# On Scalable Internet Multimedia Conferencing Systems

Mark Handley

November 14, 1997

# Abstract

In this thesis I examine scaling aspects of IP-multicast based multimedia conferencing systems. The thesis is that application level semantics must be used in protocol design to cope with various forms of failure, and that these systems should be designed to permit inconsistencies in order to scale.

I present an examination of the network conditions that such applications must face as motivation, and examine application designs for two aspects of multimedia conferencing as validation. These applications are a distributed shared editor and a distributed session directory.

The general design principle of Application Level Framing (ALF) was applied during the design of these applications. I show that ALF can and should be applied in a wider context than that stated in the original ALF paper from Clark and Tennenhouse, and that it results in applications that perform well and are very robust to a wide variety of conditions. However it can also lead to designs that are difficult to generalise from.

The design methodology of lightweight sessions as proposed by Van Jacobson and based on IP multicast presents a large design space in which very few points have been mapped. This thesis explores some of this design space.

In the chapter on shared editors, I examine the effects of designing for robustness and redundancy in shared tools, and conclude that solutions resulting from such design perform well but are very specific to the design task and cannot easily be abstracted. In the chapter on session directories I examine the scaling limits of lightweight sessions through the example of a session directory which must scale at least as well as the sessions it describes, and examine the scaling limits of multicast address allocation schemes.

In conclusion I reflect on the contradictory design goals in conferencing applications; those of producing abstraction layers to allow reuse of code and simplify the design task, and of designing for good performance in distributed over unreliable networks and attempt to draw some general guidelines for the design of such systems.

# Acknowledgements

My thanks go to Peter Kirstein, Angela Sasse, Steve Wilbur, Allison Mankin and Deborah Estrin for their encouragement, and for allowing me time to work on the ideas in this thesis when I should really have been doing other things. I would also like to thank Derek McAuley and Tim Kindberg for their feedback which has greatly improved the final version of this thesis.

Most especially my thanks are due to Jon Crowcroft for his many insightful comments, suggestions and intriguing discussions, particularly those in the Crown and Anchor, and to Maria Dahl for everything.

# Contents

# List of Figures

# Chapter 1

# Introduction

In this thesis I examine scaling aspects of the design of IP multicast-based multimedia conferencing systems. IP multicast provides a data distribution mechanism that can scale to extremely large groups, but using multicast does not by itself result in scalable conferencing systems. There are many additional obstacles that need to be overcome before this goal can be achieved. Many of these problems are caused by the nature of the internet itself, which exhibits a large degree of heterogeneity, in particular in terms of available bandwidth, packet loss rates and other failures. In attempting to solve some of these problems, the thesis is that *application level semantics* must be used in protocol design to cope with various forms of failure, and in particular conferencing systems should be *designed to permit inconsistencies* in order to scale in the presence of such network problems.

We provide a history of conferencing systems as background information in chapter 2, and present some aspects of the multicast environment and some existing systems in chapter 3 that are useful to help understand the chapters that follow.

In chapter 4 we present an analysis of the performance and failures that are observed by applications wishing to use IP multicast today. Several problems are observed, including a high degree of spatially uncorrelated packet loss, and many receivers that occasionally experience times when extremely high loss rates are observed. These failures force us to conclude that to build scalable conferencing systems, communications problems should be expected and often should be tolerated to the extent that systems are designed to maintain near normal operation in spite of such failures, and any resultant problems this causes should be resolved afterwards.

In chapters 5 and 6, systems were designed and implemented which exhibit such behaviour to examine whether such designs are feasible in practice.

Chapter 5 presents the design of a multicast based shared text editor. This illustrates the use of application level semantics to cope with the network problems identified in chapter 4. Although the scaling requirements of a shared editor are somewhat limited, the principles of using application-level redundancy to cope with uncorrelated packet loss and tolerating inconsistencies to cope with temporary network partitioning are shown to be feasible for such an interactive tool.

Chapter 6 addresses the issues of a session directory tool that must perform scalable multicast address allocation. The scaling issues here are much greater than with the shared editor, and the design goal is to allocate and distribute multicast addresses in a decentralised manner without causing duplicate allocations within the same scope region. Such a system should hope to scale towards millions of participating systems. We discover that to perform this task adequately, the address allocation algorithm critically depends on the properties of the distribution mechanism and the scoping properties of the network. Taking into account the network conditions observed in chapter 4, we show, by designing the distribution protocol based on the limitations of the allocation algorithm, and designing with inconsistencies in mind, that this does result in an address allocation mechanism that scales well. However some extreme scaling limitations are also observed, and an even more scalable future solution is suggested.

## 1.1 IP Multicast: A Framework for Solutions

IP multicast provides efficient many-to-many data distribution in an internet environment. It is easy to view IP multicast as simply an optimisation for data distribution, and indeed this is the case, but IP multicast can also result in a different way of thinking about application design. To see why this might be the case, examine the IP multicast service model, as described by Van Jacobson[51]:

- Senders just send to *the group*
- Receivers express an interest in *the group*
- Routers conspire to deliver data from senders to receivers

With IP multicast, the group is indirectly identified by a single IP class-D multicast address.

Several things are important about this service model from an architectural point of view. Receivers do not need to know who or where the senders are to receive traffic from them. Senders never *need* to know who the receivers are. Neither senders or receivers need care about the network topology as the network optimises delivery.

An IP multicast group is scalable because information about group membership and group changes at the IP level are kept local to routers near the relevant members. How this is performed depends on the particular multicast routing scheme in use local to the member, and although it is not a trivial task, several solutions do exist and therefore multicast routing will not be discussed in detail here. As technical background, a brief introduction to multicast routing is given in chapter 3. For more detailed information on multicast routing, see [22][18][15][2][61]. Typically, as a group with $s$ senders and $r$ receivers increases in size, state in routers scales $O(s)$ or $O(1)$ depending on the routing scheme in use. This state may be in on-tree routers for newer so called sparse-mode algorithms such as PIM, or in off-tree routers for older so-called dense-mode algorithms such as DVMRP. Thus the most scalable current multicast routing algorithms require $O(1)$ state in on-tree routers, and hence the total routing state scales $O(g)$ in a router that is on-tree for $g$ groups. We can also envisage multicast routing schemes which require less than $O(g)$ state[1], but the requirement is not currently urgent, so none of these are currently implemented.

The level of indirection introduced by the IP class D address denominating the group solves the distributed systems binding problem, by pushing this task down into routing; given a multicast address (and UDP port), a host can send a message to the members of a group without needing to discover who they are. Similarly receivers can “tune in” to multicast data sources without needing to bother the data source itself with any form of request.

IP multicast is a natural solution for multi-party conferencing because of the efficiency of the data distribution trees, with data being replicated in the network at appropriate points rather than in end-systems. It also avoids the need to configure special-purpose servers to support the session, which require support, and which cause traffic concentration and can be a bottleneck. For larger broadcast-style sessions, it is essential that data-replication be carried out in a way that only requires per-receiver network-state to be local to each receiver, and that data-replication occurs within the network. Attempting to configure a tree of application-specific replication servers for such broadcasts rapidly becomes a “multicast routing” problem, and thus native multicast support is a more appropriate solution.

IP multicast groups are symmetric, in that there is no server/client split or multi-point control unit. Thus as all instances of an application in principle are equal, the design of multicast-based communication solutions starts from an unusual place. If a server were required to bootstrap com-

[1] with IP encapsulation, not all on-tree routers need hold the state for a group whose traffic they are forwarding - traffic for the group can be encapsulated (either unicast of multicast) between on-tree routers nearer the edge of the network, reducing some of the state burden on backbone routers

munication, it would be a natural design philosophy to continue this asymmetry in higher levels of the design. Multicast solutions do not start with this asymmetry, and start with the ability to communicate to all group members equally. Although this does not dictate that multicast based applications employ a symmetric, peer-to-peer architecture, it makes this much more natural than it would otherwise be.

However, it should also be noted that multicast can also make some problems harder as some semantics which were clear-cut with only two parties become ill-defined in the multi-party case, and common mode failures must be addressed which can be dealt with separately in the multiple-unicast flows case.

## 1.2 The problems of IP multicast

In chapter 4 we will examine the behaviour of multicast data traffic on the internet in an attempt to discover the sort of problems with which multicast-based applications must cope. The intention is to get a feel for several parameters which may be of importance to application designers:

- How heterogeneous is the Internet in terms of observed congestion?
- To what extent is multicast packet loss spatially correlated?
- How reliable or predictable is the network behaviour?

If the internet was fairly homogeneous this would imply that designing for typical behaviour would be reasonable. If packet loss was caused by a very few bottlenecks, then there would be a good degree of spatial loss correlation, and data missing at one site would likely be missing at others too. If the network was generally reliable, then designing systems that assume continuous connectivity to all sites might be feasible.

Unfortunately the network appears to be very heterogeneous, packet loss is not well spatially correlated, and the network is neither very reliable or very predictable - short temporary dropouts are fairly commonplace.

For live audio and video transmission, layered codings[2] provide a potential solution for heterogeneity, and spatially uncorrelated loss is not a problem. However, temporary dropouts do cause a problem which is only likely to be solved in the network rather than the applications.

[2] compressed data is split into multiple layers consisting of a base layer and a number of enhancement layers. Each layer is sent to a different multicast group, and receivers can choose an appropriate number of layers depending on their observed network conditions

For applications requiring some degree of reliable data distribution, the problems are somewhat different. Coping with heterogeneity is more difficult because unlike audio and video, quality can rarely be decreased. Often the only available tradeoff is between duration of transmission and bandwidth. Uncorrelated packet loss presents a particular problem for applications requiring reliability, because different receivers are likely to be missing different data, which can result in very high retransmission rates if traditional techniques are used. Temporary dropouts also present a special problem if total consistency is required or the missing participants hold any critical state, as the application now has to choose between stopping to wait for the partition to resolve itself, or allowing potential inconsistencies to occur and then solving any resulting problems when communication is restored.

Clearly not all reliable-multicast applications can continue to perform useful work with arriving data while waiting for missing packets to arrive, but with good protocol design, many shared applications and control protocols can do so. In this thesis, we postulate that to do this effectively, we must apply application level semantics to network protocol design, and design the applications to tolerate, discover and resolve inconsistencies as they occur.

There are many steps along this path of application and protocol design. The first few steps can be expressed in generic terms without referring to specific applications, but beyond these lies a great deal of potential for designing protocols and applications that perform well by taking advantage of particular properties of the application. Application Level Framing and the Announce/Listen Model are two general design principles, and we discuss them in the remainder of this chapter. Chapter 5 and particularly chapter 6 examine how we can use application-level semantics to design protocols that perform and scale well in spite of the network problems we observe, and provide validation of the feasibility of these design principles.

## 1.3 Application Level Framing

Up until 1990, and still today to a large extent, most discussions of protocol design revolved around the ISO reference model[44] or similar models that are structured around layering to decompose communication protocols into modules. However, in reviewing requirements for future networks, Clark and Tennenhouse[10] identified problems with such a layered approach for new classes of application including audio and video transmission. They argue that the traditional concept of presentation and transport layers which provide a reliable abstraction over an unreliable network is inappropriate for many of these new protocols. They present two new design principles; Application

Level Framing (ALF) and Integrated Layer Processing (ILP). ILP is not of direct relevance to this thesis, but ALF is of importance.

The key argument underlying ALF is that there are many different ways an application might be able to cope with misordered or lost packets. These range from ignoring the loss, to resending the missing data (either from a buffer or by regenerating it), and to sending new data which supersedes the missing data. The application only has this choice if transport protocol is dealing with data in "Application Data Units" (ADUs). An ADU contains data that can be processed out-of-order with respect to other ADUs. Thus the ADU is the minimum unit of error recovery.

The key property of a transport protocol for ADUs is that each ADU contains sufficient information to be processed by the receiver immediately. An example is a video stream, wherein the compressed video data in an ADU must be capable of being decompressed regardless of whether previous ADUs have been received. Additionally the ADU must contain "header" information detailing its position in the video image and the frame from which it came.

Although an ADU need not be a packet, there are many applications for which a packet is a natural ADU. Such ALF applications have the great advantage that all packets that are received can be processed by the application immediately. However, for such applications, many of the functions of traditional transport and presentation layers have largely been moved up into the application. Thus the application is then solely responsible for reliability.

### 1.3.1 ALF and Multicast

Although IP multicast predated ALF by a few years, the original ALF paper does not mention multicast at all. However, whereas ALF is useful as a principle for some unicast protocols, it becomes much more important with multicast applications. With a unicast transport protocol, performance is dictated by the properties of a single path, and a single sender and receiver; with multicast, performance cannot be so simply characterised. In a unicast connection, data is either received or not received, and the protocol typically concerns itself with communicating this information to the sender in a timely manner. With multicast, even talking about a "connection" is not appropriate. Different receivers may have received different data, and if all the receivers try to inform the sender of their state at once, an acknowledgment implosion is likely with the sender overloaded with responses.

Thus the idea of sending data in ADUs, capable of being processed out of order becomes much more important due to heterogeneity of network paths and receivers. With a large group, feedback must be delayed to prevent response implosions. Network performance varies widely. Any multicast protocol that doesn't allow inconsistencies between receivers whilst still being able to perform useful work is likely to achieve performance that at best tracks the performance of the worst receiver at any time, and typically gets even worse as the session size increases. The ability to cope with incomplete data, whilst, if required, designing to ensure that missing data is eventually filled in, is critical to the design of scalable multicast protocols. Layered protocol models help little here, as only the application has sufficient information to be able to decide which data can be dealt with and which cannot, and to cope with the concept that, due to the interaction between network heterogeneity and application policy, failure is not a binary function, and appropriately designed application-level protocols may perform useful work in the presence of many forms of failure. In a multicast context, multiplexing and reliability both become application level issues.

An example of multiplexing taking place at the application level is the Real-Time transport Protocol (RTP)[68] which was designed with ALF in mind. RTP provides a packet header for real-time data that includes a timestamp to uniquely identify the ADU, a payload type field to demultiplex different coding schemes, and a source identifier to identify a source independently of the IP source address. Thus RTP sessions cope with loss (up to codec-dependent limits), cope with heterogeneity of codecs for whatever reason, and permit application-level relays to adapt data streams to heterogeneous network conditions. However, RTP is an unreliable stream protocol, and in this thesis we will concentrate on applications requiring reliable and interactive behaviour.

## 1.4 Scaling and Robustness

For a multicast-based system to scale, basic data distribution is not an issue, as IP multicast provides us with a solution that scales well to very large groups, but in practice reliable data distribution is still an issue due to packet loss and other network failures.

From a data distribution point of view, spatially uncorrelated packet loss causes a significant problem which will require an increase in the bandwidth used for retransmission of missing data as the size of the group grows even if no particular site is suffering a very high loss rate. Thus multicast-based systems requiring reliable transmission must address this issue in order to scale.

From an application point of view, a more difficult problem presents itself. Many applications including those discussed in this thesis have a data transmission that is driven by external events. In the event of loss or network partitioning, the application may require to continue to perform useful work as it may not be able to trade delay for reliability. This is particularly an issue as such systems scale, because the probability of there being a failure significant enough to affect the application increases as the group size increases. The precise relationship between group size and probability of experiencing a failure is clearly a complex one, but the expected number of link failures per unit time in a network connecting $n$ active sites is likely be between $O(n)$ (star topology) and $O(n^2)$ (full mesh). Applications that can pause while such failures are resolved or routed around will suffer decreasing performance as the session scales, and applications that cannot pause suffer from a decreased probability of success as the session scales. For some scales and some applications, this may be acceptable, but applications where this is not the case cannot be designed around a network protocol designed to maintain consistency.

## 1.5 General Techniques

Whilst protocols that attempt to maintain consistency tend to be based around acknowledgments, those that allow consistency to be relaxed are better based around negative acknowledgments (NACKs) and the announce/listen model.

Although ACK-based protocols might be made to scale to moderate sizes for some problems by using a tree-based architecture[55], in general it is easier to make NACK-based protocols scale. However, as we saw above, even NACK-based protocols can scale badly in the face of spatially uncorrelated packet loss. Application level redundancy can improve these scaling properties as we will show in chapter 5, but there will still be network failures such as temporary partitionings that will cause problems.

If we design applications to continue working in the presence of such failures then we permit inconsistencies to arise. Multicast-based applications that do this then need to have a mechanism to detect such inconsistencies and resolve them. The announce/listen mechanisms are trivial techniques that are possible with IP multicast. An application simply announces its state periodically, and listens to similar announcements from other instances of the application. In this way inconsistencies can be discovered.

An important aspect of the announce/listen model is that an application does not need to know anything about the other listeners. It simply assumes that there might be inconsistencies that need to be resolved. Typically each participant's announcement rate is controlled so that the total announcement rate from all participants is a constant.

As applications increase in scale, the probability of inconsistencies increases. At very large scales, the probability of inconsistency is high enough that it is not worthwhile requiring receivers that are missing date to send a NACK; the sender can just as easily repeat the original data and save a packet exchange.

Thus announce/listen mechanisms can be used in two ways:

- At smaller scales data-state summaries can be sent to allow inconsistencies to be discovered and resolved using NACK-based retransmission techniques.
- At large scales sending summaries is not worthwhile because the probability of inconsistency is so high, and simply repeating the data periodically makes a better use of bandwidth.

We use the former technique with the shared editor in chapter 5 and the latter for multicast address allocation in chapter 6.

## 1.6 Application Level Semantics and Relaxed Consistency

Application Level Framing suggests that to achieve good performance in the face of packet loss, data should be framed in units that are meaningful to the applications and so can be processed out-of-order. We believe this is necessary for multicast applications, but for many applications it is not sufficient.

To design scalable and robust multicast conferencing applications there are two additional requirements: that packet loss is not simply repaired by requesting retransmission of missing data, and that applications should be designed to cope with failure by permitting consistency to be relaxed.

The retransmission issue can be dealt with using packet-level forward error correction (FEC)[63], or by exploiting redundancy inherent in the application. For bulk-data transfer applications, packet-level FEC mechanisms are probably appropriate, but for shared tools application-level redundancy can provide a more timely repair. We illustrate this in chapter 5.

Relaxing consistency can be a trivial issue for many applications, where it simply means some sites are allowed to temporarily get behind others. However, for applications where one site's change to the data set affects changes made by others, relaxing consistency is not so simple. Inevitably, relaxing consistency in this latter class of applications involves application semantics. In the shared editor we use constraints imposed by properties of a text editor to restrict the operations that may be made on the data set so that when a consistent data-set is reached it is likely to be one that is meaningful. In the case of multicast address allocation, the scale is large enough and the data set is fluid enough that total consistency can probably never be achieved. As a result we are forced to work with probabilistic solutions based on the application requirements and network constraints to the problem of allocating addresses in a consistent and clash-free manner.

We do not claim that these techniques are feasible for all reliable multicast applications, but they can be applied to many problem areas, and seem particularly suitable for applications that are used as a communication tool, where humans may prefer to tolerate occasional and temporary inconsistencies rather than experience unacceptable delays or interruptions to communication.

# Chapter 2

# Historical Background

There are many different ways to build multimedia conferencing systems, and over the last ten years many different approaches have been tried. Few have achieved any real popularity, in part because of restrictive design and in part because of high bandwidth costs. Bandwidth costs are now becoming acceptable, and there has been a rush by commercial software developers to standardise multimedia conferencing systems. To understand the evolution of the emerging standards, and why they are often sub-optimal, it is useful to consider a potted history of conferencing systems. The history presented here is of course incomplete, as the purpose for presenting it is to illustrate certain points and trends.

## 2.1 General Trends in Multimedia Conferencing Architectures

The most significant general trend of the last twenty years has been the move from using switched circuits for multimedia data towards using packet-switched networks and the Internet in particular for multimedia data. The development of ATM attempted to create a middle ground, whereby virtual circuits could support traditional circuit-switched applications and also support best-effort traffic effectively. It now looks like ATM will never become sufficiently ubiquitous to provide *end-to-end* services, and so it has failed to capture this middle ground.

With packet-switched networks themselves, IP multicast has been the most significant extension to the internet architecture.[1] The internet was not designed to handle real-time traffic, although the first experiments carrying audio were as early as 1973[11], and there have been several attempts to modify the architecture to better handle streaming data. The trend here has been away from complex

[1] TCP Congestion Control is probably the most significant extension to internet protocols, but doesn't change the architecture itself.

hard-state protocols such as the internet streams protocol *ST*[28], to the less complex hard-state *ST-II*[72], and eventually to the development of *RSVP*[8], which is soft-state and (at least in principle) simple as it decouples reservation from the actual service models.

The trend has been towards simplicity and fault tolerance. Many people including the author believe that even RSVP is undesirable for per-flow state, and that simpler mechanisms that do not provide per-flow guarantees but instead provide increased priority services (typically at increased cost) on an aggregate basis are more appropriate as real-time internet traffic increases.

The table below gives a potted history of some of the milestones in multimedia conferencing development concentrating mostly on systems that related to the internet. The remainder of this chapter discusses each of these systems briefly to provide context for later chapters, and to illustrate where some systems architectures went wrong with respect to scaling issues.

| Dates | Conferencing System or Component |
|---|---|
| 1973 | Experiments with packet voice on the ARPAnet. |
| 1977 | Network Voice Protocol (for voice over IP) standardised. |
| 1985-1988 | IP multicast development. |
| 1986-1992 | Evolution of ISDN networking standards. |
| 1988-1991 | CAR Multimedia Conferencing System at UCL. |
| 1990 | videoconferencing on DARTnet using ST-II and circuit-oriented codecs. |
| 1988-1990 | MMconf conference control system. |
| 1991 | IP Multicast deployed on DARTnet. |
| 1990-1991 | ITU H.320 ISDN conferencing standard. |
| 1992 | Vat multicast-based audio-conferencing tool released. |
| 1992 | Formation of the Mbone (IP Multicast Backbone). |
| 1992 | First IETF live audio multicast. |
| 1992 | Sd session directory developed at LBL. |
| 1992 | IVS multicast-based videoconferencing tool released by INRIA. |
| 1992 | NV multicast-based videoconferencing tool released by Xerox PARC. |
| 1992-1993 | MICE Conferencing Management and Multiplexing Centre. |
| 1993 | CU-SeeMe unicast IP-based conferencing tool released by Cornell. |
| 1993 | MMCC conference control tool released by ISI. |
| 1993 | LBL multicast-based shared whiteboard released. |
| 1993 | IETF attempts development of an agreement protocol. |
| 1993-1996 | ITU T.120 family “data” conferencing standards. |
| 1994 | Conference Control Channel Protocol developed at UCL. |
| 1994 | NTE multicast-based shared editor developed at UCL. |
| 1994-1995 | vic multicast-based videoconferencing tool released. |
| 1995 | sdr multicast session directory released by UCL. |
| 1992-1996 | Development of IETF Real-time Transport Protocol (RTP). |
| 1994-1997 | Development of IETF Resource Reservation Protocol (RSVP) |
| 1995-1997 | IETF drafts of standards for session directories. |
| 1995-1997 | ITU H.323 conferencing standard for packet networks. |

### Network Voice Protocol (NVP)

The earliest work on packet "multimedia" probably took place on the ARPAnet in 1973. This led to the standardization of the Network Voice Protocol (NVP) in 1977. NVP included a call setup and close-down mechanism, data transport and two audio encodings. Control and data are separated, but the protocol intrinsically assumes two-way communication, both for control and data.

### H.320

The network technologies chosen have always heavily influenced the design of multimedia conferencing systems. H.320[45] typifies early conferencing systems. It uses point-to-point ISDN channels to support two-way calls, and later, permitted the use of a multi-point control unit (MCU) to support small multi-party conferences where participants can dial into the MCU. This is essentially telephone-style technology extended to support H.261 video[47] and a small range of audio codings. There is no concept of shared tools, although a raw bit-pipe is available within the serial-line framing protocol.

### The CAR Conferencing System

At the same time as H.320 was being developed, the CAR[2][36] project was developing a multi-party multimedia conferencing system. CAR also used ISDN channels, but used one channel for audio/video data and the second ISDN channel for IP data. A centralised conference control protocol ran over this secondary channel permitting joining and leaving the session, video switching, floor control, and the introduction and control of shared applications. These applications included existing X-window based tools shared using Shared-X and a shared sketchpad which was a dedicated shared application using a centralised TCP-based protocol.

CAR was much more advanced than an H.320 MCU, but suffered badly from the unstable nature of ISDN services and terminal equipment available at the time. In particular, its centralised architecture, and its use of a TCP-based remote procedure call (RPC) mechanism (ANSAware) meant that it did not deal gracefully with failure. The use of an RPC mechanism influenced the system design to the extent that reasoning about failure and the resulting inconsistencies was difficult, and so mechanisms to cope with failure tended to be add-ons rather than part of the fundamental system design.

[2] CAD/CAM for the Automotive industry under RACE (Research for Advanced Communications in Europe)

Shared-X also proved to be a lesson in how not to design a collaboration mechanism. It consisted of a “bridge” which acted as an X-server to the client application to be shared, and mapped all the X resources and identifiers for connections to multiple real X-servers. In addition to causing traffic concentration, this centralised model attempts to keep all X-servers in step (although not precise lockstep), and thus the delays through the system tended to sum, and the performance on a local ethernet or dedicated ISDN tended to be unacceptable with more than 4 participants. Although Shared-X did cope with the failure of a site, it locked up all participants displays for around 90 seconds whilst it timed out the failed connection.

The lessons to be learned from this were that complex system designs need to consider failure from the outset, and that permitting inconsistency and repairing problems later is the desirable course of action from the users' point of view. Degrading everyone's service because of one problematic site is not normally acceptable in a communication system used between humans. In addition, the floor control system in CAR, which controlled access to the shared tools and the video channel, caused interactions to be unnatural. We were seeking alternatives to this floor control mechanism when IP multicast appeared on the scene and changed everything.

Years later, almost exactly the same functionality supplied by the CAR system resurfaced in the T.120 protocol standard. As BT Labs were instrumental in the development of T.120, and were a partner in the CAR project, we have always wondered why the lessons of CAR failed to be learned.

### The MBone

During 1991, IP multicast was deployed on the DARTnet testbed network. This spurred the development of the *vat*, *nevot* and *vt* audio conferencing tools. Although the principle behind these tools is similar, vat was most widely accepted and used, principally because its user interface was vastly superior to the other tools. In spring 1992, the DARTnet's native multicast network was extended to cover a small number of additional sites by tunneling IP multicast across parts of the internet without native multicast support. As temporary measure, the Mbone (Multicast Backbone) was put together to allow reception of live audio from the IETF meeting in San Diego. Forty subnetworks in 4 countries and three continents were able to receive audio and talk back to the meeting. Although the audio quality was poor[3], the principle had been demonstrated, and sufficient interest shown that the Mbone was not taken down again.

Initially, I do not believe there was any philosophy of “lightweight sessions”. To build a mul-

[3] due in part to the use of the IP loose-source-route option for tunneling

ticast audio tool, the minimum amount of code simply takes data from the audio device (in native u-law format), packetises it with sufficient header to be able to recover timing at the receiver, has an adaptive playout buffer at the receiver to allow network jitter to be removed before playout, and performs receiver-based mixing if multiple people speak. Silence suppression helps to cope with mismatched audio device clocks. To see who the receivers are, simply multicasting a session message periodically is the simplest possible solution given that there is no server to perform this task. Thus when setting out to design such a tool, something approximating to lightweight sessions emerges as a first working prototype. At this point, if one comes from a telco/circuit-switched background, the next step would be to integrate the tool into a complete conference control framework. We tried this with the MICE project at UCL and abandoned the effort. Later, the ITU went down this track with the H.323 standard. The unusual step that the LBL group made at this point was to decide that such conference control mechanisms were not necessary for the most part, and it was from this decision that the lightweight sessions philosophy emerged.

### Sd

*Sd* was the original multicast session directory developed by Van Jacobson at LBL. It performed two tasks - multicast address allocation, and announcement of multicast sessions to communicate which sessions will take place and their multicast addresses and ports.

At the time sd was developed, several other proposals were being floated for mechanisms to organise multicast sessions. One of these involved a hierarchy of conference groups organised like Usenet-news groups by category. However, such a hierarchy merely translates a known organisation into the new medium, and would have missed the point - multicast sessions are mostly event-organised rather than category-organised.

Instead of attempting to organise multicast sessions, sd adopted the same announce/listen mechanism used by vat, and simply periodically announced session descriptions. Thus sd was essentially a session advertisement tool, and implemented a server-less session directory that relied entirely on soft-sate; state that is not refreshed continually is eventually timed out.

### The MICE Conference Management and Multiplexing Centre

By early 1993, two multicast based video-conferencing tools had been developed - IVS from INRIA and nv from Xerox PARC. Both of these achieved only very low frame rates (2-4 fps) with workstations available at that time. However, hardware video codecs were available that achieved

30 fps. Although it was clear that eventually software codecs would be able to achieve similar frame-rates, at that time only hardware codecs could achieve good quality video, and so we set out to build a conference management and multiplexing centre (CMMC) to permit the inter-working of H.261[47] hardware codecs with the H.261-based IVS conferencing tool, and the interconnection of ISDN-based H.320 conferencing streams with IP-multicast based sessions. The design of the CMMC is discussed in [35], although the details are not relevant here. A subset of this functionality was later standardised in H.320/H.323 gateways.

The CMMC was not continued beyond the first implementation because it became clear that better software codecs would make the difficult to control hardware codecs irrelevant, especially for multi-way sessions where software codecs can easily decode multiple video streams whereas hardware codecs could generally only decode a single stream and hence required the multiplexing centre. In addition, H.320 failed to create a huge market, primarily because of equipment and line costs. It was rare that we needed to provide H.320 gateway functionality, and when we did, the lack of shared tools in H.320 videophones made it a poor substitute for IP connectivity.

Apart from the expense, the CMMC also suffered from the problems that it required a control protocol to marshal the hardware codecs and ISDN connections, whereas multicast software-only participants only needed to join the session. As such hardware codecs are a limited resource, a booking mechanism was also needed, which is not needed for software-codecs. In the end, the very nature of such a centralised resource makes it undesirable to use when the gain over software-only codecs is not sufficiently great.

For group meetings we began to realise that many low-rate video streams were more valuable than one (or a few) higher quality streams. A sense of presence was often all that was required, and being able to see all the participants was useful and integrated the group, whereas seeing just the chair adversely affected the group dynamics and made the participants less equal. However, bandwidth is usually limited, and so even multicast-based lightweight video sessions could benefit from simple opportunistic conference control mechanisms that move a fixed bandwidth budget between the different participants as they become (vocally) active, or in a more formal setting, and the chair chooses.

There is still a need for gateways to ISDN H.320 videophones, but these gateways do not need to be a multiplexing *centre*. Instead, they can simply be a multicast participant, and receive information (hints) from any informative conference control system being used, or simply infer reasonable video-switching behaviour from the audio stream.

It was very much these revised requirements that led to the development of the Conference

Control Channel[38].

### CU-See-Me

Around the same time as nv and IVS were being released, Cornell University developed the CU-See-Me videoconferencing tool, originally only for the Apple Macintosh and with no audio support. As the Macintosh did not have IP multicast support, CU-See-Me took a more traditional approach and developed a multipoint server that CU-See-Me clients could connect to. The server would then relay multiple streams to each receiver. In part, the lack of multicast support was turned to their advantage, because Mbone connectivity was (and still is to some extent) difficult to arrange, whereas anyone with a Mac could run CU-See-Me and anyone with a Sun workstation could run the server providing they had sufficient bandwidth to cope with the traffic concentration. CU-See-Me also incorporated a rudimentary congestion control mechanism whereby a server can provide feedback that reduces the data rate from a source, and a receiver can choose to receive only a subset of the session participants if insufficient bandwidth is available.

Aside from this feedback mechanism and server-based access control, CU-See-Me also does not require an explicit conference control protocol.

### MMCC

Although sd allowed multicast conferences to be advertised, it did not allow one user to call another to participate in a session. IVS provided a rudimentary mechanism to call another IVS user, but being tied to a single tool, this was rarely used. MMCC was the first multimedia conference control system that implemented a call model for multicast sessions. It did not provide any form of control during a session, but did allow sessions to be started, new users invited, and clean session termination. It fitted in well with the Mbone lightweight sessions as it was distributed, but seems to have fallen into disuse primarily because the community simply was in the habit of announcing all sessions through sdr. Announcing all sessions was considered “politically correct” as this allowed sd's address allocation mechanism to work and provided a social restriction on too much simultaneous use of the Mbone which was useful when the Mbone was restricted to 512Kbps and mrouted did not implement traffic pruning. As the Mbone has started to grow up, the need for such a invitation scheme has returned, and now we have both H.323 for tightly coupled sessions and SIP[42] for any multicast session which perform this task.

### The LBL whiteboard and SRM

In many ways the LBL shared whiteboard (wb) best typifies an application *designed* to be a lightweight sessions application. Whilst for audio and video, IP multicast itself tends to imply that a minimal conferencing tool initially evolves along the lightweight sessions lines, there are many more ways to build shared tools than there are to build live-media tools. Wb stays with the symmetric model of lightweight sessions, in which all sites are equal with respect to the distribution and storage of data.

Wb implements a distributed data model, where each site attempts to hold a complete copy of the data in the whiteboard. The reliable multicast protocol at the heart of wb was later abstracted as *Scalable Reliable Multicast* (SRM)[24]. Although SRM is not in principle a point-to-multipoint protocol, the way it is used in wb in effect implements multiple point-to-multipoint data flows at the application level. Thus although many users can all draw simultaneously, each user's drawing on the whiteboard can only be modified by that user. This prevents wb having to solve complex consistency issues that emerge when an object created by one user can be modified by other users.

SRM typifies lightweight sessions as a design philosophy because its symmetric design is a less obvious a way to design a shared whiteboard than multicast audio or video tools. When data is transmitted, it may be lost in transmission. Any site that discovers it is missing data can request retransmission of that data by multicasting a retransmission request. Any site with the missing data can multicast that data to those sites missing it. All sites effectively implement a replicated database, with each site attempting to have a complete copy of all of the data. A site missing data may not be the only site missing that data, and so it delays making the request for a short while in order to give others the chance to do so. As all requests are multicast, one site making such a request can suppress other sites so that only one request is made. A similar mechanism is used to suppress multiple responses. This mechanism is described in more detail in section 3.1.

In chapter 5 I explore the design of a shared text editor. Although the problem seems similar, the need to exploit redundancy (see chapter 4) and the need to solve distributed consistency problems meant that a totally symmetric solution such as the SRM model did not seem so appropriate, although some of the SRM mechanisms could have been used.

### The MMUSIC Agreement Protocol

An interesting effort to generalise conference control functionality was made in the IETF Multiparty Multimedia Session Control (MMUSIC) WG by Eve Schooler, Abel Weinrib and Scott Schenker, by designing an *Agreement Protocol*. The general idea was that not all state in a session is equal, and policy regarding such state varies from session to session. The agreement protocol was a general purpose way of expressing constraints on the consistency of such state so that policies could be imposed. An example might be that leaving a session requires that at least one person remaining in the session be informed, joining the session requires a majority of members to agree, and a change of floor requires only the current floor holder to agree. The agreement protocol could exchange this state and perform handshaking of this data to ensure sufficient consistency that the policy is satisfied without understanding the semantics of that state.

