.2 Forward Error Correction (FEC)

FEC methods [29, 56, 57, 63, 64] take a different approach for effecting repairs. Instead of waiting for receivers to detect losses and then ask for repairs, FEC methods assume that some faction of packets will always be lost and that it is more efficient to encode and transmit the data with some additional redundancy so that receivers can reconstruct the original packets even if some packets are lost in transmission. Usually FEC methods work by placing k packets into a group and then creating additional packets from them so that the total number of packets in the group becomes n where $n > k$. This group of n packets is transmitted to the receivers, each of whom receivers some number of packets less than n, as shown in Figure 4-11.

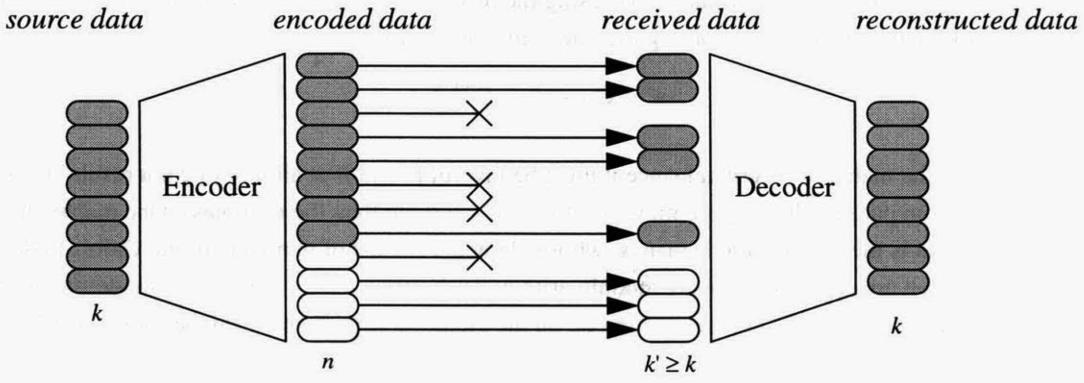


Figure 4-10 Graphical Representation of FEC operation [63]

Provided a receiver receives at least k packets from the group, it can reconstruct the k original packets. In cases where no original packets are lost, the receiver simply discards the extra packets, and the amount of overhead at receiver is nil. When original packets are lost, the amount of work required to reconstruct a group's missing packets is roughly proportional to the number of packets that must be reconstructed [63].

The exact amount of work depends on the method used for encoding and decoding the redundant packets. A simple method that works for the case where $n = k + 1$ is to the create the redundant packet by 'XOR'ing the k originals together. Unfortunately, this is a trivial case that does not work for situations where more than one packet must be reconstructed. When this occurs, one must employ a more complicated method that is capable of generating multiple orthogonally redundant packets. Methods that generate multiple redundant packets usually involve placing the original k packets into a vector x which is then multiplied by a matrix G.

$$\underline{y} = G\underline{x} \qquad \text{Eqn 4-5}$$

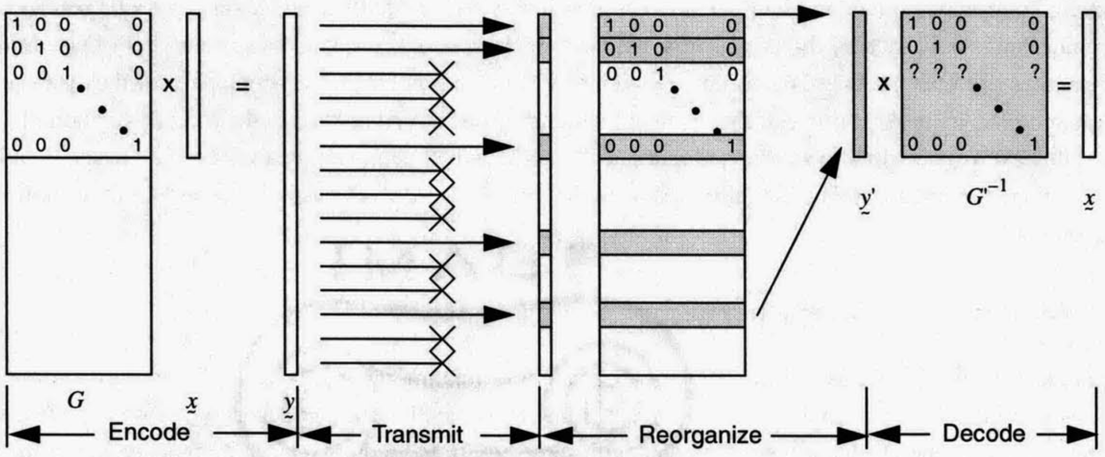


Figure 4-11 Detailed FEC operation

The matrix G in Eqn 4-5 is constructed by starting with an identity matrix and adding rows that combine the various original packets in differing ways. The decoding process is performed by first constructing a new vector $\underline{y}'$ using k of the k' received packets and a new matrix G' using the rows of G that were used to construct the elements of $\underline{y}'$. The original packets are recovered by solving the linear equation below

$$\underline{y}' = G'\underline{x} \rightarrow \underline{x} = G'^{-1}\underline{y}' \qquad \text{Eqn 4-6}$$

The choice of the matrix G is crucial since it must be invertible and if at all possible not result in expansion of the data (i.e., result in packets that require more bits to represent them than the originals). One matrix that satisfies both of these criteria is the Vandermonde matrix, whose elements consist of members of the Galois Field $GF(p)$, where p is prime that are operated on using modulo arithmetic. Further details of how the Vandermonde Matrix is constructed from $GF(p)$ and how modulo arithmetic on the elements is performed can be found in [63].

[63] also reports that when large packets in excess of 100 bytes are used, the majority of the work in building and reconstructing the redundant packets lies in the multiplication and not in the inversion of G'. This observation is important since it highlights two important ideas. First—to minimize the work done by the sender—$n - k$ should be kept to a minimum; and second—to minimize the work done by the receiver—original packets should be used in lieu of redundant packets during reconstruction.

To summarize, while FEC does remove the need for receivers to keep track of the state used to prevent NACK implosions, it does incur some potentially significant costs:

- Latency is increased since packets are now processed in groups instead of individually.
- Extra processing is required to encode and decode the data.
- To guarantee data delivery, n may have to be much larger than k.
- Redundancy can result in extra bandwidth to ensure delivery.

4.2 Scoped Hybrid Automatic Repeat reQuest with Forward Error Correction (SHARQFEC)

Recent work in [29, 57] has attempted to limit the number redundant packets processed during the transmission process by combining FEC with ARQ. The merger of these two techniques yields better performance than either method on its own. From this beginning, SHARQFEC is developed.

4.2.1 Existing non scoped techniques

To understand SHARQFEC's design, it is easiest to start with a simple example and examine the traffic flow generated by standard ARQ and nonscoped FEC protocols. Consider the delivery tree shown in the top of Figure 4-12. A single source at the root of the tree delivers data to a number of receivers. Different branches in the tree are subject to loss at different rates, with some being virtually lossless and others experiencing considerable losses due to congestion. The total loss between the source and a given node is calculated by compounding the loss rates of each link between the source and that node.

$$\mathit{Total\ Loss} = 1 - \prod \mathit{Pr(No\ loss)} \forall \mathit{links\ between\ source\ and\ node} \qquad \text{Eqn 4-7}$$

Furthermore, if one assumes that losses are independent, the probability that all receivers will receive a given packet is

$$\mathit{Pr(all\ receivers\ receive\ a\ given\ packet)} = \prod \mathit{Pr(No\ Loss)} \forall \mathit{links\ in\ the\ delivery\ tree} \qquad \text{Eqn 4-8}$$

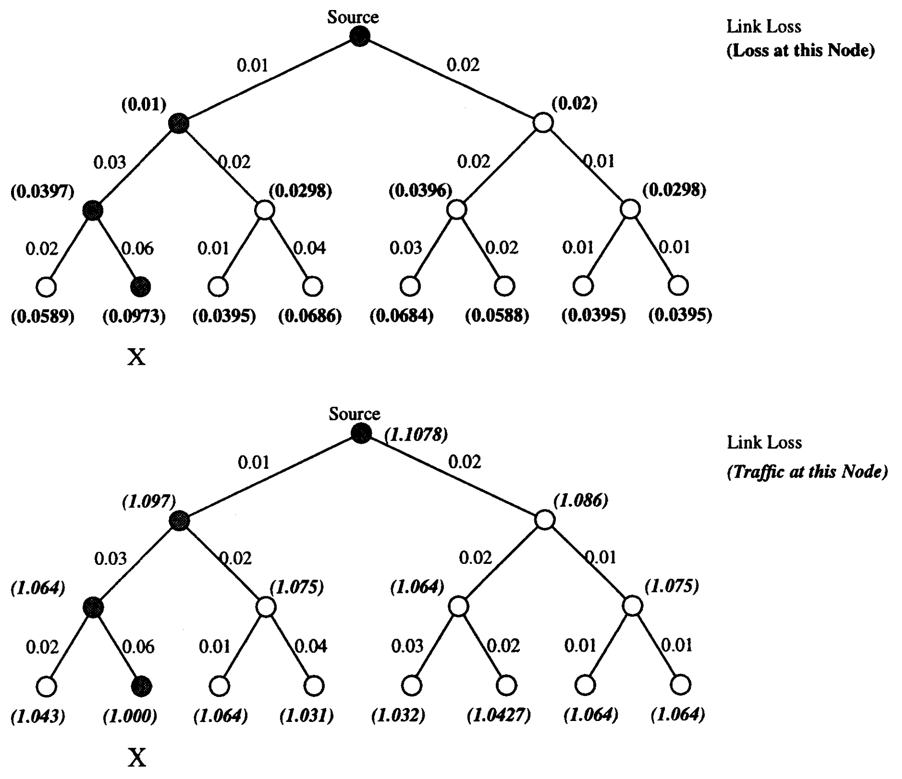


Figure 4-12 Example Delivery Tree/ Normalized Traffic Volume for Nonscoped FEC

For the tree shown in Figure 4-12, this equation yields a probability of 27.0%. In other words, every time the source multicasts a packet there is a better than 70% probability that at least one receiver will fail to receive it. In real world situations, where there are many more receivers connected by trees containing considerably more links, this figure has been shown to be close to 100% [30]. This simple analysis exposes the global nature of simple non-scoped ARQ techniques. Either the source must transmit each packet multiple times, or, if local recovery is being used, every session member must take part in the repair of every packet transmitted. This problem is exacerbated when the repairs themselves are lost.

The other popular approach for effecting repairs has been to send packets in groups and to then send additional packets created with Forward Error Correction (FEC) techniques. As each FEC packet has been generated from the original packets in the group, it can repair the loss of any one packet. Multiple missing packets are repaired using multiple FEC packets. This method reduces the volume of repair traffic considerably. Now the sender can add just enough redundancy to compensate for the greatest loss experienced by any one receiver.

Unfortunately, losses tend to be unevenly distributed, making it difficult to determine how much redundancy is needed. Consequently, receivers experiencing higher losses usually end up requesting sufficient repairs to cover their own losses. Thus, the additional FEC traffic needed to repair this one receiver is sent needlessly to receivers experiencing lower loss. The tree on the bottom of Figure 4-12 shows the normalized traffic volume seen at each node when the source adds enough redundancy to compensate for receiver X, which experiences 9.73% loss.

4.2.2 Local Recovery through administrative scoping

The ability to restrict the scope of repair traffic is a key tool for increasing the scalability of reliable multicast protocols. In the previous section, it was shown that even hybrid ARQ/FEC techniques result in additional traffic on lightly congested links and on links near the source. This task can be achieved through the use of a hierarchy of nested administratively scoped regions.

This mechanism has two advantages over other hierarchical mechanisms, such as the installation of dedicated receivers [33, 59] or modifying the routers to support selective multicast [58]. First, the system must be robust without being inefficient. A reliance upon specially configured receivers creates the potential for catastrophic failure. In addition, hierarchical schemes that elect designated receivers dynamically and then use combinations of Time-to-Live (TTL) scoping and multiple multicast groups have the potential to be inefficient, since traffic still has the potential to seep outside the hierarchy. Second, the resulting system must be deployable in a quick, easy, and scalable fashion. Protocols that require router modifications, while possibly being the most efficient, become much less efficient when run on unmodified routers. Furthermore, these protocols have a tendency to create additional state in the router, and therefore are subject to some concern about their ability to scale when numerous sessions are present. This concern is magnified for routers in the backbone,

The creation of a hierarchy of administratively scoped zones affords greater localization through several means. Consider the example in Figure 4-12 from the previous section, this time overlaid with a three level hierarchy of zones as shown in Figure 4-13. In this scheme, there is a single data channel with maximum scope, and an additional repair channel for each zone. Data is transmitted over the data channel and repairs are selectively added within each zone to compensate for the losses between the current zone and the next zone with the highest loss rate. Thus, the source in Figure 4-13 need only add sufficient redundancy to guarantee delivery of each group to receiver Y, which will in turn add just enough redundancy to ensure delivery of each group to receiver Z.

As the loss rates will vary over time, the level of redundancy will change from group to group. Should too much redundancy be injected at one level in the hierarchy, receivers in subservient zones will add less redundancy. Should greater losses occur than expected, receivers will use ARQ to request additional FEC packets as needed from larger scoped zones. Receivers perform suppression on all NACKs as appropriate.

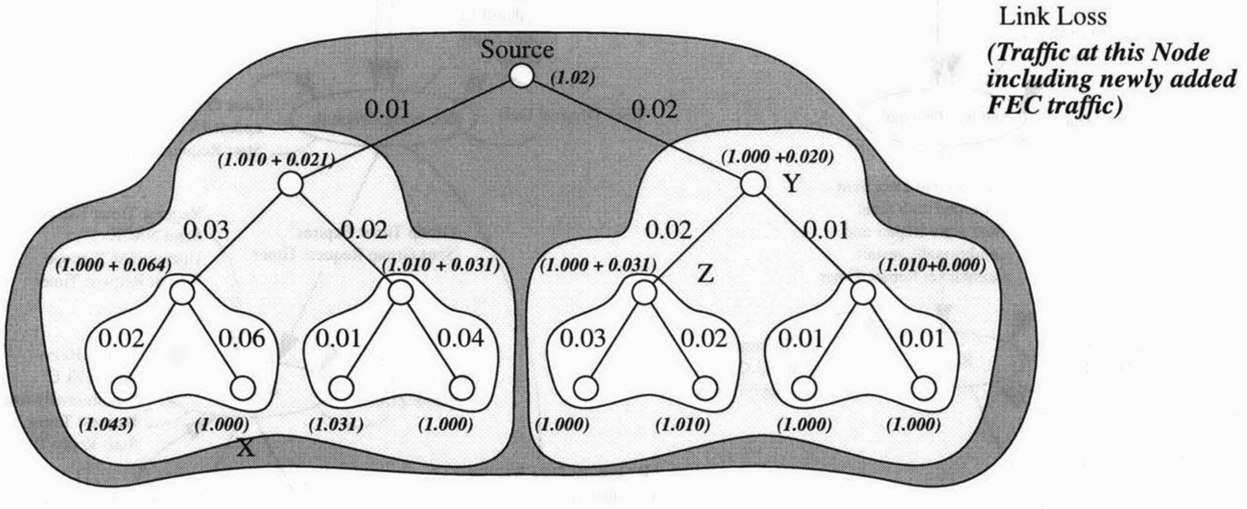


Figure 4-13 Redundancy Injection using FEC

This scheme requires the election of a representative for each zone. This receiver, called a Zone Closest Receiver (ZCR), should be the zone's closest receiver to the ZCR of the next-largest zone. While the existence of ZCRs does create singular points of failure in the distribution tree, the ability of receivers to increase the scope of their NACKs without reconfiguring the hierarchy minimizes the consequences of ZCR failure. When the scope is increased in this manner, all receivers within the higher scoped region participate, and not just the ZCR at the next highest level. Thus, the system merges the benefit of efficiency afforded by hierarchy with the robustness afforded by peer-to-peer recovery. As the process by which receivers determine the ZCR for each zone has implications for session traffic, the process for electing ZCRs is deferred for discussion later in Section 4.2.6, on page 69.

4.2.3 SHARQFEC Algorithm Description

Packet groups are delivered in a two-phase process; a loss-detection phase and a repair phase. In the loss-detection phase, the source multicasts the original packets over the data channel and receivers announce the number of packets lost to the other receivers within their zone. Once all the packets within a group have been sent, the repair phase begins.

During the repair phase, receivers attempt to effect repairs as needed to other receivers within their zone. Repairs are effected in two ways. First, receivers may request them in a manner similar to ARQ methods, with the small modification that the NACK now indicates how many additional FEC packets are needed to complete the group and not the identity of an individual packet. Second, Zone Closest Receivers in each zone may automatically inject additional FEC packets into the stream without waiting for the NACKs to arrive. The number of FEC packets injected is determined from previous losses and decays over time. Should the number of additional FEC packets injected by a Zone Closest Receiver prove insufficient, a receiver may request further repairs using the first method.

This two-phase delivery process can be represented in the form of state diagrams, as shown in Figure 4-14. A more detailed point-by-point description of the algorithm used by SHARQFEC senders and receivers follows:

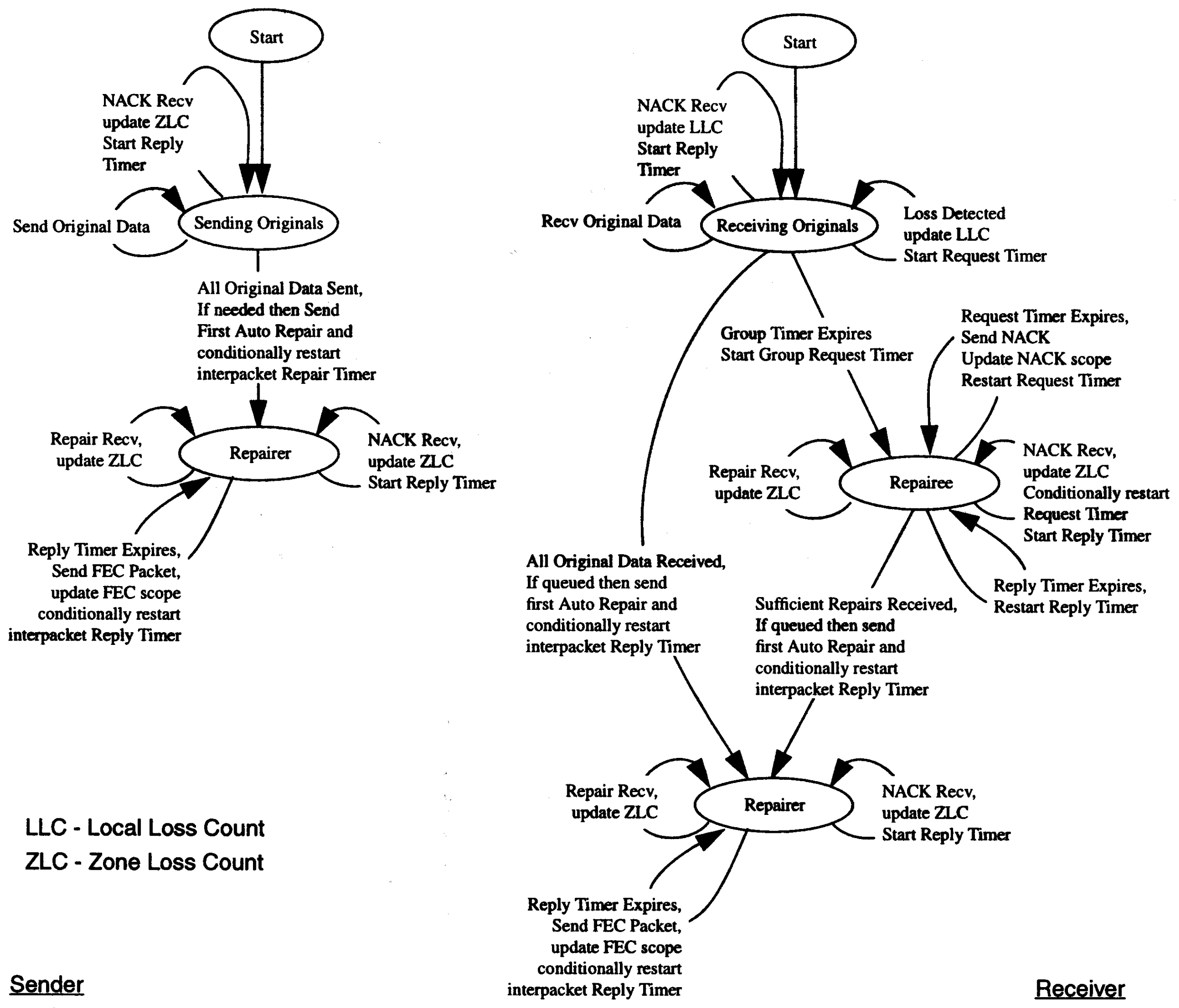


Figure 4-14 State Transition Diagram for the SHARQFEC sender and receiver

Loss Detection Phase (LDP)

- The sender divides its packet stream into groups that are transmitted sequentially along with a small number of redundant FEC packets. (The decision as to how many redundant packets to send will be explained shortly.) After sending the last unrequested FEC packet, the sender automatically enters the repair phase.
- Receivers take note of the order in which packets arrive and maintain a count of how many original packets were lost in transit, the Local Loss Count (LLC). In addition, each receiver also maintains state about the maximum number of losses experienced by any one receiver in each of the n administratively scoped repair zones of which it is a member. The maximum LLC for a zone n is known as the Zone Loss Count, ZLC(n).

- Upon the reception of a packet in a new group, receivers estimate the time by which all packets within the group should be received. They then set a Loss Detection Phase (LDP) timer to expire at this time. This estimate is calculated on the basis of the inter-packet arrival time, which is refined on a group-by-group basis. Initially, the inter-packet arrival time is itself estimated using the advertised channel bandwidth and the advertised size of the data packets.
- When a receiver detects the loss of one or more original packets, it increments its LLC according to the number of missing packets detected. Should the LLC exceed the ZLC for the current scoped repair zone, it starts a request timer in the same manner as specified by the SRM protocol with fixed timers. That is, delay is chosen on the uniform interval $2^i[C_1 d_{S,A}, (C_1 + C_2)d_{S,A}]$, where $C_1 = 2$, $C_2 = 2$, and $d_{S,A}$ is the receiver A's estimate of the one-way transit time for traffic between the source, S, and itself. The i variable is initially set to 1 and is incremented by 1 every time a NACK that does not increase the appropriate ZLC(n) is received. Any time a repair arrives, i is reset to 1.
- If a receiver's request timer expires, the receiver transmits a NACK containing its LLC (which will become the new ZLC for that scope zone), the greatest packet identifier seen, and the number of repair packets needed. NACKs are transmitted to the repair channel with scope equal to that of the smallest scope zone for a particular partition, subject to the restriction that if the originating source is a member of the smallest partition, the repair channel corresponding to the largest scope is used instead.
- Receivers will suppress NACKs if their LLC is less than or equal to the receiver's own estimate of the ZLC for a zone. NACK suppression will also take place should a receiver receive a NACK that increments the ZLC so that the LLC is less than or equal to the new ZLC value for a zone.
- When a receiver receives a NACK, it updates the appropriate ZLC and then checks to see if the NACK's last received packet identifier causes the detection of any further lost packets. If additional packets (either original or FEC) are missing, the receiver follows the operations outlined immediately above.
- In addition to using received NACKs to update the ZLC information, receivers also use NACKs to maintain information about the number of repairs needed by other receivers. This information is then used to speculatively schedule replies, with the expectation that by the time the timer expires a sufficient number of repairs will have been received to complete the packet group. Increases to the number of speculatively queued repairs do not reset the reply timer. The reply timer is set in the same manner as specified by the SRM protocol with fixed timers. That is, the delay is chosen on the uniform interval $[D_1 d_{S,A}, (D_1 + D_2)d_{S,A}]$, where $D_1 = 1$, $D_2 = 1$, and $d_{A,B}$ is the receiver B's estimate of the one-way transit time for traffic between the sender of the NACK, A, and itself. Note that the back-off mechanism used by SRM repair timers is omitted for SHARQFEC.
- Speculatively queued repairs are dequeued upon the reception of repair packets. As repairs from larger zones will also be received by any smaller administratively scoped zones known to a session member, speculatively queued repairs for smaller zones will be decremented as well.
- When a receiver sends a NACK, it includes the largest known packet identifier. Similarly, when a session member starts to send one or more repairs the member includes what will be the new highest packet identifier. These two measures are designed to minimize the likelihood of the same repair packet being sent twice by different repliers.
- When a receiver's LDP timer expires, or it receives enough packets to reconstruct the group, it enters the repair phase. A sender enters the repair phase immediately upon the completion of sending the original packets.

Repair Phase (RP)

- After transmitting the last data packet of the group, the sender enters the repair phase, immediately generating and transmitting the first of any queued repairs in the largest scope zone. Since this repair will also be received by any smaller administratively scoped zones known to the sender, the pending repair queues for these zones are shortened appropriately. The sender then starts its internal repair reply timer with a short interval designed to spread out any subsequent repairs. In the simulations reported later in 4.2.8, on page 74, the interval was set to half that of the inter-packet interval observed for successive data packets.
- Upon the reception of packets sufficient to reconstruct the group, ZCRs likewise become repairers and generate and transmit the first of any additional queued repairs to the zone for which they are responsible.
- Once a non-ZCR receiver successfully receives sufficient packets to reconstruct all the packets within a packet group, that receiver becomes a repairer.
- Upon receiving a NACK, the sender and repairers update their state to reflect the number of outstanding repairs needed to complete the administrative scope zone defined within the request. They also update their maximum packet identifier to minimize the likelihood that any repair packets they subsequently transmit will not already have been sent by another receiver.
- Before sending any repairs, non-ZCR repairers perform suppression using a reply timer in the same manner mentioned earlier. As replies may consist of multiple repairs, a repairer will only cancel its reply timer when a sufficient number of repairs has been received to effect the entire repair.
- Should a repairer's reply timer expire before the reception of enough repair packets, it follows the same set of steps outlined above for senders entering the repair phase.
- Should a repairee detect that it has lost a repair and that further repairs will be needed, it transmits a new NACK indicating that it needs more packets, subject to the same suppression rules described in the Loss Detection Phase. The scope of successive attempts will be increased after two attempts at each zone to the next-largest scope zone, until the largest scope zone is reached.

One aspect of SHARQFEC's operation not covered by this set of rules is the determination of how many automatic repairs a ZCR should add to its zone upon having successfully received a group's packets. If one makes the assumption that losses (however bursty in nature) tend to vary slowly from group to group, then a receiver can estimate the number of automatic repairs to send by applying a simple exponential weighted moving average (EWMA) filter to the ZLC for previously received groups. As the automatic repairs will suppress NACKs in situations when the ZLC for the current group is less than the predicted ZLC, the EWMA filter will use the receiver's LLC in cases where no NACKs are received to indicate the true ZLC.

The coefficients used within the test simulations were as follows:

$$\begin{aligned} {}_{dicted}(n) &= 0.75 \times zlc_{predicted}(n-1) + 0.25 \times zlc(n) && \text{if true ZLC known} \\ &= 0.75 \times zlc_{predicted}(n-1) + 0.25 \times zlc(n) && \text{if true ZLC unknown} \end{aligned}$$ Eqn 4-9

Another problem in estimating the ZLC for subsequent groups is knowing when to measure the ZLC for a group during the repair phase. Measure it too early, and the value might be too low, as a NACK might still be outstanding from a distant receiver. Measure it too late, and the measurement's accuracy will be lessened. This problem can be solved by noting that the maximum delay a receiver's NACK will experience is equal to the Round-Trip-Time (RTT) between itself and the ZCR plus the maximum delay due to its suppression timer. Thus, a ZCR is guaranteed of learning the true ZLC for a zone if it waits for a period equal to two and half times the RTT between itself and the most distant known receiver.

4.2.4 Administratively Scoped Session Management

One of the major shortcomings of using SRM-like timers for suppression is the requirement that each session member have an estimate of the Round-Trip-Time (RTT) to every other member. This requirement results in $O(n)$ state per receiver where n is the number of receivers in the session. However, the real problem is that $O(n^2)$ session traffic is required to maintain this state. Thus, attempts to scale beyond more than a few hundred members will result in catastrophic congestion.

The key to solving this dilemma is to realize that when a loss occurs, the ensuing NACK is likely to originate from a receiver near to the link on which the loss occurred. Furthermore, the receiver generating the repair is likely to be upstream from the receiver that sent the NACK. This means that it is more important for session members to know the RTTs to nearby members than distant ones. With this realization, it becomes possible to reduce the amount of session traffic and state by constructing a session mechanism that allows receivers to collect detailed information about other nearby receivers and a summarized view of more distant receivers.

Administratively scoped repair zones can assist in the reduction of this state information. Consider the figure below. Here, a cloud of receivers receives data from a single source, node 1, that subscribes only to the largest administratively scoped repair zone, Z0. Receivers below the source subscribe to one of two intermediate scoped zones: Z1 and Z2. Finally, the receivers farthest from the source subscribe to one of four locally scoped repair zones: Z3, Z4, Z5, and Z6.

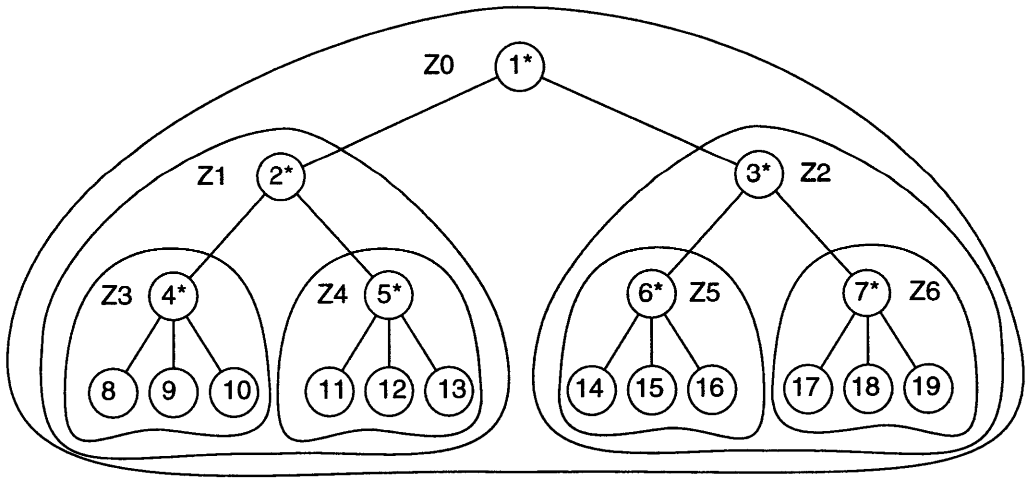


Figure 4-15 SHARQFEC Hierarchical Scoping

Now consider the receivers in zone Z4. Thanks to the SRM-like timers, Zone Z4's receivers are unlikely to receive repairs originating from receivers in zone Z3, and are even more unlikely to receive repairs originating from receivers in zone Z2. The majority of repairs generated in response to a Z4 receiver's NACK will originate from the source, other receivers within zone Z4, and receivers between the source and Z4. Thus, it is important that the receivers have very good estimates of the RTT to the source and other receivers within Z4, and good estimates to the receivers between the source and Z4. Estimates of the RTT to the remaining receivers are less important.

The observation that the accuracy in RTT estimates can vary according to the relative position of receivers affords several opportunities to reduce session traffic volume. First, one can aggregate estimates of the RTT for several receivers into a single estimate. For example, the receivers in zone Z4 need not know the RTT to every receiver in zone Z2. It is likely that an estimate of the RTT from node 5 to node 3 would be sufficient to perform suppression for a request/repair transaction made in the global scope zone Z0. Second, RTT measurement need not be done in a single step; instead, it can be broken up into several independent RTT measurements. For example, if one accepts that node 3 is the closest receiver in zone Z2 to the source, then the remaining receivers in zone Z4 could make an estimate of their own RTT to the source by adding the node 2's RTT to the source to their own estimate of the RTT to node 2.

SHARQFEC's session management exploits the opportunities this observation affords, by aggregating session traffic and limiting its scope according to the following set of rules:

- Each Zone determines the node closest to the data source to act as a representative for the receivers within that zone. This receiver is known as the Zone Closest Receiver (ZCR) and is determined either by design—in which case a cache is placed next to the zone's Border Gateway Router—or by a process that will be described shortly.
- A ZCR at the next-largest scope zone is known as a parent ZCR.
- Nodes other than the ZCR within a particular zone limit the scope of their RTT determination session traffic so that it is carried within the smallest-known scope zone. This enables them to determine the RTT between themselves and the other nodes within the scope zone.
- The ZCR for a particular zone participates in RTT determination for that scope zone, and also the next-largest scope zone (known as the parent zone).
- Nodes maintain state information about the RTT between themselves and other nodes within the smallest-known scope zone. In addition, they also maintain state about the RTT between the ZCR at each successively larger scope zone and the other nodes within that ZCR's parent zone.
- When a node sends nonsession traffic that requires nodes hearing it to determine the RTT between the sending node and themselves, the sending node includes estimates of the distance between itself and each of the parent ZCRs that will hear the message.
- Nodes calculate the distance between themselves and their ancestral ZCRs by adding the observed RTTs between successive generations.
- Nodes that have just received a packet from another node, and need to calculate the RTT between themselves and that node, do so as follows: First, they examine the RTT information included in a packet to determine the largest scope zone for which they can find a match between sibling ZCRs. Next, they calculate the RTT between themselves and the sending node by adding the RTT between the sibling ZCRs to the RTT between themselves and their parent ZCR, and the RTT between the sending node and the appropriate ancestral sibling as extracted from the packet.

- Nodes shall randomly stagger their session messages to reduce the likelihood of convergence. (In the simulations that follow, the delay was chosen on the uniform interval [0.9, 1.1] seconds. To speed up convergence, the delay for the first three session messages was chosen on the uniform interval [0.05, 0.25] seconds.)

Details of how ZCRs can be selected automatically are given in 4.2.6, on page 69.

4.2.5 Indirect Round-Trip Time Determination

The application of the rules above to restrict session traffic via administrative scoping has a marked effect on both the amount of traffic and state that must be handled by each receiver. For example, at the largest scope zone in the example given in Figure 4-15, Z0, the source need only participate in RTT estimation between itself, the other session members that only subscribe to Z0, and the ZCRs of the next smallest zones, Z1 and Z2 (Figure 4-16). Similarly, in the intermediate scope, Z1, only those receivers whose smallest scope zone is Z1 and the ZCRs of the next smallest zones, Z3 and Z4, participate in the RTT process for that level. Finally, in the smallest zone, Z4, only those receivers who subscribe to Z4 take part in the process.

This system of session announcements results in the receivers having a complete set of information about all the other receivers within their zone, the receivers in the regions directly between their zone, and the largest scope ZCR receiver behind each obscured region. Figure 4-17 shows explicitly how this scheme results in a dramatic reduction in the amount of state that must be maintained by each receiver. Whereas before each receiver would have had to maintain state information for 18 other receivers, in this example the worst case is now 7 receivers. The reduction in state is more pronounced as the number of levels and fanout in the hierarchy increase.

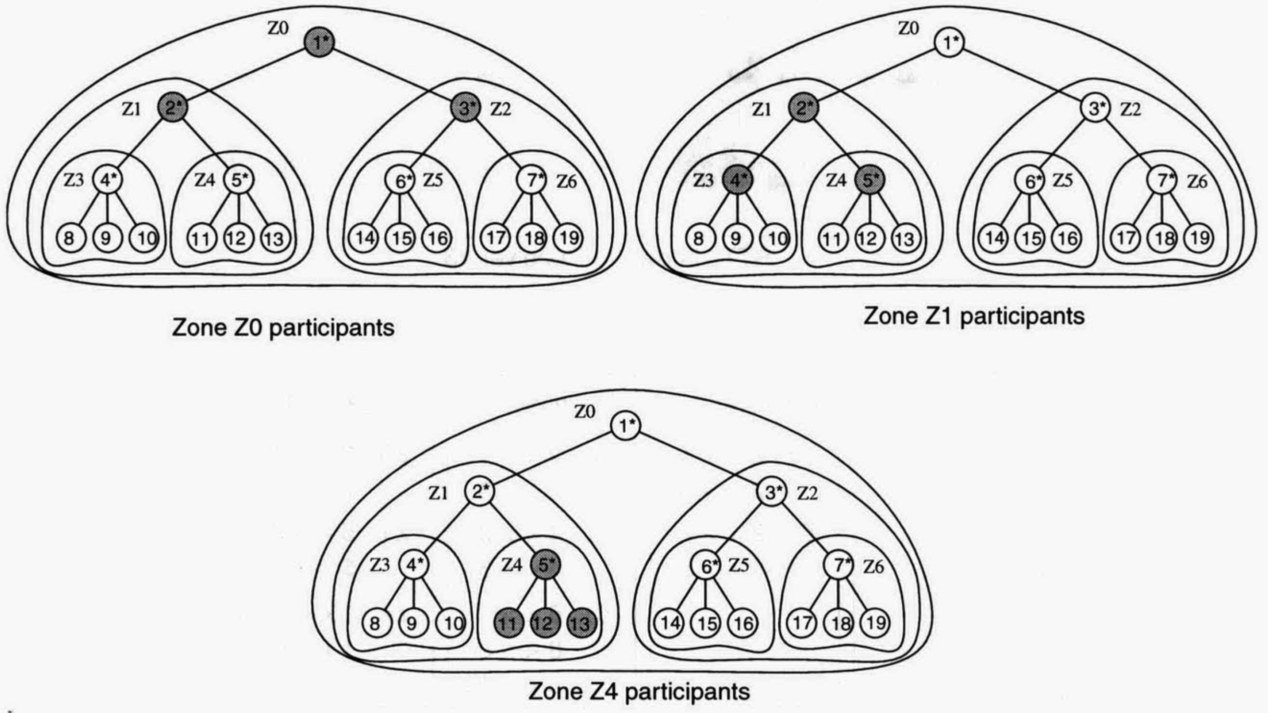


Figure 4-16 Receivers involved in RTT determination phase in Zone Z0, Z1, and Z4 respectively

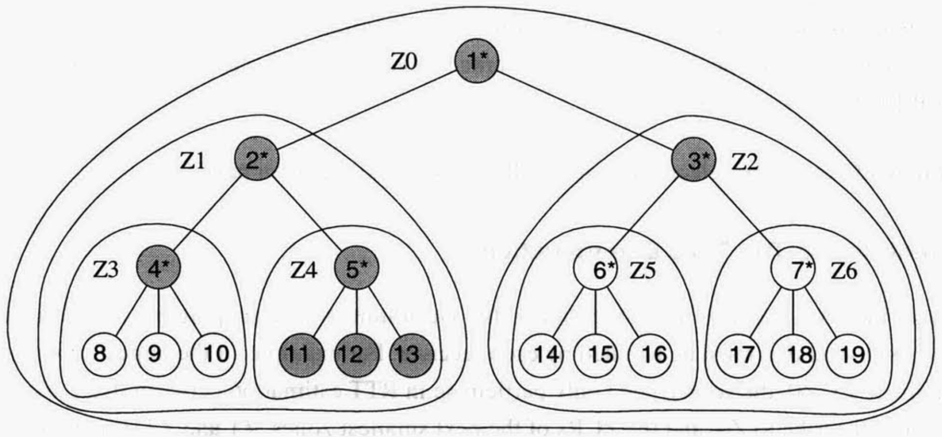
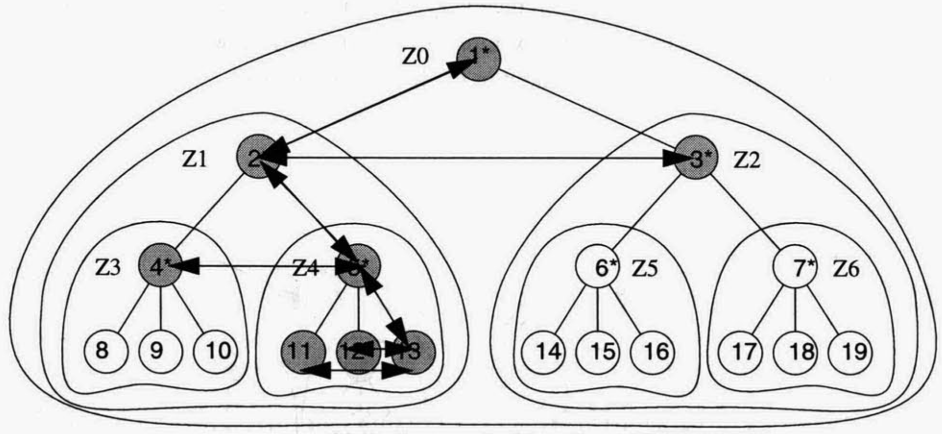


Figure 4-17 Receivers viewable by receivers in Zone 4/



RTTs known to Receiver 13

Receiver 1		Receiver 2		Receiver 8		Receiver 13	
Z0	1 <-> 2	Z0	**2** <-> 1	Z0	**2** <-> 1	Z0	**2** <-> 1
	1 <-> 3		**2** <-> 3		**2** <-> 3		**2** <-> 3
		Z1	2 <-> 4	Z1	**4** <-> **2**	Z1	**5** <-> 2
			2 <-> 5		**4** <-> 5		**5** <-> 4
				Z3	8 <-> 4	Z4	13 <-> 5
					8 <-> 9		13 <-> 11
					8 <-> 10		13 <-> 12

Figure 4-18 Session State maintained by selected receivers

To construct these reduced state tables, receivers selectively listen to and record the state traffic as follows. Starting with the smallest scope zone, each receiver constructs a state table with entries for each of the receivers that participate solely in that zone. Next, the receiver extracts the ZCR from that zone and listens to the session announcements for that receiver in the next-largest scope zone. Thus, in Figure 4-17, receiver 8—having determined that receiver 4 is the ZCR for the smallest scope—listens for session announcements originating from receiver 4 at the next-highest scope zone, and thereby determines the RTTs to receivers 2 and 5. This process is applied to all larger scope zones.

