Scalable Multimedia Communication with Internet Multicast, Light-weight Sessions, and the MBone

Steven McCanne
University of California, Berkeley

Abstract

In this survey article we describe the roots of IP Multicast in the Internet, the evolution of the Internet Multicast Backbone or "MBone," and the technologies that have risen around the MBone to support large-scale Internet-based multimedia conferencing. We develop the technical rationale for the design decisions that underly the MBone tools, describe the evolution of this work from early prototypes into Internet standards, and outline the open challenges that remain and must be overcome to realize a ubiquitous multicast infrastructure.

We[1] and others in the MBone research community have implemented our protocols and methods in "real" applications and have deployed a fully operational system on a very large scale over the MBone. This infrastructure — including our audio, video, shared whiteboard tools and protocols — is now in daily use by the large and growing MBone user and research communities and the success and utility of this approach has resulted in commercialization of many of the underlying technologies.

1 Introduction

The Internet Multicast Service [27, 29], a fledging but promising new network technology, extends the traditional, best-effort *unicast* delivery model of the Internet Protocol (IP) with efficient multi-point packet transmission. With Internet multicast, a single packet is sent to an arbitrary number of receivers by replicating the packet within the network at fan-out points along a distribution tree rooted at the packet's source. This extension to IP, often called *IP Multicast*, is now widely supported in commercial routers and is undergoing incremental deployment in the Internet. Because Internet multicast requires new routing protocols and forwarding algorithms but not all routers can be upgraded simultaneously, requisite infrastructure has been incrementally put in place as a virtual multicast network embedded within the traditional unicast-only Internet called the Internet Multicast Backbone, or *MBone*.

In addition to offering an efficient multipoint delivery mechanism, IP Multicast has a number of other attractive properties that make it especially well suited for large-scale applications of multimedia communication. First, IP Multicast is both efficient (in terms of network utilization) and scalable (in terms of control overhead). There are no centralized points of failure and protocol control messages are distributed across receivers. Group maintenance messages flow from the leaves toward the source and are coalesced within the network to facilitate scaling to arbitrarily large groups. Second, IP Multicast offers a rich range of scope-control mechanisms. Applications can limit the "distance" with which a multicast packet travels and the network can be configured with hierarchical "administrative scope" boundaries to localize the reach of a multicast transmission in accordance with administrative policies. Finally, IP Multicast provides a flexible, dynamic, and "anonymous" model for group membership. Senders need not explicitly know about every receiver and receivers need not know an individual sender's identity to receive its traffic. Instead, sources simply send packets destined to a multicast *group* and receivers "tune in" to the transmission by subscribing to that group.

These three properties — scalable protocol performance, user-defined scope-controls, and flexible group membership — combine to make IP Multicast an excellent building block for robust and scalable application-level protocols. Consequently, IP Multicast and its scalability have been heavily exploited in the design of a number of end-to-end protocols and multicast applications such as real-time media transport of audio and video with accompanying control information (e.g., *vat* [57], *rat* [48], *vic* [69], *nv* [40] and *ivs* [101]); reliable multicast transport of persistent data that underlies shared tools like whiteboards and text editors (e.g., *wb* [68] and *nte* [46]) and session directories to create and

[1] Our work on the MBone tools was carried out in collaboration with Van Jacobson at the Lawrence Berkeley National Laboratory under the support of the Director, Office of Energy Research, Scientific Computing Staff, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098 and of the National Science Foundation under grant MIP-93-21302.

announce the existence of multicast sessions (e.g., *sdr* [44]). Over the past few years, these applications — collectively referred to as the "MBone tools" — and their protocols were developed by the MBone research community and fielded at large scale over the MBone itself.

The resulting network architecture, where "thin" application-level protocols are built on top of IP Multicast and its powerful "group concept", has been dubbed the "Light-weight Sessions" (LWS) architecture to emphasize its light-weight, loosely-coupled, and decentralized communication model [55]. In LWS, the collection of multicast senders and receivers, communicating within one or more interdependent multicast groups, and their respective protocols and protocol instances are collectively called a *session*.

An important design goal that has consistently underscored the development of the LWS architecture is an explicit attempt to accommodate the highly heterogeneous and loosely-controlled nature of the Internet. The Internet is large, complex, and composed of heterogeneous components with diverse failure modes and mixed levels of reliability. To account for this, LWS protocols are deliberately designed to be robust and forgiving. For example, if a network link outage causes a temporary network partition, then the higher-level LWS protocols do not "reset" or terminate the session. Instead, the session continues its existence as a set of partitioned sub-sessions. Later, when the network re-forms, the session gracefully recovers and integrates independently generated activity and state across the healed partition. LWS achieves this "robustness" to network failures by exploiting receiver-driven, "soft-state" protocols that treat recovery from such failures as part of normal protocol function.

The robustness and scalability of the LWS architecture are embodied in several design principles that are exploited consistently across the protocols and applications in this framework:

- **Shared Multicast Control.** The LWS framework exploits the multicast service extensively to share information across the session. By multicasting control information in addition to data, hosts can intelligently share information to improve the scalability and robustness of their underlying protocols. Though the use of unicast in place of multicast often appears as an attractive means to save bandwidth by reducing traffic overhead, the overall advantages of employing multicast often amortize the overhead because distributed knowledge gained through the multicast primitive can be exploited to scale higher-level protocols gracefully.

- **Adaptation.** A key tenet in LWS is to adapt the application to the network rather than the network to the application. LWS applications are typically delay-adaptive, load-adaptive, loss-adaptive, or a combination thereof. They make no a priori assumption about the underlying network's quality of service and instead adapt to the available network resources through dynamic measurements. Provisioning real-time network service can then be viewed as an incremental optimization that is deployed in the network only when and where needed.

- **Soft State.** Many LWS applications, application protocols, and network protocols exploit "soft state" [18] based on the "announce/listen" metaphor [93]. In this model, each session member periodically announces its presence and/or a "state binding" by multicasting a message to the group. In turn, receivers tune into the group, listen to the announcements, and assemble the announced state. Each piece of state eventually times out unless refreshed by a subsequent announcement. With soft state, error recovery is "designed into" the protocol and need not be treated as a special case. This leads to highly robust distributed systems that gracefully accommodate network partitions, partial system failures, misbehaving implementations, and so forth.

- **Receiver-driven Design.** If a multicast protocol makes decisions or carries out actions at the source on behalf of its receivers, then the scalability of that protocol is immediately constrained because the source must explicitly interact with an arbitrary number of receivers. Instead, in a receiver-driven design, heterogeneity and scalability are simultaneously tackled by distributing application or protocol decisions across the receivers. To maximize scalability, the sender's protocol should be as "receiver ignorant" as possible.

In this paper, we attempt to articulate these principles in detail and develop the rationale for why IP Multicast and LWS are powerful building blocks for large-scale multimedia communication. We first provide a brief overview and history of the deployment of IP Multicast in the Internet and the development of a number of applications that exploit multicast. Once this historical context is set, we survey the current state of the art and ongoing work in Internet multicast. In particular, we describe:

- the details of the LWS architecture and the development of the Real-time Transport Protocol (RTP) within LWS from prototype protocols in the MBone tools;

- the advent of announce/listen protocols for the creation and advertisement of public multicast sessions;
- the emergence of reliable-multicast applications whose semantics are reflected directly into the design of their underlying network protocols;
- the development of a particular scalable reliable-multicast framework that underlies the MBone whiteboard tool *wb*;
- the design of a layered transmission system and adaptation protocol for heterogeneous transmission of video; and finally,
- the widely-used, albeit preliminary, security model for LWS, based on end-to-end symmetric-key cryptography.

1.1 MBone History

The design of the LWS architecture and MBone tools was an evolutionary process that has deep roots in two earlier and important developments. First, LWS builds heavily upon the Internet architecture and the IP Multicast delivery model. Much of the IP Multicast design philosophy (e.g., its soft-state, receiver-driven architecture and light-weight group membership model) is echoed in the multicast applications and their protocols. Second, extensive experience and experiments with unicast packet audio in the late seventies and early eighties offered an "existence proof" that packet audio was both feasible and practical. This earlier work demonstrated that packet-switched networks in tandem with delay-adaptive applications could deliver a reasonable quality voice service [21, 20, 94]. The MBone research community built on this existent packet audio technology by extending and enhancing it for large-scale multicast transmission. To motivate these developments and provide context for our later discussions, we now describe the history of the MBone and the evolution of the MBone tools from prototype applications into Internet standards.

Figure 1 depicts a timeline of the MBone, which originated from a collaborative project that was conceived to carry out joint network research among a number of institutions including the University of Southern California's Information Sciences Institute (ISI), the Massachusetts Institute of Technology, the Xerox Palo Alto Research Center, the Lawrence Berkeley National Laboratory, and a number of other sites. The project was sponsored by the Defense Advanced Research Projects Agency (DARPA) and a wide-area research network called the "DARPA Research Testbed Network" (DARTNet) was created to interconnect the sites and serve as the principal research testbed. By 1990, the infrastructure, composed of Unix workstations serving as programmable routers and interconnected via T1 links, was in place.

Early in the project, the research community deployed a preliminary version of IP Multicast over DARTNet, which provided the first opportunity to study and experiment with wide-area network-layer multicast on a non-trivial scale. In support of one of its key research charters, the DARTNet community developed a number of real-time, interactive multimedia applications that exploited IP Multicast to study the problem of multiparty remote-conferencing over packet-switched networks. Building on their pioneering packet audio work from the seventies [21] and on earlier work at Bolt, Beranek, and Newman Inc., researchers at ISI crafted an audio conferencing application called the Voice Terminal, or *vt*. In February of 1991, the first packet audio conference was held over DARTNet using the Network Voice Protocol (NVP) [20] implemented within *vt*. By June, weekly research meetings among the DARTNet participants were held using NVP audio and the DARTNet multicast infrastructure.

Because *vt* was a pioneering research prototype that focused principally on network research issues, little effort went into developing its user interface. Consequently, *vt's* text-based interface was cumbersome and provided limited user control and feedback. To improve upon *vt*, we elaborated ISI's work with new algorithms for counteracting network packet jitter through receiver buffering and "playback point" estimation [55] and enhanced the tool with a graphical user interface. Our new application became the LBL Visual Audio Tool, *vat*, and the DARTNet community gradually began using our new tool upon its release in the fall of 1991[2].

The success and utility of the weekly DARTNet meetings generated interest in extending the multicast infrastructure beyond the reaches of the testbed and into other segments of the Internet. However, at that time, production Internet routers could not carry multicast traffic. In anticipation of this problem, the IP Multicast designers enabled multicast routers not only to communicate with other routers over physical links, but also to forward packets and exchange routing messages over virtual links using IP-in-IP encapsulation[3]. In effect, a multicast router at the edge of a multicast-capable subnetwork *tunnels* through the non-multicast capable portion of the Internet to another router on an otherwise disjoint multicast-capable subnetwork.

[2] To ease the transition, *vat* supported NVP for backward compatibility and for some time, conferences were held using a mixture of *vt* and *vat*.

[3] IP source-routing was used originally as the mechanism for tunneling through non-multicast capable networks.

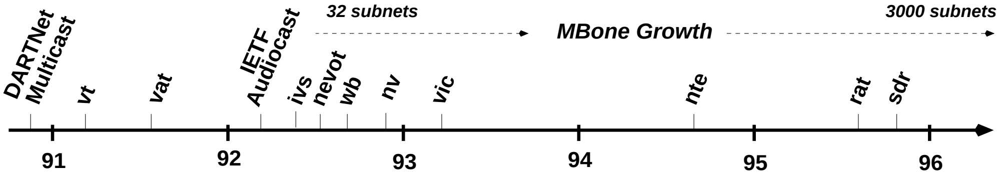

Figure 1: **MBone Timeline.**

The ability to bridge together multicast subnets using tunnels prompted an ambitious experiment in March of 1992. Thirty-two isolated multicast sites spread over four countries were configured into a large virtual multicast network, which in turn was used to *audiocast* the 23rd Internet Engineering Task Force (IETF) meeting. The virtual network backbone used to glue together the multicast-capable subnetworks was dubbed the "Multicast Backbone", or MBone [13]. For the first time, IETF participants were able to attend working group meetings and participate in the conference from a distance. Despite a few technical difficulties, the experiment proved enormously successful.

This success prompted continued work on remote collaboration tools like *vat* and many new pieces came together in 1992. INRIA released their video conferencing tool, *ivs*, an integrated audio/video conferencing system that relies exclusively on H.261 [1] for video compression. In the fall of 1992, Schulzrinne released an MBone audio tool similar to *vat* called *nevot* [95]. This tool was the principal development vehicle for the earliest versions of the Real-time Transport Protocol (RTP) [97], now the IETF standard for Internet packet audio. RTP and the *vat* audio protocol were eventually combined, culminating in a revamped version of the RTP specification in 1995.

In the spring of 1992, we begun work on a shared whiteboard application, *wb*, that augments audio and video channels with a medium for shared graphical annotation and presentation of on-line documents. Unlike audio and video streams, which can continue in the presence of packet loss through momentary degradation in quality, *wb* requires reliable transport since drawing state is persistent and a lost drawing update should eventually be retransmitted and delivered to all of the affected participants. But the design of a protocol that meets this requirement is a fundamentally difficult problem because the loss recovery algorithms must scale to very large numbers of receivers in an environment like the Internet where network partitions are common and packet loss rates are highly variable. In *wb*, we prototyped a new approach to reliable multicast based on transport semantics that are much more relaxed than those of traditional reliable multicast protocols. Our approach called *Scalable Reliable Multicast (SRM)* is detailed in §5.

In December 1992, Frederick released the Xerox PARC "Network Video" tool, *nv*, a "video only" application that utilizes a custom coding scheme tailored specifically for the Internet and targeted for efficient software implementation [40]. *Nv* quickly became the de facto standard for MBone video. About the same time, Jacobson created the Session Directory tool, *sd* [53]. A user creates and advertises a conference or "session" with *sd*. In turn, each participant uses *sd* to automatically launch all the media tools pertaining to that session, freeing the user from burdensome configuration of multicast addresses, ports, and scopes. This work was refined by Handley and Jacobson into the Session Description Protocol (SDP) [47] and Handley developed a much improved SDP-based session directory tool called *sdr* [44] first released in late 1995.

In winter 1993, we started work on the UCB/LBL video conferencing application, *vic*, which we conceived to simultaneously demonstrate the Tenet real-time networking protocols [36] and support the evolving LWS architecture. Work on *vic* has since driven the evolution of RTP. As RTP evolved, we tracked and implemented protocol changes, and fed back implementation experience to the design process. Moreover, our experience implementing the RTP payload specification for H.261 led to an improved scheme based on macroblock-level fragmentation, which resulted in a revised protocol [104]. Finally, the RTP payload specification for JPEG [11] evolved from a *vic* implementation.

More recent additions to the MBone application suite include the Network Text Editor, *nte* [46], and the Robust Audio Tool, *rat* [48], both developed by researchers at the University College London. Like *wb*, *nte* requires reliable delivery and adopts the LWS design philosophy, but it uses alternative protocol machinery for scalable loss recovery. *Rat* is an audio conferencing application that uses a novel forward error correction scheme where redundant information is coded at lower quality. Hence, modest packet loss rates cause minor quality degradation rather than more noticeable audio break ups.