The down-side of such an abstraction layer is that the semantics of the state are represented incompletely by such a protocol, and so the API to the protocol starts to become complex in order to specify the desired behaviour when the consistency requirements *cannot* be satisfied. However, by making the requirements explicit, the agreement protocol does provide an interesting bridge between traditional layered protocols and ALF style protocols. In practice though the agreement protocol suffered from its generality in that it didn't seem to provide a specific solution to conferencing problems, and so more specific and less flexible protocols have been standardised by the ITU.

### T.120

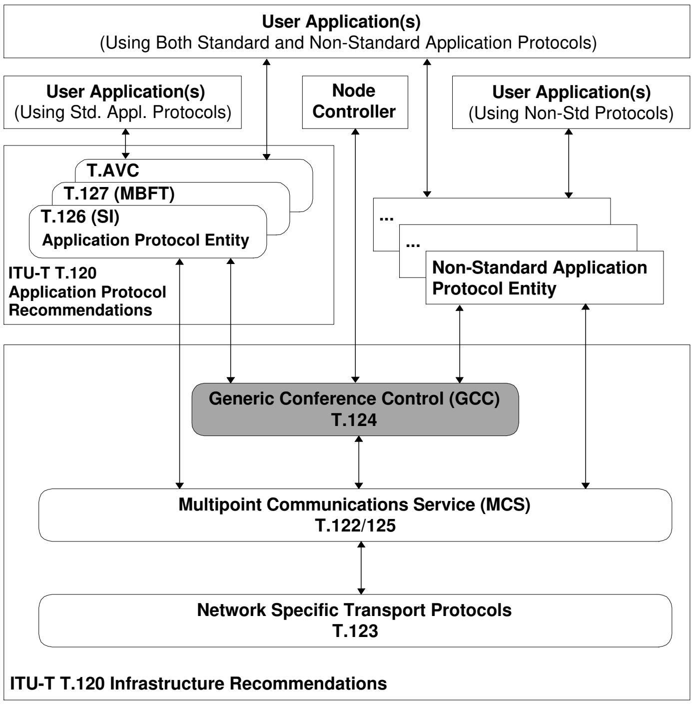


Figure 2.1: T.120 protocol layered protocol architecture

The ITU's T.120 family of protocols provide multi-party conference control and limited shared applications including a whiteboard. T.120 is designed in a layered manner, as shown in figure 2.1, and assumes a reliable multi-point communications service as the transport layer. Such a transport layer is relatively simple to implement over circuit-switched ISDN links with one or more MCU's arranged in a hierarchy. However, implementing such a generic reliable multi-way transport protocol over a best-effort IP service in a manner that can be guaranteed to perform acceptably is extremely difficult. In particular, assuming MCS can achieve total consistency within very short time-scales is unlikely to be realisable, and the T.120 upper-layer protocols are not designed to cope with failure semantics other than the unambiguous failure of a link. Thus T.120 is simply not suitable for use on the public internet without some manner of service guarantee from the network, and even then, an active keep-alive mechanism will be required to rapidly detect silent link failures. It is clear that such a protocol will not scale to session sizes of more than a few participants without suffering unacceptable performance problems, or without being redesigned to explicitly cope with inconsistencies.

### NTE

The Network Text Editor (NTE) was first released in 1994, and is presented in detail in chapter 5. Like wb, it adopts a distributed data model, and follows a lightweight sessions philosophy. However, almost all of the mechanisms it uses to achieve this are different from the mechanisms used in wb. They have in common the use of an announce/listen model to convey current state summaries, and the use of a receiver driven multicast scheme. NTE was developed based on experience with using wb, but without knowledge of how wb worked internally. Interestingly, using the same general design philosophies does not necessarily lead to the same specific solutions.

### SDR

Sdr is a session directory tool which started out in 1994 as a clone of the LBL sd session directory. Sdr evolved beyond sd to become more general purpose and to include session invitation mechanisms as well as session announcement mechanisms. Sdr has now replaced sd and MMCC as the multicast session directory in use on the Mbone and as a conference invitation tool. Some of the design decisions in the evolution of sdr are discussed in chapter 6.

### RTP

RTP is the IETF's Real-time Transport Protocol. The name is somewhat confusing as it isn't a transport protocol in the traditional sense, and it typifies the ALF approach to protocol design. RTP is primarily a packet header format for the transport of multimedia data using UDP[4]. It provides timestamp and payload type information, along with conventions for how to use them. In addition, the Real-time Transport Control Protocol (RTCP) provides approximate membership information and reception feedback using session messages. RTP cannot be used effectively without RTCP. RTP and RTCP are described in a little more detail in section 3.2.

RTP typifies the ALF principle because the payload formats designed to be used with RTP conform to the principle of packetising data in application data units. RTP provides no reliability or retransmission mechanism - it is generally assumed that some low level of packet loss is tolerable for real-time streams such as audio and video so long as each packet is idempotent and thus the payload format allows continued decoding after a missing packet.

When used stand-alone over IP multicast, RTP also typifies the lightweight sessions style of conferencing, as no explicit conference control mechanism is required for a session to proceed. RTCP's session messages then provide membership information, and IP multicast serves to provide joining and data distribution functionality.

RTP was finally standardised in 1996, but it has its roots in the early work done using NVP2 with *vat*, *vt* and *nevot* in 1991, which in turn has its roots in the NVP experiences in the early 1970s.

### H.323

H.323 was standardised in 1996, and represents an extension of the ITU's original H.320 ISDN conferencing standard to work over packet-switched LANs. For transport, H.323 uses RTP and so it does not require an MCU for data traffic. However it was constrained by the H.320 model of a multimedia call, and so the point-to-point conference control model is adopted directly from the H.320 model except that conference control uses TCP for reliable transport.

It is easy to see why this is attractive for the ITU - an H.323 terminal can interwork with an H.320 terminal through a gateway as the call semantics are the same. However, by being constrained by an architecture that is not relevant for packet-switched networks, H.323 represents an inflexible architecture which cannot scale, and is likely to perform poorly in the wide area internet without

[4] other lower layer “packet” transports such as AAL5 can also be used with RTP, but RTP over UDP is the most common usage mode.

significant modifications.

## 2.2 Conferencing System Components

The different multimedia conferencing systems devised all contain approximately the same components:

- audio and video codecs
- a transport protocol for audio and video
- shared tools such as a whiteboard
- a mechanism for inviting members
- mechanisms for joining and leaving the conference
- a mechanism for determining and controlling membership
- mechanisms for controlling access to channels of communication
- mechanisms for controlling access to the underlying network resources

For tightly coupled conferencing such as H.323 and T.120, most of these have now been mapped out in great detail[5].

Within the lightweight sessions model, there is a great deal of flexibility in how to instantiate these components. Audio and video transport are now relatively mature with the standardisation of RTP, but the other components are still relatively under-explored.

For shared applications, the only tools currently available are wb and NTE. SRM, which underlies wb, is described briefly in section 3.1, whilst NTE is presented in detail in chapter 5. These two applications use different mechanisms and are not all that similar, but there is still a great deal of unknown territory in this area. In the conclusions I reflect on how a broader range of applications could be supported using common mechanisms.

Channel access control is not needed in an internet environment, although enhanced quality of service may be desirable in some circumstances. This is largely orthogonal to issues concerning the conferencing architectures. However a key part of any architecture that uses IP multicast is multicast address allocation. Although this does not have to be tied to multimedia conference control in any

[5] some might say “excruciating detail”, although their shared tools are still very immature

way, we inherited a model from sd in which this was the case. Thus announcements of multimedia sessions also serve to reserve the multicast addresses used. Whether this is a good idea or not, and the scaling properties of such a scheme are subjects of some debate and have not previously be analysed. These are examined in detail in chapter 6.

A protocol for inviting users to participate in lightweight sessions is also currently under development. The Session Initiation Protocol attempts to preserve the lightweight sessions model by separating the session initiation phase from participation in the lightweight session itself.

Between these different pieces of work we are now filling in the pieces of the lightweight sessions architecture for internet multimedia conferencing. Much work still remains to be done, but it is hoped that what was previously unexplored territory now has some signposts to indicate feasible solutions.

# Chapter 3

# Technical Background

Several topics occur throughout this thesis and are either built upon or affect aspects of the behaviour of the designs considered. This chapter provides a brief technical introduction to several of these issues that are referenced frequently in later chapters. All these topics are described in detail in the references.

## 3.1 Scalable Reliable Multicast

SRM is a generalisation of the underlying protocol that was implemented in the *wb* whiteboard[53] developed at LBL by Van Jacobson, Steve McCanne and Sally Floyd. At several points in this thesis mechanisms will be compared to SRM and so a brief explanation is in order.

The general idea is that each instance of wb maintains a copy of the complete data set. Any site can make additions to the dataset by contributing new drawing operations (drawops). A drawop is not an object (which may have many versions over time) as such, but an specific version of an object. There is a single separate drawop sequence space for each participant. A drawop may contribute a new object, or may delete an existing object and replace it with a new version of that object, or it may permanently delete an object. As there is a separate drawop sequence space per participant, only the originator of an object can modify or delete that object.

Each site maintains a complete list of all the drawops performed at all sites. Drawops are only deleted from the list when they are superseded by a drawop that deletes the old object and replaces it with a new instance of the same object. This drawop sequencing makes it very easy to detect missing drawops due to packet loss. It also makes summarising each site's current state a trivial operation - all a site need advertise is the latest drawop that it generated via a session message, and other sites can easily see if there is a mismatch between the latest drawop they saw and the one being advertised.

When a site detects that it is missing a drawop, it wishes to send a retransmission request. However, if all receivers missing such a drawop do this, their requests will be synchronised, and a NACK implosion will result. Instead, each receiver sets a retransmission request timer to a value drawn randomly from a uniform interval $[D_1..D_2]$. If this timer expired, the retransmission request is sent. However, if another site's timer expires first, that site will request the retransmission, and our site will receive the request. If this happens, it doubles the original value of its timer and waits. Eventually either a retransmission will be received, in which case it cancels its timer, or its own timer expires and it sends its own request. If the values of $D_1$ and $D_2$ are chosen carefully, approximately one receiver will respond for each packet loss, and this response will suppress other receivers.

SRM is also symmetric in who can respond to a retransmission request. Because of the simple drawop model, all sites can know unambiguously whether or not they can respond to a retransmission request. Any site which has can retransmit the requested information sets a timer in a similar way to those requesting the data, and the site whose timer expires first is the one that sends the data and because the repair is multicast, this suppresses other potential respondees.

In practice, ensuring that only one request is sent and only one reply is forthcoming is not so easy. Either a very long randomisation interval is required, or additional information is needed to make the decision more deterministic. Although the wb implementation does not do so, the descriptions of SRM in the literature propose that sites calculate the delay matrix from each site to all other active sites using timestamped session messages. These end-to-end delays are then used to determine the values of $D_1$ and $D_2$[1] so that the further a site is from the source or from the requester, the longer the delay. This helps ensure that the closest site to the packet drop requests the retransmission and the nearest site to the requester performs the retransmission.

SRM is not the only reliable multicast scheme that has been proposed, but the principles behind it are more general than many, and can be used in many circumstances. Randomised timers are used in a similar manner to SRM in the address allocation mechanism for the session directory in chapter 6. However this mechanism is not used in the NTE shared editor for a range of reasons that are explained in chapter 5. In part, it is more difficult to determine unambiguously in NTE whether or not a site should send the repair, and so the originator of a change is a better choice than in wb. Also

[1] The SRM paper uses the interval $[C_1 d_{s,A}, (C_1 + C_2) d_{s,A}]$ for requests and $[C_1 d_{A,B}, (C_1 + C_2) d_{A,B}]$ for responses where $d_{x,y}$ is host $y$'s estimate of the one-way delay from host $x$, and $s$ is the original source, $A$ is the requester and $B$ is the responder. Throughout this thesis, a slightly different notation is used as the delay matrix is often unavailable to us.

NTE relies heavily on the natural redundancy of its data model, and so it requires a retransmission scheme that is more sender-controlled than that of SRM.

## 3.2 Real-time Transport Protocol

The Real-time Transport Protocol (RTP) is the standard for transmitting so-called real-time data such as audio and video across the internet. It was explicitly designed with multicast in mind based on prior experience with multicast conferencing tools such as nevot[67] and vat[52], and is essentially a packet header format for multimedia data, along with conventions for using that format and an associated "control" protocol. RTP typically uses UDP as its "transport" mechanism so on a best-effort IP network, packet loss is to be expected. RTP provides no additional reliability and assumes that low levels of loss are acceptable for audio and video applications.

The RTP data packet header contains the following:

- A packet sequence number to discover packet loss and re-ordering.
- A timestamp to allow the original intra-stream timing relationship of the data to be recovered at the receiver.
- A payload type field to identify the media encoding used.
- A source identifier to allow the source to be identified even after a transcoder or relay.

The source identifier is a 32 bit randomly assigned integer and the timestamp is in data units (such as samples for audio), so to recover information about who the source is and what real-time the timestamps correspond to, additional information is required. This additional information is provided by the Real-time Transport Control Protocol (RTCP), which consists of session messages sent periodically to the same destination (multicast group) as the data. Receivers may also send RTCP packets to identify themselves and to provide feedback on the quality of reception, and so RTCP provides an approximation of membership information for each media using the announce/listen model. RTCP packets may contain any of the following:

- Information to identify the participant (name, email address, etc) and associate them with the source identifier in the data packets.
- Sender reports that associate the data timestamps with the sender's real-time clock to permit inter-stream synchronisation.

- Receiver reports to provide feedback as to how each sender is being received in terms of packet loss and network jitter.

As explained in chapter 1, the rate at which these RTCP session messages are sent depends on the number of other participants so that the total session message rate is a small constant proportion of the data rate. As a session scales, feedback information from each receiver is received less often, although the total information rate is a constant. The interval is also modified by a random amount each time for each receiver to prevent synchronisation of session messages.

In chapter 4 the information from RTCP receiver reports for video streams is used along with information from the multicast routing tracing program *mtrace* to build up a picture of what happens to multicast traffic in the MBone.

## 3.3 Multicast Routing

Several different multicast routing protocols exist and are in use on the Mbone today. Although all these protocols implement an IP multicast service that conforms to the service model described in chapter 1, they differ in how they do this, and these differences may affect multicast applications to some degree.

Multicast routing schemes can be loosely divided into those that provide *shortest path trees* and those that provide *shared trees*.

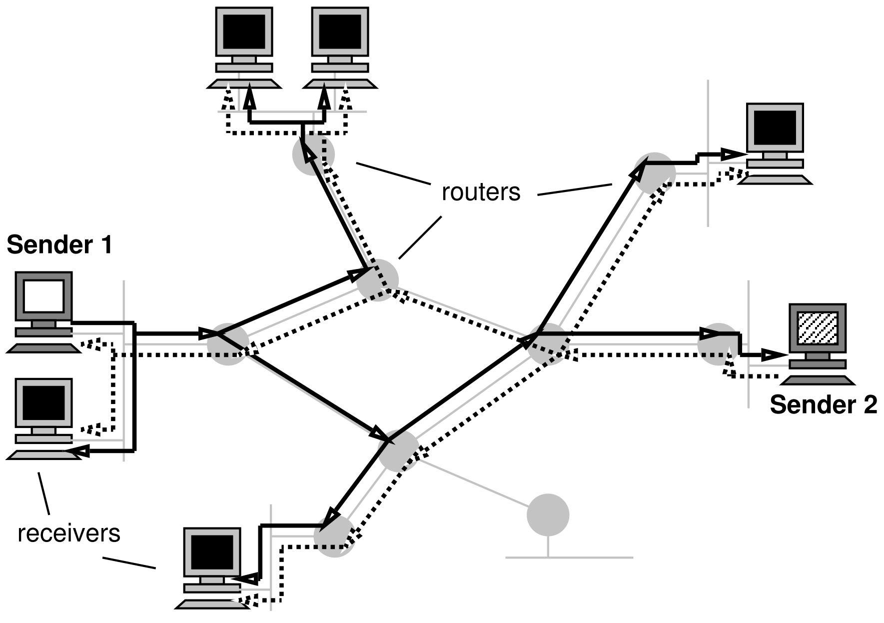


Figure 3.1: Shortest path multicast routing with multiple senders

DVMRP[15],MOSPF[61] and Dense-mode PIM[19] are examples of shortest path multicast routing schemes. A shortest path scheme delivers data from a sender to each receiver along the shortest path from the sender to that receiver[2]. If several sources send simultaneously, each of them may use a different tree as illustrated in figure 3.1.

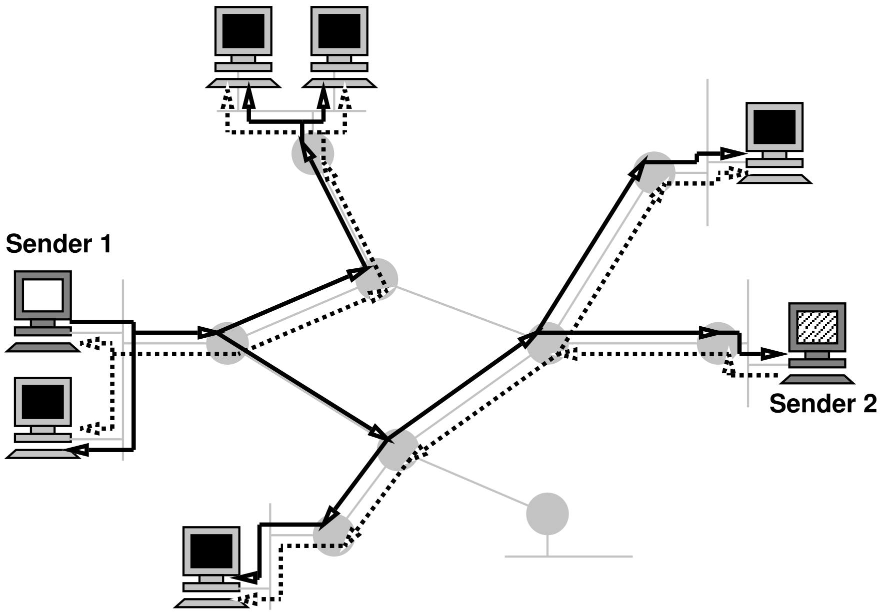


Figure 3.2: Shared-tree multicast routing with multiple senders

CBT[2] is an example of shared tree multicast routing scheme, as illustrated in figure 3.2. Shared tree algorithms build a single tree that is used by traffic from all senders[3]. Typically this is done by using the shortest path tree to a single router (known as the Core or Rendezvous-Point) to build the tree.

Sparse-mode PIM[22] is an example of a hybrid scheme. It starts out building a variant of a shared tree to get data flowing from senders to receivers, but then if the traffic is sufficient, a router near to a receiver can initiate a switch to the shortest path tree.

### 3.3.1 Effects of multicast routing on applications

On average, shortest path trees provide lower delay, fewer hops, and slightly less jitter than the equivalent shared trees. These effects are usually fairly small, but may be significant for some very delay constrained applications. However the effect of the type of tree can also show up in other

[2] In practice, the shortest *reverse* path is usually used; i.e. the shortest path from the receiver to the sender. To a first approximation, this produces the same behaviour but is simpler to implement.

[3] although traffic from different sources may flow in opposite directions on any particular link

ways.

One example is with probabilistic suppression schemes such as used by SRM. This relies on a retransmission request from near to a loss-point traveling down the multicast tree to suppress other potential duplicate requests. In chapter 6, the effect of different tree types shows up in the suppression results obtained. Shortest path trees are slightly better for such schemes because the response performing the suppression can often take a shorter path to other receivers; with a shared tree, such a response often needs to travel up-tree[4] to the branch point and then travel down-tree to the other receivers, which takes longer and hence shows up as a greater number of responses if the other parameters are left unchanged.

Other small effects can come to play in the interaction between multicast routing and applications. The service model does not explicitly state that the first packet to be sent to a group need actually be forwarded. It would be very simple to design a multicast routing scheme requiring some form of lookup or state initiation on receipt of the first packet, and to drop the first few packets whilst such state initiation is being performed. However, to do this would cause problems for many potential multicast applications. For example, in chapter 6 the session directory sends a single announcement packet every few minutes from an announcing host. If a multicast routing scheme needed to perform *per-source* state initiation and dropped packets until this had occurred, and if its state timeout was shorter than the announcement interval, then all session announcements would be deterministically dropped.

Another aspect of multicast forwarding that is used in chapter 6 is TTL scoping. Each time a packet is forwarded, the time-to-live field is decremented, and the packet is discarded when TTL reaches zero. With shortest-path multicast routings schemes, this can be used for scoping of traffic to a region local to a particular source. This can also be used to implement techniques such as an expanding ring search. However, during its shared-tree startup phase, Sparse-Mode PIM encapsulates each data packet at a sender's local router and forwards it directly to the Rendez-vous Point (RP) at the root of the tree. Thus, depending on how the encapsulation is performed, either all TTL 2 packets reach the RP, or all packets with TTL insufficient to reach the RP are dropped without reaching any other receivers. Neither of these is desirable behaviour, and we are currently working to remedy this aspect of SM-PIM.

Ideally, the entire multicast service model would be written down, including definitions of correct first-packet behaviour and scoping behaviour. Without this, we are forced to rely on precedents from how multicast routing schemes have performed in the past, and to talk to router vendors about

[4] up-tree in this context is towards the original data source

how their protocols behave in response to these boundary effects.

### 3.3.2 Effects of applications on multicast routing

When designing scalable applications, the effects of the application on the multicast routing scheme should also be considered, although this is clearly secondary to how the application performs itself. An example of this is the interaction between RTCP and DVMRP as the group size increases. DVMRP is a flood-and-prune multicast routing protocol, meaning that new sources are flooded to all possible locations within the scope zone, and destinations that have no receivers for the group prune back the distribution tree so that traffic only flows over links that lead to receivers. This per-source prune state is timed out periodically if no data has been received. However, as an RTCP session scales, the number of participants sending session messages increases. Not only can this lead to a large amount of per-source prune state in the network, but when the mean inter-report interval from each source is greater that DVMRP's state timeout interval, session messages end up being flooded throughout much of the scope zone. This either places a limit on the size of multicast sessions that use session messages, or alternatively places a limit on the size of single DVMRP routing domains if the behaviour is to be acceptable. Clearly shared-tree schemes that rely on explicit join messages do not suffer from this problem, and thus they are a good candidate for the upper levels of a hierarchical multicast routing scheme.

## 3.4 Multicast Scoping

When applications operate in the global MBone, it is clear that not all groups should have global scope. This is especially the case for performance reasons with flood and prune multicast routing protocols, but it also the case with other routing protocols for application security reasons and because multicast addresses are a scarce resource. Being able to constrain the scope of a session allows the same multicast address to be in use at more than one place so long as the scope of the sessions does not overlap.

Multicast scoping can currently be performed in two ways which are known as TTL scoping and administrative scoping. Currently TTL scoping is most widely used, with only a very few sites making use of administrative scoping.

### TTL Scoping

When an IP packet is sent, an IP header field called Time To Live (TTL) is set to a value between zero and 255. Every time a router forwards the packet, it decrements the TTL field in the packet header, and if the value reaches zero, the packet is dropped. The IP specification also states that TTL should be decremented if a packet is queued for more than a certain amount of time, but this is rarely implemented these days. With unicast, TTL is normally set to a fixed value by the sending host (64 and 255 are commonly used) and is intended to prevent packets looping forever, and also forms a part of the formal proof that the TCP close semantics are safe.

With IP multicast, TTL can be used to constrain how far a multicast packet can travel across the MBone by carefully choosing the value put into packets as they are sent. However, as the relationship between hop-count and suitable scope regions is poor at best, the basic TTL mechanism is supplemented by configured thresholds on multicast tunnels and multicast-capable links. Where such a threshold is configured, the router will decrement the TTL, as with unicast packets, but then will drop the packet if the TTL is less than the configured threshold. When these thresholds are chosen consistently at all of the borders to a region, they allow a host within that region to send traffic with a TTL less than the threshold, and to know that the traffic will not escape that region.

An example of this is the multicast tunnels and links to and from Europe, which are all configured with a TTL threshold of 64. Any site within Europe that wishes to send traffic that does not escape Europe can send with a TTL of less than 64 and be sure that their traffic does not escape.

However, there are also likely to be thresholds configured within a particular scope zone - for example most European countries use a threshold of 48 on international links within Europe, and as TTL is still decremented each time the packet is forwarded, it is good practice to send European traffic with a TTL of 63, which allows the packet to travel 15 hops before it would fail to cross a European international link.

### Administrative Scoping

There are circumstances where it is difficult to consistently choose TTL thresholds to perform the desired scoping. In particular it is impossible to configure overlapping scope regions as shown in figure 3.3, and there are a number of other problems with TTL scoping, and so more recently, administrative scoping has been added to the multicast forwarding code in mrouted[16] and in most router implementations. Administrative scoping allows the configuration of a boundary by specifying a range of multicast addresses that will not be forwarded across that boundary in either direction.

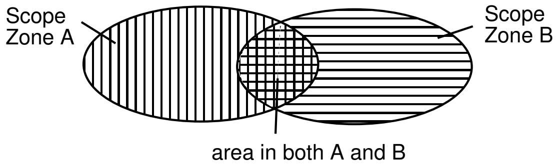


Figure 3.3: Overlapping scope zones possible with administrative scoping

### Scoping Deployment

Administrative scoping is much more flexible than TTL scoping, but suffers from a number of disadvantages. In particular, it is not possible to tell from the address of a packet where it will go unless all the scope zones that the sender is within are known. Also, as administrative boundaries are bi-directional, one scope zone nested within or overlapping another must have totally separate address ranges. This makes their allocation difficult from an administrative point of view, as the ranges ought to be allocated on a top-down basis (largest zone first) in a network where there is no appropriate top-level allocation authority. Finally, it is easy to misconfigure a boundary by omitting or incorrectly configuring one of the routers - with TTL scoping it is likely that in many cases a more distant threshold will perform a similar task lessening the consequences, but with administrative scoping there is less likelihood that this is the case.

For these reasons administrative scoping has been viewed by many network administrators as a specialty solution to difficult configuration problems, rather than as a replacement for TTL scoping, and the MBone still very much relies on TTL scoping.

# Chapter 4

# An Analysis of Mbone Performance

## 4.1 Introduction

In order to design tools that will perform a task effectively, it is necessary to have a good understanding of the environment in which those tools will function. In this chapter, we monitor and analyze the performance of the Internet's Multicast Backbone (Mbone), and attempt to loosely characterize the environment in which Mbone tools have to operate.

This characterization leads us to make suggestions about the design of multicast applications. In particular, it leads us to conclude that the designers of reliable multicast algorithms should think about the use of redundancy, either at the application level, or in the form of packet level forward-error-correction.

These results also have implications for the design of rate-adaptive multicast applications, and lead us to conclude that such applications will not perform well with current FIFO queuing schemes as the loss feedback they obtain is not useful for such applications to use in a rate-adaptive feedback loop. Thus we believe that queuing algorithms such as RED and some form of fair queuing are extremely desirable to allow such applications to obtain more meaningful feedback from the network about appropriate transmission bandwidths.

Two other studies [7][74] of this area have been performed. The work of Bolot et al [7] studies temporally correlated loss in fixed bit-rate audio streams, and finds results which are in general agreement with this study. Yajnik et al [74] study both spatial and temporal loss correlation by using dedicated monitoring software at 17 sites. They get more detailed correlation information than we are able to manage, but because these sites are members of the networking research community, their results are not necessarily representative of the actual distribution of sites involved in large sessions.

When monitoring MBone sessions at real receiver sites we discover higher loss rates than they encounter, although the pattern of their results is in general agreement with ours. Both these studies use fixed bit-rate audio data as their test traffic source. By using variable-rate video as our test traffic sources, we are able to draw additional conclusions about congestion control mechanisms.

### 4.1.1 MBone Monitoring

The ways we can monitor the state of the Mbone from an end-system are necessarily limited, but in the last two years the options have increased significantly. Multicast routes and (multicast) link loss statistics can be collected with *mtrace*[23], and with the wide-scale deployment of RTP based conferencing systems, the tools themselves report limited loss statistics using RTCP. Both mtrace and RTCP only provide loss averages, rather than information about individual packet losses.

Prior to the development of mtrace and RTP based tools, Stuart Clayman and Atanu Ghosh at UCL had written the tools *mr*, *mwatch*[32] and *rtpmon* which performed similar tasks. However, mr is unreliable due to its reliance on being able to contact a multicast router using IGMP[1] and also due to the fact that DVMRP multicast routing in older versions of mrouted is non-deterministic in its choice of route when two equal metric paths are available. Rtpmon was reliable, but required that sites involved in a session run the monitor daemon, which is difficult to achieve in sessions of any significant size.

In addition, we can examine the routing tables from *mrouted* directly to monitor route stability, and we can observe traffic as it arrives at sites to which we have access to look at individual packet losses.

Of primary importance for audio, video, reliable multicast and conference control applications is information about packet loss. In the design of audio and video tools, information about delay and jitter is also of significance, but currently there is no way to monitor this information as current RTP based tools do not report jitter correctly.

## 4.2 Data Collection

Three programs were written to collect data on which the following examination is based:

- An RTP logger, which logs data packet arrivals at a receiver.

[1] IGMP is the Internet Group Management Protocol[17]. Mwatch uses IGMP to obtain the set of multicast neighbours of an mrouter

- An RTCP logger, which logs RTCP reception reports from all receivers in a session.
- An mtrace daemon, which attempts to trace the multicast routes from a sender to all the receivers in a session.

### The RTP logger

For every RTP data packet that arrives at a receiving site on the specified address, this logs the synchronisation source, packet Length, sender timestamp and arrival time.

### The RTCP logger

For every RTCP control packet that arrives at a receiving site on the specified address, this logs the synchronisation source of the reporting host, RTP SDES information of the reporting host including NAME, CNAME and TOOL, and the report arrival time. In addition it logs a list of reception reports for each sender heard at this reporting host, giving the synchronisation source of the sender and the fraction of expected packets lost in the last reporting interval.

An example of a logged RTCP report is:

```
TIME: Sun May 26 14:47:27 1996
RR: ssrc id=296197894 fraction lost=0
SDES: ssrc id=295962065
CNAME: jim@199.45.1.66
NAME: Jim Martin (InteropNet)
TOOL: vic-2.7a38
EMAIL: jim@interop.net
```

### The Mtrace Daemon

To monitor where loss is occurring in the Mbone, we wrote an mtrace daemon. This listens to the RTCP packets sent from receivers to a multicast group and builds sender and receiver lists for the session. It then attempts to *mtrace* the multicast route from the chosen sender to all the receivers.

Many multicast trees have been traced with this tool. Figure 4.1 shows the distribution tree for the NASA Shuttle video session on 21st May 1996. This was one of the larger sessions monitors. At the time these statistics were gathered, the *mwatch* daemon reported multicast router version information as:

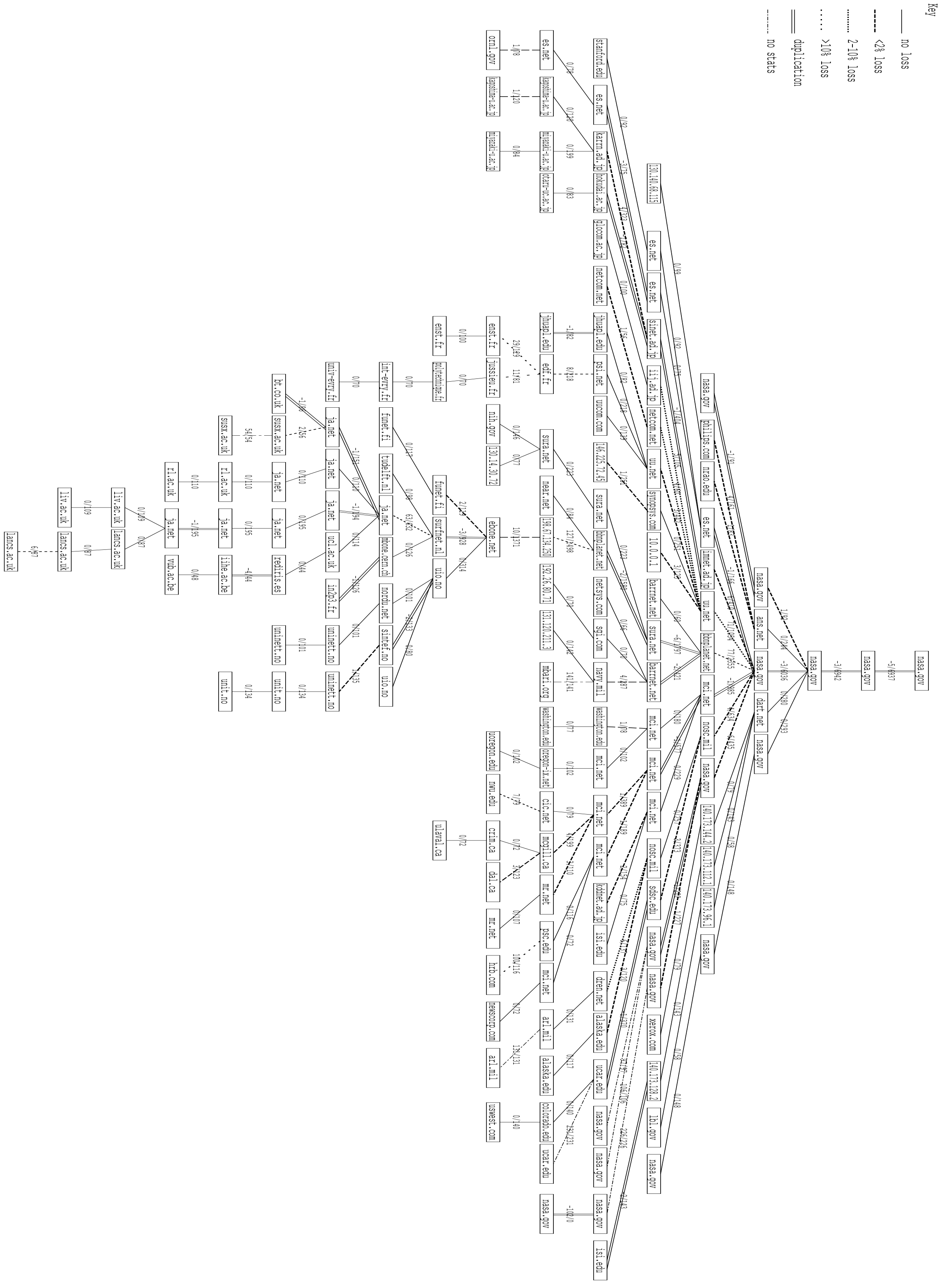

Figure 4.1: The Loss Distribution Tree for NASA Shuttle Video, 21/5/96, 19:57 BST

| version | hosts | percent | version | hosts | percent |
|---|---|---|---|---|---|
| 3.8 | 763 | 47.0% | 3.3 | 31 | 1.9% |
| 11.0 | 194 | 11.9% | 2.0 | 21 | 1.3% |
| 10.3 | 168 | 10.3% | 1.0 | 14 | 0.9% |
| 11.1 | 140 | 8.6% | 3.5 | 12 | 0.7% |
| 3.6 | 88 | 5.4% | 11.2 | 7 | 0.4% |
| 2.2 | 70 | 4.3% | 3.2 | 5 | 0.3% |
| 10.2 | 68 | 4.2% | 3.1 | 5 | 0.3% |
| 3.4 | 34 | 2.1% | 3.7 | 4 | 0.2% |
| | | | Total | 1624 | |

Of these, versions prior to 3.3 do not properly support mtrace, and so it should be possible to mtrace through approximately 93% of the multicast routers in the Mbone. For example, on May 21st 1996, there were 112 hosts listening to the NASA STS-27 video session, but of the 112 mtraces attempted, only 68 were successful. Of the successful traces, the mean path length was approximately 8. Assuming this to be typical of unsuccessful mtraces, we expect a path to contain one or more mrouters that do not support mtrace approximately 44% of the time, which is in line with these figures, and so implies that it is reasonable that routes where mtrace was unsuccessful are similar to those where it was successful except that there is an old mrouter somewhere on the path.