With these tables in place, receivers can estimate the RTT to any given receiver, provided that a message from that receiver contains its own estimate of the RTT to each of the ZCRs above it. Consider a packet originating from receiver 8 that is sent to the intermediate scope zone, Z1. Receivers 4, 9, and 10 will already have direct estimates of the RTT to 8, and receivers 2, 5, 11, 12, and 13 will not. However, if receiver 8 includes its estimate of the RTT between itself and receiver 4, these other receivers can compute an estimate by adding one or more RTTs together. For example, receiver 13 would add the entries for the RTTs between itself and receiver 5, receiver 5 and receiver 4, and the supplied RTT between receivers 8 and 4 to arrive at an estimate of the RTT between itself and receiver 8.

$$\underbrace{d_{8,13}}_{\text{Estimated}} = \underbrace{d_{8,4}}_{\text{From Received Packet}} + \underbrace{d_{4,5} + d_{5,13}}_{\text{From Internal Tables}}$$

Figure 4-19 Indirect RTT estimation example

It should be noted that the indirect calculation in this manner is most accurate in situations where the ZCR is immediately adjacent to the border gateway router (BGR) that enforces administrative scoping between the ZCR's zone and its parent. If the ZCR is located away from the BGR, traffic from any receivers connected directly to the BGR will not pass by the ZCR. Consequently, the RTTs estimated by these receivers will be greater than the actual RTT. Receivers that generate inaccurate RTT estimates will not generate traffic that breaks SHARQFEC's operation; in fact, the greater delays that measurements cause will result increased suppression.

Breaking up session management traffic into hierarchical overlapping regions results in a huge savings. The amount of traffic devoted to the determination of RTTs in each zone now drops from a constant $O(n^2)$ for all receivers to $O(\sum n_\alpha^2)$ where n_α is the number of session members participating at each of the α regions that are observable by a receiver. One notes two important characteristics in this new relationship. First, the farther a receiver is from the source, the more zones it will likely participate in, and therefore the more session traffic it will receive. And second, since the n^2 relationship is still present, efforts should be made to limit the number of receivers participating at each level.

The process of updating the internal tables of RTTs and electing the ZCRs at the various scope zones occurs periodically and is performed in a top-down fashion. ZCRs are elected at the largest scope zones first, with each smaller zone backing off until the parent zone has elected a ZCR. ZCR election occurs in two phases: the RTT determination phase and then an optional ZCR challenge phase. During the RTT determination phase members of a particular zone, exchange session messages that contain:

- the time at which the message was sent,
- the identifier of the ZCR for that zone,
- the recorded distance between the ZCR of that zone and the ZCR of the next-largest zone,

and a list with the following information about each receiver heard from that zone.

- receiver's identity,
- Time elapsed since the last session message was received from the receiver,
- The sender's own estimate of the RTT between itself and the receiver.

Receivers that are the ZCR for a zone participate not only in that zone but that zone's parent as well, and as a result send out two session messages when their session timers expire. The first session message lists entries for the child zone's receivers and is sent to the child zone, while the second message is sent to the parent zone and lists the parent zone's receivers.

When a receiver receives a session message from a peer in its smallest-known scope zone, it records the time, the sender's timestamp, and the time the message was received for inclusion in its next session message. The receiver also scans the list of entries and if it finds an entry corresponding to its own identifier, it extracts the information and updates its estimate of the RTT to the sender. When a receiver receives a session message from its local ZCR, or one of its parents, it records that node's identifier and the RTT listed with each entry, thereby obtaining a record of how far the ZCR is from each of its peers. It should be noted that, thanks to the scoping mechanisms imposed, a receiver will only receive session messages from peers and the ZCRs of the largest obscured zones.

To see how SHARQFEC's scoped RTT estimation mechanism translates into savings in session state in the real world, consider the construction of a national distribution network to deliver a sporting event to 10,000,000 receivers using a 4 level hierarchy consisting of 10 regions, with each region encompassing 20 cities with 100 suburbs and 500 subscribers in each suburb. To assist with intermediate caching, dedicated caching receivers have been distributed at each of the bifurcation points to act as ZCRs, except at the suburb level, where one of the 500 subscribers will be elected to perform this task.

In this example, there is one sender and 10,000,210 receivers. When direct nonscoped RTT estimation is used, the sender and receivers are required to transmit and maintain state for each of the 10,000,210 other session members. Clearly, direct nonscoped RTT estimation will not work. However, the table shows that the state (and hence traffic) that each receiver must keep track of is reduced to a much more manageable level.

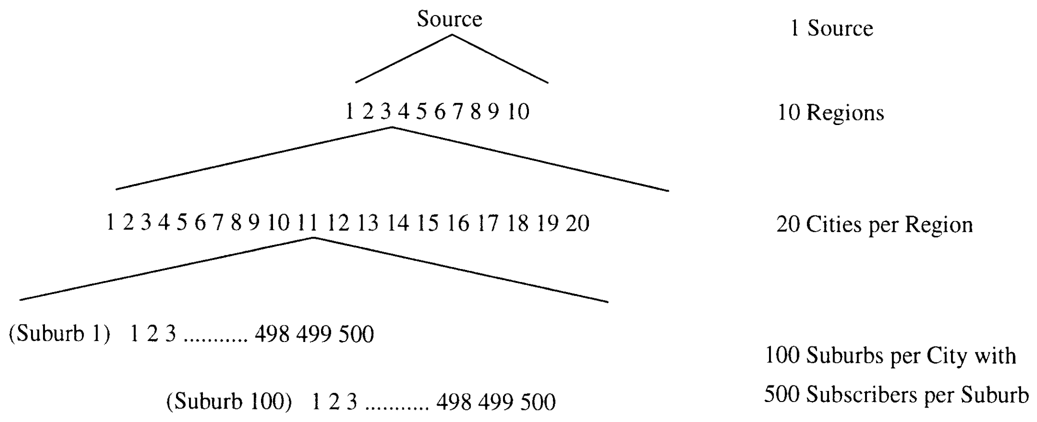

Source
1 Source
1 2 3 4 5 6 7 8 9 10
10 Regions
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
20 Cities per Region
(Suburb 1) 1 2 3 .... 498 499 500
100 Suburbs per City with
500 Subscribers per Suburb
(Suburb 100) 1 2 3 .... 498 499 500

Figure 4-20 Receiver distribution in a hypothetical National Distribution Hierarchy

Zone Level	Number of Receivers per Zone	Number of Zones	Number of Receivers	Number of RTTs to maintain per receiver	Ratio of Scoped to Non-Scoped Traffic	Ratio of Scoped to Non-Scoped State
0 National	0	1	0	10	$100/10{,}000{,}210^2$	1/1,000,021
1 Region	1	10	10	30	$500/10{,}000{,}210^2$	3/1,000,021
2 City	1	20	200	130	$10{,}500/ 10{,}000{,}210^2$	13/1,000,021
3 Suburb	500	100	10,000,000	630	$35{,}500/ 10{,}000{,}210^2$	63/1,000,021

Table 4-1 Receiver state reduction through the Use of Scoping for RTT estimation

4.2.6 Adaptive Selection of Zone Closest Receivers

As receivers join and leave sessions, it is almost certain that the multicast distribution tree between the source and receivers will change. The ZCR challenge phase is designed to accommodate this change, and provides a means for the receivers within a zone to elect a new ZCR, should the old ZCR leave the session or should the distribution tree change so that it is no longer the closest receiver to the source for that zone. Service providers may configure their networks so that there is a static ZCR adjacent to the router that enforces the boundary between the local-scope zone and the parent-scope zone. In these cases, the ZCR challenge phase will only be necessary should one wish to provide robustness in the event that the dedicated receiver ceases to function.

Like the RTT determination phase, the ZCR challenge phase makes the assumption that the distribution trees created by the routers do not differ significantly near the boundaries between regions of different scopes, or as a function of the source of the packet. For the most, part this assumption holds, since the majority of source-based optimizations made by multicast routing protocols [89, 97, 98, 100, 101, 107] tend to occur at the leaves of the tree and not within the backbone of the network. To assist in understanding of the ZCR challenge phase Figure 4-21 shows the four possible challenge cases. Node 1 in both cases is the current ZCR, with nodes 2 and 4 closer to the parent ZCR, and nodes 3 and 5 farther away. For the purposes of the following discussion, it is assumed that the transit times for messages between two receivers is the same as the distance shown.

Consider first the chain example shown on the left. Here, node 1 is currently the ZCR for the zone containing nodes 1, 2, and 3. Now, let node 1 transmit a ZCR challenge to the parent ZCR, node 0. This challenge will be received by nodes 0, 2, and 3 at d_{01}, d_{12}, and d_{13} respectively. After a processing delay, node 0 transmits a ZCR response containing the delay between when the ZCR challenge was received and the ZCR response was sent. Assuming this delay to be negligible, the ZCR response will be received by nodes 2,1, and 0 at $(d_{01} + d_{02})$, $2d_{01}$, and $(2d_{01} + d_{03})$ respectively. Upon receiving the ZCR, response each node then estimates its distance from the parent ZCR using the following formula.

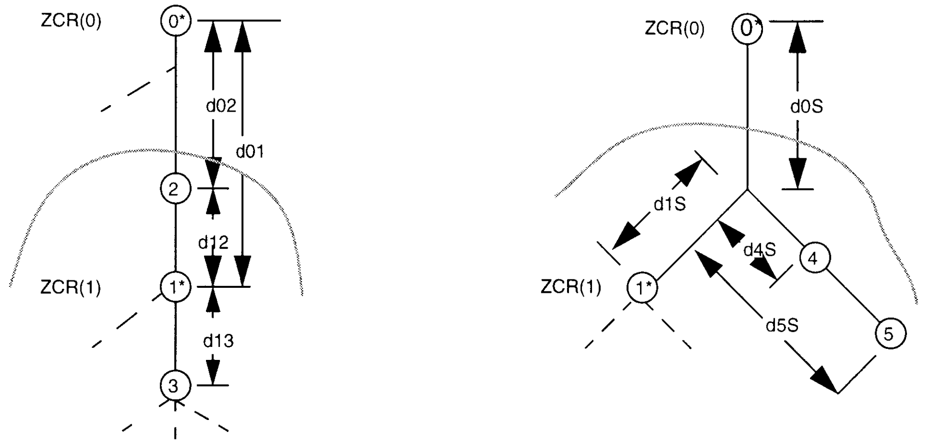


Figure 4-21 The Four Possible SHARQFEC ZCR Challenge cases

$$dist_{toparentZCR} = dist_{tolocalZCR} + (t_{ZCRreplyrecv} - t_{ZCRchallengerecv}) - dist_{fromlocalZCRtoparentZCR} \qquad \text{Eqn 4-10}$$

The application of this formula yields d_{02}, d_{01}, and d_{03} for nodes 2, 1, and 3, respectively. At this stage, node 2 will note that it is closer to the parent ZCR than node 1, the current ZCR, and immediately transmits two ZCR takeover packets containing its calculated distance to the parent ZCR. The first packet sent to the child zone informs the other nodes that it is closer to the parent ZCR than node 1, while the second sent to the parent zone informs the parent ZCR that a new representative has been elected for the child zone. Node 3, on the other hand, determines that it is farther away and remains silent. When nodes receive a takeover message from a new ZCR, they replace the identity of any entries that match the old ZCR with the new one. Receivers in the parent zone will not know the distance between themselves and the new ZCR for the child zone. However, as the RTT between the old child ZCR and the parent ZCR is greater than for the new child ZCR and the parent ZCR, using the old RTT value until it is updated does not adversely affect the back-off timers or cause extra retransmissions. One further notes that the application of this formula to receivers 1, 4, and 5 in the fork example shown in the right of Figure 4-21 also yields the true RTTs, d_{01}, d_{04}, and d_{05} respectively, and therefore the logic for the star case is the same as for the chain case.

This process of transmitting ZCR challenges is performed periodically by each ZCR, with the time between successive challenges being randomized to prevent synchronization between ZCRs in sibling zones. Since a ZCR might expire, nodes within a child zone maintain their own ZCR timers, but set them so that their firing window is always slightly larger than that of their ZCR. Thus, a non-ZCR will only issue a challenge to the parent in the event that it fails to hear from the local ZCR. The issuance of a ZCR challenge by a non-ZCR node does not automatically result in that node becoming the ZCR, since the old ZCR will still determine that it is closer than the usurper and reassert its superiority as soon as the usurper attempts to issue a takeover message. Furthermore, since ZCR takeover messages contain the new ZCR's estimate of its distance to the parent ZCR, other potential ZCRs should perform suppression as appropriate. The challenge process always results in the closest receiver in the zone being elected as the ZCR.

4.2.7 Session Maintenance Simulations

To prove that SHARQFEC's hierarchical session management scheme allows RTT estimation to be performed in a piecemeal fashion, it was simulated using the UCB/LBNL/VINT network simulator, *ns* [81], and network animator, *nam* [82], packages. Restricted by local memory requirements to simulating networks of only a few hundred receivers, simulations were run using the hybrid mesh tree topology shown in Figure 4-22. The nodes within this network were arranged such that the sender or top ZCR, node 0, fed data to a 3 level hierarchy of 112 receivers arranged as a mesh of 7 receivers that each fed balanced trees. The links connecting the source to the top 7 receivers in each tree were initialized to 45Mbit/sec with all other remaining links set to a rate of 10Mbit/sec. Latencies between the receivers located within each tree were set to 20ms for each link while the latencies used for the backbone links are shown in Figure 4-22. (The link loss rates shown do not apply for session traffic).

Other networks that were purely chain- or tree-based were also simulated, and, as expected, the appropriate receivers were elected as the ZCR for each zone with each election at each zone taking either one or two challenges. The application of the dynamic ZCR election process in the network shown in Figure 4-22 also resulted in the appropriate receiver being elected as the ZCR for each zone. At this stage, tests were run to prove that it was possible for receivers to accurately determine the RTT between themselves and the sender of a packet carrying the necessary partial RTTs between ZCRs at different scopes.

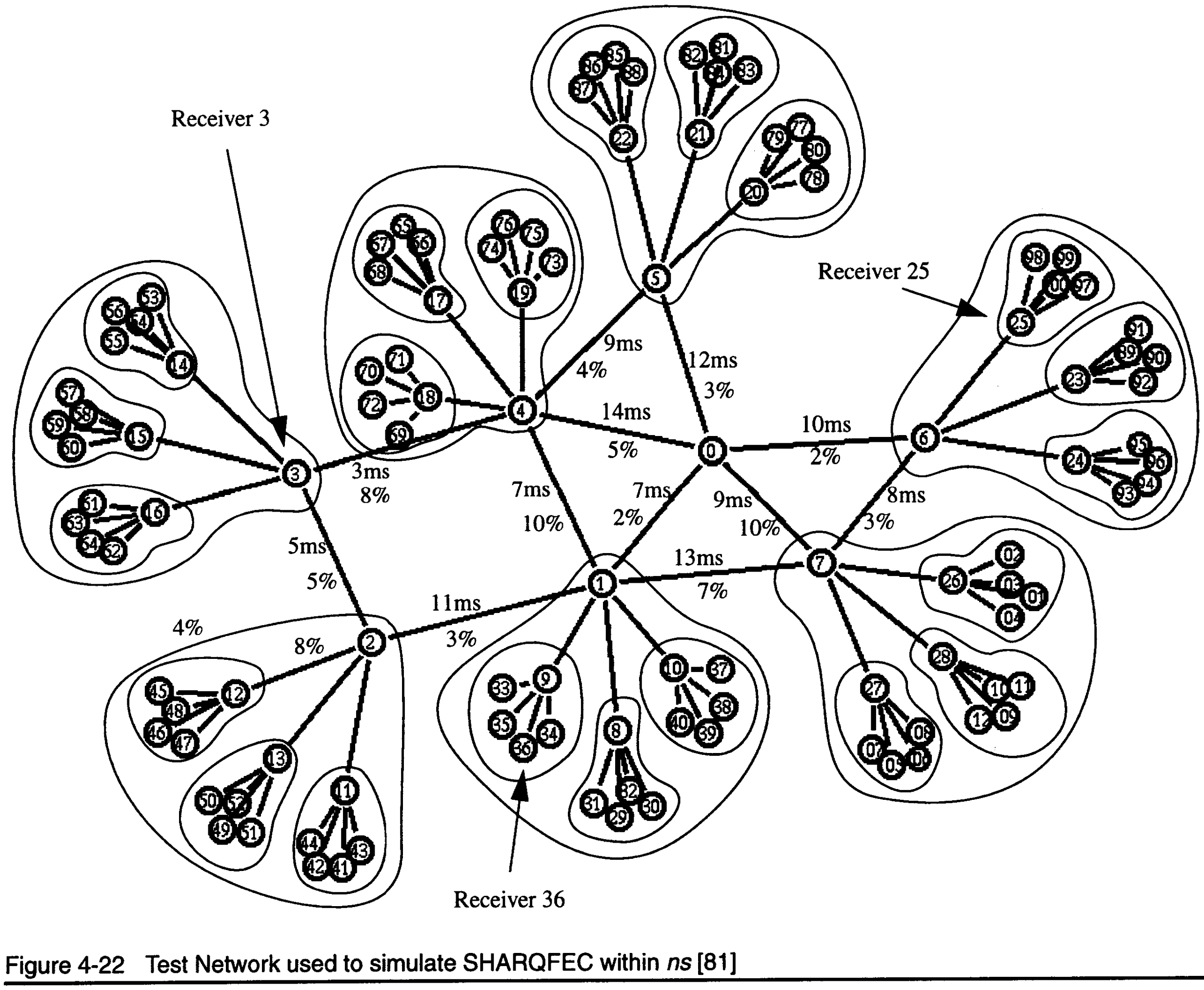


Figure 4-22 Test Network used to simulate SHARQFEC within *ns* [81]

The first part of the test involved choosing a receiver at random from each level in the hierarchy and having it send a fake NACK to the largest scope so that it would be heard by every other receiver. If the receiver could not determine an estimate to the sender, the scheme was flawed. The second part involved the selected receivers sending NACKs at regular predetermined times. Other receivers that heard these NACKs noted the time of reception, thereby enabling them to determine the actual RTT to the receiver that sent the NACK. The reason for sending multiple NACKs was twofold: first, to prove that estimates were stable; and second, to show that any inaccuracies introduced by a suboptimal ZCR being elected during initialization diminished over time as successive local RTT measurements were made. The results of this test for messages originating from receivers 3, 25, and 36, are shown in Figures 4-23 and 4-24.

These three figures show that more than 50% of receivers were able to estimate the RTT to a NACK's sender to within a few percent. These figures also show that even when a receiver other than closest one within a zone is initially chosen, the estimates improve over time with each successive measurement. This improvement occurs asymptotically since new measurements are merged with the old using an exponential weighted moving average filter.

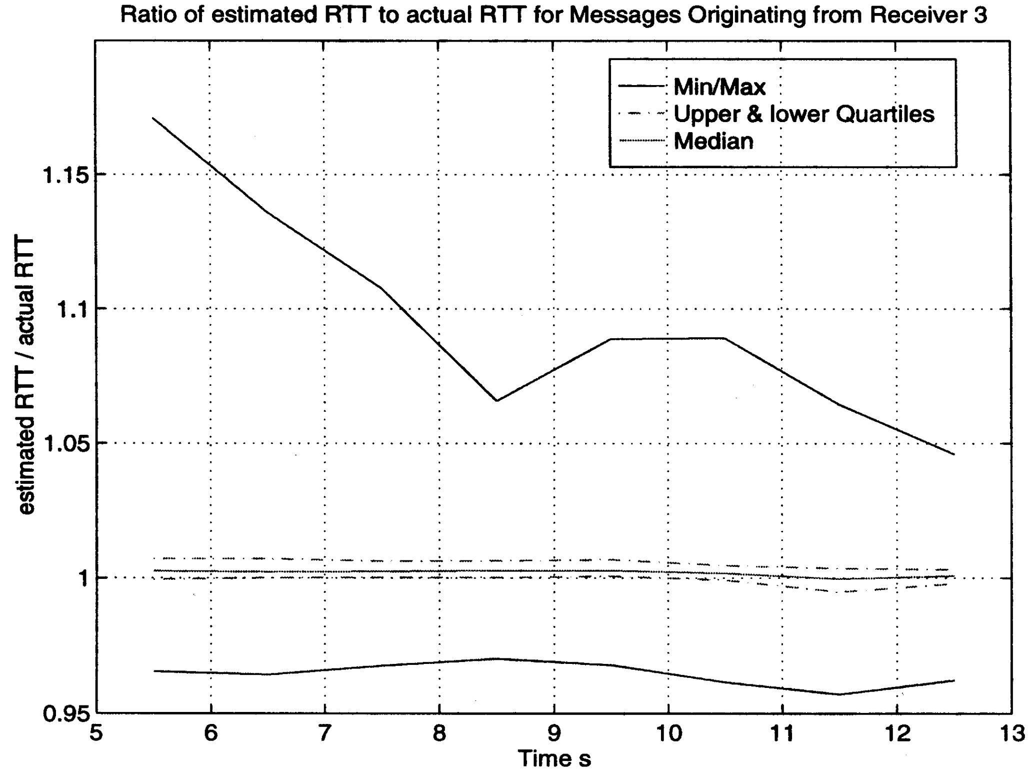
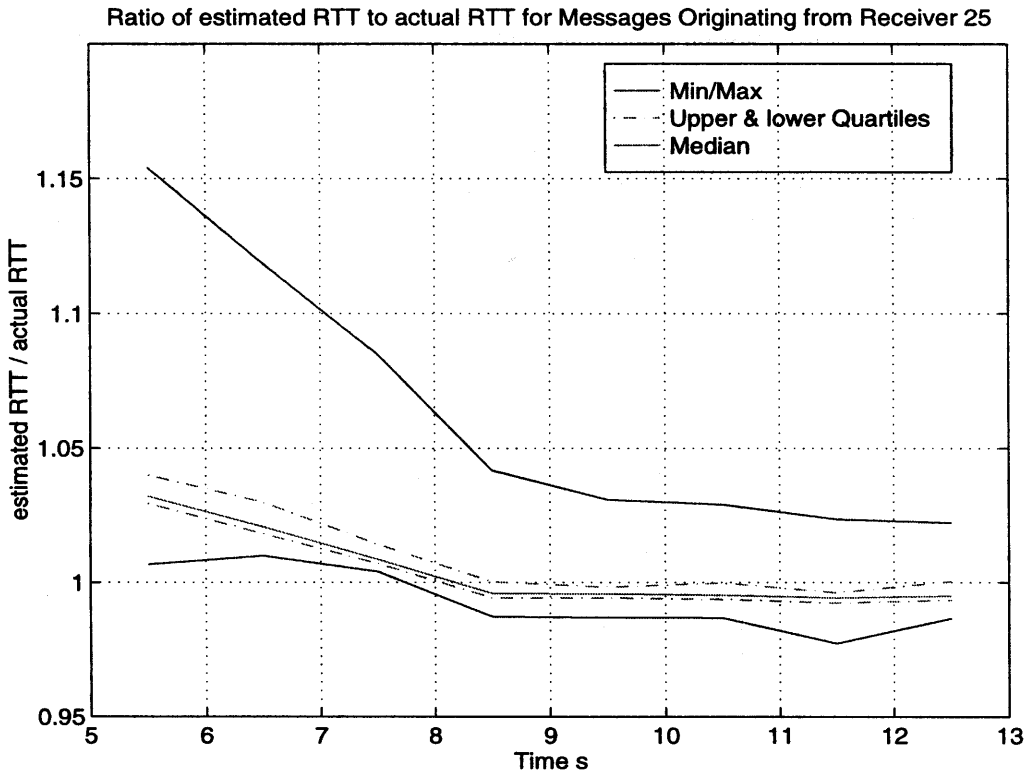


Figure 4-23 Ratio of estimated RTTs to actual RTTs for Messages Originating from Receiver 3



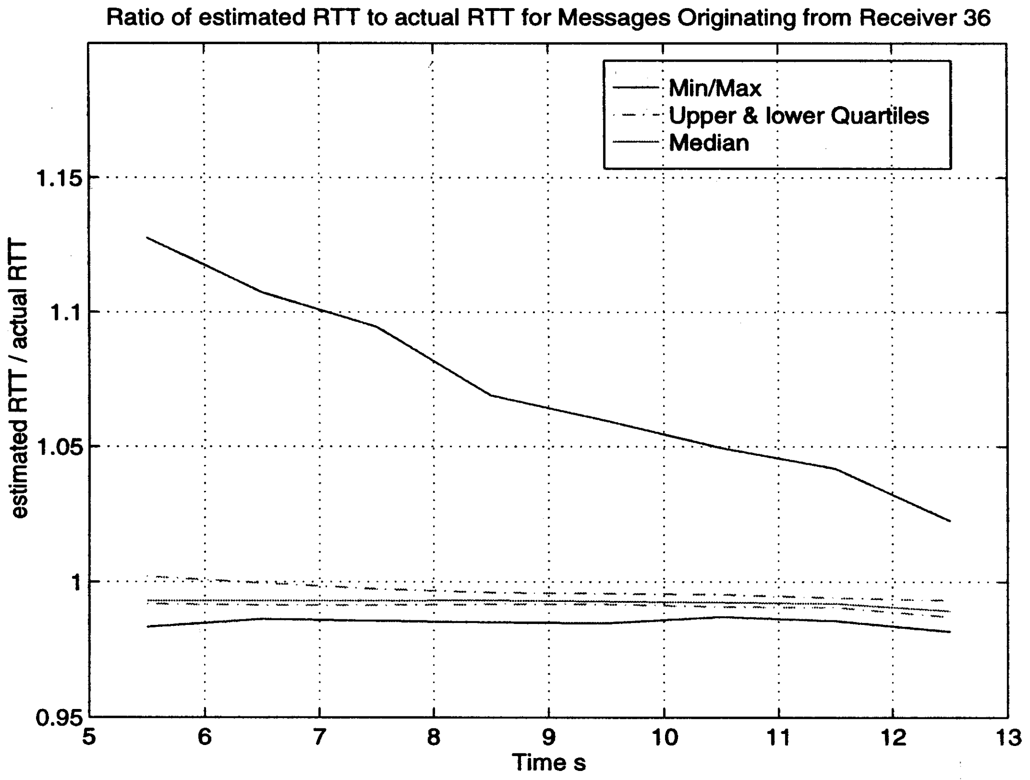


Figure 4-24 Ratio of estimated RTTs to actual RTTs for Messages Originating from Receivers 25 and 36

4.2.8 Data/Repair Traffic Simulations

Having verified that SHARQFEC's session-maintenance strategy worked, simulations were run to qualify SHARQ-FEC's performance in delivering data against other delivery schemes. SHARQFEC was compared with an ARQ protocol, and several hybrid ARQ/FEC protocols. SRM was chosen as the ARQ protocol, and its simulation was performed with adaptive timers turned on for best possible performance. The various hybrid ARQ/FEC protocols were simulated by turning off various features within SHARQFEC. The six protocols simulated are described in table 4-2. It should be noted that the SHARQFEC(ns,ni,so) protocol is nearly identical to ECSRM [29] except that ECSRM uses fixed timer windows for its suppression timers, while SHARQFEC(ns,ni,so)'s timer windows are based on the RTTs measured between receivers.

Name	Type	Administratively Scoped Localization?	Preemptive Injection of Repairs?	Repairs can be sent by
SRM	ARQ	no	no	All nodes
SHARFEC(ns,ni,so) / ECSRM	Hybrid ARQ/FEC	no	no	Sender-only
SHARFEC(ns,ni)	Hybrid ARQ/FEC	no	no	All nodes
SHARQFEC(ns)	Hybrid ARQ/FEC	no	yes	All nodes
SHARQFEC(ni)	Hybrid ARQ/FEC	yes	no	All nodes
SHARQFEC	Hybrid ARQ/FEC	yes	yes	All nodes

Table 4-2 Attributes of Simulated Protocols

Simulations were performed on the same topology used for simulating the viability of session maintenance strategy. To stress the protocols as realistically as possible, the loss rates for the links were set to ensure that every link suffered some loss, and that some parts of the network suffered greater losses than others. The losses for the mesh part of the topology are included in Figure 4-22. The loss rate between each of the seven mesh nodes and their three children was set to 8%, while the loss rate between the three children and their children was set to 4%. Thus, when the resulting routing trees were taken into account to the outermost receivers, receivers 53 through 62 experienced the worst loss (on the order of 28.3%) while receivers 89 through 100 experienced the least loss (on the order of 13.4%). Realism was further enhanced by subjecting repair packets to the same loss patterns. Session traffic and NACKs were not subject to losses.

Simulations for each of the protocols were then performed. Each simulation consisted of the nodes joining the session at t = 1 sec, at which time they began sending session messages to each other. After allowing five seconds for the session state to be established and stabilized, a constant bit rate source at node 0 was turned on at t = 6 sec. The source then emitted 1024 thousand-byte data packets at a rate of 800Kbit/sec, turning itself off at t = 16.24 sec. The length of the run was chosen to be long enough so that any dependency upon *ns*'s internal random number generator would be minimized when the results were viewed as a whole. Packets were sent in groups of 16 for each of the SHARQFEC runs.

The recovery mechanisms of the SHARQFEC protocol work on groups of packets, so performance for both SRM and SHARQFEC simulations was measured by comparing the sum of data and repair traffic visible at each session over 0.1 second intervals. Thus, if transmission were lossless, one would expect to observe the arrival of ten data packets per measurement interval.

Figures 4-25 and 4-26 show the minimum, maximum, and average number of data plus repair packets, and NACK packets heard by each receiver for the SRM and SHARQFEC(ns,ni,so)/ECSRM protocols respectively. Even with adaptive timers present, SRM required roughly three to four times the bandwidth of the original data stream. This excessive bandwidth is attributable to the fact that lost repairs caused the repair-suppression mechanism to fail in other receivers. In addition, subjecting repairs to loss also caused a sizeable tail of repair traffic that persisted long after the source had finished transmitting the original data. The SHARQFEC(ns,ni,so)/ECSRM protocol fared considerably better and clearly shows the benefits afforded by using hybrid ARQ/FEC; both the number of NACKs and repairs sent is drastically reduced.

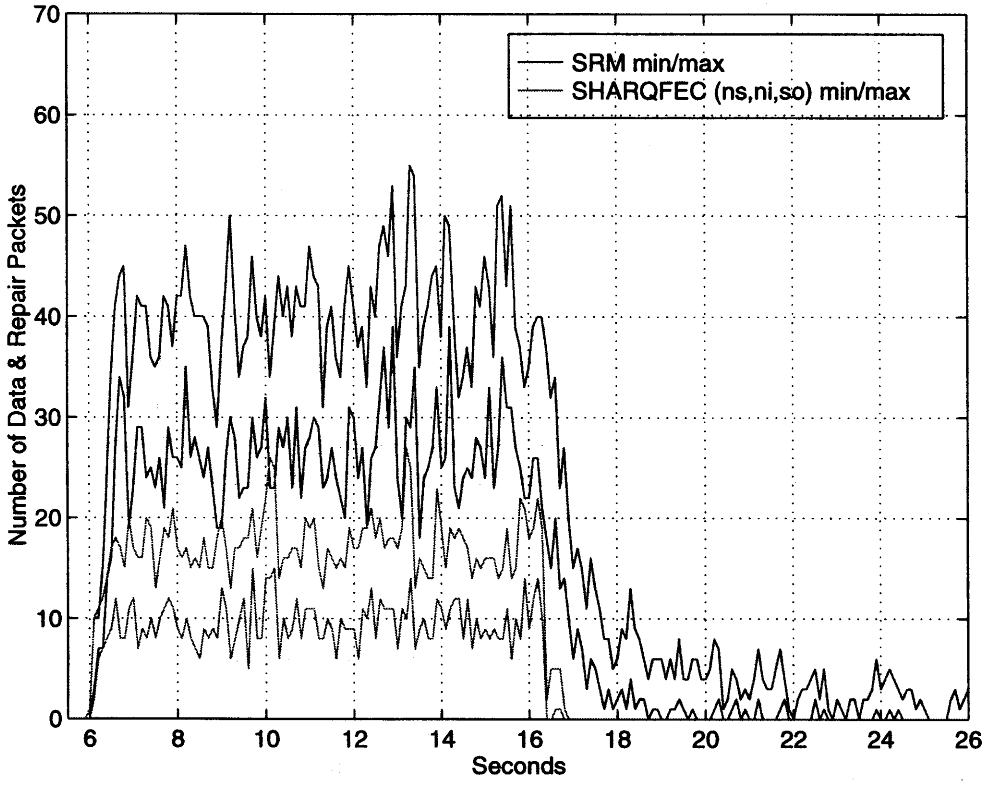
The top graph of Figure 4-27 shows—somewhat counter-intuitively—that relaxing the constraint of only allowing the sender to send repairs results in markedly worse performance. In this case, the bandwidth peaks between six and seven times, and sometimes as high as nine times, of the original stream. This marked reduction in performance is caused by repairs being lost in transit to potential repairers, which then needlessly transmit further repairs. The addition of preemptively injected repairs at the source improved performance; however, the bandwidth was still greater than that of the original sender-only scheme.

The bottom graph of Figure 4-26, however, shows that the addition of scoping in the SHARQFEC scheme resulted in better performance than the sender-only protocol. This graphs shows that the SHARQFEC scheme is better behaved than the SHARQFEC(ns,ni,so)/ECSRM scheme, and results in fewer bandwidth peaks and a more constant use of bandwidth in general. When one factors out the ten packets that should be received during each interval when perfect recovery occurs one finds that the SHARQFEC scheme reduces the excess traffic in these bandwidth peaks by a factor of two to four. The top graph in Figure 4-28 shows that the introduction of scoping considerably reduces NACK traffic volume.

The bottom graph in Figure 4-28 shows that the SHARQFEC and SHARQFEC (ni) schemes perform very similarly. Turning preemptive repair injection on within SHARQFEC does not result in a measurable increase in bandwidth. Given that this is the case, preemptive repair injection should be left on, since it results in repairs being received sooner and hence reduces latency.

The argument for the use of SHARQFEC supported by Figures 4-25 through 4-28 is further strengthened when one examines the traffic at the various levels in the hierarchy. Figure 4-29 shows the traffic observed at the sender, node 0. When the ten data packets the sender transmits during each interval are factored out, one finds that the SHARQFEC scheme reduces the repair traffic volume at the topmost level in the hierarchy to a level comparable with the greatest loss experienced by the ZCRs at the first level (receivers 1 through 7). The spikes observed in 4-29 occur as a result of the repairs themselves being lost.

Figures 4-30 and 4-32. show that the traffic volumes at each level in the hierarchy remained roughly constant. Thus, the additional redundancy injected at each level was absorbed through losses in transit to the next. Examinations of the plots at each of the four levels for time t = 14 sec shows that even when excessive repair traffic was generated due to pathological loss, the excess traffic was eventually absorbed by the time the lowest level in the hierarchy was reached. Finally, Figure 4-26 shows that the repair tail has shrunk even further. This shrinkage is due to the fact the receivers higher up in the hierarchy are repaired first and then only take part in further repairs if so requested by receivers lower in the hierarchy.