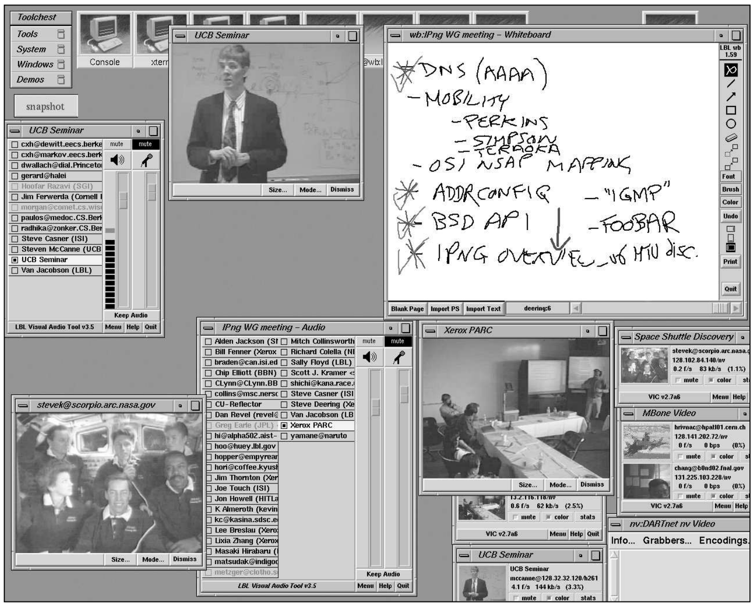

Figure 2: **The MBone Tools.**

Throughout the course of these developments, the MBone has steadily grown — from 24 subnets in the initial IETF audiocast to about 3000 subnets in mid-1996. And the content has evolved dramatically as well. No longer used exclusively for esoteric IETF meetings, the MBone has been a conduit for rock concerts, art exhibits, Presidential addresses, mainstream technical conferences, university seminars and remote teaching, dissertation defenses, NASA space shuttle broadcasts, live surgical demonstrations for medical meetings, and so forth. In just a few years the MBone has risen from a small, research curiosity to a large scale and widely used communications infrastructure.

Figure 2 shows a screen dump of a typical active day of MBone sessions from spring of 1995. The windows that contain lists of user names are each an instance of *vat*, while the lower right corner contains a number of *vic* application windows. Three *vic* viewing windows showing active video streams are opened on three different sessions and *wb* appears in the upper right corner. There are three active sessions: live coverage of NASA's space shuttle mission, a research meeting originating from Xerox PARC, and a seminar broadcast from U.C. Berkeley[4]. Close inspection of the video window labeled "Xerox PARC" shows a copy of *wb* running on a large in-room display device (a Xerox "LiveBoard"). Annotations to the LiveBoard appear on our local copy of *wb* as well as on everyone else's *wb*. Similarly, annotations made by any participant in the network appear everywhere, including the in-room LiveBoard.

2 IP Multicast

Having described the historical development of the MBone tools, we now detail the key elements of the Internet architecture and IP Multicast that make them especially suited to robust and scalable multipoint multimedia communication.

2.1 Best effort vs. Real-time

The traditional Internet service model is *best-effort*. The network does not guarantee that packets reach their destination (i.e., they can be dropped because of errors or buffer overflow) nor does it guarantee that packets are delivered in the order they were produced. Rather, routers simply attempt, without guarantee, to deliver packets toward their destination. Because these delivery semantics place few constraints on the design of the network layer, extremely robust systems can be built on this model. For example, if a link fails, the routing protocol inevitably takes time to compute new routes around the failure. During this time, packets may be reordered, lost in routing loops, or even duplicated. Because the service model does not preclude this behavior, an application or

[4] "Motion in Video Compression: What's it good for?", Michael Orchard, University of Illinois. See http://bmrc.berkeley.edu/courseware/cs298/spring95/ for more details.

transport protocol built on top of IP must be made robust to these events and, as a result, failures are gracefully tolerated.

A common criticism of best-effort networks like the Internet is that they cannot effectively support real-time traffic, and instead, the prevailing wisdom holds that new mechanisms for controlling "quality of service" are required (e.g., RSVP [113], ST-II [14], or Tenet [10]). While this argument holds for for underprovisioned networks, we initially conjectured[5] and found that effective real-time communication could be carried out over uncontrolled, best-effort networks through application adaptation, i.e., by adapting the application to the network rather than outright replacing the network with one that supports end-to-end quality of service guarantees. In this framework, the application adapts to observed variations in packet delays and mechanisms for assuring that the network has adequate capacity (e.g., through capacity planning or cost incentives) are factored out as orthogonal design elements.

We do not claim that real-time services should never by added to the network, but rather that the design of multimedia-networking systems should be decomposed as two orthogonal and complementary pieces: (1) adaptive, end-to-end media transport protocols and (2) mechanisms to enhance the performance of the underlying network to effectively support our real-time media transport when and where needed. Even with the luxury of real-time services, applications still must adapt to a degree (e.g., to effectively utilize a statistically guaranteed service class, to carry out clock-recovery, or to accommodate non-real-time operating systems in the end hosts) and they must be able to operate over a range of end-to-end delays to account for the inherent variability of propagation delay across the local and wide areas. By separating the design of application adaptation from the optimization of the underlying network service, work on adaptive end-to-end applications can proceed independently of work on real-time network services and great progress can be made in understanding the problems induced by multimedia networking before real-time network services are standardized and deployed at large. For the rest of this paper, we assume this separation and focus exclusively on the former component.

[5] Although the prevalence today of Web-based streaming audio has convincingly demonstrated the utility and viability of packet audio, when we embarked upon this work, the Web did not exist, multimedia hardware was relatively rare, and many network researchers believed that experimentation with real-time multimedia applications was not worth considering without companion support for real-time service guarantees in the underlying network.

2.2 Unicast to Multicast

The original Internet architecture provided only point-to-point *unicast* delivery, in which a source host transmits datagrams to exactly one destination host. But a large and growing collection of group-oriented applications require *multicast* delivery, in which a source host transmits packets simultaneously to an arbitrary number of destination hosts or "receivers". Although a multicast forwarding service can be implemented on top of unicast delivery by sending a copy of each datagram to every receiver, this is extremely inefficient because multiple copies of each packet will traverse the same underlying links. Instead, a much more efficient technique is to replicate packets only at fan-out points in the network so that at most one copy of each packet appears on a given link. Deering proposed this approach in the *IP Multicast* service model, which extends the traditional IP unicast delivery semantics for efficient multipoint transmission [27, 29]. In IP Multicast, packets are forwarded along a *distribution tree*, rooted at the source of the data and extending to each receiver in the multicast group.

A key strength of IP Multicast is its *host group* concept, which provides a powerful "level of indirection" both to the network and to the end-system applications. In this group-oriented communication framework, senders need not know explicitly about receivers and receivers need not know about senders. Instead, a sender simply transmits packets to an IP *group address* and receivers inform the network (via the Internet Group Management Protocol or IGMP [35]) of their interest in receiving packets sent to that group. Moreover, the process by which receivers join and leave multicast groups is reasonably timely and efficient.

A number of multicast routing protocols compute spanning trees from an anchor point in the network (either the sending host or a "rendezvous point" or "core") to all of the receivers, e.g., Protocol Independent Multicast (PIM) [28], Distance Vector Multicast Routing Protocol (DVMRP) [108], or Core Based Trees (CBT) [9]. Because the anchor point (i.e., source address) uniquely identifies the spanning tree, multicast routers can determine the forwarding action for any given packet by consulting that packet's source address. That is, to forward a packet, a multicast router might index a routing table using the packet's source address, which yields a routing entry containing a set of outgoing links. The router then transmits a copy of the packet along each outgoing link. To avoid sending traffic where there are no interested receivers, a multicast router omits links that have no downstream receivers for a particular group (e.g., the router might keep a list of "pruned" groups for each outbound link). In short, the *source* address determines the routing decision and the *destination* address determines

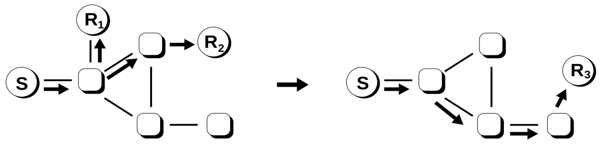

Figure 3: **Dynamic Group Membership.** IP Multicast provides a communication model where sources and receivers need not explicitly know about each other. Instead, hosts join a multicast group and the network delivers packets sent to that group to all interested receivers. The membership protocol is dynamic and timely. For example, if receivers R_1 and R_2 leave the group while receiver R_3 joins, the network will prune back the distribution path from R_1 and R_2 and expediently graft on the flow to R_3.

the prune decision. This contrasts with the unicast case where routing is (generally) determined exclusively by the destination address.

2.3 Group Maintenance

The performance of protocols built on top of IP multicast often depends on the multicast join/leave latencies of the group maintenance protocol. Once a receiver leaves a group, the network should suppress the flow in a timely manner to avoid the forwarding unnecessary traffic, which in fact, might be contributing to congestion. Fortunately, IGMP executes both "join" and "leave" operations on reasonable time scales. In this section, we describe how group membership information is exploited to prune traffic flows where not required according to the the "broadcast and prune" model used by DVMRP; other multicast routing protocols exploit different mechanisms to achieve the same ends.

When a receiver joins a new group, the host immediately informs the last-hop router, which in turn, immediately propagates a *graft* message up the multicast distribution tree if the group had been earlier pruned. If the group is new, the graft message is suppressed. When a receiver leaves a group, the situation is more complex because the last-hop router must determine when all the hosts on a subnet have left that group. To do this, when a host drops a group, it broadcasts a "leave group" message on the subnet and the router reacts by momentarily accelerating its normal membership query algorithm, which expediently determines whether any hosts subscribed to the group in question remain. When the last-hop router learns that no members remain, it sends a *prune* message up the distribution tree to suppress the group.

Figure 3 illustrates how the group membership protocol evolves over time for a simple scenario using a broadcast-and-prune multicast routing protocol like DVMRP. A source S transmits packets to some group. Initially, hosts R_1 and R_2 issue group membership requests to the network using IGMP. Each last hop router sees the IGMP message and propagates a graft message up the distribution tree toward S. In the time it takes to send the message up the tree, the paths are grafted on and data begins to flow down the tree as illustrated in the left half of the diagram.

At some later point in time, R_1 and R_2 leave the group again using IGMP. After a timely exchange of messages (usually no more than a few milliseconds), each leaf router determines that there are no downstream hosts subscribed to the group. In turn, they prune the group by not forwarding packets sent to that particular group on the pruned interface. Moreover, if a router receives a multicast packet addressed to a group that is pruned from all outgoing interfaces, it propagates a prune message further up the distribution tree. In this data-driven fashion, traffic is pruned back to the earliest point in the spanning tree.

At the same time the flows to R_1 and R_2 are pruned back, a third receiver R_3 may decide to receive traffic sent the group. Again, it uses IGMP to join the group and the new path is grafted on, leading to the configuration depicted in the right half of Figure 3.

This example illustrates a key mechanism that allows IP Multicast sessions to scale to arbitrary size: changes in membership are handled *locally* so that the maintenance of the distribution tree is distributed across the network and control messages do not concentrate or "implode" at a centralized coordinator. When a prune message arrives at a router for which the prune has already been propagated upstream, that control message is dropped, thereby attenuating the overall amount of traffic that flows toward the source. By coalescing control messages in this fashion, the control traffic load for a given group is fixed and independent of the group size [99].

2.4 Multicast Scope

While receiver interest *implicitly* constrains a multicast flow through the network's use of traffic pruning, multicast scope *explicitly* limits a stream to a topologically constrained set of receivers. For example, a university lecture that is of interest only to a local campus community should be broadcast only to that site rather than entire world. A source may exploit two classes of scope control to constrain its traffic:

- distance-based scope, and
- administrative scope.

In distance-based scope, the time-to-live (TTL) field in the IP packet header constrains the distribution of a packet. The TTL limits the number of forwarding hops that a packet can sustain before being dropped. The source host sets the TTL field to some initial value and each router along the path decrements the TTL by one. If the TTL ever reaches zero, the packet is discarded. (This mechanism was originally introduced in IP to prevent packets from circulating forever in a routing loop.)

Distance-based scopes are implemented by assigning a *threshold* to each multicast link. If a multicast packet's TTL is less than the threshold, then the packet is dropped. Through careful choice of thresholds, hierarchical scope regions can be defined. In the MBone, the accepted convention establishes three universal scopes: the "local site" (threshold 16), the "local region" (64), and the "world" (128). A fourth, less often used scope is the "low-bandwidth world" (> 128). Each site configures its multicast topology so that all external links have a threshold of 16, thereby constraining all packets with TTL less than 16 to the local site. Because regional and worldwide traffic originates with a TTL greater than 16, it is forwarded across the local site's boundary.

Since TTL threshold checks are based on simple numerical inequalities, a distance-scope can only become more restricted as a packet travels farther from the source. Hence, distance-based scope regions can only be arranged in a containment hierarchy, where the intersection of any two scope regions must be exactly the smaller region. For example, an individual subnet cannot simultaneously be in an "LBL-scope" and a "UCB-scope" unless the LBL-scope is contained entirely within the UCB-scope, or vice versa.

To address this problem, Deering originally proposed an administrative scoping scheme that provides overlapped regions by associating the scope region with a multicast address rather than a TTL [29, §3.2], and Jacobson and Deering presented specific mechanisms for incrementally realizing this scheme [56]. Here administrative scopes are imposed by configuring administrative boundaries at the borders between organizations. A special block of multicast addresses is reserved for administrative scope and since administratively scoped traffic does not flow across boundaries, scoped addresses need not be unique across organizational boundaries.

Unlike distance-based scopes that simply shed traffic at scope boundaries, administrative scope actively complements the normal multicast routing machinery. At an administrative boundary, unwanted traffic is pruned back up the distribution tree to prevent packets from unnecessarily flowing all the way to the edge of the boundary. In contrast, a router at the edge of a distance-based scope boundary cannot prune a flow because the TTL field in the underlying packet stream can vary dynamically and its value is a function of the source not of the destination group.

3 Light-weight Sessions

Building on IP Multicast and its scalable group-management protocol, we now elaborate the light-weight sessions model introduced earlier. LWS is decentralized and free from the burden of explicit peer-to-peer signaling or connection setup. This is in stark contrast to some traditional multimedia communication systems that were heavily influenced by the telecommunication-oriented, point-to-point model of communication. We briefly describe one such system architecture — the International Telecommunication Union (ITU) H.320 audio/video conferencing system — to provide a counterpoint to the LWS design philosophy. H.320 defines compression formats, bit-stream syntax, and channel-coding algorithms for multimedia communication over point-to-point ISDN lines [2]. While well suited to small-scale conferencing like point-to-point "video phone calls" or small multipoint meetings, the H.320 architecture does not scale gracefully beyond a few tens of sites.

The natural extension of H.320 for multipoint communication is simply to enhance the point-to-point protocols with multipoint negotiation and to add new services within the network that multiplex and filter streams. Such services have, in fact, been defined by the ITU (e.g., H.243 [3]) and the agents within the network that implement these services are called Multipoint Control Units (MCU). In this MCU-based communication architecture, small conferences are typically deployed by manually configuring a single MCU in a star-shaped topology. To scale beyond small groups, MCUs can be cascaded into a hierarchy arranged in a tree, but the scalability of this approach is limited because the terminals and MCUs are relatively tightly coupled. For example, whenever a new terminal joins a conference, its terminal identifier is broadcast to each attached port and in turn propagated to all of the MCUs in the conference. A master MCU maintains a database of all of the terminal IDs, which thereby imposes a practical limitation on the size of the conference. And because the MCU design is an enhancement of the basic service rather than the central building block of the architecture, MCU configuration, maintenance, and placement is carried out explicitly as a high-level task rather than automatically as a side effect of the underlying network service.