However, an alternative explanation is that mrouters that do not support mtrace are located in less well supported locations (as these are older versions of mrouted), so are more likely to be located away from the "backbone" routes, and less likely to be on any particular mtrace path. If so, then mtrace is also failing for some other reason. If this is due to excessive loss, then the statistics reported by successful mtraces will not be representative of typical conditions in the Mbone. This seems unlikely though, as many mtrace trees were constructed, and over the short term, the trees do not change to any great extent.

If the latter explanation were the case, then any prediction of typical global Mbone conditions based on successful mtraces will be an lower bound on loss rates likely to be seen. As we are unsure how representative these trees are, we try to avoid using loss statistics gathered from mtrace alone. Repeating this examination over time as the number of older routers decreases has been difficult because Cisco introduced a bug into their mtrace daemon software in the summer of 1996 which made mtrace useless as a diagnostic tool for almost a year, and later introduced a bug causing multicast route flaps, which adversely affected multicast routing worldwide throughout the summer of 1997. Repeating the experiment in Oct 1997, when 96% of mrouters support mtrace, during a quiet weekend shows about 50being successful. The vast majority of the unsuccessful mtraces for which we can obtain information are now for sites whose local router is a Cisco running PIM, implying that this routing instability is still commonplace at leaf sites. Thus the spring 1996 mtraces are still the most successful ones we have available.

The tree shown in figure 4.1 is typical of the many mtrace trees that were gathered. It is one of the larger trees because it was taken at a time when the number of participants in the Shuttle Video session was near its peak, but the general distribution of losses is repeated in the other traces, with a few links contributing a disproportionate amount of loss but many other links contributing some loss. It serves to illustrate where problems are occurring in the mbone, and that a relatively large number of links are contributing packet loss. In this particular trace, two links close to the entry to France and one to HRB are contributing greater than 10% loss (22%, 14% and 86% respectively), but there are another seventeen links contributing between 2% and 10% loss, and many more contributing less than 2% loss.

## 4.3 Analysis of network logs

RTP and RTCP packets for the NASA Shuttle Video session were logged from Sunday 26th May 1995 to Tuesday 28th May inclusive. The RTP logfile totalled 279MBytes and logged 2.7 million packets. The RTCP logfile totalled 125MBytes and logged approximately 840,000 RTCP packets from a total of 306 receivers. A similar number of RTP and RTCP packets were also captured between the Monday 16th and Thursday 19th September, and a smaller number captured for the Usenix session in Jan 1997.

Although it is not claimed that these traces are representative of all Mbone sessions, the May traces cover two quiet days (a Sunday and a public holiday in Europe and the US) and a normal working day, and routes covered by the traffic cover most of the Mbone backbone links. The September traces cover several weekdays, although the network may be lighter loaded than normal because this is still during the period when UK university students are on holiday.

RTCP statistics from the vic video tool which was used by all participants give the loss rate calculated by dividing the packets received by the packets expected in that interval. This fraction is sent as an integer in the range 0-255, and so loss rates are accurate to approximately the nearest 0.5%. It is possible that some of the RTCP receiver reports are reporting loss which occurs in the end-system rather than the network due to vic suffering from processor starvation and overflowing its UDP receive buffer. However, given that the traffic measured constitutes a relatively low frame-rate video stream (typically only 3-4 fps when the video is in motion, higher when there's no motion, but very little decoding needs to be done then) it is unlikely that this is a significant source of loss. On the author's 60MHz SparcStation 20, decoding and displaying this video stream requires between 4% and 10% of the CPU time. Even low performance workstations are unlikely to be CPU starved by this video stream alone. Thus although we cannot be sure that the statistics have not been affected to some degree, it is expected that the influence of this effect is small.

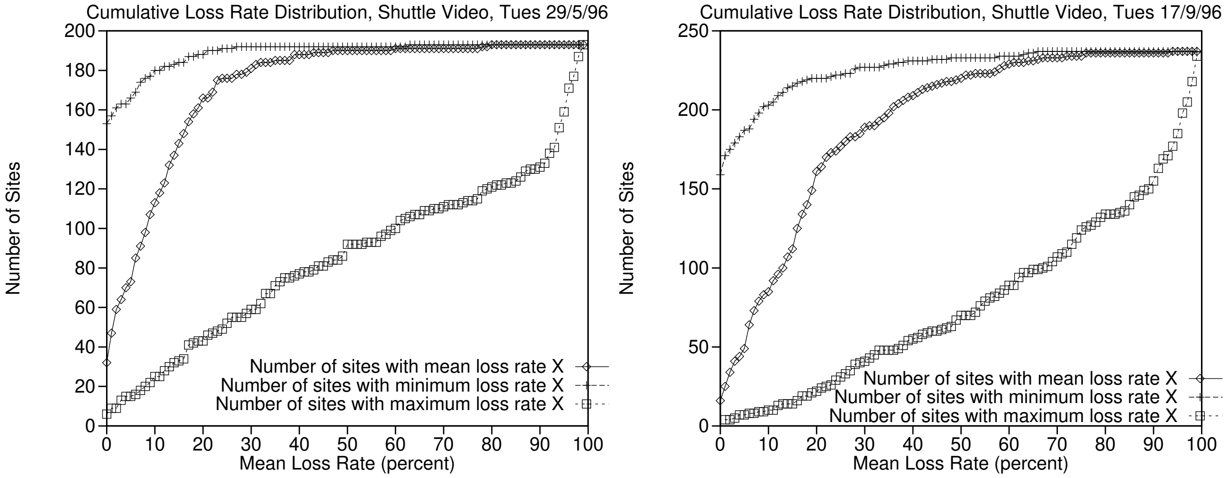


Figure 4.2: Loss Rates by Site

### 4.3.1 Loss Rates by Site

Figure 4.2 shows the breakdown of loss by site from the working day of the May 1996 trace and the Tuesday from the September 1996 trace as a cumulative graph. Each RTCP reception report gives the mean loss rate seen by that receiver for the NASA source in that RTCP reporting interval. Reporting intervals vary depending on the number of receivers, but for these sessions they are typically between 10 and 60 seconds. For each receiving site, the minimum reported loss rate, the mean reported loss rate, and the maximum reported loss rate during the day are calculated.

The May graph shows that 50% of receivers have a mean loss rate of about 10% or lower, and that around 80% of receivers had some interval during the day when no loss was observed. However, 80% of sites reported some interval during the day when the loss rate was greater than 20%, which is generally regarded as being the threshold above which audio without redundancy becomes unintelligible, and about 30% of sites reported at least one interval where the loss rate was above 95% at some time during the day.

The September graph shows a similar overall pattern, but the loss rates are significantly higher. Although this may reflect an overall increase in congestion in the internet, it would be foolish to make such an assumption from these figures, although similar trends that have been reported elsewhere[65]. In practice, congestion levels seem to increase progressively, and then show step-function decreases as new infrastructure is put in place. Thus long-term trends require studies on the order of years to deduce real trends.

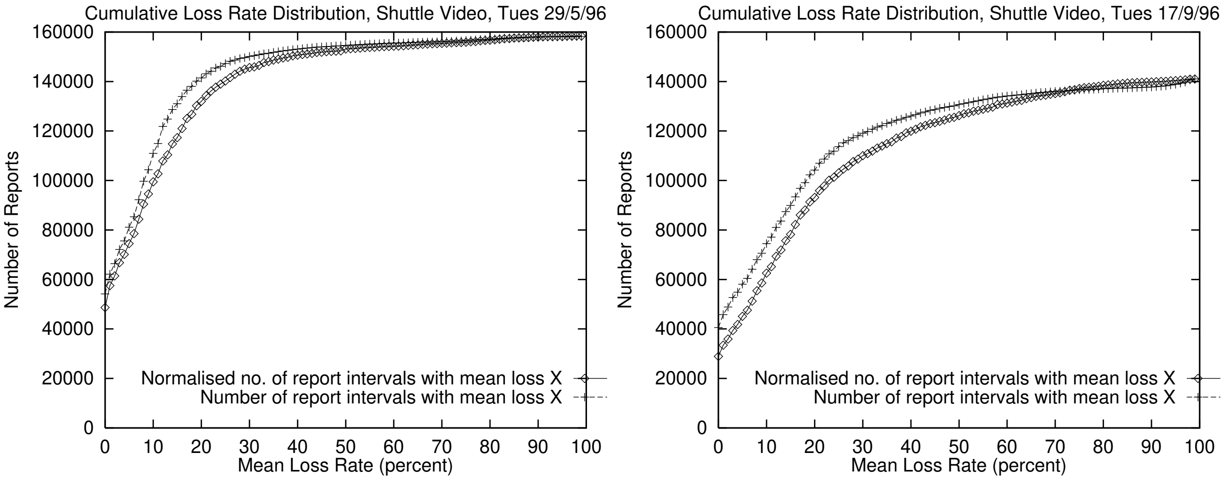


Figure 4.3: Loss Rate Report Distribution

### 4.3.2 Loss Rates Report Distribution

Figure 4.3 shows the number of loss rate intervals reporting each loss rate as a cumulative graph. The graphs are taken from the same data as figure 4.2, and include reports from all receivers. As these may have been affected by the loss of RTCP reports, we show the raw results and the results achieved by normalising the reports from each individual receiver, such that a report from receiver $i$ is given weight $w_i = n_i/\overline{n}$ where $n_i$ is the number of reports received from $i$ and $\overline{n}$ is the mean number of reports from a receiver[2]. The normalised curve is intended to weight receivers whose RTCP reports get lost on the way to the monitor as if their reports had arrived. Whether this is in fact the case is not clear, as it may overweight such receivers as their duration in the session may also be shorter because the loss rate is high. However, as the true situation is likely to lie between the two curves they approximate upper and lower bounds.

These graphs show the loss distribution to be very long tailed, with large amounts of relatively low loss rates observed, and although *very* high loss rates are also seen, they are rare in the May

[2] $\overline{n} = N/R$ where $N$ is the total number of reports from all receivers and $R$ is the total number of receivers

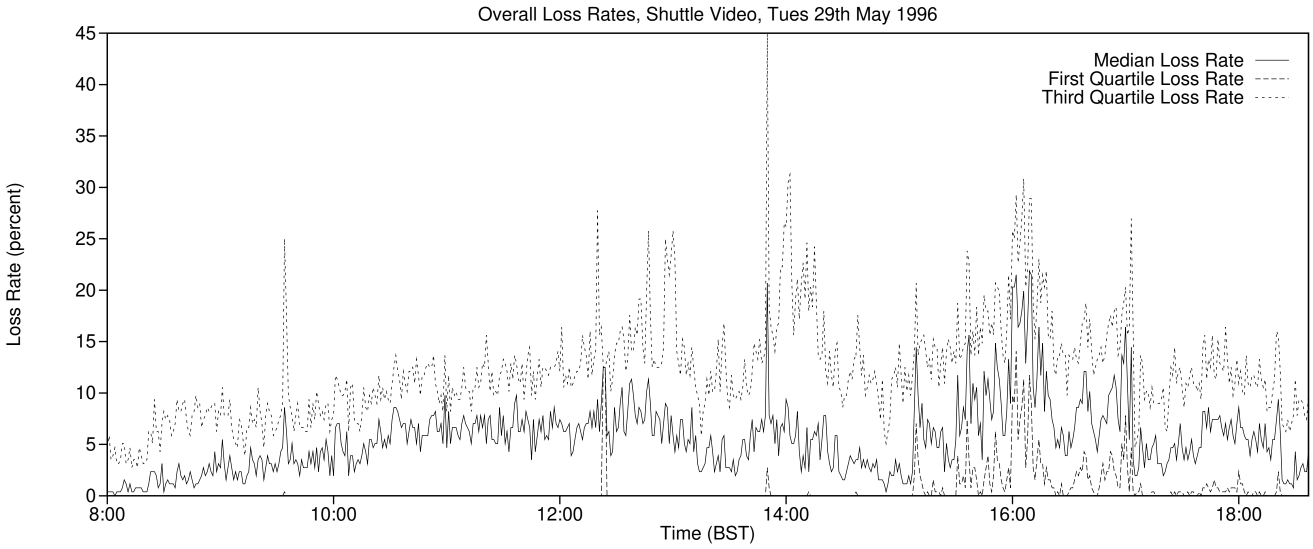


Figure 4.4: Loss Rate Distribution against Time

figures; 90% of all reports give un-normalized loss rates of less than 20%. In September, this number is more like 73%.

A interesting artifact is visible in the September figures, where the un-normalised figures show a small peak above 90% loss which is not visible in the normalised figures. This indicates that some long term participants which normally suffer relatively low loss rates have suffered some form of short-term failure at some time, but that normalisation has reduced their contribution to the curve significantly.

### 4.3.3 Loss Rate Distribution with Time

Figure 4.4 shows the loss rate distribution against time in minutes for the shuttle video session on Tuesday 29th May. Three lines are shown, giving the median loss rate reported from all receivers reporting in a one minute interval, and the first and third quartiles. Note that although 25% of the receivers are suffering no loss for much of the day, the median loss rate is between 5 and 10% for the majority of the day. Also 25% of receivers are suffering loss rates of greater than 15% for all the afternoon (between 12:00 and 4:30 UTC) when both the European and US communities are active.

Figure 4.5 shows the loss rate against time for each receiver during a 33 minute interval of the shuttle video session from 11:30am BST on Tuesday 29th May and for two intervals on Tuesday 17th Sept 1996. These graphs were obtained from the RTCP receiver reports for each receiver by taking a 3 minute windowed average to minimize boundary effects caused by the de-synchronized nature of RTCP receiver reports. Receivers are shown in order of mean loss rate. Although the

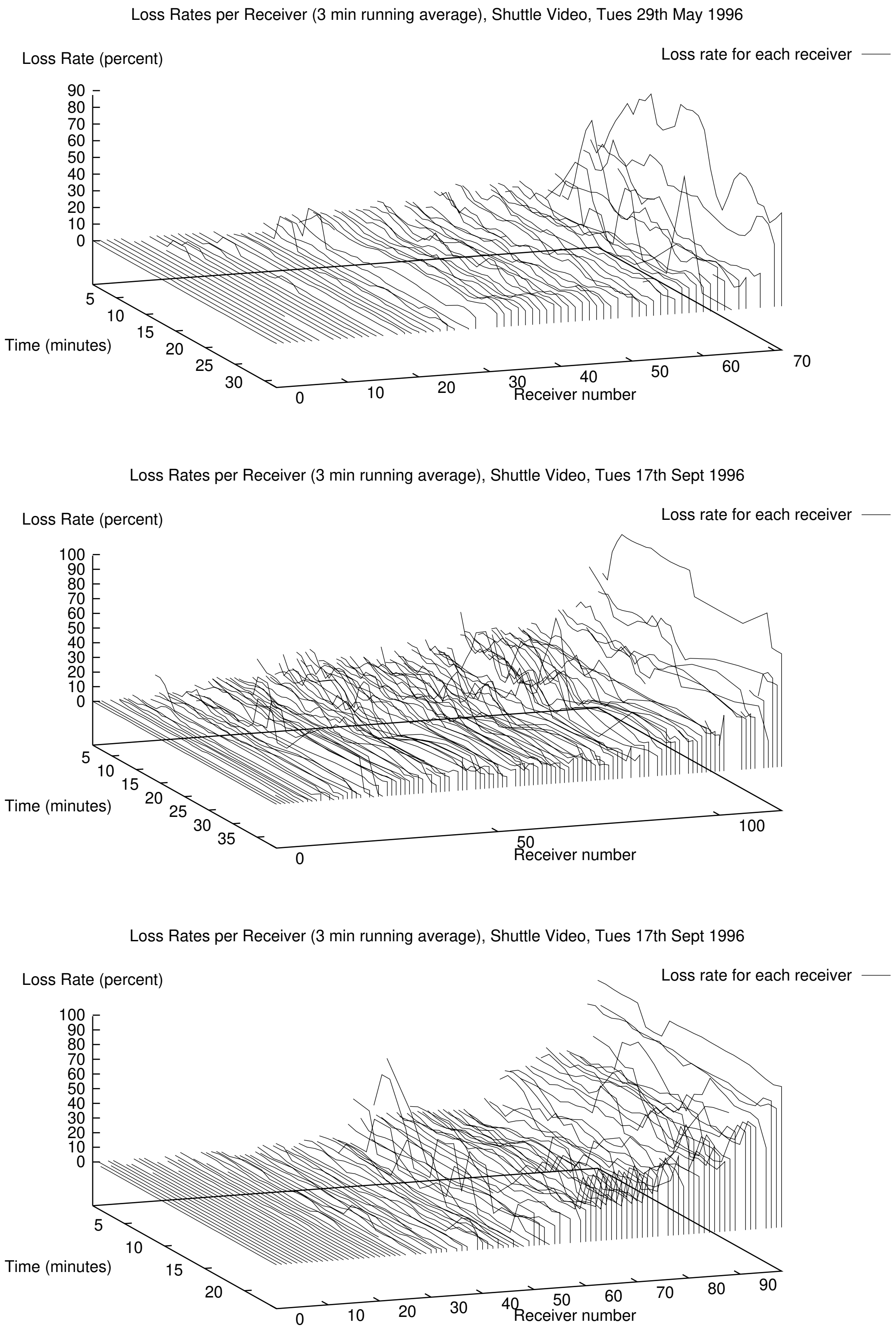


Figure 4.5: Loss rate against time for each receiver

precise time intervals are arbitrary, they were deliberately chosen from a period when fairly high loss rates are visible in figure 4.4 and the equivalent for the 17th Sept. Several graphs were taken from these data sets collected on 29th May and 17th Sept, and show similar degrees of correlation of loss between receivers.

The graphs shown are representative of the range seen, and vary from reasonably good correlation (on May 29th) to less good correlation (on the first graph from Sept 17th). They show some correlation of loss between receivers, as would be expected, but they also show a large amount of additional low-amplitude noise, which would only be expected if many different points in the distribution tree are causing usually small amounts of loss independently. In addition, there are always a small number of receivers suffering very high uncorrelated loss. This agrees with the loss distribution shown in figure 4.1, which shows a small number of high loss links which are responsible for the larger scale correlation in 4.5 and a large number of links with low loss rates which add noise to the correlation.

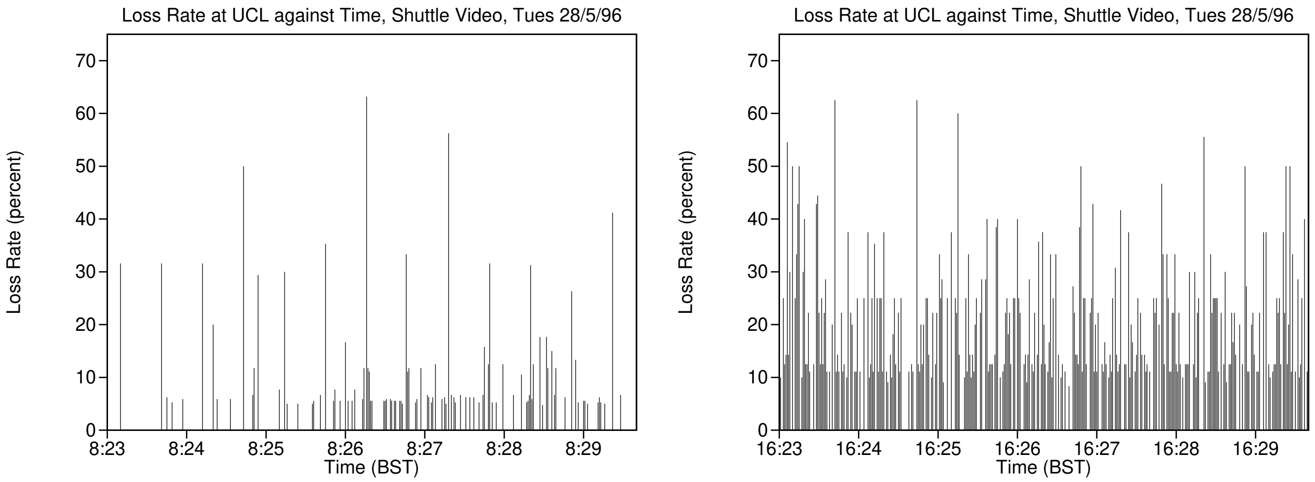


Figure 4.6: Loss Against Time

Figure 4.6 shows loss rate against time measured from the RTP traffic received at UCL, London from NASA. Unlike 4.5 these graphs give very fine grain loss statistics, but only do so for one site. The first graph in figure 4.6 was taken early in the morning (UK time) on Tuesday 28th May 1996, when the networks were lightly loaded. The second graph was taken at the time of maximum loss as shown on figure 4.4. Both graphs show a clear 31 second periodic loss pattern, and although this is less obvious in the second graph because of other loss, the loss autocorrelation graph taken from the same data in figure 4.7 does show that this effect is still significant when the network loss levels are high. Yatnik et al [74] also reported similar periodic losses.

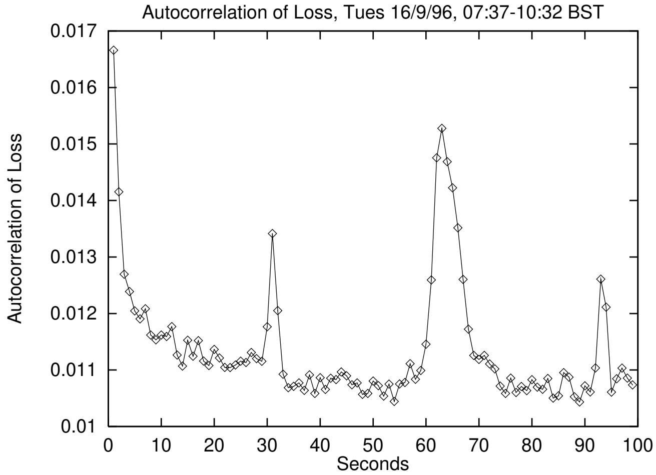


Figure 4.7: Autocorrelation of Packet Loss

We have attempted to verify that this effect is not a measurement artifact by performing the same dump on different workstations running different operating systems. The main dumps are all taken on a SparcStation 20 running Solaris 2.5. Short dumps were also taken on an HP 9000/725 running HPUX on the same ethernet as the SS20, and a SparcStation IPX running SunOS 4.1.3 on a different ethernet at UCL. The effect is still visible on these traces. Test traffic sent through the UCL multicast router monitored on the same machines does not show this effect.

This effect cannot be a periodic effect in the sending workstation at NASA because it would be visible in the RTCP receiver reports. At UCL, this effect is approximately 50% of packets being lost in one second, repeating every 31 seconds. This corresponds to approximately 1.5% loss, which is large enough to be reported by RTCP. We see significant numbers of receivers which report no loss, which rules out an artifact in the sending machine at NASA.

Such period loss patterns are likely to result from a periodic effect in the network itself. In the past, Van Jacobson has attributed similar effects to a bug in certain routers, where they invalidate their interface routing cache while processing routing updates, and to synchronisation of routing updates [27]. Although these seem likely there is not enough data in these RTP or RTCP figures to verify either hypothesis.

Both local tests and the absence of such effects for significant numbers of receivers tend to imply that this is not likely to be a ubiquitous artifact of DVMRP routing itself or of the mrouted implementation. However, an interaction between multicast routing and some other periodic process on a tiny minority of workstations running mrouted cannot be ruled out from this data.

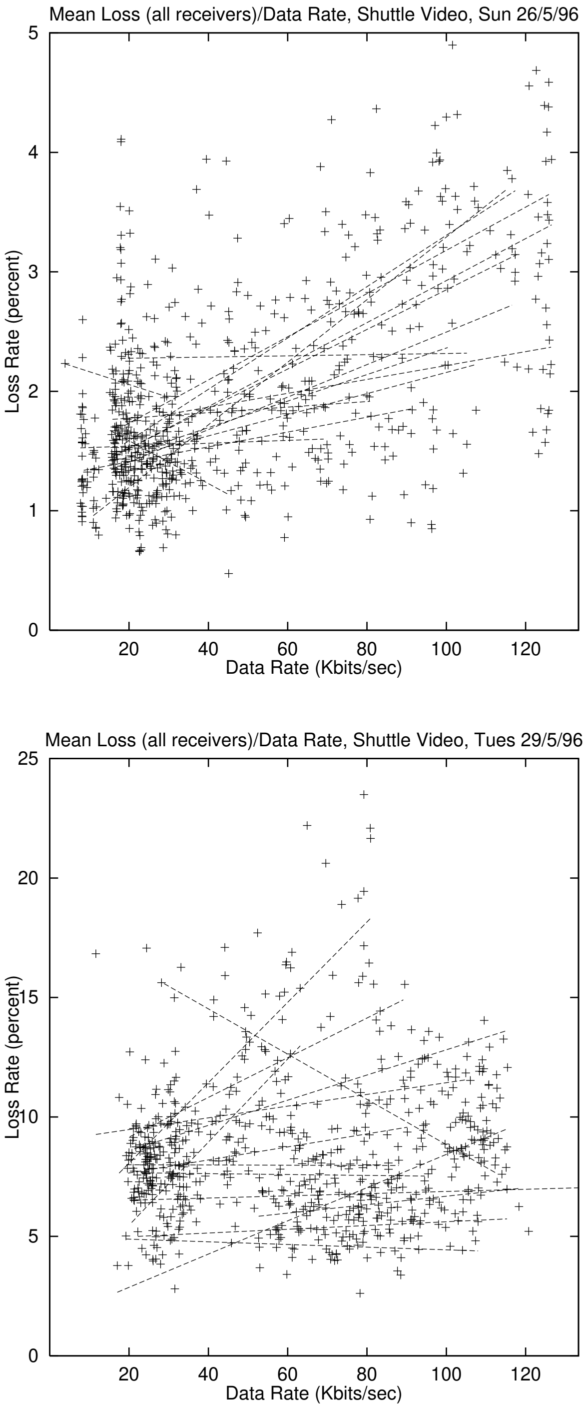


Figure 4.8: Loss Against Sent Data Rate

### 4.3.4 Loss Rate Distribution with Senders Data Rate

The graphs in figure 4.8 show the mean loss rate for each one minute interval calculated from all the RTCP receivers' reports plotted against an estimate of the mean transmitted data rate for that minute. This estimate is calculated by assuming that RTP packets missing when they arrive at UCL were of mean size. As vic tends to send fairly constant sized packets, this assumption appears justified. The data plotted in both cases represents a fourteen hour UK daytime section of video reception.

The lines plotted over this data are least-squares fits for each 50 minute interval in the source data. They attempt to illustrate whether any effects may have only been present for part of the sampled data, but they should be read as illustrative only, as there may be coincidental boundary effects such as a short period of low loss rate due to reduced cross-traffic coinciding with a period of increased sender bandwidth.

The data from the Tuesday graph illustrates the difficulty of building a congestion control mechanism for multicast traffic that must adapt on significantly longer time-scales than TCP does. In this data there are long periods of time where there is no discernible correlation between transmitted data rate and mean loss rate measured at the receivers even though the bandwidth changes by an order of magnitude. If even a fairly small proportion of receivers were showing such a correlation, then such a trend should be visible in this graph.

This should not be too surprising as most of the links involved are high bandwidth running at T3 or greater except at some of the leaves, and employ drop-tail FIFO queuing. 128Kbps of traffic is likely to be a very small proportion of the total traffic on such a congested link, and reducing this traffic does not noticeably change the total load on the link, and hence the loss rate remains constant.

If this data is extrapolated, it shows that multicast applications that attempt to adjust so that no receiver is suffering significant loss will almost inevitably adjust so that they send no data (or whatever their minimum data rate is configured to). Thus adaptation schemes such as [4] are unlikely to be useful in the wide area Mbone as we see it today.

### 4.3.5 Consecutive Losses

The graphs in figure 4.9 (A-C) show the frequency of RTP sequence number increments as observed at UCL on each of several days. We know from figure 4.6that at least two processes combine to produce loss, and this is reflected in figure 4.9. The base losses appear to obey a simple rule where the probability $p$ of a sequence number increase of $n, n >= 1$, is approximated by $p(n) = an^b$ where for the graphs shown, $b$ is in the range -8.7 to -3.5. In addition there is a peak in all the graphs

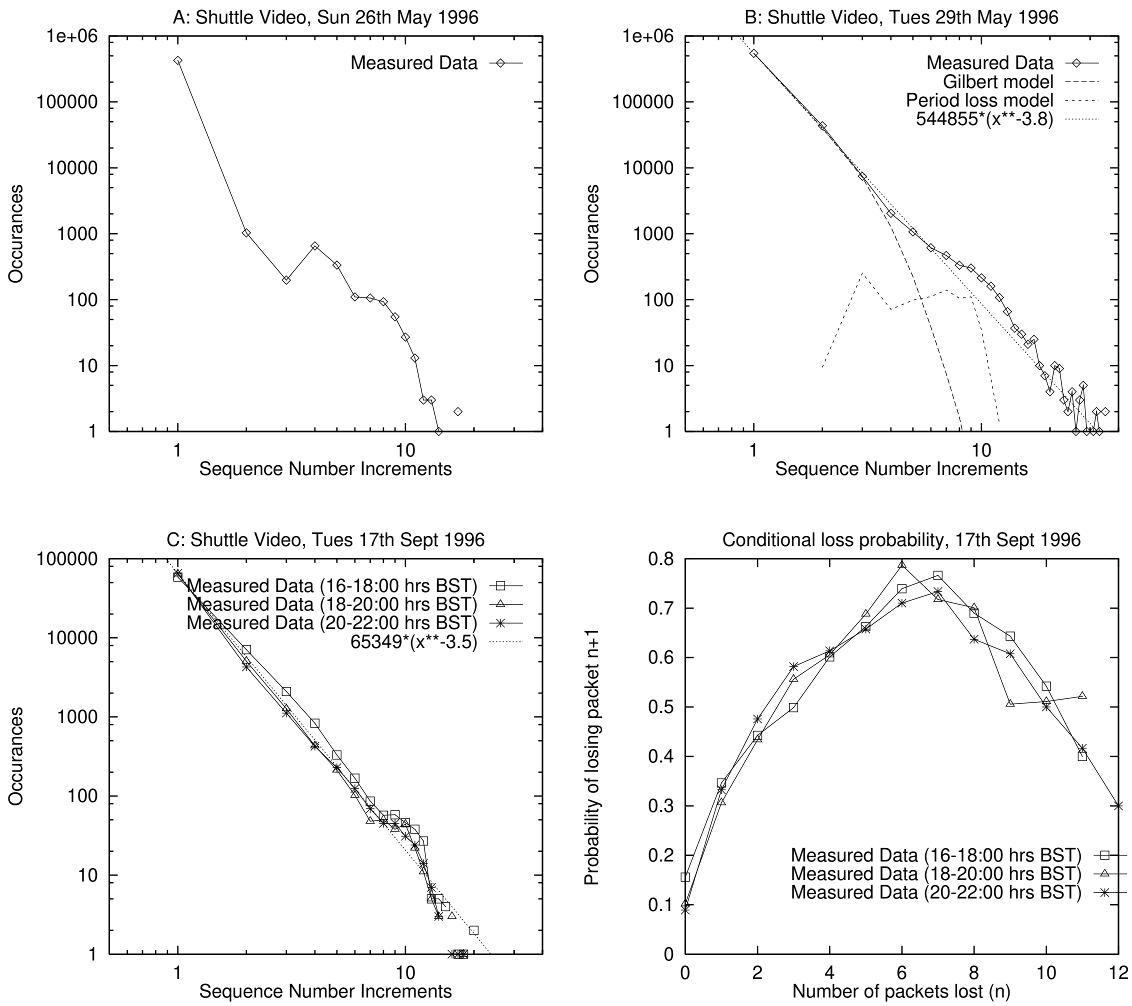


Figure 4.9: Frequency of Sequence Number Increments (Consecutive Losses - 1)

in figure 4.9 at around 5 to 15 consecutive losses which does not follow the simple rule. These peaks are consistent with the thirty-second periodic loss bursts observed in figure 4.6.

Figure 4.9d shows the conditional loss probability; that is given that packets $x$ to $x + n - 1$ were lost, the probability that packet $x + n$ is also lost[3]. As these curves show, packet losses are not independent, and tend to occur in longer bursts than would be the case if they were independent. However, with the exception of the thirty-second periodic losses, the excess of bursts of 2-5 packet losses compared with what would be expected from random loss, although statistically significant, is not sufficient to greatly influence the design of most applications, and single packet losses still dominate.

In the case of applications such as UCL's Robust-Audio Tool[39] and INRIA's FreePhone audio

[3] n=0 gives the probability that a packet is lost if the previous packet was not lost

tool that provide redundancy by carrying a highly compressed redundant copy of data one or more packets behind the primary encoding, these curves show that there is some gain from adding the redundant encoding more than one packet behind the primary. For example, from the data shown in 4.9c (16:00-18:00), 77243 packets were sent and 18640 (24.1%) were lost. Adding a single redundant payload one packet behind the primary encoding allows the reconstruction of audio from 7082 missing packets resulting in an effective loss rate of 15.0%. Adding the same redundant payload two packets behind the primary allows the reconstruction 7885 packets resulting in a effective loss rate of 13.9%. Unfortunately this extra gain comes at the expense of extra delay in the case of this audio coding scheme so may or may not be worthwhile depending on the purpose of the stream itself.

Whilst these results agree with the conclusions of Bolot[5], the two-stage Gilbert model used only provides a first order approximation to these results. Figure 4.9b also shows output from a two-state Gilbert model (figure 4.10, p=0.083, q=0.823).

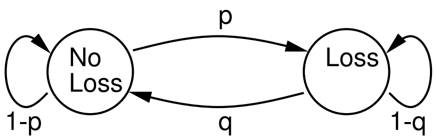


Figure 4.10: Gilbert Model

This is not entirely unexpected given that the loss process measured clearly consists of a congestive loss process which the Gilbert model tracks quite well and an additional periodic failure which the model does not take into account. Modeling this periodic failure using the traffic data rate distribution in figure 4.8b and assuming a 500ms failure every 30 seconds gives the distribution shown in 4.9b as Periodic model. This does go some way towards explaining the tail on these distributions, but does not do so entirely, so some other failure must be occurring fairly infrequently.