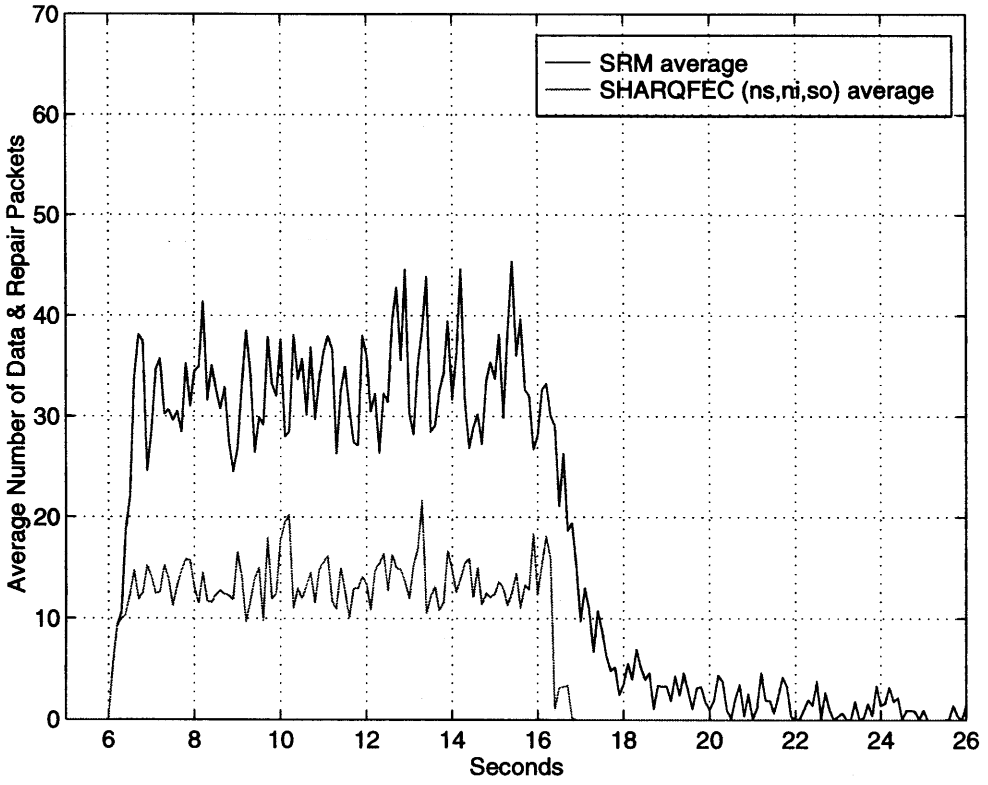
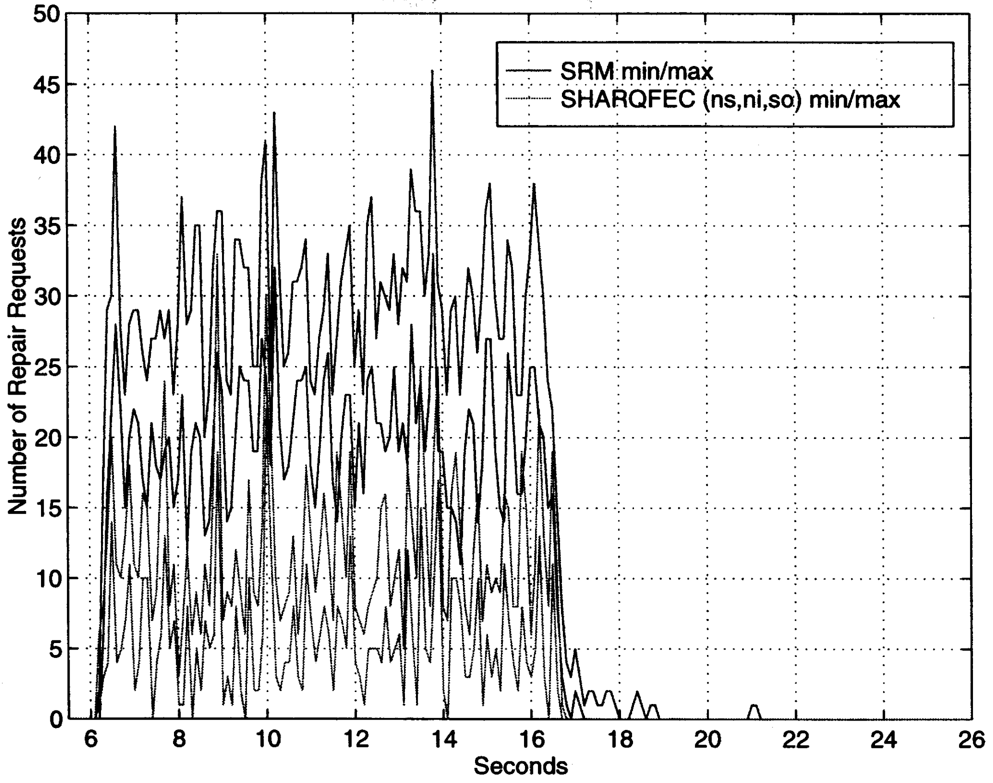


Figure 4-25 Data and Repair Traffic - SRM and SHARQFEC(ns,ni,so) / ECSRM[29]



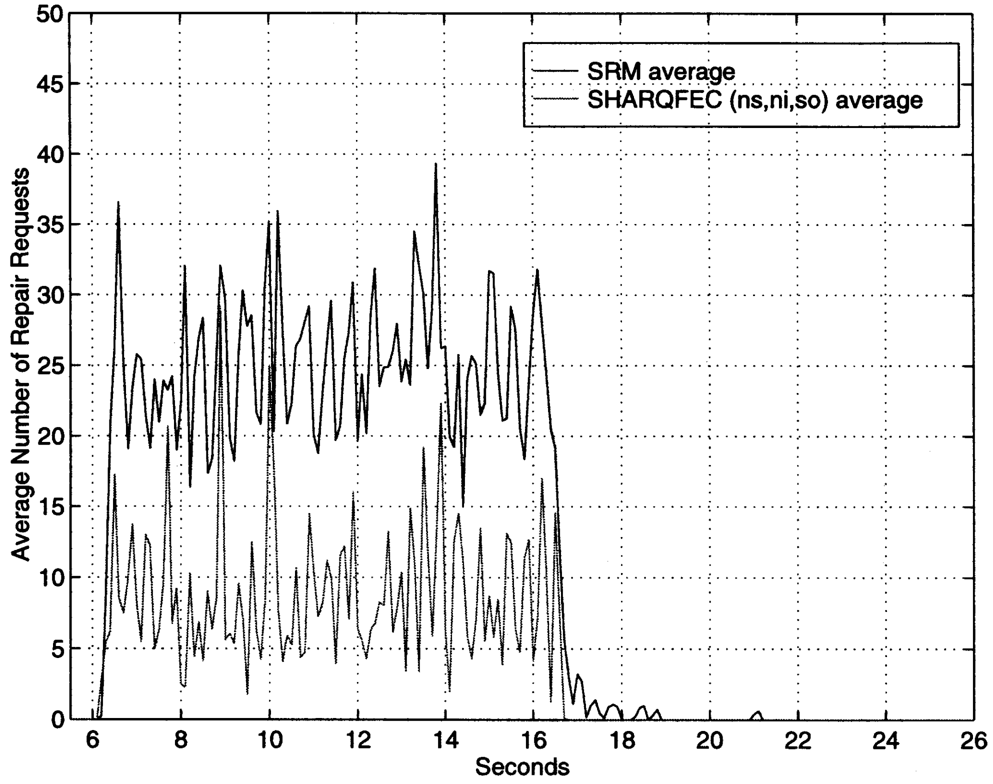
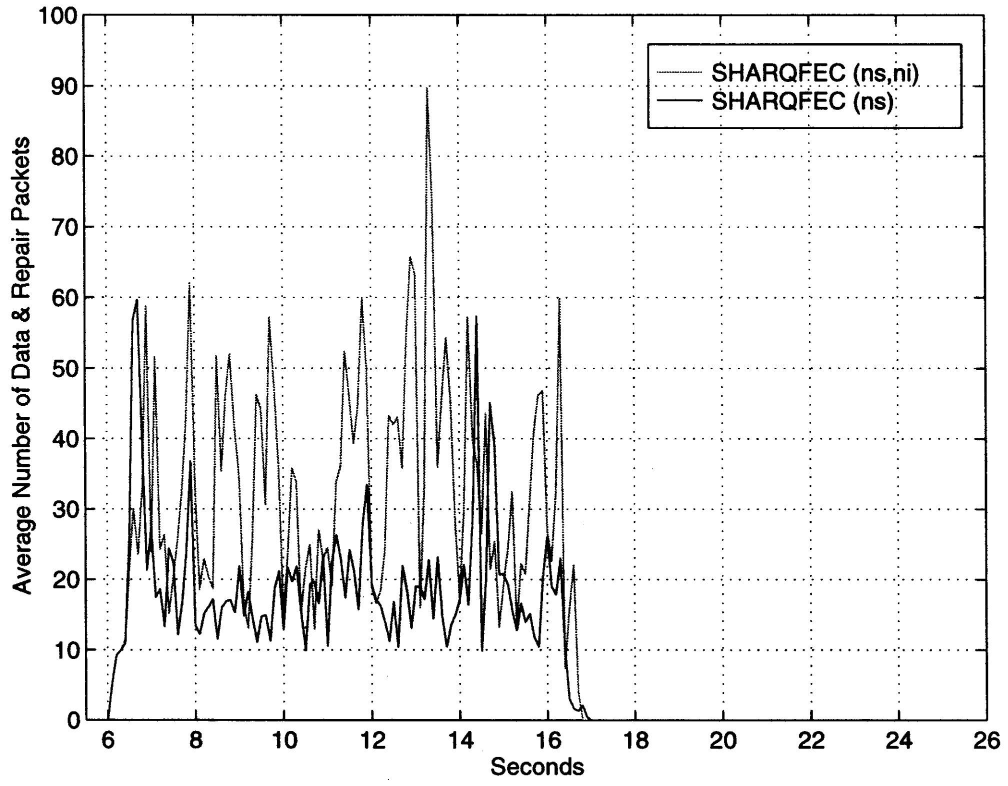


Figure 4-26 NACK Traffic - SRM and. SHARQFEC(ns,ni,so) / ECSRM[29]



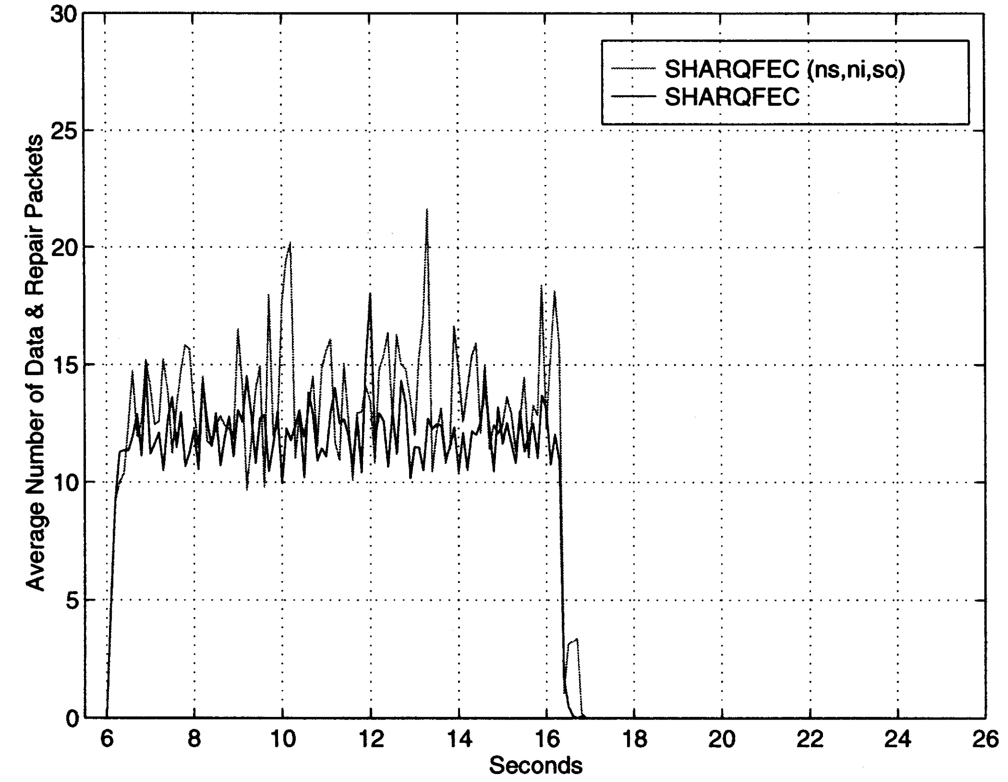
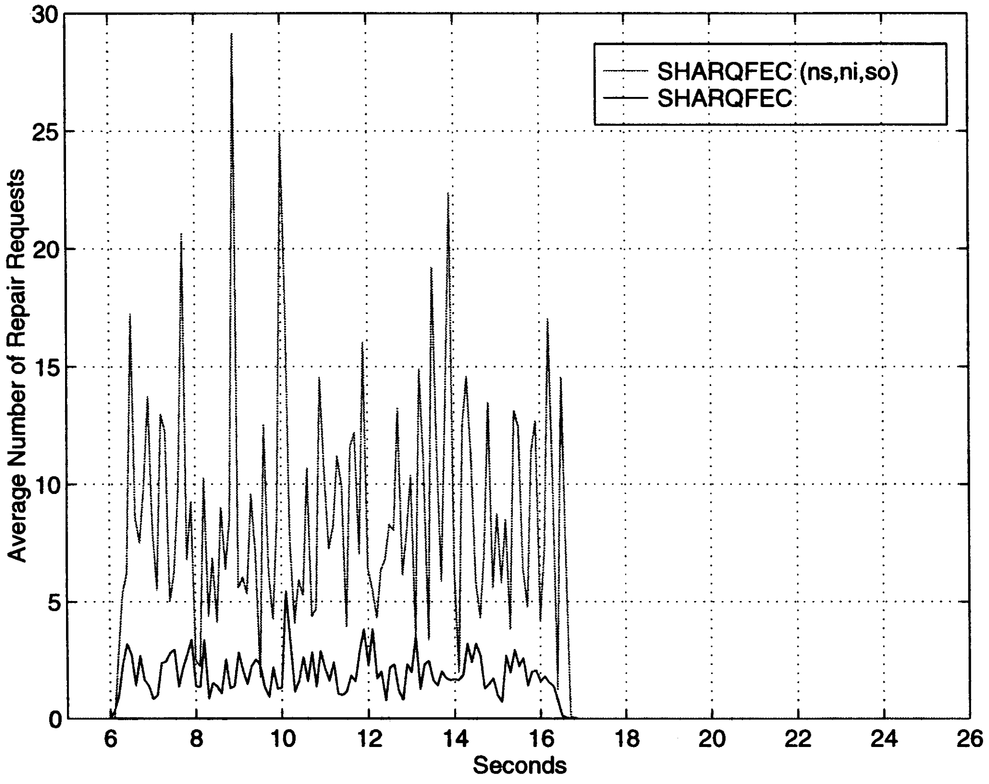


Figure 4-27 Data and Repair Traffic - SHARQFEC(ns,ni), SHARQFEC(ns), SHARQFEC(ns,ni,so) & SHARQFEC



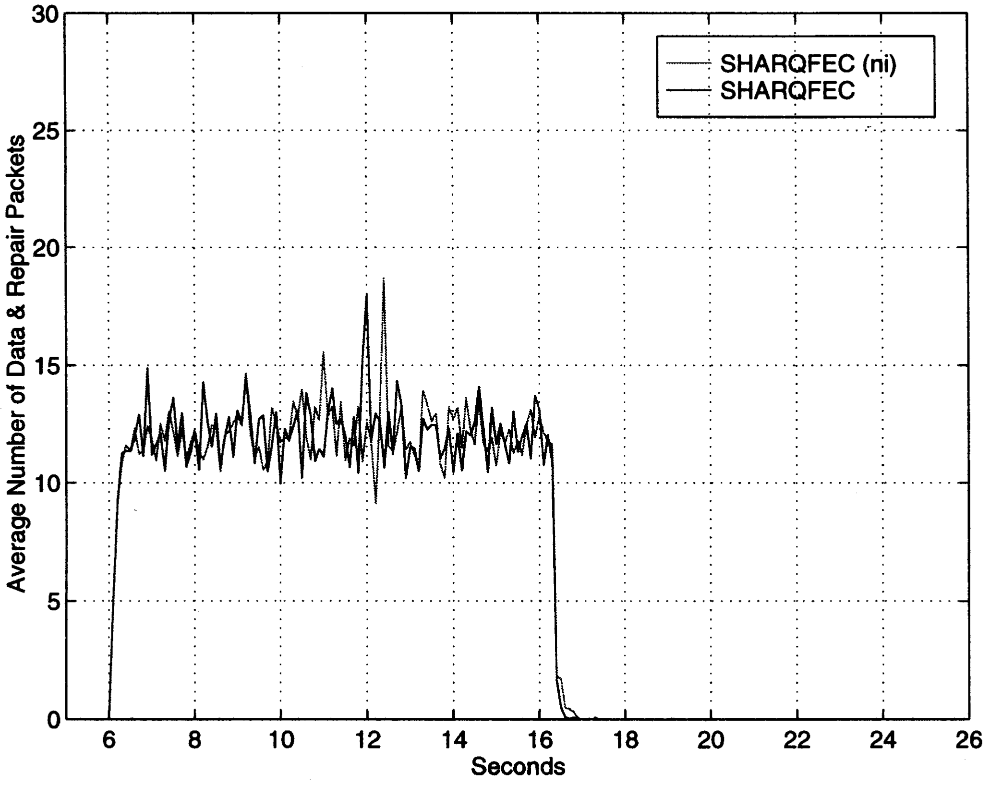
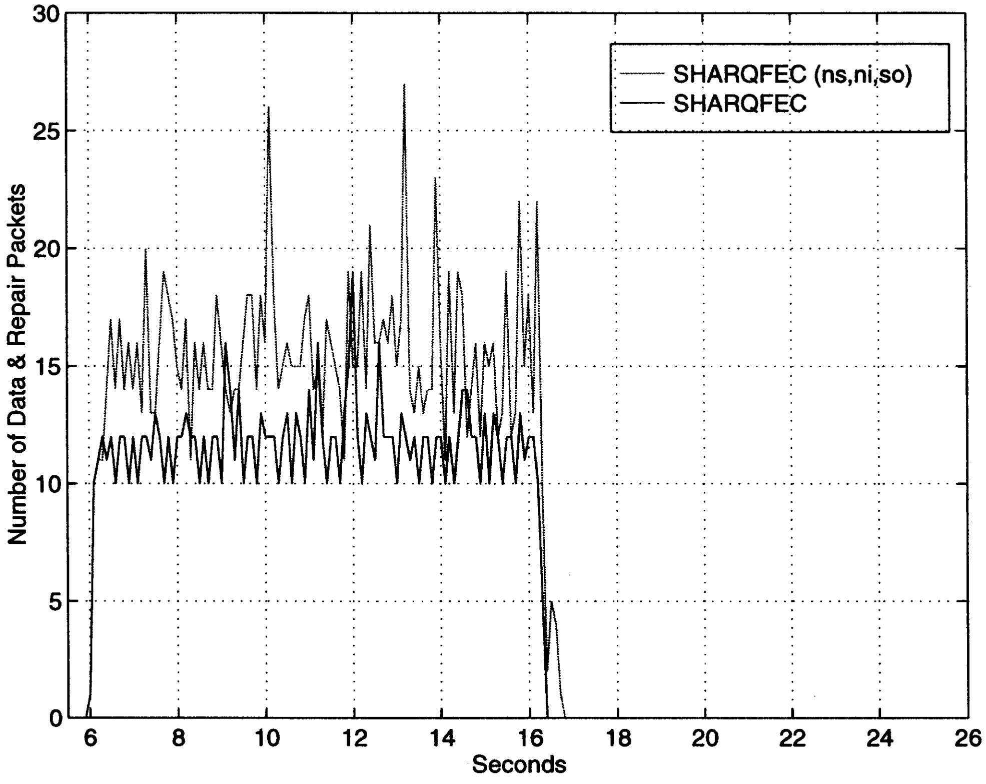


Figure 4-28 NACK traffic-SHARQFEC(ns,ni,so)&SHARQFEC / Data & Repair Traffic-SHARQFEC(ni)&SHARQFEC



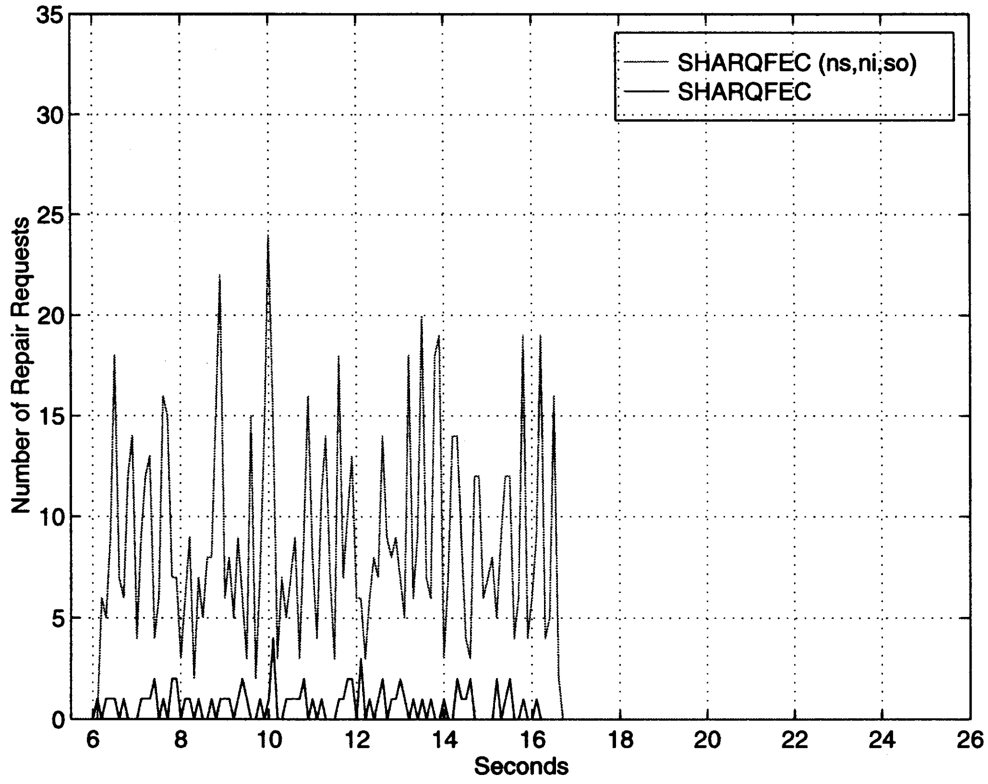
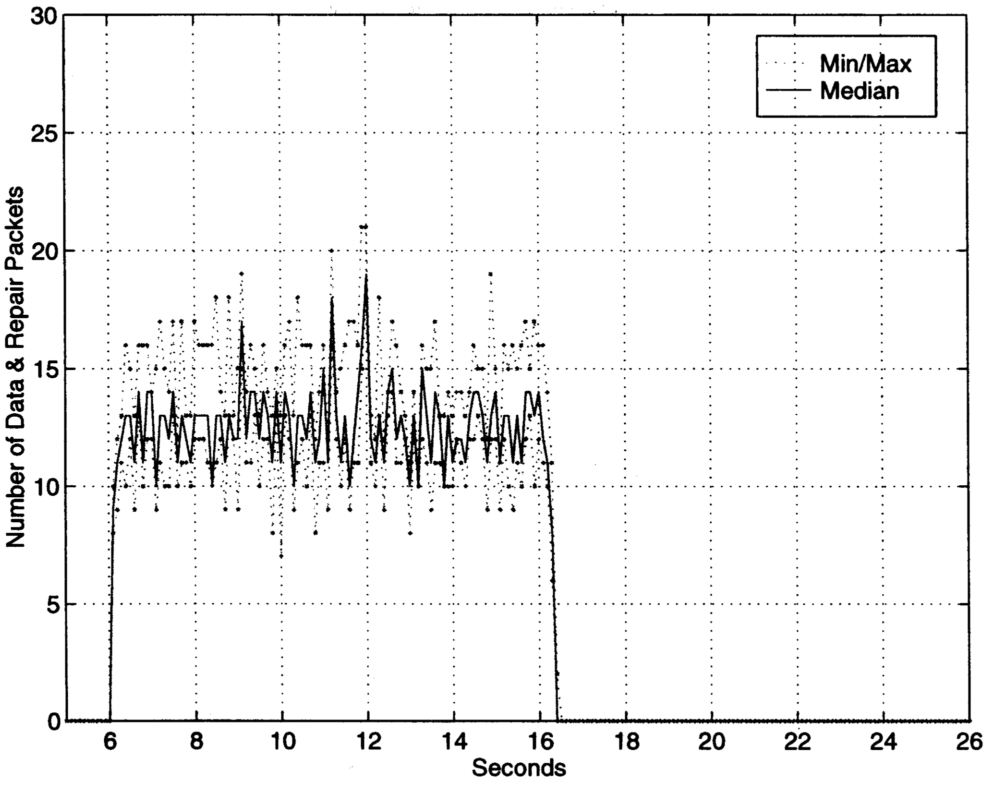


Figure 4-29 Data plus Repair, and NACK Traffic seen by the Source for SHARQFEC(ns,ni,so) and SHARQFEC



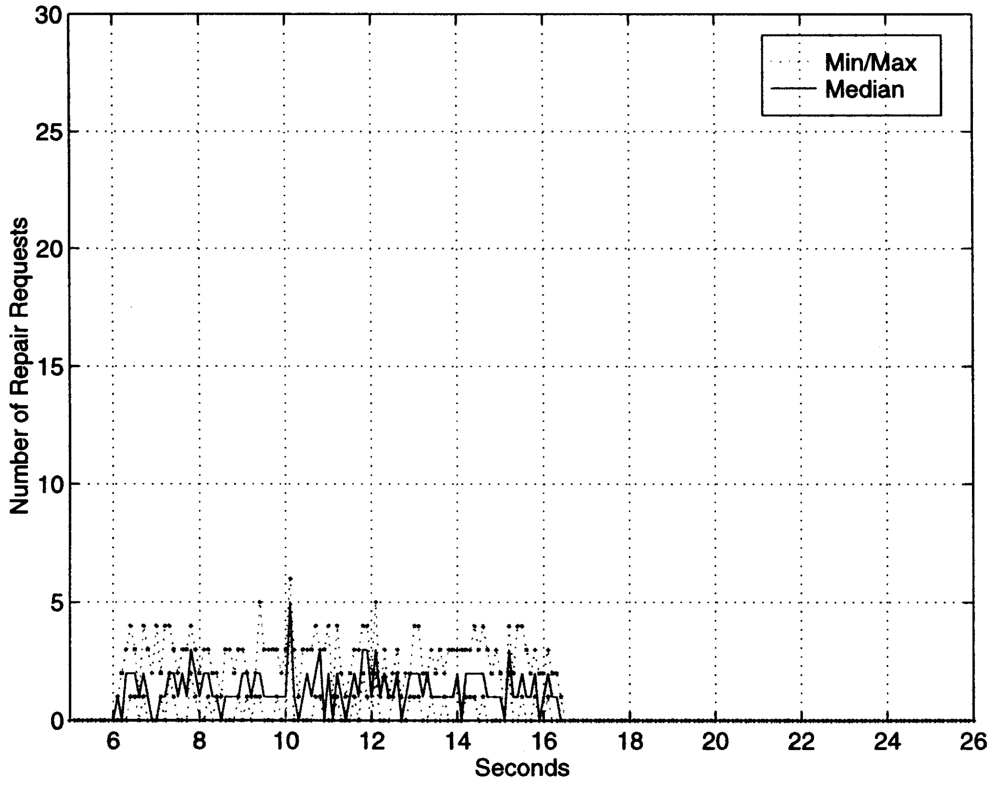
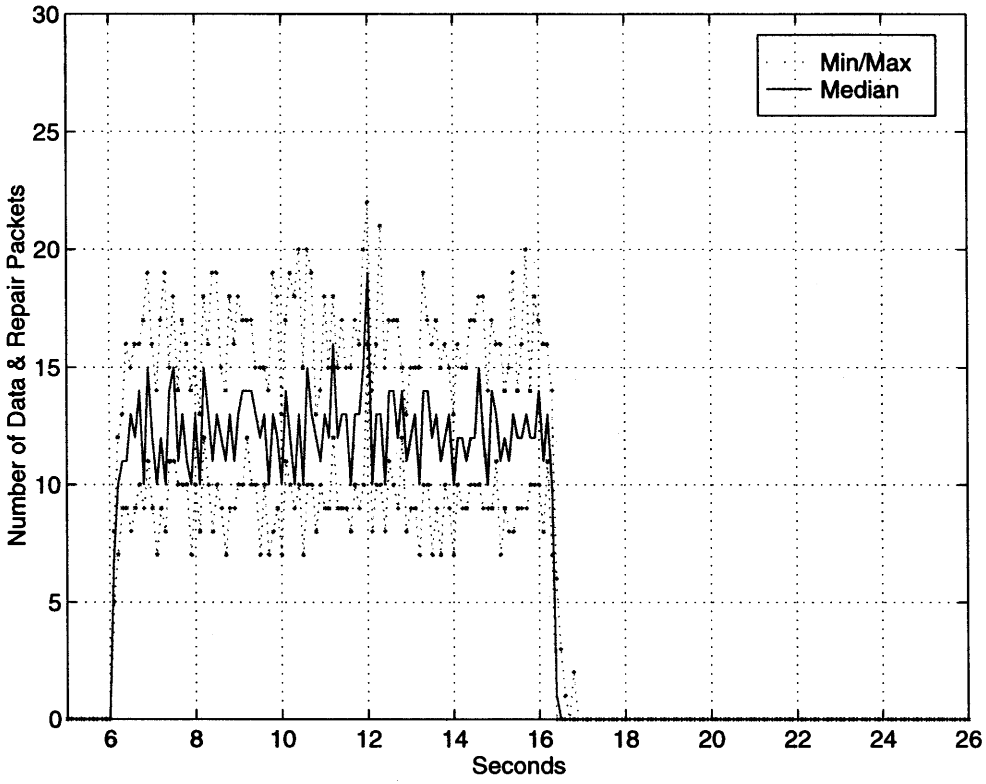


Figure 4-30 Data plus Repairs and NACK Traffic seen by Level 1 (Receivers 1 - 7) for SHARQFEC



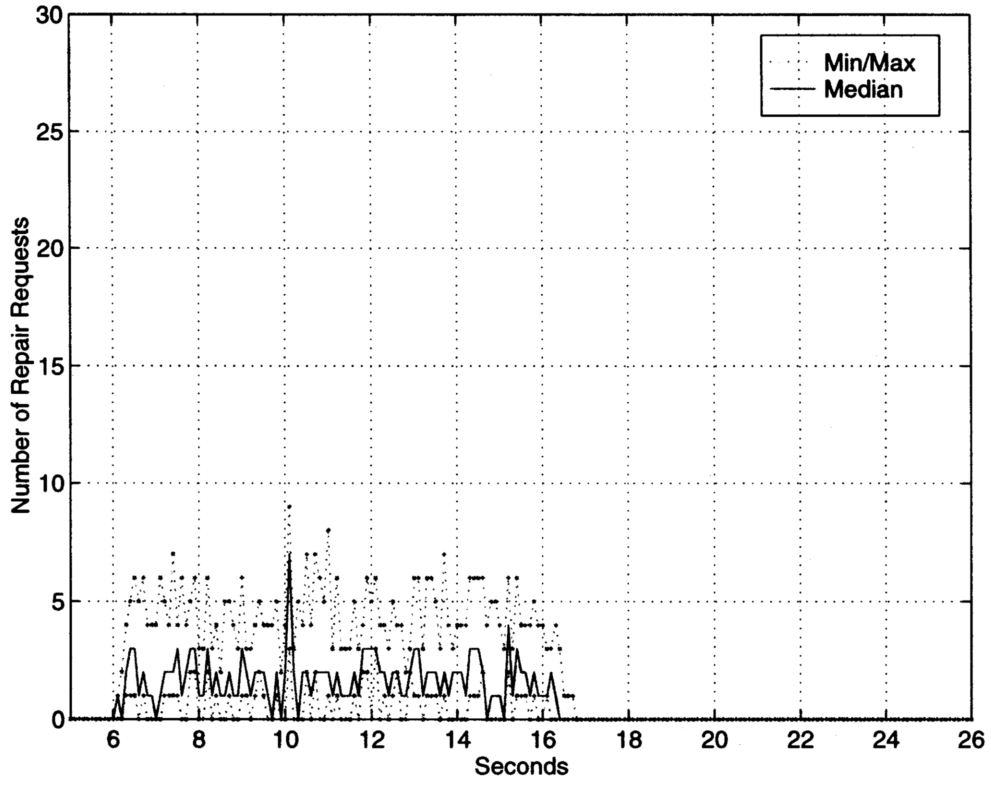
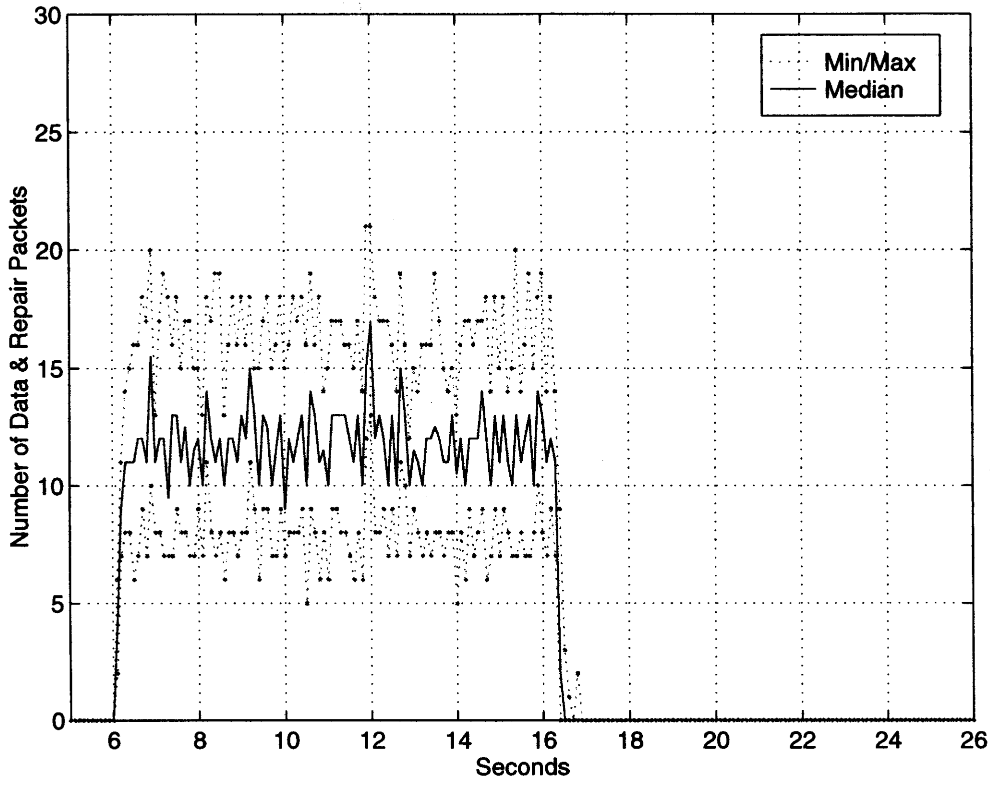


Figure 4-31 Data plus Repair, and NACK Traffic seen by Level 2 (Receivers 8 - 28) for SHARQFEC



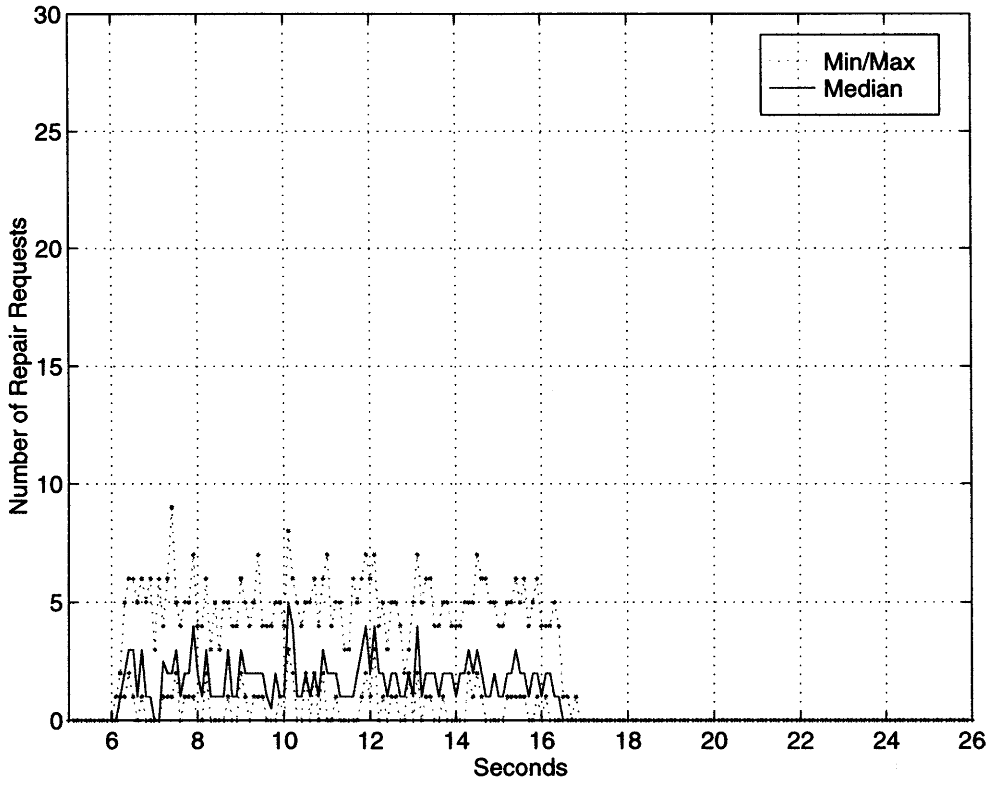


Figure 4-32 Data plus Repair, and NACK Traffic seen by Level 3 (Receivers 29 - 112) for SHARQFEC

4.3 Summary

The development of a reliable multicast data delivery mechanism that scales has been a much sought after goal ever since the inception of multicast delivery. Many attempts have been made with varying success to achieve this goal. The Scoped Hybrid Automatic Repeat reQuest with Forward Error Correction (SHAQRFEC) mechanism developed in this chapter merges the most promising features of the more successful methods—suppression and Forward Error Correction (FEC)—into a loose distributed hierarchy. Hierarchical methods involving Automatic Repeat reQuest (ARQ) have been attempted before, however, SHARQFEC is the first hierarchical scheme to employ both ARQ and FEC in a hierarchy composed of administratively-scoped regions.

Simulations show that SHARQFEC's use of disjoint regions affords greater repair traffic localization than possible with other protocols that use hierarchy or hybrid ARQ/FEC alone. This reduction in repair traffic is most pronounced near the source combined with the reduced session overhead afforded by SHARQFEC's indirect session maintenance algorithm makes SHARFEC particularly attractive for large scale distribution applications. Finally, SHARQFEC's end-to-end design does not require the modification of existing multicast routers in order to work. Therefore, it can be deployed in an incremental fashion as new applications become available without adversely effecting existing applications.

Chapter 5

Hierarchically Partitioned Reliable Transport Protocol (HPRTP)

A multicast transport protocol's ability to scale is increased in two fundamental ways. The first is to enhance the ability of session members to limit the traffic they must process, and the second is to distribute the load among them. The Hierarchically Partitioned Reliable Transport Protocol (HPRTP) developed in this chapter combines this realization with the research presented in the previous three chapters to provide a flexible scalable scheme for data delivery.

5.1 Logical Data Stream Control Channel

HPRTP enhances a receiver's ability to minimize its needs for bandwidth by separating traffic for transmission over disparate channels on the basis of function. Receivers subscribe to these channels according to the data they carry.

The previous chapter developed a reliable delivery protocol that uses four message classes:

- Original Data Transmissions
- Requests for Repairs (NACKs)
- Repair Responses
- Session Maintenance

Given this taxonomy, one finds that the NACKs and repair responses serve not one but two purposes. NACKs notify sources and capable receivers that a repair is needed. They also inform other receivers that experienced the same loss to suppress their own NACK by delaying its transmission. Similarly, repair responses not only effect repairs in receivers that need them, but also cancel the repair response timers in potential repairers.

The size of each of these traffic types is important. NACKs are typically very small, while repairs are typically much larger. Receivers in need of repair use all of the information contained in both of these packet types. However, session members effecting repairs use all of the information in the NACK, but only the header information in the repair; the repair data itself is discarded. This is shown in Figure 5-1.

Savings can be made if a way can be found to separate the repair traffic into two parts. The first part is a small message the same size as a NACK that informs the source and repair-capable receivers to cancel their reply timers. The second is the actual repair, and is only delivered to receivers that need that it. Receivers could possibly join and leave a separate repair channel to receive some repairs and not others. However, the overhead associated with the joining and leaving of multicast groups in this rapid manner overwhelms any benefit derived from doing so.

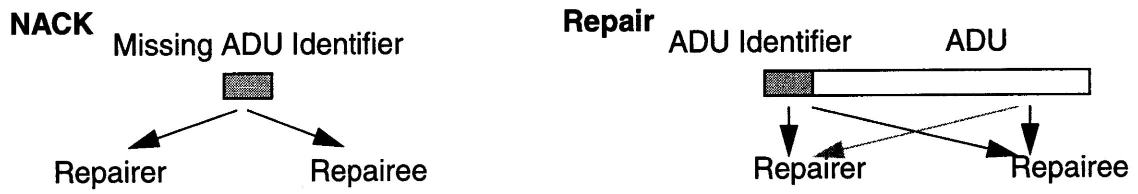


Figure 5-1 NACK and Repair composition and utilization

HPRTP merges these observations with another—not all receivers will subscribe to all Logical Data Streams (LDSs) or all partitions—to segregate data in the following manner.

- Traffic from different LDSs is carried over separate channels.
- Repairs for each LDS/partition combination are transmitted separately from the data.
- NACKs for each LDS are aggregated into a single Logical Data Stream Control (LDSC) channel.
- Since SHARQFEC relies on the ability to transmit both NACKs and repairs at different scopes, the LDSC and repair channels are replicated at each valid scope smaller than the maximum allowable for that partition.

The application of this segregation policy yields a hierarchy of multicast channels as shown in Figure 5-2. Note that Figure 5-2 shows only the channel associated with a single LDS, should more than one LDS be present within an object this hierarchy will be repeated for each LDS present.

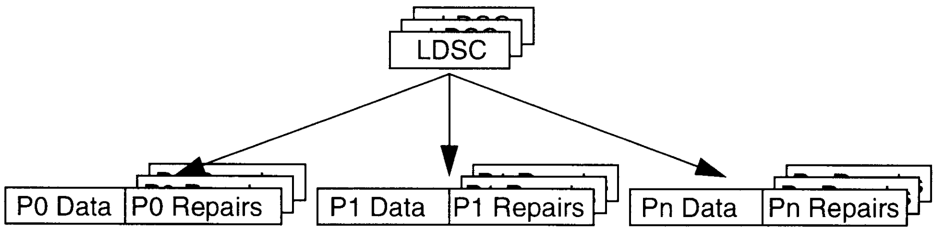


Figure 5-2 Multicast group hierarchy for a single Logical Data Stream

This hierarchy allows repairs to be effected as follows. Consider a receiver listening to partition P0, and receiving data over the P0 Data group. Should this receiver miss a packet, it will transmit a NACK to the smallest scoped LDSC group. This NACK will be received by all receivers that subscribe to this LDS within the administratively scoped region corresponding to the smallest scope LDSC channel. When one of the receiver's reply timers expires, it will transmit the repair to the smallest scope P0 Repairs group, and also a repair-indication to the smallest scope LDSC group. The repair cancels the repair-request timers on receivers needing the repair, while the repair-indication cancels the reply timer on other receivers preparing to send a repair. Should no receiver in the smallest scope zone respond, the SHARQFEC algorithm will increase the scope of each subsequent request until a response is heard.

This hierarchy allows all receivers—regardless of whether they subscribe to a particular partition or not—to effect repairs for any partition. The scope of both NACKs and repair requests is minimized and hence bandwidth is reduced. Receivers are free to subscribe, or not subscribe, to repair groups and can therefore tailor the object's delivery to their own needs without having to send feedback to the source. Furthermore, receivers hearing NACKs corresponding to partitions other than the ones to which they currently subscribe may either respond to or ignore them. Whether they choose to respond or not is a local decision that is made according to a receiver's capabilities.

5.2 Intra-Object Control Channel

The segmentation of an object's data into Logical Data Streams (LDSs) and Partitions requires session members to exchange and store more meta-information than before. The new meta-information they must exchange includes:

- Session and sender hint information about the object, such as whether or not it can be cached, the format of the common ADUI, the rate at which common ADUs are consumed during normal speed playback.
- How many LDSs are used.
- Descriptions of each LDS: LDSC address, ADUI formats, data type, range of valid ADUIs, and nominal transmission rate for real time playback.
- Descriptions of each Partition: the address of the Data and Repair groups for each LDS, the range of ADUs covered, the transmission start time, the transmission rate.
- How to request and create partitions.

HPRTP provides a separate channel called the Intra-Object Control (IOC) channel for session members to exchange this information. The use of a separate channel allows them to access this meta-information without joining a particular LDS or partition. HPRTP further exploits the structures imposed by the creation of SHARQFEC's hierarchy described in 4.2.2 on page 58, and allows the IOC channel to be replicated in the same manner as the repair and LDSC channels. These reduced-scope IOC channels are used for the negotiation, creation, and advertisement of broadcast partitions and dynamic partitions used to effect late-joins.

An example of how the various multicast groups associated with Data, Repair, LDSC, and IOC channels are related is shown in Figure 5-3. Given the following conditions:

- the administrative scope of organization-local-scope (scop = 8)
- two logical data streams
- two organization scoped partitions (P0, P1, scop = 8) that originate in zone *Z1*
- one locally scoped late join partition (P2, scop = 5) that originates in Zone *Z3*

receivers R4 through R7 in Figure 5-3 would expect to see the address tree shown in Figure 5-4.

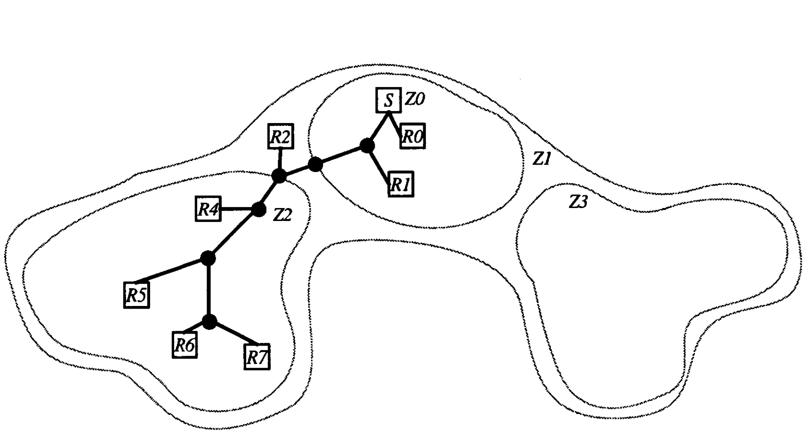


Figure 5-3 A typical 2 tier scoped delivery topology

This address tree is composed of 11 multicast groups of administrative scope; scop = 8, from the organizational scoped address pool defined for zone Z1. Another 11 multicast groups of administrative scope; scop = 5, from the site local scope address pool defined for zone Z2. Traffic carried within the multicast groups with scop = 8 is heard by all session members in all four zones, but the traffic carried on the multicast group shown with scop = 5 is confined solely within zone Z2.