The LWS model, in contrast, turns the ITU model upside down — multicast service is the primitive building block from which we build higher-level protocols. Senders and receivers communicate with each other with simple, light-weight, announce/listen protocols running over some agreed upon set of multicast addresses. The level of indirection supported by multicast groups effectively decouples participants in a session so that, unlike the ITU conferencing standards, there is no explicit call-setup phase that requires unscalable peer-to-peer negotiation. Moreover, there is no negotiation of "capabilities" as in ITU, and instead, media formats are defined and advertised in the session setup protocol.

In LWS, group management is implicit. Unlike many traditional conference management schemes, there is no protocol for controlling access to the group or for maintaining a consistent group view at each receiver. Privacy, for instance, is achieved not through access control, but instead through end-to-end encryption [58]. Similarly, conference control mechanisms are effected through the "receiver orchestration" techniques discussed in [69]. This style of conference management is often characterized as *loosely-coupled* control because explicit synchronization among group members is unnecessary.

3.1 The Real-time Transport Protocol

While LWS provides a framework for group communication, it does not specifically define the details of how session members intercommunicate. Richer application semantics must be built on top of the basic "send a packet" interface of IP Multicast. To this end, within LWS and on top of the IP service interface there must exist specific transport protocols and data formats to facilitate communication as well as methods for mapping these data formats onto multicast transmission channels.

Through our work on the MBone tools [69, 57] and other similar efforts [101, 39, 94, 95], the following communication model emerged. An LWS *session* is a collection of end-hosts that communicate using a particular, somehow agreed upon, set of IP Multicast group addresses. This session is further comprised of a number of media and each media type is allocated to two transport channels — one for data and one for control — each using the same multicast group. Unlike traditional audio/video conferencing systems like H.320 and MPEG that multiplex different media by interleaving their bit streams, LWS streams are transmitted independently; they are multiplexed only in the sense that they share the underlying network (excepting legacy and ITU-defined codecs that are encapsulated, for backward compatibility, by an RTP-based format). Traditional transport protocols are typically implemented within the operating system kernel and entail "hard" connection state, whereas the LWS model for transport is relatively "thin" and is implemented within the application. This approach is embodied in the Real-time Transport Protocol [97], recently standardized by the Audio/Video Transport Working group of the IETF.

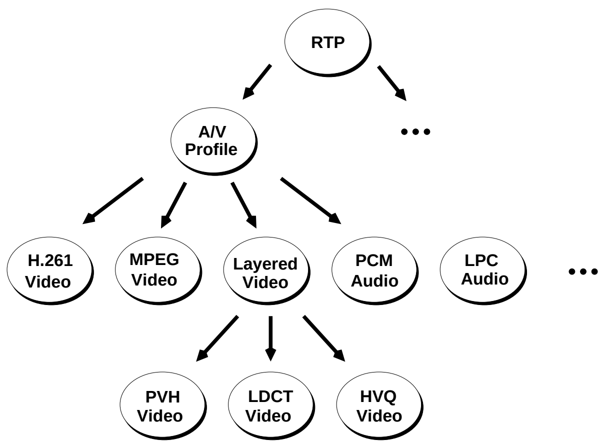

Figure 4: **RTP Protocol Architecture.** The Real-time Transport Protocol is composed of a number of interdependent specifications arranged in a "class hierarchy" to support the Application Level Framing architecture.

3.2 Application Level Framing

The RTP protocol builds heavily on the notion of Application Level Framing (ALF). In 1990, Clark and Tennenhouse proposed that an application's semantics should be reflected in the design of its network protocol to better optimize that application across the network and into the end-system [19]. About the same time that Clark and Tennenhouse proposed ALF, the work on the MBone tools was just getting under way. After several design iterations over transport protocols for several different audio/video compression formats, it became clear that a "one size fits all" protocol was inadequate [38, 69]. Instead, a framework based on ALF emerged where a "thin" base protocol defines the core mechanisms and profile extensions define application-specific semantics.

A common misconception surrounding ALF is that each time an application is built, the programmer must design and implement that application's protocol entirely from scratch. RTP, however, is living proof that this need not be the case. The solution is to cast the protocol not as a layer in the traditional layered protocol architectures like OSI [114], but rather as a tree-based hierarchy of interdependent protocols as depicted in Figure 4. We can view RTP as composed of a number of sub-component specifications roughly arranged in a "class hierarchy" to support the Application Level Framing architecture without sacrificing specification and implementation "re-use". At the base of the hierarchy lies the core RTP specification [97] while the Audio/Video Profile [96] specializes the protocol for LWS-style audio/video applications with "minimal control". The leaves of the hierarchy represent "RTP Payload Format Specifications" and define detailed application-specific parameters like coding syntax. For instance, payload format specifications define how H.261, Motion-JPEG, and MPEG bit streams are framed in RTP packet streams. Finally, a payload format may be further decomposed into a hierarchy, e.g., as indicated in layered video formats like PVH [73], LDCT [6], and HVQ [107, 16]. (The refinement of these experimental layered video formats into Internet standards is an active area of research and is currently under development.)

A key goal in RTP is to provide a very thin transport layer without overly restricting the application designer. The protocol specification itself states that "RTP is intended to be malleable to provide the information required by a particular application and will often be integrated into the application processing rather than being implemented as a separate layer." In accordance with ALF, the definitions of several fields in the RTP header are deferred to the A/V profile, which specifies their semantics according to the given application. For example, the RTP header contains a generic "marker" bit that in an audio packet indicates the start of a talk spurt but in a video packet indicates the end of a frame. The interpretation of fields may be further refined by the payload format specifications. For example, an audio payload might define the RTP timestamp as a audio sample counter while the MPEG/RTP specification [50] defines it as the "Presentation Time Stamp" from the MPEG system specification.

3.3 LWS/RTP Details

RTP is but one component of the LWS architecture and the term "session" is often used in either context. To clarify, an "RTP session" is a sub-component of an "LWS session" and represents a collection of two or more end-systems exchanging a single media type and related control information over two distinct underlying transport channels. For UDP [88] over IP Multicast, these data and control channels share a common multicast address and are assigned separate UDP port numbers to distinguish them. An active source transmits its signal on the data channel by generating packets that conform to the payload format specification for the underlying compression format.

Simultaneous with data distribution, all of the end-systems in a session exchange information over the control channel using an announce/listen protocol. Periodically, each source generates a Real-time Transport Control Protocol or RTCP message and simply "announces" it to the group. These messages provide sender identification, data distribution monitoring and debugging, cross-media synchronization, and so forth. There is no synchronization phase or ordering imposed upon session members with respect to RTCP. In fact, each source generates successive announcements at randomly spaced intervals to avoid potential protocol synchronization effects [37]. To scale the protocol to large groups, the mean interval between session reports is dynamically sized so that the aggregate bit rate used by all control traffic is limited to some low, fixed rate, typically a small percentage of the session's data bandwidth allocation (e.g., five percent is commonly used).

Note that this announce/listen style of protocol relies chiefly upon *soft state* [18]. That is, session information retained at each member site eventually expires but is continuously refreshed by the announcement process. If, for instance, a network partition occurs, the protocol continues to function normally. When the partition heals, the announce/listen protocol simply restores the state across the reformed session. No explicit fault recovery is necessary because it is built into the design. This greatly simplifies LWS applications in contrast to systems based on *hard state*, where error recovery often accounts for a significant portion of the implementation.

Each source in an RTP session is identified by a 32-bit Source-ID. Source-ID's are allocated randomly and conflicts are handled by a simple resolution algorithm. Random allocation was adopted for its simplicity: distributed assignment of unique identifiers across the session can be done quite easily without global synchronization and consistency rules. Since conflicts can cause Source-ID's to change at any time, a source's "canonical name" or CNAME provides a persistent and globally unique identifier. Data packets are identified only by their Source-ID but the RTCP control messages include the binding between CNAME and Source-ID. The CNAME is a variable length ASCII string.

Since RTP is independent of the underlying network technology, it simultaneously supports multiple network protocols. Figure 5 illustrates how RTP fits into several protocol stacks. For IP and IP Multicast, RTP is layered over UDP, while in the Tenet protocols, it runs over RMTP/RTIP [10]. Similarly, applications can run directly over an ATM Adaptation Layer. In all these cases, RTP is realized in the application itself.

3.4 Coping with Packet Jitter: Receiver Adaptation

The basic premise of packet audio/video is simple: digitize an analog signal into a serial bit stream, fragment the digitized bit stream into packets, transmit the packets

RTP		
UDP	RMTP	AAL5
IP	RTIP	ATM

Figure 5: **RTP and the Protocol Stack.** RTP is independent of the underlying transport protocol. This feature of RTP has allowed tools like *vic* and *vat* to be successfully deployed over a variety of network transports, including the Tenet Real-time Message Transport Protocol (RMTP) [10], the ATM Adaptation Layer (AAL5), and, the most commonly used configuration, UDP over IP and IP Multicast.

across the network to one or more receivers, reassemble the packets at each receiver to recover the original bit stream, and deliver the bits to the receiver's output audio or video codec. If the network faithfully delivers each packet with a fixed delay, then the received signal is reconstructed exactly. Since RTP data packets contain a media specific timestamp (e.g., a sample counter for audio and a frame clock for video) and RTCP control packets advertise the mapping between media time and the sender's real-time clock, the receiver can reconstruct the exact timing of the original media stream provided all of the packets arrive in time. But, because the network can induce arbitrary packet delays, the receiver's codec can potentially underflow before the next packet arrives thereby causing a glitch or stall in the reconstructed signal.

To prevent this, we can either engineer the network to carefully schedule packets at each router so that every flow sees a constant amount of delay (much as the phone network guarantees that voice traffic experiences a constant end-to-end delay), or we can simply induce an artificial buffering delay at the receiver to absorb and thus counteract network jitter. If we size the buffer large enough to absorb the maximum interarrival variance, then the buffer never underflows. Provided we avoid buffer underflow in this fashion and provided the packet loss rate is low enough, the delivered signal quality will be high. Since the packet loss rate is effectively controlled by ensuring adequate bandwidth in the network, we do not consider it here (i.e., in § 1, we argued that the design of a real-time service is orthogonal to the design of media transport and LWS), and instead, we focus on the buffer underflow problem.

How large should we make this buffer? If too small, the buffer is likely to underflow; if too large, the buffer never underflows and thereby provides high signal quality — but delay is increased. The optimal size depends

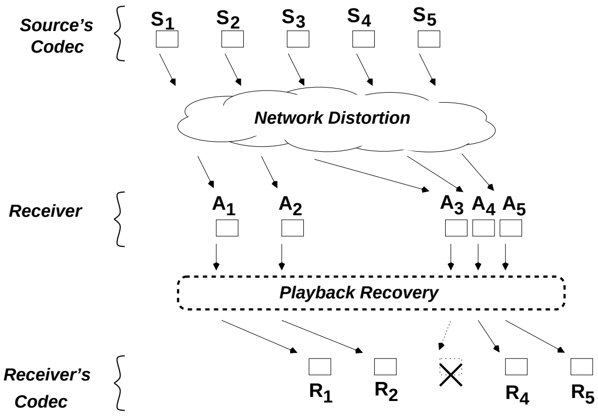

Figure 6: **RTP Delay Adaptation.**

on the amount of jitter induced by the network, which unfortunately, cannot be known a priori. To account for the dynamic and unpredictable nature of network jitter, RTP applications are typically *adaptive*. That is, an RTP receiver observes network performance and adaptively adjusts the degree of artificial buffering delay that it induces for each source's media stream. This process is often called *delay adaptation*.

The basic delay adaptation scheme or "playback point" algorithm is depicted in in Figure 6. The source generates a sequence of packets, each identified with a media-specific timestamp that indicates when the signal was transduced from the analog to digital domains. These "source" timestamps appear in the diagram as $S_1, S_2, \ldots$ Upon transmission across the network, the timing structure is distorted and the packets arrive at the receiver at skewed times, $A_1, A_2, \ldots$ The playback adaptation algorithm then schedules each arriving packet by computing a deadline from the source timestamp and an offset that accounts (dynamically) for the measured network jitter.

As seen in the diagram, packets 3 and 4 are delayed substantially through the network. Because the delay adaptation algorithm has artificially delayed the entire stream to account for jitter, packet 4 can be successfully scheduled before its "playback time". Unfortunately, the algorithm can mis-predict and occasionally a packet can arrive after its allotted playback time. This condition is depicted in the diagram where packet 3 arrives after its deadline and must therefore be dropped. Although packets may occasionally arrive late, this condition is rare because the algorithm is adaptive and would otherwise counteract an increase in variation by dynamically inflating the playback delay. To counteract the occasional packet loss, the application might attempt *error concealment*, i.e., to conceal the error by exploiting human perceptual biases through signal processing based analysis and synthesis.

To ground this discussion with details of a specific playback point algorithm, we now describe the delay adaptation algorithm employed in *vat*. This formulation is based on Jacobson's description in [55].

We first define the interarrival variance of packet i, Δ_i, as follows:

$$\Delta_i \quad = \quad |A_i - S_i - (A_{i-1} - S_{i-1})|$$

where, as above, S_i is the sender's timestamp stored in each media packet and A_i is the time at which the packet arrived at the receiver, and i is an arbitrary index that we can assume begins at $-\infty$. Note that the sender's and receiver's clocks need not be synchronized since we can rewrite the above expression as

$$\Delta_i \quad = \quad |A_i - A_{i-1} - (S_i - S_{i-1})|$$

and observe that the reference time base at each of the sender and receiver is "canceled out".

Thus, the time series $\{\Delta_i\}$ comprises a set of samples of the packet interarrival times and we can estimate the mean delay variance by low-pass filtering this time series:

$$\begin{aligned}\hat{\Delta}_i &= (1-g)\hat{\Delta}_{i-1} + g\Delta_i \\ &= \hat{\Delta}_{i-1} + g(\Delta_i - \hat{\Delta}_{i-1})\end{aligned}$$

where g controls the degree of smoothing and $\hat{\Delta}_i$ is the smoothed jitter estimator. Finally, we can use this estimate of the interarrival variance to adjust the amount of playback delay, which in turn minimizes the chance that packets arrive late while minimizing the playback delay (hence maximizing interactive performance). By the Central Limit Theorem, we can roughly guarantee that the number of late arrivals is negligible by choosing a playout delay that is some number of standard deviations larger than the mean, e.g., we set the playout delay:

$$\delta_i = \beta\hat{\Delta}_i$$

where β is the control constant. With this representation, we can explicitly control the tradeoff between the frequency of late packets and the buffering delay. By making β large, we decrease the frequency of late packets but we increase delay. Conversely, by making β small, we decrease delay but increase the occurrence of late packets.

For signals like video where fine-grained fluctuations are barely perceptible, adjusting the reconstruction delay on a per-packet basis works well. But for audio, such an approach introduces phase distortion that results in unacceptably low quality. Instead, the approach taken in *vat* is to adjust the playback point only during silent periods. To this end, we compute the operational $\hat{\delta}_i$ from the measured δ_i simply as:

$$\begin{aligned}\hat{\delta}_i &= \delta_i && \text{if packet } i \text{ starts a talk spurt} \\ \hat{\delta}_i &= \hat{\delta}_{i-1} && \text{otherwise}\end{aligned}$$

Finally, we compute the actual playback point for each packet in the stream $(R_1, R_2, \ldots)$ by adjusting the packet timestamp upward by the playback point. At this point, we also must convert the sender's timestamp from its time base to the receiver's, which is a trivial computation if we assume each site runs NTP [74]. However, this assumption is unrealistic in the current Internet so an application like *vat* uses a very simple heuristic: at the start of each talk spurt, we compute a clock offset, θ_S, relative to source S, by subtracting the timestamp in the first packet of the spurt from the current value of the receiver's local clock. Then, the playback point for each subsequent packet from source S is simply:

$$R_i = S_i + \theta_S + \hat{\delta}_i.$$

Building on the delay adaptation algorithm designed and implemented within *vat*, a number of more recent works have refined the concept and comprehensively explored the design space. For example, [78] derives an optimal performance bound on the playout delay for a target loss rate and presents a new playback point algorithm that approaches this bound.