## 4.4 Conclusions

Drawing general conclusions from limited data is a process fraught with dangers. We believe that the RTCP data we have presented is reasonably representative of network conditions on the Mbone as a whole. The same general patterns show up in data gathered at different times and (to a lesser extent because the sessions are smaller) data sourced from different locations. We are less certain about how representative the mtrace data is because we do not know sufficient information about the sites to which the mtraces did not succeed. However the loss distribution expected from the mtrace data matches fairly well the loss distribution reported from the RTCP data, so we believe these results too are fairly accurate. The RTP data was only gathered at UCL, and we do not claim it to be representative of the Mbone as a whole. However we do believe that the path to UCL is not atypical. Although this path is longer than the mean (both physically and topologically) the loss rates in figure 4.6 correspond quite well with the median loss rate in figure 4.3. We do not know how prevalent period loss patterns such as those in figure 4.7 are, nor where in the network they arise, so we make no claims about how typical this feature is. The loss produced by this period failure corresponds to 1% - 1.5%, so is only a relatively small proportion of the loss seen at busy times.

Based on tentative confidence in how representative these figures are, we attempt to draw some guidelines for tool designers.

### 4.4.1 Reliable Multicast

Taking the link loss statistics from the mtrace daemon data in figure 4.1 as independent loss probabilities[4], the probability of a single packet reaching all destinations during the time the mtrace statistics were gathered is 0.03. Other sessions monitored produce similar probabilities; for example in the USENIX 97 conference the mtrace tree built from the 17 successful mtraces shows a probability of a single packet reaching all receivers of 0.06.

The accuracy of these probabilities is open to some debate. The mtrace statistics may contain some inaccuracies due to boundary effects of the sampling, but these are thought to be small. Some links have been assumed to have $p_{loss} = 0$, either because the statistics showed some small duplication, or 100% loss indicating that the downstream mrouter didn't collect traffic statistics. In addition, between 60 and 70% of sites were not able to be mtraced, although it is likely that many of these sites would have joined the traced tree nearer to leaves than the root. The assumption of short-term[5] independence of congestion for small variable rate flows is not conclusively shown, but we have circumstantial evidence that it is a reasonable first approximation. We therefore believe these statistics overestimate the probability that a packet will reach all destinations, though we do

[4] We treat links with 100% loss to be mtrace statistic gathering errors and assume no loss occurs on them. Assuming losses on one link are independent of losses on another link is based in part on the lack of correlation between data rates and loss rates shown in figure 4.8b. In addition, experiments with rtpmon on the MICE project implied that this assumption is not unreasonable as a first order approximation for flows that do not constitute a large proportion of the traffic on any individual link. We do not however have firm evidence that this is the case.

[5] long-term independence of congestion points is definitely not the case - congestion is very correlated with the day of the week and the time of day

not know how much of an overestimation this is.

Assuming that these probabilities are not unreasonable, they have great importance for the design of reliable multicast protocols (RMPs). Most current RMPs tend to rely on retransmission of missing packets, and the primary difference between such RMPs is in the mechanisms for scalably deciding who can report loss, and who can perform the retransmission. For example, SRM[24] is based around both timer and probabilistic delay of NACKs, then multicasting NACKS which serve to suppress duplicate retransmission requests. Whilst this ensures that close to one receiver NACKs a particular packet loss, it is not geared well to environments in which a large proportion of packets are lost by at least one receiver. In such situations the majority of packets will be NACKed and retransmitted at least once. In the vanilla SRM described in [24] all NACKs and retransmissions are sent to all receivers, and this may result in very significant retransmission overhead. In [25] the possibility of local recovery mechanisms is discussed. These show promise to help this situation in the case of a small number of particularly congested links by dynamically discovering significant loss points and performing local retransmission across the lossy link to the downstream receivers. However, although our traces do show a small number of particularly lossy links, they also show a larger number of slightly lossy links which can combine to cause a significant additional retransmission load even if local retransmission deals with the significant loss points.

These observations lead us to conclude that packet-level or ADU-level forward error correction (FEC) techniques[63] should be seriously considered by the designer of any reliable multicast protocol. Even when receivers suffering high loss rates are removed from the multicast groups we measured, significant retransmission rates would be required even though no single receiver is in a significant overload state. Packet-level FEC can result in greatly reduced reliability overheads for such multicast applications because the additional traffic for the FEC scheme serves to fix many *different* small losses at each different site. Such FEC schemes can be combined with local-retransmission mechanisms to avoid the need to provide excessive error-protection to all receivers, or varying degrees of protection could be provided on different multicast groups allowing receivers to select the level of FEC required by joining or leaving these groups. Alternatively, FEC packets can be sent on demand, as described in [63].

Note that whether or not to provide FEC is orthogonal to the issue of congestion control, which should be performed irrespective of the reliability mechanism chosen.

### 4.4.2 Bulk Data Reliable Multicast and Congestion Control

One of the major problems in congestion control for reliable multicast is that a sender cannot know on a timely enough basis whether congestion feedback is representative of a problem to only one receiver or to many receivers. If congestion control is performed from time-averaged results from many receivers such as those in figure 4.8, or if the sender performs congestion control based on per-packet NACKs (as described above) then the sent data rate is likely to decrease to zero.

Figure 4.5 shows that loss correlation is fairly readily observable from RTCP-style receiver reports. This suggests a mechanism by which congestion control may be performed for reliable-multicast bulk-data-transfer. Using this correlation data, receivers can be subdivided into separate multicast groups, each of which receive well correlated loss. It does not then matter which receiver sends a NACK because the sender can assume it is significant with reasonable confidence. Thus a representative may be elected to perform undelayed NACKing (although others can send delayed NACKs if required as with SRM) and the sender may perform TCP-style congestion control on the same time-scales that TCP does. This will allow non-zero throughput and achieve a fair-share[6] of the bandwidth in the presence of background TCP traffic. This does mean that the sender must send all the data multiple times, once to each different correlation groups, but this is still much more efficient than unicasting the traffic many times. There are likely to be a small number of receivers for which very high loss rates and a lack of correlation with other receivers mean that unicast is still the best option.

### 4.4.3 Audio and Video

For packet audio traffic without any form of loss protection or masking, the limits to comprehensibility are reached at between 10% and 20% loss[39], although it may become annoying at lower loss rates. Simple error correction techniques[39][6] exist that increase this comprehensibility limit, and make loss much less noticeable at lower loss rates, although the distribution of consecutive losses shown in figure 4.9 mean that noticeable artifacts are likely to occur at lower loss rates than suggested.

However, it is clearly not desirable to keep adding forward error correction to protect an audio or video stream in the face of persistent high levels of congestion as the dropped packets unnecessarily use up bandwidth between the sender and the bottleneck, and more well behaved traffic sharing the bottleneck will suffer significantly.

[6] in as much as TCP receives a “fair share” of the bandwidth

Layered encodings[1] have been suggested as one solution to this problem of heterogeneity, and indeed show great promise as a mechanism to allow different data rates to be delivered to different receivers depending on available resources. However, as figure 4.8 indicates, determining whether the traffic being received is a significant *cause* of loss is not an easy problem in complex scenarios where that traffic represents only a very small proportion of the traffic flowing over the congested links. Thus it is very unclear how techniques such as RLM[57] would fair in such scenarios.

We are forced to conclude that one of the following must be the case:

- 15Kbps is too high a rate for the current Mbone to support, or;
- to multicast audio and video over the current best-effort internet, loss rates of around 10% must be tolerated and that congestion control mechanisms that operate in a background of TCP traffic on significantly longer time-scales than TCP without assistance from router queue management will not work well because loss statistics do not provide useful feedback.

We do not believe the former to be the case although "too high a rate" is not a well defined concept. Typically a "reasonable" rate is regarded as being that which TCP would achieve over the same path. We cannot test TCP performance over the same routes because the DVMRP routing is often set up specifically to avoid congested links, but over similar paths TCP connections do normally achieve a *mean* throughput of greater than 15Kbps. However, whilst TCP adjusts its rate rapidly to lessen congestion when it observes it, adaptive layered mechanisms such as RLM cannot adapt so rapidly, and so must tolerate congestion transients.

Thus for such real-time adaptive streams, it appears that we need support from router forwarding engines such as would be provided by RED[26] and a fair queuing scheme[33][64] before we can design even receiver driven multicast congestion control mechanisms that will accurately sense appropriate network bandwidth.

# Chapter 5

# NTE: A Scalable Multicast-based Shared Editor

The many-to-many model of IP multicast lends itself readily to distributed data applications, where everyone holds all the data and multicasts changes to this data-set to the other participants. Combining multicast with the Application Level Framing approach allows flexible and simple solutions to consistency and reliability problems: only the application has sufficient context to cope with these problems that can occur in a sufficiently flexible manner. With this in mind, we set out to see where IP multicast and ALF might lead the development of a shared editor for the MBone.

In the Internet multicast backbone significant numbers of points in a distribution tree generate small amounts of loss, and these often result in a low probability that a single packet reaches all the receivers in a session. In their experiments with 12 sites, Yajnik[74] finds that retransmission would have been necessary for between 38% and 72% of packets, and this would have been the case for around 95% of packets sent to the much larger multicast video sessions observed in chapter 4 if these had required a reliable multicast protocol. Thus the use of a form of redundancy would appear to be very desirable in designing reliable multicast applications.

These observations also make it clear that there will inevitably be times when there is significant latency between a change occurring and the last site receiving that change. As systems scale, the problem of restricting feedback to prevent response implosions tends to increase these latencies, because even when a site discovers that it is missing some data, it may not be able to immediately initiate any mechanism to correct the situation.

ALF allows us to use application-level redundancy (a specialised form of FEC) to help reduce this problem somewhat. Redundancy does not solve the problem completely: we can not use it for all repairs, but must fall back on more traditional retransmission techniques when it fails. However, it does work in the vast majority of cases and the scheme we will present works best for the changes that are most important for text changes to be quickly understood by other users. When redundancy works, it achieves similar latency to that achieved by the best retransmission techniques with a predictable and much reduced cost in bandwidth.

The observations also suggest that we take a loose approach to consistency, and design towards detecting and correcting inconsistencies when they occur rather than preventing their occurrence in the first place.

In the case of a shared text editor, these network constraints may conflict with user requirements when assessing design goals.

### 5.0.4 Intended Usage Scenarios

The requirement for a scalable shared text editor emerged in 1993 from the MICE project[35]. The twelve partner organisations were scattered throughout eight countries, and to manage the project we held weekly meetings using multicast based conferencing. Typically such a session would involve twenty to thirty participants using audio, video and a shared whiteboard. The whiteboard served as a focus for such meetings, holding the agenda to be discussed, and documenting decisions as they emerged. Often sessions would be left active for days at a time, and the whiteboard would accumulate comments helping people catch up if they missed a particular discussion. Unfortunately the LBL whiteboard, wb[24], was proving to be inadequate for the task for three reasons:

- It did not cope well with the higher loss rates seen in Europe.
- It was not possible for one participant to manipulate another's input, and so the chair could not correct or incorporate written contributions.
- A whiteboard is oriented around drawing functions rather than textual functions which turned out to be our primary use.

Thus the design requirements were to scale to around 30 participants connected by a fairly unreliable and high loss network without suffering significant quality degradation or causing excessive network loading. In practice, we discovered that to satisfy these requirements effectively required using techniques which actually allowed the shared editor to scale to much larger groups of participants.

### 5.0.5 Conflicting Goals

When designing a distributed data shared application such as a shared text editor, the following goals should be satisfied by the dataset distribution mechanism:

- Many users should be able to *manipulate the same data object* over the duration of a session.
- *Eventual consistency* - the dataset should converge on one dataset after a change (though it may be temporarily inconsistent while changes are propagating due to loss or network failures).
- *Deterministic behaviour* - if a user is allowed to modify a data object, they expect it to stay modified.
- *Fully interactive* - users usually don't want to have to wait for locks to be granted to be able to manipulate a data object, as this leads to indeterministic delays due to loss or failures.

Unfortunately these goals are contradictory in a typical internet environment with unpredictable loss and failures.

Different tools choose not to satisfy one of these goals.

- LBL's shared whiteboard, wb, is the only multicast based distributed data application in widescale use. It does not allow different users to be able to manipulate the same object.
- Dissemination applications such as web page multicast [12] or Usenet news feeds[54] only have one data source per group, and so avoid this conflict.
- Traditional distributed applications make use of locking, and suffer performance problems as a result.

Relaxing eventual consistency is not an option, and so if we are to achieve good performance and allow different users to modify the same object over time, relaxing deterministic behaviour seems to be worth exploring. By this we mean that under some circumstances (that we can aim to occur rarely by careful design), the system asserts a change that wasn't what some subset of the users expected. The hypothesis is that we can make this happen rarely enough and provide sufficient feedback or recovery options to the users when it does occur that it will not cause significant problems.

Simply relaxing determinism does not necessarily result in usable applications, and there are additional mechanisms and constraints that are required before the problems caused by this design choice can be said to have been overcome. In particular, achieving global consistency with such a distributed data application is not trivial. We will explore these problems in section 5.1 and propose solutions that aim to minimise their frequency of occurrence and gracefully deal with them when they do occur.

### 5.0.6 Related Work

Most related work is in the areas of computer supported collaborative work (CSCW) and replicated databases. In general, the majority of CSCW work has centered around preventing inconsistencies from arising. We believe that allowing and accommodating temporary inconsistencies is required to achieve good performance. Some work does explicitly allow inconsistencies in the area of so-called “asynchronous” collaboration. Haake and Haake [34] use a versioning system to manage parallel versions, but do not concentrate on subsequent re-synchronisation. Munson and Dewan[62] focus on object merging, but their techniques are applicable only to explicit complete merges in an “asynchronous” environment rather than the “synchronous” environment we are considering where divergence is usually unforeseen, and merges need to be timely, must be performed on an unscheduled basis when the network allows, and cope with partial merges if network conditions dictate it.

More applicable to synchronous conferencing systems are techniques from the area of operational transformation[21][3]. In Grove[21] concurrency control is achieved by transforming operations that arrive out of date based on local context so that they can be executed without disrupting the session. In GroupDesign[3], a simpler model is used which is more similar to that used in NTE. A logical clock is used to timestamp operations, and when an operation arrives out of date, operations triggered by events with a later timestamp are undone, the late operation is performed, and then the later operations are re-performed. Although these systems might seem appropriate for our shared editor, we cannot use their techniques directly here. Both systems make the assumption that there is a reliable underlying transport mechanism, but that due to transmission delays, operations from different sites can arrive out of order. The assumption of a reliable and ordered underlying transport mechanism is one that we cannot make if conferencing systems are to scale to more than a handful of sites in the face of wide area network conditions. We wish to utilise application level redundancy to avoid the need to ensure that every operation needs to be conveyed to every site, and thus keep retransmission overheads comparatively low in the face of the network conditions observed in chapter 4. Also if we wish to be able to apply changes as they appear and are using UDP multicast, then we must be able to cope with changes from the same site appearing out of order. This combination of misordered and missing operations mean that operational transformation techniques cannot be used.

Although the world of replicated databases is driven by the immediate requirement for programs, rather than for humans, a very similar set of properties are identified there. The four so-called *ACID* properties, which are used to categorise transactions in a distributed system are:

- *Atomicity.* Either all or none of the transactions operations are performed. If a transaction is interrupted by failure, then its partial changes are undone.
- *Consistency.* A transaction takes the system from one internally self-consistent state to another.
- *Isolation.* An incomplete transaction can never reveal its partial changes or internal state to other transactions before commit.
- *Durability.* Once a transaction has committed, the system must ensure that the results of its operations will never be lost, independent of any subsequent failure.

With replicated transactions, there is an utmost requirement to satisfy all four of these properties simultaneously. Typically, the system is engineered to a scale that is affordable for a level of replication that gives sufficient availability. The tradeoffs and scaling properties for NTE are such that these are neither all necessary, nor feasible.

In general, although there is a significant body of work in the areas of merging inconsistent data sets, no lightweight shared application *protocols* have so far emerged. The closest work in the internet community is the work on Scalable Reliable Multicast (SRM)[24] in the LBL whiteboard tool, wb, which takes the more restrictive approach of preventing inconsistencies from arising.

## 5.1 Design

To achieve resilience, we adopt a distributed, replicated data model, with every participant holding a copy of the entire document being shared. End-systems or links can then fail, but the remaining communicating sites[1] still have sufficient data to continue if desired. *NTE* uses IP multicast to provide unreliable many-to-many communication at near constant cost to the application, irrespective of the number of receivers. The ALF design principle suggests that data should be transmitted on the network in units which can be utilised independently of other application data units. This is exactly what is required for multicast based shared applications, so that loose consistency may be

[1] In this chapter we use the term *site* to indicate a single instance of the application, wherever it is located

maintained, and data can be presented to the user as soon as it is available. Permitting this heterogeneity is essential for the application to scale in a robust manner. The application can then handle reliability and consistency issues as it sees appropriate depending on the application context.

### 5.1.1 Application Data Units

NTE's data model is determined by both by user requirements and by network requirements:

- The network requirements are to introduce application-level redundancy to cope with uncorrelated packet loss, to explicitly tolerate inconsistency, and to divide the data set up into parts that are as independent as possible to reduce potential consistency conflicts caused by failures and partitioning.
- User requirements dictate that several users must be able to work on the same document simultaneously. Also as a shared editor is as much used as a communications channel as it is as a document editor, there is a particular need to be able to keep annotations separate from the primary text being worked on.

These requirements suggest a hierarchical data model, based around blocks of text, each consisting of a number of lines of text. Each block is independent of other blocks - it can overlap them if required although this does not aid readability. An example of blocks used for annotation, is given in figure 5.1.

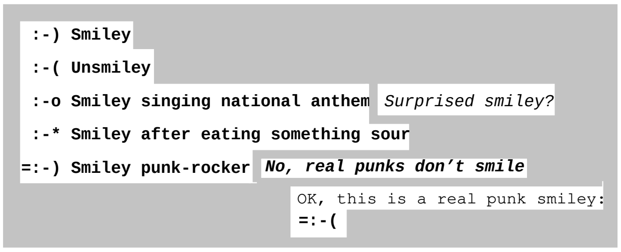


Figure 5.1: An example of blocks of text used for annotation

Most annotations will not be modified by multiple users simultaneously, and so this allows a number of users to be working simultaneously on the document in separate blocks. However, restricting users to simultaneous *annotation* of documents would impose too great a constraint on potential usage modes, and so each line of text is also a separate entity, allowing users to be working on separate lines in the same block.

Line ADUs can provide a large amount of natural redundancy that greatly aids in coping with uncorrelated packet loss because of the nature of modifications in a text editor. This is because it is usually unnecessary to receive all the individual changes to the line as a user types - the most recent version of the line is sufficient. A receiver suffering packet loss while someone types will see some changes as they occur and not see others until slightly later, but the later changes convey sufficiently the textual information[2] of the earlier changes so that the editor's use as a communication tool is not impeded.

When a line is transmitted, it carries the id's of the previous and next lines and the id of the block it forms a part of. Although lines and blocks are not completely independent, blocks can be moved without modifying the lines contained in the block, and lines can be created, deleted and edited independently of other lines or blocks. There are however a number of desirable operations on lines that cannot be carried out independently. We discuss these and their consequences briefly in section 5.4.

### 5.1.2 Distributing the data model

The choice of a line as the ADU was in part due to the simple observation that most consecutive changes are made to a single line because a user continues to type. If the whole line is sent for every character typed, the additional overhead is not great[3], the data transfer is idempotent, and a great deal of "natural" redundancy is available - so long as a user continues typing on the same line, lost packets are unimportant. However this is only the case if we take a loose consistency approach with changes displayed as soon as they arrive, irrespective of whether other sites have received them.

In order to be able to use this "natural redundancy" property, it must be possible to identify whether a version of a line that just arrived is more recent than the copy of it we have already. This is necessary to cope with misordered packets from a single source, and to cope with retransmitted information from hosts with out of date versions of the data.

If we assume synchronised clocks at all sites, recognising out of date information is achieved by simply timestamping every object with the time of its last modification. Copies of objects with out of date timestamps can be ignored at a receiver with a later version of the same data. If we wish to take advantage of redundancy by not requiring retransmission of many lost packets, a receiver must

[2] the result is that with high loss, text appears in small bursts and some typographic errors immediately corrected are never seen at the receiver

[3] For example, the typical IETF RFC has around 40 characters per line. When being created (modified) the mean line length will therefore be 20 (40) characters. An NTE packet requires 28 bytes of IP and UDP header and the NTE line header comprises 50 bytes to ensure that the line is idempotent. Thus for document modification, the redundancy comprises about 33% of the packet, for creation it comprises about 20%, and less still for annotations which tend to be very short. This only comprises redundancy when a user is actually typing - for file loading this data is not redundant.

not care if it receives all the changes to an object as they happen; rather it only needs to receive the final version of an object, although receiving changes as they happen is desirable.

In practice, we can't assume synchronised clocks, but we can implement our own clock synchronisation protocol in the application. This is described in section 5.2.

There are alternatives to this mechanism, including maintaining a change log with each object, but they do not help greatly. Either they require locking, or they suffer from the same merging of changes problem that timestamping suffers from without significantly helping solve the problem.

### 5.1.3 Reliability Mechanisms

Due to the redundancy inherent in the data distribution model, NTE will sometimes perform reasonably well with no mechanism for ensuring reliability. However, there are also many situations where this is not the case, and so we need a mechanism to detect and repair the resulting inconsistencies.

Inconsistencies may result from:

- Simple packet loss not corrected by subsequent changes (particularly where the last change to a line has been lost, or where data was loaded from a file)
- Temporary (usually bi-directional) loss of large numbers of modifications due to network partition.
- Late joining of a conference.
- Effectively simultaneous changes to the same object.

#### Inconsistency Discovery

Unlike the mechanisms used in SRM [24] and INRIA's whiteboard [13], inconsistencies due to simple packet loss cannot be discovered simply from a gap in the packet sequence number space as we wish most such changes to be repaired by redundancy, and therefore do not need to see a copy of every packet at a receiver.

Instead we use a mechanism that ensures inconsistencies are resolved, irrespective of the number of packets lost. There are three parts to this inconsistency discovery scheme.

The primary mechanism uses a concept of the current site - this is the site which has most recently been active.[4] The current site periodically multicasts out a summary packet giving the

[4] If more than one site is active, any of those sites can be chosen as current site. If two sites both think they are the current site, the one with the lowest IP address stops sending.

timestamps and IDs of all the most recently changed objects. If a receiver has a different version of one of these objects then it is entitled to either request the newer version from the current site, or to send its newer version. In principle, the current site could simply send the actual data for the most recently changed objects, and thus not require the receivers to request this data, but this would only be worthwhile if the probability of a receiver missing the current version of each of the recently changed lines was high enough. In any event, a mechanism for the current site to discover that its version of one of these objects was out of date would still be required, and so sending this "partial index" accomplishes both tasks and uses minimal bandwidth.

The current site may change at the end of each retransmission round (see section 5.3.1) each time a new user modifies the document; however the rate that these summary packets are sent is a constant whilst any users are modifying the document - a new current site simply takes over from the previous one. Once a document becomes quiet, the rate of sending summary packets is backed off exponentially to a low constant rate. Having the current site switch between active sites is desirable as it ensures that all the active sites get a degree of preferential treatment in the inconsistency resolution process.

In addition, we use two mechanisms designed to detect inconsistencies missed by the primary mechanism. These are based on the session messages each instance of the application sends periodically[5] to indicate conference membership. To detect inconsistencies, each session message carries two extra pieces of information - the timestamp of the most recent modification seen, and a checksum of all the data. If the timestamp given by another site is later than the latest change a receiver has seen, the receiver can request all changes from the missing interval without knowing what that data actually was. This may not fill in sufficient information to ensure consistency, and so the checksum is a last resort to discover that a problem has occurred. This is followed by an exchange of checksums to discover which blocks the differences are in, and then a summary of the line timestamps in the inconsistent block.

An alternative to sending explicit summary packets from the current-site might be for session and data packets to have an additional object ID and its modification timestamp added to them, and for all sites to take turns to report the state of the most recently modified objects in a distributed manner. The choice of sending summary messages from the current-site rather than using a distributed approach was made because, with our architecture, the current-site also regulates the retransmission process (section 5.3.1) and so it is more natural for it to list the status of its data set. Typically the

[5]These session messages are sent by each site with a rate that is dependent on the total number of sites in the conference, so that the total session message rate is constant and low.

current site has the most up-to-date data as it has been most recently active. If the current site has out-of-date data for any reason, this also ensures it is updated fastest, which is in keeping with it also being the most active site. If we had adopted a symmetric retransmission architecture such as SRM, distributed summaries would probably have been more suitable.

## 5.2 Clock Synchronisation

Given that all changes to a document are multicast and are timestamped, we have a simple mechanism for clock synchronisation amongst the members of a group:

- If a site has not sent any data and receives data from another site, it sets its application clock to the timestamp in the received message.
- If a site has not received any data, and needs to send data, it sets its application clock to its own local clock time.
- If a site receives a message with a timestamp greater than its current application clock time, it increases its application clock time to match that of the received message.

These rules are illustrated in figure 5.2 and ensure that all sites' application clocks are synchronised to within one round trip time which is sufficiently accurate for our purposes. More details of this clock synchronisation mechanism are given in [43].

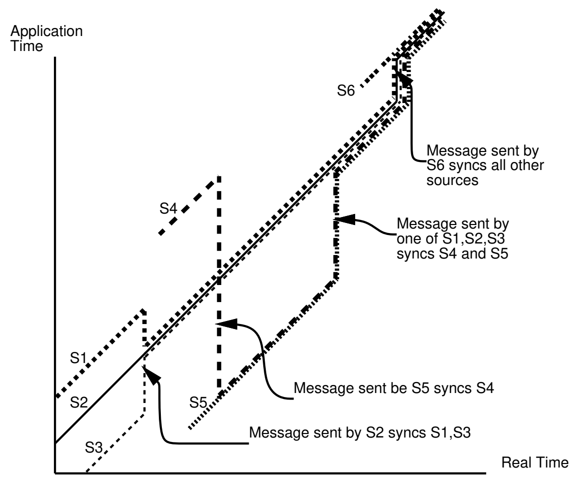


Figure 5.2: Application based clock synchronisation

## 5.3 Scalable Retransmissions

When a receiver discovers there is an inconsistency between its data and that of another site, it cannot just send a message to resolve the inconsistency immediately because there is a high probability that its message would be synchronised with identical messages from other receivers and cause a NACK implosion. Instead we require a mechanism to ensure that approximately one site sends the message. If this message is an update, it should be multicast as this may update other sites with out of date information. If this a message is a retransmission request, it should also be multicast, as then the reception of the request can be used at other sites to suppress their retransmission requests.

SRM[24] uses a mechanism by which retransmission requests are delayed by a random period of time partially dependent on the round-trip time between the receiver and the original source. Requests are then multicast and serve to suppress further duplicate requests from other sites. To work most effectively, this requires all participants to calculate a delay (round trip time) matrix to all other sites, and this is done using timestamps in the session messages.

As it has no redundancy mechanism, SRM's wb implementation is more dependent on its retransmission mechanism than NTE is, and thus it requires its retransmission scheme to be extremely timely. NTE does not wish its retransmission scheme to be so timely, as it expects most of its loss to be repaired by packets sent as the next few characters are typed with lower latency than SRM would achieve. This results in very significantly fewer packet exchanges because in a large conference on the current MBone, the probability of at least one receiver losing a particular packet can be very high. Thus what we require is a retransmission scheme that ensures that genuine inconsistencies are resolved in a bounded length of time, but that temporary inconsistencies due to loss which will be repaired anyway do not often trigger the retransmission scheme.

SRM can be tailored for redundancy by adding a “dead time” to the retransmission timer to allow a window during which the next change could arrive. If we used SRM's randomised time based scheme, then we might have opted for not sending summary messages but instead adding ID/timestamp pairs to the session messages as described above. These would then tend to spread the retransmission requests more evenly over time.

At the time NTE was designed and implemented, SRM's mechanism had not been described in detail, and we used a different sender driven retransmission request scheme. For many purposes we believe SRM is superior, but in the context of NTE there is little to choose between them most of the time. However, in NTE, inconsistencies can result from latency, and so it is important that active sites are updated faster than passive sites, and NTE's sender-controlled scheme does help here by ensuring that the current site (which rotates between the active participants) gets updated fastest.

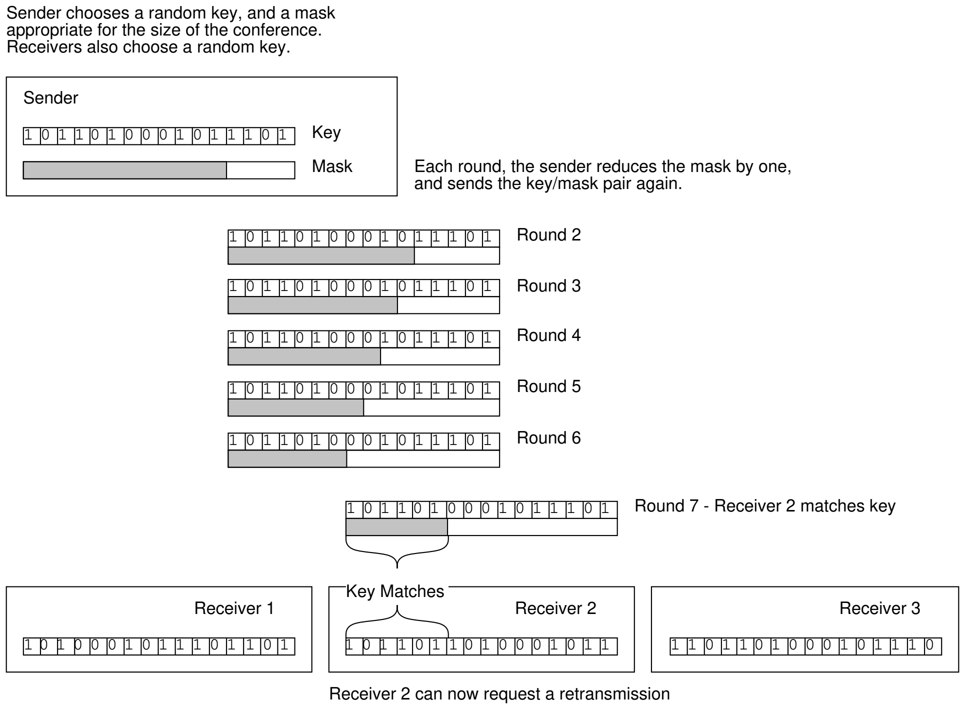


Figure 5.3: Sliding Key Triggered Retransmission Requests

### 5.3.1 Sliding Key Triggered Retransmissions

When the current site sends a summary packet, it starts upon the process of sending a sequence of keys. When a receiver matches a key sent by the sender, it can immediately send its retransmission request (which can be many objects if necessary) along with the key that was matched. On receiving this request, the sender then starts the retransmission of the missing data.

The sender generates a random integer (key) when it creates its summary message. Upon receipt of the summary message, the receiver also generates a random key. Then the sender sends its key along with a key mask which indicates the bits in the sender's key that must be matched in order for the receiver to send a retransmission request. This key/mask pair is sent several times, and if no retransmission request is forthcoming, the bits indicated by the mask are reduced by one, and the key/new-mask pair is sent again. If no retransmission request is forthcoming by the time the mask indicates no bits need to be matched, then the process is started again with a new random key, a new summary report, and possibly a new current site. If no change has occurred since the previous summary report, the rate of sending sliding keys is reduced to half the rate for the previous round until it reaches a preset lower rate limit. This process is illustrated in figure 5.3.

This is loosely based on a scheme devised by Wakeman[4] for congestion control in multicast based adaptive video.

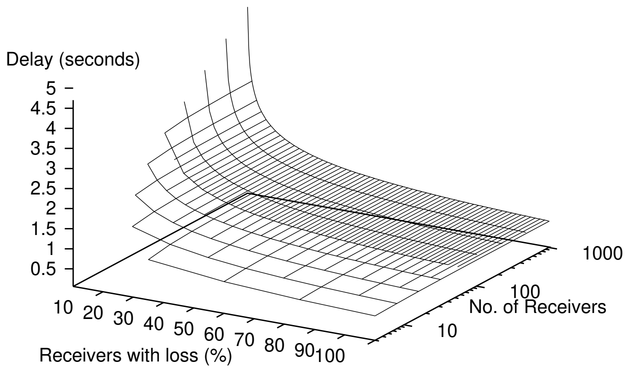


Figure 5.4: Expected delay before a retransmission request (RTT=250ms)

As the session messages give a reasonable approximation of the size of the conference at the point when we generate the summary message, the sliding key can be started close to the point where it would be expected to elicit the first response if all receivers need a retransmission. The delay then before receiving a retransmission request scales $O(log(n))$ where $n$ is the number of participants. This is shown in figure 5.4. For a typical 1000 way conference, where only one receiver requires a retransmission, with each key/mask pair sent twice per round and an estimated worst case RTT of 250ms, this results in 4 (small) packets per second and a maximum delay of 5 seconds before requesting retransmission. If the conference was smaller, or more sites had suffered loss, this time would be reduced.

Simulation results and real world experience show that the mean number of responses is not dependent on number of receivers suffering loss or session size, and it is typically around 1.4 to 1.5. However, loss *rates* do affect the number of responses, because sliding key messages start to be lost, and so several receivers that would match the same key do not always all see the same transmission of that key, and key retransmission stops once one receiver has responded. This actually serves to *decrease* the number of responses by a small amount as shown in figure 5.5.

## 5.4 Inconsistency Avoidance Mechanisms

Using a data model and distribution mechanism as described above does not guarantee that inconsistencies can be detected and corrected - it is possible for two sites to have internally consistent

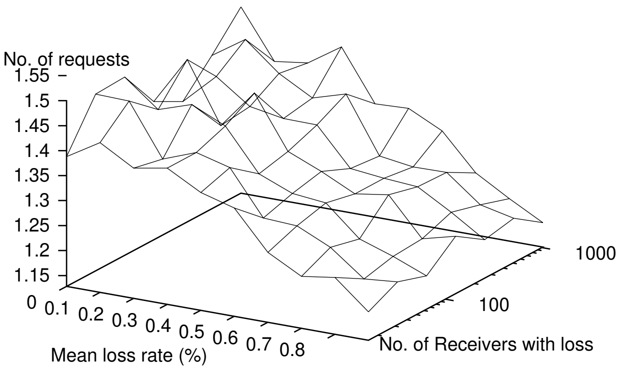


Figure 5.5: Simulated number of retransmission requests with two transmissions of each sliding key mask

state, and mutually inconsistent state, and not to be able to resolve their states into any useful mutually consistent state. To prevent this happening, a set of rules for manipulation of the data set is required so that when eventual consistency is restored, the data is likely to be semantically meaningful. These consistency rules are described in some detail in [43], but we summarise them here for completeness.