The separation is important since messages concerning the existence and attributes of partition P2 are transmitted on the IOC channel with scop = 5, not its parent, with scop = 8.Therefore, P2 messages are only heard by receivers within Zone Z2. Receivers located outside of Zone Z2 are not made aware of partition P2's existence. The other site local zones, Z0 and Z3, can have their own locally scoped partitions that use the partition identifier for P2. These partitions will be similarly confined to their own local scope zones and hence be unknown to the receivers in zone Z2.

Each zone has its own address tree that is constructed as follows. First, the tree inherits all the addresses from zones with larger scope. Next, the IOC and LDSC channels along with each Repr channel with larger scope are duplicated at the current scope. These replicated channels afford traffic localization as required by SHARQFEC. Finally, any new partitions that originate within this scope level are inserted into the tree. Although this scheme consumes many multicast groups, it is designed to consume them from the smallest administrative scope zones possible. This should minimize the effects of multicast group shortage, since groups assigned in one zone can be reused in other disjointed zones of the same scope.

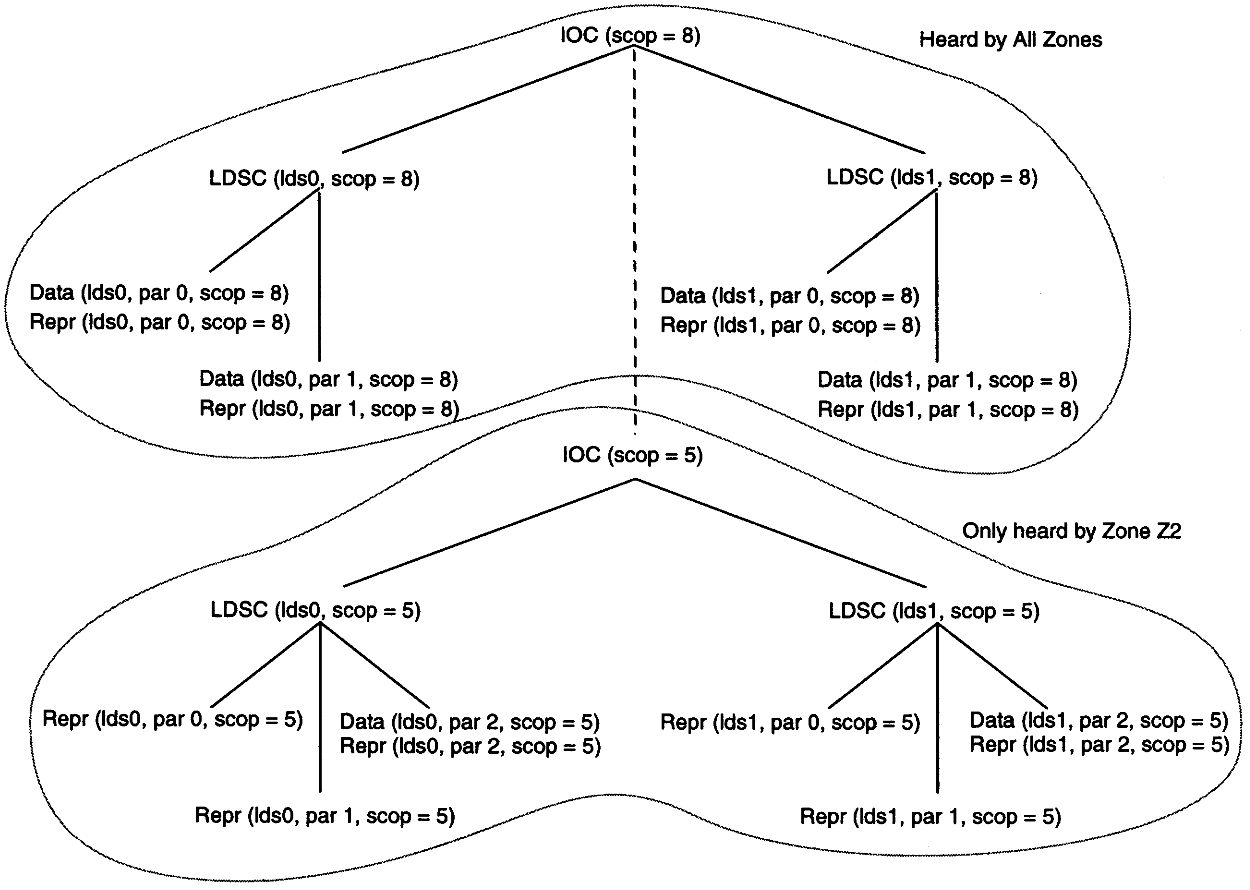

IOC (scop = 8)
Heard by All Zones
LDSC (lds0, scop = 8)
LDSC (lds1, scop = 8)
Data (lds0, par 0, scop = 8)
Repr (lds0, par 0, scop = 8)
Data (lds1, par 0, scop = 8)
Repr (lds1, par 0, scop = 8)
Data (lds0, par 1, scop = 8)
Repr (lds0, par 1, scop = 8)
Data (lds1, par 1, scop = 8)
Repr (lds1, par 1, scop = 8)
IOC (scop = 5)
Only heard by Zone Z2
LDSC (lds0, scop = 5)
LDSC (lds1, scop = 5)
Repr (lds0, par 0, scop = 5)
Data (lds0, par 2, scop = 5)
Repr (lds0, par 2, scop = 5)
Repr (lds1, par 0, scop = 5)
Data (lds1, par 2, scop = 5)
Repr (lds1, par 2, scop = 5)
Repr (lds0, par 1, scop = 5)
Repr (lds1, par 1, scop = 5)

Figure 5-4 HPRTP address tree observed by Receivers R4 through R7 in Figure 5-3

5.3 Summary of HPRTP Message Types

HPRTP's hierarchy has three levels. Each level is designed to carry certain types of traffic. At the top of the hierarchy, level 0, the Intra-Object Control (IOC) channel is used to exchange session-level information and the meta-information needed for accessing the lower levels. The middle level, level 1, consists of a number of Logical Data Stream Control (LDSC) channels. These channels are used for transmitting NACKs and Repair Indications. The lowest level, level 2, consists of the Data and Repair channels for each LDS/partition combination. These two channel types carry the actual data and repairs. A summary of the valid message types for each level is shown in Table 5-1. The format of each of these messages, and a description of the information they carry can be found in Appendix A.

Msg ID	IOC (level 0)	LDSC (level 1)	DATA/REPR (level 2)
0	Session	Sequential NACK	Payload Data
1	ZCR Challenge	SHARQFEC NACK	SHARQFEC Payload Data
2	ZCR Response	Application Specific NACK	Heartbeat
3	ZCR Take Over	Sequential Repair IND	
4	LDS Description REQ	SHARQFEC Repair IND	
5	LDS Description	Application Specific Repair IND	
6	Partition Description REQ		
7	Partition Description		
8	Session Description REQ		
9	Session Description		
10	Sender Description REQ		
11	Sender Description		

Table 5-1 HPRTP Message Types

5.4 Dynamic Partition (DP) Creation

HPRTP supports the automatic creation of partitions to be used for providing late-join data. These partition tend to be short-lived and are created as needed; as a consequence, they are called dynamic partitions (DPs). Receivers are not mandated to create DPs when the combination of the existing contents of their cache and the data they can procure from existing partitions results in the inability to meet peer's request for data on time. The option exists to wait until an existing partitions can deliver the data. Nor are session members required to respond and subsequently serve the missing data for another receiver's DP. It may well be that the client has insufficient resources to meet the extra demands of serving a DP.

Be that as it may, DPs provide a useful mechanism for effecting the delivery of late-join data. DPs help to further distribute the load of delivering an object's data amongst the session members, and therefore increase scalability. To understand how DPs are created, one needs to understand that they are the logical extension of the localized recovery procedures made popular by SRM. If one views SRM's repair mechanism as a tool for repairing small losses, one can view the DP mechanism as a big brother to SRM that is used to repair larger losses. The fact that the losses are substantially larger results in a key difference between the SRM and DP mechanisms. Unlike SRM, DPs cannot tolerate duplicate repairs. The creation of duplicate multimegabit repair streams by multiple session members to effect the repair of 10 minutes of a missing television show would have catastrophic effects on the network. Therefore, the creation of DPs involves additional safeguards to ensure that duplicate responses do not occur.

The creation of a DP consists of four phases; *Tender*, *Bidding*, *Acceptance*, and *Confirmation*.

When a client determines that the data it needs is not in its cache and will not be delivered in time by existing partitions, it enters the *Tender* phase. Upon doing so, it creates a partition description request message (see Appendix A) containing the following information

- the DSI of the missing data
- a list of missing ADUs
- the minimum partial reply it is willing to accept

This tender is then multicast to the other session members over the IOC channel with the smallest known administrative scope.

Upon receiving this message, other session members check to see if they can meet the request (both in terms of having the data cached and also having the DP response turned on). If they can, they enter the *Bidding* phase by first waiting for a short random amount of time, and then sending off a bid in reply indicating their willingness to supply this missing data. The length of this delay is calculated in the same manner as for repair responses within SRM. Responding session members will suppress their own bids upon hearing an equivalent or superior bid from another session member. The minimum partial reply field allows session members not possessing the entire contents of the request to make a secondary or partial bid for the data they do have cached. Session members who have less than the minimum partial reply cached do not return bids

Should the client not receive bids that can completely satisfy its tender, it saves any partial bids received, and retransmits the tender again on the IOC channel with the next-highest scope. The client increases the scope in this manner until the highest possible scope has been reached. Should no bids ever reach the client, or should the partial bids received not cover the entire tender, the receiver informs the application that the creation of a DP failed.

Session members responding with bids in the Bidding phase do so after a short random interval in order to reduce the number of bids. Since the same scoped hierarchical SRM-like mechanisms used for SHARQFEC are employed here and partial bids are allowed, it is likely that more than one bid may result. To differentiate between bids, each bidder tags its bid with its own SSRC.

When the client receives one or more bids that result in its tender being covered, it enters the *Acceptance* phase. In this phase, the client accepts one or more bids and notifies unsuccessful bidders that their bids have been rejected. The determination of which bids to accept is made using several criteria that attempt to minimize the number and scope of partitions that must be created to satisfy the tender. First, bids for larger ADU ranges are accepted over smaller partial bids. Second, should a secondary bid of smaller scope provide partial coverage of a larger bid of larger scope, the secondary bid is accepted and the larger is bid truncated.

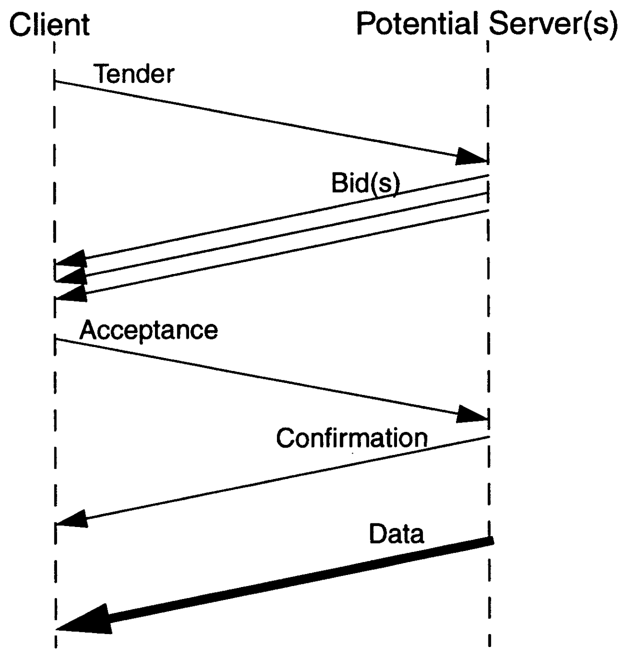


Figure 5-5 Dynamic Partition Creation

Once the client has determined which bids it will accept, it enters the *confirmation* phase. In this phase, the client notifies successful bidders by retransmitting their partition description request outlining their bid at the same scope as it was sent. However, this time the message contains the DSI of the successful session member in addition to the DSI of the missing data. When this message is received by bidders, each checks the DSI and ADU range of the successful bidder against it own. If the DSIs match, the bidder knows that its bid has been accepted and it makes preparations to start transmission of the new DP. If the DSIs do not match, the ADUI range overlaps with the bid extended, or the session member did not return a bid, the session member saves the partition description for possible future use.

5.5 Congestion Control

Any higher level transport protocol that adds reliability must address the issue of congestion control, since the IP protocol provides a best-effort service with no QoS guarantees. If this issue is not addressed, congestion events risk becoming self-perpetuating, since the extra traffic required to request and effect repairs exacerbates the congestion, causing further losses, which in turn generate more repair traffic. Thus, a single congested session risks entering an unrecoverable state, that, even worse, causes other sessions to become congested as well.

Congestion control mechanisms for point-to-point traffic are relatively easy to design and implement; for example, the receiver can instruct the sender to slow down or lower its bit rate [42, 102]. However, designing congestion control mechanisms for point-to-multipoint traffic is considerably more difficult. Instead of a single receiver, there are now multiple receivers, with the link capacities connecting them to the sender possibly varying by orders of magnitude. Employing a feedback rate-limiting mechanism only serves to reduce the level of service provided to the lowest common denominator.

For this reason, several congestion control methods have been proposed that attempt to give a receiver more than one level of service to choose from.

The Destination Set Group (DSG) mechanism proposed by Cheung *et al.* [17] allows the receivers to organize themselves into groups that experience similar amounts of loss and throughput and to instruct the sender to simulcast data at a rate that ensures a similar level of performance within each group. Receivers join a particular group, and together they decide how to set the bit rate for that group's channel. Should a receiver observe that the other members of the group are reporting greater losses than its own and are unwilling to increase the bit rate further, it may decide to leave that group and become a member of a group whose channel is operating at a higher bit rate. Similarly, a receiver may find that it is a laggard compared to its peers and decide to subscribe to a lower bandwidth group.

The Receiver-Driven Layered Multicast (RLM) mechanism developed by McCanne [54], sees the sender employ a layered transmission scheme with multiple streams. RLM operates in an open loop manner and relies on receivers to police their own use of network resources. Receivers actively measure and record the losses associated with each layer within a presentation. When congestion occurs, one or more layers will experience significant losses. Should the losses exceed a threshold, the receiver will drop a layer, thereby reducing its demands for bandwidth and also the quality of the reconstructed image or audio. Should no losses be experienced, receivers may decide to listen to additional layers, thereby increasing both the bandwidth required and the quality of the reconstructed image or audio.

This mechanism relies on topologically proximate receivers to arrive at a similar decision as to which layers to subscribe to at roughly the same times. Should this not happen, the network will continue to deliver an unwanted layer to receivers when they believe it has been dropped. These receivers will then mistakenly interpret the persisting congestion as evidence that they should drop further layers. To help synchronize the behavior of receivers within localized regions of the network, each receiver notifies nearby receivers of which layers they subscribe to and when they decide to add or drop them.

Likewise, HPRTP employs a number of mechanisms to limit congestion. One will note that the various traffic-limiting mechanisms already described play a major role in reducing congestion. SHARQFEC's combination of administrative scoping and a hybrid ARQ/FEC technique adds reliability and minimizes the both the volume and scope of repair traffic, while the use of Dynamic Partitions (DP) minimizes the scope of late-join traffic.

5.6 Security Considerations

Since HPRTP is a multicast protocol designed for the mass distribution of media, security is an issue that must be considered. The fact that HPRTP relies heavily on the use of multicast transport and distributed caching and retransmission by the receivers makes the issue of security even more important as the media is not delivered under a receive-use-immediately-and-discard model. If due care is not paid to this issue, HPRTP's usefulness will be severely limited.

5.6.1 HPRTP Denial of Service Attacks

HPRTP's use of multicast significantly increases its susceptibility to denial-of-service attacks. This susceptibility is further increased by its reliance upon distributed caching, and the SHARQFEC session maintenance algorithm's use of summarized RTT information. Below are a number of ways in which HPRTP could be attacked if its messages were transmitted as clear text.

- An attacker could supply false addresses for the LDSC, DATA, and REPR channels in a corrupt Session Description message, thereby prohibiting receivers from tuning into the correct groups.
- An attacker could supply erroneous ADU formats in LDS Description Messages, thereby preventing receivers from understanding the data being transmitted.
- An attacker could supply false ADU ranges in LDS or Partition Description messages, thereby preventing receivers from subscribing to the correct partitions.
- An attacker could make numerous false requests for Dynamic Partitions, thereby tying up receiver and network resources.
- An attacker could respond to Dynamic Partition requests and then not send the data, thereby causing the receiver's cache to fault.
- An attacker could issue false ZCR takeover messages, thereby corrupting the distribution tree and causing extra repair traffic.
- An attacker could issue false repair requests, thereby causing excessive repair traffic and network congestion.
- An attacker could inject false or duplicate data or repair traffic, thereby causing the token bucket rate limiters of the receivers to shut down.
- An attacker could respond to Dynamic Partitions and supplant false data instead of the original data.

5.6.2 Authentication, Confidentiality, and Billing

As one can see above, HPRTP is particularly vulnerable to any number of attacks when deployed in a clear text fashion. Fortunately, security is currently being examined within the IETF with the aim of generating a general purpose solution that can be implemented directly within the IP protocol. Once this aim is achieved, application programmers will be freed of the task of implementing their own solutions. To date, three Requests For Comments (RFCs) have been generated that defined a general security architecture: RFC 1825 [85], and two extensions to the IP protocol, RFC 1826 [86] and RFC1827 [87]. Rather than develop yet another solution, HPRTP will use the security solutions proposed by these and future RFCs when they become available. To understand the specifics addressed by the current RFCs, the remainder of this section will examine how HPRTP relates to the issues of *Authentication*, *Integrity*, and *Confidentiality* as defined within RFC 1825 and furthermore to the issues of *Anonymity* and *Billing*.

First, consider the issue of *Authentication*, which is defined within RFC 1825 as "the property of knowing that the data received is the same as the data that was sent and that the claimed sender is in fact the actual sender." This service is provided through the use of the IP Authentication Header [86], which adds some extra encrypted information to each packet that is generated from "all of the fields in the IP datagram (including not only the IP Header but also other headers and the user data), which do not change in transit." This issue is particularly tricky since HPRTP assumes the extensive use of caching. Data originating from a particular source may pass through one or more caches first and therefore will most likely appear to originate from the last cache rather than the originating source. One solution to the dilemma that caching poses is to include extra information in the HPRTP packet that enables a receiver to determine that the packet was generated by a trusted HPRTP implementation. This small addition to the HPRTP protocol would effectively create a web of trusted caches from each receiver in a session.

Integrity is defined within RFC 1825 as "the property of ensuring that data is transmitted form the source to destination without undetected alteration." One will note that the IP Authentication header already provides this service as the user data is included in the calculation of the authentication data. Hence, providing authentication functionality within HPRTP addresses this issue as well.

Confidentiality, defined within RFC 1825 as "the property of communicating such that the intended recipients know what was being sent but unintended parties cannot determine what was sent," is not addressed through the use of the IP Authentication Header. IP's well-known vulnerability to packet-sniffing by intermediate nodes is further exacerbate by HPRTP's reliance on multicast. Nodes need not be part of the chain of nodes immediately between the source and a receiver to receive the traffic between the two; all they need do is join the necessary groups and the traffic will automatically flow to them. For this reason, one must assume that all traffic carried over HPRTP can and will be intercepted.

Fortunately, the problem of intercepted traffic can be solved through the use of encryption that renders intercepted data readable only by those receivers possessing a valid key. The encryption algorithm used should be sufficiently strong enough to prevent a receiver from performing a traffic analysis attack that sees it recover the data by reverse engineering the algorithm employed. Since the ability to reverse engineer encryption schemes depends mostly on processing power, key lengths should be increased regularly. Basic support for encrypting IP packets is provided by the mechanisms defined within RFC 1827, "IP Encapsulating Security Payload (ESP)."

HPRTP's intended use as a means for distributing streaming entertainment media within a commercial environment raises issues apart from the issues of authentication, integrity, and confidentiality, as just discussed. First, confidentiality must be strengthened to incorporate notions of *Anonymity*. Not being able to decode a stream of data between a sender and receiver does not prevent a third party from making an association between the sender and receiver. Consider a situation in which a third party, say a militant antipornography group, uses an HPRTP receiver to issue a fake Dynamic Partition request in order to determine the identity of receivers currently viewing a particular adult-oriented pay-per-view program.

This problem is partly addressed with the use of randomly generated DSIs for the purposes of session-member identification. Unfortunately, this is still insufficient since packets still carry the IP address from which they originated inside their IP headers. Deliberately setting the IP source-field to zero to prevent the association between sender and receiver is not a solution either, since these packets will most likely be rejected by packet filters designed to prevent IP spoofing. However, setting the IP source to a valid dummy address from within the same subnet is a solution. Packets so addressed can pass through filters which can still reject traffic from outside hosts. Further work is needed in this area.

Finally, one must pay at least some thought to the issue of *Billing*. Within the commercial realm, being able to deliver data in a scalable fashion is of little use if one cannot determine who consumed it and therefore charge for its use. This issue can be thought of a special case of Nonrepudiation, defined by RFC 1825 as "the property of a receiver being able to prove that the sender of some data did in fact send the data even though the sender might later deny to ever having sent the data," in which the data in question is the request to open and access a particular session, and the roles of the receiver and sender are played by the movie-server and the client respectively. Thus, the application of the aforementioned authentication and encryption mechanisms to session information as well as to the data itself should be sufficient for validating a receiver's intent to consume a particular media object and hence its obligation to pay.

It should be noted that both the authentication and encryption mechanisms alluded to here and in RFCs 1825, 1826, and 1827 assume the confidential delivery of keys to the receivers. This is a nontrivial task that it is still subject of on going research [88, 55].

5.7 HPRTP encapsulation of other Multicast Protocols

At the time of this writing, reliable multicast protocols are still the subject of intense research and as yet there is no one clear standard for reliable multicast transport. For this reason, the Internet Research Task Force (IRTF) formed a Research Group in late 1996 to study the requirements for reliable multicast (RM) and to start the work needed to bring either a small set of reliable multicast protocols or a generic reliable multicast protocol into the Internet Engineering Task Force (IETF) for eventual publication and use as an Internet standard. However, the work of the RM Research Group of the IRTF is ongoing, with a number of protocols being put forward as potential solutions, some of which incorporate hierarchical distribution and caching and some of which do not.

HPRTP's inclusion of a flexible ADUI meta-description allows for the easy addition of hierarchical caching and the replication of data into a hierarchy of partitions, as shown in Figure 5-6.

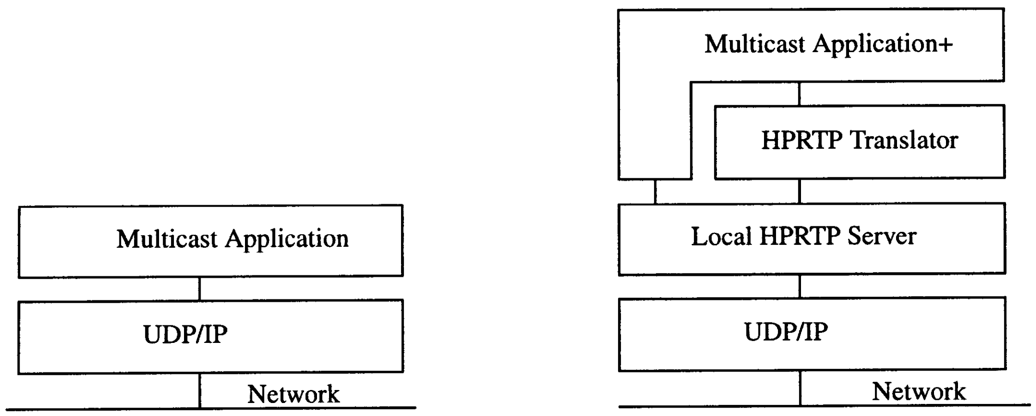


Figure 5-6 Architecture for encapsulating other multicast protocols within HPRTP

Chapter 6

Partitioning Schemes

The ability to transmit multiple phase-shifted copies of a media object is a key tool for increasing scalability. A phase or partition may deliver data at a rate that is slower, faster, or the same as the rate of consumption at a receiver; it may start at an arbitrary point in the object and last for an arbitrary length of time. The task is therefore to allocate the available partitions so as to maximize scalability and flexibility.

This task is usually achieved through the use of batching algorithms that aggregate the requests from many receivers into a smaller number of multicast channels. Two broad classes of batching aggregation techniques exist: user-centered, and data-centered [72]. User-centered aggregation techniques allocate data channels in response to user requests. Conversely, data-centered techniques allocate transmission channels to a piece of the object in a pre-defined manner and rely on the receivers to determine which channel they should listen to in order to receive the data required. Examples of user-centered batching algorithms can be found in [6, 21, 22, 71] while data-centered ones can be found in [34, 72].

Recently, several data-centered methods have emerged that achieve close to on-demand control of a movie's playback by partitioning the movie into several parts that are then cached in the receiver before viewing. To facilitate the following discussion, the same notation used in Hua and Sheu's paper describing Skyscraper Broadcasting [34] is adopted:

- B - The total outgoing bandwidth supportable from each sender/server (this takes network bottlenecks into consideration as well).
- M - The number of media objects to be transported.
- D - The total playback time of the media object in minutes.
- K - The number of partitions (and hence multicast groups).
- b - The average display rate of the media objects in Mbit/sec.

This set of parameters is augmented by the following:

- β - The total incoming bandwidth supportable by each receiver.
- S - The amount of storage available at the receivers.
- R - The number of receivers.
- L - The maximum access latency.
- The size of administratively scoped regions.

The first section in this chapter examines a number of broadcast schemes for allocating partitions that attempt to optimize performance based upon the variables just mentioned; while the second section examines how one attempts the same optimization using purely on-demand schemes.

6.1 Data-Centered Schemes

Data-centered schemes are so named because the senders transmit the object data over partitions whose characteristics are predefined, with no receiver consultation. Receivers are unable to affect the way the data is delivered and must manage with what can be obtained. As a consequence, data-centered schemes tend to work best when delivering very popular objects to large number of receivers, each of which performs similar accesses. One media type that fits this description is recently released movies, where a small-to-medium number of objects are favored out of a much larger set. Studies in [21,22] have shown that the access distribution for movies can be described by a Zipf distribution with a skew of 0.271, meaning that one can satisfy roughly 80% of all requests for movies with only a small selection of 10 to 30 titles. Given the popular nature of movies as a form of entertainment, the various data-centered schemes discussed herein will focus on the delivery of single, linearly viewed streaming media.

6.1.1 Linear Partitioning

The first data centered-scheme examined attempts to decrease the access latency, by preassigning multiple partitions so that the object data is simultaneously transmitted over multiple channels that continuously loop from beginning to end. The start times of each channel are spaced equally so that at most one will have to wait $\frac{D}{K}$ time units before a partition transmits any given part of the object. This scheme is currently employed by analog cable television services that schedule the same Pay-Per-View (PPV) movie to start on different channels at half-hour offsets. An example of linear partitioning for a 2 hour movie into six partitions spaced 20 minutes apart is shown below in Figure 6-1

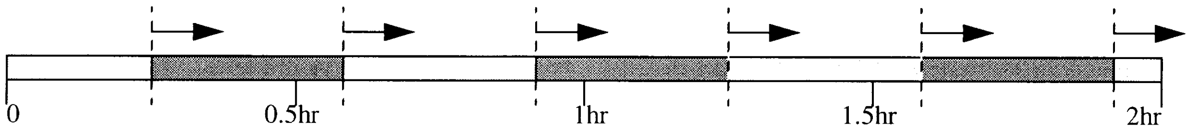


Figure 6-1 Linear Partitioning Example

Linear Partitioning is infinitely scalable, since the server load and network bandwidth remain constant, regardless of how many receivers tune in. Several other benefits arise; for example, the granularity is decreased so that, on average, a receiver need only wait $\frac{D}{2K}$ time units before a partition will broadcast any given part of the movie. In addition, when given sufficient local storage the receiver has the ability to download an entire object in less time than it takes to play it. The receiver achieves this feat by tuning into multiple partitions at the same time and caching the data for later use. In the most aggressive case, the receiver tunes into all of the partitions and downloads the entire object in the inter-partition spacing time (for the example above, this corresponds to 20 minutes). Once the object is downloaded into the local cache accesses can be performed at will with minimal latency.

The ability to simultaneously download data presupposes that all the network links between the sender and the receiver have sufficient bandwidth to handle the increased data flow. It is also further assumes that receiver is capable of handling the increased data flow. These additional requirements on the receiver and its connection to the source result in a more expensive solution that may not always be possible or affordable.

Fortunately, HPRTP's design incorporates features to assist in the provision of more partitions to further reduce granularity and in mechanisms that restrict the scope of these additional partitions to the outer portions of the network, thereby reducing the load on the network backbone. An example of how one might deploy HPRTP capable servers to achieve these aims is shown in Figure 6-2. Here, the source broadcasts one or a few partitions at the desired scope that are received by an intermediate caching server, which subsequently saves the object data. It then rebroadcasts the data sometime later with smaller administrative scope, thereby time-shifting it and creating the new partitions. Network traffic can be further localized once the caching server has received the entire object, as it can leave the partitions sent by the original source and take over the role of broadcasting them locally with smaller scope for those local clients who need them. This example also shows HPRTP's adherence to the "works without modification, but works better managed" principle, as more partitions can be added by simply adding more caching servers where needed.

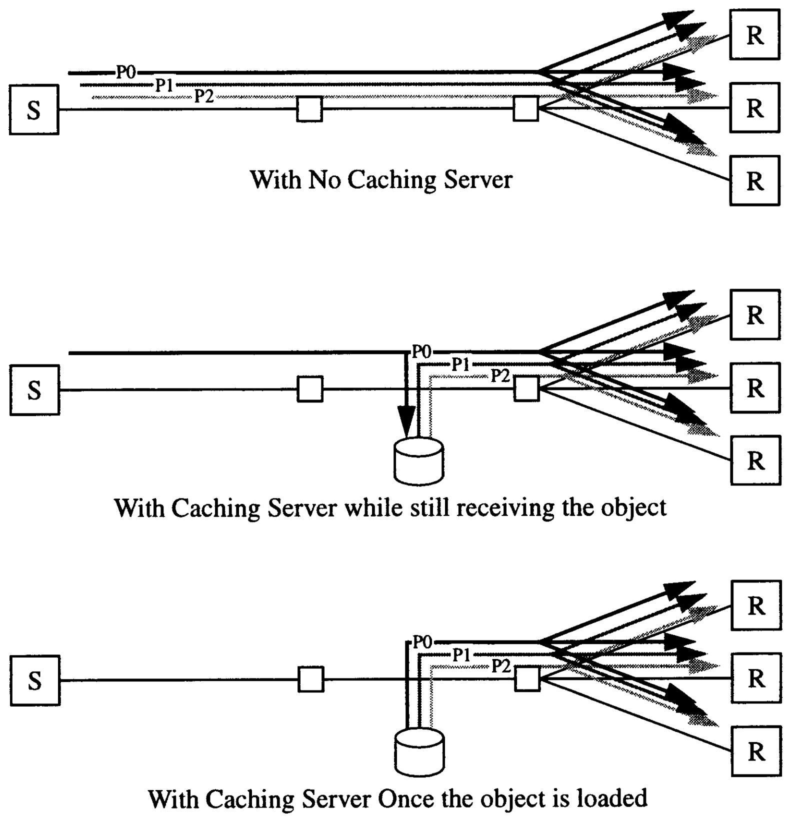

S
P0
P1
P2
R
R
R
With No Caching Server
S
P0
P1
P2
R
R
R
With Caching Server while still receiving the object
S
P0
P1
P2
R
R
R
With Caching Server Once the object is loaded

Figure 6-2 Increasing the number of Partitions Using a Caching Server

This example highlights a common way that HPRTP's premise of large-scale store-and-forward delivery is realized. Given the aim of preventing backbone congestion, one could amortize the cost of priming the caching server by allowing it to receive a single partition that trickles object data at much slower than real time at off-peak hours. Such a scheme would allow popular objects such as hit movies to be distributed to caching servers in remote cities in the early hours of the morning when network usage is lowest.

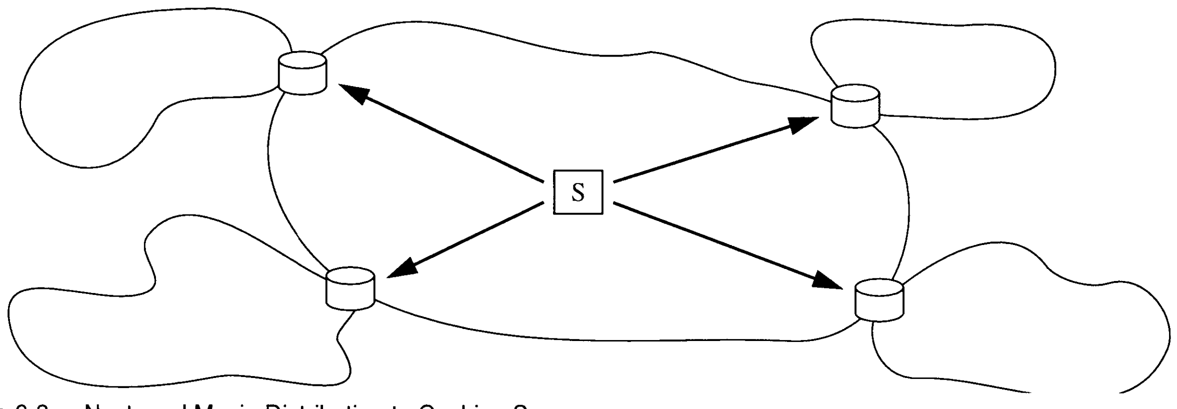


Figure 6-3 Nocturnal Movie Distribution to Caching Servers

6.1.2 "Skyscraper" Partitioning

One data centered partitioning scheme that attempts to minimize both the access latency L and the storage requirement S is Hua and Sheu's Skyscraper Broadcasting (SB) scheme [34]. Building upon the earlier work of Viswanathan and Imielinski on Pyramid Broadcasting (PB) [72], and Aggarwal, Wolf, and Yu's refinement, Permutation-based Pyramid Broadcasting (PPB) [6], Skyscraper Broadcasting uses the following design principles:

- Divide the available bandwidth, B Mbit/sec, equally amongst the M movie titles and further divide the $\frac{B}{M}$ Mbit/sec of bandwidth allocated to each title into K partitions of $\left\lfloor \frac{B}{MK} \right\rfloor$ Mbit/sec.
- Divide each movie into K segments, assign each segment to a different partition, and have the partition loop repeatedly over its assigned segment.
- Choose the lengths of the segments so that the receiver need tune into at most two channels at any time.

The key to SB's operation lies in the application of the third principle. SB uses the finite series generating formula given below to determine the relative lengths of each segment.

$$f(n) = \begin{cases} 1 & n = 1, \\ 2 & n = 2, 3, \\ f(n-1) & n \ mod \ 4 = 0, \\ 2f(n-1)+1 & n \ mod \ 4 = 1, \\ f(n-1) & n \ mod \ 4 = 2, \\ 2f(n-1)+1 & n \ mod \ 4 = 3, \end{cases}$$

Eqn 6-11

The first few terms in the series are as follows.

$f(n)$= [1, 2, 2, 5, 5, 12, 12, 25, 25, 52, 52, 105, 105, 212, 212, 425, 425, 852, 852, 1705, 1705...]

Each SB partition loops over its segment in a continuous fashion, as shown in Figure 6-4. Receivers wishing to receive the broadcast subscribe to each partition in turn, tuning in to each successive partition only after starting to play the contents of the previous partition. Due to the irregular way in which SB's segments increase, receivers are required to use two special loading routines. The first is called an "Odd Loader" and downloads segments that correspond to the odd terms in the SB series. The second is called an "Even Loader" and downloads segments that correspond to the even terms in the SB series. Sometimes receivers will have to load segments from two segments as shown in Figure 6-4.

As is the case for PB, SB's maximum storage requirements are tied to the size of the last segment. In an effort to minimize the size of the final partition, Hua and Sheu introduced a new parameter, W, that indicates an upper bound for elements in the SB series. Truncating the series in this manner allows one to trade access-latency for lower storage requirements.

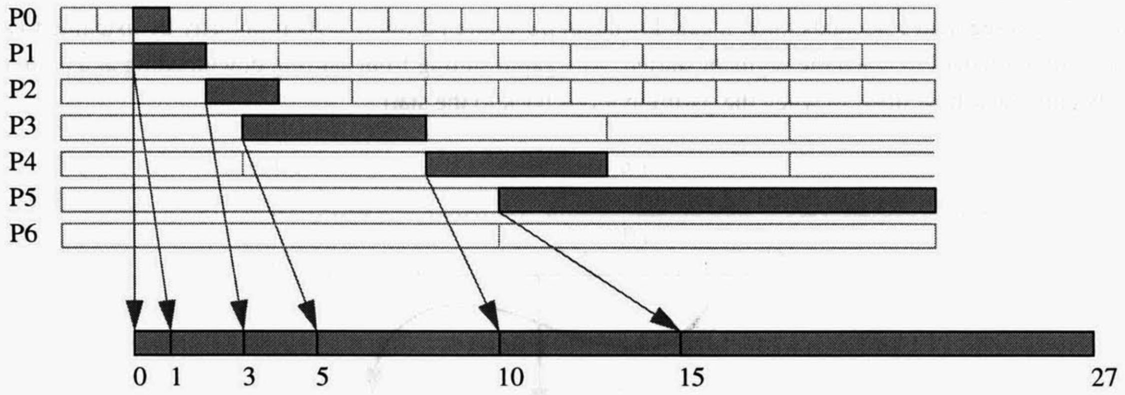


Figure 6-4 Skyscraper Broadcasting Bandwidth, Worst Case

The access latency for SB is simply defined as the longest period a receiver must wait until it can begin loading the first segment. Thus, SB's access latency is defined as

$$\text{Access Latency} = \frac{D}{\sum_{i=1}^{K} min(f(i), W)} = D_1 \text{ sec} \qquad \text{Eqn 6-12}$$

SB's maximum receiver bandwidth occurs when the receiver downloads from two segments while playing a previously cached one from disk. Thus, SB receivers require disk capable of supporting I/O at rates of

$$\text{Disk Bandwidth} = 2b \text{ or } 3b \text{ Mbits/sec} \qquad \text{Eqn 6-13}$$

SB's storage requirements are tied to the length of the partitions, which must commence loading before playback. The worst case results when the maximum possible amount of the final segment is cached before playback, thus the storage requirement for SB is defined as

$$\text{Storage Requirement} = 60D_1(W-1) \qquad \text{Eqn 6-14}$$

6.1.3 Truncated Exponential Partitioning (TEP)

Truncated Exponential Partitioning (TEP) is a new scheme developed in this dissertation that is similar to SB. The delivery channel is divided amongst the movies and then into a number of partitions, each transmitting a different movie segment in a looping fashion. Receivers tune into no more than two partitions at a time, saving data as necessary for future playback. The major differences between TEP and SB lie in the algorithm used to determine the segment length, and the manner in which segments are loaded.

Both PB and SB determine the segment lengths using algorithms that result in the receiver occasionally listening to only one segment instead of two. As a result, the segment sizes cannot grow as quickly as they would have had the receiver listened to two segments at a time. The ability to maximize the size of later segments is critical in minimizing the relative size of the first segment, which in turn minimizes the access latency. In addition, both PB and SB take great pains to load segments synchronously; that is, receivers wait until the partition loops back to the beginning before commencing downloading. In the case of SB this is unnecessary. If a receiver knows that a particular partition is carrying a segment *t* seconds long, it can download the entire partition asynchronously by listening to any *t* seconds without regard to when the segment started. Any gap resulting from starting downloading at a point other than the beginning will be filled in when the partition loops back to the start.

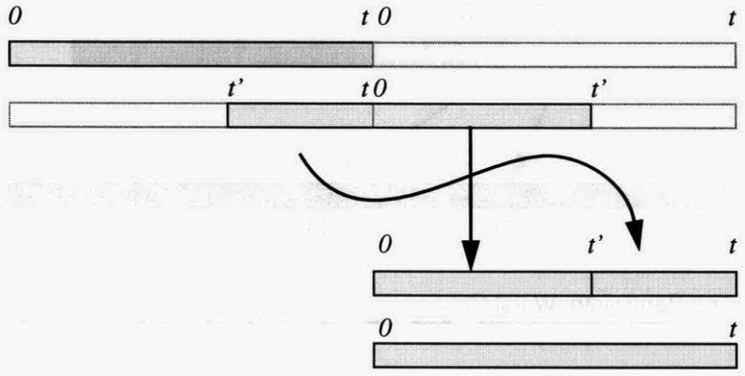


Figure 6-5 Asynchronous segment download

TEP's segment sizing algorithm, using these two observations, derives three simple rules for determining segment length:

1. Receivers download an entire segment before it can be played back.
2. Segment lengths should grow as fast as possible.
3. Segment lengths are set so the receiver commences asynchronous download of the next segment upon completing the download of a previous one.