Even if the receiver computes an effective playback point using the algorithm in *vat* or an improved variant, the actual delay through the playback buffer may be distorted by a mismatch in the sampling rates of the source and receiver(s). Typically, this rate differential is quite small, but over time, even a slight drift can lead to delay backlogs on the order of seconds. For example, if the source produces samples slightly faster than the receiver, then a backlog of samples builds up and thereby increases the end-to-end delay. Although sampling rate discrepancies are usually quite small and therefore not often problematic, one particularly popular PC-based audio codec cannot be programmed with exactly an 8kHz sampling rate — the most commonly used sampling rate for MBone audio — and the margin of error is thus quite large. A substantial delay backlog builds up in just a few minutes.

To solve this problem, the receiver must estimate the sampling rate mismatch and carry out sampling rate conversion. Since this is typically expensive to do accurately (especially when converting between two closely-spaced frequencies), simpler but less precise schemes are often used. In audio, for example, individual samples could simply be dropped or inserted every so often to effectively match some target rate. If the signal is low-pass filtered around the insertion or deletion points, the perceptual artifacts are typically minimal. Even better, if the application employs silence suppression, then samples can be imperceptibly dropped or inserted during silent periods; in practice, audio applications like *vat* simply resynchronize after each silent period and never explicitly insert or delete samples. To the best of our knowledge, none of the existing Internet audio applications actually implements the more complex rate adjustment techniques. Perhaps when continuous audio becomes more prevalent on the Internet (e.g., radio and television broadcasts), we will start to see sampling rate conversion algorithms implemented in mainstream audio applications.

Some real-time media applications do not require interactive performance (e.g., a one-way seminar or "video on demand") and thus do not require careful computation of the jitter estimator. But these applications can still benefit from the playback point model. Here, we can artificially set the playback delay to some large value independent of the interarrival variables. If large enough, say ten seconds, then for all practical purposes packets never arrive late. In web-based applications, this approach is commonly used and is called "streaming media" [90] and the *vat* application makes precisely this tradeoff in its so called "lecture mode".

In addition to counteracting network jitter, the delay adaptation algorithm can be exploited to provide cross-media synchronization by aligning each individual media stream with the stream that has the maximal playback point [55]. For example, if the video playback delay were 103ms and the audio playback delay were 63ms, then we could adjust the audio playback upward to 103ms and thus easily achieve audio/video synchronization. This approach has been successfully implemented and fielded in the UCL audio tool *rat* [61].

3.5 Coping with Packet Loss: Resilient Coding

While the delay adaptation algorithm can successfully accommodate variations in end-to-end packet delay, it does nothing to accommodate packet loss. When packet loss occurs, the signal potentially degrades in quality, but the degree of this degradation can be controlled through a number of techniques:

- *Automatic repeat request (ARQ)* retransmits packets when they are lost in the network. This approach maximizes signal quality because all packets are eventually delivered, but achieving this level of reliability incurs variable delays due to retransmissions, which confounds interactive performance.

- *Forward error correction (FEC)* enhances the packet transmission with redundancy so that the original signal can be recovered from a subset of the received packets.

- *Resilient source-coding* modifies the signal representation so that the media stream is less perceptually susceptible to packet loss, perhaps at the cost of compression efficiency.

While a great deal of research has focused on many variants and hybrids of these three core approaches [112, 48, 4, 52, 82, 91, 25, 102, 69, 40], the predominant scheme exploited in the MBone tools is that of resilient source coding because of its simplicity, low delay, and amenability to low-complexity implementation.

On the one hand, we can interpret resilient source-coding as a manifestation of ALF — the application and its network protocol work in tandem to cast its data into a representation that is well matched to the underlying network. On the other hand, we can view resilient source-coding as a manifestation of joint source/channel coding (JSCC), a well-developed concept from formal communications theory. JSCC states that one can often achieve better system performance by combining the design of compression and error-control coding rather than treating these sub-problems independently [23]. Garrett and Vetterli [42, 43] first applied this terminology to packet networks by viewing the packet transport mechanism as the channel-coding algorithm. In his thesis, Garrett argues that real-time traffic is better served by jointly designing the compression algorithm with the transmission system [41]. In a sense JSCC and ALF are two manifestations of the same underlying design principle.

3.5.1 TCP/H.261 Straw-man

While JSCC was established by theoretical work in the communications research community, the parallel notion of application level framing arose from engineering principles developed in the data networking community. To illustrate the reasoning that leads us to ALF-based design, we propose an anecdotal design of an Internet unicast video delivery system. Under the premise of modularity (and Shannon's "separation principle" [98]), such a system might be designed using TCP for the channel-coding algorithm and H.261 for the source-coding algorithm. Although each of these schemes has been independently optimized and performs extremely well in its targeted environment, they interact poorly together.

Because Internet transmission errors are manifested as burst erasures (i.e., packet losses), TCP would have to impose many packet delays at the source to compute effective error-correcting codes with packet granularity. Moreover, the efficiency of such a code depends strongly on the accuracy of the channel model known to the source (e.g., when packet loss rates are high, the data requires more protection, but when losses are low, the data requires little or no protection). And because residual capacity in the Internet is highly dynamic, it is difficult to estimate the channel model at the source. In part for these reasons, TCP uses ARQ instead of error-correcting codes to combat transmission errors and network buffer overflows, and includes a set of congestion control mechanisms that interact tightly with its ARQ algorithms [54]. Together these algorithms comprise a transport protocol — the analog of a channel-coding algorithm — that is highly optimized for packet-switched best-effort networks like the Internet.

Even though H.261 and TCP each perform well in their respective environments, their combination for Internet video transmission leads to poor end-to-end performance for the following reasons:

- The ARQ mechanisms in TCP use acknowledgments and retransmissions to guarantee delivery of data, which is unnecessary and detrimental to video streams because such retransmissions cause large variation in packet delivery times and this stalls the video playback process.

- Interactive video streams require timely delivery. Often it is advantageous to discard late data rather than wait for retransmission. In effect, video is loss-tolerant because we can simply suffer a momentary degradation in quality rather than wait for a retransmitted packet. Since TCP provides just one specific delivery semantics (i.e., a reliable byte stream), an application cannot instruct the transport layer to cancel a retransmission for a given piece of data. Instead, it must wait for the retransmission to complete before receiving subsequent data.

- When packets are lost or reordered in the network, they arrive at the receiver in an arbitrary order. But often performance is improved by processing data as soon as it arrives and then patching any errors after the fact when the retransmissions arrive. Because TCP provides only in-order delivery, packets are never delivered out of order and thus the application cannot immediately consume misordered data even when it is timely and useful.

- TCP provides a byte stream-oriented API that hides the underlying packet representation from the application. If the application could control how data was mapped onto packets, it could use intelligent framing to minimize the impact of loss by allowing the H.261 decoder to proceed in the presence of missing packets.

Fortunately, we can solve all of these problems with application level framing. ALF leads to a design where the application takes an active role in the encapsulation of its data into network packets, and hence, can optimize for loss recovery through intelligent fragmentation and framing. In this model, the data is tailored for the network by explicitly defining the Application Data Unit, or ADU, into the application/network interface, e.g., by applying intelligent fragmentation schemes to existing compression standards (e.g., JPEG, H.261, and MPEG) [49]. The approach taken in the MBone tools elaborates the original ALF concept further: not only should we optimize the network protocol for the application, but we should also optimize the application for the network. For example, [49] treats the compression algorithm as an immutable black box. Rather than adapt the source-coding algorithm to the network, it develops intelligent framing techniques for the fixed bit syntax produced by the black box. Conversely, several of the video algorithms developed for the MBone tools break open this sacred box and tailor its behavior for the network through modification or restriction of the underlying compression algorithm.

By structuring the RTP architecture according to ALF and JSCC, payload format specifications can be effectively tuned for and tailored to Internet transmission. In essence, the framework allows us to easily modify the source-coding algorithm to match the channel (i.e., JSCC) and likewise to modify the "application data" representation to match the underlying network (i.e., ALF).

3.5.2 RTP Video

One example of the ALF/JSCC design process is our formulation of the RTP-based Intra-H.261 packet video compression format [69]. Our goal was to design a video compression algorithm for RTP that could cope with the burst erasures of Internet packet loss. As with TCP, separation does not work well here because burst erasures require many packets in order to construct burst loss-resilient codes, thereby increasing end-to-end delay substantially. Furthermore, a burst of lost packets can occur across many consecutive packets (i.e., when a link becomes congested) and protecting against this requires error control codes that span even larger time scales. Moreover, the dynamic variability of the underlying channel statistics (i.e., level of congestion) causes inefficiencies in our choice of error correcting codes at the source.

Before we propose a compression format that is jointly designed for a packet erasure channel, we first examine how a specific source-coding algorithm, H.261, is sensitive to burst losses. The fundamental challenge in adapting H.261 for packet transmission is overcoming its assumption that the underlying channel is a continuous serial bit stream with isolated and independent bit errors. As a result, the source-coding algorithm assumes that all bits reach the decoder intact with very high probability and thus that recovery procedures for missing data are rarely needed. When bits are lost or corrupted, the decoder must resynchronize its state along three dimensions:

- **Entropy-code Synchronization.** The coded symbol stream consists of entropy-coded variable length codewords. If the bitstream is fragmented within a codeword, the decoder cannot correctly interpret the remaining codewords. Furthermore, the syntax is context dependent — that is, the entropy codes at the current point depend strongly on how the bit stream was previously parsed. For example, the decoder uses one Huffman code to parse DCT coefficients and a different Huffman code to parse macroblock types.

- **Spatial Synchronization.** Even if the codec can resynchronize its entropy-decoding stage after a transmission error, spatially predicted state may remain corrupt because the predictor cannot subsume the lost data. For example, block addresses are differentially encoded. Hence, a loss of just one block-address update will corrupt the block address until it is reset at a synchronization point thereby causing image blocks to be rendered into the wrong location within the frame.

- **Temporal Synchronization.** H.261's temporal prediction model assumes transmission errors are rare and therefore codes most block updates differentially (as they compress better than "intra-mode" updates). Consequently, lost packets result in lost differential updates, which cause reconstruction errors to persist in the decoder's prediction loop.

Because the H.261 source-coding algorithm assumes a reliable bit-oriented channel, the resynchronization mechanism is relatively simple. A frame is partitioned into a number of spatially disjoint units each called a "Group of Blocks", or GOB. Each GOB is prefixed by a special "start code" that is not a legal codeword and therefore cannot appear anywhere else in the bit stream. When the decoder loses synchronization, it scans for the next GOB start code to resume decoding (at the cost of discarding all intervening bits). Although a GOB boundary is a synchronization point for the entropy coder and spatial prediction state, it cannot resynchronize the temporal prediction state without loss of compression efficiency. Moreover, GOBs are coarse-grained objects within a frame and thus occur infrequently.

Nevertheless, we can provision the H.261 resynchronization algorithm for packet transmission by simply aligning the start of each packet on a GOB boundary. Because a GOB provides a well-defined starting point where the decoder state is synchronized to well-known initial values, we can decode a given GOB independent of other GOBs, and in particular, we can decode it independent of packet loss in other GOBs. This approach was taken in early versions of Turletti and Huitema's "RTP Payload Format Specification for H.261" [104], but it had several shortcomings:

(i) A GOB does not necessarily fit within a packet and often must be fragmented across multiple packets. The initial payload format specified mechanisms for fragmentation, but because these mechanisms did not explicitly account for the underlying bit stream semantics, a lost fragment could cause later packets to be undecodable and thus useless.

(ii) Not only might a GOB be too large for a packet, but the variability in GOB sizes can cause an inefficient packing of GOBs into packets. That is, some packets may be full while others are almost empty thereby increasing the number of packets needed to encode a frame (consequently increasing the per-packet overhead).

(iii) Under packet loss, the coding scheme suffered from artifacts caused by mismatched temporal prediction state.

To overcome these limitations, we adopted a JSCC/ALF approach and proposed modifications to the source-coding algorithm to better match the packet transmission model. From the source-coding perspective, our changes introduce additional overhead that increases the bit rate for a given level of quality, but the advantages they provide in robustness to packet loss outweigh this overhead. In the next two sections, we describe a macroblock-based fragmentation scheme that overcomes the limitations of GOB-oriented processing and a block-based conditional replenishment algorithm that overcomes the limitation of the temporal prediction model.

3.5.3 Macroblock-based Packet Fragmentation

To solve the spatial synchronization problem, we flatten the coding hierarchy. Rather than decompose a frame into GOBs, which in turn are decomposed into macroblocks, we adopt a single level decomposition where a frame is composed of macroblocks[6]. Further, we compactly represent the spatial prediction in a header at the

[6]The macroblock-based fragmentation strategy arose from a series of email exchanges with Atanu Ghosh and Mark Handley in the summer of 1994.

start of each packet, exploiting the fact that a packet arrival implies that all of the bits of that packet have arrived completely intact.

Because macroblocks are small compared to packets, we have fine-grained control over formatting them into packets and thus can map ADUs explicitly onto network packets by ending a packet on an arbitrary block boundary. Unfortunately, a source does not in general know the maximum packet size supported by the underlying networks. If the source uses packets that are too large, then the IP layer will fragment the datagrams into smaller packets and confound our one-to-one mapping between an ADU and a packet. For unicast transmission, a source can use "Path MTU Discovery" [60, 77] to learn the maximum transmission unit allowed along a certain unicast path through the network. (Note that this is another example of joint source channel coding where we are using network properties to control the source-coder.) But for multicast transmission, we have no such path discovery mechanism and rely on a rule of thumb for choosing packet sizes: we choose a size that is small enough to be carried by all of the prevalent physical-layer network technologies and is simultaneously large enough to sufficiently amortize the cost of packet headers. In the MBone, we currently use packet sizes of roughly 1000 bytes.

To summarize, macroblock-based framing of compressed bits into packets performs well because:

- a packet-switched network already includes overhead to identify packet boundaries, so there is no need to include resynchronization codes in the source-coding syntax;

- a packet identifies an explicit boundary that facilitates decoder resynchronization by including full decoder state at the start of each packet (e.g., first block number, spatial DC predictor, motion vector predictor if in use, and so forth);

- the overhead of the decoder state is amortized by sufficiently large packet payloads; and,

- every received packet can be decoded even when other packets in the stream are lost.

3.5.4 Conditional Replenishment

To solve the temporal synchronization problem, we adopt a simple temporal compression model called *conditional replenishment* [80]. In block-based conditional replenishment, the input image is gridded into small blocks (e.g., 8x8 or 16x16 pixels) and only the blocks that change in each new frame are encoded and transmitted. To avoid dependencies between the new block and previously transmitted blocks, we intra-code each block update rather than code a residual prediction error.

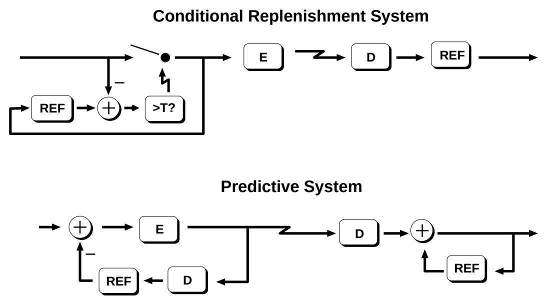

Figure 7: **Temporal Compression Models.** A conditional replenishment system encodes and transmits blocks as independent units, while a predictive system encodes and transmits the residual error between a prediction and the input signal.