To ensure eventual consistency after the resolution of one or more network partitioning:

- Deletion must always override modification, irrespective of the timing of the two operations.
- Deleted items must be transmitted to and stored at all sites to prevent re-assertion.
- Deleted lines must be kept in their original place in a block's list of lines, and must remain referenced by their neighbouring lines - this is a precondition for simultaneous insertion detection.
- Moving of text must be performed by deleting all the original lines containing the text to be moved and inserting the same text as new lines, thus preserving the original line ordering.
- If all the above are performed, line ordering within a block is only changed by adding new lines. This makes simultaneous insertion of lines during a network partition a detectable and resolvable situation.

These restrictions will ensure that eventual consistency is achievable, even in the face of continuing modification during a network partitioning. However, they are not always sufficient to ensure that the contents of the document eventually converge on what all of the users actually wish it to be.

Although this could be achieved by locking of blocks to ensure that only one person can modify the a block at a time, we believe this restriction is often unnecessary and would restrict usage of the editor. Indeed, under many circumstances, the usage patterns of the editor are likely to be such that large scale simultaneous editing of a block during a network partitioning will not happen because the vocal discussion needed to do so will not be possible. If users are concerned about simultaneous editing of a block during a network partition, they should checkpoint the block to ensure no unseen changes can be made to it. For paranoid users, this check-pointing procedure could be automated, although we do not believe this is normally desirable or necessary.

## 5.5 Summary of Reliability Mechanisms

As described above, several mechanisms come together to provide reliable and scalable transmission of the data set in NTE:

**Redundant transmission of lines of text.** As most consecutive changes are to the same line, sending the whole line each time eliminates the need for the majority of retransmissions during editing.

**Inter-object references using globally unique names.** Each packet containing an object references its neighbouring objects by application-level name, allowing the discovery of missing objects, and a retransmission request to be made for them by name. This also allows inconsistencies caused by partitioning to be detected.

**Periodic announcement of recent changes.** A list of the most recently changed object names and timestamps is multicast periodically. The primary purpose is to detect inconsistencies caused by missing the last change to a line, and to cope with the last in a sequence of block-position changes for which there is no redundancy, but this also serves to assist recovery from short-lived partitioning.

**Periodic announcement of checksums using session messages.** Each site periodically sends a session message indicating receiver liveness, and also the timestamp of the most recent change seen, and a checksum of all the blocks. The timestamp provides a low cost mechanism to discover some common inconsistencies and request update by range of timestamp rather than object name. As a last resort, the checksum is used in a recursive checksum exchange to initiate retransmissions needed after a long-term partition has been resolved.

**A sliding-key triggered retransmission request and reply mechanism.** Apart from the redundancy which repairs most loss, all retransmissions to repair inconsistencies use this mechanism to control who can send, and therefore control network loading. Objects are normally requested by name, but can additionally be requested by range of modification time, which under some circumstances can significantly reduce update delays.

With the exception of the request-reply mechanism, these basically fall into two categories - open loop transmission of changes as they occur (including redundant data), and open loop periodic transmission of meta-data so that inconsistencies not cured by the redundancy can be resolved. These mechanisms are all aimed at implementing the approach that the application will handle inconsistencies rather than prevent then. We therefore work under the expectation that other sites have a different data-set, and these mechanisms attempt to discover and rectify those differences as network conditions permit.

## 5.6 Using NTE

NTE was first written during the summer of 1994, but due to an incomplete consistency model was not properly usable until the summer of 1995, when it was put in the public domain. It has been used extensively within the MICE and subsequent MERCI multimedia conferencing projects for weekly meetings over the course of two years, and has proved very successful in the environment for which it was designed - that of sessions with twenty to thirty people. During such sessions there is usually a main conversation taking place using the audio channel, and being documented using NTE. In addition, simultaneous side conversations often take place using NTE because someone wants to make a comment that is not directly related to the main session, and so it has often been the case that four or five users are typing in NTE simultaneously. The GUI provides a visual indication of who is making a change at the point in which the change is being made, and also colour-coded in a mini-map (see figure 5.6) which prevents confusion about who is doing what.

The robustness of NTE to bad network conditions can be noted from the observation that it occasionally becomes the only communication channel with sites that are suffering very high loss rates, where even the redundant audio streams from the RAT audio tool[39] are incomprehensible, and wb has given up resending due to repeated failed retransmission attempts. Even under these circumstances NTE's retransmission rate is observed to be very low. The exception to this is when files are loaded where there is no redundancy mechanism. Such a mechanism could be added, for example using packet-level FEC, but file loading is not inherently rate-based whereas typing has a natural rate and is used as a communications channel requiring low latency. Adding additional redundancy protection to file-loading operations without an effective congestion control scheme[6] would tend to cause people to use NTE to transfer files across badly congested networks, which is almost certainly undesirable.

NTE is also often used by other communities on the MBone. Examples are that it was used in the collaborative writing of the PIM multicast routing specification between a handful of geographically distributed sites, and is used internally in the EECS department at Harvard as a persistent notice and chat board, where the ability for someone to clean up out of date information written by other users means it tends to not get into the same sort of mess that occurs with wb after a long period of time.

## 5.7 Conclusions

NTE, its data model, and its underlying protocol were all designed to solve one specific task - that of shared text editing. We used general design principles - those of IP multicast, light-weight sessions, and application level framing as starting points. However, the application data model is intended only for text. The data distribution model uses the redundancy achieved through treating a line as an ADU combined with the fact that most successive modifications are to the same line to avoid the need for most retransmissions.

Although NTE is not general purpose, the restrictions the data distribution model impose on a data structure consisting of a ordered doubly linked list of application data units can perhaps be generalised somewhat. The imposition of a strict ordering of ADUs, combined with marking deleted ADUs whilst leaving them in position in the ordering, allows the detection of inconsistencies caused by network partitioning in a loose consistency application.

The stacking order of blocks in NTE is a local issue so that overlapping blocks can be edited. In a shared drawing tool, stacking order is a global issue, allowing the overlaying of one object over another to produce a more complicated object. In such a tool, each drawing object (circle, polygon, rectangle, line, etc) is the drawing equivalent of a line of text. The concept of a block does not exist as such, but there is a strict ordering (analogous to line ordering in a block) which is imposed by the stacking order. Thus, the same set of constraints that apply to lines of blocks in NTE should also be applied to the stacking order of drawing objects. We believe many shared applications have similar requirements.

The retransmission mechanism used in NTE is novel. For many applications, SRM might be

[6] no multicast congestion control scheme had been shown to work at the time NTE was implemented

a more appropriate choice of retransmission mechanism, as, given a stream of packets with sequence numbers, it is likely to be more timely. However in NTE, retransmission is a relatively rare phenomena due to redundancy which achieves lower latency at low cost. We believe that all reliable-multicast applications should attempt to use some form of redundancy to lower the retransmission load, so vanilla SRM does not seem to us to be the best choice. The SRM timer algorithms could be used in a redundant environment, and this seems to be a good general purpose solution. However in applications like NTE, where consistency is an issue, our sender-initiated scheme used in the context of a current site means that active sites get updated first, which is useful to avoid consistency conflicts arising from increased latency.

NTE's retransmission scheme also has some additional benefits in the context of other applications. Although we do not use the property in NTE, sliding key schemes can be used to ration retransmission requests - this might be useful where the reverse path from receivers to senders is bandwidth limited. In addition, for multicast networks that only support one-to-many multicast with a unicast back channel such as some satellite networks, a sender initiated retransmission request scheme is required.

### Lessons Learned

So many parts of the NTE design are interwoven that the task of extracting lessons learned is made more difficult.

We note that the general *design principles* of ALF and Lightweight Sessions seem to point towards solutions that are distributed, fault-tolerant and robust because they tend to enhance the designer's awareness of the potential failure modes more than is the case with more traditional layered designs.

There are clearly many mechanisms for discovering that data is missing and arranging for its retransmission in a manner that does not result in request or response implosions. This chapter presents one such method; SRM presents another very different method; both have a similar architectural view of the world consisting of distributed data applications. To a first approximation it does not matter which mechanism is chosen, but *secondary issues* such as desired delay and minor architectural constraints such as NTE's desire to update active sites faster than inactive sites may have an effect of the solution chosen.

The use of *redundancy* in reliable multicast applications appears to be very desirable to achieve good performance and scalability when network conditions result in significant spatially uncorrelated packet loss. Although very effective mechanisms exist for using packet-level forward error correction[63], these are best suited to bulk transfer applications rather than interactive shared applications. In the latter, application level redundancy provides a very effective low latency technique for reducing the load on the retransmission mechanisms and hence reducing network load. The mechanism described in this chapter is very simple (although its side effects are less so), and much more subtle variations can be envisaged that reduce this redundancy load and still provide effective protection against the sorts of uncorrelated loss described in chapter 4.

The most difficult part of the NTE design turned out not to be reliable multicast, but the consistency control mechanism. The approach presented here, of constraints on the data representation and manipulation operations, can be generalised to other applications. We believe that this does achieve eventual consistency, but have no formal proof of this. However, we do believe that a relaxed consistency approach similar to the one we have taken is *essential* to achieve good scalability and performance in the face of real-world unreliable network conditions. Clearly this is an area for future work.

## 5.8 Future Work

The consistency mechanisms implemented in NTE can be generalised as described above, and combined with more traditional deterministic mechanisms and SRM in a single reliable-multicast framework for building shared applications. The goal is that different applications and indeed different objects within the same application can require different reliability modes at different times by relaxing different constraints from section 5.0.5.

One area that NTE does not address adequately is the issue of congestion control. Although NTE maintains a bandwidth budget, and shapes its transmissions within this budget, no current mechanism allows us to set this budget effectively and NTE operates with a default budget of 8Kbps. We are investigating mechanisms to allow better congestion control mechanisms within the extended framework to allow higher bandwidths to be used safely when conditions allow.

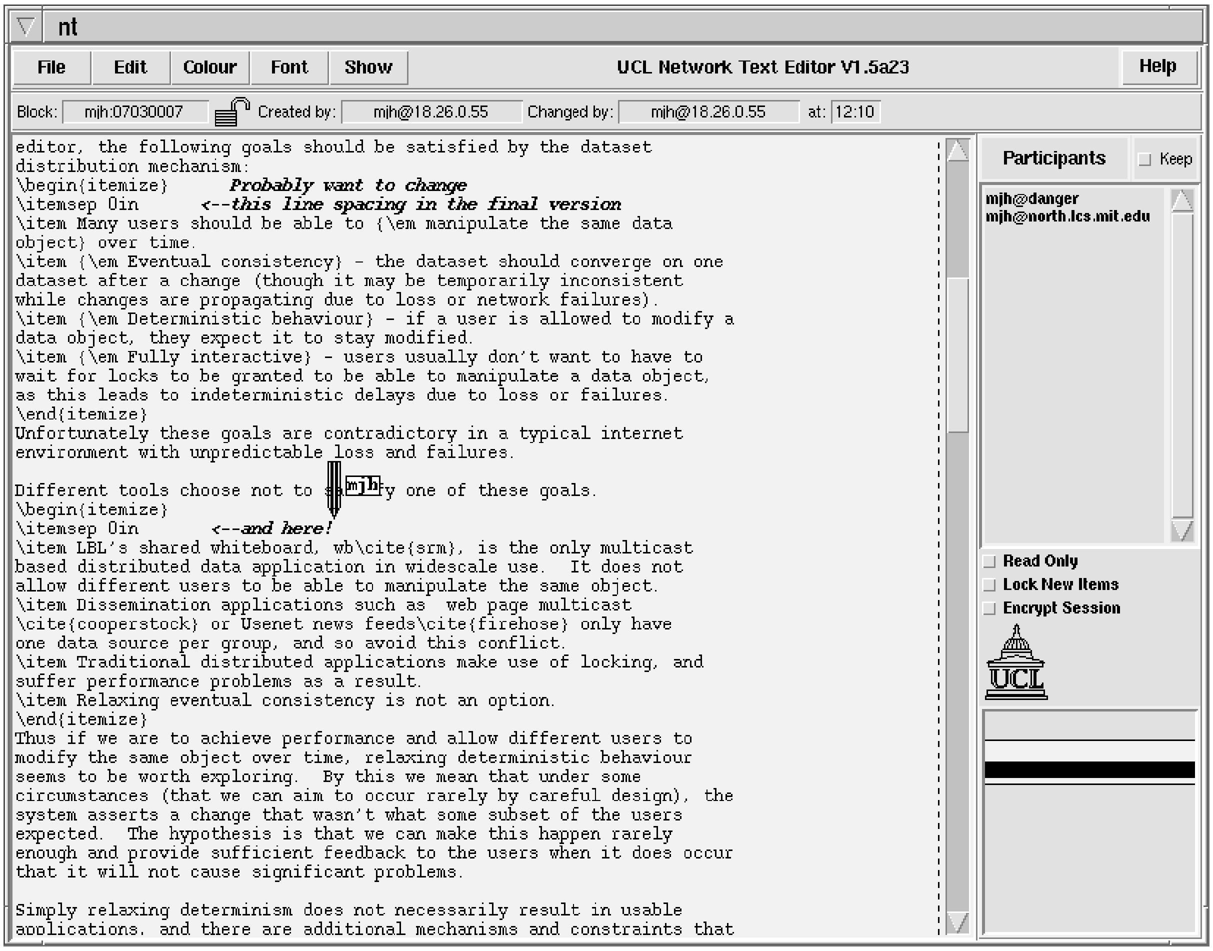


Figure 5.6: The Network Text Editor

# Chapter 6

# Multicast Session Directories and Address Allocation

A multicast session directory is a mechanism by which users can discover the existence of multicast sessions, and can find sufficient information to allow them to join a session.

A multicast session is minimally defined by the set of media streams it uses (their format and transport ports) and by the multicast addresses and scope of those streams. Additional information such as encryption keys may also be necessary to participate in a session, and description or categorisation information is required to distinguish desirable sessions. A session directory distributes this information so that receivers can decide which sessions they would like to join.

In this chapter we shall spend a little time examining the architecture of a multicast session directory, but shall largely concentrate on the issue of multicast address allocation which a session directory needs to perform in order to create new sessions. Multicast address allocation has traditionally been performed by the session directory itself based on the information about existing sessions. Thus session announcement messages have also served as multicast address reservations - a dual purpose that is efficient, but which may cause some side-affects as session directories scale. It is these scaling issues that we will examine in this chapter.

## 6.1 Requirements

From a design point of view, the two most important properties of multicast session directories are robustness and timeliness.

#### Robustness

If a session directory fails, it will prevent all the sessions utilising that directory from being joined. Thus robustness to failure is a critical design feature. Ideally, the only conditions where it is permissible for a session directory to fail are when the same failure will also prevent the sessions described by that directory from occurring. Specifically, the session directory used to initiate a session should not significantly increase the possible failure modes of that session. In particular, centralised approaches to directories are not suitable unless a high degree of replication can ensure sufficient robustness.

From an address allocation point of view, a primary requirement is that an application requiring the allocation of an address should always be able to obtain one, even in the presence of other network failures.

#### Timeliness

There are two basic modes for using session directories:

- Advertise a session in advance of its start time.
- Advertise a session, and immediately join it.

For advance sessions, there is no timeliness requirement, other than that the session description must have been conveyed to all potential participants by the start time of the session.

However, for immediate sessions, it is necessary to be able to announce the session and expect users to be able to receive the session description and join the session immediately. For large lightweight sessions, where the participants are not known in advance, and where the potential participants may not know of the session in advance, this is somewhat problematic.

For address allocation, a short delay of a few seconds is probably acceptable before the client can be given an address with reasonable confidence in its uniqueness. If this is an advance session, the address should be allocated as soon as possible, and should not wait until just before the session starts. It is acceptable to change the multicast addresses used by an advance session up until the time when the session actually starts, but this should only be done when it averts a significant problem such as an address clash that was discovered late.

**Availability, Correctness, and Address Space Packing**

A multicast address allocation scheme would like to be always available, and always able to allocate an address that can be guaranteed not to clash with that of another session. However, to *guarantee* no clashes would require a top-down partitioning of the address space, and to do this in a manner that provides sufficient spare space in a partition to give a reasonable degree of assurance that an addresses can still be allocated for a significant time in the event of a network partitioning would result in significant fragmentation of the address space. In addition, to provide backup allocation servers in such a hierarchy so that failures (including partitioning of a server and its backup from each other) do not cause problems would add further to the address space fragmentation.

Given that we cannot achieve constant availability, guarantee no clashes, and achieve good address space usage, we must prioritise these properties. We believe that achieving good address space packing and constant availability are more important than guaranteeing that address clashes never occur. What we aim for is a high probability that an address clash does not occur, but we accept that there is a finite probability of this happening. Should a clash occur, either the clash can be detected and addresses changed, or hosts receiving additional traffic can prune that traffic using source-specific prunes available in IGMP version 3, and so we do not believe that this is a disastrous situation.

Thus tolerating the possibility of clashes is likely to help address allocation scale to allocating a very high proportion of the address space in the presence of network conditions such as those observed in chapter 4. This is the “relaxed consistency” design principle discussed in chapter 1 that we aim to show is feasible.

## 6.2 System Model

A multicast session may take many forms ranging from a single sender and huge numbers of passive receivers through to a highly interactive session with a handful of participants. Whatever form the session takes, there is a need to discover that the session is going to take place, to discover the details required to participate in the session, and to allocate the multicast addresses to be used. A multicast session directory must perform the first two tasks, and can also perform the latter task.

Of critical interest both for session announcement and for multicast address allocation is the scope of sessions - which part of the network the data from the session will reach. There are two mechanisms for scope control in the Mbone: TTL scoping and administrative scoping. We describe these in section 3.4. In this chapter we concentrate primarily on TTL scoping, as this is the principle mechanism in use today.

For a session announcement, the primary scoping requirements are that the session announcement is heard at all the places where the data for the session can be received, and that the announcement is not heard in places where the session cannot be received. These requirements are most easily satisfied by simply multicasting session announcements with the same scope as the session they describe. In principle, with misconfigured or misused TTL scoping or with misconfigured administrative scoping, such a region is not well defined, as different sources within the same intended scope region may have different scopes. An example of such a problem would be sending with a TTL that is not just under the threshold for the intended scope region, because the data from different sources within the outer boundary might be dropped at different internal boundaries because the TTL decremented too far. However, with correctly configured administrative scoping and correctly configured and used TTL scoping, this is not in fact a problem.

For multicast address allocation, the primary scoping requirement is that no multicast address is allocated in such a way that the session using it (and hence the session announcement) can clash with the same address being used by another session. Here, TTL scoping and administrative scoping give us significantly different problems.

Administrative scoping is a relatively simple problem domain, in that barring failures, two sites communicating within the scope zone are be able to hear each other's messages, and no site outside the scope zone can get any multicast packet into the scope zone if it uses an address from the scope zone range.

TTL scoping suffers from an asymmetry problem - an address, either in use or being announced with the same scope as the session it describes, will not be detected outside the scope zone, but sites outside the scope zone can use the same address to get data into the scope zone. This makes multicast address allocation for TTL scoping hard. We would like to be able to use the same multicast address in multiple non-overlapping scope zones as the address space is limited and we envisage a large number of locally scoped multicast sessions will be in use, but when choosing an address we cannot be sure that it is not in use behind some smaller TTL threshold that would clash with the session we are allocating the address for.

This chapter presents a range of solutions to the problem of allocating addresses within the context of TTL scoping. The solutions do not prohibit the use of administrative scoping; indeed the simpler solutions work well for administrative scope zone address allocation. However, as efficient address allocation for TTL scoping is the harder problem we shall initially concentrate on the issues it raises.

## 6.3 Background: Multicast Session Directories

### The LBL session directory

The LBL Session Directory, sd, was used on the Mbone for several years to advertise multimedia sessions and communicate the conference addresses and tool specific information necessary for participation in the sessions. Sd combined a number of tasks into one tool:

- a graphical user interface for creation of new sessions
- multicast address allocation
- multicast distribution of the session description
- reception of multicast session descriptions from other sites and the local caching of these descriptions
- a graphical user interface to allow users to browse available sessions and to start the relevant tools to join the sessions.

Session descriptions in sd were ASCII text based in a compact format to reduce the bandwidth required to communicate them. No description of the sd session description protocol exists, but a second generation version of the protocol is described in [40].

Sd's model of distribution of session descriptions was to periodically multicast them to a well-known multicast address. Each session description message was multicast with the same TTL as the session it described which ensured that users receiving the session announcement are within the scope of the session itself. The rate of multicasting a particular session description was determined by the length of the session description, by the TTL of the session and by the number of other sessions being announced, such that the bandwidth used by session announcements was kept constant at any particular TTL and each session advertised at that TTL received an equal share of the available bandwidth.

Van Jacobson has partially described [51] a multicast address allocation algorithm called Informed Partitioned Random Multicast Address Allocation (IPRMA) which was intended to be used in sd but never tested. IPRMA depends on having a good knowledge of the sessions currently advertised to minimise the probability of an address clash, and this is the primary reason why sd combined multicast address allocation with the reception of session descriptions. We devote most of this chapter to an analysis of multicast address allocation in the context of a session directory as this is the primary scaling limitation for such tools.

Since 1995, the author's session directory, sdr, has replaced sd as the multicast session directory in use on the Internet multicast backbone.

**Sd performance.**

If all potential recipients of session descriptions keep sd running constantly, then sd's session announcement protocol (simple constant bandwidth periodic announcement) works well for advance advertised sessions, as periodic multicast over an extended time-scale is very reliable. For immediate sessions, sd's announcement mechanism works well in a low loss environment (it satisfies the timeliness criteria at very low cost) but at sites which encounter packet loss, it satisfies the timeliness criteria a little less well.

Unfortunately potential recipients of session descriptions do not all keep sd running constantly. To partially address this, sd saves a cache of descriptions between invocations so that long lived sessions are available immediately upon startup. However, this cache is per user, so sd users often suffer long delays upon sd startup before new sessions are received. Sharing this cache between users would help address this somewhat but this is not done in sd. These startup delays also adversely affect multicast address allocation, which depends on having a large sampling of the sessions currently advertised.

Sd also suffers from a lack of authentication mechanisms, so it is possible to subvert or make fake announcements. Sdr attempts to prevent this through the use of cryptographic authentication mechanisms[41].

### 6.3.1 SDP - an enhanced session description protocol

Since sd was released, some of the requirements of the session descriptions themselves have changed slightly. In particular these changes include the increased use of multiple multicast addresses where some receivers may not want to receive all the media streams associated with a session due to bandwidth constraints, and the need to specify more detailed timing information to aid session scheduling. As a result of these changes, sdr implements an extended session description description format, which is specified as the Session Description Protocol (SDP)[40].

The design of SDP is not of direct relevance to this chapter, except that it provides sufficient information to allow a receiver to join a multicast session without requiring any intermediate stages. However, when encrypted SDP messages are exchanged, an out of band key exchange mechanism is also required to allow decryption of such session descriptions before the session can be joined.

An example of an SDP description is shown in figure 6.1:

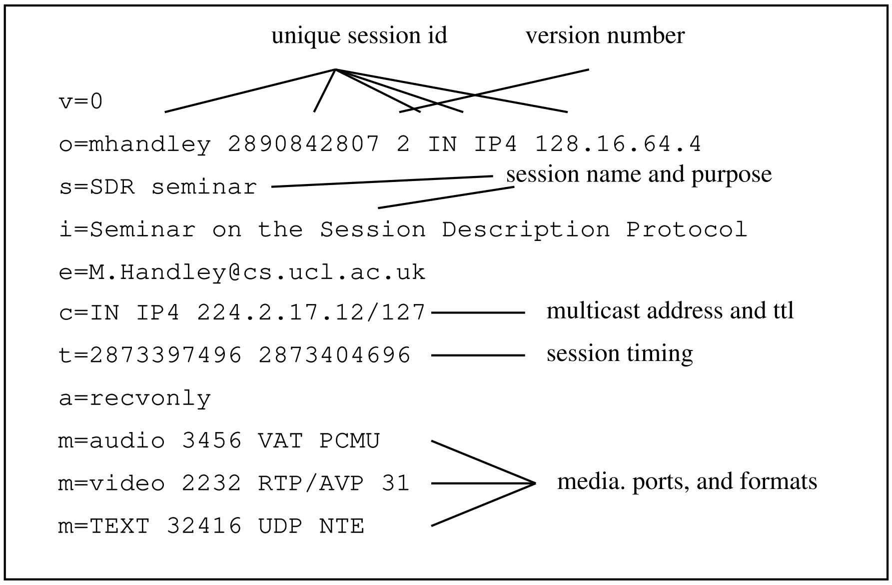


Figure 6.1: An example SDP session description

### 6.3.2 Session Announcement Protocol (SAP) - multicast distribution of session descriptions

There are three significant failure modes in the original sd session directory implementation:

- the GUI needs to be running constantly to advertise a session. If a session is not constantly advertised no new receivers can receive it and it eventually times out and is deleted at receiving sites.

- when starting sd, the delay before receiving a particular session description which is not in the disc cache results in unacceptable waits and sub-optimal multicast address allocation.

- packet loss causes unacceptable delays for new immediate session creation for some users.

In addition there is no way for an application that requires a dynamically allocated multicast address to obtain one without either using sd's GUI, or implementing IPRMA and listening for a significant length of time to wait for session descriptions to arrive.

However, despite these failure modes, sd does come close to satisfying our robustness and timeliness requirements of a session directory, and it has the additional advantage that it is simple.

### Client/Server SAP Models

To attempt to address some of the deficiencies of the sd model, we propose a client/server split with the wide area distribution model being an enhancement of simple periodic announcement, but with additional local area distribution mechanisms.

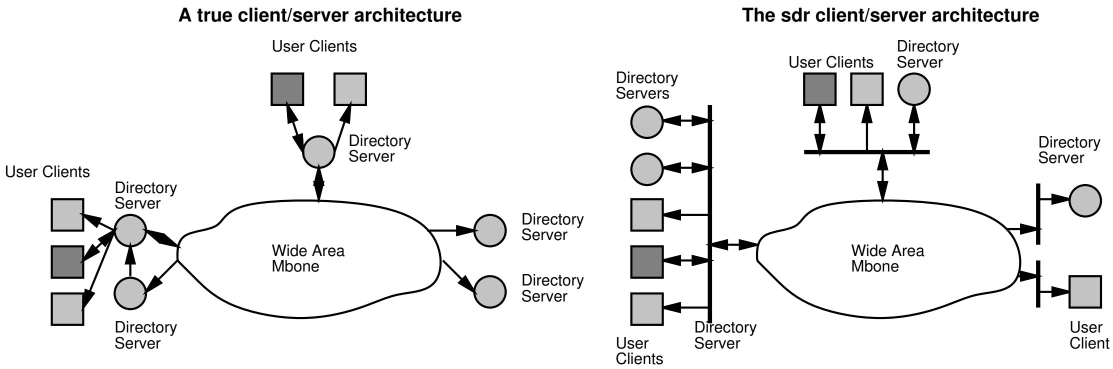


Figure 6.2: Possible SAP client/server models

At each "site" one or more Session Directory Servers (SDSs) runs continuously. The server has no user interface, but listens to session announcements from other sites and caches them locally. Where a server is in several administratively scoped multicast regions, the server must listen to one session directory multicast address per administratively scoped region.

At each site where there are multiple servers, the servers perform a designated server election to choose the server with the largest number of cache entries, or if there is a tie, the server with the lowest IP address. This server is responsible for proxy announcements when a copy of sdr is not running to announce a session itself.

We could have chosen a mechanism like that shown in 6.2a, where session directory clients only talk to session directory servers, and in the wide-area the servers talk to each other to distribute session descriptions. However, this model has little benefit over a more distributed approach, whereby the server does not have a special role except in making proxy announcements, and it mandates the setup and configuration of servers at each site.

Instead we chose a more distributed approach as illustrated in figure 6.2b. When a new instance of a session directory client is started, it may obtain the current session listing from any local session directory instance, whether it is a server or not. Thus for reception purposes the server merely is a more up-to-date cache. For transmission purposes, we could also have taken this approach and allowed any local session directory instance to be a proxy, but this makes it difficult to be sure that the deletion of a session is performed deterministically, so we allow a handshake whereby the servers takes over responsibility for the announcement of a session when a user's GUI is no longer running.

We will not discuss this local protocol further in this chapter because there is little research involved, and instead we will concentrate on wide-area session announcements and their use to perform effective multicast address allocation. We have presented these issues here because these aspects of the overall architecture solve problems that otherwise would adversely impact the overall behaviour of the address allocation schemes presented later.

#### Wide-Area Session Announcement Protocol

One problem with multicast sessions is that although multicast solves the distributed-system binding problem[1], it is still necessary to know the multicast addresses and ports a session uses in order to be able to participate in that session. There are many possible channels by which we could distribute this information, although SDP provides a standard payload format that can be used with all these different transport protocols. Which channel is most appropriate depends on the purpose of the session and its intended participants. If the intended participants are known in advance, and it is intended that a session be restricted to this list of participants, then the most appropriate channel for session advertisement is a session invitation protocol designed to seek out the location of participants and alert them to the session. Such a protocol is specified in the Session Initiation Protocol (SIP)[42].

However, because IP multicast scales well to very large groups and solves the binding problem, many sessions more closely resemble broadcast television, where the receivership is not known in advance (if at all), and what is required is a broadcast-like advertisement mechanism which will reach all of the *potential* participants. The most appropriate way to distribute advertisements for multicast sessions is by using multicast itself and sending advertisements to a well-known multicast address. So long as the multicast announcements are scoped the same as the session they describe, then recipients of multicast announcements should be able to participate in the session (pending the necessary encryption keys), and no-one should receive an announcement for a session that is out of scope, nor be able to participate in the session but not receive the announcement.

[1] binding in this context is how data consumers discover data providers

## 6.4 Multicast address allocation

Multicast addresses may be well-known addresses which are used for years, but most multicast groups are only used for a single purpose (a meeting, conference, game, etc) and then not needed again. In IPv4, there are $2^{28}$ (approximately 270 million) multicast addresses available. Over time, the total number of multicast sessions is likely to greatly exceed the address space, but at any one time, this is not likely to be a problem so long as addresses are allocated in a dynamic fashion, re-used over time, and so long as scoping is used to allow the same address to be in use simultaneously for multiple topologically-separate local sessions.

As sdr is already advertising the existence of multicast sessions and their addresses to the appropriate scope zones, one possibility is to leverage this distribution process as a part of a multicast address allocation mechanism.

An alternative approach involves the hierarchical allocation of address prefixes to Internet Service Providers (ISPs) who in turn can allocate sub-sets of their prefix to their customers. However, this approach does not achieve good address packing if it ignores scoping, or is done without regard for the actual multicast address usage patterns.

We shall examine what may be achieved by an entirely distributed multicast address allocation scheme based on the sdr session announcement architecture, and afterwards examine how this may be combined with a dynamic hierarchical scheme.

### 6.4.1 IPRMA

Van Jacobson has partially described[51] a scheme for multicast address allocation called Informed Partitioned Random Multicast-Address Allocation (IPRMA). This is intended to allow a session directory instance to generate locally a multicast address with minimal chance that this multicast address will conflict with another multicast address already in use.

Schemes like IPRMA depend on the session directory component that is performing address allocation knowing a large proportion of the addresses already in use. Information about each existing session is distributed with the same scope as the session. Session directories use an announce/listen approach to build up a complete list of these advertised sessions, and a multicast address is chosen from those not already in use. However, as different sessions have different scopes, an announcement for a local session at one site will not reach all other sites, so the address can then also be chosen for a global session at another site leading to an address clash. IPRMA attempts to avoid this by partitioning the address space based on the TTL of the session.

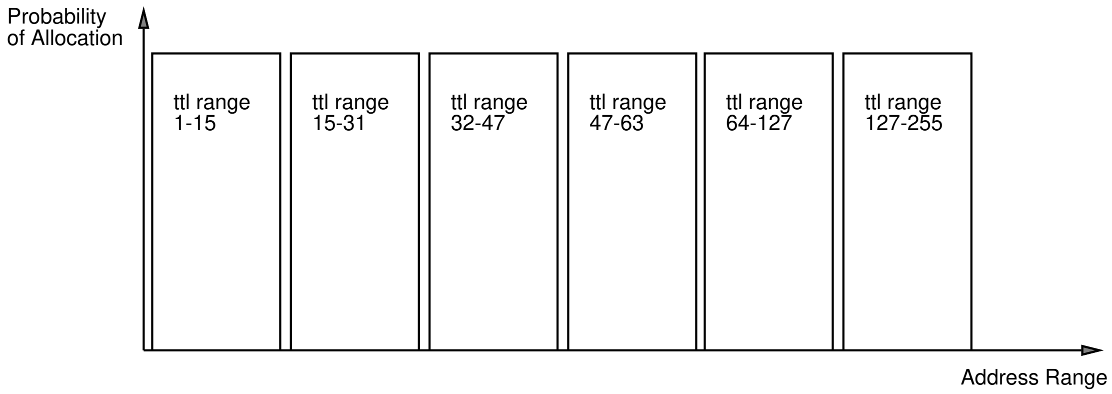


Figure 6.3: Probability Density Functions for Address Allocation for each of 6 TTL ranges

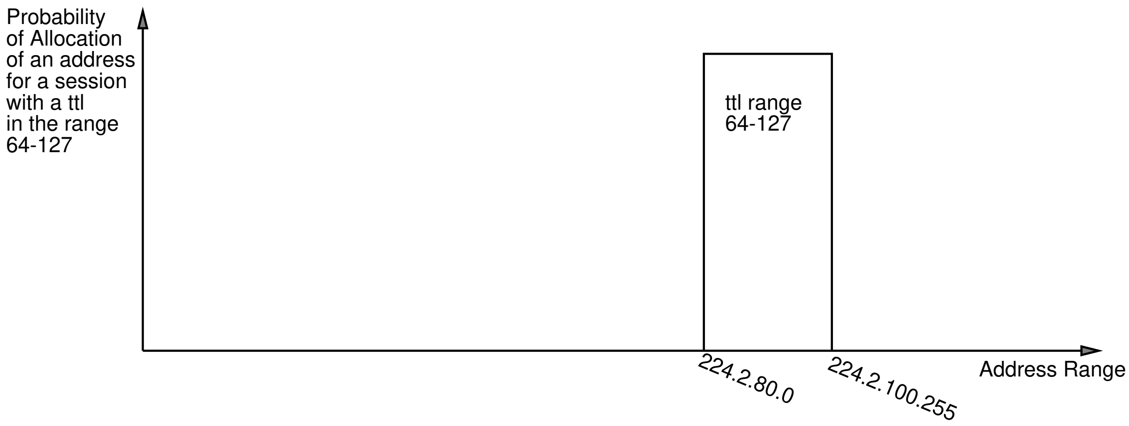


Figure 6.4: Probability Density Function for Address Allocation of sessions with TTL 64-127

The general principle of IPRMA is illustrated in figure 6.3. The figure illustrates the probability of allocating a particular multicast address. The area under each of the segments of the curve is one. For a particular TTL, only one of the segments of the curve is valid, as illustrated in figure 6.4.

Thus, although a session directory at a particular location can only see sessions advertised that will reach its location, and cannot see sessions advertised locally[2] elsewhere, the partitioning of the address space prevents a new global allocation clashing with an existing local allocation elsewhere.