The application of these rules results in three forms of TEP. The "infinite" form is the simplest and allows segment size to increase indefinitely. The second restricts the growth of segments in the same manner as SB and is known as the "basic" form. The third form, the "general" form, allows segments to be transmitted at faster than real-time rates in an effort to grow their successive segment sizes more quickly.

TEP (Infinite Form)

The infinite form of TEP is included as a proof to show that asynchronous downloading is viable. It is a degenerate case of the other two forms about to be discussed. Consider the simple example shown in Figure 6-6. In this example, a viewer attempting to watch a movie has just turned on his TEP Set Top Box (STB) at random time $t-1$. According to the rule 1, the STB will immediately start downloading the first segment from partition P0. As the first segment has been set to one minute, the viewer will have to wait one minute before his movie starts playing.

Given that the receiver commences playback immediately after saving the first segment, one can deduce that it will exhaust its supply of data exactly two minutes after tuning in. Thus, in order to satisfy rule 1, it follows that the second partition, P1, can have a repeat interval of at most two minutes. For the third partition, P3, one discovers that rule 2 precludes the receiver from joining it until it has left the first partition at time $t+1$. Given that the second partition's data will be exhausted at time $t+4$, the third partition can be at most three minutes long and will loop over minutes three to six of the movie. When one applies these rules, one discovers that the relative lengths of each partition form the Fibonacci sequence, excluding the first element.

$f_{EP}(n)$= [1, 2, 3, 5, 8, 13, 21, 43, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, 10946...]

Partition	Start Time (min)	Stop Time (min)	Partition Loop Time (min)	Segment Length (min)	Cumulative Distribution (min)
0	-1	0	1	1	1
1	-1	1	2	2	3
2	0	3	3	3	6
3	1	6	5	5	11
4	3	11	8	8	19
5	6	19	11	11	32
6	11	32	19	19	53

Table 6-1 TEP (Infinite Form) Partition Descriptions

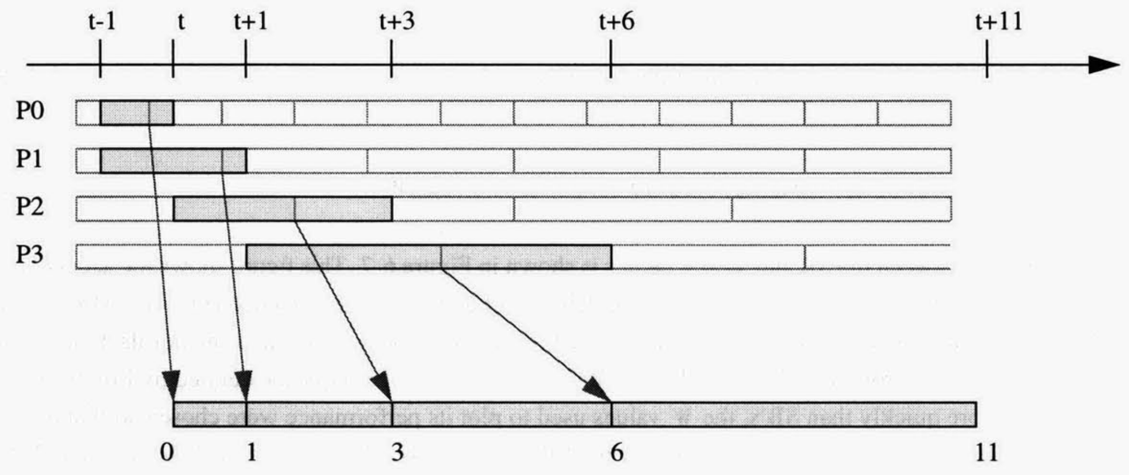


Figure 6-6 TEP (infinite form) example

TEP (Basic Form)

As one can see, the segment sizes used by the infinite form TEP scheme grow much more quickly than those used within SB. Thus, using infinite form TEP over SB results in either fewer partitions (and hence less bandwidth) to provide service with a given access latency, or lower access latencies for a given total server, or network bandwidth. Infinite form TEP can also be further refined to limit or truncate the relative partition sizes through the use of the parameter W in the same manner as used for SB partitioning. When this refinement is included, the basic form TEP results. Basic form TEP partitions have relative and actual lengths defined as

$$L_{TEPbf}(n) = min(f_{EPbf}(n), W)$$

where

$$f_{TEPbf}(n) = \begin{cases} 1 & n = 1 \\ 2 & n = 2 \\ f_{TEPbf}(n-1) + f_{TEPbf}(n-2) & n \geq 3 \end{cases}$$

Eqn 6-15

and

$$D_{n,\,TEPbf} = \frac{L_{TEPbf}(n)}{\sum_{1}^{p} L_{TEPbf}(n)} D$$

Eqn 6-16

respectively.

One notices from Figures 6-4 and 6-6 that the SB and TEP schemes are very similar. Furthermore, the access latency of each is dependent upon the size of the smallest partition, while the storage requirement is dependent upon the size of the largest partition. It is therefore not surprising that TEP's equations for access latency and storage requirements are similar to those for SB.

$$\text{Access Latency} = \frac{D}{\sum_{i=1}^{K} min(f_{TEPbf}(i), W)} = D_{1,\,TEPbf} \text{ sec}$$

Eqn 6-17

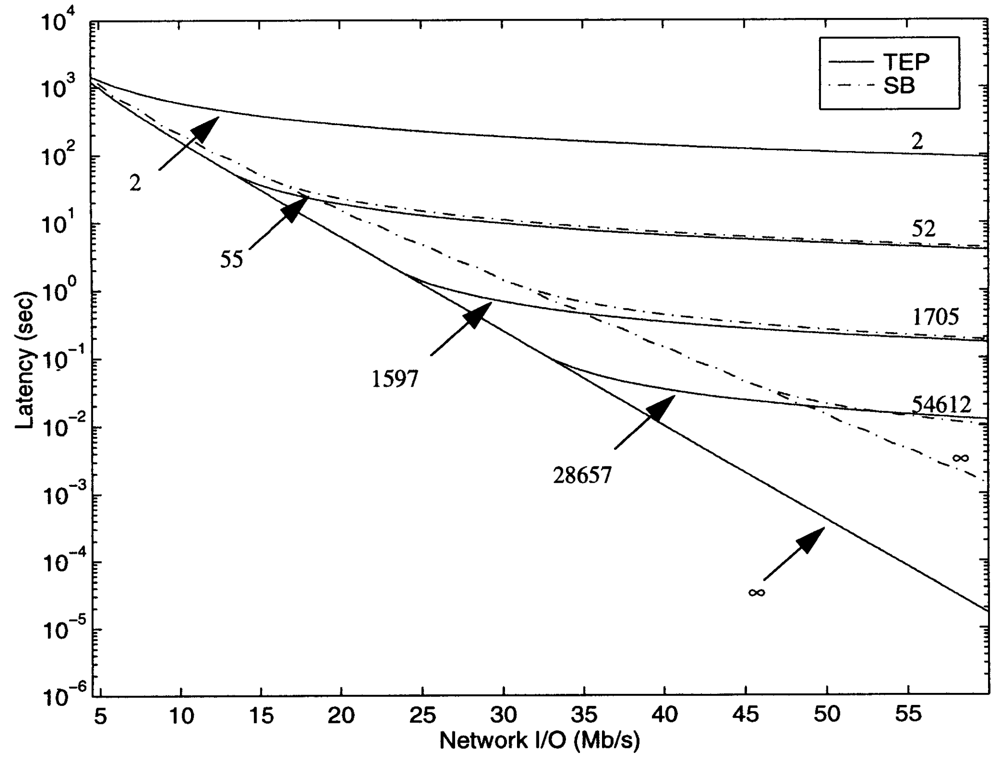
$$\text{Disk Bandwidth} = 2b \text{ or } 3b \text{ Mbit/sec}$$

Eqn 6-18

$$\text{Storage Requirement} = 60bD_{1,\,TEPbf}W$$

Eqn 6-19

A comparison between the SB and TEP schemes is shown in Figure 6-7. This figure shows the access latency and storage requirements for various values of W for the delivery of a two-hour movie encoded at 1.5 Mbit/sec. In keeping with the results reported in [34], the W values plotted for the SB scheme (shown as the dot-dash lines) were chosen (2, 52, 1705, 54612) from the 2nd,10th,20th, and 30th elements from the sequence defined by Eqn 6-11. As TEP's partitions grow more quickly than SB's, the W values used to plot its performance were chosen so that the resulting latencies were as close as possible to those used for the SB plots in [34]. This resulted in W being set to 2, 55, 1597, and 28657, which corresponds to the 2nd, 9th, 16th, and 22rd elements in the sequence defined by Eqn 6-15. Examination of Figure 6-7 reveals that the access latencies for TEP are indeed lower than those for SB, however it also becomes apparent that if one is not careful in choosing the W value, TEP results in higher storage requirements. This effect—caused by TEP's more aggressive design—favors preloading of data into the cache as early as possible, and



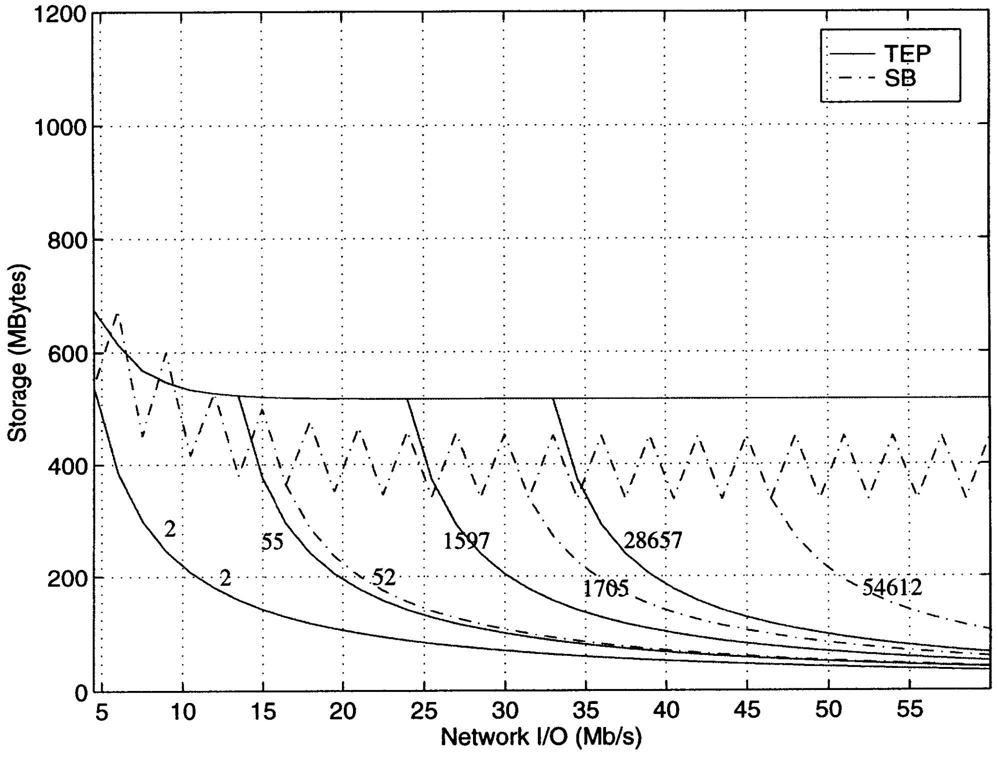


Figure 6-7 Skyscraper Broadcasting (SB) vs Truncated Exponential Partitioning (TEP) for a 2hr movie encoded at 1.5Mbit/sec

can be avoided by limiting W to an appropriately small value dependent upon available bandwidth.

TEP (General Form)

The general form of TEP results from taking the aggressive measure of running partitions at faster than real-time rates to grow segment size at an even greater rate. This measure corresponds to moving from $\frac{B}{K} = b$ to one where $\frac{B}{K} > b$. To see the consequences of this refinement, consider the case where data is downloaded at a rate 25% higher than that needed for real-time playback, as shown in Figure 6-8.

Starting once again with a partition that loops every minute, one now finds that the segment carries an additional 15 seconds for a total length of 75 seconds or 1.25 minutes. The reception of this extra 15 seconds, or 25% of additional data, means that the second partition (P1) can be 15 seconds longer. Consequently, the third partition (P2) can start 15 seconds later than before. The effect of longer partitions and later starting times is compounded with each successive partition and, for a given access latency, results in fewer partitions that are longer in length.

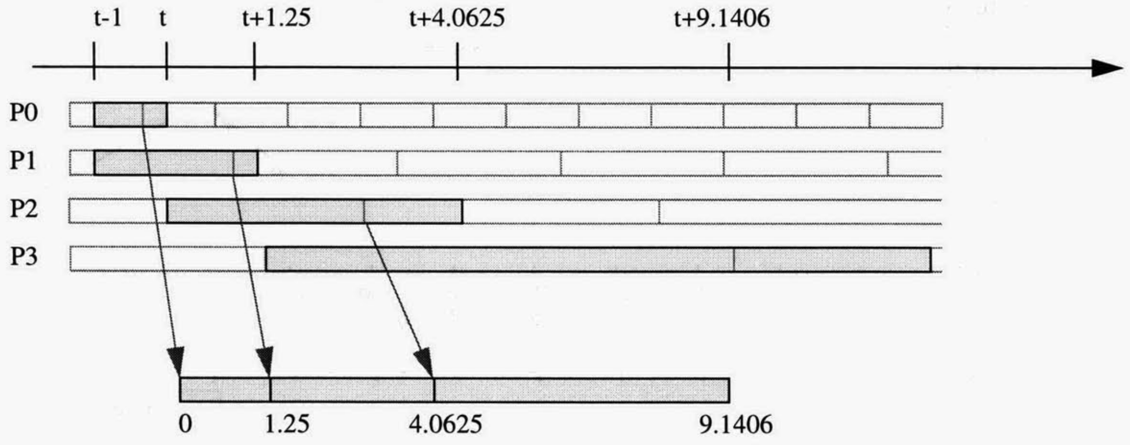


Figure 6-8 Truncated Exponential Partitioning example with Faster than Real Time Partitions

In the general case where each partition n carries data at a rate $r(n)$ multiples of the normal playback rate b, the relative and actual partition lengths are:

$$L_{TEP_{gf}}(n) = min(f_{TEP_{gf}}(n), W)$$

where

$$L_{TEP_{gf}}(n) = \begin{cases} 1r(1) & n = 1 \\ (1 + f_{TEP_{gf}}(1))r(2) & n = 2 \\ (f_{TEP_{gf}}(n-1) + f_{TEP_{gf}}(n-2))r(n)) & n \geq 3 \end{cases}$$

Eqn 6-20

and

$$D_{n,\,TEP_{gf}} = \frac{L_{TEP_{gf},\,rel}(n)}{\sum_{1}^{p} L_{TEP_{gf},\,rel}(n)} D$$

Eqn 6-21

respectively. In addition, the I/O bandwidth at the receiver now becomes

$$(1 + 2r)b$$

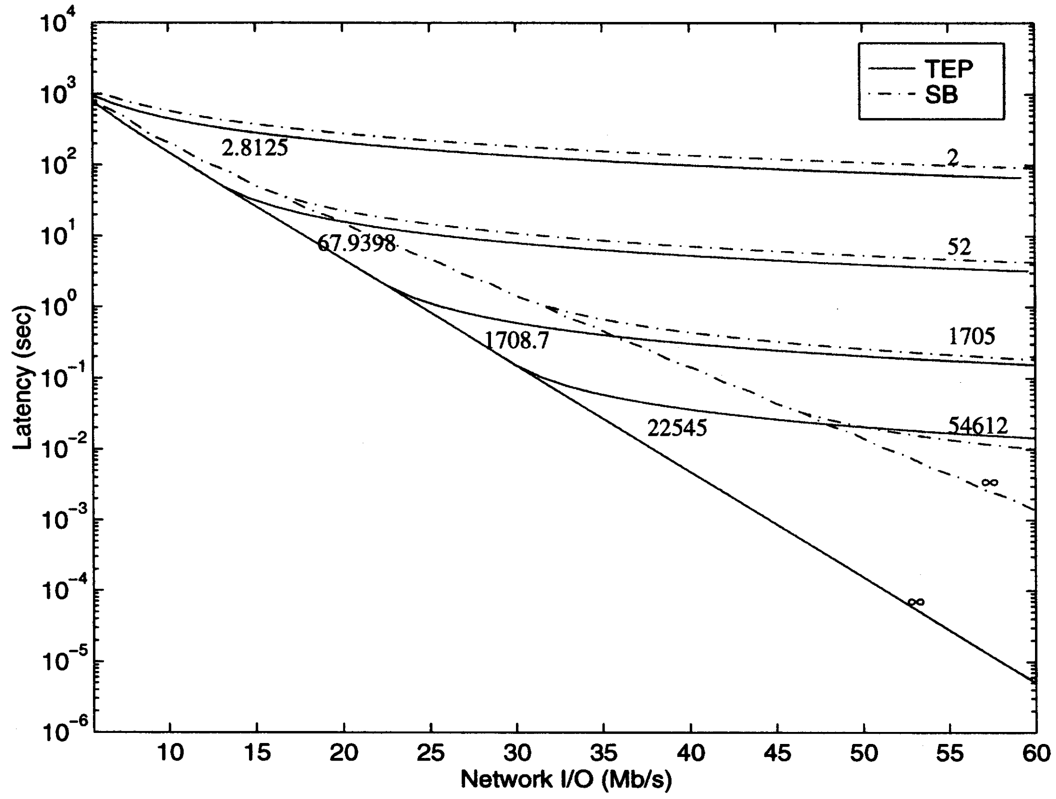
Eqn 6-22

Once truncation starts, better performance is achieved when W is set to a value of $D_{n, TEP_{gf}}$ and $r(n) = 1$. Failure to do so results in violation of design rule three, and therefore results in either the receiver downloading segments prematurely or entering a cycle of downloading one partition and then waiting for the next to appear. To prevent this from happening, the partition allocation program used to generate the results that follow took in a new parameter, K', (in lieu of W) that indicates the number of partitions for which $r(n)$ could be greater than one before truncation occurred. $r(n)$ was then set to one for partitions with $n > K'$.

Partitioning Scheme	I/O Bandwidth (Mbit/sec) β	Access Latency (minute) L	Storage (Mbit) S
SB	$2b$ or $3b$	$\frac{D}{\sum_{i=1}^{K} min(f(i), W)} = D_1$	$60.bD_1(W-1)$
TEP (Simple Form)	$2b$ or $3b$	$\frac{D}{\sum_{i=1}^{K} min(f_{TEP}(i), W)} = D_{1,TEP}$	$60.bD_{1,TEP}W$
TEP (General Form)	$2b$ or $(1+2r)b$	$\frac{D}{\sum_{i=1}^{K} min(f_{TEPgf}(i), W)} = D_{1,TEPgf}$	$60.bD_{1,TEPgf}W$

Table 6-2 Comparison between Skyscraper Broadcasting and Truncated Exponential Broadcasting

Figure 6-9 shows how Figure 6-7 changes for values of $r(n)$ set to 1.25. When $r(n)$ is increased to 1.25, the latency drops markedly since segments grow more quickly. However, the worst-case storage requirements are increased for poorly chosen combinations of network I/O and W; this is a side-effect of the increased segment length growth rate. The increased rate results in fewer partitions required in order to reach a given latency at the expense of having a larger last partition, and hence storage requirement in cases where segment truncation is precluded.



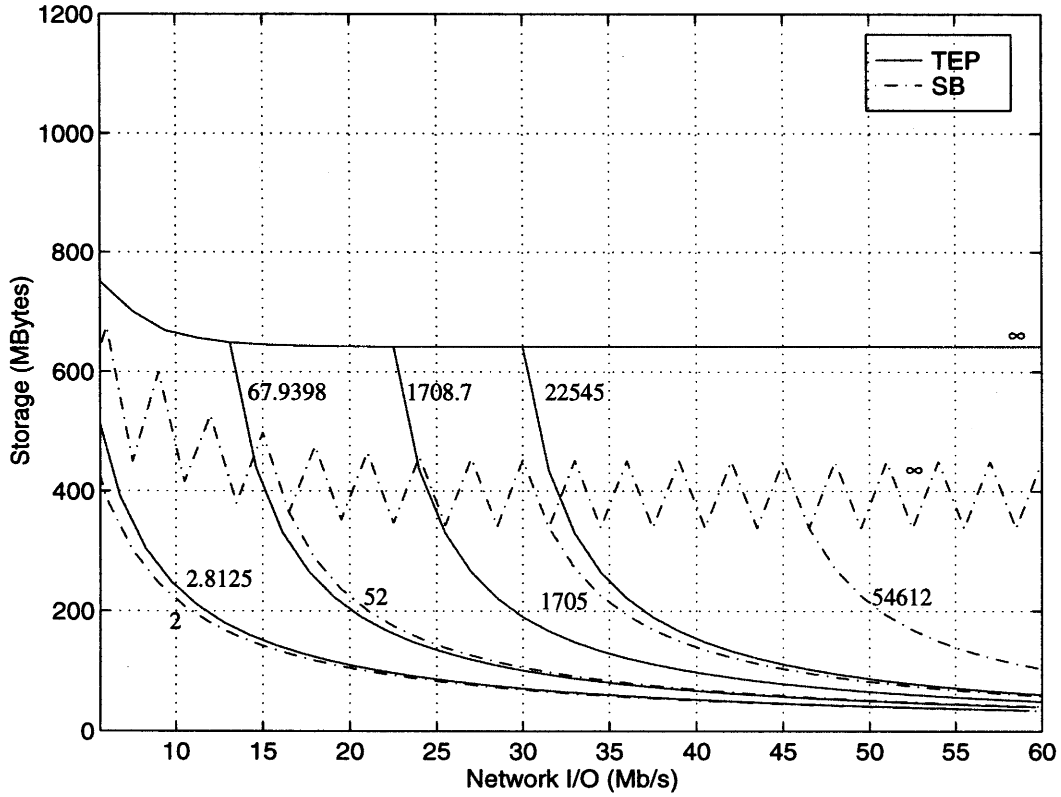


Figure 6-9 SB vs TEXP for a 2hr movie encoded at 1.5Mbit/sec with TEP partitions at r(n) = 1.25

6.1.4 Digital Cable TV Channel Replacement

At this stage, it is worth hypothesizing as to what performance metrics could be achieved were a HPRTP scheme using Truncated Exponential Partitioning deployed as part of a digital cable TV network. At the time of writing, cable head-ends using 256 QAM modulation are capable of transmitting at 42.88 Mbit/sec into each 6MHz analog bearer. Once over head and error correction is allowed for, roughly 37Mbit/sec is usable for data traffic to the home[1]. Given a typical spectrum of 600MHz for downstream traffic, one could allocate roughly 50 of the 100 available channels to VoD with the remaining channels used for live or pure broadcast material. This conversion of the 50 channels corresponds to the creation of 1850Mbit/sec of bandwidth for digital traffic.

For the purposes of this hypothetical system, movies are assumed to be encoded at a constant bit rate of 4Mbit/sec. Some may argue that this figure is unrealistic as it is low and movie bitstream bandwidth varies greatly with content. Typically, movie bit rates vary between 3 to 8 Mbit/sec and tend to be very bursty. Sequences containing excessive motion or high numbers of cuts result in higher bit rates; however, quiescent scenes result in much lower ones. Some groups, such as Phillips and Imedia, have developed statistical multiplexing methods to take advantage of this observation by filling in the troughs of one digital stream with peaks from another. The following hypothetical system does not assume the existence of such capabilities in the network.

Before applying the TEP scheme to the available 1850Mbit/sec of bandwidth, it is worth noting that SB and TEP share a common weakness: To achieve low access latencies, the first segment must be made very short when compared to the sum of all segment lengths. This mandates the creation of many segments and hence partitions to carry them. Consequently, receivers will move quickly through the initial partitions before achieving greater degrees of overlap with others in the higher-order partitions. If one could find a way to eliminate the shorter, lower-order partitions and start with a larger first partition, the bandwidth could be significantly reduced.

The simplest way to achieve this goal is to use longer partitions in the first place, and to have the receivers store the first partitions from many popular movies in anticipation of their being needed. Receivers could then play movies back with no latency while loading in higher-order segments as needed. Should a movie's initial partition not be present in the disk cache, the viewer could wait for the first partition to load.

Given current rates of decline in disk drive prices [45] and increases in capacity [24, 45], it is not too outrageous to envision a super set-top box with a 10 Gbyte storage capacity. If 1.5 Gbytes of this drive were reserved for the higher order segments, that leaves 8.5 Gbytes for the storage of the movies' first few segments. Assuming that movies are coded at an average rate of 4.0 Mbits/sec and the first 120 seconds of each movie is cached, one could store enough material to give on-demand access to approximately 148 movie titles.

Given that there is 1850 Mbit/sec of available bandwidth for delivering movies two hours in length encoded at 4Mbit/sec with the initial partition set to one minute, one can now determine how many movies could be delivered through a simple iterative process of tuning r, K, W. Experience has shown the best performance occurs when the choice of W matches exactly with a term in $F_{TEP,frt}(n)$. Using this technique, the following TEP case was derived for $r = 1.1955$, $K = 8$, and $K' = 6$.

1. Thanks to Robert Wallace of Motorola for furnishing these performance figures.

Partition	Start Time (min)	Stop Time (min)	Partition Length (min)	Segment Length (min)	End of Segment (min)	Cumulative Bandwidth(Mbit/sec)
1	-1	0	1.0	1.1955	1.1955	4.7819
2	-1	1.1955	2.1955	2.6247	3.8202	9.5639
3	0	3.8202	3.8202	4.5669	8.3871	14.3458
4	1.1955	8.3871	7.1916	8.5975	16.9846	19.1278
5	3.8202	16.9846	13.1644	15.7379	32.7224	23.9097
6	8.3871	32.7224	24.3353	29.0925	61.8150	28.6917
7	32.7224	61.8150	29.0925	29.0925	90.9075	32.6917
8	61.8150	90.9075	29.0925	29.0925	120.0000	36.6917

Table 6-3 Example TEP scheme for 2hr Movie encoded at 4Mbit/sec

The access latency in this case is exactly one minute. The maximum amount of storage required is 873 Mbytes. The first six partitions each use a total of 1.1955x4 = 4.7817 Mbit/sec, while the final two each use a total of 1x4 = 4 Mbit/sec. The total bandwidth required by each movie is 36.6917 Mbit/sec, which allows 50 movies to be delivered on demand using the 1850 Mbit/sec available. The remaining 15.425Mbit/sec (or 0.3085Mbit/sec per channel) could be used for other purposes, such as billing.

Were one to cache the data loaded during the initial minute, one could provide instantaneous access to the movies. For the table above, the first 2x1.1955x60 = 143.46 seconds of each movie would be stored ahead of time. The amount of storage required in this case is about 68.41Mbytes per movie, which corresponds to a total of 3.42GBytes for a 50 movie delivery system—a figure well below the 8.5 Gbytes of available disk space. In other words, every movie in the hypothetical system just described is available for viewing at any time with no waiting whatsoever.

When one compares the performance in the example above to that provided by the standard Near Video On Demand (NVoD) methods employed on today's analog cable networks, one finds a significant improvement. If one were to use the NVoD technique of simply staggering the start times, a system which gave 2hr/(37Mbit/sec / 4Mbit/sec) ~= 12.9 minute latencies would result. Thus, even if the first few minutes of data were not pre-cached, the application of TEP in this case results in a roughly 13-fold performance increase at the modest cost of locating a disk at the receiver.

Further gains become apparent when one remembers that the system just described operates in a broadcast manner. Every partition of every movie is broadcast into every home whether someone is watching a particular movie partition or not. Thus, the next logical means for improving performance is to move from the broadcast model to a multicast one. This would give neighborhood distribution nodes the ability to block the delivery of unwanted partitions and replace them with partitions from additional desired movie titles.

An example of how such a system might look is shown in Figure 6-10. In this example, a TEP server farm broadcasts every partition needed for several hundred movies to a local high-speed router connected to a regional backbone. The regional backbone feeds into a number of exchanges dispersed around the city. Each exchange then feeds into a number of neighborhood distribution nodes, which finally connect to 500 to 1000 homes. At each junction between the server farm and the receivers, partitions can be pruned from the delivered data, thereby allowing a greater number of titles to be offered.

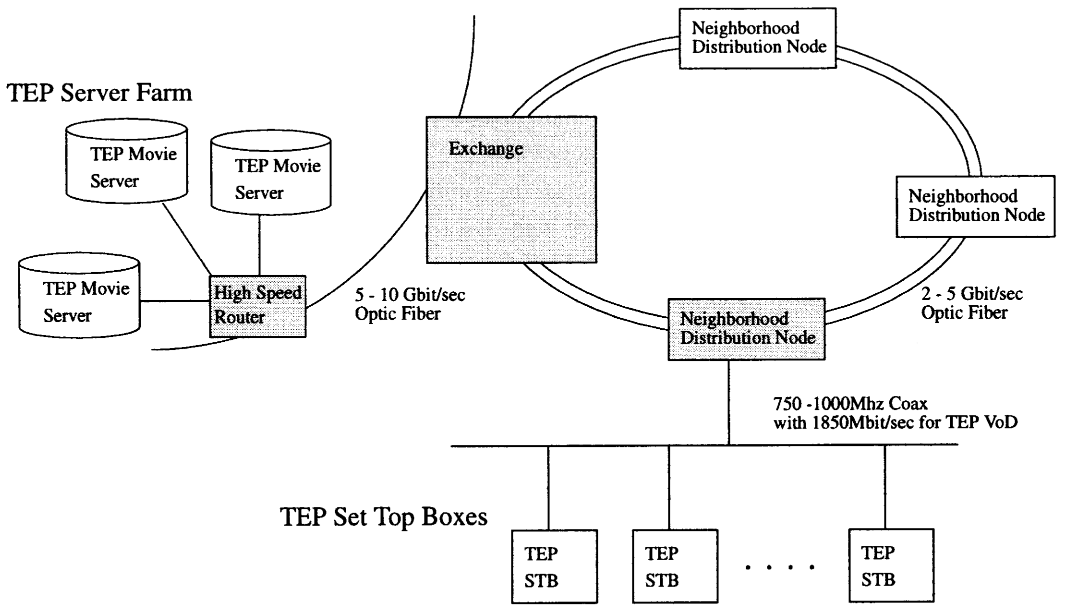

TEP Server Farm
TEP Movie Server
TEP Movie Server
TEP Movie Server
High Speed Router
5 - 10 Gbit/sec Optic Fiber
Exchange
Neighborhood Distribution Node
Neighborhood Distribution Node
Neighborhood Distribution Node
2 - 5 Gbit/sec Optic Fiber
750 -1000Mhz Coax with 1850Mbit/sec for TEP VoD
TEP Set Top Boxes
TEP STB
TEP STB
. . . .
TEP STB

Figure 6-10 Hypothetical Metropolitan VoD system based on TEP

6.2 User Centered Schemes

In 6.1, it was shown that it is possible to employ completely open-loop data-centered techniques for the delivery of linearly viewed objects such as movies. The best of these techniques is Truncated Exponential Partitioning (TEP), which carefully distributes the object's data over several partitions that the receivers tune to in successive order. Both TEP and its predecessor Skyscraper Broadcasting (SB) build upon the assumptions that there are many simultaneous viewers for an object, that the object is of considerable length, and that its display starts from the beginning and progresses in a linear fashion to the end.

Unfortunately, these assumptions do not always hold. Not everybody wants to watch the most popular movies; some people may want to watch classics released decades ago. Also consider the selective manner in which people consume news and sports commentary. Not everybody wants to hear about how the Australian stock market's All Ordinary Index performed yesterday, nor do they want to see how the Boston Red Sox fared in their latest home game. In these cases—and also especially in the case of Olympic coverage—people tend to consume the presented media objects in smaller pieces, rather than as one long continuous stream.

Finally, the first few looping partitions created by TEP and SB tend to be very short in length and will therefore have fewer simultaneous listeners. In fact, it is possible to imagine a situation where these smallest partitions become so small that no overlap exists between viewers tuning into them, thereby indicating that the data they transport is consuming valuable bandwidth when nobody needs them.

For all these reasons, one needs to also consider not only data-centered schemes, but user-centered schemes as well that enable the receiver to request information immediately instead of waiting for it.

6.2.1 On-Demand Partitioning

On-demand partitioning is based upon the premise that receivers should have complete control over the delivery of data from the sender. It therefore offers maximal flexibility since receivers can perform random accesses at will. This ability translates into a small lower bound on the receiver's cache since it need only be large enough to compensate for network jitter. However, this freedom comes at a cost: the number of receivers listening to a given partition will be low and hence the number of partitions will tend to be high. The reason for this is that requests from receivers will tend to be uncorrelated, hence the probability that a receiver will subscribe to another's partition is small. This fact limits overall scalability since both network and sender resources will tend to increase linearly with the number of receivers.

One means for increasing the scalability of on-demand partitioning is to employ caching servers at the edges of the network. These caching servers can act as proxy servers in the same manner shown earlier in Figure 6-2. Another means for increasing scalability is to employ the same localized recovery techniques used by SRM for the recovery of individual packets at the macroscopic level of recovering whole sections of an object. Unfortunately, while this shifts the majority of the load from the originating or caching servers to receivers that have already accessed the object, it does not decrease the number of partitions and likewise fails to lessen the load on the network. Thus, while on-demand access provides maximal flexibility at the cost of limiting scalability, it should be used sparingly.

A network simulator was built to ascertain how much bandwidth on-demand partitioning uses. The simulator differed from normal network simulators in that it did not model network traffic in terms of packets that traverse a mesh of nodes connected by links. Instead, it modeled network traffic in terms of streams that flood regions. The regions were arranged hierarchically so that they could pass or block streams from larger parent regions or from any smaller ones they may contain. This arrangement roughly mimicked the filtering that would occur between administratively scoped multicast regions were HPRTP used as the transport protocol.

Test Topologies

The base topology from which the test topologies were derived was a hypothetical city-wide cable television plant that contained a head-end fed from a national distribution system. It was assumed that the head-end contained sufficient storage to cache the entire content of the national distribution feed. Larger schemes involving the distribution of content to the city head-end were not considered, since they would have no bearing on the city head-end's ability to deliver content to subscribers' homes.

The distribution network consisted of a four-level hierarchy. The city head-end sat at the top of the hierarchy and was linked to a primary distribution node in each of the six districts. The primary distribution nodes of each district were linked to a small number of suburban distribution nodes, which in turn were linked to block distribution nodes that finally fed into subscriber set-top boxes. The basic hierarchy structure is shown in Figure 6-11.

To remove symmetry effects, the branching factor at the district, suburb, and block distribution nodes was varied, using a gaussian distribution. Four topologies were simulated in all. The upper and lower bounds of the branching factors used are shown in Table 6-4. The gaussian function employed at each level used a mean taken from the average of the upper and lower bound, and a standard deviation set to one quarter of their difference. The resulting topologies served 74, 446, 2963, and 16154 receivers, respectively.

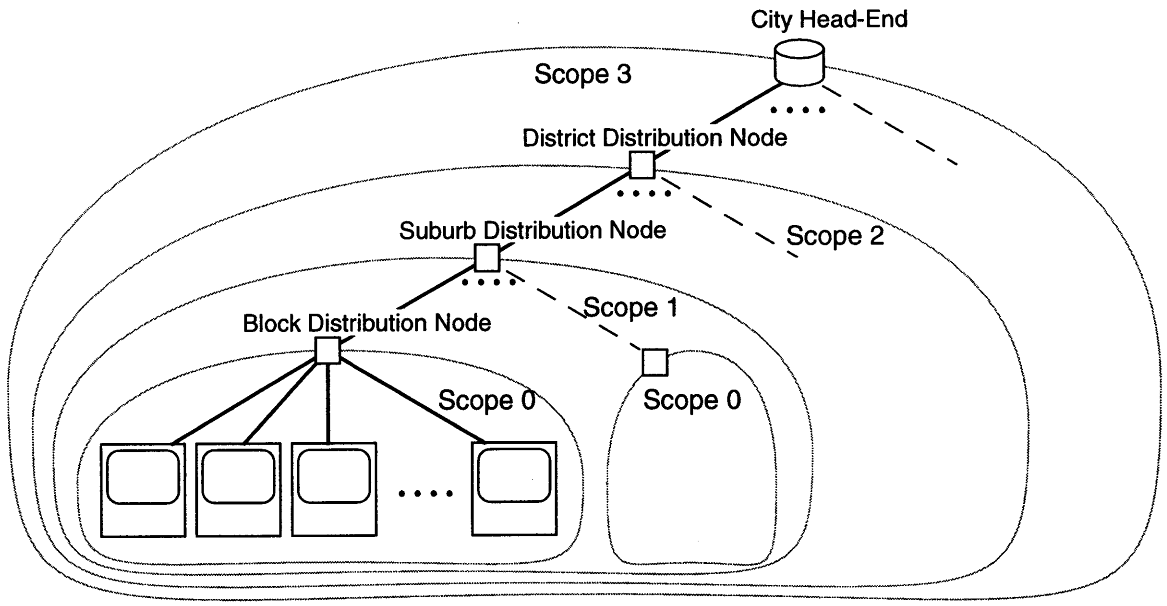


Figure 6-11 Basic Structure of the Hypothetical Cable Television System

Topology	District Branch Factor [min, max]	Suburb Branch Factor [min, max]	Block Branch Factor [min, max]	Total Number of receivers
1	[2,4]	[3,5]	1	74
2	[3,10]	[4,7]	2	446
3	[4,30]	[5,10]	[2,6]	2,963
4	[4,30]	[5,10]	[10,30]	16,154

Table 6-4 User Centered Simulation Topology Characteristics

Simulations

In each simulation of the four topologies, a single two-hour movie was delivered using the dynamic partition creation scheme described in 5.4. Receiver start-times were chosen from a uniform random distribution over an eight-hour period. Only the head end and receivers contained disk caches; the district, suburb, and block distribution nodes only forwarded and blocked streams as appropriate. Simulations were run in each topology with the receiver's maximum disk cache capacity set to 15, 20, 40, 50, 75, and 100% of the movie. At the start of each simulation run, receiver caches were emptied and only the head end contained the movie. The results from these simulations are shown in Figures 6-12 through 6-20.

Figures 6-12 through 6-16 show detailed results for the simulation run made using topology 4 with the maximum receiver disk cache size set to 40% or 48 minutes of footage. Figure 6-12 shows the number of receivers actively displaying the movie at any given time. Note that at the two-hour mark the first receivers finish and from then on the number of new receivers starting to view is roughly equal to the number that are finishing. Figure 6-13 shows the average number of streams carried in a region of scope N, where scope 3 is the largest scope and scope 0 is the smallest.

One observes that traffic within scope 3 serves at most three partitions and ceases to transmit any traffic at all shortly after the two-hour mark. At this stage, the movie's contents are distributed amongst the receivers, hence they can supply missing portions of the movie to one another without involving the source. Since the start times of each receiver are randomly chosen, the first receivers within a block must increase the scope of their search for missing data to include receivers in other blocks, suburbs, and districts. As time passes, more receivers tune in and the distance subsequent receivers must search to find the data they need decreases.

This hypothesis is verified by Figure 6-13, which shows that traffic in scope 1, the district-level scope, peaks at around the two hour mark before falling. Similarly the traffic for scope 2, the suburb-level scope, peaks slightly later and drops at a slower rate. However, the traffic from scope 3, the block-level scope, increases steadily to an average of around 4.6 partitions per region. Given that there are, on average, twenty receivers per block for topology 4, and that at any given time an average of five will be viewing a movie, this figure shows that the bulk of missing data is being supplied to receivers within the same block. This conclusion is further supported by Figures 6-13 through 6-16, which show the average bandwidths within each region, flowing downwards into smaller scoped regions, normalized according to number of active regions at a scope and the number of receivers served by an individual stream.

Figures 6-17 through 6-20 show the number of streams per region at each scope in the hierarchy averaged between the third and eighth hour. One notes that as the amount of storage available in each receiver increases, the number of higher-scoped streams decreases. In other words, receivers need not search as far to find a peer whose cache can supply the missing data they need. In addition, Figures 6-17 through 6-20—and in particular Figure 6-20—shows that increases in receiver density also decrease the need for higher scoped partitions. This effect results from an increased chance that a receiver within the smallest-scope region will have the desired data.