Figure 7 depicts a block diagram for the conditional replenishment algorithm. The encoder maintains a reference frame of transmitted blocks. For each new block, a distance between the reference block and the new block is computed. If the distance is above a threshold, the block is encoded and transmitted across the network. At each receiver, the new block is decoded and placed in a reconstruction buffer for rendering and eventual display.

In contrast, compression algorithms like H.261 (or MPEG and H.263) employ temporal prediction to achieve higher compression performance. To first order, these schemes compute a difference between the current block and the previously transmitted block and code this "prediction error". If the block does not change much, then the difference signal has low energy and can be substantially compressed. Often, the encoder compensates for camera pan and scene motion by sending a "motion vector" with each block that accounts for a spatial displacement between the current block and the reference frame at the decoder (a copy of which is maintained at the encoder).

While the compression performance of motion-compensated prediction exceeds that of conditional replenishment in the absence of packet loss, there are a number of significant advantages of conditional replenishment:

- **Reduced Complexity.** Because the encoder decides very early in the coding process not to code a block, many of the input blocks are simply skipped, thereby saving computational resources. Moreover, because the encoder does not form a prediction signal, there is no need to run a (partial) copy of the decoder at the encoder.

- **Loss Resilience.** Coding block differences rather than the blocks themselves substantially amplifies the adverse effects of packet loss. When a loss occurs, the resulting error persists in the decoder's prediction loop until the coding process is reset with an "intra-mode" update. That is, the loss of a single differential update causes the error to propagate from frame to frame until the decoder resynchronizes. In H.261, for example, these updates can be very infrequent—as little as once every 132 frames. As a result, packet loss causes persistent corruption of the decoded image sequence. Alternatively, the use of "leaky prediction" lessens the impact of errors but incurs increased complexity and slower recovery [81, Ch. 5].

- **Decoupled Decoder State.** In the temporal prediction model, there is a tight coupling between the prediction state at the encoder and that at the decoder. But in a heterogeneous multicast environment, each decoder might receive a different level of quality and hence have a different reference state from which to construct the prediction. Since the "base layer" state is common across all receivers, the encoder can use it to perform the prediction. But in practice, the base layer provides inadequate conditional information to improve compression performance significantly across all of the layers. In contrast, conditional replenishment gives the advantage of temporal block suppression across all layers without relying on a matched decoder state.

- **Compute-scalable Decoding.** Heterogeneity exists not only in the network but also across end-systems, where some receivers might be outdated workstations while others are high-performance PCs. Consequently, in addition to packet loss in the network, messages can be lost in the end-system when the decoder cannot keep up with a high-rate incoming bit stream. In response, the decoder could gracefully trade off reconstruction quality to shed work [22, 34]. However, such adaptation is difficult under the temporal prediction model because the decoder must fully decode *all* differential updates to maintain a consistent prediction state. In contrast, with conditional replenishment, compute-scalability is both feasible and simple. The decoder simply collapses multiple frame updates by discarding all but the most recent compressed representation of each block.

  Moreover, conditional replenishment does not suffer from the well-known *decoder drift* effect. In predictive algorithms, the decoder's prediction state can gradually drift away from the encoder's because of numerical inconsistencies in the encoder and decoder implementations. (To limit the degree of decoder drift, compression specifications typically define the tolerances and the time extent between synchronization points.) On the other hand, conditional replenishment accommodates compute-scalable algorithms at both the decoder and encoder because there is no prediction loop to cause decoder drift. Here we can exploit numerical approximations to trade off reconstruction quality for run-time performance [73]. For example, the inverse DCT could be replaced by an approximate algorithm that runs faster at the expense of decreased accuracy [59]. Likewise, the degree of quantization applied to the DCT coefficients can be dynamically manipulated to meet a computation budget [63].

- **Self-correlated Updates.** The update heuristic that transmits only blocks that change works well in practice because block updates are "self-correlated". If a certain block is transmitted because of motion in the scene, then that same block will likely be transmitted again in the next frame because of the spatial locality of motion. Thus a block update that is lost in a dropped packet is often soon thereafter retransmitted and recovered as part of the natural replenishment process.

- **Static-background Video.** Finally, much of the video content currently sent over the Internet consists of tele-seminars and video conferences where large static backgrounds often dominate the scene. In these cases, conditional replenishment is highly effective.

For these reasons, we sacrifice the compression advantage of temporal prediction for the simplicity and practical advantages of conditional replenishment. In short, our compression algorithm exploits temporal redundancy only through conditional replenishment.

Although we believe that conditional replenishment provides a good tradeoff between compression efficiency and loss resilience, it is an extreme point in the continuum between full intra-coding and very infrequent inter-coding of block updates. If there is no packet loss in the network, then it is advantageous to use temporally predictive coding to improve compression efficiency. But under high loss, intra-only coding achieves higher overall performance. Thus, a scheme that monitors the underlying network performance and adjusts its coding algorithm reactively (i.e., JSCC) will likely perform better. Turletti and Huitema proposed such a loss-adaptive source-coding technique where the interval of intra-mode block updates (i.e., the amount of rate allocated to redundancy) is controlled by observations of the network [105]. When packet loss rates are low, then the coder uses a large intra-mode interval; when loss rates are high, a small interval is used. At the extreme, intra-only coding is used. Although this style of adaptation works well for unicast transmission, the update interval is fixed at the source and cannot accommodate multiple receivers with heterogeneous packet loss rates.

3.6 Session Rendezvous

The LWS and RTP frameworks as discussed thus far implicitly rely upon established assignments between multicast communication channels and the particular multicast group addresses that carry these channels. Essentially, a process for discovering these bindings is equivalent to the "session rendezvous" problem — that is, the task of each session participant locating all others in a distributed group. Once we know the group address bindings, we have found the other members in the group because we can communicate with them simply by sending multicast packets addressed to the group. But, how exactly are these bindings established? All of the work in defining scalable LWS protocols is futile unless we can create and manage these bindings in a scalable and robust fashion.

One obvious solution to this problem is to simply configure a centralized database within which clients can create or query "advertised sessions". However, such an approach has limited scalability and fault-tolerance. In a large network with many sessions, the load placed on the centralized database would quickly overwhelm the service, and further, centralization leads to a single point of failure, which adversely impacts server availability.

Rather than a centralized system, we could instead use a distributed database like the Domain Name System (DNS) [76]. The DNS's hierarchical and distributed caching strategy leads to high scalability, while its replicated server model leads to fault tolerance and high availability. However, the DNS was designed for relatively long-lived data records like Internet name to address mappings and mail-exchange pointers. DNS caches typically maintain state for hours, if not days, leading to potential inconsistencies that, while tolerated for host name resolution and the like, would be inappropriate for session creation and advertisement especially if frequent, short-lived sessions become commonplace.

In the absence of an existing directory service that met the needs of the LWS framework, a new "session directory" service was developed for the MBone. As with many of the other LWS protocols, the protocols underlying this session directory service utilize soft-state design principles and the light-weight announce/listen metaphor. The basic premise is to bootstrap all sessions from a single, well-known multicast group, or set of groups, which together serve as the conduit for the distributed directory service. To this end, each site in the global multicast network administers a session directory *agent* (analogous to a DNS server) that periodically announces each session known to and owned by that agent (on the well-known address for the directory service). Simultaneously, all agents monitor these announcements and build an information base of the existing sessions. Each session announcement lists generic information about the session (e.g., its name, scope, contact information, related links to web pages, and so on) as well as the actual mappings from media types to multicast addresses. And each session advertisement is created with a finite lifetime, i.e., a session description record is "soft state" that eventually times out without requiring explicit tear down. The Session Description Protocol (SDP) [47] defines the specific formats of session announcements while the Session Announcement Protocol (SAP) [45] defines the announce/listen protocol for disseminating these SDP messages.

If each directory agent advertised all of its known sessions to the entire world, the service would not scale. To overcome this, the session announcement protocols exploit the IP Multicast scope mechanisms described in §2.4. Each session announcement is multicast with a scope equal to the session's advertised scope since there is no point multicasting a session announcement where it will not be used, and further, where it should not be seen. This containment of session announcements greatly increases the scalability of the directory service since the "load" of session advertisements is partitioned throughout the network and only where needed.

While a scalable advertisement protocol is one critical component of the session directory service, another equally critical component is the address allocation algorithm. Given that sessions can be created at arbitrary times, at arbitrary locations in the network, and by arbitrary users, how can the system assign addresses to the newly created session while avoiding conflicts? Obviously, we could easily avoid conflicts with a centralized authority that allocates all addresses, but this would suffer the scaling problems noted above.

Instead, the LWS directory service currently uses a scheme called "informed, partitioned, random allocation" [55]. In this scheme, addresses are randomly allocated (the "random" part) within each scope (the "partitioned" part), but *known* conflicts are explicitly avoided (the "informed" part). In effect, the allocation scheme is intertwined with the announcement protocol to provide conditional information for collision avoidance. That is, the allocation algorithm uses the information base assembled by the announcement protocol. But because LWS provides only loosely-coupled operation with potential network partitions and asynchronous and concurrent events, not every conflict is avoided (e.g., two agents might simultaneously choose the same random address). However, by adequately sizing the address space, we can make the probability of collision vanishingly small. Unfortunately, the practical range of IPv4 multicast addresses is limited and this limitation is exacerbated by the "Birthday Problem" [32]. Complete analysis of these problems and their solutions is still open research and the overall address allocation architecture is undergoing revision in the IETF.

With this infrastructure in place, a session directory *client* can contact a local session directory *agent* and request a list of all known sessions. This client presents to the user a listing of known sessions along with accompanying session-specific information. When the user then joins an advertised session, the tool "launches" and configures all of the applications necessary to interact with each media channel listed in the advertised session. At present, the session directory agent and client are typically embedded in a single application (e.g., the session directory tool *sdr* [44]), but a modification of the protocols that factors this agent into a locally administered server is currently under development by the IETF. This interdependency is a historical remnant of the early evolution of the session protocols developed in the original Session Directory tool, *sd* [53].

4 Network Heterogeneity

In a world of uniform network technologies with uniform bandwidths, uniform capabilities, and uniform traffic loads, the LWS architecture, as described thus far, would offer an entirely adequate solution for large-scale multicast communication. However, this idealized uniformity simply does not exist. Because the Internet, by its core nature, interconnects disparate network technologies, it is inherently heterogeneous.

To illustrate the challenge presented by network heterogeneity in the context of multicast media, Figure 8 depicts the network topology that underlies the multimedia seminar broadcasts that U.C. Berkeley began transmitting in spring 1995. In this scenario, some users participate from their offices over the high-speed campus network, while other users interact over the Internet, and still others join in from home using low-rate dial-up or ISDN telephone lines. However, to maximize the quality delivered to the largest audience, Berkeley runs the transmission at a rate suitable for the MBone, which as a current rule of thumb, is 128 kb/s. But at this rate, home users cannot participate because the transmission exceeds their access bandwidth, and campus users must settle for unnecessarily low quality because

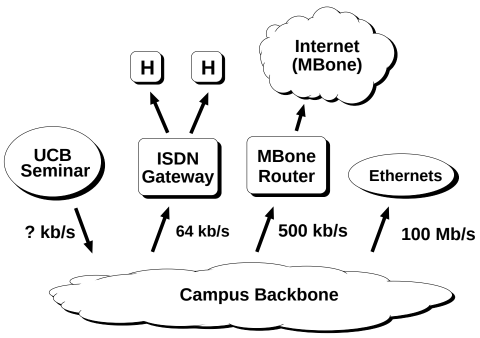

Figure 8: **U.C. Berkeley MBone Seminar.** U.C. Berkeley transmits a multimedia seminar over their campus network, to users at home via ISDN, and over the Internet. A single rate at the source cannot meet the conflicting bandwidth requirements of this heterogeneous set of users.

the low-rate video stream underutilizes the abundant local bandwidth. If we run the broadcast at a lower rate, then users behind ISDN lines would benefit but the Internet users would experience lower quality. Likewise, if we run the transmission at a very high rate, then local users would receive improved quality, but the MBone and ISDN users would receive greatly reduced quality due to the resulting congestion. A uniform transmission rate fails to accommodate the bandwidth heterogeneity of this diverse set of receivers.

Rather than adjust the transmission rate at the source, recent research efforts have proposed that adaptation be carried out either within the network, which we call "intra-network adaptation", or at the receivers, which we call "receiver-driven adaptation". In this way, network heterogeneity is "locally" accommodated by fine-tuning the transmission rate to exactly match the available capacity on each link in the network. In the subsequent two sections, we describe variations of these two approaches in more depth.

4.1 Intra-network Adaptation

In [86, 87], Pasquale et al. cite the shortcomings of closed-loop congestion control for multicast networks. Their Multimedia Multicast Channel (MMC) is designed specifically for heterogeneity by imposing only a loose coupling between the receivers and sources. In their architecture, receivers connect to a multicast channel through a port. By attaching a "filter" to the port, a receiver instructs the network to transform a flow to a lower rate. In turn, the network optimizes the delivery of heterogeneous flows by propagating the receiver-specified filters up the multicast distribution tree. At merge points in the tree, filters of the same type are combined and further propagated up the tree. While this architecture supports heterogeneous transmission, it does not provide a mechanism for end-systems to adapt dynamically to the available capacity in the network. Instead, MMC relies on admission control to explicitly allocate available bandwidth to heterogeneous receivers.

The *CU-SeeMe* system [31] from Cornell University distributes video over a manually configured multicast distribution mechanism. So called *reflectors* are manually placed throughout the network to effect a multicast distribution tree using native unicast transmission. Receivers generate loss reports that are sent back up the distribution tree via the reflectors. Hence, the reflectors can collapse loss reports to avoid the implosion problem. But they do so by averaging statistics across each reporting host. Because the sending rate is based on an average over different receivers, the overall transmission quality is proportional to the worst-case receiver. Moreover, the manual placement of reflectors works poorly in practice because end users are rarely capable of deploying the reflectors at strategic locations not only because of administrative access restrictions but also for lack of knowledge of the underlying topology. Consequently, "reflector networks" are most often deployed by creating a centralized server to which each user connects in a star topology. Typically, a user creates a "conference" by setting up a reflector and advertising its Internet host address. Each participant in turn connects to this advertised address, placing not only a computational load on the reflector host but also a traffic load on the network near the reflector. In this configuration, the server replicates each incoming packet to each of the outgoing connections causing quadratic growth of traffic.

Because RTP is an application-level protocol that makes few assumptions about the underlying transport and network layers, it is possible to bridge one or more distinct "RTP sessions" into a single logical session using application-level gateways. Turletti and Bolot [103] describe an architecture based on this model where *video gateways* are placed throughout the network to transcode portions of the multicast distribution tree into a lower bit-rate coding either through format conversion or requantization. This video gateway architecture was first fully developed and fielded by Amir et al. [7] in a specific application called *vgw*, which provided design feedback for the RTP specification and contributed an efficient software-based method for transcoding high-rate Motion-JPEG video into low-rate H.261. For example, the network heterogeneity underlying the UCB MBone seminar transmissions in Figure 8 is managed by broadcasting video at multiple rates using Amir's video gateway. As illustrated in Figure 9, gateways are deployed at strategic locations within the campus network to adapt a high-rate video transmission for the MBone and ISDN. A 1 Mb/s Motion-JPEG stream is transmitted over the campus network and two gateways are deployed. One gateway transcodes the high-rate stream to a 128 kb/s H.261 stream for the Internet, while the other produces an 80 kb/s stream for users over ISDN.