The problem with partitioning the address space in this way is that some partitions may be virtually empty, and others will be densely occupied. If the session advertisement mechanism is perfect and all sites within a scope band can see all sessions advertised within that band, then we can fully populate a scope band. However this ideal is not achievable in practise for a number of reasons:

[2] locality is determined by the scope of the session, which in turn is determined by the TTL of the session

- packet loss causes delays in discovering new sessions advertised elsewhere.

- any delay in discovering new sessions elsewhere means the same address can be allocated at more than one site.

- inconsistencies between TTL zone boundaries and IPRMA partition boundaries may mean that not all sites allocating addresses within a partition can see all the other addresses in use in that partition.

The problem with inconsistencies between TTL boundaries and IPRMA partition boundaries is illustrated in figure 6.5.

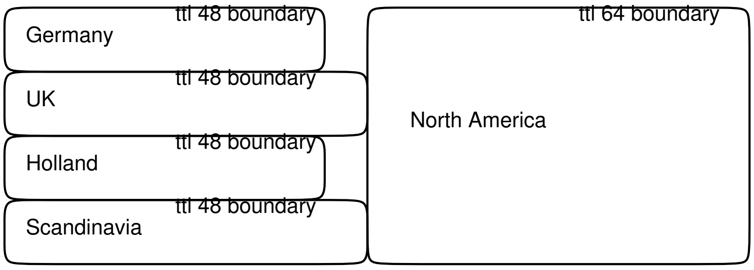


Figure 6.5: An example of inconsistent TTL boundary policies

In the current Mbone, boundaries between most countries are at TTL 64, but within Europe, the boundaries between countries are at TTL 48. The boundaries into and out of Europe are at TTL 64. This allows some groups to be kept within each country by sending at TTL 47 and some groups to be kept within Europe by sending at TTL 63. In the US, no TTL 48 boundaries exist, and so no TTL 47 sessions are used. Now if there is an IPRMA partition that covers the range 33-64 (which would be appropriate for the North America region) then both Europe wide sessions and UK only sessions fall into the same partition. However, a session directory running in Scandinavia would not see the UK TTL 47 sessions, and might cause a clash when allocating a Europe-wide TTL 63 session.

Splitting the IPRMA address space into a larger number of ranges reduces this problem, but also reduces the number of addresses available in each range. The TTL allocations are not evenly distributed throughout the possible TTL range, and in fact occur at only a few discrete values. Splitting the available address range into a set of fixed ranges means that many of those ranges are empty whilst a few are full.

To exacerbate this problem, the lifetime of software performing address allocation appears to be longer than the lifetime of TTL boundary allocation policies.

As there are delays in one site discovering that another site has announced a session, IPRMA randomly assigns addresses from the relevant partition. Using a random mechanism is necessary because we do not know which sites with which we are likely to clash and do not know how many of them there are. Using a purely random allocation mechanism within a scope band would lead to an expected address clash when approximately the square root of the number of available addresses in the scope band are allocated. Figure 6.6 illustrates the probability of a clash for allocations from an address space of 10,000. This is the well known "birthday problem", so named because the probability of their being two children in the same class with the same birthday is high for typical class sizes.

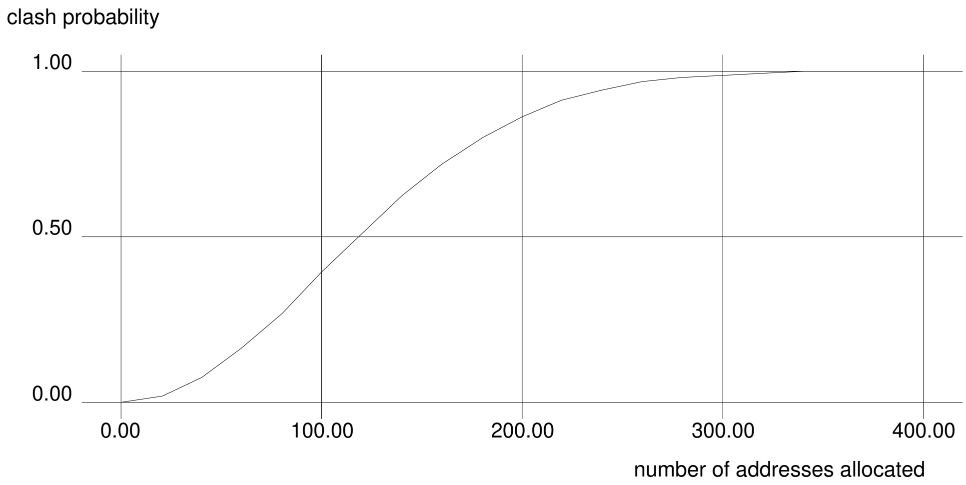


Figure 6.6: Probability of an address clash when allocating randomly from a space of 10,000

IPRMA's mechanism is not purely random. Addresses that an address allocator knows are in use are not chosen, so the random allocation is only from those addresses which are unallocated and from those which the session directory has failed to inform the address allocator are in use. Thus the probability of an address clash is dependent on how well IPRMA's partitions match the TTL boundaries in use and on how good a view the address allocator has of the sessions already allocated. If the address allocator is using SAP for receiving session descriptions and the session directory server has been running continuously, then the accuracy of the address allocator's model is dependent on the mean propagation delay (taking into account packet loss) and the rate of creation of session advertisements.

### 6.4.2 Examination of the Algorithms

To examine these algorithms, both simulation and mathematical approaches are possible. The simpler algorithms such as random and partitioned random are clearly mathematically tractable, but the more complex informed algorithms and especially the adaptive informed algorithms presented later proved too difficult to analyze, and so were forced to resort to simulation. Simulation also has the additional desirable property that we can use real-world measured data such as network maps which introduce irregularities that would be lost in a pure mathematical approach, and which might affect the performance of some of the algorithms. Should network scoping mechanisms or policies change in the future, these same simulations can be repeated to examine precisely the effects of the change on the address allocation process. To ensure that any comparison between the schemes is comparing like with like, we have simulated all algorithms where we wish to compare performance.

### 6.4.3 Simulation-based Comparison of Algorithms

To illustrate the different algorithms in a more realistic setting, we took a map of the real Mbone as gathered from the *mcollect*[31][32] network monitor, and built a simulation model of the Mbone topology including all the TTL thresholds and DVMRP routing metrics in use. The mcollect data is not a complete mapping of all of the Mbone because some mrouters do not have unicast routes to the mwatch daemon, but it represents a large proportion of mrouters in use. Any disconnected subtrees of the network were removed, and the resulting connected graph includes 1864 distinct nodes.

Nodes in this graph were chosen at random as the originator of a session, and the TTL for the session was chosen randomly from the following distributions:

**ds1** {1,15,31,47,63,127,191}

**ds2** {1,1,15,15,31,47,63,127,191}

**ds3** {1,1,1,1,15,15,15,15,31,47,63,127,191}

**ds4** {1,1,1,1,1,1,1,1,15,15,15,15,15,15,31,31,47,47,63,63,127,191}

Although these TTL distributions are not based on realistic data, they help illustrate the way that local scoping of sessions helps scaling, even where it defeats the informed allocation mechanisms.

Four algorithms were tested:

**R** - pure *random allocation*

**IR** - *informed random allocation*. An address is not allocated if it is seen in another session announcement

**IPR 3-band** - *informed partitioned random allocation* with 3 allocation bands separated at TTLs 15 and 64

**IPR 7-band** - *informed partitioned random allocation* with 7 allocation bands separated at TTLs 2, 16, 32, 48, 64 and 128

IPR 3-band illustrates the effect of imperfect partitioning as discussed in reference to figure 6.5. IPR 7-band is basically perfect partitioning in this case, as no two different TTL values from the distribution fall into the same band.

In this simulation we assume no packet loss, and this gives unrealistically good results for the informed schemes. This is not a realistic situation in the real world, and we will look at the effects of loss later. Routing is performed using the DVMRP routing metrics, and scoping achieved using the TTL thresholds configured in the Mbone as reported by mcollect. The simulations were performed in parallel on 6 Sun SparcStations and 3 DEC Alpha workstations, and represent approximately 700 hours of processing time.

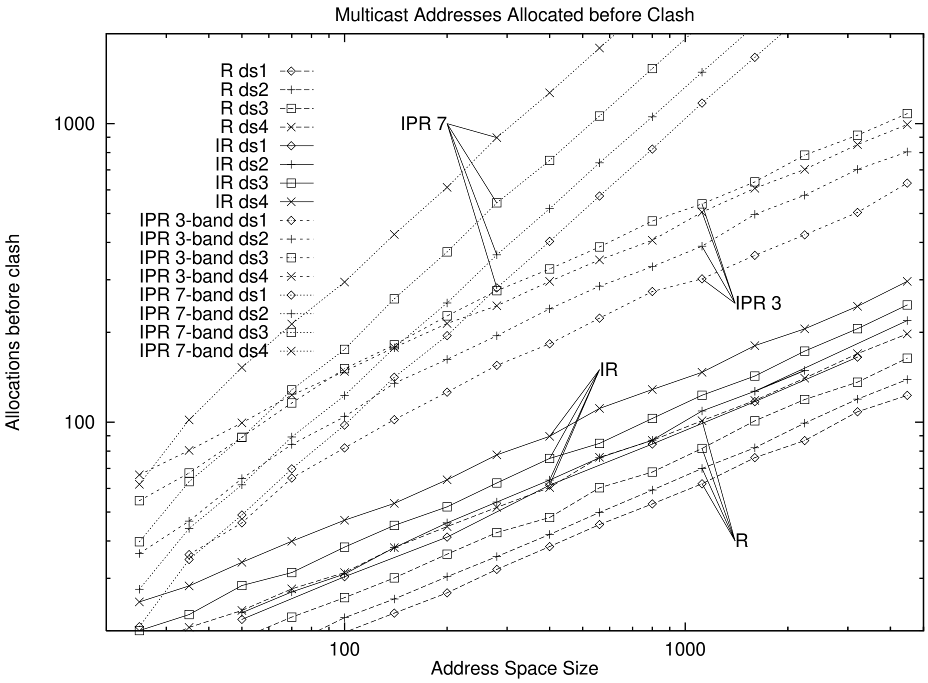


Figure 6.7: Simulations of address allocation algorithm performance

The results of this simulation are shown in figure 6.7 on a log/log graph. As can be seen, random (R) and informed random (IR) achieve a mean allocation of $O(\sqrt{n})$ before an address clash occurs, where $n$ is the number of addresses available. Also interesting is that informed-random is not a great improvement on random allocation.

Informed Partitioned Random with 3 bands does significantly better than Informed Random, but still only achieves a mean allocation of approximately $O(\sqrt{n})$ before a clash occurs for larger values of $n$.

Informed Partitioned Random with 7 bands (perfect partition placement) achieves an optimal mean allocation of $O(n)$, and with the TTL distributions used, is limited by higher scope bands filling completely.

### Partition sizing

As the above simulation illustrated, IPRMA only scales well if partitions are placed exactly in alignment with TTL boundaries in use and TTL allocations. To avoid clashes due to mismatches between the IPRMA TTL partitions and the TTL boundaries in use, TTL partition sizes should be as small as possible. To minimise the proportion of the address space unused due to only a few discrete TTL values actually being used, the TTL partition sizes used by IPRMA should be large. If we could be sure which TTL values should be in use, a pre-allocation of address space to TTL ranges would be possible such that mismatches do not occur.

To illustrate the issue, let us first consider *un-informed* partitioned random allocation. We will assume an address space of 65536 addresses (this is the size of the address range currently allocated by IANA for dynamic multicast address allocation). Currently TTL boundary values of 1, 16, 32, 48, 64, 128 and 192 are in use in significant numbers on the Mbone, requiring at least 8 TTL ranges if IPRMA is to be expected to perform reasonably. If we divide the address range into 8 equal ranges, use random allocation within each partition, and allocate session TTLs such that each site sees an equal number of sessions in each of these ranges, then the following gives the expected number of sessions that can be allocated:

For each partition,
let $n$ be the number of potential addresses in the partition.
let $m$ be the number of the addresses previously allocated in that partition.
The probability, $c$, of any single address allocation not clashing with an existing address is given by:

$$c_m = \frac{n-m}{n}$$

Thus the probability of no clash, $p$, when allocating $m$ addresses in a partition of size n is determined by:

$$p_m = \prod_{i=0}^{m-1} c_i$$

$$= \frac{n(n-1)(n-2)(n-3)...(n-m-1)}{n^m}$$

$$= \frac{n!}{(n-m)!n^m}$$

With 8 partitions, the probability of no clash in *any* partition is given by $(p_m)^8$.

Thus we expect approximately 39 sessions to be allocated in each range before the probability of a clash in at least one of the ranges exceeds 0.5, and so this scheme breaks down with approximately 312 sessions visible from each site. If we increase the number of TTL ranges to 16, this scheme breaks down at approximately 224 sessions visible from each site, but we have the advantage of being much more flexible to changes in TTL boundary allocation policy. Note that these are best figures for partitioned random allocation, in that sessions are evenly distributed across our address partitions.

### 6.4.4 Effects of Announcement Delay and Loss on Perfectly Informed Partitioned Random Allocation

The illustration above does not represent how Informed Partitioned Random Allocation performs, as it ignores the information about existing sessions that SAP provides us. To take into account the benefits of IPRMA over partitioned random allocation, we need to have more information about the length of sessions and of the effectiveness of SAP.

Let us assume that the mean length of a session is 2 hours, that the mean advance announcement time is 2 hours, that the mean end-to-end delay across the Mbone is 200ms, that the mean packet loss is 2% and that SAP retransmits an announcement every 10 minutes. Allowing for packet loss, these figures give a mean end-to-end delay for SAP approximated by (0.98*0.2)+(0.02*600)= 12 seconds. Given that a session is advertised for a mean of 4 hours, approximately 0.1% of sessions currently advertised are not visible at any time.

Thus given the same perfect situation for IPRMA given in the example above, the probability of a clash in any given partition is determined as follows:
let $n$ be the number of addresses potentially available in the partition.

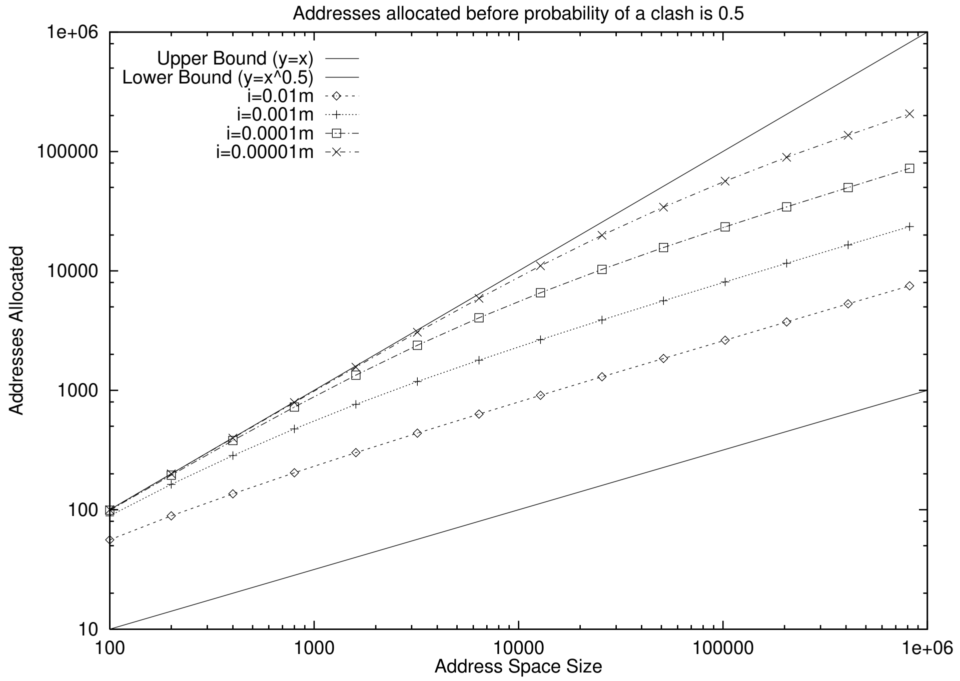


Figure 6.8: Addresses allocated in one IPRMA partition such that the probability of a clash is 0.5

let $m$ be the number of addresses currently allocated

let $i$ be the number of addresses invisibly allocated

$$i = 0.001m$$

the probability, c, of any single new address allocation not clashing with an existing address is thus given by:

$$c_m = \frac{n-m}{n+i-m}$$

If we assume the total number of sessions allocated, $m$, is a constant, and that no session is advertised for less than 10 minutes, then the probability, $p_m$, of no clashes occurring within the mean lifetime of a session is given by:

$$p_m = \left(\frac{n-m}{n+i-m}\right)^m \tag{6.1}$$

Thus with the same address space of 65536 addresses partitioned into 8 equal regions, and even distribution of sessions (as seen from each site) across the TTL regions, IPRMA gives us a total of approximately 16496 concurrent sessions as seen from each site before the probability of a clash exceeds 0.5. Figure 6.8 shows a graph (computed from equation 6.1) of the address space size within a partition against the number of addresses allocated in that partition before the probability of a clash within any four hour period exceeds 0.5. Results are given for several different values of $i$. As can be seen, the address space packing is good for small partitions, but gets less good as the size of the partition increases. Clearly the efficacy of the announcement protocol is of paramount importance, as shown by the significantly better results with smaller values of $i$.

These numbers serve to illustrate the performance of IPRMA under near perfect conditions. As figure 6.8 shows, even IPRMA only manages to allocate $O(\sqrt{n})$ before the probability of a clash becomes significant when loss rates are higher because its limiting factor is the random element introduced to cope with failures of the announcement mechanism. The curve given by $i = 0.00001m$ is probably an upper bound on the performance of IPRMA as this is approximately the value given with zero packet loss and a 200ms end-to-end delay. However, we can come close to this curve by not announcing sessions at a constant interval, but starting from a high announcement rate (say a 5 second interval) and exponentially backing off the rate until a low background rate is reached. Combined with local caching servers so that new session directory instances get a complete current picture, and assuming a mean loss rate of $2\%$, repeating the announcement 5 seconds after it is first made gives a mean delay of about 0.3 seconds, and hence $i = 0.00005m$. Note also that many sessions are announced for much longer than the four hours assumed here.

It is important to note how the address allocation mechanism and the address announcement mechanism need to be closely coupled in this architecture. Small changes to the announcement model can greatly affect the scalability of the address allocation model.

In this analysis, it has been assumed that sessions are evenly allocated across the TTL regions and the IPRMA regions are pre-allocated and coincide precisely with the TTL boundaries in use on the Mbone. However, making these assumptions in an address allocation tool would be a dangerous path to take, as changes in TTL boundary policy could make the pre-allocation of address space highly sub-optimal.

An alternative approach is to try and make the partitions initially small and numerous, but to make their size adapt depending on the sessions already allocated.

### 6.4.5 Adaptive Address Space Partitioning

Until this point, all the calculations have made the assumption that IPRMA partition boundaries are fixed. If we do not know in advance which TTL values will be used by sessions and we do not know the distribution of sessions between those partitions, then it makes sense to make the partitions themselves adaptive. There are a number of options for how this adaptation can be performed, and these are illustrated in figure 6.9.

Initially the address range is divided into even sized partitions, as shown in figure 6.9a. As some of the partitions start to become densely occupied whilst others are sparsely occupied, it is necessary to adapt the size of the partitions. As the size of some partitions increases, it is necessary to correspondingly reduce the size of other partitions to make space.

This can be done in one of several ways:

- Changing just the width of the partitions (figure 6.9b). The partitions become asymmetric, but remain centered on their original location.
- Changing both the width and location of the partitions (figure 6.9c). The partitions remain symmetric, but change location as some address bands start to become densely populated.

In figures 6.9b and 6.9c, many sessions which fall into TTL range C are allocated, and C needs to expand. In both examples, C eventually starts to allocate addresses that were originally allocated from range B. At this point, unless all sites have similar information about which addresses are in use, there is a possibility of clashes occurring between new sessions in the more widely scoped range and existing sessions in the less widely scoped range that are not visible at the sites allocating the new session.

#### Deterministic Adaptive Address Space Partitioning

The major failing of adaptive IPRMA, as described above and in [51], emerges from one of two circumstances:

- two or more TTLs of sessions fall into the same IPRMA address partition. This results in some lower TTL sessions in that partition not being visible at sites wishing to advertise a higher TTL session falling into the same partition.
- a densely packed partition expands at one site to overlap a lower TTL partition at another site. The higher TTL partition is constrained in its growth at one site by a large number of allocations in a lower TTL partition. At another site, these lower TTL allocations are not visible, so the higher TTL partition sees different constraints to its growth, resulting in an overlap with the densely packed partition at the first site.

Both of these situations result in a failure of the "informed" mechanism in IPRMA. The first situation can be prevented by increasing the number of partitions, but at the expense of exacerbating

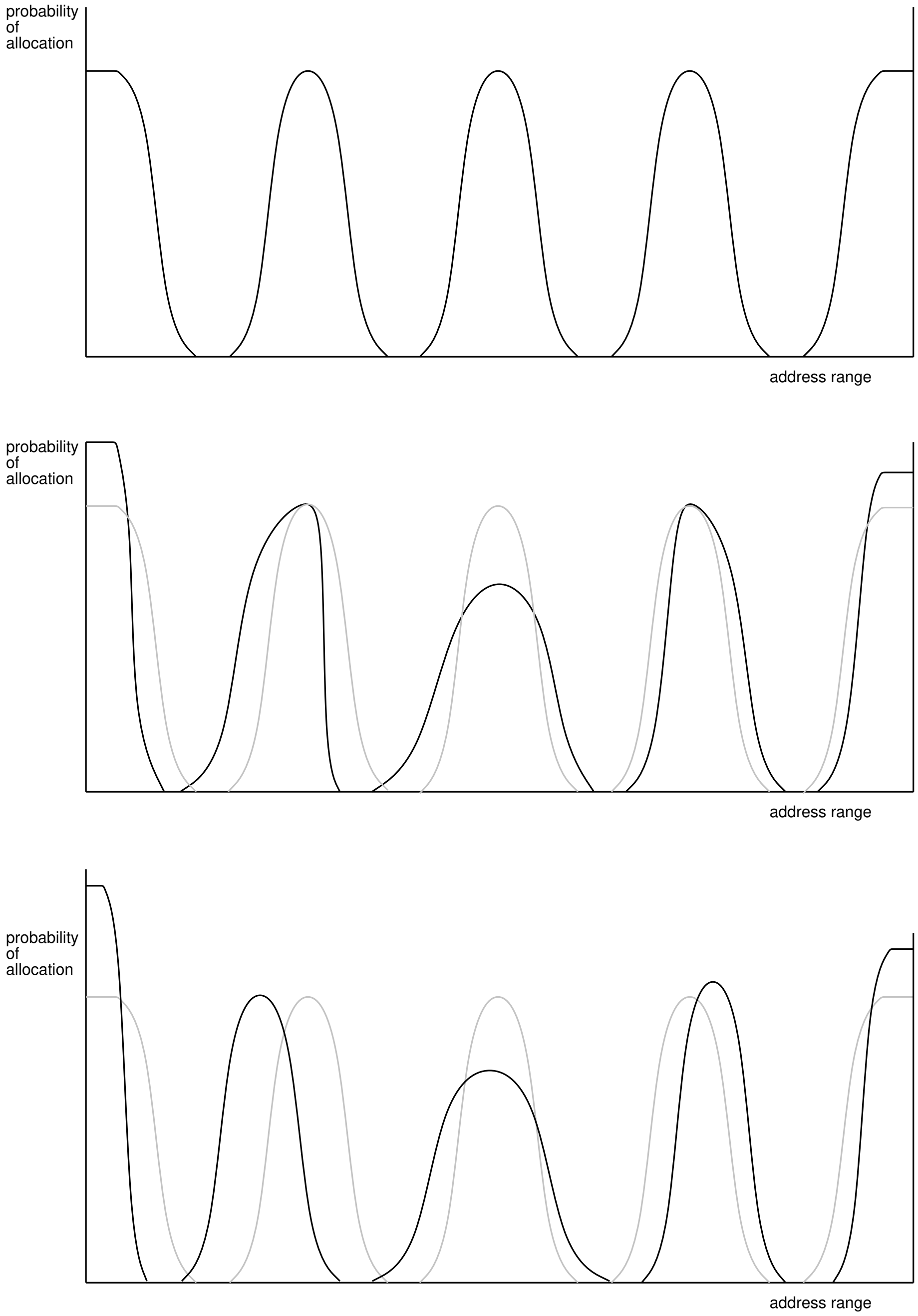


Figure 6.9: Illustration of two options for adaptive address space partitioning

the second situation as partitions start smaller and therefore need to grow more to avoid address clashes caused by delays in announcement propagation. As figure 6.8 showed, even small failures in the informed mechanism have a significant effect on the number of addresses that can be allocated without a clash occurring.

If we assume that SAP communicates all sessions announced at a particular TTL to all the sites reachable at that TTL with no delay, then *there is a deterministic solution to the above problem*.

To achieve this, we partition the address space into sufficient partitions that only one frequently used TTL value falls into each address range. Under these circumstances, the original adaptive IPRMA can fail because the size and/or position of an IPRMA partition is affected by both higher and lower TTL partitions, and space is wasted by empty partitions. However, if we assume a perfectly reliable SAP[3], then any site wishing to allocate a session $S$ with a TTL of $x$ can see the session announcements for all sessions that $S$ can clash with that have TTLs greater than or equal to $x$. Thus, in our variant of IPRMA, every site bases the position and size of the partition corresponding to TTL $x$ only on session announcements for sessions with a TTL greater than or equal to $x$. This ensures that no clash can occur due to the failings above. It requires that the initial partition sizes are very small, and that partitions are initially clustered at the end of the space corresponding to maximum TTL. Figure 6.10a illustrates an initial starting partitioning before any addresses have been allocated. Figures 6.10b and c illustrate the IPRMA partitioning at two sites after a significant number of addresses have been allocated. These two sites can communicate with a TTL of $t$ or greater.

### Determining Adaptive IPRMA Partitioning

To implement Adaptive IPRMA, a set of parameters need to be defined:

- The number of initial partitions and the TTL ranges covered by them.
- The shape of the probability density function determining a partition.
- The desired occupancy of a partition.

### Initial Partitioning

In deciding the initial partitioning, we have two choices:

[3]SAP is of course not perfectly reliable, but packet loss affects the categories much less than individual address allocations, so the simplification is not unreasonable in this case

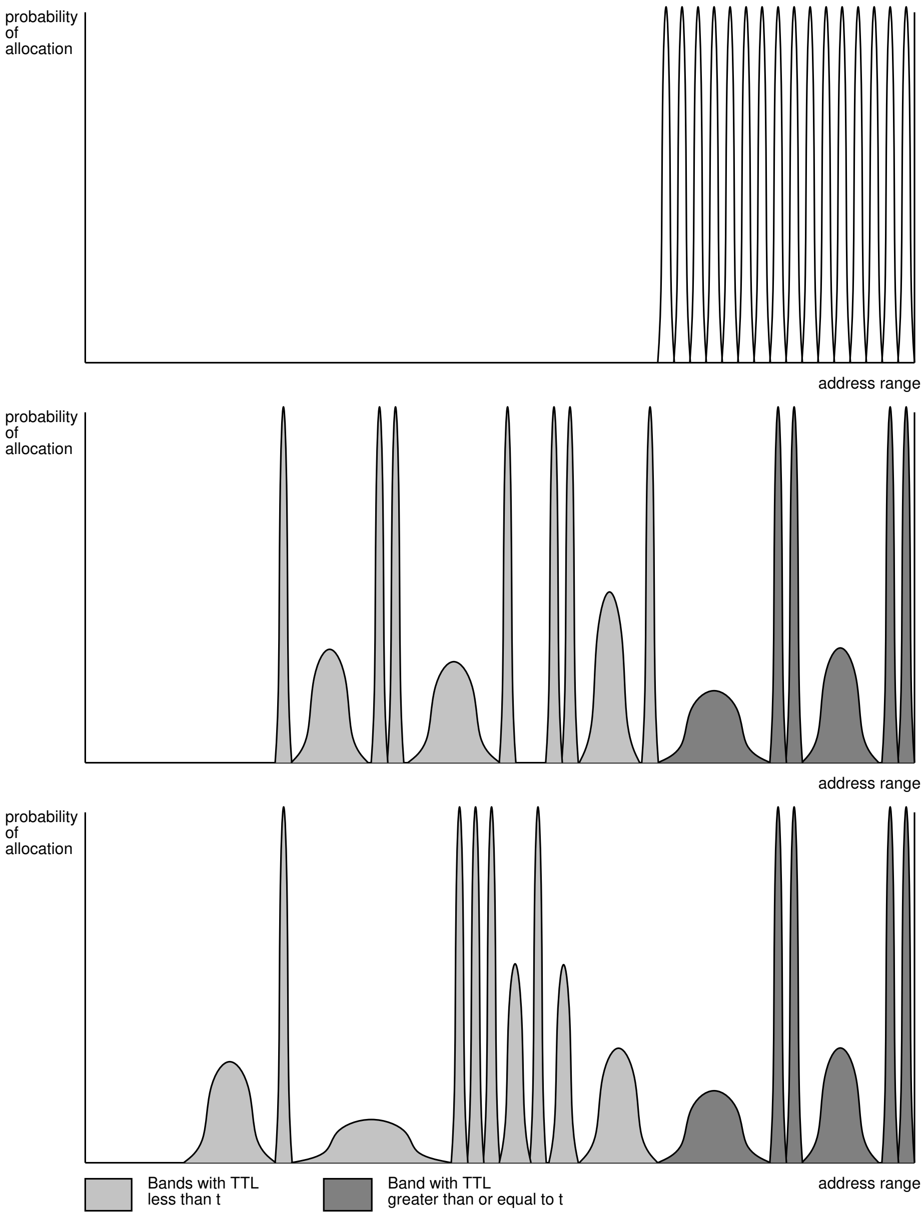


Figure 6.10: Illustration of Deterministic Adaptive IPRMA

- Choose a partitioning based on values currently in use in the Mbone.

- Choose a partitioning that will work for any TTL boundary allocation policy.

Obviously the latter is more desirable if the overheads of choosing such a partitioning are sufficiently negligible.

The only initial partitioning that will work for all possible boundary allocation policies is to partition the address range into one partition for every TTL value.

If each initial partition occupies $n$ addresses then in the worst case we have $255n$ addresses we cannot allocate unless we reduce unused partition sizes at some later time when a significant number of addresses have been allocated. However, reducing unused partitions can present its own problems.

We would like to be able to treat consecutive empty partitions as if they were a single partition. However, no site can be sure that one of the consecutive empty partitions is not in use elsewhere using a TTL value just below the TTL threshold value necessary to reach them. In addition, any scheme which compresses several empty partitions into a minimal set of partitions suffers from a bifurcation of the empty partition range if an address is ever allocated from one of the previously empty ranges. This bifurcation results in a sudden change of address range position for all the lower TTL ranges, and we wish to avoid such sudden changes as they can cause additional problems.

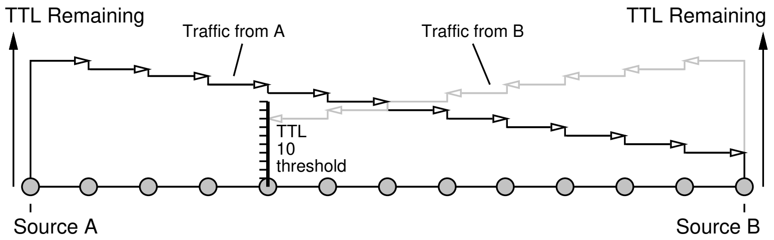


Figure 6.11: Potential Asymmetry due to TTL threshold

Fortunately, there is an additional effect that comes into play at higher TTL values which makes one band per TTL value unnecessary: the TTL on data packets is decremented at each multicast router traversed (see figure 6.11). This means that our assumption that “if site A can see site B's TTL $x$ traffic, site B will be able to see site A's TTL $x$ traffic” may not hold if a high threshold boundary separating A and B is not equidistant from them.

At low TTL values, this effect is small as hop-counts and therefore potential asymmetries are small. For large scopes with higher TTLs, the effect becomes more pronounced. In the real MBone, this problem is explicitly avoided by sending traffic with TTL $y-1$ when it is intended to stay within a zone with boundary threshold $y$. This allows sufficient leeway for the decrementation of TTL to ensure the traffic still passes any thresholds internal to the TTL $y$ bordered region. If the TTLs chosen for data traffic do not suffer from this problem, then neither will the session announcements describing this traffic. Thus at high TTL values, many closely spaced TTL ranges which are also close to TTL threshold border values (and therefore need to be in separate partitions) will not occur.

Allocating one partition per TTL value is necessary at very low TTLs, but because of this property, it is unnecessary at high TTLs. The general guideline here is that the TTL range allocated to an address space partition must not be significantly greater than the typical hop count of sessions advertised at that TTL, but if boundary values are consistently chosen worldwide, then the TTL range need not be significantly smaller that the maximum multicast hop count.

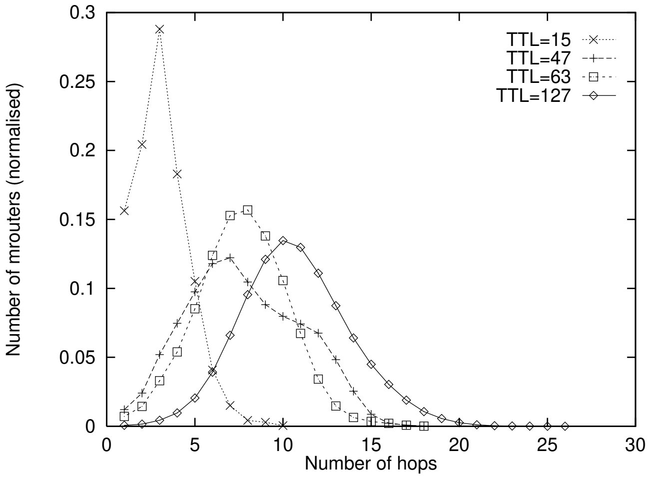


Figure 6.12: MBone hop count distribution for several TTL scopes

Figure 6.12 shows the distribution of hop-counts available in the real Mbone. The graph is created from the mcollect network map by taking each mrouter and calculating a histogram of number of mrouters against distance from that mrouter for each of four commonly used TTLs. The graph shows the combined histogram for all potential sources. TTL 47 is unusual because it is used to separate countries in Europe, but is not usually used as a boundary elsewhere in the world, where TTL 47 traffic will behave just like TTL 63 traffic.

For example, from UCL, some fairly typical cross-region hop counts are:

| TTL | Route | hop count |
|---|---|---|
| 255 | DVMRP routing metric infinity | 32 |
| 127 | UCL (London) to LBL (California) | 15 |
| 63 | UCL (London) to INRIA (France) | 10 |
| 47 | UCL (London) to Edinburgh | 6 |
| 16 | within UCL | 3 |

The hop count curves in figure 6.12 give more typical figures for the whole world:

| TTL | Most frequent hop count | Maximum hop count |
|---|---|---|
| 127 | 10.6 | 26 |
| 63 | 7.7 | 18 |
| 47 | 7.0 | 18 |
| 16 | 3.1 | 10 |

In general, the expected hop counts are approximately proportional to the TTL as this is of primary importance to network managers when setting up a TTL boundary. If our scheme is to cope with any *likely* boundaries, the size of the highest TTL band (up to TTL 255) should be less than the DVMRP infinite routing metric of 32. Boundary values are not always chosen consistently, so we also wish to build in a margin of safety to the partition sizes.