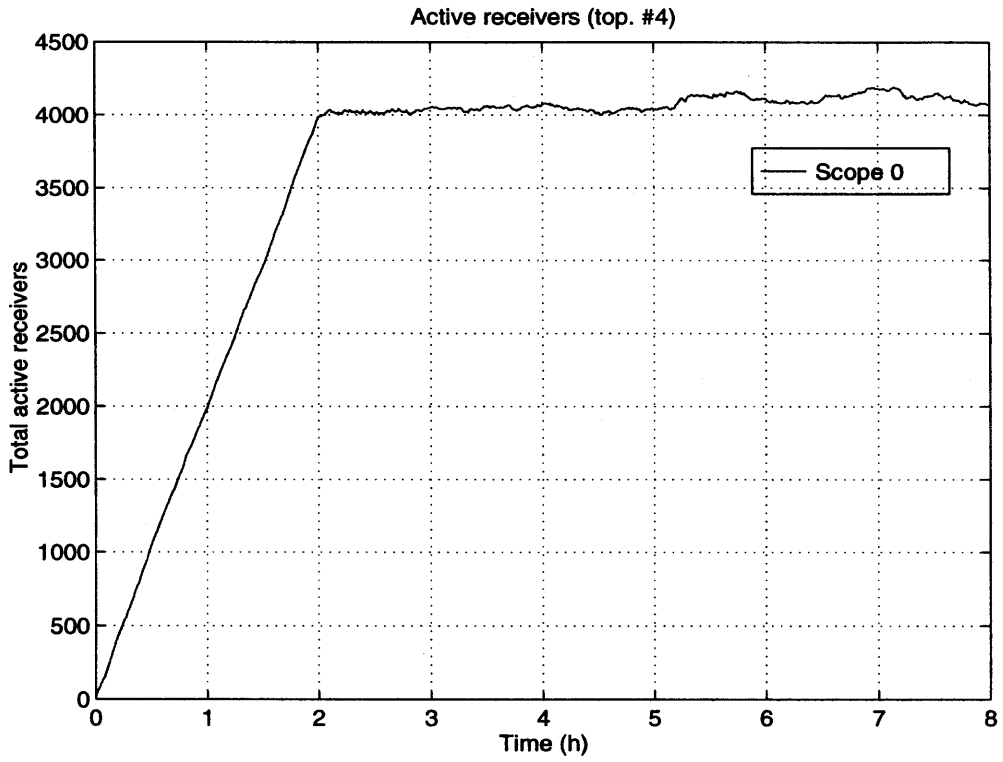


Figure 6-12 Number of Active Receivers for topology 4, cache size = 40%

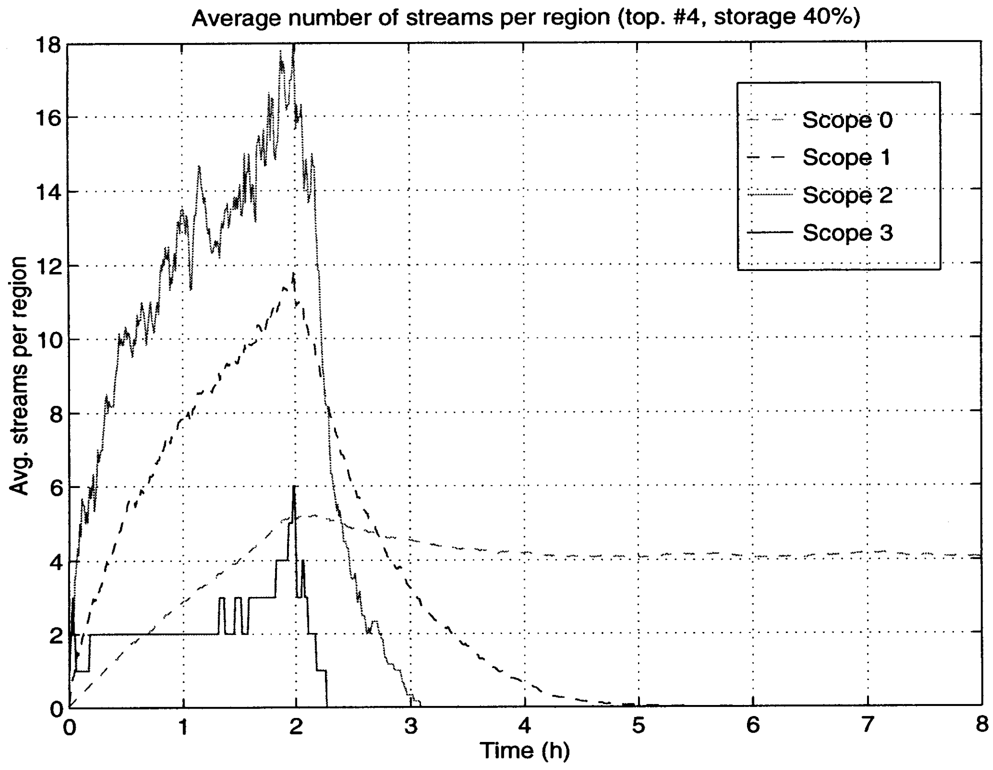


Figure 6-13 Average number of streams per region for topology 4, cache size = 40%

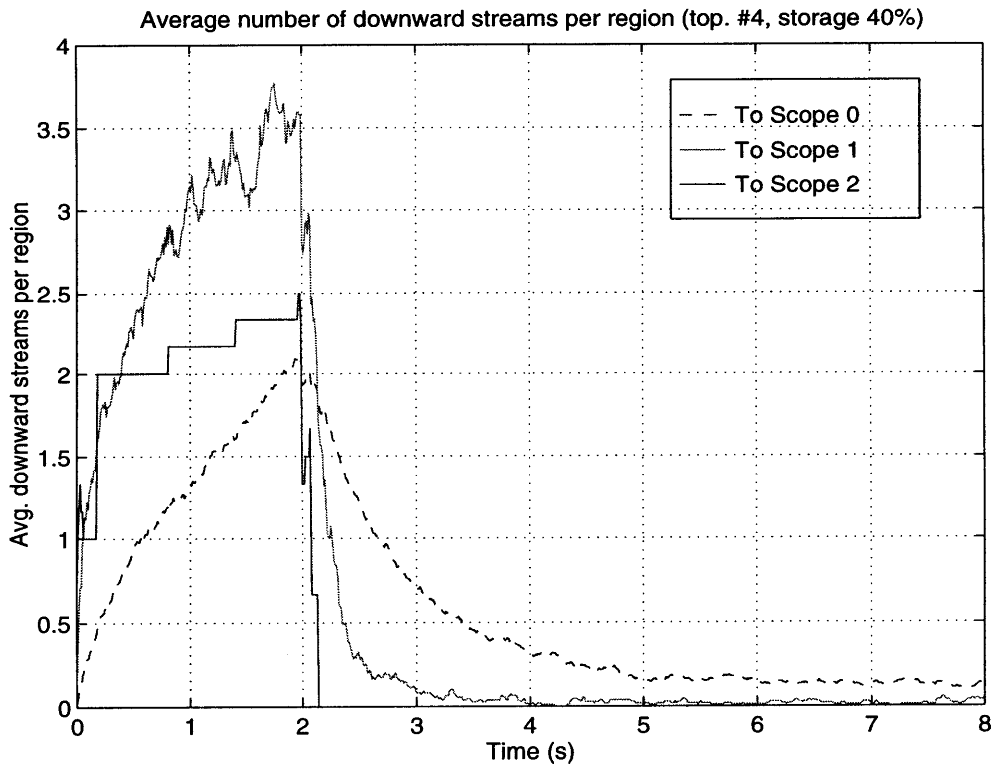


Figure 6-14 Average number of downward streams for topology 4, cache size = 40%

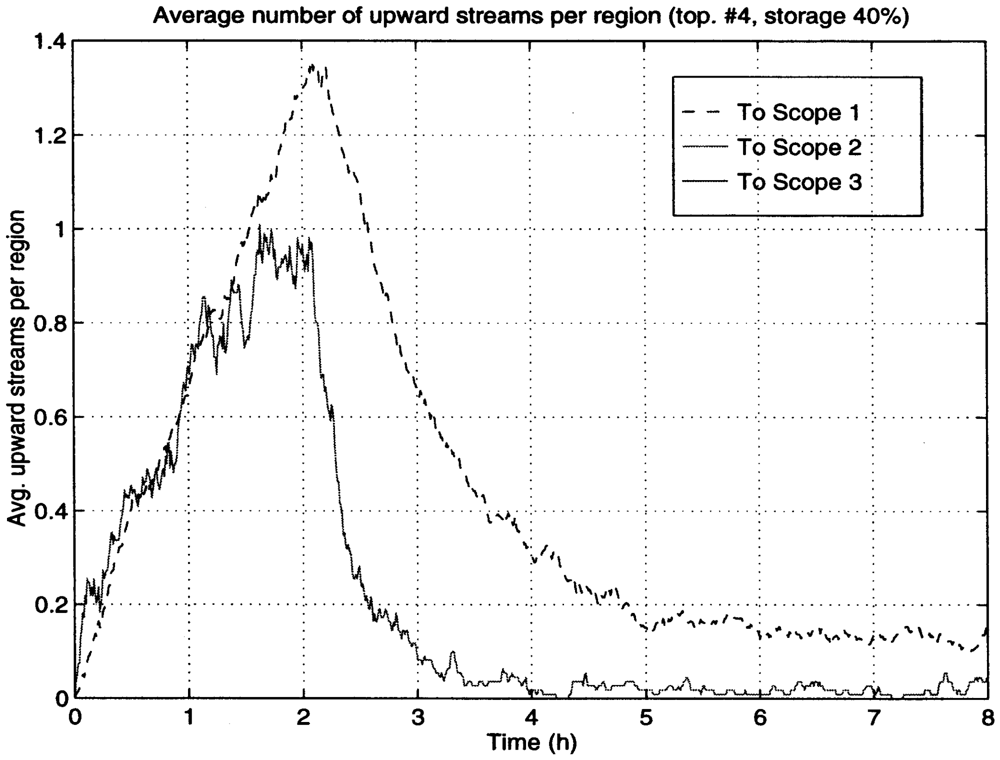


Figure 6-15 Average number of upward streams for topology 4, cache size = 40%

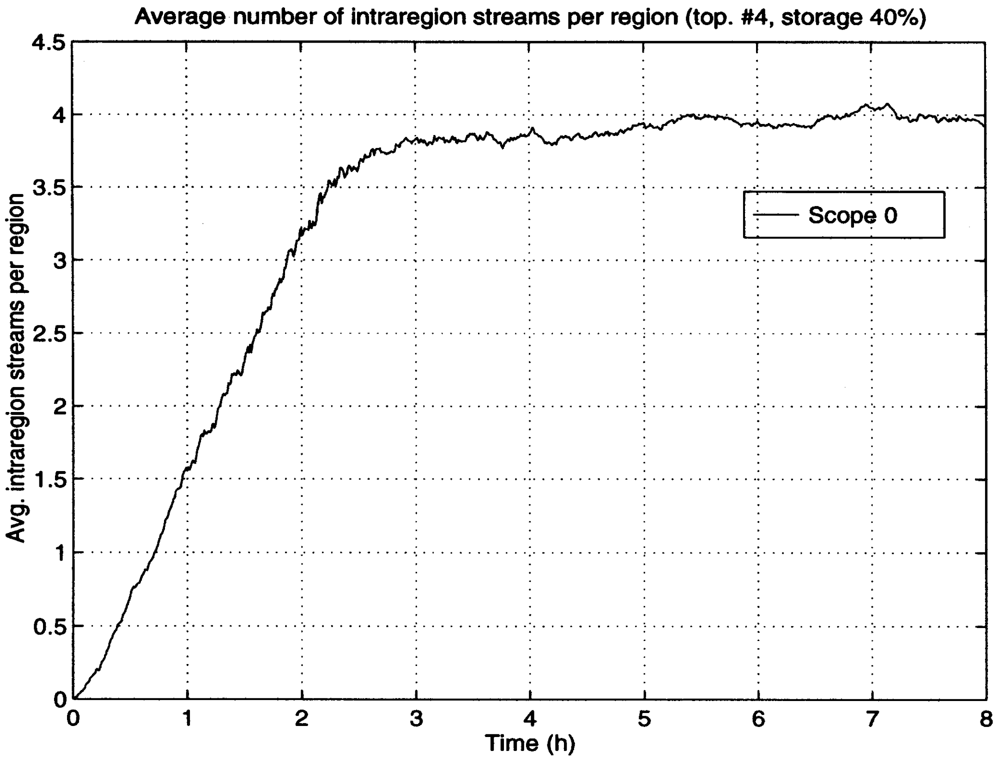


Figure 6-16 Average number of intra region streams for topology 4, cache size = 40%

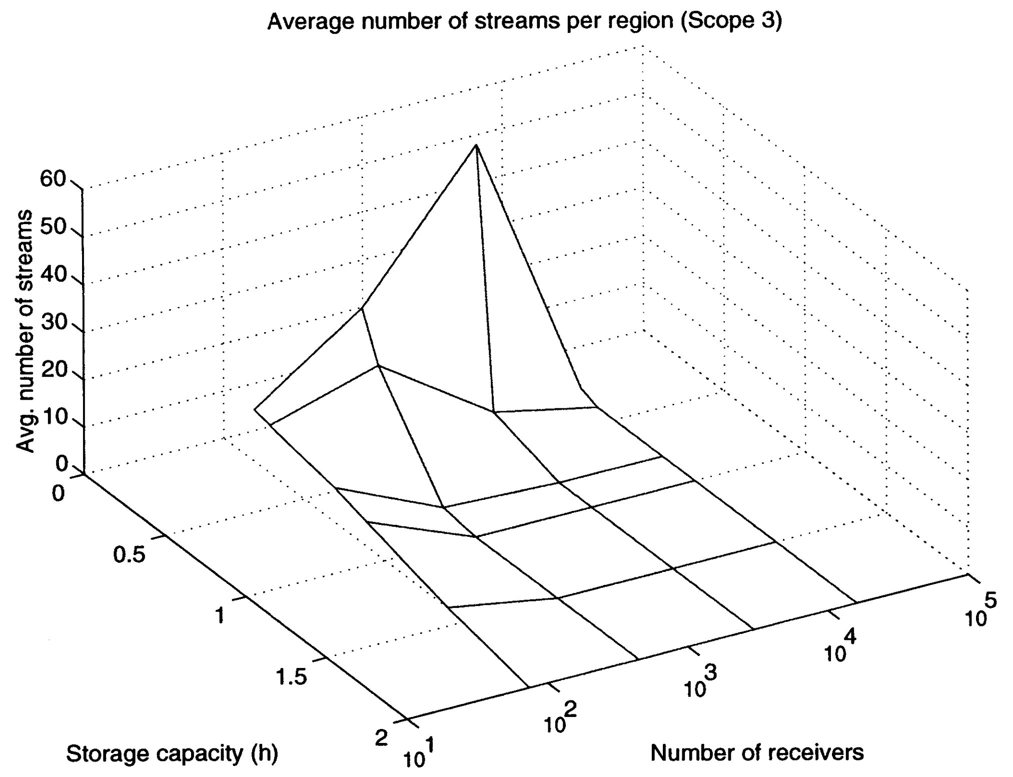


Figure 6-17 Average number of streams per region with scope 3

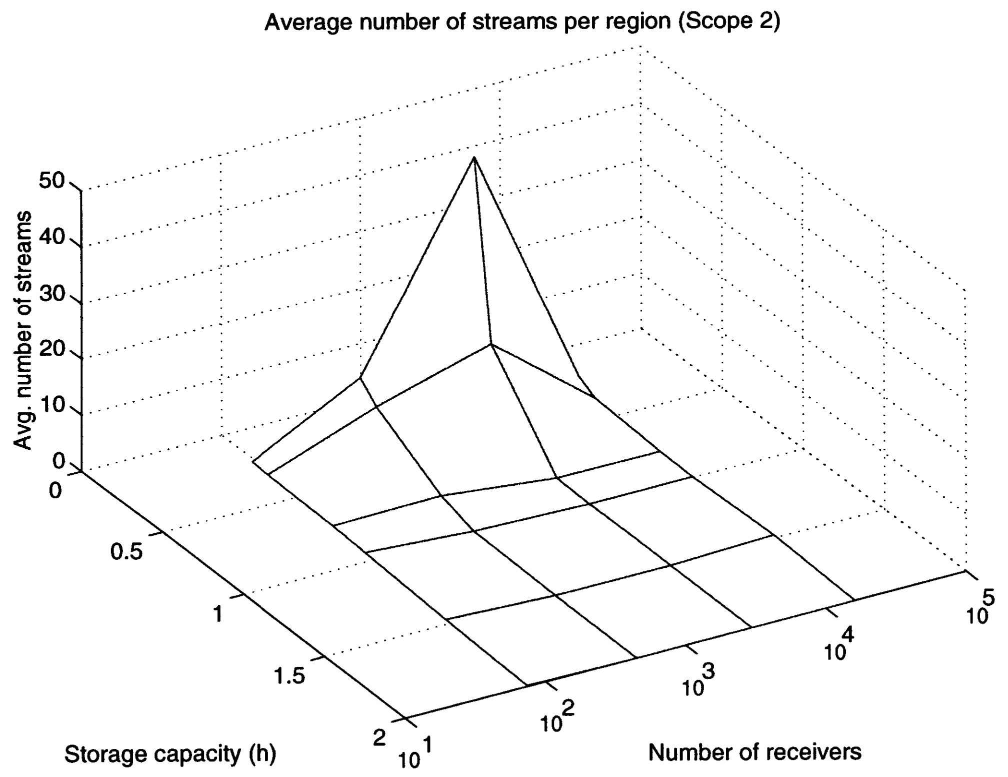


Figure 6-18 Average number of streams per region with scope 2

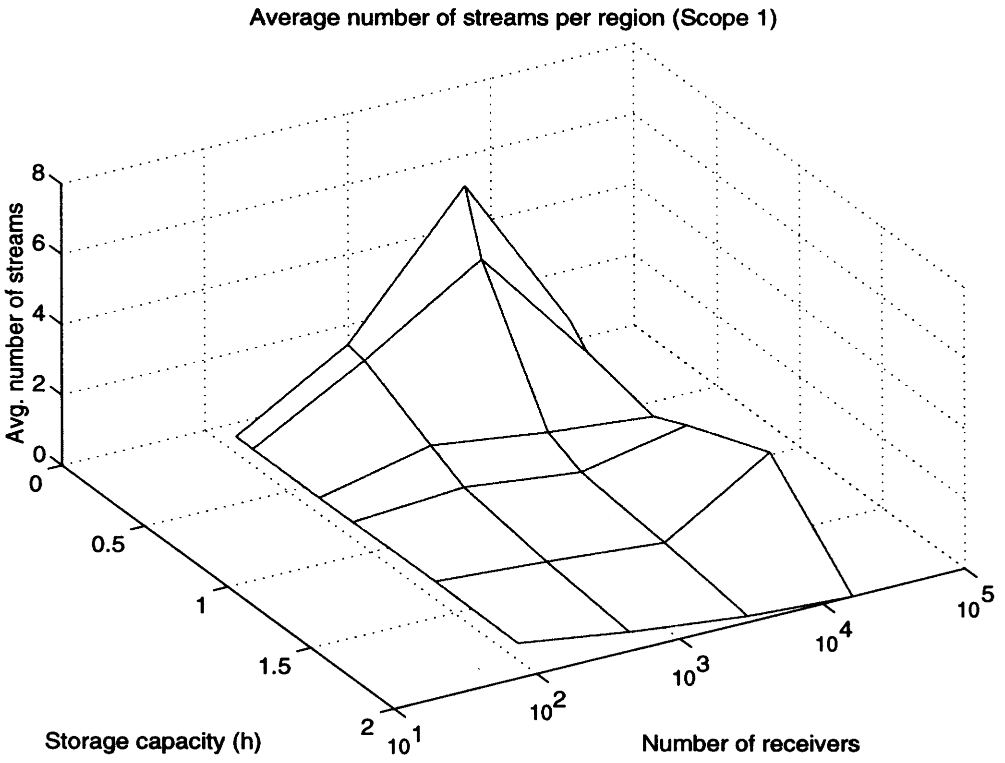


Figure 6-19 Average number of streams per region with scope 1

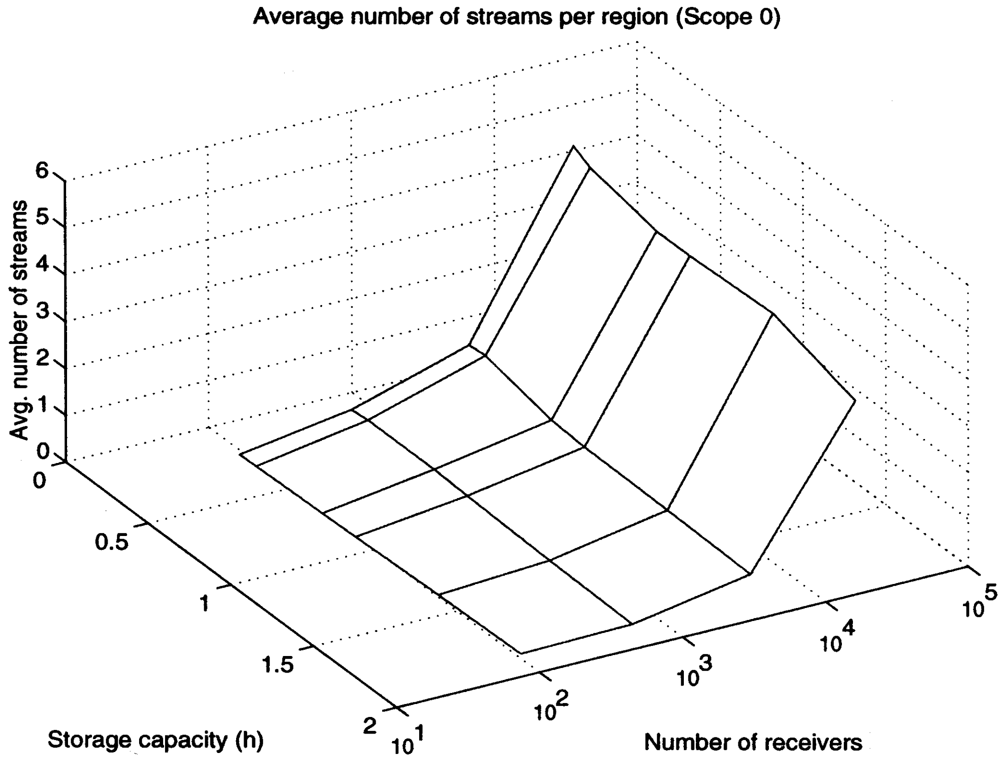


Figure 6-20 Average number of streams in the top region with scope 0

Chapter 7

Test Applications

The fact that the mechanisms developed within this thesis are designed for multimedia delivery on extremely large scales makes them difficult to test over real networks without enlisting the help of thousands of geographically dispersed testers. Furthermore, HPRTP assumes the widespread deployment of administrative scoping. At the time of this writing, native IP multicast routers are being deployed within the Internet; however, administratively scoped zones are only just beginning to appear. In addition, the current Mbone provides access to at best only a few megabits of bandwidth that must be shared by all applications. To use HPRTP as a general purpose delivery service for multi-megabit media streams in a wide-scale fashion under these conditions would be ill-advised and antisocial.

Rather than build and deploy full scale applications, two example applications were developed and deployed in the controlled environment of the MIT Media Lab ATM [7, 99] Local Area Network. The aim was not to test scalability—since this had been achieved through simulation—but to validate, test, and verify the HPRTP protocol, and the feasibility of supporting local HPRTP caches on commonly available computing hardware. While the platforms upon which the applications were developed (500Mhz DEC Alpha Stations running Digital Unix 4.0, and high-end dual 200MHz Pentium Pro and dual 300MHz Pentium II Windows NT machines) are currently beyond the means of the general public, this will not remain true for long.

7.1 Reflection of Presence

The Reflection of Presence [4] application (ROP) represents a new take on the application of video conferencing and was developed using the ISIS multimedia scripting language [2]. Participants do not view images of others in the conference in separate windows (as is the case for *vic* [52]). Instead, the image from each participant is segmented from the surrounding background and combined with others using a "magic mirror" metaphor. The composited image is then compressed using JPEG [37] and transmitted to each participant. The audio portion of the conference is transmitted in a multipoint-to-multipoint fashion similar to that used by *vat*.

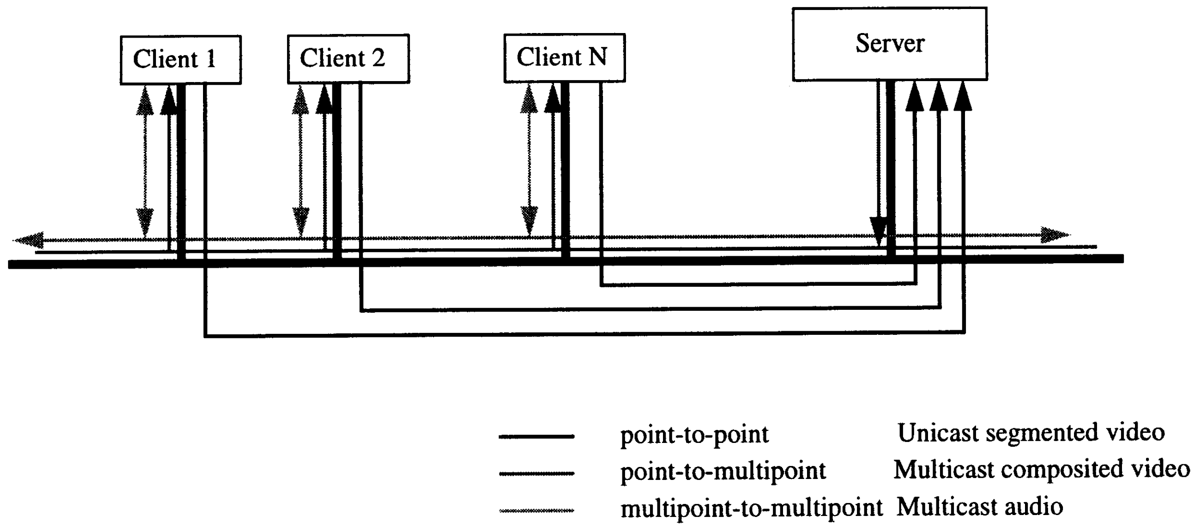


Figure 7-1 Reflection of Presence

Hardware limitations in the current system presently require the segmentation and composition processes to be performed on different machines, thus yielding the architecture shown in Figure 7-1. As the capabilities of the machines within the lab increase, the video distribution model will move away from the reflector topology shown toward one that merges a local compositing server within each client. This refinement will result in a fully decentralized system, more accurately reflect the multipoint-to-multipoint nature of the application, and also match the model used for audio distribution within the application.

As currently implemented, the system uses HPRTP to provide buffered, reliable point-to-multipoint distribution of video and multipoint-to-multipoint distribution of audio. Disk caching is not used. The application uses two streams—one for video, the other for audio. Since the ROP video compositor is computationally bound, it runs at an irregular frame rate. Silence suppression is incorporated within each of the session members to reduce network bandwidth. Both of these characteristics mean that the assumption of continuously sampled media made by HPRTP's Logical Data Stream and Partition descriptions is rendered invalid.

The Reflection of Presence application demonstrated, tested and validated the following ideas and functionality with the reference implementation:

- HPRTP protocol message completeness
- Variable Format ADUs
- Reliable multipoint-to-multipoint distribution
- RAM cache operation

7.1.1 ADU Formats

The Reflection of Presence application used the Application Data Unit (ADU) formats shown in Figure 7-2. Video frames and audio samples are then sent under the restriction that the frame number and audio sample numbers increase monotonically. Since these ADUs do not increase monotonically in time and are independent from one another, they were assigned to different objects. Receivers polled their local HPRTP cache to determine when new data had arrived by performing reads on the last-known ADU. Had a new frame of video or collection of audio samples arrived, it was returned to the ROP application; if not, the reads failed and the ROP application continued.

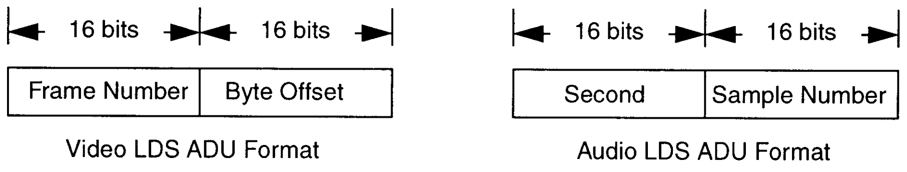


Figure 7-2 Reflection of Presence ADU Formats used by the Video and Audio LDSs

7.1.2 Application Hints

Since hyperlinks were not present, and the previous access-mode made no sense, these application hints were not supplied. The Intra-Object Locality hint was, however, supplied, and was set to Linear locality.

7.1.3 Content Hints

The content hints supplied by the ROP server resulted in minimal functionality. No disk caching was allowed, nor were dynamic partitions. The content hints used for the ROP application were as follows:

Content Hint	Audio LDS	Video LDS
LDS Hint payload type layering index layer drop order	 RTP_PT_L16_2 0 0	 RTP_PT_JPEG 0 0
Reliability/Latency Model	Bounded Effort Lossy Real Time	Bounded Effort Lossy Real Time
Personness	First	Third
Volatility	Live	Live
Persistence	Transient	Transient

Table 7-1 Content Hints for the Reflection of Presence application

7.2 Distributed CD Juke Box

The Distributed CD Jukebox application (DCDJB) was designed to illustrate the concept of fully distributed media delivery system. Built for PCs running Windows NT, the DCDJB leveraged the presence of CD-ROM drives on these machines to provide a distributed music delivery service. Audio information was read directly from CDs and stored by the HPRTP cache on the machines' local hard drive. People running the DCDJB on remote machines could then listen to audio stored not only on the local hard drive of their machine, but others' machines as well. Furthermore, people could tune into streams being fetched by others and if need be schedule dynamic partitions to retrieve sections of music that had been missed. The basic architecture of the DCDJB system is shown in Figure 7-3.

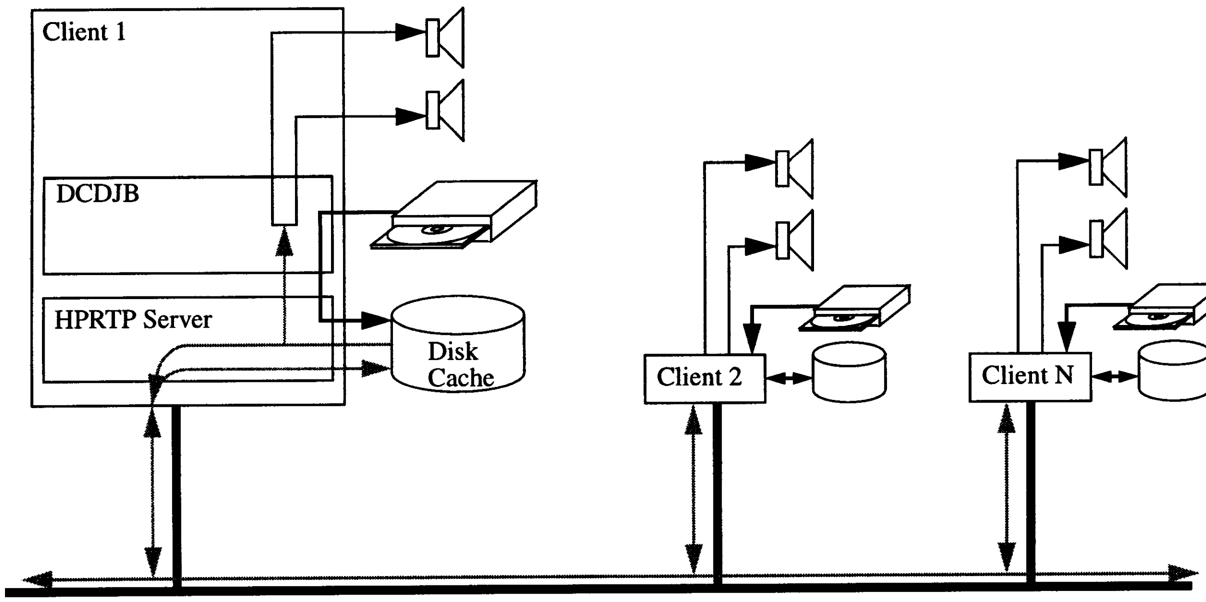


Figure 7-3 Distributed CD Juke Box

An example of the freedom afforded by this scheme is shown in Figure 7-4. Imagine that the source is playing a song five minutes in length and that around the 75 second mark in the song, client one's user decides that she wants to listen to the first two minutes of that song. She informs the local smart network cache that this is the case. It determines that it has none of the song in its cache, and therefore that it must retrieve it from somewhere else.

After consulting its cache of active session announcements, client one's cache determines that the song is currently being transmitted by another cache. It joins the song's session and determines that it has missed the first part of the song. It also computes the current playback position of the current partition and concludes that by subscribing to this partition it can receive the data corresponding to seconds 75 through 120 (two minutes) of the song from that partition. It then issues a request for the first 75 seconds that have been missed.

Upon hearing this request, the source's cache determines that it has the missing data and has the resources to serve it. After a short delay to allow for suppression to occur, the source completes the negotiation with client one and a dynamic partition is created that delivers the missing data from the source to client one. Client one caches the data from both partitions and immediately commences playback.

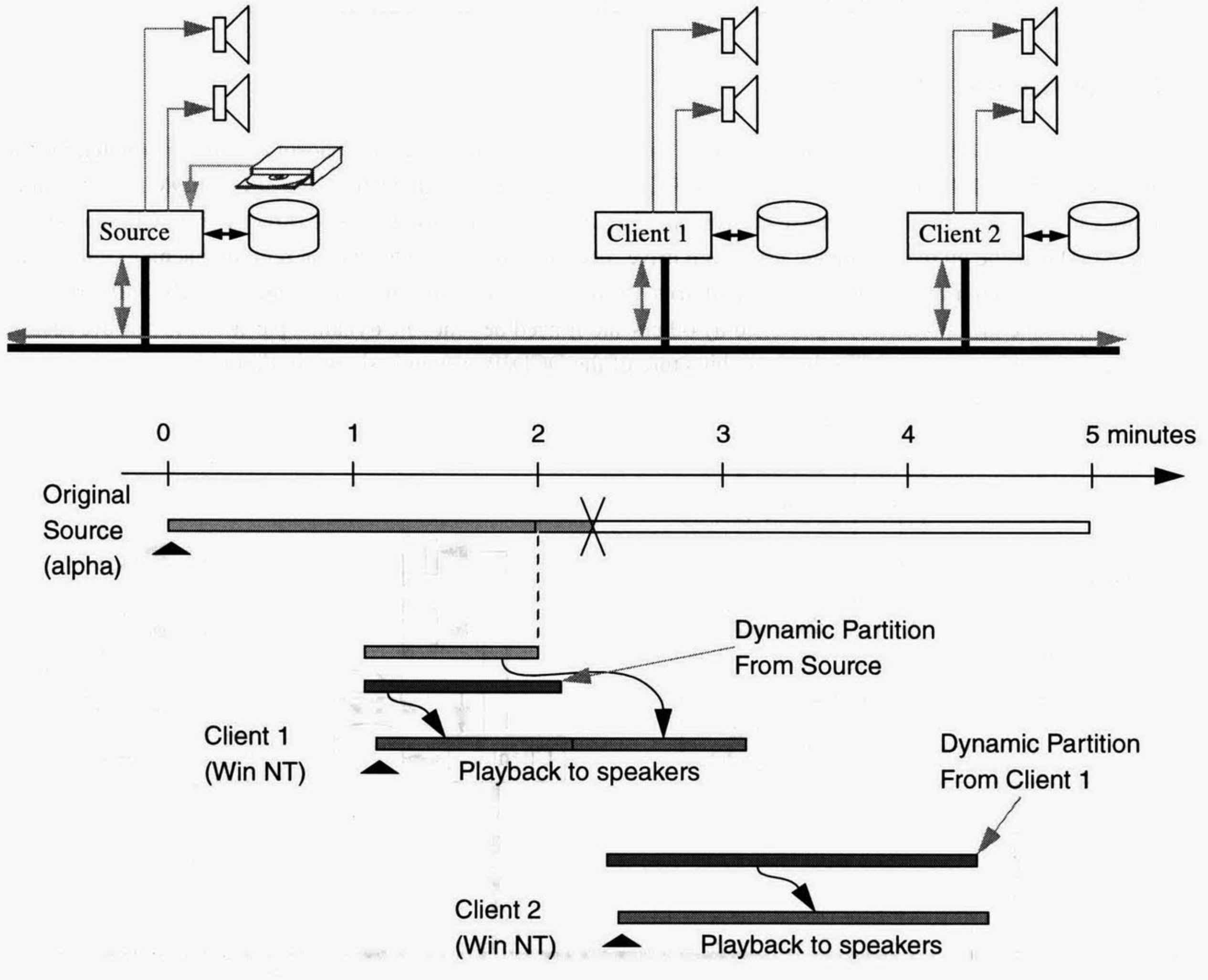


Figure 7-4 Distributed CD Juke Box recovery example

Now suppose that the source is terminated by an external force shortly after completing transmission of he dynamic partition created for client one. At this stage, client one has the first two minutes of the song in its cache, which it continues to play. Further, suppose that a second client—client two—arrives shortly after the server terminated, and that this second client's user has just heard from client one's user that he should listen to the song.

Client two's user will instruct his local smart network cache to locate and retrieve the song. It will then follow the same process that client one's smart network cache did with the following differences that result from the absence of the original source and the presence of the data in client two's cache. Since no active partitions exist for the song, client one makes a request for the entire 120 seconds. Client two, upon receiving this request, will confirm that it has the missing data, the resources to serve it, and also that it has permission to do so. A new dynamic partition source by client one is then created in the same manner as before.

The DCDJB application tested the following functionality:

- late joins
- autonomous dynamic partitioning
- disk access scheduling
- the performance of the HPRTP reference implementation under Windows NT

7.2.1 ADU Formats

Since audio was represented and transported as raw data, the same simple ADU format for the Reflection of Presence application was used for the Distributed CD Juke Box application.

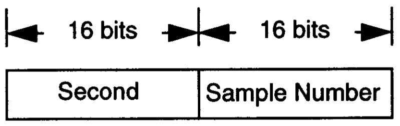


Figure 7-5 ADU Formats used by the Distributed CD Juke Box Application

7.2.2 Application Hints

As was the case for the Reflection of Presence application, neither hyperlinks were present, nor was the previous access mode used. Both of these hints may be used in later revisions of the application. For example, one could supply links to a web page for the CD's artist, or a list of the CD's tracks detailing the order in which a user played them. The Intra-Object Locality hint was set to Linear locality, thereby enabling the HPRTP cache to prefetch audio from disk and aid in the smooth playback of songs.

7.2.3 Content Hints

The content hints supplied by the primary server of each CD could be tailored according to the capabilities of the server and the desires of the person transmitting a particular CD. Disk caching was normally allowed, however, users could restrict access so that a CD's contents were not saved. In cases where caching was allowed; late-joins could be effected using dynamic partitions. Dynamic partitions were not allowed for CDs that were prohibited from being cached. The content hints used for the DCDJB application were as follows:

Content Hint	Caching allowed	Caching not allowed
LDS Hint payload type layering index layer drop order	 RTP_PT_L16_2 0 0	 RTP_PT_L16_2 0 0
Reliability/Latency Model	Reliable Bounded Delay Real Time	Reliable Bounded Delay Real Time
Personness	Third	Third
Volatility	Live Delayed	Live Delayed
Persistence	Indefinite	Transient

Table 7-2 Content Hints for the Distributed CD Juke Box application

Chapter 8

Conclusions, Deployment, and Future Work

8.1 Conclusions

This dissertation demonstrates the feasibility of scalable reliable on-demand multimedia delivery using the Internet Protocol. Quality of Service (QoS) or selective transmission augmentations to routers, while offering perhaps slightly better service, are shown to be nonessential for on-demand delivery. In addition, the questions regarding the scalability and deployment of schemes requiring router augmentation do not apply to the proposed delivery architecture. Thus, on-demand multimedia services can be provided with minimal modification to existing routers, and hence at a reduced expense to methods that require router modification.

The proposed delivery architecture consists of several novel technologies that span the gamut of data delivery, ranging from techniques for reliably delivering individual packets, to retrieving and caching entire missing streams. Some of these techniques are extensions and mergers of existing ideas, while others provide features that existing techniques lack. However, the main enabling feature of these technologies is the fact that their design and implementation were developed simultaneously. Numerous synergies result as a consequence.

All of these technologies are based upon the philosophy of Content and Application Negotiated Delivery (CANDY). CANDY extends the Application Level Framing (ALF) and Light Weight Session (LWS) philosophies. It espouses the exchange of meta-information about objects and receiver capabilities to expedite object delivery to session members. As a corollary, CANDY also requires that all session members participate in full peer-to-peer communication; there is no difference between senders and receivers.

The application of the CANDY philosophy resulted in the realization of the following ideas and technologies:

- A delivery taxonomy that relates reliability to latency.
- A set of hints that sending applications can use to characterize object data.
- A set of hints that receiving applications can use to characterize their likely future needs for an object's data.
- The extension of a localized recovery mechanism put forward by Scalable Reliable Multicast to incorporate late-joins.
- The Truncated Exponential Partitioning (TEP) technique for Video on Demand.
- The Scoped Hybrid Automatic Repeat reQuest with Forward Error Correction (SHARQFEC) method for the scalable reliable delivery of data over networks using IP multicast.
- The merger of the above ideas and techniques into a single integrated transport protocol, the Hierarchically Partitioned Reliable Transport Protocol (HPRTP), and an integrated media cache that merges data transport and caching into a single transparent service.

8.2 Outstanding Problems

While the realization of integrated media caches with HPRTP transport is a major step toward realizing a scalable, reliable transport mechanism for delivering on-demand multimedia, several outstanding issues remain. Some directly relate to HPRTP, while others are orthogonal and hence should be addressed through separate mechanisms.

8.2.1 Congestion Control

Congestion Control — the ability to ensure that a protocol's traffic volume remains bounded at all times— is an issue of key concern. Unicast protocols such as TCP solve this problem by requiring senders to slow their transmission rate when they detect that the receiver has missed packets. Such mechanisms are not advisable for multicast transport protocols, since the cumulative effect of uncorrelated losses at many receivers results in transmission rates invariably shrinking to accommodate the worst receiver, or in some cases zero.

Schemes that operate at the lowest common denominator may be acceptable for multicast file transfer; however, they are not usable for the delivery of streaming media. For this reason, HPRTP explicitly supports layered schemes that disperse an object's content over multiple partitions to which receivers may subscribe and leave in order to limit their bandwidth consumption. Layered congestion control schemes are not yet mature, and as McCanne notes in his Ph.D. thesis [53], several problems still remain. In particular:

- the deployment of scalable routing mechanisms,
- the creation of mechanisms that allow receivers to learn which layers others subscribe to without actually subscribing to them first,
- the requirement that receivers achieve consensus as to which layers they should subscribe to, and
- the interaction of layered session-traffic with traffic using other protocols.