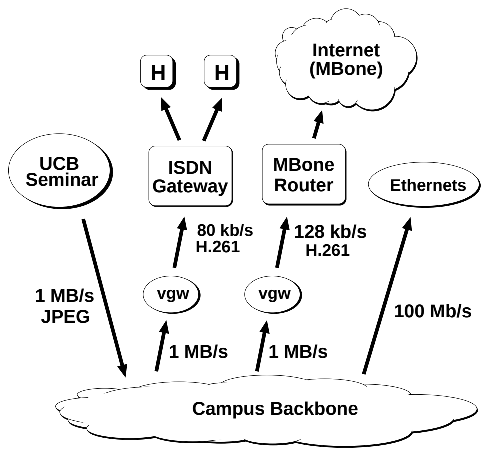

Figure 9: **Network-assisted Adaptation.** In network-assisted bandwidth adaptation, the rate of a video transmission is adjusted within the network. For the UCB Seminar transmissions, we transmit high-rate JPEG streams over the campus network and use multiple copies of our video gateway *vgw*, placed at strategic locations in the network, to transcode the high-rate video into low-rate H.261 for the MBone and for home users across ISDN lines.

A marked disadvantage of the video gateway architecture is that compute-intensive agents must somehow be placed, either manually or automatically, within the network infrastructure. This is a challenging task because the ideal location might be infeasible due to insufficient computational resources or it might be administratively impossible. One promising approach for overcoming these administrative barriers is the "active networks" initiative [100], which proposes a new network architecture that permits the placement and execution of user-specified computation inside the network. In this model, gateway programs can be positioned in the network by end-users at any point or points desired. At present, however, active networks remains very much an open research problem and a comprehensive solution in which video gateways exploit active networks has yet to be demonstrated.

Even if the gateway placement problem were solved, the architecture has other potential shortcomings. Because gateways must understand the semantic content of the packet streams they manipulate, they must be entrusted with encryption keys when acting upon private sessions. Yet this transfer of trust across administrative boundaries is unacceptable in many application contexts. Moreover, the scalability of the gateway architecture is constrained: a large multicast session with many heterogeneous receivers and diverse requirements would require the explicit management of gateways on behalf of all of these users, potentially leading to concentration of control actions. Despite these hard problems, work on application-level gateway architectures has shown promise and such systems have been used to support real user communities at non-trivial scale [5].

4.2 Receiver-driven Layered Multicast

To overcome the limitations of the gateway architecture, several researchers have proposed an end-to-end approach based on "receiver-driven adaptation" that moves the burden of rate-adaptation from the source or the network to the receivers [26, 70, 103, 110, 15, 30, 51, 71, 103, 17]. In this approach, the source signal is represented in a hierarchical format and its constituent layers are striped across distinct multicast groups. In turn, receivers adjust their reception rate by controlling the number of groups they receive. In other words, selective forwarding is implicit in receiver interest — if there are no receivers downstream of a given link in the network, the multicast routers "prune back" that portion of the distribution tree.

Although this general mechanism had been discussed for several years in the research community, a specific adaptation scheme was not presented until 1996, when we proposed Receiver-driven Layered Multicast or RLM [70]. To complement RLM, we developed a low-complexity layered source coder based on a hybrid wavelet/DCT transform. When combined with RLM, our "Progressive Video with Hybrid-transform" codec, or PVH, provides a comprehensive solution for scalable multicast video transmission in heterogeneous networks.

In this article, we provide only a brief survey of RLM; details can be found in [70, 73, 72]. In RLM, a session is comprised of a number of active sources transmitting layered streams to a number of receivers distributed throughout the network. Since bandwidth between a given receiver to any particular active source may vary, RLM treats each source independently and runs a separate instance of the adaptation protocol for each incoming stream. In effect, the source takes no active role in the protocol: it simply transmits each layer of its signal on a separate multicast group. The key protocol machinery is run at each receiver, where adaptation is carried out by joining and leaving groups. Conceptually, each receiver runs the following simple control

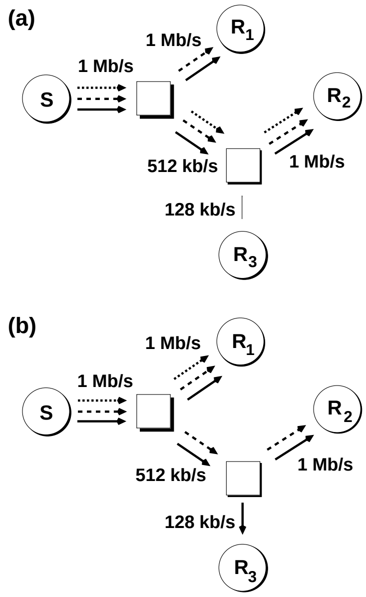

Figure 10: **Receiver-driven Adaptation.** Receivers join and leave multicast groups at will. The network forwards traffic only along paths that have downstream receivers. In this way, receivers define multicast distribution trees implicitly through their locally advertised interest. A three-layer signal is illustrated by the solid, dashed, and dotted arrows, traversing high-speed (1 Mb/s), medium-speed (512 kb/s), and low-speed (128 kb/s) links. In (a), we assume that the 512 kb/s is oversubscribed and congested. Receiver R_2 detects the congestion and reacts by dropping the dotted layer. Likewise, receiver R_3 eventually joins just the solid layer. These events lead to the configuration in (b).

loop:

- on congestion, drop a layer;
- on spare capacity, add a layer.

Under this scheme, a receiver searches for the optimal *level of subscription* much as a TCP source searches for the bottleneck transmission rate with the slow-start congestion avoidance algorithm [54]. The receiver adds layers until congestion occurs and backs off to an operating point below this bottleneck rate.

Figure 10 illustrates the fundamental mechanism underlying RLM. Suppose source S transmits three layers of video to receivers R_1, R_2, and R_3. Because the S/R_1 path has high capacity, R_1 can successfully subscribe to all three layers and receive the highest quality signal. However, if either R_2 or R_3 try to subscribe to the third layer, the 512 kb/s link becomes congested and packets are dropped. Both receivers react to this congestion by dropping layer 3, prompting the network to prune the unwanted layer from the 512 kb/s link. Finally, because of the limited capacity of the 128 kb/s link, R_3 drops down to just a single layer. In effect the distribution trees for each layer are implicitly defined as a side effect of receiver adaptation.

Although the RLM/PVH framework provides a promising foundation for scalable delivery of video in heterogeneous environments, a number of outstanding hurdles must be overcome to deploy the system at large scale over the current and future Internet:

- RLM requires layered media formats, but most existing codecs do not provide layered representations (and even if defined by the specification, they are often not implemented),
- the current multicast infrastructure is limited and not universally deployed (especially to end-users behind dial-up and ISDN lines), and
- RLM does not provide fine-grained and customized control over rate allocation in contrast to gateways.

In light of these concerns, we believe that heterogeneity in multicast transmissions will likely be accommodated through a combination of gateways and layered-transmission. In some ways, these architectures are complementary because a gateway can be simplified by leveraging layered representation, e.g., by simply filtering the appropriate number of sub-flows through the gateway.

5 Adding Reliability: Scalable Reliable Multicast

In the RTP applications discussed earlier, when a packet is dropped in the network, no effort is made to recover the lost information. Instead, the receiver simply suffers a momentary degradation in quality to sustain the lost packet. Since audio and video streams are more or less ephemeral and often contain perceptually redundant information, this scheme works well in practice when packet losses are light. However, many new applications like shared whiteboards, webcast tools, network games, and distributed simulation are not loss-tolerant at all. Whiteboard state, for example, is persistent. If a piece of a drawing update is lost, the application cannot leave the drawing in an incomplete state; instead, it must repair the damaged portion of the drawing by recovering the missing packets from the multicast session. In short, such applications require a *reliable multicast* transport protocol.

Unfortunately, reliable multicast is a difficult problem. Though a number of wide-area reliable multicast protocols have been proposed [65, 38, 64], none currently satisfies the requirements for "safe deployment" in the global Internet because no existing scheme simultaneously provides (1) scalability to very large multicast sessions, (2) congestion control that accommodates constrained and potentially heterogeneous link bandwidths, and (3) a rich and configurable range of reliability semantics. In fact there are many possible solutions with important variations and tradeoffs and the approach described in [38] proposes not a specific solution for reliable multicast, but rather a a *framework* that enables the construction of customized protocols tailored to specific applications in a reusable fashion. This work additionally proposes a set of protocol mechanisms, collectively called *Scalable Reliable Multicast* (SRM), to support reliable multicast for both very large sessions and networks.

Unlike many traditional works in reliable multicast, the approach in the IP Multicast community, and in particular SRM, does not attempt to guarantee any semantics for message delivery order. Protocols like ISIS [12], HORUS [106], Totem [79], and RMP [109] all support the notion of causal message ordering (i.e., all events triggered by message arrivals obey Lamport's notion of "happens before" [62]) as well as global message ordering (i.e., the same message order is seen by all receivers). But providing such semantics comes at a high cost: scalability is limited because each end-system must maintain state for all senders and system robustness is sacrificed because network failures or partitions require costly fault recovery algorithms. In contrast, SRM makes no such guarantees; messages can arrive in any order. The only guarantee is that each message is eventually delivered to all "interested" receivers. This model is often called "concurrent reliable multicast" [64] because message delivery is asynchronous and the order non-deterministic; thus in Lamport's terminology all events are "concurrent". And because the burden of ensuring reliability is placed on receivers in SRM, a receiver that is not interested in some data need not request retransmission. In this way, the protocol can accommodate heterogeneity in receiver reliability requirements.

Receiver-reliability and the framework-based approach are, in essence, another instance of ALF — different applications require different types of error recovery, flow control, and congestion adaptation schemes, and this variability is effectively managed by an ALF-based framework. Further, this ALF framework allows an application's protocol to explicitly account for the structure of the underlying application data through intelligent fragmentation, framing, packet reordering, and retransmission prioritization, thus optimizing that appli-

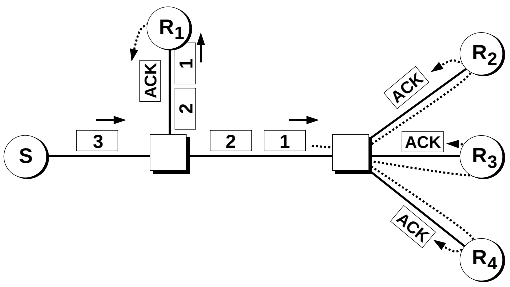

Figure 11: **Ack Implosion.**

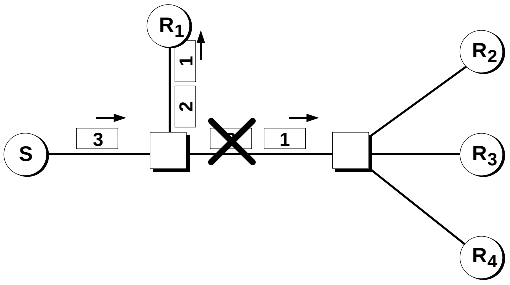

Figure 12: **NACK Implosion Trigger.**

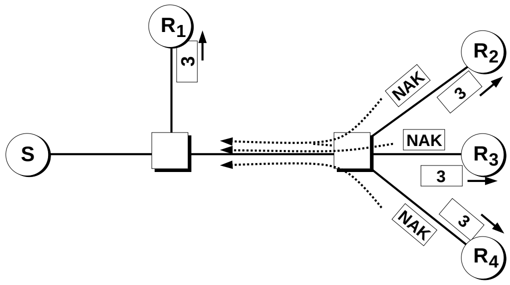

Figure 13: **NACK Implosion.**

cation across the network.

5.1 Loss Recovery

A key challenge in the design of a reliable multicast transport lies in defining a loss recovery algorithm that can scale to very large numbers of receivers. Why is this difficult? An obvious approach to providing reliability for multicast is to simply extend a unicast reliable transport like TCP to multicast. Here, TCP's ARQ algorithm generates an ACK packet for every data packet. However, in a multicast setting, if every receiver sent an ACK back to the source, then each transmission of a data packet would result in the roughly simultaneous generation of an ACK packet from every receiver in the multicast session. As depicted in Figure 11, these ACKs concentrate or *implode* at the source, resulting in congestion transients that adversely impact not only the reliable multicast protocol but other traffic flows as well. This pathological behavior is known as the ACK implosion effect [33].

We can reduce the impact of ACK implosions by using negative acknowledgments (NACKs) rather than positive ACKs. Here, a receiver generates a NACK only when it detects missing data. The receiver infers such a loss by noting a gap in the sequence space since

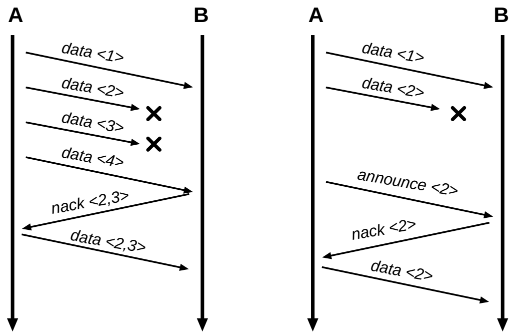

Figure 14: **Receiver-driven Loss Recovery.** This figure illustrates the two modes of receiver-driven loss recovery used by SRM: data-driven recovery and control-driven recovery. Each diagram is a time-space plot, where time proceeds down the page and the horizontal axis differentiates two hosts in space, A and B. In the left hand transfer, recovery is data-driven: A sends four packets, two of which are dropped. On receipt of the fourth packet, B detects missing data, requests a repair, and A repairs the two missing data units with a single packet. On the right hand side, recovery is control-driven using announce/listen "session packets". Here, the last packet in a sequence of packets from A is dropped, but B does not detect the data loss until A sends its periodic state announcement, at which time B issues the repair request. Note that SRM's random recovery delays have been elided to simplify the diagram.

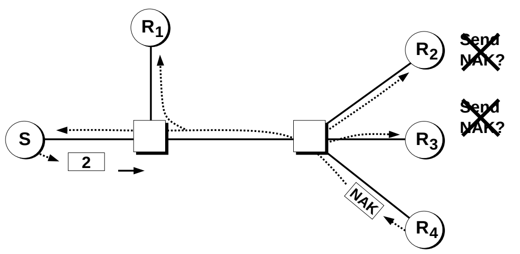

Figure 15: **NACK Suppression using Multicast.**

the source assigns a monotonically increasing sequence number to each packet in the stream. However, this gap detection algorithm cannot detect "tail losses," that is, losses that occur at the end of a packet burst. Thus, SRM includes a companion announce/listen protocol in which each announcement packet includes the highest sequence number sent so far. The announce/listen messages can then be used by a receiver to detect tail losses by comparing the announcement packet sequence number against the receiver's state. In effect, the receiver detects gaps either when a source sends data subsequent to a loss, or upon state announcement. Figure 14 illustrates these two recovery modes.

While the NACK mechanism works well when the location of the loss is near the receiver, the implosion effect still occurs for losses near the root of the multicast distribution tree. For example, Figure 12 shows a loss (packet 2) among a sequence of packets generated from source S. Subsequently, packet 3 arrives at roughly the same time at each of the receivers R_2, R_3, and R_4, in turn, triggering the simultaneous generation of NACKs back toward the source as depicted in Figure 13.

SRM avoids this problem by exploiting multicast to dampen control traffic in a distributed fashion. Rather than unicast a NACK to the original source, an SRM receiver multicasts the NACK to the entire group. As a result, a receiver in need of that same data can suppress its own generation of the identical control message. However, if all receivers generate a NACK at roughly the same time, the finite propagation delays between receivers will prevent them from learning of each other's attempts. Thus, SRM adds a random delay to the time each receiver generates a control message. In short, all receivers in a common "loss sub-tree" all detect the loss roughly simultaneously, they each pick a random delay, and some receiver "wins" the contest by picking the smallest random number. This receiver then multicasts its NACK and suppresses all other receivers' attempts at generating the same NACK. Finally, the source receives the NACK and as shown in Figure 15 retransmits the data for the missing packet 2.