Given this, the number of TTL values, $n$, allocated to a partition with lowest TTL $t$, with a margin of safety $m$, is given by the following, with n rounded up to the nearest integer:

$$n = \frac{32t}{255m}$$

Choosing a margin of safety of 2 gives 55 partitions, as shown in figure 6.13. This will work well for existing partitioning and for any likely future partitioning.

### 6.4.6 Sizing partitions

Given that the above partitioning can be regarded as “sufficient” in the sense that it will result in an IPRMA allocation scheme having information about all the addresses in use in each partition, there are several issues that can be used to determine the size of a partition:

1. Packet loss rates

2. Session duration statistics

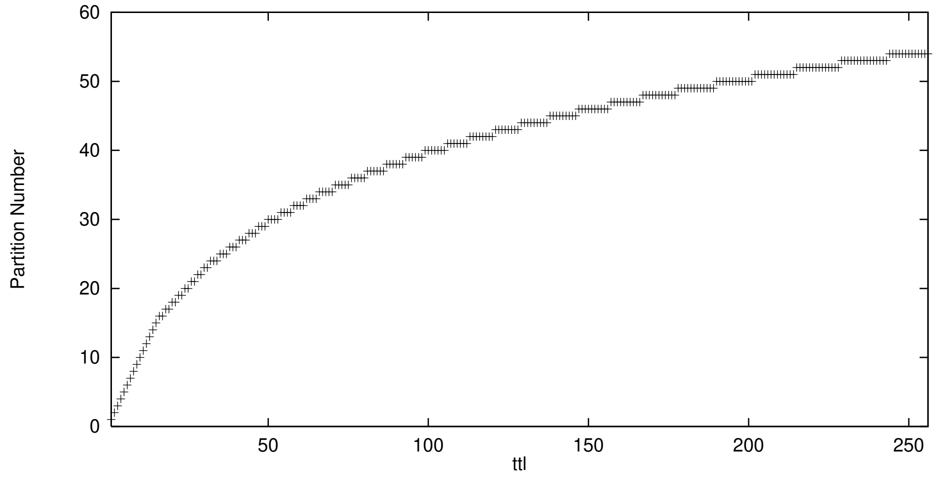


Figure 6.13: Mapping of TTL values to IPRMA partitions

3. The session announcement mechanism

4. Statistics about the change rates of partitions themselves

If we have information about the first three of these, then we can use figure 6.8 or a derivative of it to determine a safe minimum size for the address space in a partition given the number of sessions currently allocated in that partition. However, measuring packet loss rates and using them to derive such a minimum size adaptively is likely to cause problems because we need every site to come up with the same answer to such a calculation. Session duration statistics might be a different matter, as all sites in a partition should be seeing the same sessions, but requiring a session directory to continually keep such statistics is probably more work than is necessary because it will be masked by uncertainties about loss rate and about statistics about the change rates of partitions.

For our variant of IPRMA to work well, it must be able to cope with any “flash crowds” it is likely to see. This requires an additional margin to be able to cope, and a small gap between partitions with sessions in them so that partitions can move in response to such allocation bursts without “colliding” with neighbouring partitions.

### 6.4.7 Simulations of Adaptive IPRMA

With static IPRMA (as simulated in figure 6.7), a reasonable method to examine the scalability of the scheme is to simulate filling up the address space until a clash occurs. With adaptive variants of IPRMA, such an approach is not useful as these schemes are designed to provide good address space utilisation in an environment where the number of sessions and their distribution is to some extent likely to be stable.

To simulate the effect of steady state behaviour is more difficult, as we need some definition of the steady state, and also some criteria for deciding whether the address allocation scheme is performing acceptably.

As a criterion for acceptable performance we chose the following:

> *An address allocation scheme is acceptable if during the mean lifetime of a session the probability of an address clash anywhere in the world is less that 50%.*

This criterion is convenient, but somewhat arbitrary. The choice of a 50% probability of a clash is not important except that we have to choose some threshold. The choice of a mean session lifetime was chosen because this means we do not need to consider session lifetime in our criteria - we need only de-allocate and re-allocate as many sessions as we started with.

Thus to simulate performance of these algorithms, we use the following method and the same real-mbone topology as before:

1. Allocate $n$ sessions with TTLs chosen from the appropriate distribution and sources chosen at random without regard for address clashes.
2. Re-allocate the addresses using the algorithm being tested so that no clashes exist.
3. Remove one existing session chosen at random.
4. Allocate a new session.
5. Repeat from 3 until $n$ sessions have been replaced keeping score of the number of address clashes.

This process is repeated 100 times to obtain a mean value for each choice of the address space size and each choice of $n$ to produce a table of clash probabilities. The precise value of $n$ for each address space size where the probability of a clash exceeds 0.5 is discovered by using a median filter to remove remaining noise.

Although a search algorithm can be used to locate the approximate range on $n$ for each address space size, this is still an expensive simulation to run to any degree of accuracy or scale.

Figure 6.14 shows the results of running this simulation with various algorithms. The TTL distribution is DS4 as in figure 6.7. The algorithms shown are:

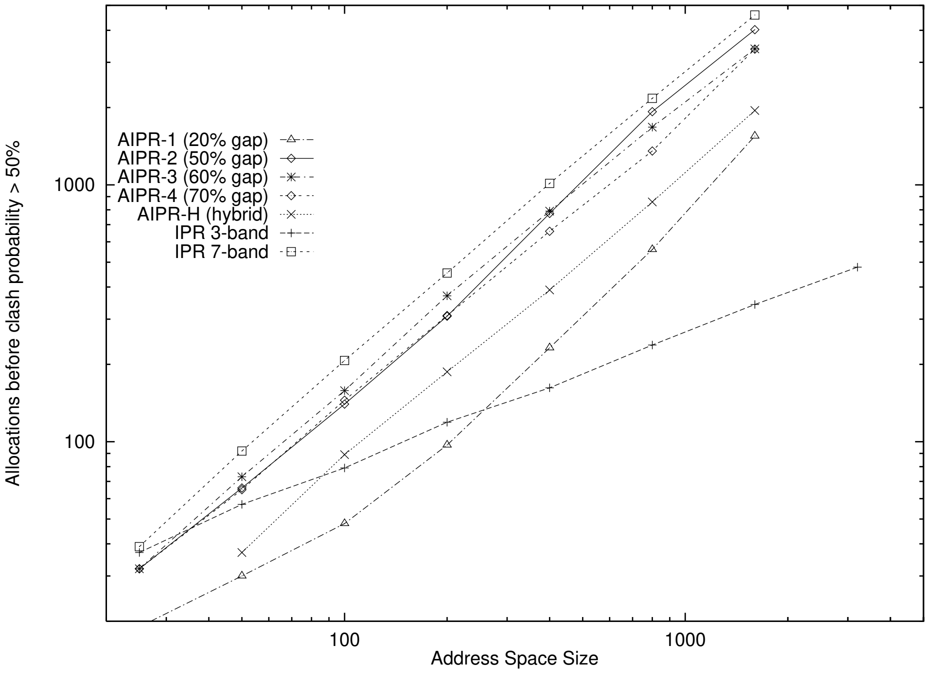


Figure 6.14: Steady state behaviour of Adaptive Informed Partitioned Algorithms

**AIPR-1** - an adaptive informed partitioned random algorithm as illustrated in figure 6.10. In this case, the bands are rectangular, 20% of the address space is evenly allocated to inter-band spacing, and the target band occupancy is 67%. The initial band allocation allocates only a single address to each band.

**AIPR-2** - as AIPR except 50% allocated to inter-band spacing.

**AIPR-3** - as AIPR except 60% allocated to inter-band spacing.

**AIPR-4** - as AIPR except 70% allocated to inter-band spacing.

**AIPR-H** - an adaptive informed partitioned random algorithm forming a hybrid between IPRMA-7 and AIPR-1. 20% of the address space is evenly allocated to inter-band spacing, and the target band occupancy is 67%. The initial band allocation occupies the upper 50% of the address space.

**IPR 3-band** - the static 3-band informed partitioned random algorithm as used in figure 6.7. We use it here as a control experiment to compare static and dynamic partitioning schemes.

**IPR 7-band** - the static 7-band informed partitioned random algorithm as used in figure 6.7.

As before, IPR 3-band and IPR 7-band use static allocation bands based on TTL. No gap between the bands is required.

AIPR-1, AIPR-2, AIPR-3 and AIPR-4 allocate bands depending on the number of sessions in the band. Each band begins with only a single address and expands with the goal of 67% occupancy[4]. Bands are initially positioned at the top of the address space, and higher TTL bands which expand "push" lower TTL bands down the address space. AIPR-1 allocates 20% of the available address space to inter-band gaps. AIPR-2 allocates 50% of the space to inter-band gaps, AIPR-3 60% and AIPR-4 70%. These gaps are needed to absorb natural variations in band occupancy, where a higher TTL band can expand and move down the address space potentially causing clashes with old addresses in lower TTL bands.

AIPR-H is a hybrid between IPR-7 and AIPR-1. It has 7 bands as in IPR-7. These bands are initially positioned so that they occupy the top 50% of the address space with 20% of the space being used for inter-band gaps. When a high TTL band expands, it pushes downwards, but the band below it does not move downwards unless the occupancy is greater than 67%. If the occupancy is less than 67% the band is reduced in width.

Of the adaptive schemes, AIPR-3 performs best in this simulation. This result was somewhat surprising as so much of the address space is reserved for gaps between the allocation bands. However, the explanation for this is that this behaviour is due to the nature of the simulation. As session originators are chosen at random and TTLs are chosen randomly from the DS4 distribution, the number of highest TTL sessions does not fluctuate beyond what would be expected from the variation in number of high TTL session. However the lower TTL sessions not only change globally in number, but also change in their location, and this leads to large variations in number of low TTL sessions *visible from a particular site*. We postulate that this behaviour does not accurately represent what happens in the real world, where a particular community chooses a TTL for their sessions and the number of sessions that community creates varies within more restricted bounds than would be the case in our simulator. Thus the adaptive schemes which assume some degree of stability in the number of *visible* sessions in any particular band are actually performing surprisingly well given the nature of the simulation, but the rapid variations in visible sessions are reflected in the need for excessively large inter-band gaps. This is also reflected in the relative improvement of AIPR-1 and AIPR-2 as the address space increases. With a larger number of addresses allocated, the relative size of the fluctuations in lower TTL bands is reduced, and the algorithms start to perform

[4] 67% was chosen from figure 6.8 as approximately the proportion of the address space that can be allocated for a band of 10000 addresses before propagation delay and loss alone increase the clash probability to 0.5

somewhat better. It would be interesting to simulate larger address spaces than the 1600 addresses that are simulated here, but to test $n$ addresses allocated, the simulation requires $O(n^3)$ time and $O(n^2)$ space (or $O(n^4)$ time and $O(n)$ space), and this makes simulating larger spaces infeasible with the resources available. [5].

The nature of clustering of session allocations in the real world is not well understood, and so devising an appropriate simulation of this is somewhat difficult. Producing an upper bound is somewhat simpler, and can be done by replacing a session advertised from a site with a particular TTL with a session advertised from the same site with the same TTL. This is not a particularly interesting simulation as it doesn't test the adaption mechanism itself, but merely the limits to how far the mechanism can adapt. In the case of the static schemes, this tests the point at which one band in the scheme typically becomes full. For the adaptive schemes, this typically tests the point at which the TTL=1 band runs out of address space at some point in the topology.

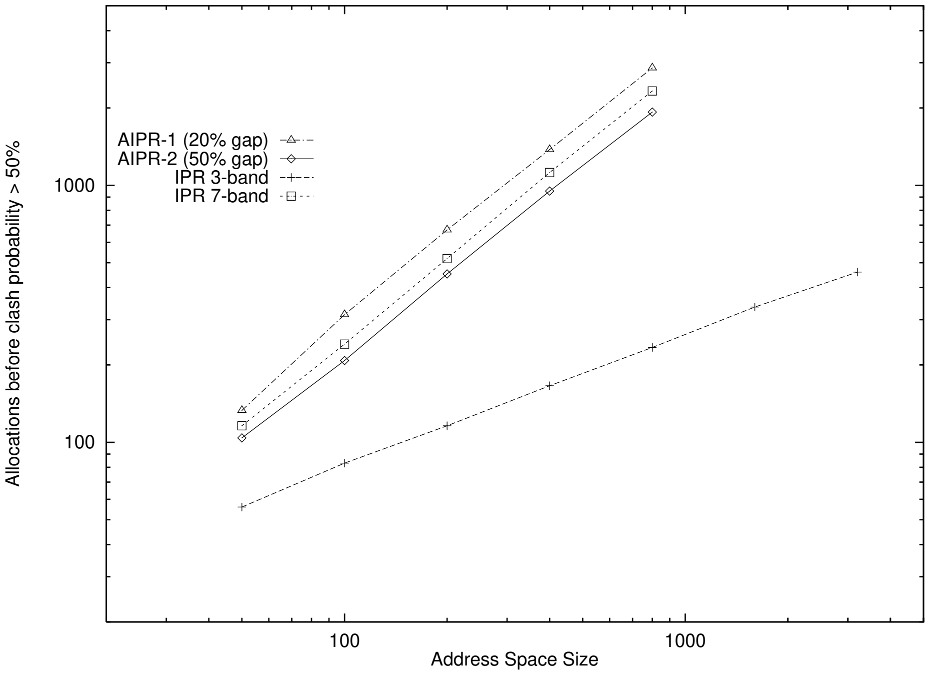


Figure 6.15: Upper-bound on steady state behaviour of Adaptive Informed Partitioned Algorithms

Simulations of this upper bound are shown in figure 6.15. As would be expected, AIPR-1 with 20% of the space allocated to bandgaps does the best of the schemes previously simulated, and considerably better that AIPR-2 (50% allocated to bandgaps). The static scheme IPR-7 still performs well, but as previously noted, this scheme is not practicable unless the precise TTL values

[5] To simulate a single data point for AIPR-3 for 1600 addresses ( 3000 allocations) takes approximately 24 hours and 36 MBytes of memory on a 200MHz DEC alpha using the $O(n^3)$ time algorithm

to be used are known in advance.

Without more knowledge about the nature of session clustering as multicast sessions become more common, it is difficult to go beyond the bounds we have explored here for adaptive address allocation schemes. These simulations show that with Deterministic Adaptive IPRMA the number of allocations scales linearly with the size of the address space, which was our goal, but that there are tuning parameters (such as the inter-band gap size) that can make significant (constant multiplier) changes to the performance. Without tuning these parameters AIPRMA is still robust to changes in clustering, and so rather than speculate on possible future session clustering properties, we shall explore other issues that can influence scaling.

## 6.5 Detecting an allocation clash

This far, we have been attempting to design an address allocation mechanism that avoids address allocation clashes. Given the decentralised mechanisms used, we cannot guarantee that clashes will not occur, but we can detect those that do occur and provide a mechanism to cause an announcement to be modified under such circumstances.

If a session directory instance that is announcing a session hears an announcement of another session using the same address, it may retract its own announcement or tell the other announcer to perform the retraction, or both. If a session directory has only made a single announcement then the clash is likely to be because of propagation delay, and so simply retracting the announcement is possible. However, it may be that the site cannot respond because it did not hear the new announcement due to some temporary failure, so under such circumstances we would like other sites to be able to report the clash. Thus a better mechanism is for a site discovering a clash to report it immediately if it was announcing a session using this address, and for other sites to be able to report it after some delay if they do not hear an address clash report.

Thus we end up with a three phase approach:

1. A site that has had a session announced for some time discovers a clash with that session and re-sends its announcement message immediately. This will typically not occur unless a network partition has been resolved recently.

2. A site that just announced a session (whether new or pre-existing) sees another session announced with the same address within a small time window. Such a clash may occur due to propagation delay. It immediately sends a new announcement with a modified address.

3. A third party that has not announced this session sees a session announcement with an address that clashes with one of the sessions in its cache. It waits to see if the cached entry is re-announced by someone else, or if the new session is modified to resolve the clash. If neither of these has occurred after a certain amount of time, it re-announces the session on behalf of its originator.

This approach means that existing sessions will not be disrupted by new sessions. Existing sessions can only be disrupted by other existing sessions that had not been known due to network partitioning.

Allowing third parties to defend existing addresses helps cope with cache failures and partitionings where the two announcers are partitioned from each other but a third party can still communicate with both systems. However to avoid an implosion of responses, a distributed algorithm must be used to decide which third party should send its response at which time.

The simplest such distributed algorithm involves delaying a response by some random delay to allow other sites the possibility to respond. If no other site responds before this time interval elapses, then a site sends. To investigate how long this delay should be, we simulated such a generic multicast "request-response" protocol. Similar mechanisms are used in SRM[24] but in the address allocation case the group is likely to be larger, the delay matrix unknown, and even the size of the receivership unknown. In SRM and here, a receiver that receives a "request" delays its response by a value chosen randomly from the uniform interval [$D1$:$D2$], and cancels its response if it sees another receiver respond within this delay period. In this case $D1$ is chosen so that the originator of an announcement can be expected to have had a chance to reply and suppress all other receivers. The value of $D2$, the topology, and the number of receivers will determine the expected number of responses and the delay before the first of those responses is received.

We can get an upper limit on the number of responses we obtain by simplifying the situation. If the highest round trip time between group members is $R$, then we can regard the interval [$D1$:$D2$] as being $d$ buckets of size $R$. The number of ways that $n$ packets can be placed into $d$ buckets (numbered from $1$ to $d$) is $d^n$, and each of these is of equal probability. The number of ways $k$ out of these $n$ packets can be placed in bucket $b$ and the remainder placed in the other buckets is given by

$$\frac{n!}{k!(n-k)!}(d-1)^{n-k}$$

and so the probability of $k$ packets being in bucket $b$, $p_{k,b}$, is given by:

$$p_{k,b} = \frac{n!}{k!(n-k)!}\frac{(d-1)^{n-k}}{d^n}, 1 \leq b \leq d$$

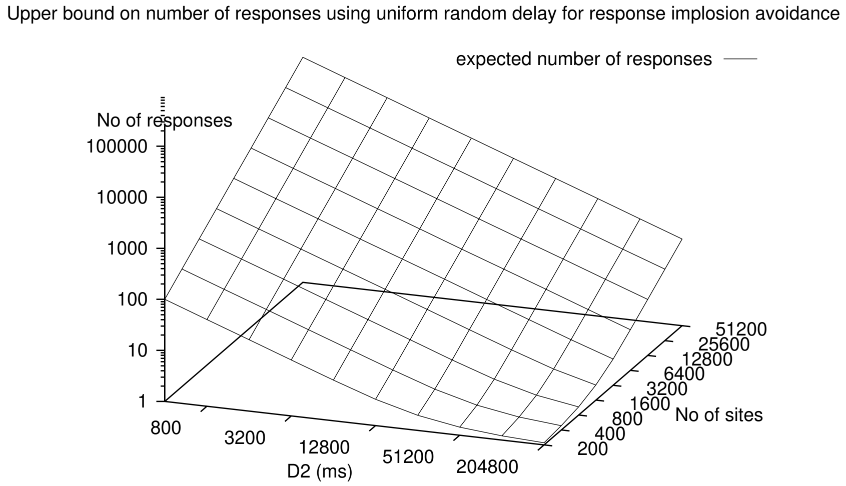


Figure 6.16: Upper bound on number of responders with discrete uniform delay interval

Given that $k$ packets are in bucket $b$, the probability, $z_b$, of no packets being in buckets 1 to $b-1$ is given by:

$$z_b = \left(\frac{d-b}{d-1}\right)^{n-k}$$

As $k$ responses can be obtained by $k$ packets being in bucket 1, or by $k$ packets being in bucket $b$ and no packets being in buckets 1 to $b-1$, the expected number of packets, $E$, is given by:

$$E = \sum_{k=1}^{n} k \sum_{b=1}^{d} p_{k,b} z_b \tag{6.2}$$

This isn't the simplest derivation of this value, but we will wish to experiment with non-uniform bucket probabilities, and so its general form suits our purposes.

Equation 6.2 merely gives an upper limit on the number of responses because it ignores the fact that shorter round trip times are possible, and it ignores suppression *within* a bucket because discrete buckets are used.

Figure 6.16 graphs equation 6.2 for a range of values of $d$ and $n$ for $R = 200ms$.

To investigate the effects of variable delay, realistic topologies, and suppression within a bucket we must resort to simulation. To generate realistic topologies, and to be able to investigate the dependence of the scheme on the multicast routing scheme, we generated topologies as follows:

- The "space" is a square grid. Nodes are allocated coordinates on this grid.
- A new node is connected to its nearest neighbour on the grid. Thus the first few nodes create

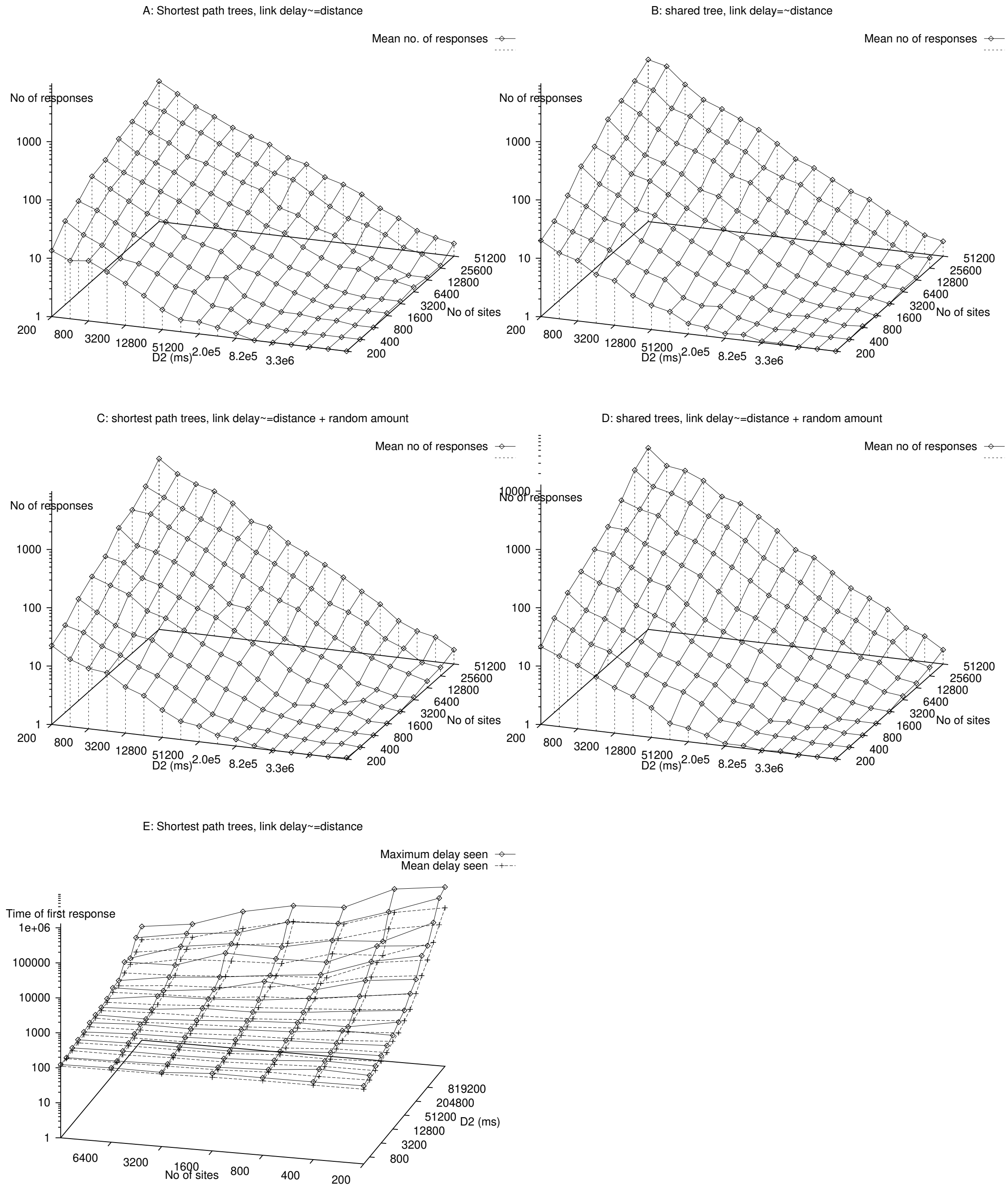


Figure 6.17: Simulations of Generic Multicast Request-Response Protocol

the "backbone" long links and later nodes provide more local clustering. This creates a tree similar to shared trees created by CBT and sparse-mode PIM.

- Optionally, nodes $a$ through $b$ are additionally connected to another pre-existing node at random. $a$ and $b$ are $n/30$ and $n/20$ respectively where $n$ is the total number of nodes. This provides redundant backbone links which can be exploited to form source-based shortest path trees to simulate DVMRP and sparse and dense mode PIM.

The resulting graph is very similar to those generated by Doar[20] and has a hierarchical structure, with a variable number of multiple paths that together make this class of simulations a reasonable model of the real internet. Link delays were primarily based on distance between the nodes forming the link, plus optionally a random per-hop amount on a per-packet basis to simulate queuing fluctuations.

Figure 6.17 shows the results of these simulations for varying sizes of networks and values of $D2$. Figures 6.17A and 6.17C show the results for a source-based shortest-path tree multicast routing algorithm and figures 6.17B and 6.17D show simulated shared trees. Figures 6.17C and 6.17D additionally simulate jitter caused by variable queuing delays.

These results indicate that suppression is insufficient to reduce the number of respondees to close to one for large group sizes without incurring significant delays when the number of potential respondees is small. They indicate a small difference between shortest-path trees and shared trees in terms of the suppression process[6], but not one that greatly affects the choice of mechanism.

Thus we need to modify the basic algorithm for use in the circumstances we are considering. SRM modifies the algorithm by making the delay dependent on the previously measured round-trip delay from the data source, but we clearly cannot do this. However there are a number of things we can do to help.

Firstly, we can reduce the number of possible responders by initially only allowing the sites that are actually announcing sessions to respond. This has the advantage that we know the number of announcing sites, and so we can use this as a parameter in any solution. These sites should be distributed throughout the network, so they should approximate the distribution of all sites. Sites that are not session announcers can always be allowed to respond later by setting their $D1$ value to the value of $D2$ of the announcing sites.

Secondly, we can change the uniform random interval to be a non-uniform random interval.

Lastly, we can arbitrarily rank the sites using any additional information that we have.

[6] the number of respondees is smaller with shortest-path trees than with shared trees

### 6.5.1 Non-uniform random interval

If instead of choosing a delay uniformly from the interval $[D1, D2]$, we choose a delay from an interval with a different distribution, we can reduce the value of $D2$ and hence reduce the worst-case delay without increasing the expected number of responses.

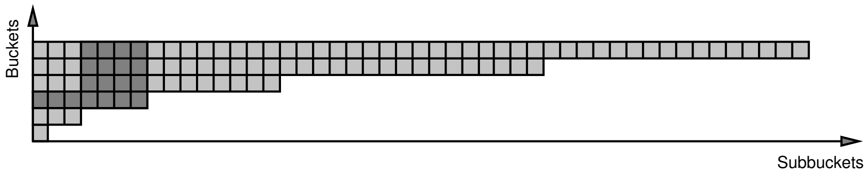


Figure 6.18: Bucket-to-subbucket mapping for the random exponential delay

An exponential distribution has desirable properties to use in such a scenario. Consider again the upper bound given by equation 6.2, but instead of having $d$ buckets of equal probability, we have $d$ buckets such that the probability of bucket $b$ (where $1 < b \leq d$) is double that of bucket $b-1$. Numbering the buckets from $1$ to $d$, this is equivalent to choosing uniformly from $2^d - 1$ sub-buckets, where the bucket $b$ contains $2^{b-1}$ sub-buckets (see figure 6.18). Thus the number of ways $n$ packets can be placed in $2^d - 1$ sub-buckets is $(2^d - 1)^n$, and each of these has equal probability. The number of ways $k$ out of $n$ packets can end up in bucket $b$ is now given by:

$$\frac{n!}{k!(n-k)!} 2^{(b-1)k} (2^d - 2^{b-1} - 1)^{n-k}$$

Thus the probability, $p_{k,b}$ of $k$ out of $n$ packets being sent in bucket $b$ is given by:

$$p_{k,b} = \frac{n! 2^{(b-1)k}}{k!(n-k)!} \frac{(2^d - 2^{b-1} - 1)^{n-k}}{(2^d - 1)^n}, 1 \leq b \leq d \tag{6.3}$$

Given that $k$ packets are in bucket $b$, the probability, $z_b$, of no packets being in buckets 1 to $b-1$ is given by:

$$z_b = \left( \frac{2^d - 2^b}{2^d - 2^{b-1} - 1} \right)^{n-k}$$

Again, the number of expected packets is given by:

$$E = \sum_{k=1}^{n} k \sum_{b=1}^{d} p_{k,b} z_b \tag{6.4}$$

Figure 6.19 graphs equation 6.4 for a range of values of $d$ and $n$ for $R = 200ms$. This gives appropriate numbers of responses for relatively small values of $D2$. Note that unlike in figure 6.16,

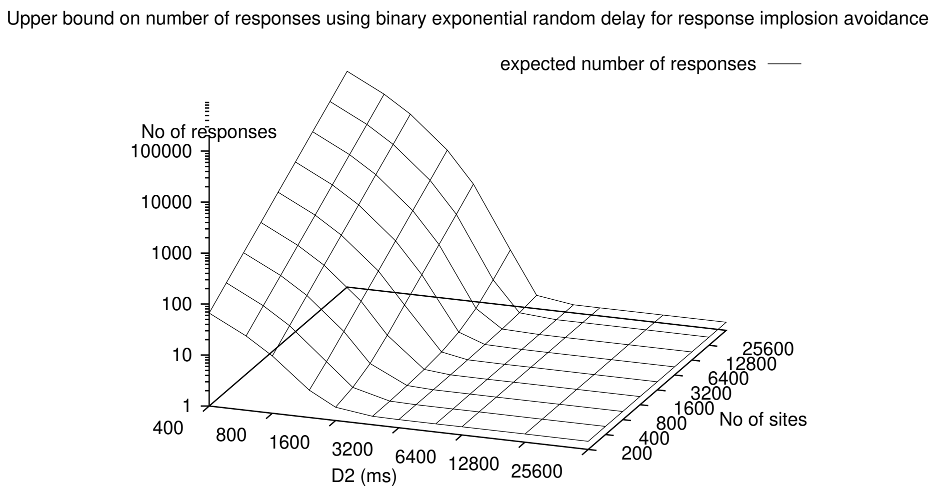


Figure 6.19: Upper bound on number responders with discrete binary-exponential delay interval

the curve does not tend to one response in the extreme[7], and this is the small price we pay for using an exponential in this formula.

Again, this curve represents an upper bound on the number of responses, and we must simulate the behaviour to take account of differing round-trip times and more natural suppression of responses. Figure 6.20 shows the results of simulating this exponential function in a continuous form to randomly choose a delay before responding. The simulator used is identical to that used for figure 6.17 except for the change in delay function. The precise delay, $D$, for a group member is generated using:

$$D = r.log_2((2^d - 1)x + 1)$$

where $d = \frac{D2-D1}{r}$, $r$ is the maximum RTT, and $x$ is a random number chosen uniformly from [0:1]. In practice, a dependence on an *accurate* estimate of RTT is unnecessary, and is introduced here to ensure that figures 6.19 and 6.20 have the same time axis. The value of $d$ is of primary importance in determining the number of responses.

We can conclude from comparing the simulation with the simplified theoretical prediction that suppression occurring within one RTT is significant only when the number of responses is large, and that this is a regime in which we do not wish to operate.

Using a delay chosen from an exponential random distribution results in a sharp distinction below which the number of responses is large and above which it is small. This cutoff point only

[7] the limit in this case is a mean of 1.442698 responses

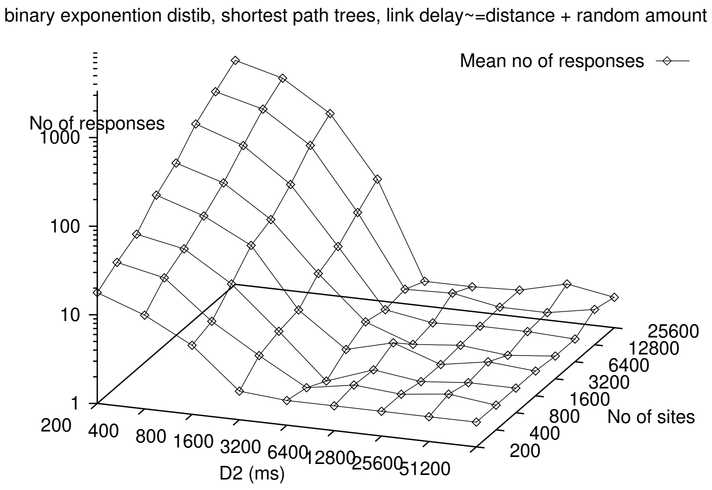


Figure 6.20: Simulations of Generic Multicast Request-Response Protocol with delay chosen from an exponential distribution rather than a uniform interval

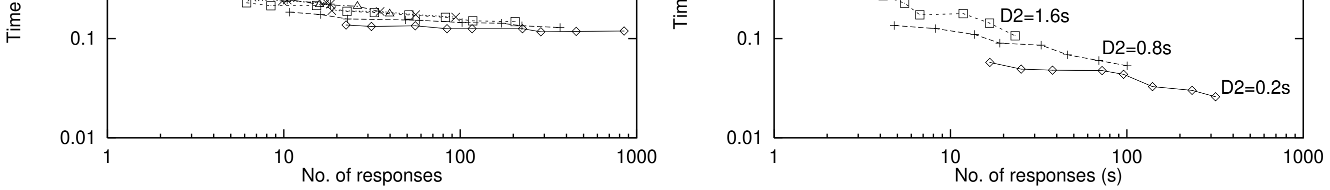


Figure 6.21: Simulations of Generic Multicast Request-Response Protocol performance for both uniform and exponential random delay

increases slowly with the size of the multicast group.

The number of responses is not the only important value; equally important is that the delay before the first response is not excessive. A few seconds delay is acceptable for this application, but hundreds of seconds is probably not so as this will not permit sufficiently rapid retraction of a announcement with an address clash. Figure 6.21 shows the mean number of responses against the mean delay before the first response for the simulations in figure 6.17C and figure 6.20. A curve is shown for each value of $D2$; each curve shows eight points corresponding to the number of receivers ranging from 200 to 25600. Thus although both uniform and exponential random delays provide acceptable behaviour (around two responses and one second delay), the uniform random delay is very dependent on the size of the potential receiver set, whereas the exponential random delay allows us to choose a value of $D2$ that suits a wide range of receiver sets. As we do not know a-priori how many receivers might know about a particular clash, it is clear that the exponential random delay is much simpler to deploy to achieve acceptable behaviour.