HPRTP partially addresses the second of these concerns, but not the others. Further work is needed in this area.

8.2.2 Multicast Address Allocation

Multicast address allocation is another area that must be addressed before scalable multimedia delivery over the Internet can be realized. Currently, multicast addresses are allocated in an *ad hoc* manner using the "announce and listen" method provided by Mark Handley's Session Directory program, `sdr` [30]. Each `sdr` instance listens to announcements from other instances of itself, recording the address claimed within each announcement. When a new session is required, `sdr` generates an address or addresses at random from the valid address range of the desired scope. Should the new address clash with one being announced, and hence used, by another `sdr` instance, the process is repeated until no clash results.

While this program works well for the limited number of sessions that currently exist on the Mbone, it will not scale to the thousands that will be needed once multicast applications find widespread use. `sdr`'s role as multicast address allocator is further complicated by the fact that it merges the task of object resolution with address allocation. These two mechanisms should be distinct [30]. HPRTP's extensive use of multiple administratively scoped multicast addresses for achieving scalable reliability and traffic localization further exacerbates the problem.

These shortcomings are well known and are being addressed within a new Working Group on Multicast Address Allocation (malloc) within the IETF. Several Internet Drafts describing various schemes for scope discovery [111] and multicast address allocation [109, 110, 112, 113, 115, 116] are currently proposed. The schemes described in these drafts typically attempt to leverage or provide similar functionality to the Dynamic Host Configuration Protocol (dhcp) [96] and make temporary claims on small blocks of addresses. Ultimately, one or more of the ideas expressed in these drafts will result in a standard mechanism for multicast address allocation that will be described in an RFC.

8.2.3 Object Resolution

HPRTP deals solely with the delivery of a known object. It does not provide mechanisms for session members to find that object. As was just mentioned, the process of determining which multicast address upon which to find an HPRTP object's IOC channel is currently performed via `sdr`. This mechanism only allows currently active objects to be found, unopened objects present in remote caches cannot be found using this mechanism.

The task of determining the IOC channel for an object is essentially the same task as performing service location discovery. One can think of the provision of object content as the service that must be located. The task of service location is currently being studied with the IETF Service Location Protocol (svrloc) Working Group. This body has produced an RFC describing a protocol for performing service location within local-area networks [108], and is currently investigating ways to extend its functionality to support the location of services in a scalable fashion over wide-area networks [114, 117].

8.2.4 Ownership and Consistency Issues

The current mechanism for determining an object's originating source within HPRTP is the value of the Data Source Identifier (DSI) generated when a member joins a session. The DSI is randomly chosen to ensure that no two session members will have the same DSI. DSIs therefore uniquely identify a particular session member and can be used for the purposes of repair provision. DSIs are not known ahead of time, and are not assigned under the control of the user.

This presents a problem if one uses DSIs to determine who "owns" an object. Ownership is tied to a specific randomly generated DSI associated with a single point on the network. Should the "owner" of an object relocate to another node on the session, the tie is broken, and new object data from the new location is no longer associated with the original DSI and hence owner.

A better mechanism is needed to identify ownership. The simplest solution would be to disassociate ownership with the DSI and to use a different identifier for ownership. This would allow an object's ownership to be divorced from the place at which it originates on the network, and still allow the DSI to provide a means for anonymously identifying the origin of traffic.

8.3 Future Work

In addition to the work that must be done to solve the outstanding problems just listed, opportunities for further improvement exist in the mechanisms described within this dissertation.

8.3.1 Adaptive SHARQFEC Timers

The timers used within SHARQFEC are statically configured. That is, no attempt is made to modify the constants they use in order to minimize the number of redundant NACKs or repairs. SHARQFEC's predecessor, SRM, includes a mechanism for adjusting its timer constants. When turned on, this mechanism adjusts the timer constants according to the volume of redundant traffic observed. It can therefore limit the volume of redundant traffic to acceptable levels by increasing or decreasing the delays observed before the transmission of each NACK or repair.

The design and implementation of a mechanism for adaptive timers within SHARQFEC will be more difficult than for SRM, since SHARQFEC's timers operate on multiple losses within a group of packets instead of single packet losses. The analysis of SRM timers, upon which SHARQFEC's timers are based, presented in 4.1.1 on page 49 showed that the adaptive mechanisms are only needed in cases where significant branching is present. Thus, if SHARQFEC is deployed over well-constructed networks that create administratively scoped zones so as to minimize branching, the additional effort required to create adaptive timer mechanisms for SHARQFEC may well prove unnecessary.

8.3.2 Wide-Scale, Multiply Sourced Sessions

The ability to limit traffic scope through hierarchically arranged, administratively scoped regions is central to the operation of SHARQFEC within HPRTP. Further work is needed to explore how SHARQFEC's scoping mechanism would work in a collaborative environment where multiple sources exist in the same session. The application of HPRTP in such a situation would require additional logic within SHARQFEC to determine which scope zones should be used for the transmission of repair traffic. An example of how this might be achieved is shown in Figure 8-1.

In this example, sources exist in each of three disjoint local-scope zones. Participating receivers use different scopes according to the source of the traffic. Thus, the number of possible SHARQFEC topologies that must be supported is equal to the number of distinct local zones occupied by sources. However, this does not result in the creation of additional session traffic. The additional information needed would be included in data packets that would now carry inter-ZCR distance in the same manner as NACK and repair packets. Nodes would then determine the distance to an obscured data source within a local zone in the same manner as if the source were an ordinary node and had sent a NACK or repair.

While this solution solves the multi-sourcing problem in situations for which only one delivery tree is used between a session's nodes, it does not solve the more general problem that results when source-based routing trees are used. In these cases, there may be more than one path from a larger scoped zone into a smaller scoped zone. Hence, the ZCR may vary according to the external traffic source. One could modify the SHARQFEC session algorithm to allow for source-based ZCRs quite easily; this would result in higher session traffic. Increasing the number of ZCRs from one to two per zone results in a fourfold increase in session traffic. Further work is needed to determine the nature of administrative scope creation and how forthcoming inter-domain multicast routing protocols [89, 97] will affect multiply sourced SHARQFEC traffic.

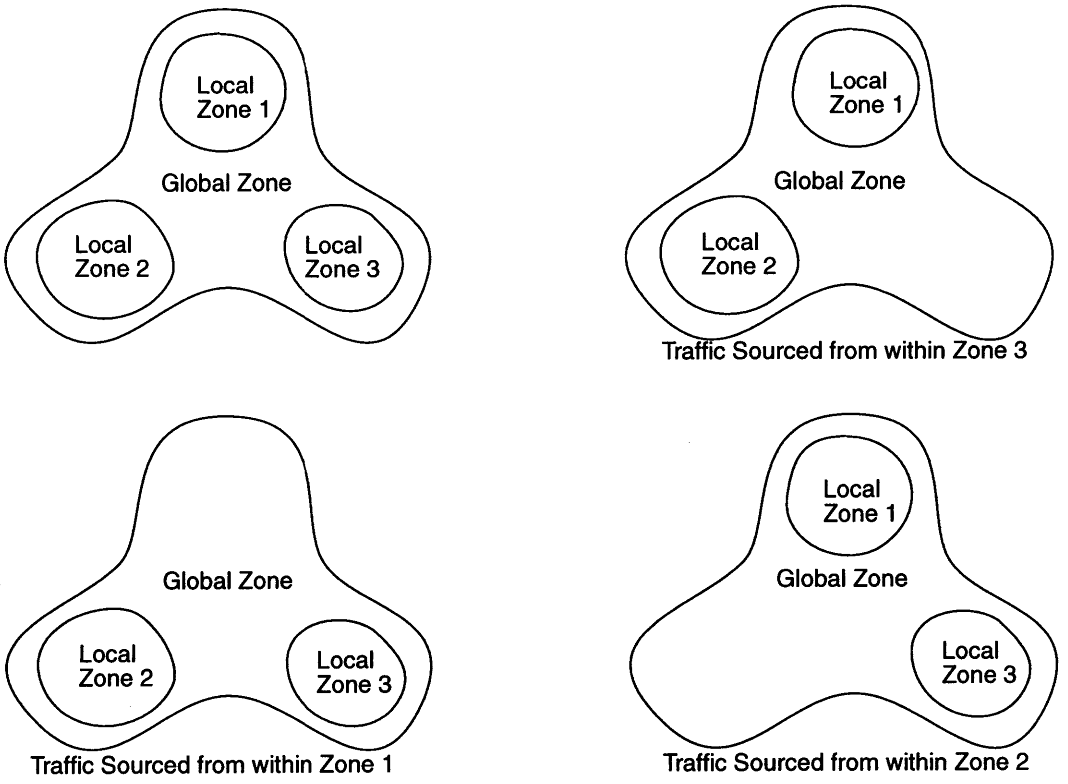


Figure 8-1 Topologies for a hypothetical multiply-sourced SHARQFEC session

8.3.3 Router Support

The HPRTP protocol is designed to work with extremely large widely, dispersed receiver sets that are separated using administrative scoping. The scalability it affords through SHARQFEC does not require router modification, and hence works with either sparse- or dense-mode multicast routing protocols. There is, however, one small restriction; all traffic crossing a smaller scoped region either to or from a larger scoped one must do so through a common border gateway router. This does not mean that there cannot be multiple avenues of egress among regions of different scope, only that one is selected for the particular set of multicast addresses associated with a session. Without this restriction, the assumptions fail that allow indirect RTT determination and automatic redundancy injection by ZCRs.

Thus, HPRTP makes the implicit assumption that routing trees from multiple groups are at least partially shared. This fact presents opportunities for reducing group maintenance overhead. Instead of constructing a multicast tree for each group, a single source-based tree could be constructed with the tree required for smaller scopes using subsets from the original. If IPv6 [9, 93, 95] were used, traffic filtering at scope boundaries could be done using the scope field within the IPv6 multicast address. The separate LDSC and repair channels created at different scopes could then be differentiated and pruned using the flow-label field. Since the amount of state that routers would maintain on a per-flow basis for a single group would be similar to what they would maintain on a per-group basis for multiple groups, the savings might be small. Further study is needed to determine if such modifications would be beneficial in the long term.

On a different track, the long-held assumption that one must not modify the routers is weakening. Levine's Tracer protocol [47, 48] — which makes clever use of routing labels and the `mtrace` facility for determining the multicast path between two hosts — and Cisco's Pretty Good Multicast (PGM) proposal [118] — which creates loss trees for repair packets to follow — point toward mechanisms that are capable of implementing subcasting [58]. Questions still remain about their ability to scale, since their use creates state directly proportional to the number of packets lost. However, should such a mechanism be deployed, a merger with HPRTP's SHARQFEC mechanism might result in further localization.

8.3.4 Data Naming

HPRTP's naming scheme is optimized for the description of ADUs arranged in fixed, multiple, monotonically increasing segments. It does not cope well with arbitrary naming schemes, nor can the naming scheme for an object be extended easily once an object has been opened. These limitations would not prevent HPRTP from being used for gaming applications in which each player's actions are transmitted as sequentially numbered events.

They present a significant problem were HPRTP to be used as the transport mechanism for communication between a networked text editor for authoring dynamic, hierarchically configured documents. In this case, the naming scheme must be fine-grained and extensible. In addition, one should be able to attach textual descriptions, such as a chapter title or section name, to ADUs in lieu of numbers. The addition of arbitrary naming mechanisms to HPRTP would not be overly difficult, and could probably be achieved through the use of out-of-band means. However, the addition of the necessary synchronization and consistency mechanisms needed to manage the changing namespace is not so simple as it is still the subject of active research.

8.3.5 Small, Discrete Objects

While HPRTP's delivery mechanism was designed to be as general as possible, the majority of example applications explored within this dissertation have concentrated on the delivery of streaming audio and video. Thus, there is an implicit assumption that the volume of data to be delivered is large and has isochronous qualities. This assumption led to the development of methods in which multicast delivery channels were created with the expectation that other session members would be able to use data originally destined for other session members.

Since the probability that a multicast channel would be useful to more than one session member at a given time is tied to how long it is active, it is not clear that using single, multicast channels to deliver a small object — such as a Web page — would result in any bandwidth savings. Thus, one must group multiple objects for delivery over the same channel. This idea has been explored by Clark and Ammar for the case where Web objects are transmitted over separate groups whose address is generated by hashing the text of the Web object's URL [23]. HPRTP's flexible naming scheme and scoping mechanisms point toward a different architecture in which traffic is segregated upon the basis of its source, content type, and location of receivers.

8.4 Deployment

Multicast is still a fledgling technology that—despite being over ten years old—has many unresolved issues. Reliability, congestion control, billing, security, and political issues still inhibit network providers from deploying multicast in a widespread fashion over the public Internet. The IP Multicast Initiative (IPMI), formed in 1996 and consisting of 40 companies worldwide, is attempting to remove this inhibition. IPMI's task, however, will not be easy.

As of February 1998, nearly two-thirds of the routers on the Mbone support native IP multicast. The majority of these multicast routers are located on the edges of the network and not in the backbone. Service providers, used to billing both sender and receiver for network usage, are finding that their current business models do not work for multicast traffic. In addition, the current lack of congestion control mechanisms, combined with the receiver's ability to join and receive multicast traffic without sender consent, makes many providers nervous. Thus, real and well-founded fears about maintaining control over traffic originating from third-party networks have led many providers to adopt expensive "sender pays" charging models that cost tens of thousands of dollars per source.

This does not bode well for reliable multicast schemes that require all members to communicate in a peer-to-peer fashion over multicast IP. Efficiency is limited by restricting multicast traffic to the point-to-multipoint delivery model, (by blocking the multipoint-to-multipoint model), but that is not important to service providers. Their interests lie more in maintaining network control, and moving toward new services that will allow them to move away from the flat rate "all you can eat" charging models used today. Ironically, restricting which nodes can source traffic may prevent them from achieving this shift over the long term.

In the short term, however, the restricted few-to-many model is paying dividends in private networks outside the public Internet. Many companies, such as General Motors, use multicast over corporate intranets to distribute software and sales materials. Also, the Internet Engineering Task Force's (IETF) Unidirectional Link Routing (udlr) working group is seeing multicast increasingly deployed over satellite links that use terrestrial return paths. These asymmetric satellite solutions are particularly interesting since they provide high-bandwidth, high-latency delivery to the user over the satellite, and limited low-bandwidth, low-latency delivery via the terrestrial return. As such, the pre-emptive repair strategy used by the SHARQFEC mechanism within HPRTP makes it a prime candidate for mass media delivery via satellite.

Cable television (CATV) is another network type poised to exploit multicast in a few-to-many fashion [8, 36, 60, 62, 75]. Cable access providers not only have access to extremely high bandwidth paths that feed into people's homes, but also the means to manage them with greater control than possible with satellite links. Typically, cable systems are deployed so that the final link is shared by several hundred to a few thousand homes. Should a link become overused, cable companies divide the link into two, half the number of subscribers served by a link, and hence double the available bandwidth. Thus, cable delivery systems enjoy an advantage over satellite delivery systems in that the channel spectrum can be reused to provide additional bandwidth.

Ultimately, it is likely that hybrid systems consisting of both wireless and wireline systems will emerge. Long haul links and wide-distribution services will be provided by satellite links that feed strategically placed caches. These caches will then provide content locally over wireline systems that may include cable television and Digital Subscriber Line (x-DSL) [16] links. Real-time delivery of two-way content, such as conferencing, multi-user game events, and shared workspaces will likely use pure wireline systems to avoid the propagation delay inherent in the use of satellite links. The HPRTP protocol is ideally poised to exploit such hybrid systems when they become available.

References

Papers, Articles, Standards, and Books

[1] N. Abrahamson and W. Bender, "Context-sensitive multimedia," Proceedings of the SPIE, Vol. 1785, September 1992.

[2] S. Agamanolis, and V.M. Bove, Jr., "Multilevel Scripting for Responsive Multimedia," IEEE Multimedia, Volume 4, Number 4, pp. 40-50, Oct-Dec 1997.

[3] S. Agamanolis, A. Westner, and V.M. Bove, Jr., "Reflection of Presence: Toward more natural and responsive telecollaboration," Proc. SPIE Multimedia Networks, Vol. 3228, 1997.

[4] J. A. Al-Marri and S. Ghandeharizadeh, "An Evaluation of Alternative Disk Scheduling Techniques in Support of Variable Bit Rate Continuous Media," University of Southern California, Computer Science Department Research Report, 98-666.

[5] K.C.Almeroth, M.H. Ammar, "The Interactive Multimedia Jukebox (IMJ): A New Paradigm for the On-Demand Delivery of Audio/Video," Proceedings of the Seventh International World Wide Web Conference, Brisbane, Australia, April 1998.

[6] A.C. Aggarwal, J.L. Wolf, and P.S. Yu, "A permutation-based pyramid broadcasting scheme for video-on-demand systems," In Proc. of the IEEE Int'l Conf. on Multimedia Systems '96, Hiroshima, Japan, June 1996.

[7] U. Black, "ATM: Foundation for Broadband Networks," Prentice Hall, 1995.

[8] C. Bisdikian, K. Maruyama, D.I. Seidman, and D.N. Serpanos, "Cable Access Beyond the Hype: On Residential Broadband Data Services over HFC Networks," IEEE Communications Magazine, November 1996.

[9] S.O. Bradner and A. Mankin (editors), "IPng, Internet Protocol Next Generation," Addison Wesley, 1995.

[10] S. Brand, "The Media Lab: Inventing the Future at M.I.T.," Penguin Books, 1987.

[11] F. Brockners, "Bulk multicast data transfer - towards the integration of FEC and ARQ using a lightweight feedback control tree," University of Cologne, Technical Report TR97-279, July, 1997.

[12] S.Casner and S. Deering, "The First IETF Internet audiocast," ConneXions, 6(6):10-17, 1992.

[13] V. Cerf and R. Kahn, "A Protocol for Packet Network Interconnection," IEEE Trans. on Commun., vol. COM-22, pp. 637-648, May 1974.

[14] J.Chang, and N. Maxemchuk, "Reliable Broadcast Protocols," ACM Transaction on Computer Systems, Vol. 2, No. 3, pp. 251-275, August 1984.

[15] A.Chankhunthod, P. Danzig, C. Neerdaels, M.Schwartz, & K.Worrel, "A Hierarchical Internet Object Cache," http://newbruno.cs.colorado.edu/harvest/technical.html#cache

[16] W. Y. Chen, D. L. Waring, "Applicability of ADSL to Support Video Dial Tone in the Copper Loop," IEEE Communications Magazine, pp. 102-109, May 1994.

[17] S.Y. Cheung, M.H. Ammar, and X. Li, "On the Use of Destination Set Grouping to Improve Fairness in Multicast Video Distribution," College of Computing, Georgia Institute of Technology, Technical Report: GIT-CC-95-25.

[18] Cisco Systems, "Shared Network Caching and Cisco's Cache Engine," White Paper, 1997.

[19] D. Clark, and D. Tennenhouse, "Architectural Considerations for a New Generation of Protocols," Proceedings of ACM SIGCOMM '90, September 1990, pp. 201 -208.

[20] R.J. Clark and M.H. Ammar, "Providing Scalable Web Service Using Multicast Delivery," Computer Networks and ISDN Systems Journal, Vol. 29, pp. 841-858, 1997.

[21] A.Dan, Y.Heights, and D.Sitaram, "Generalized interval caching policy for mixed interactive and long video workloads," Proceedings of ACM Multimedia Computing and Networking, pages 15 - 23, San Francisco, October, 1994.

[22] A.Dan, D.Sitaram, and P.Shahabuddin, "Scheduling policies for an on-demand video server with batching," Multimedia Systems, 4(3):112-121, June 1996.

[23] P.Danzig, R. Hall, M. Schwartz, "A Case for Caching File Objects Inside Internetworks," Proceedings of ACM SIGCOMM '93, pp. 239-248, September 1993.

[24] E.X. DeJesus, "Storage: Part II, 1Infinite Space," Byte Magazine, Vol. 23. No.2, February 1998, pp.112 -124.

[25] K.Dienes, "Information Architectures for Personalized Multimedia," Master's Thesis, MIT Media Laboratory, June 1994, Cambridge MA, USA.

[26] C.Federighi and L.A. Rowe, "A Distributed Hierarchical Storage Manager for a Video-on-Demand System," Storage and Retrieval for Image and Video Databases II, IS&T/SPIE, Symp. on Elec. Imaging Sci. & Tech., San Jose, CA, February 1994.

[27] S.Floyd, V. Jacobson, S McCanne, C. Liu, & L. Zhang, "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing," SIGCOMM '95, Cambridge USA, pp. 342-356.

[28] L. Gautier and C. Diot. "MiMaze, a Multiuser Game over the Internet," INRIA Research Report 3248. INRIA Sophia Antipolis (France). September 1997.

[29] J. Gemmell, "Scalable Reliable Multicast Using Erasure-Correcting Re-Sends," Microsoft Research Technical Report, MSR-TR-97-20, June 30, 1997.

[30] M. Handley, "On Scalable Multimedia Conferencing Systems," Ph.D. Thesis, University of London, 1997.

[31] M. Hofmann, "A Generic Concept for Large-Scale Multicast," International Zurich Seminar on Digital Communication, February 21-23, 1996, Zurich, Switzerland, Ed.: B. Plattner, Lecture Notes in Computer Science, No. 1044, Springer Verlag, 1996.

[32] M. Hofmann, "Adding Scalability to Transport Level Multicast," Third International COST 237 Workshop, November 25-27, 1996, Barcelona, Spain, Ed.: G. Ventre, J. Domingo-Pascual, A. Danthine, Lecture Notes in Computer Science, No. 1185, Springer Verlag, 1996.

[33] H.W. Holbrook, S.K.Singhal, and D.R. Cheriton, "Log-based receiver-reliable multicast for distributed interactive simulation," Proceedings of ACM SIGCOMM '95, October 1995.

[34] K.A. Hua and S.Sheu, "Skyscraper Broadcasting: A New Broadcasting Scheme for Metropolitan Video-on-Demand Systems," Computer Communications Review ACM SIGCOMM, vol. 27, no. 4, 1997.

[35] C. Huitema, "The case for packet level FEC," Proceedings of IFIP 5th International Workshop on Protocols for High Speed Networks (PfHSN'96), INRIA, Sophia Antipolis, FRANCE, October 1995, IFIP, Chapman & Hall.

[36] P.A. Humlet, and M.G. Troulis, "The Information Driveway," IEEE Communications Magazine, 1997.

[37] ISO/IEC DIS 10918-1, ITU- T Rec.T.81 (JPEG) "Information Technology - Digital compression and coding of continuous-tone still images," 1992.

[38] ISO/IEC 11172 (MPEG-1) "Information technology - Coding of moving picture and associated audio for digital storage media as up to about 1.5 Mbit/s," 1993.

[39] ISO/IEC 13818-1 (MPEG-2), "Generic Coding of Moving Pictures and Associated Audio - Part 1, Systems," Draft International Standard, November 1994.

[40] ISO/IEC 138182-2 (MPEG-2), "Generic Coding of Moving Pictures and Associated Audio - Part 2, Video," Draft International Standard, November 1994.

[41] ISO/IEC 138182-3 (MPEG-2), "Generic Coding of Moving Pictures and Associated Audio - Part 3, Audio," Draft International Standard, November 1994.

[42] V. Jacobson, "Congestion Avoidance and Control," Computer Communication Review ACM SIGCOMM, vol. 18, no.4, pp.314-329, August 1988.

[43] V. Jacobson, "Multimedia Conferencing on the Internet," Tutorial Notes - ACM SIGCOMM94, London, September 1994.

[44] M. Kaashoek, A. Tananbaum, S Flynn Hummel, and H.E. Bal, "An Efficient Reliable Broadcast Protocol," Operating Systems Review, October 1989.

[45] R. Kay, "Storage: Part I, 15 Disks Cover More Data Than Ever," Byte Magazine, Vol. 23. No.2, February 1998, pp. 112 -119.

[46] R.G. Kermode, H.N. Holtzman, & A.B. Lippman, "The Media Bank: A Fault Tolerant Heterogeneous Distributed Pull Based Architecture for the Delivery of Object Oriented Multimedia across Packet Switched Networks," 7th International Workshop on Packet Video, March 18-19, 1996, Brisbane Australia.

[47] B.N. Levine, J.J. Garcia-Luna-Aceves, "Improving Internet Multicast with Routing Labels," IEEE International Conference on Network Protocols (ICNP-97), October 28 - 31, 1997. p. 241-250.

[48] B.N. Levine, S. Paul, and J.J. Garcia-Luna-Aceves, "Deterministic Organization of Multicast Receivers based on Packet-Loss Correlation," Presentation, Reliable Multicast Research Group meeting Orlando, Florida, February 22-24, 1998.

[49] X. Li, S. Paul, P. Pancha, & M. Ammar, "Layered Video Multicast with Retransmission (LVMR): Evaluation of Error Recovery Schemes," Proceedings of NOSSDAV 97, St. Louis. MO, May 1997.

[50] J. Lin and S. Paul, "RMTP: A Reliable Multicast Transport Protocol," Proceedings of IEEE INFOCOM '96, March 1996, pp. 1414-1424.

[51] A.B. Lippman and R.G. Kermode, "Media Banks: Entertainment and the Internet," IBM Systems Journal, Vol. 35, No3&4, October 1996.

[52] S.McCanne and V. Jacobson, "vic: A Flexible Framework for Packet Video," ACM Multimedia, November 1995, San Francisco, USA.

[53] S. McCanne, "Scalable Compression and Transmission of Internet Multicast Video," Ph.D. Thesis, University of California at Berkeley, UCB/CSD-96-928, December 1996.

[54] S.McCanne, V. Jacobson, and M. Vetterli, "Receiver Driven Layered Multicast," Proceedings of ACM SIGCOMM '96, August 1996, Stanford CA.

[55] S. Mittra, "Iolus: A framework for Scalable Secure Multicasting," Proceedings of ACM SIGCOMM '97, September 14 - 18, 1997, Cannes, France.

[56] J. Nonnenmacher, E.W. Biersack, Reliable Multicast, "Where to use Forward Error Correction," Proceedings of the 5th Workshop on Protocols for High Speed Networks, pp. 134-148, Sophia Antipolis, France, Oct. 1996.

[57] J. Nonnenmacher, E. Biersack, D. Towsley, "Parity-Based Loss Recovery for Reliable Multicast transmission," Computer Communications Review ACM SIGCOMM, volume 27, number 4, 1997.

[58] C. Papadopoulos, G. Parulkar, and G. Varghese, "An Error Control Scheme for Large-Scale Multicast," Seventeenth IEEE International Conference on Computer Communications, Infocom 98, San Francisco, April 1998.

[59] S. Paul, K.Sabnani, J. Lin, and S. Bhattacharyya, "Reliable Multicast Transport Protocol (RMTP)," To appear IEEE Journal on Selected Areas in Communications, special issue on Network Support for Multipoint Communication.

[60] T.S. Perry, "The trials and travails of interactive TV," IEEE Spectrum, April 1996, pp. 22-28.

[61] R. Rada, "Interactive Media," Springer-Verlag, New York Inc., 1995, pp. 155-164.

[62] A. Reinhardt, "Building the Data Highway," Byte Magazine, March 1994.

[63] L. Rizzo, "Effective Erasure Codes for Reliable Computer Communication Protocols," ACM Computer Communications Review, Vol. 27, n.2, April. 97, pp. 24-36.

[64] L. Rizzo and L. Vicisano, "A Reliable Multicast data Distribution Protocol based on software FEC techniques," Proceedings of the Fourth IEEE, HPCS'97 Workshop, Chalkidiki, Greece, June 1997.

[65] M. Rose, "The Open Book: A Practical Perspective on OSI," Prentice Hall, 1990.

[66] L.A. Rowe, J.S. Boreczky, & D.B. Berger, "A Distributed Hierarchical Video-on-Demand System," Proceeding of IEEE International Conference on Image Processing Conference, Washington, D.C., U.S.A., October1995, Vol I, pp. 334 - 337.

[67] W.R. Stevens, "TCP/IP Illustrated Volume 1: The Protocols," Addison Wesley, 1994.

[68] W.R. Stevens, "TCP/IP Illustrated, Volume 3 TCP for Transactions, HTTP, NNTP, and the UNIX Domain Protocols," Addison Wesley, 1996.

[69] A. Tanenbaum, "Computer Networks," Third Edition, Prentice Hall, 1996.

[70] D. Thaler; S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, and L. Wei, "Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification," July 1997.

[71] H. M. Vin and P.V. Rangan, "Designing a multiuser HDTV storage server," IEEE Journal on Selected Areas in Communications, Vol 11., no 1, January 1993, pp. 152-164.

[72] S. Viswanathan and T. Imielinski, "Metropolitan are video -on-demand service using pyramid broadcasting," Multimedia Systems, 4(4):197-208, August 1996.

[73] J. Werner and L.C. Wolf, "Scheduling Mechanisms Reducing Contention Situations in Multimedia Systems," Proceeding of European Workshop Interactive Distributed Multimedia Systems and Services (IDMS), Berlin, Germany, March, 1996.

[74] B. Whetten, S.Kaplan, & T. Montgomery, "A High Performance Totally Ordered Multicast Protocol," submitted to INFOCOM '95.

[75] R.Whittle, "The Great Australian Cable Race," Australian Communications Magazine, December/January 1995 - 96.

[76] G.R. Wright & W.R. Stevens, "TCP/IP Illustrated, Volume 2: The Implementation," Addison Wesley, 1995

[77] M. Yajnik, J. Kurose, and D. Towsley, "Packet Loss Correlation in the MBone multicast network," IEEE Global Internet Mini-Conf. GLOBECOM'96, Nov. 1996.

[78] R. Yavatkar, J. Griffioen, and M. Sudan, "A Reliable Dissemination Protocol for Interactive Collaborations," Proceedings of ACM Multimedia 95, 1995.

World Wide Web Sites

[79] Real Networks Home Page, http://www.real.com

[80] The Netrek Home Page, http://www.netrek.org

[81] UCB/LBNL/VINT Network Simulator, ns, http://www-mash.cs.berkeley.edu/ns

[82] UCB/LBNL/VINT Network Animator, nam, http://www-mash.cs.berkeley.edu/ns/nam.html

[83] Squid Internet Object Cache Page, http://squid.nlanr.net/Squid/

[84] S. McCanne, "A Distributed Whiteboard for Network Conferencing," unpublished report, http://HTTP.CS.Berkeley.EDU/~mccanne/unpublished.html#wb-work

IETF Request for Comments

[85] R. Atkinson, "Security Architecture for the Internet," RFC 1825, August 1995.

[86] R. Atkinson, "IP Authentication Header," RFC 1826, August 1995.

[87] R. Atkinson, "IP Encapsulating Security Payload (ESP)," August 1995.

[88] A. Ballardie, "Scalable Multicast Key Distribution," RFC 1949, May 1996.

[89] A. Ballardie, "Core Based Trees (CBT version 2) Multicast Routing," RFC 2189, September 1997.

[90] M. Borden, E. Crawley, B. Davie, and S. Batsell, "Integration of Real-time Services in an IP-ATM Network Architecture," RFC 1821, August 1995.

[91] R. Braden and D. Clark, "Integrated Services in the Internet Architecture: An Overview," RFC 1633, June 1994.

[92] R. Braden (Ed.), L. Zhang, S. Berson, S. Herzog, S. Jamin, "Resource ReSerVation Protocol (RSVP Version 1 Functional Specification," RFC 2205, November 1995.

[93] S. Deering, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification," RFC 1885, December 1995.

[94] S. Deering, "Host Extensions for Multicasting," RFC 1112, Stanford University, August 1989.

[95] S. Deering, and R. Hinden, "Internet Protocol, Version 6, Specification," RFC 1883, December 1995.

[96] R. Droms, "Dynamic Host Configuration Protocol," RFC 2131, March 1997.

[97] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, and L. Wei, "Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification," RFC 2117, June 1997.

[98] W. Fenner, "Internet Group Management Protocol, Version 2," RFC 2236, November, 1997.

[99] M. Laubach, "Classical IP and ARP over ATM," RFC 1577, January 1994.

[100] J. Moy, "Multicast Extensions to OSPF," RFC 1584, March 1994.

[101] J. Moy, "MOSPF: Analysis and Experience," RFC 1585, March 1994.

[102] J. Nagle, "Congestion Control in IP/TCP Internetworks," RFC 896, January 1984.

[103] J. Postel, "Internet Protocol," RFC 791, September 1981.

[104] R. Rivest, "The MD5 Message-Digest Algorithm," RFC 1321, January 1992.

[105] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A transport Protocol for Real-Time Applications," RFC 1889, January 1996.

[106] H. Schulzrinne and G. Fokus, "RTP Profile for Audio and Video Conferences with Minimal Control," RFC 1890, January 1996.

[107] D. Waitzman, C. Partridge, and S. Deering, "Distance Vector Multicast Routing Protocol," RFC 1075, November, 1988.

[108] J. Veizades, E. Guttman, C. Perkins, S. Kaplan.S. Kaplan, "Service Location Protocol," RFC 2165, June 1997

Internet Drafts[1]

[109] D. Claim (MASC) Protocol," draft-ietf-idmr-masc-00.txt, November 20, 1997.

[110] M. Handley (ISI), "Multicast Address Allocation Protocol (AAP)," draft-handley-aap-00.txt, December 15, 1997.

[111] M. Handley (ISI), "Multicast-Scope Zone Announcement Protocol (MZAP)," draft-ietf-mboned-mzap-00.txt, December 15, 1997.

[112] M. Handley (ISI), D. Thaler (University of Michigan), D. Estrin (ISI), "The Internet Multicast Address Allocation Architecture," draft-handley-malloc-arch-00.txt, December 15, 1997.

[113] S. Hanna (Sun Microsystems, Inc.), "Multicast Address Request Protocol (MARP)," draft-hanna-marp-00.txt, January 30, 1998.

[114] R. Moats (AT&T), M. Hamilton (Loughborough University), P. Leach (Microsoft), "Finding Stuff (How to discover services)," draft-ietf-svrloc-discovery-05.txt, October 31, 1997.

[115] B.V. Patel (Intel Corporation), M. Shah (Microsoft Corporation), "Multicast address allocation extensions options," draft-ietf-dhc-multopt-02.txt, November 20, 1997.

[116] B.V. Patel (Intel Corporation), M. Shah (Microsoft Corporation), "Multicast address allocation extensions to the Dynamic Host Configuration Protocol," draft-ietf-dhc-mdhcp-03.txt, November 20, 1997.

[117] J. Rosenberg, H.Schulzrinne, and B.Suter (all authors with Bell Laboratories), "Wide Area Network Service Location," draft-ietf-svrloc-wasrv-01.txt, November 14, 1997.

[118] T. Speakman, D. Farinacci, S.Lin, and A. Tweedly (all authors with Cisco Systems), "Pretty Good Multicast (PGM) Reliable Transport Protocol Specification," draft-speakman-pgm-spec-01.txt, January 29,1998.

1. Internet Drafts are working documents that have a maximum lifetime of six months. If the draft becomes an RFC or authors do not revise draft within six months of the date the current version was published, it ceases to exist. Draft names include a version number that is incremented by 1 for each revision. For example a revision to the draft draft-hanna-marp-00.txt would become draft-hanna-marp-01.txt. Internet drafts may be retrieved from the IETF web site http://www.ietf.org/

Appendix A

Hierarchically Partitioned Reliable Transport Protocol

The Hierarchically Partitioned Reliable Transport Protocol(HPRTP) uses multiple multicast channels to reduce the amount of duplicated data that a receiver is forced to retrieve. The multicast channels are arranged in a tree with three levels as shown below

```
    Level 2 - Intra Object Control Channel                      IOC
                                                                /    \
    Level 1 - Logical Data Stream Control Channel(s)       LDSC0 . . . LDSCl
                                                           /  |  \
    Level 0 - Partition(s)                             L0P0  L0P1..L0Pp
```

The root of the hierarchy is the Intra Object Control (IOC) channel, this channel is used to identify the address of the Logical Data Stream Control (LDSC) Control channels and the Partition (P) channels. In addition, it is also to identify the Logical Data Stream payload types, Application Data Unit Identifier format, and the partitioning scheme characteristics used on the Logical Data Streams. In cases where dynamic partitioning is allowed, it is also used to request and schedule partitions.

The second level of the hierarchy consists of the Logical Data Stream Control (LDSC) channels. These channels are used for making repair requests and repair indications.

The third of the hierarchy consists of the logical data stream partition data and repair channels for each logical data stream. These channels carry the actual object data.

A.1 HPRTP Level Zero (Intra Object Control) Packet Formats

A.1.1 Level Zero Common Header Format

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=0|L=0|  CMD   |  ADUI Length  |      LDS ID     |               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Version (V): 2 bits
    Identifies the HPRTP version, zero (0).

Level (L): 2 bits
    This field identifies the level in the hierarchy of a particular packet, for the IOC level this is set to zero (0).

```
Command (CMD) : 4 bits
    The IOC command which may take one of the following values:
      0x0 Session Message
      0x1 Zone Closest Receiver Challenge (ZCR_CHALL)
      0x2 Zone Closest Receiver Response  (ZCR_RESP)
      0x3 Zone Closest Receiver Take Over (ZCR_TKO)
      0x4 Logical Data Stream Description request
      0x5 Logical Data Stream Description
      0x6 Partition Description request
      0x7 Partition Description
      0x8 Session Description Request
      0x9 Session Description
      0xa Sender Description Request
      0xb Sender Description

ADUI Length : 8 bits
    The length of the Application Data Unit Identifier (ADUI) in bits. The
    number of bits is rounded up to the nearest multiple of 32. The resulting
    ADUI is right justified so that any unused bits appear starting at the
    first octet in the identifier.

LDS ID : 8 bits
    The logical data stream identifier corresponding to the data contained
    within this packet.

Partition ID : 8 bits
    The partition identifier corresponding to the data contained within this
    packet.
```

A.1.2 Session Message

Session message packets are used by session members for estimating the host-to-host delay between each other. The basic mechanism is the same one that is used by RTP and SRM. See section 6.3.2 of rfc1889 for details.

```
                    1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |V=0|L=2|CMD=0x0|SndSCOP|ZCRSCOP|               |   Num_DSIs    |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                Data Source Identifier (DSI)                   |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                           time_sent                           |
 +=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 |                             DSI_n                             |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |      Last Time Last Session was sent from DSI_n (LTS)         |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |    Last Time Last Session was received from DSI_n (LTR)       |
 +=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

SndSCOP : 8 bits
The SCOP value of the administratively scoped region to which this message was sent.

ZCRSCOP : 8 bits
The SCOP value of the smallest administratively scoped region to which the sender of this message belongs. N.B., SndSCOP will equal ZCRSCOP for normal receivers. Receivers acting as ZCRs will participate in session maintenance for two SCOP regions, their own local region where SndSCOP = ZCRSCOP, and their parent's region where SndSCOP > ZCRSCOP.

Num_DSIs : 8 bits
Number of DSI entries (n) contained in this message.

Data Source Identifier (DSI) : 32 bits
The DSI field identifies the sender of the session message. The DSI is chosen randomly, with the intent that no two DSIs within the same HPRTP session will have the same value. This field is semantically and syntactically identical to the SSRC identifier described within the RTP protocol.

DSI_1 : 32 bits
The DSI of the receiver sending the session message.

times_sent: 32 bits
The middle 32 bits of a 64 bit NTP timestamp generated by the receiver just prior to sending the session message. The NTP timestamp format is defined in rfc2030, it consists of a 64 bit fixed point number consisting of a 32 bit integer in the first 32 bits and a 32 bit fractional part in the second 32 bits. Thus the 32 bit timestamp used here consists of a 16 bit integer in the first 16 bits and a 16 bit fractional part in the second 16 bits and has an accuracy of 1/65536 second.

Last Time Last Session was sent from DSI_n (LTS) : 32 bits
The time_sent field extracted from the session message sent by DSI_n as last seen by the sender of the packet (DSI).

Last Time Last Session was received from DSI_n (LTR) : 32 bits
The time that DSI last received a session message from DSI_n. Same format as the timestamp field.

The host-to-host delay is estimated using the following equation;

Delay = ((LTS - LTR) - (Time this packet received - time_sent))/2

A.1.3 ZCR Challenge

ZCRs in child zones, periodically issue ZCR Challenge messages to their parent ZCRs as a means of confirming that they are the closest receiver within the child zone to the parent ZCR. Receivers within the child zone that receive the ZCR Challenge message, record the time it was received, the child ZCR's estimate of the RTT delay to the parent ZCR, and await the ZCR response form the Parent. When a matching ZCR response is received the children compute their own distances to the parent and, if closer, send a ZCR take over message to assume responsibility as the new ZCR for the child zone.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=0|L=2|CMD=0x1|SndSCOP|ZCRSCOP|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Local ZCR DSI                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             time_sent                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Local ZCR to Parent ZCR transit delay           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

SndSCOP : 8 bits
    The SCOP value of the administratively scoped region to which this message was sent.

ZCRSCOP : 8 bits
    The SCOP value of the smallest administratively scoped region to which the sender of this message belongs. N.B., SndSCOP will equal ZCRSCOP for normal receivers. Receivers acting as ZCRs will participate in session maintenance for two SCOP regions, their own local region where SndSCOP = ZCRSCOP, and their parent's region where SndSCOP > ZCRSCOP.

Reply Count : 8 bits
    The number of Timestamp Replies minus one. Up to 256 replies can be contained within a single message

Local ZCR DSI : 32 bits
    The DSI of the local ZCR sending the challenge.

time_sent : 32 bits
    Time the response was sent, same format as for the time_sent field in a session messages.

Local ZCR to parent ZCR delay : 32 bits
    The Local ZCR's estimate of the transit delay to the parent ZCR formatted in the same manner as time_sent field.

A.1.4 ZCR Response

ZCRs receiving a ZCR challenge from a lesser scoped region shall immediately respond with a ZCR response message. This message enables the receivers in the lesser scoped zone to estimate the transit delay between themselves and the parent ZCR by using the following equation;

```
    Delay = delay_self_to_local_zcr +
        (time_challenge_recv - time_response_recv) -
        delay_local_zcr_to_parent_zcr
```

The local ZCR estimates the transit delay between itself and the parent ZCR using the same method as used for session messages;

```
    Delay = ((time_resp_recv - time_challenge_sent) -
         (time_response_sent - time_challenge_recv))/2
```

```
                        1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |V=0|L=2|CMD=0x2|SndSCOP|ZCRSCOP|                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                        Parent ZCR DSI ID                      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                        Local ZCR DSI ID                       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                       time_response_sent                      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                      time_challenge_sent                      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                    time_challenge_received                    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

SndSCOP : 8 bits
    The SCOP value of the administratively scoped region to which this message was sent.