Note that in this scenario both source S and receiver R_1 hear the NACK for packet 2 and both of them have that data. So either session member could potentially answer the NACK. This highly desirable consequence of the multicast-NACK scheme enhances SRM's fault tolerance because the system continues to operate even when the original source of the data has terminated or is behind a network partition. Moreover, this approach allows receivers nearer the location of packet loss to respond rather than the more distant source and thus decrease the delay and increase the efficiency of the protocol. To this end, any session member may respond to a NACK, but if all members respond to the NACK simultaneously, we again face the implosion problem. Hence, the data reply process in SRM uses the same randomized delay algorithm as the NACK generation process. Finally, to favor NACKs and responses from nearby sites, the randomized intervals are biased to account for the propagation delay between the sites.

Unfortunately, the randomized nature of the algorithm can still lead to overlapped events resulting in duplicate NACKs. To overcome this inefficiency, we can increase the range of the random delay distribution to reduce the probability of duplicate NACKs (i.e., because the spread of delays is larger leading to lower chance of overlap). While widening the delay distribution increases the protocol's efficiency, it also increases the overall expected delay. Thus we have a tension between timeliness and efficiency. If we are willing to incur extra delay we can reduce the control traffic load, but if we want lower delay, we must pay a higher cost in control traffic. SRM includes a mechanism for adapting these tradeoffs dynamically; measurements of the frequency of duplicate responses are used to adjust the randomized delay interval. The performance tradeoffs associated with adjusting the timer distributions are discussed in the original SRM paper [38] as well as in more recent work [89, 83].

To formalize the SRM protocol mechanisms, let C_1 and C_2 be design constants and $d(x)$ be the propagation delay from the receiver in question to host x, then the SRM loss recovery algorithm is roughly as follows:

- set the backoff parameter $\beta = 1$.
- upon missing data D from host S, choose a random delay τ on $2^\beta[C_1 d(S), (C_1 + C_2)d(S)]$
- schedule a request packet, REQ_D, for transmission in τ seconds.
- if we receive REQ_D from some other host before τ seconds, then set $\beta = \beta + 1$ and re-start the request timer.

- otherwise, if the data, D, or the repair reply, REP_D, is received before τ seconds, cancel REQ_D.

- otherwise, send REQ_D after τ seconds.

Note that the basic mechanism is enhanced with a backoff to guarantee that the session as a whole continues to request retransmission until the data is eventually received. To avoid sustained congestion, the successive request intervals are chosen with exponentially larger values, much as Ethernet's exponential backoff scheme ensures stable operation.

Since the request is multicast to the entire group, any host can answer it. The reply is generated by an arbitrary receiver follows:

- upon receipt of REQ_D from host A, if D is locally available, choose a random delay τ on $[C_1 d(A), (C_1 + C_2) d(A)]$

- schedule the repair packet REP_D for transmission in τ seconds.

- if REP_D is received before τ seconds, then cancel the repair timer.

- otherwise, send REP_D after τ seconds.

The net effect of the randomization and network delay term is to bias the retransmissions toward those sites that are the closest to the sender. In a network where loss is often concentrated at the backbone gateways, this scheme is effective because packet drops generally affect a large fraction of the group. Thus, the multicast repair mechanism efficiently repairs a large number of affected receivers using a single packet. However, many backbone networks typically have sufficient capacity to carry all offered traffic and instead congestion and packet losses occur more frequently at the access points to the network. Hence, "loss neighborhoods" are likely to cluster toward the leaves of the distribution tree nearer the edge of the network. In fact, recent experimental work shows that such loss neighborhoods actually occur in the MBone and, as a result, loss patterns across loss neighborhoods tends to be uncorrelated [111]. As the number of loss neighborhoods grows (i.e., as the session gets larger across the wide area), the aggregate probability that a given packet is lost by at least one loss neighborhood approaches one. And since the *global recovery* mechanism described above multicasts both the request (the NACK) and the repair (the retransmission) to the entire group, the efficiency of SRM in this naive form declines substantially.

Local Recovery. The network research community is actively pursuing two approaches to combat the adverse effects of uncorrelated packet loss. The first approach, originally proposed in [38], is called *local recovery*. In local recovery, the reach of a request or retransmission is confined to some localized region of the network, e.g., using the IP Multicast scoping controls described in §2.4. Ideally, this localized region would be just large enough to ensure that the request reaches some other receiver with the data and the response would be multicast with a scope just large enough to reach each affected receiver. In this fashion the loss recovery traffic is spatially localized to a confined region of the network and the overall protocol scales without impacting the global performance across the session. The research community is actively experimenting with different forms of local recovery, e.g., using TTL-scopes [38] or multiple multicast groups [67], but no scheme has yet been deployed or even studied at large scale or under realistic conditions.

FEC Multicast. A second approach for combating uncorrelated loss in reliable multicast is a novel application of forward-error correction coding (FEC) to the SRM repair scheme [82, 52, 84]. The basic idea is to retransmit error-correcting codes rather than the original data itself so that a distinct packet loss or losses in distinct areas of the network can be simultaneously repaired with a single packet. This "simultaneous repair" mechanism utilizes (n, k) block codes [66]. Here, a packet stream is grouped into platoons of n packets each, and whenever a loss is detected the receiver NACKs the platoon rather than the packet. If each receiver indicates the number m of packets lost from that platoon, then the responder can merely send m of k parity packets. Thus, another receiver that lost *any* set of m packets from that same platoon can recover the original data. Moreover, the scheme can be augmented so that the session as a whole computes a maximum number of missing packets, for a given platoon, across all members of the session (i.e., a receiver suppresses its NACK only upon receipt of a NACK indicating a larger or equal number of missing packets). The elegance of this scheme lies in the fact that FEC is used only in recovery not in the original data stream. Thus there is no FEC overhead for a loss-free channel.

To illustrate this algorithm, consider a (2,1) exclusive-or parity code. Suppose at roughly the same time, receiver A loses packet α and receiver B loses packet β. Under global recovery, A requests retransmission of α and B requests retransmission of β, and subsequently the source (or another receiver that has this information) responds by multicasting both α and β. But assuming that receiver A has packet β and receiver B has packet α, we can halve the amount of repair traffic by multicasting the parity of α and β (i.e., $\alpha+\beta$), which is sufficient information for each receiver to recover the missing packet. In a sense, this process is not error-control coding at all, but rather a form of source-coding (i.e., data compression) that accounts for the conditional information known throughout the session.

We have but sketched the high level ideas behind the application of FEC to reliable multicast. This is a new and active research area, and many details, variants, and generalizations are being developed. See [84] for a detailed analysis of the approach that pioneered the use of FEC for reliable multicast.

6 Security

While security and privacy are often overlooked in experimental multimedia systems, their importance in public networks like the Internet and the MBone prompted their integration into the earliest prototypes of the MBone tools. In this last section, we briefly survey the privacy mechanisms used in LWS and identify some of the open problems in multicast security.

A common belief holds that multicast is inherently less secure than unicast because information is sent everywhere. Perhaps this would be true if securing the physical network were the only reliable means of secure transmission. But as public networks become more commonplace, reliance upon securing the physical communications network is becoming difficult and uneconomical if not at times impossible. This problem is crystallized by the classic "end-to-end" argument, which says that it is best to secure data transmission on an end-to-end basis since, in a multi-stage design, the system is only as strong as its weakest link, and moreover, trust must be reflected through all interfaces in the system [92].

To this end, the MBone tools and the LWS architecture utilize end-to-end symmetric-key encryption for privacy. The RTP and RTCP session announcement standards include datagram padding for block ciphers and features to protect against known plain-text attacks, and hence, they are well suited for encryption. The current set of tools implement several variants of encryption within this framework. Because of U.S. export restrictions, the MBone tools are distributed with a trivial-to-break exclusive-or cipher; otherwise, "Triple DES" with cipher-block chaining is typically used for secure communication.

Since the MBone tools and RTP were defined before the IP Security Protocol (IPSEC) [8] was in place, they include their own mechanism for security. Eventually, RTP security may be superseded by lower-layer security protocols.

Once all participants in a session have the requisite encryption key, encrypted communication is trivial. Each packet is simply encrypted and decrypted according to the RTP encryption rules and the algorithm of the underlying cipher. However, the distribution of private keys to all the session members is potentially a very difficult problem that is confounded by the desire to support very large sessions with dynamic membership. Large and heterogeneous environments may require fairly sophisticated trust models (e.g., each participant trusts the moderator and their friends but no one else).

To illustrate the problem posed here, consider an in-progress private session in which a new guest joins. Suppose some protocol securely transports the key to the new guest, after which the guest can communicate with the rest of the group. But if this new participant had saved up all of the previously transmitted encrypted packets, he can now use his new knowledge of the secret key to decode and replay the saved session thereby compromising the privacy of the earlier discussion. Though several works have attacked this problem, none of them offer a truly scalable solution [24, 85, 75]. Dynamic security for multicast remains an area of active research.

A much simpler but less versatile mechanism has been designed for the MBone and LWS [58]. Under this model, encryption of the session announcement protocol is combined with that of the media transport channels. Here, a user registers a set of keys with a session agent or session directory tool like *sdr*. These keys are typically distributed by some secure, out-of-band communication channel (e.g., in person, on the phone, or using public-key cryptography, perhaps "privacy enhanced email" or PGP [115]). When creating a private session, the user specifies one of these registered keys (or types one in manually). Subsequently, the announcement messages for this private new session are multicast encrypted with this same key. Even though different keys are applied to different session announcements, they are all multiplexed together and multicast over the same channel. Normally, the receiving session directory agent would discard these encrypted session announcements because they are incoherent and fail the standard integrity checks. But before discarding a message, the receiving agent attempts to decrypt it with each of the registered keys. If any key produces a valid message, then the session announcement is accepted and presented to the user, and that particular key is associated with that particular session. If the user later joins the session, then the session directory tool launches all of the requisite media tools and configures them with the appropriate encryption key, thereby alleviating the burden of key configuration entirely from the user.

While symmetric-key encryption of multicast sessions supports privacy, it cannot control fine-grained access to the media channels among the trusted members within the session. Instead, access control to a session is handled by floor control mechanisms that are largely external to the media applications. For instance, if an agreed upon floor control protocol instructs some receiver to ignore a host or hosts (or conversely to focus on one or more hosts), then access to the session is effectively controlled not by controlling the communication channel, but instead by instructing the receiver who to ignore and who not to ignore. This model of "receiver orchestrated" coordination and control is elaborated to some extent in [69, 55].

In addition to privacy controls, a comprehensive security architecture must also include mechanisms for authentication. Otherwise, an adversary could subvert a group communication by masquerading as another session member. Some degree of authentication can be derived from a simple IP host address check, but IP source addresses can still be "spoofed". Yet even this simple check cannot be used in SRM, where an arbitrary receiver responds to a NACK and thus can easily alter the contents of the retransmission. To counteract this and maintain the integrity of original transmission, [38] proposes that sensitive data be signed with a cryptographic hash derived from the sender's public key. While this basic approach has been discussed for some years in the research community, to our knowledge it has never actually been implemented in a reliable multicast application. Perhaps this will change if and when a public key infrastructure is in place and more sensitive applications are developed.

7 Summary

In this article, we surveyed the LWS architecture and the protocols and applications that exploit this infrastructure for scalable and effective multicast communication. We described the history of the MBone and the results of this development in the standardization of RTP and the refinement of the whiteboard protocols into SRM. We also discussed the challenges in delivering real-time media streams to heterogeneous receivers and sketched some of the proposed works that address this problem, namely video gateways and Receiver-driven Layered Multicast. Finally, we described the primitive yet effective LWS security model currently in widespread use over the MBone.

While many open problems stand before the universal deployment of IP Multicast (e.g., multicast congestion control, pricing, scalability of the routing infrastructure, and so forth), LWS has proven highly promising and more scalable and flexible than any other known approach for multipoint communication. Together, LWS, RTP, SRM, and the related technologies discussed herein all combine to provide a comprehensive solution for scalable multicast media transmission in heterogeneous packet-switched networks like the Internet. As a natural consequence, many of the ideas and protocols of the LWS framework have undergone or are undergoing commercialization both by large established Internet-oriented companies as well as several smaller start-up ventures. Ultimately, we believe that these technologies may eventually supplant our traditional mainstream media carriers and provide the conduit for digital television broadcasts, global information dissemination, and the emerging "push model" for web-based applications.

8 Acknowledgment

Our work on the MBone tools was carried out in collaboration with Van Jacobson at the Lawrence Berkeley National Laboratory under the support of the Director, Office of Energy Research, Scientific Computing Staff, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098 and of the National Science Foundation under grant MIP-93-21302. Equipment grants and additional support were provided by Sun Microsystems, Digital Equipment Corporation, Silicon Graphics Inc., and Philips. Several colleagues edited drafts of this paper and contributed substantially to its presentation and technical accuracy, including Elan Amir, Deborah Estrin, Randy Katz, Suchitra Raman, Angela Schuett, and Andrew Swan. Last but not least, we thank the MBone research and user communities who collectively made many key contributions to the ideas that underly the technologies presented herein.

References

[1] Video codec for audiovisual services at p*64kb/s, Mar. 1993. ITU-T Recommendation H.261.

[2] Narrow-band visual telephone systems and terminal equipment, Mar. 1996. ITU-T Recommendation H.320.

[3] Procedures for establishing communication between three or more audiovisual terminals using digital channels up to 1920 kbits/s, Mar. 1996. ITU-T Recommendation H.243.

[4] ALBANESE, A., BLÖMER, J., EDMONDS, J., AND LUBY, M. Priority encoding transmission. Tech. Rep. TR-94-039, International Computer Science Institute, Berkeley, CA, Sept. 1994.

[5] AMIR, E., MCCANNE, S., AND KATZ, R. Receiver-driven bandwidth adaptation for light-weight session. In *Proceedings of ACM Multimedia '97* (Seattle, WA, Nov. 1997), ACM.

[6] AMIR, E., MCCANNE, S., AND VETTERLI, M. A layered DCT coder for Internet video. In *Proceedings of the IEEE International Conference on Image Processing* (Lausanne, Switzerland, Sept. 1996), pp. 13–16.

[7] AMIR, E., MCCANNE, S., AND ZHANG, H. An application-level video gateway. In *Proceedings of ACM Multimedia '95* (San Francisco, CA, Nov. 1995), ACM, pp. 255–265.

[8] ATKINSON, R. *Security Architecture for the Internet Protocol*. ARPANET Working Group Requests for Comment, DDN Network Information Center, Aug. 1995. RFC-1825.

[9] BALLARDIE, T., FRANCIS, P., AND CROWCROFT, J. Core based trees (CBT) an architecture for scalable inter-domain multicast routing. In *Proceedings of SIGCOMM '93* (San Francisco, CA, Sept. 1993), ACM, pp. 85–95.

[10] BANERJEA, A., FERRARI, D., MAH, B., MORAN, M., VERMA, D., AND ZHANG, H. The Tenet real-time protocol suite: Design, implementation, and experiences. *IEEE/ACM Transactions on Networking 4*, 1 (Feb. 1996), 1–10.

[11] BERC, L., FENNER, W., FREDERICK, R., AND MCCANNE, S. *RTP Payload Format for JPEG-compressed Video*. Internet Engineering Task Force, Audio-Video Transport Working Group, Nov. 1995. Internet Draft (work in progress).

[12] BIRMAN, K., CHIPER, A., AND STEPHENSON, P. Lightweight causal and atomic group multicast. *ACM Transactions on Computer Systems 9*, 3 (Aug. 1991), 272–314.