### 6.5.2 Other possible approaches

Instead of adjusting the distribution from which members choose a random delay, we can also use other natural differences to avoid multiple responses. SRM uses the delay from the data source to choose receivers close to a loss point to request retransmissions, and this has the very desirable property that members downstream from the responding member are *deterministically* suppressed. We do not have a delay matrix available in a session directory, so we cannot achieve this deterministic suppression, but we could choose between receivers based on other properties.

Of the properties we have available, the hop-count from the "requester"[8] is available if the announcement was made with a fixed time-to-live (TTL). This is not a really good predictor of delay from the requester, but does have some correlation with the delay.

Figure 6.22 shows the number of responses expected with delay, $D$, derived from the hop-count $t$ using:

$$D = \frac{(D2 - D1)t^2}{t_{max}} + D1$$

where $t_{max}$ is the maximum hop count. We use $t^2$ to increase the effect of hop-count and more heavily favour sites close to the requester.

In practice this does not produce a vary fast fall-off in number of responses, although it is a substantial improvement on choosing delay from a uniform random distribution. However, for

[8] the "requester" is the sender of a session announcement for which other members know there is an address clash

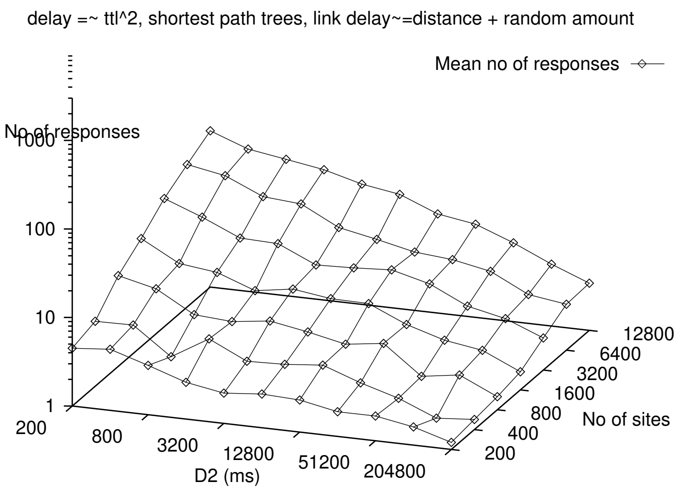


Figure 6.22: Simulations of Generic Multicast Request-Response Protocol with delay chosen from a uniform interval based on $hop - count^2$

our purposes, it is envisaged that sites close to the requester will often *not* be in possession of the announcement that the request clashes with as they may have suffered from the same failure, and so such a weighting is not as useful as it would be in a reliable multicast protocol. Thus *for this application* we prefer a response mechanism that does not distinguish between respondees by location but instead by purely random mechanisms as we believe this is more likely to result in a fast response from a site with the relevant information.

## 6.6 Conclusions

The announcement mechanism in sdr is extremely robust, and can be very timely (compared to alternative mechanisms). Using the same mechanism for session announcement and multicast address allocation is elegant, but this analysis shows that this approach has some limitations.

Using the same mechanism means that if multicast address allocation is to scale reasonably, the following requirements are placed on the session announcement mechanism:

***The session announcement rate must be non-uniform.*** To get multicast address allocation to scale reasonably, figure 6.8 shows that the mean propagation delay must be low. This indicates that the announcement rate should be non-uniform. Optimally, it should start from a high announcement rate (say a 5 second interval) and exponentially back off the rate until a low background rate is reached. This uses the bandwidth effectively from the point of view of reducing mean propagation delay due to packet loss, but as the background rate will now be lower for the same total bandwidth, missing sessions will take longer to be discovered.

***The same announcement channel must be used by all announcements of the same scope.*** This is necessary for the session directory to be able to build up a complete list of sessions in the scope range so that it can perform multicast address allocation. However, this means that all sites must receive all appropriately scoped session announcements, which may be desirable whilst the MBone is relatively small, but ceases to be so as the MBone scales and distinct user groups emerge. As this happens, the amount of bandwidth dedicated to announcements would have to increase significantly or the inter-announcement interval would become too long to give any kind of assurance of reliability.

To support these distinct groups we would like to be able to dynamically allocate new announcement addresses for certain categories of announcement, and only announce the existence of the category on the base session direction address. This mechanism would function in a very similar way to dynamic adaptive naming in CCCP[38], and would allow receivers to decide the categories for which they receive announcements, and hence the bandwidth used by the session directory.

Whilst a session directory is using the same mechanism for announcements and address allocation, this is not possible[9]

***A mechanism must be provided to detect and correct address allocation clashes.*** Thus a global session announcement may be made, and then be "corrected" as the allocation clash is discovered.

### 6.6.1 Beyond sdr: Further Improving Scalability

Despite the simplicity of the sdr session directory model, if we are to achieve scalable session directories and scalable multicast address allocation, this study implies that multicast session announcement and multicast address allocation should be separated from each other.

From the point of view of multicast session announcement, this would mean that multiple groups can be used using either a manually configured hierarchy of announcement groups or, more ideally,

[9] In theory, we could partition the address space by category. However, as many categories will only exist in local scopes, this introduces further problems. Such a category-partition-based solution could probably be made to work along similar lines to AIPRMA given a total ordering of categories sorted using scope as a primary index, with an additional announcement address for category address usage summaries, but this introduces more complexity, and is open to denial of service attacks on the summary address. In addition, a locality-based solution is more likely to help with making sparse-mode multicast routing scale than a category-based solution is.

a dynamic arrangement of session categories across announcement groups. This would reduce the state that a session directory needs to keep to be only that in which the user is interested, and would reduce session announcement bandwidth, at least towards the edges of the network.

For multicast address allocation, figure 6.8 which indicates the effects of announcement packet loss on allocation gives most cause for concern. The conclusion to be drawn is that for global sessions, even an extremely good session announcement mechanism with a perfect version of IPRMA cannot expect to allocate an address space of 270 million addresses effectively. It could probably allocate an address space of 65,536 addresses (the current size of the IANA range for dynamically-allocated addresses), but we should be aiming for higher goals.

To further improve the scalability of multicast address allocation, a hierarchy needs to be introduced.

At the lower level of the hierarchy, an address allocation scheme similar to the one described here can be used to allocate addresses from a space of up to 10,000 addresses - this work in this chapter implies that this is a good upper bound on flat address space allocation.

At the higher level of the hierarchy, a dynamic “prefix” allocation scheme should be used based on locality. At a particular location, the lower-level scheme assigns addresses from the prefix allocated to the region encompassing that location. To get good address space packing, the prefixes themselves need to be dynamically allocated too based on how many addresses are in use from the prefix by the lower level address allocation scheme. Dynamic “prefix” allocation (and especially garbage collection) is hard with contiguous masks, but is somewhat simpler with discontiguous masks, and so a Kampai[29] style scheme would be preferable.

Such a hierarchical scheme would associate discontiguous prefixes with regions of the network, and this can also greatly help the scalability of multicast routing. Thus we are proposing to use such a higher-level “prefix” allocation scheme as a part of Border Gateway Multicast-routing Protocol (BGMP)[71], a new upper-level multicast routing protocol intended to introduce hierarchy to multicast routing. This mechanism will use BGP routing exchanges as a form of “multicast” to implement an announce/listen model similar to sdr to perform prefix allocation. The scheme would be operate on a much slower timescale than sdr's allocation of single addresses because the timeliness requirements are much more relaxed and the consequences of a clash much greater.

These mechanisms would help multicast routing scale because the restriction of operating on only one timescale is removed, and hence packet loss only affects the lower-level allocation scheme, and the upper-level scheme can use a more reliable hop-by-hop mechanism as it has much more time to operate in. In addition, the lower-level scheme would only need to announce the addresses in use within the local region, and this improved locality means that more address-usage announcement messages can be sent increasing the timeliness significantly.

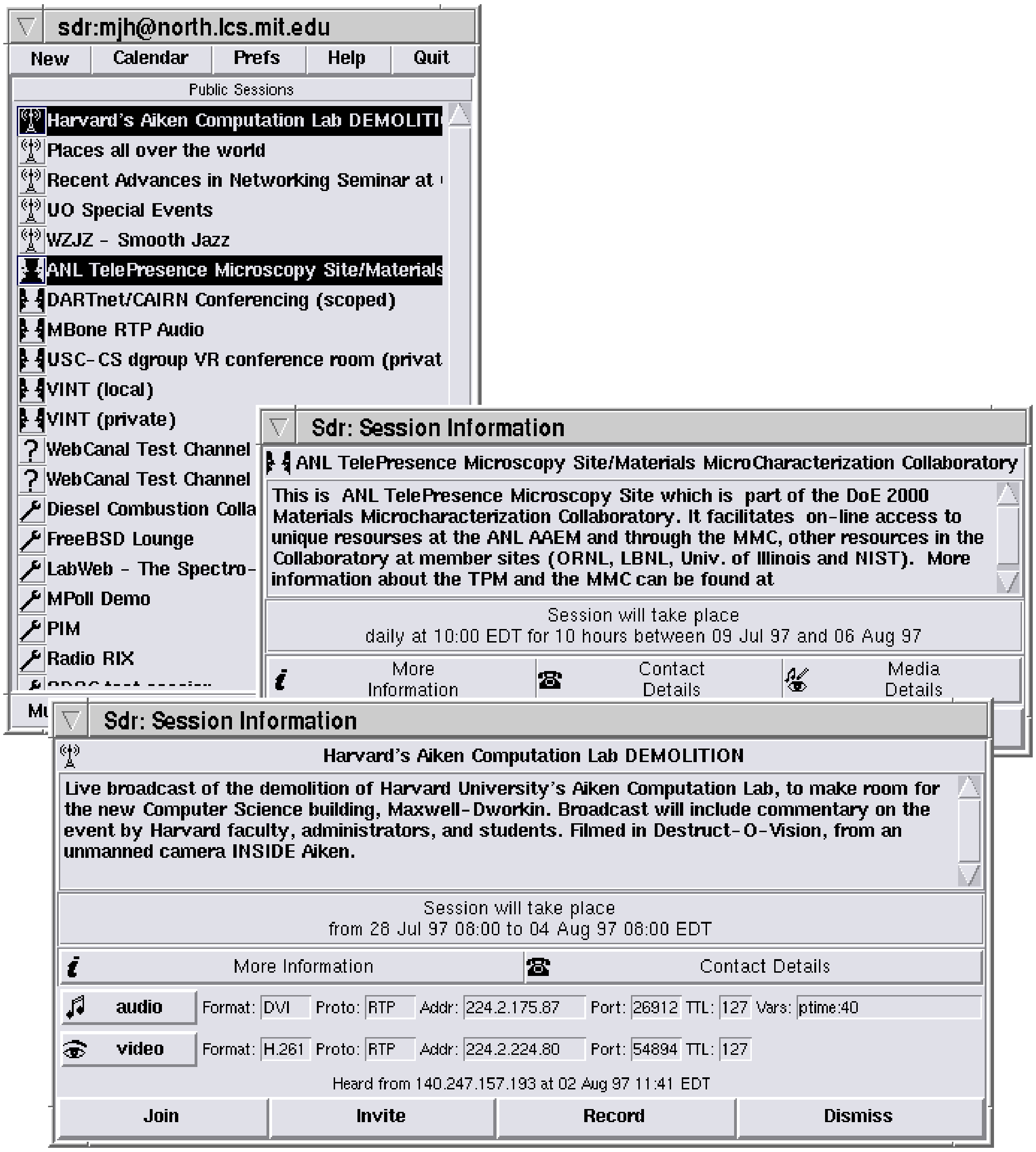


Figure 6.23: The SDR multicast session directory

# Chapter 7

# Conclusions

In this thesis we have examined architectures for building scalable multicast-based multimedia conferencing systems. Most of the prior work in this area had concentrated on audio and video transmission, leaving many aspects relatively unexplored. This thesis has attempted to explore some of the more important parts of this unexplored territory, namely, shared tools, and conference initiation mechanisms. The primary goals of this work were:

- to examine whether more complex and interactive scalable conferencing systems could be built, and to explore novel solutions for such systems.
- to see which factors affected the scaling of these systems.
- to evaluate the design principles of application level framing and design for inconsistency as applied to conferencing systems.

In this final chapter, we will first summarise lessons learned from each of the individual systems and finally attempt to extract general conclusions about the design of scalable conferencing systems.

## 7.1 MBone Performance

In chapter 4 we presented an analysis of the performance of the current MBone so that we could design systems that addressed real problems. The principle questions we asked were about the nature of packet loss in the network. A rough idea of mean loss rates was already known from using audio and video conferencing tools, but information about loss correlation was hard to come by. Three forms of loss correlation are of particular interest:

**spatial loss correlation** - whether packet losses seen at one site are also seen at others. This is primarily related to how many links in the multicast distribution tree are suffering some form of congestion.

**temporal correlation** - whether packet losses appear to be random to receivers, or whether they occur in bursts or periodically.

**correlation of loss with data rate** - how the transmitted data rate affects packet loss, and hence how to design applications that are network friendly.

We should be careful about drawing too general a set of conclusions from a limited observation of the network behaviour, but we believe we the measurements presented are not atypical of MBone behaviour during the period between 1995 and 1997.

All the large sessions monitored showed broadly similar spatial loss correlation patterns - usually a significant set of receivers experience good reception quality with very low loss rates, but a larger set of receivers are located behind a small number of significantly congested links, and hence experience loss rates of several percent. A small number of receivers suffer loss rates in excess of 30% - the links causing this loss tend to be either feeding a single leaf site, or tend to be international links where there is a discontinuity in the price of bandwidth. Scattered across this general picture are a large number of links that provide relatively low but still noticeable loss. Thus although there is often fairly good large scale spatial loss correlation between receivers, the uncorrelated loss is significant enough to cause problems for reliable multicast applications that do not take this affect into account. This property significantly affected our design for the shared text editor in chapter 5.

For temporal correlation, modeling loss as random may be sufficient for many applications. However, for applications that are attempting to provide loss protection, for example through redundancy or packet-level forward error correction, it is worth examining how loss deviates from a random model as there are small gains to be made by doing so. Periodic loss was also found, which appears to be not universal, although it has been observed by other studies too. Such loss is probably a defect in certain routers or routing protocols, and the problem should be diagnosed and cured rather than designing applications to work around this “feature”.

For the design of congestion control schemes, information about how loss rate is determined by data rate is of prime significance. Our results indicate that for data flows which do not use a large proportion of the bottleneck link, as is the case in the network backbone, there is not a good correlation between data rate and loss rate. Although this result should not be all that surprising, several multicast congestion control schemes ([4],[57]) seem to make the assumption that they can detect the effect of their own traffic in the loss rates observed. We conclude that such simple approaches to congestion control will not work, and one or more of the following is likely to be required for multicast congestion control:

- The distribution tree can be broken up into multiple trees, each with only one significant bottleneck. Within each tree, a TCP-style congestion control scheme is used which can adapt at timescales on the order of a round-trip time.
- Loss rate feedback from the receiver suffering most loss might be used to determine an appropriate medium-term bandwidth based on estimating the equivalent bandwidth a TCP-connection would have achieved to that receiver.
- Drop-tail FIFO queuing in the network routers could be replaced with RED and/or a form of fair queuing to provide better feedback as to an appropriate data rate. In such a network, we would expect better correlation between loss rate and bandwidth, and therefore be more likely to be able to design schemes that adapt on timescales longer than a few round-trip times.

The results in chapter 4 indicate that there is significant research remaining to be done in this area.

## 7.2 The Network Text Editor (NTE)

In chapter 5 we presented the design of a multicast based shared text editor. NTE was in part an attempt to see how far ALF can be taken in pursuit of scalability.

The requirements for an interactive system such as a shared editor, so that it is robust, scalable, consistent, deterministic, and never locks users out due to network problems, are unfortunately self-contradictory. The approach we took with NTE was that consistency can be temporarily sacrificed so that the system can perform well and scale. There are many details that need to be worked through in order to design scalable data distribution mechanisms, but the principle problem remaining then is one of consistency control. The systems must be able to detect inconsistencies, and in a distributed fashion, eventually converge on a consistent solution. In the general case, this cannot be guaranteed, and so in chapter 5 we devote much space to restrictions that must be placed on the data model for such a shared editor to ensure that inconsistencies are always detectable and correctable.

The reliable multicast protocol embedded in NTE is designed specifically around the requirements for such a shared text editor. There is a great deal of natural redundancy inherent in the application that can be exploited if the network protocol is allowed to do so. The results about uncorrelated loss from chapter 4 make it vital that we do design such tools around redundancy mechanisms if they are to scale without generating very significant additional load due to retransmissions.

The resulting shared editor protocol is completely ALF oriented. Application-level redundancy is used to prevent network retransmissions, the application data model is designed to permit consistency problems caused by network failures to be repaired, and the data update mechanism relies upon an clock-synchronisation algorithm that is implicit in data exchanges. Thus the reliability, consistency and data storage and data distribution models are inter-linked. This is an extreme in the spectrum of ALF designs, and results in a system that scales and performs extremely well even in the presence of network failures and packet loss.

However, going to such an extreme makes generalisation of the concepts difficult. From a software engineering or protocol design perspective, we would like to be able to reuse much of a design for similar systems without losing the scalability and robustness properties. The next challenge is to take the properties of this design that make it robust and scalable - those of redundancy and relaxed consistency - and to see if the core of a more general purpose application framework can be abstracted so that new tools can be built without needing to start from scratch.

## 7.3 Multicast Session Directories and Address Allocation

In chapter 6 we started out by trying to build a scalable multicast session directory, sdr, that would remedy some of the deficiencies of LBL's original sd session directory. Sdr has been very successful to the extent that it is now the principle multicast session directory in use, and therefore examining the scaling issues of its address allocation mechanisms was of great importance. In examining the problem, we performed simulations of an idea by Van Jacobson for scalable multicast address allocation called IPRMA. Unfortunately this identified severe scaling limitations of the original IPRMA scheme.

These scaling problems led us to develop a variant of IPRMA called Deterministic Adaptive IPRMA, which does not suffer from the same scaling problems. However, when modeling the performance of even a perfect IPRMA-style scheme in the context of a session directory, it becomes clear to us that scaling limitations are introduced by packet-loss affecting the session announcement mechanism, and that these limitations become more severe as the packet loss rate increases and as the address space being allocated becomes larger. A realistic upper bound on the size of the address space that should be allocated if reasonable address packing and low address-clash probability are both required is about 100,000 addresses.

These limitations then led us to investigate the feasibility of schemes for detecting and rectifying address clashes when they occur. We modeled and simulated a multicast request-response protocol with up to 50,000 participants, and devised a modified version of the original SRM algorithm that is especially suited to this sort of scenario.

Thus with a session directory that uses the same mechanism for session announcement and session allocation, we expect to be able to scale multicast session directories far beyond where they are today, to allocate addresses with low probability of clashes for medium sized address spaces, and to detect and remedy clashes when they do occur.

However, it also becomes clear that if multicast really becomes ubiquitous, then such a combined session announcement and multicast address allocation mechanism is the cause of some scaling limitations. If we split address allocation from session announcement, we can build a (two-layer) hierarchical address allocation system that we believe will suffice to allocate the whole IPv4 class D address space efficiently if required. We are currently in the process of designing such an upper-level "prefix" allocation scheme. Such a split also allows session announcements to be hierarchically distributed over multiple announcement groups depending on the categorisation of the session. This would greatly help the scaling of the session announcement mechanism, both from the point of view of session directory state and network bandwidth.

The session directory is an example of an applications that must scale far better than any other conferencing system component because it must service a user set that is the union of all other conference user sets. Although sd and sdr use multicast and the announce/listen model, and although they are explicitly designed so that the size of the receiver set does not affect their scaling, they can still experience scaling problems. We have rough draft solutions to each of these scaling problems, but what originally seemed like a good optimisation - using the same messages to announce sessions and allocate addresses - turns out to prevent the solutions from being deployed. We are currently working on a revised architecture that will avoid this problem in the future.

It is interesting to note that the same solution that is likely to make multicast address allocation scale also makes multicast routing scale. Hierarchy is often used to solve scaling issues, but usually it involves compromise elsewhere. In this case it appears that a single hierarchical solution can solve both significant problems facing multicast, but we will have much further evaluation to perform to be sure that new problems are not introduced in the process.

## 7.4 Application Level Framing and Design for Inconsistency

Our final goal in this thesis was to throw light on the usefulness of ALF and "Design for Inconsistency" as design principles for scalable synchronous groupware systems.

NTE was designed with ALF and inconsistency in mind from the start. The result is a shared editor that performs well, but is not general purpose at all.

The multicast session directory occupies a slightly different part of the design space because it is a meta-conferencing tool rather than a component of a conference. This made the scalability requirements much more extreme than with NTE, but the announce/listen model is clearly up to this task.

There is a possibility that it seems as if multicast-based applications would necessarily always employ application level framing - this is emphatically not the case. Many people (for example [13][70][73]) have attempted to build reliable multicast protocols as a separate middleware layer, and have designed solutions that create a reliable multicast transport abstraction. These solutions either do not scale, or do not suitably address the many possible failure modes that can occur with multicast, and we do not see them in widescale use today.

The only *general* design principle we know of that helps build multicast systems that scale and respond appropriately to failures is Application Level Framing. It is probably possible to build transport protocols for multicast bulk-data transfer that perform reasonably and scale fairly well without conforming to the ALF model, but the reason such transport layers can be built is because the application semantics have been built into the transport layer. For other applications, and particularly for interactive groupware, the desired failure semantics are much more complicated, involving application-level consistency issues. As the failure semantics can only be understood in application terms, and as failure is by no means a binary function in most multicast systems, ALF seems to be a requirement.

Although ALF is *necessary* for the design of scalable and robust systems, this does not mean it is *sufficient*. The principle of lightweight sessions attempts to provide additional constraints so that conferencing systems will be robust and scalable. The most important aspect of lightweight sessions turns out to be the announce/listen model for multicast applications. This principle combined with the peer-to-peer architectures it typically implies goes a great deal of the way to providing a suitable guideline for building many scalable systems.

There comes a point though where design principles are not sufficient, and the details of the individual application must be taken into account. Because appropriate responses to the spectrum of multicast failures are much more varied than for unicast, it becomes important that the application is involved in all these decisions. This makes the design of general purpose middleware for conferencing much more difficult, and in many cases it possibly is undesirable.

The key theme that runs through all the chapters of this thesis is that to build scalable and robust conferencing or meta-conferencing systems, the application must allow consistency to be relaxed. This occurs all through the design of NTE, and especially in session directories with multicast address allocation. Making explicit design decisions around the expectation of inconsistency is a crucial design principle for these systems. We regret we have no general guidelines for how to deal with inconsistency appropriately as this appears to be totally dependent on the application itself.

# Bibliography


[1] E. Amir, S. McCanne and M. Vetterli, “A Layered DCT Coder for Internet Video.” IEEE International Conference on Image Processing. September, 1996. Lausanne, Switzerland.

[2] A. Ballardie, “Core Based Trees (CBT) Multicast Routing Architecture”, INTERNET-DRAFT, May 1997

[3] M. Beaudouin-Lafon, A. Karsenty, “Transparency and Awareness in Real-time Groupware Systems”, Proc. ACM UIST'92, Monterey, Ca., Nov. 1992.

[4] J.-C. Bolot, T. Turletti, I. Wakeman, “Scalable Feedback Control for Multicast Video Distribution in the Internet” Proc ACM Sigcomm '94, London, ACM, 1994.

[5] J-C. Bolot, T. Turletti, “Adaptive error control for packet video in the Internet”, Proc. ICIP '96, Lausanne, Sept. 1996.

[6] J-C. Bolot, A. Vega-Garcia, “Control Mechanisms for Packet Audio in the Internet”, Proc IEEE Infocom '96, San Francisco, CA, pp. 232-239, April 1996.

[7] J-C. Bolot, H. Crepin, A. Vega-Garcia, “Analysis of Audio Packet Loss in the Internet”, Proc 1995 Workshop on Network and Operating System Support for Audio and Video, pp. 163-174, 1995.

[8] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, “Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification”, INTERNET-DRAFT, May 1997.

[9] S. Casner, S. Deering, “First IETF Internet Audiocast”, ACM Computer Communication Review, Vol. 22, No. 3, pp. 92-97, July 1992.

[10] D. Clark, D. Tennenhouse, “Architectural Considerations for a New Generation of Protocols”, Proc ACM SIGCOMM '90, Philadelphia, Pa, 1990.

[11] D. Cohen, “Specifications for the Network Voice Protocol NVP”, IETF RFC 741, Nov 1977

[12] J. Cooperstock, S. Kotsopoulos, “Why use a fishing line when you have a net? an adaptive multicast data distribution protocol,” in Proc. of Usenix Winter Conference, 1996.

[13] W. Dabbous, B. Kiss, “A Reliable Multicast Protocol for a White Board Application”, RR-2100, INRIA, Sophia Antipolis, France, Nov. 1993.

[14] S. Deering, D. Cheriton. “Multicast routing in datagram internetworks and extended LANs.” ACM Transactions on Computer Systems, pp. 85–111, May 1990.

[15] S. Deering, C. Partridge, D. Waitzman, “Distance Vector Multicast Routing Protocol”, IETF RFC 1075, Jan 1988.

[16] S. Deering, A. Thyagarajan, B. Fenner, “mrouted - IP multicast routing daemon”, Unix Manual Pages, Xerox Palo Alto Research Center, 1997

[17] S. Deering, “Host extensions for IP multicasting”, IETF RFC 1112, Jan 1989.

[18] Deering, Estrin, Farinacci, Handley, Helmy, Jacobson, Liu, Sharma, Thaler, Wei, “Protocol Independent Multicast-Sparse Mode (PIM-SM): Motivation and Architecture”, Internet Draft, Nov 1996

[19] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. G. Liu, and L. Wei, “An Architecture for Wide-Area Multicast Routing,” in SIGCOMM Symposium on Communications Architectures and Protocols, (London, UK), pp. 126-135, Sept. 1994.

[20] M. Doar, “A Better Model for Generating Test Networks”, IEEE Global Telecommunications Conference/GLOBECOM'96, London, November 1996.

[21] C. Ellis, S. Gibbs, “Concurrency Control in a Groupware System”, in Proc. ACM SIGMOD '89, Seattle, Wa., 1989

[22] Estrin, Farinacci, Helmy, Thaler, Deering, Handley, Jacobson, Liu, Sharma, Wei, “Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification”, INTERNET DRAFT, Sept 1997.

[23] W. Fenner, S. Casner, “A “traceroute” facility for IP Multicast”, Internet Draft (Work in Progress), 1996

[24] S. Floyd, V. Jacobson, S. McCanne, "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing", Proc ACM SIGCOMM 95, August 1995, pp. 342-356.

[25] S. Floyd, V. Jacobson, C. Liu, S. McCanne, and L. Zhang, "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing". To appear in IEEE/ACM Transactions on Networking.

[26] S. Floyd and V. Jacobson, "Random Early Detection gateways for Congestion Avoidance" IEEE/ACM Transactions on Networking, V.1 N.4, August 1993, p. 397-413.

[27] S. Floyd and V. Jacobson, "The Synchronisation of Periodic Routing Messages", ACM Transactions on Networking, Vol. 2, No. 2 (April 1994), pp. 122 - 136.

[28] J. Forgie, "ST - A Proposed Internet Stream Protocol", IEN 119, M. I. T. Lincoln Laboratory, 7 September 1979.

[29] P. Tsuchiya, "Efficient and Flexible Hierarchical Address Assignment", TM-ARH-018495, Bellcore, February 1991.

[30] Ron Frederick, "nv - X11 videoconferencing tool", UNIX Manual Pages, Xerox Palo Alto Research Centre

[31] A. Ghosh, "mcollect - collect data from the mserv server (mwatch)", Unix manual page, University College London.

[32] A. Ghosh, "mserv - multicast map server (mwatch)", Unix manual page, University College London.

[33] A. Greenberg and N. Madras, "How fair is fair queuing?," Journal of the ACM, vol. 39, no. 3, pp. 568-598, 1992.

[34] A. Haake, J. Haake, "Take CoVer: Exploiting Version Management in Collaborative Systems", in Proc. InterCHI' 93, Amsterdam, Netherlands, 1993.

[35] M. Handley, P. Kirstein, A. Sasse, "Multimedia Integrated Conferencing for European Researchers (MICE): Piloting Activities and the Conference Management and Multiplexing Centre", Computer Networks and ISDN Systems, vol 26, 275-290, 1993

[36] M. Handley, S. Wilbur, "Multimedia Conferencing: from prototype to national pilot", Proceedings of INET ' 92, Kobe, Japan, pp 483-490, Internet Society, Reston, VA, 1992

[37] M. Handley, P. Kirstein, A. Sasse, "Multimedia Integrated Conferencing for European Researchers (MICE): Piloting Activities and the Conference Management and Multiplexing Centre", Computer Networks and ISDN Systems, V 26, pp275-290, Nov 1993

[38] H. Handley, I. Wakeman, J. Crowcroft, "The Conference Control Channel Protocol (CCCP): A Scalable Base for Building Conference Control Applications", Proc. ACM Sigcomm '95, Cambridge, MA., USA, ACM, 1995.

[39] V. Hardman, A. Sasse, M. Handley, A. Watson, "Reliable Audio for Use over the Internet" Proc INET '95, Hawaii, Internet Society, Reston, VA, 1995.

[40] M. Handley, V. Jacobson, "SDP - Session Description Protocol", Internet Draft, Sep 1997.

[41] M. Handley, "SAP - Session Announcement Protocol", Internet Draft, Nov 1996.

[42] M. Handley, H. Schulzrinne, "SIP: Session Initiation Protocol", Internet Draft, July 1997

[43] M. Handley, J. Crowcroft, "Network Text Editor (NTE): A scalable shared text editor for the MBone", Proc. of ACM Sigcomm 97, Canne, France, ACM, 1997

[44] ISO, "Information Processing Systems - Open Systems Interconnection - Basic Reference Model" ISO-7498, 1984.

[45] ITU, "Recommendation H.320 - Narrow-band visual telephone systems and terminal equipment", Version 3, March 1996, ITU, Geneva, Switzerland.

[46] ITU, "Recommendation H.323 - Visual telephone systems and equipment for local area networks which provide a non-guaranteed quality of service", November 1996, ITU, Geneva, Switzerland.

[47] ITU, "Recommendation H.261 - Video codec for audiovisual services at p x 64 kbit/s", Version 3, March 1993, ITU, Geneva, Switzerland.

[48] ITU, "Recommendation T.120 - Data protocols for multimedia conferencing" July 1996, ITU, Geneva, Switzerland.

[49] ITU, "Recommendation T.124 - Generic Conference Control", August 1995, ITU, Geneva, Switzerland.

[50] ITU, "Recommendation T.126 - Multipoint still image and annotation protocol" August 1995, ITU, Geneva, Switzerland.

[51] V. Jacobson, “Multimedia Conferencing on the Internet”, Tutorial Notes - ACM Sigcomm 94, London, Sept 1994.

[52] V. Jacobson, S. McCanne, “vat - remote audio conferencing tool”, UNIX Manual Pages, Lawrence Berkeley Laboratory, Berkeley, CA.

[53] V. Jacobson, S. McCanne, “Using the LBL Network Whiteboard”, Lawrence Berkeley Laboratory, Berkeley, CA.

[54] K. Lidl, J. Osborne, J. Malcolm: “Drinking from the Firehose: Multicast USENET News”, Proc USENIX Winter 1994, San Francisco, Ca., Jan. 17-21, 1994.

[55] J. Lin, S. Paul, “RMTP: A Reliable Multicast Transport Protocol”, IEEE INFOCOM '96, pp. 1414-1424, March 1996.

[56] S. McCanne, “vic - video conference”, UNIX Manual Pages, Lawrence Berkeley Laboratory, Berkeley, CA.

[57] S. McCanne, V. Jacobson and M. Vetterli, “Receiver-driven Layered Multicast”. ACM SIGCOMM, August 1996, Stanford, CA, pp. 117-130.

[58] S. McCanne, M. Vetterli, “Joint Source/Channel Coding for Multicast Packet Video”. Proceedings of the IEEE International Conference on Image Processing. October, 1995. Washington, DC.

[59] M. Macedonia, D. Brutzman, “MBone Provides Audio and Video Across the Internet”, IEEE Computer, Vol.27 No.4, pp. 30-36, April 1994.

[60] Suvo Mittra, “Iolus: A Framework for Scalable Secure Multicasting”, Proc ACM Sigcomm '97, Canne, France, 1997.

[61] J. Moy, “Multicast Extensions to OSPF”, IETF RFC 1584, March 1994.

[62] J. Munson, P. Dewan, “A Flexible Object Merging Framework”, Proc. ACM CSCW '94, Chapel Hill, North Carolina, 1994.

[63] J. Nonnenmacher, E. Biersack, Don Towsley, “Parity-Based Loss Recovery for Reliable Multicast Transmission”, Proc ACM Sigcomm '97, Canne, France, 1997.

[64] A. K. Parekh, and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks - the multiple node case," in Proc. IEEE Infocom '93, vol. 2, (San Francisco, California), pp. 521-530 (5a.1), IEEE, March/April 1993.

[65] V. Paxson, "Measurements and Analysis of End-to-End Internet Dynamics", UC Berkeley PhD thesis, 1997.

[66] J. Saltzer, D. Reed, D. Clark, "End-to-End Arguments in System Design," ACM Trans. on Computer Systems, Vol. 2, No. 4, November 1984, pp. 277-288.

[67] Henning Schulzrinne, "Voice communication across the Internet: A network voice terminal," Technical Report TR 92-50, Dept. of Computer Science, University of Massachusetts, Amherst, Massachusetts, July 1992.

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

[69] S. Shenker, A. Weinrib and E. Schooler: "An Algorithm for Managing Shared Teleconferencing State", Internet Draft (Expired), July 1995
ftp://ftp.isi.edu/confctrl/docs/draft-ietf-mmusic-agree-00.txt

[70] R. Talpade, M. Ammar, "Single Connection Emulation: An Architecture for Providing a Reliable Multicast Transport Service", In Proceedings of the 15th IEEE Intl Conf on Distributed Computing Systems, Vancouver, June 1995.

[71] D. Thaler, D. Estrin, D. Meyer (editors), "Border Gateway Multicast Protocol (BGMP): Protocol Specification", Internet-draft, Oct 1997.

[72] C. Topolcic, "Experimental Internet Stream Protocol, Version 2 (ST-II)", 10/30/1990 IETF RFC 1190.

[73] B. Whetten, T. Montgomery, S.Kaplan, "A high performance totally ordered multicast protocol", Theory and Practice in Distributed Systems, Springer verlag, LCNS 998.

[74] M. Yajnik, J. Kurose, D. Towsley, "Packet Loss Correlation in the MBone Multicast Network" Proc IEEE Global Internet Conf. , London, Nov. 1996.

[75] L.Zhang, S.Deering, D.Estrin, S.Schenker and D.Zappala, "RSVP: A New Resource ReSerVation Protocol", IEEE Network, pp-8-18, September 1993.