ZCRSCOP : 8 bits
    The SCOP value of the smallest administratively scoped region to which the sender of this message belongs. N.B., SndSCOP will equal ZCRSCOP for normal receivers. Receivers acting as ZCRs will participate in session maintenance for two SCOP regions, their own local region where SndSCOP = ZCRSCOP, and their parent's region where SndSCOP > ZCRSCOP.

Parent ZCR DSI : 32 bits
    The DSI of the parent ZCR sending the response.

Local ZCR DSI : 32 bits
    The DSI of the local ZCR that issued the challenge.

time_response_sent : 32 bits
    Time the response was sent, same format as for the time_sent field in session messages.

time_challenge_sent : 32 bits
    The time_sent field from the ZCR challenge that triggered this reply.

time_challenge_received : 32 bits
    Time the challenge was received by the parent ZCR, formatted in the same manner as time_response_sent.

A.1.5 ZCR Take Over

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=0|L=2|CMD=0x3|SndSCOP|ZCRSCOP|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Parent ZCR DSI ID                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Old Local ZCR DSI ID                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     New Local ZCR DSI ID                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Old Local ZCR to Parent ZCR distance             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              New Local ZCR to Parent ZCR distance             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Parent ZCR DSI : 32 bits
    The DSI of the parent ZCR for the local ZCR.

Old Local ZCR DSI : 32 bits
    The DSI of the receiver that was the local ZCR.

New Local ZCR DSI : 32 bits
    The DSI of the receiver that is to take over from the old local ZCR.

Old Local ZCR to Parent ZCR distance : 32 bits
    The Old Local ZCR's estimate of the distance to the parent ZCR expressed in time and formatted in the same manner as time_sent field for session messages.

New Local ZCR to Parent ZCR distance : 32 bits
    The New Local ZCR's estimate of the distance to the parent ZCR expressed in time and formatted in the same manner as time_sent field for session messages.

A.1.6 Logical Data Stream Description Request

Logical Data Stream Description Requests are issued by receivers in order determine the characteristics of an individual Logical Data Stream as identified by the LDS ID field.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=0|L=0|CMD=0x4|               |    LDS ID     |               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

A.1.7 Logical Data Stream Description

A Logical Data Stream Description provides a complete set of information about a single Logical Data Stream (LDS). The information it contains includes the total length and syntactic format of the Application Data Unit Identifier (ADUI) used, the sampling rate, the sample size (if constant) regular samples are used), the nominal bandwidth for playback at normal real time rates, an indication of whether IPv4 or IPv6 addresses are used, the addresses of the LDS Control channel, the addresses of the partitions associated with this LDS, and a description of the valid ADUI values.

Additional information about the Logical Data Stream can be appended after the mandatory elements in an application specific section of arbitrary length. Examples of such information are the semantic meanings of the various fields within the ADUI, image resolution, the drop priority of the layer for use during receiver driven bandwidth control.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|v=0|L=0|CMD=0x5|  ADUI Length  |     LDS ID    |Num Scop Zones |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| ADUI num dims | ADU hash dims |   ADUI dim 0  |   ADUI dim n  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       ADUI Max Dim Vals                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|S|6|R| Seg_Len |                  Sample Size                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Sampling Rate                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Normal Rate Bandwidth                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Num Segs      | Num Par Ids   |       AppSpecDataLength       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Payload Type  |  Layer Index  | RLM Drop Order|    Reserved   |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                        LDSC IP Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       LDSC Port Number        |      SCOP     |      TTL      |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
:                              ...                              :
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                        LDSC IP Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       LDSC Port Number        |      SCOP     |      TTL      |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                      First Partition Id                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                              ...                              :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       nth Partition Id                        |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|     Seg 0 Start Application Data Unit Identifier (ADUI)       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Seg 0 Stop Application Data Unit Identifier (ADUI)        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                              ...                              :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Seg n Start Application Data Unit Identifier (ADUI)       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Seg n Stop Application Data Unit Identifier (ADUI)        |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                          AppSpecData                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
Num Partitions : 8 bits
    The number of partitions for this object minus one.

ADUI Length : 8 bits
    The length of the Application Data Unit Identifier (ADUI) in bits. The
    number of bits is rounded to a multiple of 32 and the resulting ADUI is
    right justified so that any unused bits appear starting at the first octet
    in the identifier.

ADUI num dims : 8 bits
    The number of dimensions present in the ADUI for this Logical Data Stream
    minus one. Note this figure does not include the DSI of the sender as
    this is automatically assumed.

ADUI hash dims : 8 bits
    The number of dims to be used internally by the cache when generating a
    hash value for storing a particular packet in the RAM cache.

ADUI dim n : 8 bits
    The length in bits required to represent dimension n in the ADUI.
    The zeroth dimension corresponds to the least significant bits in the ADUI
    and dimension lengths are listed in order from the zeroth dimension
    upwards. The sum of all the ADU dim lengths should be equal to the ADUI
    length field.

ADUI Max Dim Vals : variable length
    An ADUI in which each dimension contains the maximum allowable value. For
    example, the ADUI format Hours(5 bits) : Minutes (6 bits) : Seconds (6bits)
    : Bytes (15 bits) the values would be 23:59:59:(2^15-1) or 0xbf9d7fff.

Regular Sized Samples (S) : 1 bit
    When set this bit indicates that size of each ADU in this LDS is
    constant (e.g. 4 bytes for stereo LIN16 audio).

IPv6 Addresses (6) : 1 bit
    When set this bit indicates the IPv6 addresses are used for following
    address elements, if not set then these addresses are IPv4 addresses.
```

Separate Repair Channels (R) : 1 bit
When set this bit indicates that repairs for the various partitions will be sent over a separate multicast group than the original data. The addresses of these groups will follow those identifying the original partition addresses.

Segment Length (Seg_Len) : 8 bits
The number of quad words in an aduI start/stop segment pair.

Sample Size : 24 bits
When the R bit is set the ADU size indicates the size in bytes of a single ADU.

Sample Rate : 32 bits
A fixed point number indicating the number of ADUs per second consumed for single speed real time playback. The Sample Rate is expressed in 1/1000 of a hertz. (e.g. 29970 represents 29.97Hz as would be used for NTSC video)

Normal Rate Bandwidth : 32 bits
The Normal Rate Bandwidth indicates the bandwidth in bits per second required to play a single partition stream of data from this LDS at single speed in real time. Note individual partitions may transmit data at rates that are faster or slower than this rate.

Number of ADUI Segment Defs (Num Segs) : 8 bits
The number of following ADUI segment definitions. Each Segment defines a range of ADUIs that are valid for this LDS. For example, a movie in three acts would have three segment definitions that indicate the running time of each act.

Number of Partition Ids (Num Par Ids) : 8 bits
The number of Partition Identifiers present in the message. The sender of the Logical Data Stream Description should list all known active Partition Ids for the Logical Data Stream being described.

Logical Data Stream Control (LDSC) IP Address : IPv4 32 bits / IPv6 128bits
The multicast IP address of the Logical Data Stream Control channel used for repair requests and indications for this LDS's various partitions.

Logical Data Stream Control (LDSC) Port Number : 16 bits
Port number to be used for the LDSC channel corresponding to the IP address address just given.

SCOP : 8 bits
SCOP value as defined in RFC 1884 for the immediately preceding IP address.

TTL:
Time-To-Live value for the immediately preceding IP address.

Partition Ids : multiple of 32 bits
List of all known active Partition Ids for the Logical Data Stream being described.

```
Segment Start ADUI : nearest multiple of 32, greater than ADUI length bits
    The starting ADUI for a segment.

Segment Stop ADUI : nearest multiple of 32, greater than ADUI length bits
    The stopping ADUI for a segment.

AppSpecDataLength : 16 bits
    The length of the following application specific data in bytes. This
    value includes the two bytes required to represent itself.
```

A.1.8 Partition Description Request

Partition Description Requests are issued by receivers in order determine the characteristics of an individual partition as identified by the Partition ID field.

```
                     1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |V=0|L=2|CMD=0x6|                MBZ            | Max Scope Zone|
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                        Partition Id                           |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

A.1.9 Partition description

A Partition Description provides a complete set of information about a single partition. The information it contains includes the length of ADUI, whether or not the partition loops (repeats), the number of distinct segments to be played, the partition ID, the playing start time, and the start ADUI, stop ADUI, ADUI sub sampling factor, and the relative-to-realtime playrate of each segment to be sequentially played in this partition.

```
                     1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |V=0|L=0|CMD=0x7|  ADUI Length  |L|MBZ| NumSegs |Num Scope Zones|
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                        Partition Id                           |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                Data Source Identifier (DSI)                   |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                             DSI_1                             |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |       Last Time Last Session was sent from DSI_n (LTS)        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |     Last Time Last Session was received from DSI_n (LTR)      |
 +=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 |                      Start Time Seconds                       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                 Start Time Seconds Fraction                   |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                      Stop Time Seconds                        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                 Stop Time Seconds Fraction                    |
 +=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                 LDS 0 Data Channel IP address                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| LDS0 Data Channel Port Number |      SCOP     |      TTL      |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|               LDS 0 Repair Max Scop IP Address                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  LDS 0 Max Scop Port Number   |      SCOP     |      TTL      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                              ...                              :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               LDS 0 Repair Min Scop IP Address                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  LDS 0 Min Scop Port Number   |      SCOP     |      TTL      |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
:                              ...                              :
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                 LDS L Data Channel IP address                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| LDSL Data Channel Port Number |      SCOP     |      TTL      |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|               LDS L Repair Max Scop IP Address                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  LDS L Max Scop Port Number   |      SCOP     |      TTL      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                              ...                              :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               LDS L Repair Min Scop IP Address                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  LDS 0 Min Scop Port Number   |      SCOP     |      TTL      |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|  Seg 0 Length |Seg 0 SubSamp  |         Seg 0 Playrate        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Seg 0 Start Application Data Unit  (ADU)           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Seg 0 Stop Application Data Unit  (ADU)            |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
:                             . . .                             :
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
| Seg N Length  |Seg N SubSamp  |         Seg 0 Playrate        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Seg N Start Application Data Unit  (ADU)           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Seg N Stop Application Data Unit  (ADU)            |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

ADUI Length: 8 bits

The length of the Application Data Unit (ADU) in bits. The number of bits is rounded to a multiple of 32 and the resulting ADUI is left justified so that any unused bits appear at the end of the last octet in the identifier. Note that the ADUI Length for partition descriptions will most likely be smaller than that for the logical data streams within an object. This is because the partition description only describes the common dimensions between each of the logical data streams. LDS specific information such as frame number or audio sample number should be appended to the common information to yield a full ADUI.

Loop Flag: 1 bit
When the loop flag is set the partition repeats over the segments in sequential order.

Number of Segments (NumSegs): 5 bits
The number of segments in the current partition definition minus one. Up to 32 different segment runs can be defined, thereby allowing for some degree of complexity above that of a single linear run.

Number of Administratively scoped zones (Num Scop Zones) : 8 bits
This field indicates how many levels are in the SHARQFEC hierarchy and hence how many repair channels must be listed for each LDS partition entry.

Partition ID: 8 bits
The partition identifier corresponding to the data contained within this packet.

DSI_1 : 32 bits
The DSI of the receiver sending the partition description.

Last Time Last Session was sent from DSI_n (LTS) : 32 bits
The time_sent field extracted from the session message sent by DSI_n as last seen by the sender of the packet (DSI).

Last Time Last Session was received from DSI_n (LTR) : 32 bits
The time that DSI last received a session message from DSI_n. Same format as the timestamp field

Start Time seconds: 32 bits
The first half of a 64 bit NTP timestamp that indicates when the segments defined in this partition commence playing. Segments are played sequentially starting with the first defined segment and will repeat (loop) if the Loop bit is set.

Start Time Seconds Fraction: 32 bits
The second half of the 64 bit NTP timestamp. This 32 bit number is the fractional part of the time stamp.

LDS n Data Channel IP address : IPv4 32 bits / IPv6 128bits
The multicast IP address of the Logical Data Stream Partition data channel for carrying the original data for LDS n.

LDS n Repair IP Addresses : IPV4 32 bits / IPv6 128bits
The multicast IP address of the Logical Data Stream Partition data channel for effecting repairs for this LDS's various partitions.

SCOP : 8 bits
SCOP value as defined in RFC 1884 for the immediately preceding IP address.

TTL:
    Time-To-Live value for the immediately preceding IP address.

Segment Length (Seg Length) : 8 bits
    Length of the current ADUI segment in quad words.

Sub Sample Factor (SubSamp): 8 bits
    An eight bit integer which indicates the degree of any subsampling performed on the data for transmission purposes. For example, a video object which has a partition for a video object that has SubSamp equal to one will contain every frame, whereas one that has SubSamp equal to two will play every other frame.

Playrate : 16 bits
    A positive fixed point number consisting of an 8 bit integer followed by an 8 bit fractional part that indicates the rate at which the data be played relative to that required for single speed real time playback.

Segment Start ADU: nearest multiple of 32 greater than ADUI length bits
    The starting ADUI for a segment.

Segment Stop ADU: nearest multiple of 32 greater than ADUI length bits
    The stopping ADU for a segment.

A.1.10 Session Description Request

When a new receiver or sender joins a session it needs to know the details of the payload characteristics of each logical data stream payloads and the partitioning schemes used upon them. The Session Description Request packet provides the means for new receivers and senders to request this information from the members of the session using the standard SRM mechanism. The corresponding Session Description is multicast on the IOC channel.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=0|L=0|CMD=0x8|                                               |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                             DSI_1                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           time_sent                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

DSI_1 : 32 bits
    The DSI of the receiver sending the partition description.

time_sent : 32 bits
    Time the response was sent, same format as for the time_sent field in a session messages.

A.1.11 Session Description

An Session Description provides a complete description of the object. The description consists of one or more Logical Data Streams Descriptions, and Partition Descriptions that completely describe the object. This header is placed a the beginning of a packet and is followed by a complete set of Logical Data Stream Descriptions and Partition Descriptions for the object.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=0|L=0|CMD=0x9|  ADUI Length  |    Num LDS    |Num Scope Zones|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Person | Deliv | Volat_Persis  |   Bounded Effort Window       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Bounded Delay Window                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Persistence Time                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Session Bandwidth (IOCC only)              |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                    Par ADU Rate (ADU /sec)                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Num Par Ids   | ADUI num dims |   ADUI dim 0  |   ADUI dim n  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Max Dim Application Data Unit (ADU)               |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
:                             . . .                             :
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                   IOCC Max Scop IP Address                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  IOCC  Max Scop Port Number   |      SCOP     |      TTL      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                             . . .                             :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   IOCC Min Scop IP Address                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  IOCC  Min Scop Port Number   |      SCOP     |      TTL      |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                     First Partition Id                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                             . . .                             :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Last  Partition Id                        |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Num LDS: 8 bits
    The number of Logical Data Streams for this object minus one.

Number of Administratively scoped zones (Num Scop Zones) : 8 bits
    This field indicates how many levels are in the SHARQFEC hierarchy and hence how many IOC channels must be listed.

Personness (person) : 4 bits

The personnes value associated with the session. This field may take one of three values:

First Person	0x1
Second Person	0x2
Thrid Person	0x3

Delivery Mode (Deliv) : 4 bits

The type of reliability and latency model to be used when delivering this object's data. This field may take one of the following values

Reliable Non-Real-Time	0x2
Reliable Bounded-Delay	0x6
Reliable Real-Time	0xa
Bounded-Effort Real-Time	0x9
Bounded-Effort Non-Real-Time	0x1
Lossy Real-Time	0x8

Volatility and Persistence Mode (Volat_Persis) : 8 bits

The Volat_Persis field is composed by adding one or more volatility selection and one Persistence selection. Valid volatility selections are:

Recorded	0x00	
Live	0x01	
Live Delayed	0x02	N.B. Only one of Recorded, Live, or Lived Delayed may be chosen
Static	0x04	
Dynamic	0x08	N.B Choose either Static or Dynamic, but not both.

Valid persistence selections are

Transient	0x00
TTL	0x10
This session only	0x20
Indefinite	0x30

An additional field is used to indicate whether the Bounded Delay Window values required for the TTL persistence mode is specified in absolute time or relative to the current time.

Absolute	0x00
Relative	0x40

Bounded-Effort Window : 16 bits

Expressed as a fixed point number in 8bit.8bit format, the bounded-effort window specifies the minimum time receivers should attempt to keep received packets present in their RAM caches after reception before discarding them.

Bounded-Delay Window : 32 bits

The Bounded-Delay window specifies for the maximum time, in seconds, by which all data should be delivered relative to the object's first opening.

Persistence Time : 32 bits

The Persistence time value specifies the time, in seconds, by which an object's data must be purged from the cache. Absolute values are relative to Midnight January 1, 1970, while relative values are made absolute by adding to the current time measured from midnight January 1, 1970.

Par ADU Rate : 32 bits
The rate which the common part ADUs are consumed when playing back at normal speed.

Num Par Ids : 8 bits
Number of partition identifiers included later in this message

ADUI num dims

Number of dimensions in the common part ADUI format.

ADUI dim n : 8 bits
The length in bits required to represent dimension n in the common part ADUI. The common part ADUI is created from the maximum number of dimensions that are the same for all LDSs within the object. The zeroth dimension corresponds to the least significant bits in the ADUI and dimension lengths are listed in order from the zeroth dimension upwards. The sum of all the ADU dim lengths should be equal to the ADUI length field.

ADUI Max Dim Vals : variable length
An ADUI in which each dimension contains the maximum allowable value. For example, the ADUI format Hours(5 bits) : Minutes (6 bits) : Seconds (6bits) : Bytes (15 bits) the values would be 23:59:59:(2^15-1) or 0xbf9d7fff.

IOC n IP Addresses : IPV4 32 bits / IPv6 128bits
The multicast IP address of the Intra Object Control channel for each valid scop.

SCOP : 8 bits
SCOP value as defined in RFC 1884 for the immediately preceding IP address.

TTL:
Time-To-Live value for the immediately preceding IP address.

A.1.12 Sender Description Request

When a receiver or new sender joins a session it will want to know some details about the sender beyond those related to Logical Data Stream and Partitions. This message provides a means for the new session member to query individual senders for such information as their location, name, and contact information of someone for times when service difficulties are experienced.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=0|L=2|CMD=0xa|                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Data Source Identifier (DSI)                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

DSI_1 : 32 bits
The DSI of the receiver sending the session description request

A.1.13 Sender Description

A Sender Description provides a set of useful details about a sender that are not specifically related to Logical Data Streams or Partitioning. This information is not included in the Session Description as multiple senders may contribute to single session, not all of which may be of interest to a receiver. The format of the data following the header below is a subset of that used by the Session Description Protocol, specifically only the following items are implemented with (modifications noted in []). For a complete description of SDP see the latest Internet Draft.

```
Session Description Part
  v=  (SDP protocol version)
  o=  (Origin)
  s=  (Session Name)
  i=* (Session Information)
  u=* (URI of description)
  e=* (Email address)
  p=* (Phone Number)
  [the b=(bandwidth) item not included as this is LDS/Partition dependent]
  One or more Time Descriptions
  z=* (Time Zone adjustments)
  k=* (Encryption Key)
  a=* (zero or more session attribute lines)
  Zero or more Media Descriptions
Time Description Part
  t=  (time the session is active)
  r=* (zero or more repeat times)

Media Description Part
  m=  (media name)
      [ m= <media> <LDS ID> <LDS group> <format>]
  i=* (Media Title)
  [the b=(bandwidth) item not included as this is LDS/Partition dependent]
  k=* (Encryption Key)
  a=* (zero or more media attribute lines)
```

```
                    1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=0|L=0|CMD=0xb|               |      Description Length       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Data Source Identifier (DSI)                    |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                      SDP Sender Description                   |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

A.2 HPRTP Level One (Logical Data Stream Control) Packet Formats

A.2.1 Level One Fixed Header Format

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=0|L=1|  CMD  |   ADUI Length |     LDS ID    |               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Sender's Data Source Identifier (DSI)           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            time_sent                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
Version (V): 2 bits
    Identifies the HPRTP version, zero (0).

Level (L): 2 bits
     This field identifies the level in the hierarchy of a particular packet,
     for the LDSC level this is set to zero (0).

Command (CMD): 4 bits
    The LDSC command which may take one of the following values
      0x0 Single NACK
      0x1 Sequential NACK
      0x2 Application Specific NACK
      0x3 Single Repair Indication
      0x4 Sequential Repair Indication
      0x5 Application Specific Repair Indication

ADUI Length: 8 bits
    The length of the Application Data Unit Identifier (ADUI) in bits. The
    number of bits is rounded to a multiple of 32 and the resulting ADUI is
    right justified so that any unused bits appear starting at the first octet
    in the identifier.

LDS ID: 8 bits
    The logical data stream identifier corresponding to the data contained
    within this packet.

Sender's DSI: 32 bits
    Data Source Identifier of the session member sending the packet.

time_sent: 32 bits
    The middle 32 bits of a 64 bit NTP timestamp generated by the receiver
    just prior to sending the Session message. The NTP timestamp format is
    defined in rfc2030, it consists of a 64 bit fixed point number consisting
    of a 32 bit integer in the first 32 bits and a 32 bit fractional part in
    the second 32 bits. Thus the 32 bit timestamp used here consists of a
    16 bit integer in the first 16 bits and a 16 bit fractional part in the
    second 16 bits and has an accuracy of 1/65536 second.
```

A.2.2 Sequential ARQ NACK

The sequential NACK message is used by receivers to indicate that they have missed reception of a small number of sequential Payload Data messages. It is important to note that due to the expensive nature of repairs that result from long runs of missing Payload Data messages the Sequential NACK should be used sparingly. For very large runs the sequential NACK should be replaced with dynamic partitioning (where available) or passively waiting for appropriate partitions to loop.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=0|L=1|CMD=0x0|  ADUI Length  |     LDS ID    |               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Sender's Data Source Identifier (DSI)           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            timestamp                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                NACK Data Source Identifier (DSI)              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       NACK Partition Id                       |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|     NACK Start Application Data Unit Identifier (ADUI)        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     NACK Stop Application Data Unit Identifier (ADUI)         |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

```
NACK Data Source Identifier (DSI): 32 bits
    DSI of the data being requested.

NACK Partition Id: 32 bits
    Partition to which the subsequent repairs should be sent.

NACK Start Application Data Unit Identifier (ADUI): variable length
    ADUI of the first missing ADU.

NACK Stop Application Data Unit Identifier (ADUI): variable length
    ADUI of the last missing ADU.
```

A.2.3 SHARQFEC NACK

The SHARQFEC NACK message is used by receivers to indicate that they require one or more repairs to complete a particular SHARQFEC packet group.

```
                    1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |V=0|L=1|CMD=0x1|   ADUI Length |      LDS ID     |       |NumRTTs|
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |              Sender's Data Source Identifier (DSI)            |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                           timestamp                           |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |              NACK Data Source Identifier (DSI)                |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                      NACK Partition Id                        |
 +=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 |              Top ZCR Data Source Identifier (DSI)             |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                          RTT to ZCR                           |
 +=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 :                             . . .                             :
 +=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 |                ZCR Data Source Identifier (DSI)               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                          RTT to ZCR                           |
 +=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 | FEC_Group_Size|  Reps Needed  |      LLC      |  MAX_PKT_ID   |
 +=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 | FEC Group Start Application Data Unit Identifier (ADUI)       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | FEC Group Stop Application Data Unit Identifier (ADUI)        |
 +=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

NACK Data Source Identifier (DSI): 32 bits
    DSI of the data being requested.

NACK Partition Id: 32 bits
    Partition to which the subsequent repairs should be sent.

ZCR Data Source Identifier (DSI): 32 bits
    DSI of the ZCR at the implied scope.

RTT to ZCR: 32 bits
    One way propagation delay to the ZCR listed from the next smallest ZCR. At the smallest scope the delay is measured between the local ZCR and the receiver sending the NACK. The RTT is expressed as the middle 32 bits of a 64 bit NNTP time stamp.

FEC_Group_Size: 8 bits
    Number of original data packets per FEC group. Each packet contains last_dim_modulo ADU's worth of data as specified in the LDS description.

Reps Needed: 8 bits
    The number of packets the NACK's sender needs to complete the group.

LLC: 8 bits
    The NACK Sender's Local Loss Count (LLC) for the group.

MAX_PKT_ID: 8 bits
    The maximum FEC packet id seen so far by the NACK's sender. Original packets are numbered 0 through (FEC_Group_Size - 1).

FEC Group Start Application Data Unit Identifier (ADUI)
    ADUI of the first ADU in the group.

FEC Group Stop Application Data Unit Identifier (ADUI)
    ADUI of the last ADU in the group. N.B. this is usually equal to Start ADUI + plus (FEC_Group_Size*last_dim_modulo) - 1.

A.2.4 Application Specific NACK

In cases where neither Single nor Sequential NACKs are appropriate the application may wish to use its own NACK, this message provides a means for this to occur.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=0|L=1|CMD=0x2|   ADUI Length |      LDS ID   |       |NACKcnt|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Sender's Data Source Identifier (DSI)           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            timestamp                          |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|             First NACK Data Source Identifier (DSI)           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     First NACK Partition Id                   |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
| AppSpecLength                 |                               |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
:                               . . .                           :
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|             Last NACK Data Source Identifier (DSI)            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Last NACK Partition Id                    |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
| AppSpecLength                 |                               |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

NACKcnt: 8 bits
    The number of application specific NACKS in this message.

NACK Data Source Identifier (DSI): 32 bits
    DSI of the data being requested.

```
NACK Partition Id: 32 bits
    Partition to which the subsequent repairs should be sent.

AppSpecLength: 16 bits
    Length of the application specific NACK description that follows including
    the two bytes required for the AppSpecLength field itself.
```

A.2.5 Sequential ARQ Repair Indication

```
The Sequential ARQ Repair Indication is used by repairers to inform other
potential repairers to suppress their repairs. It consists of one or more ADUI
ranges that specify the repairs the repairer will make.

                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=0|L=1|CMD=0x3|   ADUI Length |      LDS ID     |       | INDcnt|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Sender's Data Source Identifier (DSI)          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            timestamp                          |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|             First IND Data Source Identifier (DSI)            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     First IND  Partition Id                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| First IND Start Application Data Unit Identifier (ADUI)      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| First IND Stop Application Data Unit Identifier (ADUI)        |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
:                              . . .                            :
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|              Last IND Data Source Identifier (DSI)            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Last IND Partition Id                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Last IND Start Application Data Unit Identifier (ADUI)    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Last IND Stop Application Data Unit Identifier (ADUI)     |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+


INDcnt: 8 bits
    The number of repair indications included in this message.

IND Data Source Identifier (DSI): 32 bits
    DSI of the data being repaired.

IND Partition Id: 32 bits
    Partition to which the repairs will be sent.
```

IND Start Application Data Unit Identifier (ADUI): variable length
    ADUI of the first missing ADU.

IND Stop Application Data Unit Identifier (ADUI): variable length
    ADUI of the last missing ADU.

A.2.6 SHARQFEC Ind

The sequential NACK message is used by receivers to indicate that they have missed reception of a small number of sequential Payload Data messages. It is important to note that due to the expensive nature of repairs that result from long runs of missing Payload Data messages the Sequential NACK should be used sparingly. For very large runs the sequential NACK should be replaced with dynamic partitioning (where available) or passively waiting for appropriate partitions to loop.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=0|L=1|CMD=0x4|  ADUI Length  |    LDS ID     |       |NumRTTs|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Sender's Data Source Identifier (DSI)            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           timestamp                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                IND Data Source Identifier (DSI)               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       IND Partition Id                        |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|              Top ZCR Data Source Identifier (DSI)             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           RTT to ZCR                          |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
:                               . . .                           :
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|             Local ZCR Data Source Identifier (DSI)            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           RTT to ZCR                          |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
| FEC_Group_Size|  Reps Needed  |      ZLC      |  MAX_PKT_ID   |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
| FEC Group Start Application Data Unit Identifier (ADUI)       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| FEC Group Stop Application Data Unit Identifier (ADUI)        |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

IND Data Source Identifier (DSI): 32 bits
    DSI of the data being requested.

IND Partition Id: 32 bits
    Partition to which the subsequent repairs should be sent.

ZCR Data Source Identifier (DSI): 32 bits
    DSI of the ZCR at the implied scope.

```
RTT to ZCR: 32 bits
    One way propagation delay to the ZCR listed from the next smallest ZCR.
    At the smallest scope the delay is measured between the local ZCR and the
    receiver sening the NACK. The RTT is expressed as the middle 32 bits
    of a 64 bit NNTP timestamp.

FEC_Group_Size: 8 bits
    Number of original data packets per FEC group. Each packet contains
    last_dim_modulo ADU's worth of data as specified in the LDS description.

Reps Needed: 8 bits
    The number of packets the IND's sender will send.

ZLC: 8 bits
    The Sender's estimate of the Zone Loss Count (ZLC) for the group at the
    scope to which the message is sent.

MAX_PKT_ID: 8 bits
    The maximum FEC packet id that will result after all the repairs have
    been sent by this sender.

FEC Group Start Application Data Unit Identifier (ADUI)
    ADUI of the first ADU in the group.

FEC Group Stop Application Data Unit Identifier (ADUI)
    ADUI of the last ADU in the group. N.B., this is usually equal to
    Start ADUI + plus (FEC_Group_Size*last_dim_modulo) - 1.
```

A.2.7 Application Specific Repair Indication

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=0|L=1|CMD=0x5|   ADUI Length |      LDS ID     |     |IND Cnt|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Sender's Data Source Identifier (DSI)            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           timestamp                           |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|            First IND Data Source Identifier (DSI)             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     First IND Partition Id                    |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
| AppSpecLength                 |                               |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
:                              . . .                            :
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|            Last IND Data Source Identifier (DSI)              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Last IND Partition Id                     |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
| AppSpecLength                 |                               |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

```
INDcnt: 8 bits
    The number of application specific repair indications in this message.

IND Data Source Identifier (DSI): 32 bits
    DSI of the data being repaired.

IND Partition Id: 32 bits
    Partition to which the subsequent repairs will be sent.

AppSpecLength: 16 bits
    Length of the application specific IND description that follows including
    the two bytes required for the AppSpecLength field itself.
```

A.3 HPRTP Level Two (Payload) Packet Formats

A.3.1 HPRTP Level Two Fixed Header Fields

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=0|L=2|  CMD  |0|B|F|StartBits|     LDS ID    |0|R|0|StopBits |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Data Source Identifier (DSI)                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Partition Id                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
version (V): 2 bits
    This field identifies the version of the HPRTP. The current version is
    zero (0).

Level (L): 2 bits
     This field identifies the level in the hierarchy of a particular packet,
     for the payload or data level this is set to zero (0).

Command (CMD): 4 bits
    The Payload command which may take one of the following values
      0x0 Payload Data
      0x1 SHARQFEC Payload Data
      0x2 Heartbeat

Must Be Zero (MBZ): 3 bits
    These three bits must be set to zero in this version of the protocol, they
    may be used in future versions.

StartBits: 5 bits
    Number of bits that should be ignored at the beginning of the first data
    word.

LDS ID: 8 bits
    The logical data stream identifier corresponding to the data contained
    within this packet.

byte offset last dim adu (B): 1 bit
    When set this bit indicate that the last dim in the ADUI is a byte offset
    and that the maximum byte offset field should be included.

repair response (R): 1 bit
    If the repair response bit is set, then the data contained in this packet
    was generated in response to a repair request.

StopBits: 5 bits
    Number of bits that should be ignored at the end of the last data
    word.

partition ID: 8 bits
    The partition identifier corresponding to the data contained within this
    packet.
```

Data Source Identifier (DSI): 32 bits
    The DSI field identifies the synchronization source. This identifier is chosen randomly, with the intent that no two synchronization sources within the same HPRTP session will have the same DSI identifier. This field is semantically and syntactically identical to the DSI identifier described within the RTP protocol.

A.3.2 Payload Data

This packet actually carries the data, whether it be audio, video, text, game events, or something else entirely different. Payload Data messages may not be combined with other messages.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=0|L=2|CMD=0x0|0|B|F|StartBits|      LDS ID     |0|R|X|StopBits |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Data Source Identifier (DSI)                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Partition Id                           |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                    Maximum Byte Offset                        |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|       First Application Data Unit Identifier (ADUI)           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Last Application Data Unit Identifier (ADUI)           |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                        Payload Data                           |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Maximum Byte Offset: 32 bits (optional, B bit must be set)
    Indicates that maximum byte offset for an ADU that has been segmented over several packets.

First Application Data Unit Identifier (ADUI): 32m bits
    The First ADUI uniquely identifies the start of the portion of the object contained in this packet. Its length is variable and defined in the LDSC layer, however for transport purposes its length is rounded up to a multiple of 32 bits.

Last Application Data Unit Identifier (ADUI): 32m bits
    Indicates the last ADU contained in this packet.

Payload Data: variable length
    Application formatted data of an arbitrary length.

A.3.3 SHARQFEC Payload Data

This packet actually carries the data, whether it be audio, video, text, game events, or something else entirely different. Payload Data messages may not be combined with other messages.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=0|L=2|CMD=0x1|P|B|F|StartBits|     LDS ID    |0|R|X|StopBits |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Data Source Identifier (DSI)                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Partition Id                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| FEC Group Size|  this_pkt_id  |      LLC      |  max_pkt_id   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        FEC Group Start Application Data Unit (ADU)            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        FEC Group Stop Application Data Unit (ADU)             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
|      MBZ      |P|B|F|StartBits|     LDS ID    |0|R|X|StopBits |   FEC
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+ applied
|                      Maximum Byte Offset                      | only to
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+ data below
|        First Application Data Unit Identifier (ADUI)          | this line
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Last Application Data Unit Identifier (ADUI)          |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                         Payload Data                          |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

FEC Group Size : 8 bits
    The number of original packets in the FEC group to which this packet belongs.

this_pkt_id : 8 bits
    An identifier used to distinguish between the packets within an FEC group. Original packets will have this_pkt_id < FEC Group Size, and will contain unencoded data, redundant packets will have this_pkt_id >= FEC Group Size and may be used to reconstruct any original packets.

Local Loss Count (LLC) 8 bits
    The number of missing original packets experienced by the session member that sent this packet. This value is used for suppression purposes and maybe set to a non zero value by the originating source it is preemptively sending redundant FEC packets without having received NACKs.

max_pkt_id: 8 bits
    Indicates the highest pkt_id so used so far. Future repairs made in response to a NACK should generate FEC packets with ids greater than this value.

Maximum Byte Offset: 32 bits (optional, B bit must be set)
    Indicates that maximum byte offset for an ADU that has been segmented over several packets.

First Application Data Unit Identifier (ADU): 32m bits
    The First ADUI uniquely identifies the start of the portion of the object contained in this packet. Its length is variable and defined in the LDSC layer, however for transport purposes its length is rounded up to a multiple of 32 bits.

Last Application Data Unit Identifier (ADU): 32m bits
    Indicates the last ADU contained in this packet.

Payload Data: variable length
    Application formatted data of an arbitrary length.

A.3.4 Heartbeat

The Heartbeat message provides a means for detecting whether or not the last Payload Data packet has been lost in open ended sessions.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=0|L=2|CMD=0x1|P|T|B|StartBits|     LDS ID    | MBZ |StopBits |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Data Source Identifier (DSI)                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Partition Id                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Maximum Byte Offset                      |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|             Application Data Unit Identifier (ADUI)           |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Appendix B

HPRTP Reference Implementation

B.1 Overview of the Server Implementation

The HPRTP server comes in two parts: a multithreaded main core, and a disk_cache_handler. Communication between applications, the main core and the disk cache handler, is achieved through the exchange of shared memory objects and unnamed pipes. More than one application can connect to the server at a given time allowing separate applications to access the same material. A diagram showing how these pieces relate to one another is shown below.

```
Local Host                                      Remote Host(s)

 +-----------------+
 | Application(s)  |
 +-----------------+
        ^ |
        | |
 .......|.|.........................
 .      | V                         .
 . +-----------+    +------------+  .
 . | Main Core |<==>| Disk Cache |  .       +-----------------+
 . +-----------+    |   Handler  |  .       | Application(s)  |
 .      ^ |         +------------+  .       +-----------------+
 .      | V              ^ |        .             ^ |
 . +-----------+         | V        .             | V
 . |  UDP/IP   |        /----\      .       ..................
 . +-----------+        |Disk|      .       .                .
 .      ^ |             \____/      .       .  HPRTP Server  .
 .......|.|.........................        ..................
        | |                                       ^ |
        | V                                       | V
 ---------------------------------------------------------
                        Internet
```

The main server core is responsible for the following functionality

* Network access, including traffic shaping.
* Multicast Address assignment
* Shared Memory Management
* Object Management
* Session Announcement, Resolution, and Management

The Disk Cache Handler is responsible for shuttling data between the RAM cache and the Disk Cache. On systems that support kernel threading it is simply another thread within the Main Core. On systems that do not support kernel threading the Disk Cache Handler is implemented as a separate process controlled by the main core.

B.2 Server Installation

B.2.1 File setup

Installing the server on a unix system requires the creation of a directory for the Disk Cache. This directory is named

```
/work/hprtp
```

and should only be writable by the HPRTP server processes.

B.2.2 Execution

The main server should be started before the disk cache handler process. Currently the main server is started by executing

```
% simple_test_server
```

At this stage the server will print out

```
"Start disk_handler_process manually now"
```

and one should then start the disk cache handler process by executing

```
% hprtp_server_disk_cache_handler
```

Both server processes should now be running.

B.2.3 Server Connection State

When stared, the HPRTP server will create another directory automatically named

```
/tmp/.HPRTP-unix/
```

This directory is used to store a named pipe for each process that attaches to the server, including the hprtp_disk_cache_handler. The names of these pipes follow the following convention:

```
HPRTP0      - named pipe to be written to by connecting client applications
HPRTP<pid> - named pipe for connected process with pid <pid>
HPRTPcm0    - named pipe for main core to connect to the disk cache handler
HPRTPcm<core pid> - named pipe for core server to communicate with
                    disk cache handler process
```

The use of named pipes allows the server to detect when an application quits unexpectedly. Should this occur then the server will close any objects left open by the application, but only if no other application is accessing them.

In order to prevent the existence of multiple shared memory segments between different runs, the server also drops a cookie at

```
/tmp/.HPRTP-server-cookie
```

that lists the name and size of the shared memory segment used along with the id for the semaphore group used internally within the server.

B.3 Server Side Debugging

The test_simple_server program includes a text driven interface that allows querying of its current state. Currently the valid commands are

```
d{d,l,r} num_pkts obj_id lds_id [par_id] Drop a data, ldsc, or repr packet
a/A                                       Adu Manager Status
t                                         Thread Manager Status
e                                         Event Scheduler Status
k/K                                       Socket Manager Status
o/O                                       Object Manager Status
r/R                                       Ram Cache Manager Status
c/C                                       Cache Manager Status
s                                         Session Manager Status
D xxxxxxxx                                Display Session announcement
S                                         HPRTP Server Status
m/M                                       Shared Memory Status
l                                         Check Lock state of semaphores
                                          and mutexs
x/X/q/Q                                   Exit/Quit
```

This menu will be printed out if 'h' or an illegal command is entered.

Issuing one of the Exit/Quit commands will also result in the termination of the disk cache handler process.

No debugging interface is provide for the disk cache handler process.

B.4 C library API

Applications communicate with the local server via the 22 function calls listed below. Some functions are common to both sending and receiving applications while others are sender or receiver specific.

```
Common methods
  hprtp_attach_server

  hprtp_open_object
  hprtp_close_object

  hprtp_join_lds
  hprtp_leave_lds
  hprtp_join_par
  hprtp_leave_par

  hprtp_get_recv_hints
  hprtp_set_recv_hints
  hprtp_get_send_hints
  hprtp_set_send_hints

  hprtp_get_current_active_sessions
  hprtp_update_sess_info
  hprtp_get_current_cached_objects

Sender Specific methods

  hprtp_sender_init_info
  hprtp_sender_init_lds
  hprtp_sender_init_par

  hprtp_sender_update_cache_strategy
  hprtp_write

Receiver specific methods
  hprtp_receiver_init
  hprtp_receiver_update_cache_strategy

  hprtp_read
```