[13] CASNER, S., AND DEERING, S. First IETF Internet audiocast. *ConneXions 6*, 6 (1992), 10–17.

[14] CASNER, S., LYNN, J., PARK, P., SCHRODER, K., AND TOPOLCIC, C. *Experimental Internet Stream Protocol, version 2 (ST-II)*. ARPANET Working Group Requests for Comment, DDN Network Information Center, SRI International, Menlo Park, CA, Oct. 1990. RFC-1190.

[15] CHADDHA, N. Software only scalable video delivery system for multimedia applications over hetrogeneous networks. In *Proceedings of the IEEE International Conference on Image Processing* (Washington, DC, Oct. 1995).

[16] CHADDHA, N., WALL, G. A., AND SCHMIDT, B. An end to end software only scalable video delivery system. In *Proceedings of the Fifth International Workshop on Network and OS Support for Digital Audio and Video* (Durham, NH, Apr. 1995), ACM.

[17] CHEUNG, S. Y., AMMAR, M. H., AND LI, X. On the use of destination set grouping to improve fairness in multicast video distribution. In *Proceedings IEEE Infocom '96* (San Francisco, CA, Mar. 1996), pp. 553–560.

[18] CLARK, D. D. The design philosophy of the DARPA Internet protocols. In *Proceedings of SIGCOMM '88* (Stanford, CA, Aug. 1988), ACM.

[19] CLARK, D. D., AND TENNENHOUSE, D. L. Architectural considerations for a new generation of protocols. In *Proceedings of SIGCOMM '90* (Philadelphia, PA, Sept. 1990), ACM.

[20] COHEN, D. *Specifications for the Network Voice Protocol (NVP)*. ARPANET Working Group Requests for Comment, DDN Network Information Center, Information Sciences Institute, 1976. RFC-741.

[21] COHEN, D. On packet speech communication. In *Proceedings of the Fifth International Conference on Computer Communications* (Atlanta, Georgia, Oct. 1980), IEEE, pp. 271–274.

[22] COMPTON, C., AND TENNENHOUSE, D. Collaborative load shedding for media-based applications. *International Conference on Multimedia Computing and Systems* (May 1994).

[23] COVER, T. M., AND THOMAS, J. A. *Elements of Information Theory*. John Wiley and Sons, Inc., 1991.

[24] CURTIS, P., DIXON, M., FREDERICK, R., AND NICHOLS, D. A. The Jupiter audio/video architecture: secure multimedia in network places. In *Proceedings of ACM Multimedia '95* (San Francisco, CA, Nov. 1995), ACM.

[25] DANSKIN, J. M., DAVIS, G. M., AND SONG, X. Fast lossy Internet image transmission. In *Proceedings of ACM Multimedia '95* (San Francisco, CA, Nov. 1995), ACM, pp. 321–332.

[26] DEERING, S. Internet multicast routing: State of the art and open research issues, Oct. 1993. Multimedia Integrated Conferencing for Europe (MICE) Seminar at the Swedish Institute of Computer Science, Stockholm.

[27] DEERING, S., AND CHERITON, D. Multicast routing in datagram internetworks and extended LANs. *ACM Transactions on Computer Systems 8*, 2 (May 1990), 85–110.

[28] DEERING, S., ESTRIN, D., FARINACCI, D., JACOBSON, V., LIU, C.-G., AND WEI, L. An architecture for wide-area multicast routing. *IEEE/ACM Transactions on Networking 4*, 2 (Apr. 1996).

[29] DEERING, S. E. *Multicast Routing in a Datagram Internetwork*. PhD thesis, Stanford University, Dec. 1991.

[30] DELGROSSI, L., HALSTRICK, C., HEHMANN, D., HERRTWICH, R. G., KRONE, O., SANDVOSS, J., AND VOGT, C. Media scaling for audiovisual communication with the Heidelberg transport system. In *Proceedings of ACM Multimedia '93* (Aug. 1993), ACM, pp. 99–104.

[31] DORCEY, T., AND COGGER, R. *CU-SeeMe*. Cornell University. Software on-line[7].

[32] DURRETT, R. *Probability: Theory and Examples*. Brooks/Cole Publishing Company, 1991.

[33] ERRAMILLI, AND SINGH. A reliable and efficient multicast protocol for broadband broadcast networks. In *Proceedings of SIGCOMM '87* (Aug. 1987), ACM, pp. 343–352.

[34] FALL, K., PASQUALE, J., AND MCCANNE, S. Workstation video playback performance with competitive process load. In *Proceedings of the Fifth International Workshop on Network and OS Support for Digital Audio and Video* (Durham, NH, Apr. 1995), pp. 179–182.

[35] FENNER, W. *Internet Group Management Protocol, Version 2*. Internet Engineering Task Force, Inter-Domain Multicast Routing Working Group, Feb. 1996. Internet Draft (work in progress).

[36] FERRARI, D., AND VERMA, D. A scheme for real-time communication services in wide-area networks. *IEEE Journal on Selected Areas in Communications 8*, 3 (Apr. 1990), 368–379.

[37] FLOYD, S., AND JACOBSON, V. The synchronization of periodic routing messages. In *Proceedings of SIGCOMM '93* (San Francisco, CA, Sept. 1993), ACM, pp. 33–44.

[38] FLOYD, S., JACOBSON, V., MCCANNE, S., LIU, C.-G., AND ZHANG, L. A reliable multicast framework for light-weight sessions and application level framing. In *Proceedings of SIGCOMM '95* (Boston, MA, Sept. 1995), ACM, pp. 342–356.

[39] FREDERICK, R. *Network Video (nv)*. Xerox Palo Alto Research Center. Software on-line[8].

[40] FREDERICK, R. Experiences with real-time software video compression. In *Proceedings of the Sixth International Workshop on Packet Video* (Portland, OR, Sept. 1994).

[41] GARRETT, M. W. *Contributions Toward Real-Time Services on Packet Switched Networks*. PhD thesis, Columbia University, 1993.

[42] GARRETT, M. W., AND VETTERLI, M. Congestion control strategies for packet video. In *Proceedings of the Fourth International Workshop on Packet Video* (Kyoto, Japan, Mar. 1991), pp. 1–6.

[43] GARRETT, M. W., AND VETTERLI, M. Joint source/channel coding of statistically multiplexed real-time services on packet networks. *IEEE/ACM Transactions on Networking 1*, 1 (Feb. 1993), 71–80.

[44] HANDLEY, M. *Session DiRectory*. University College London. Software on-line[9].

[45] HANDLEY, M. SAP: Session announcement protocol, Nov. 1996. Internet Draft (work in progress).

[46] HANDLEY, M., AND CROWCROFT, J. Network text editor (NTE): A scalable shared text editor for the MBone. In *Proceedings of SIGCOMM '97* (Cannes, France, Sept. 1997), ACM.

[47] HANDLEY, M., AND JACOBSON, V. SDP: Session description protocol, Nov. 1995. Internet Draft (work in progress).

[48] HARDMAN, V., SASSE, M. A., HANDLEY, M., AND WATSON, A. Reliable audio for use over the Internet. In *Proceedings of INET '95* (Honolulu, Hawaii, June 1995).

[7] ftp://gated.cornell.edu/pub/video

[8] ftp://ftp.parc.xerox.com/net-research

[9] ftp://cs.ucl.ac.uk/mice/sdr/

[49] HEYBEY, A. T. Video coding and the application level framing protocol architecture. Tech. Rep. MITE/LCS/TR-542, Massachusetts Institute of Technology, Cambridge, MA, June 1992.

[50] HOFFMAN, D., FERNANDO, G., AND GOYAL, V. *RTP Payload Format for MPEG1/MPEG2 Video*. Internet Engineering Task Force, Audio-Video Transport Working Group, June 1995. Internet Draft (work in progress).

[51] HOFFMAN, D., AND SPEER, M. Hierarchical video distribution over Internet-style networks. In *Proceedings of the IEEE International Conference on Image Processing* (Lausanne, Switzerland, Sept. 1996), pp. 5–8.

[52] HUITEMA, C. The case for packet level FEC. In *Proceedings IFIP 5th International Workshop on Protocol for High Speed Networks* (INRIA, Sophia Antipolis, France, Oct. 1996).

[53] JACOBSON, V. *Session Directory*. Lawrence Berkeley Laboratory. Software on-line[10].

[54] JACOBSON, V. Congestion avoidance and control. In *Proceedings of SIGCOMM '88* (Stanford, CA, Aug. 1988).

[55] JACOBSON, V. SIGCOMM '94 Tutorial: Multimedia conferencing on the Internet, Aug. 1994.

[56] JACOBSON, V., AND DEERING, S. Administratively scoped IP multicast. In *Proceedings of the 30th Internet Engineering Task Force* (Toronto, Canada, July 1994).

[57] JACOBSON, V., AND MCCANNE, S. *Visual Audio Tool*. Lawrence Berkeley Laboratory. Software on-line[11].

[58] JACOBSON, V., MCCANNE, S., AND FLOYD, S. A privacy architecture for lightweight sessions, Sept. 1994. ARPA Network PI Meeting presentation [12].

[59] KASPEROVICH, L. Multiplication free scaled 8x8 DCT algorithm with 530 additions. In *Proc. SPIE* (1995), vol. 2419, ACM, pp. 105–110.

[60] KENT, C. A., AND MOGUL, J. Fragmentation considered harmful. In *Proceedings of SIGCOMM '87* (Aug. 1987), ACM.

[10] ftp://ftp.ee.lbl.gov/conferencing/sd
[11] ftp://ftp.ee.lbl.gov/conferencing/vat
[12] ftp://ftp.ee.lbl.gov/talks/lws-privacy.ps.Z

[61] KOUVELAS, I., HARDMAN, V., AND WATSON, A. Lip synchronisation for use over the Internet: Analysis and implementation. In *Proceedings of GLOBECOM '96* (London, UK, Nov. 1996).

[62] LAMPORT, L. Time, clocks and the ordering of events in a distributed system. *Communications of the ACM 21*, 7 (July 1978), 558–565.

[63] LENGWEHASATIT, K., AND ORTEGA, A. Distortion/decoding time trade-offs in software DCT-based image coding. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing* (Munich, Germany, Apr. 1997).

[64] LEVINE, B. N., LAVO, D. B., AND GARCIA-LUNA-ACEVES, J. The case for reliable concurrent multicasting using shared ack trees. In *Proceedings of ACM Multimedia '96* (Boston, MA, Nov. 1996), ACM.

[65] LIN, J. C., AND PAUL, S. RMTP: A reliable multicast transport protocol. In *Proceedings IEEE Infocom '96* (San Francisco, CA, Mar. 1996), pp. 1414–1424.

[66] LIN, S., AND COSTELLO. *Error Control Coding: Fundamentals and Applications*. Prentice-Hall, Englewood Cliffs, NJ, 1983.

[67] LIU, C.-G. Local error recovery in SRM: Comparison of two approaches. Technical Report USC-CS-97-648, University of Southern California, 1997.

[68] MCCANNE, S. A distributed whiteboard for network conferencing, May 1992. U.C. Berkeley CS268 Computer Networks term project and paper.

[69] MCCANNE, S., AND JACOBSON, V. *vic:* a flexible framework for packet video. In *Proceedings of ACM Multimedia '95* (San Francisco, CA, Nov. 1995), ACM, pp. 511–522.

[70] MCCANNE, S., JACOBSON, V., AND VETTERLI, M. Receiver-driven layered multicast. In *Proceedings of SIGCOMM '96* (Stanford, CA, Aug. 1996), ACM, pp. 117–130.

[71] MCCANNE, S., AND VETTERLI, M. Joint source/channel coding for multicast packet video. In *Proceedings of the IEEE International Conference on Image Processing* (Washington, DC, Oct. 1995), pp. 25–28.

[72] McCanne, S., Vetterli, M., and Jacobson, V. Low-complexity video coding for receiver-driven layered multicast. *IEEE Journal on Selected Areas in Communications 15*, 6 (Aug. 1997), 983–1001.

[73] McCanne, S. R. *Scalable Compression and Transmission of Internet Multicast Video*. PhD thesis, University of California, Berkeley, Dec. 1996.

[74] Mills, D. L. *Network Time Protocol (version 2) specification and implementation*. ARPANET Working Group Requests for Comment, DDN Network Information Center, SRI International, Menlo Park, CA, Sept. 1989. RFC-1119.

[75] Mittra, S. Iolus: A framework for scalable secure multicasting. In *Proceedings of SIGCOMM '97* (Cannes, France, Sept. 1997), ACM.

[76] Mockapetris, P. V., and Dunlap, K. Development of the domain name system. In *Proceedings of SIGCOMM '88* (Stanford, CA, Aug. 1988), ACM.

[77] Mogul, J., and Deering, S. *Path MTU Discovery*. ARPANET Working Group Requests for Comment, DDN Network Information Center, SRI International, Menlo Park, CA, Nov. 1990. RFC-1191.

[78] Moon, S. B., Kurose, J., and Towsley, D. Packet audio playout delay adjustment: performance bounds and algorithms. *Multimedia Systems 6*, 1 (Jan. 1998), 17–28.

[79] Moser, L. E., Melliar-Smith, P. M., Agarwal, D. A., Budhia, R. K., and Lingley-Papadopoulos, C. A. Totem: A fault-tolerant multicast group communication system. *Communications of the ACM 39*, 4 (Apr. 1996), 54–63.

[80] Mounts, F. W. A video encoding system with conditional picture-element replenishment. *Bell Systems Technical Journal 48*, 7 (Sept. 1969), 2545–2554.

[81] Netravali, A., and Haskell, B. *Digital Pictures*. Plenum Press, New York, NY, 1988.

[82] Nonnenmacher, J., and Biersack, E. W. Reliable multicast: Where to use FEC. In *Proceedings IFIP 5th International Workshop on Protocol for High Speed Networks* (INRIA, Sophia Antipolis, France, Oct. 1996), pp. 134–148.

[83] Nonnenmacher, J., and Biersack, E. W. Optimal multicast feedback. In *Proceedings IEEE Infocom '98* (San Francisco, CA, Mar. 1998).

[84] Nonnenmacher, J., Biersack, E. W., and Towsley, D. Parity-based loss recovery for reliable multicast transmission. In *Proceedings of SIGCOMM '97* (Cannes, France, Sept. 1997), ACM.

[85] Oppliger, R., and Albanese, A. Participant registration, validation, and key distribution for large-scale conferencing systems. *IEEE Communications Magazine 35*, 6 (June 1997), 130–134.

[86] Pasquale, J. C., Polyzos, G. C., Anderson, E. W., and Kompella, V. P. Filter propagation in dissemenation trees: Trading off bandwidth and processing in continuous media networks. In *Proceedings of the Fourth International Workshop on Network and OS Support for Digital Audio and Video* (Lancaster, U.K., Nov. 1993), ACM, pp. 269–278.

[87] Pasquale, J. C., Polyzos, G. C., Anderson, E. W., and Kompella, V. P. The multimedia multicast channel. *Internetworking: Research and Experience 5*, 4 (Dec. 1994), 151–162.

[88] Postel, J. *User Datagram Protocol*. Internet Engineering Task Force, USC/Information Sciences Institute, Aug. 1980. RFC-768.

[89] Raman, S., McCanne, S., and Shenker, S. Asymptotic scaling behavior of global recovery in SRM. In *Proceedings of SIGMETRICS '98/PERFORMANCE '98 Joint International Conference on Measurement and Modeling of Computer Systems* (Madison, WI, June 1998).

[90] Rao, A., Lanphier, R., et al. Real time streaming protocol (RTSP), Oct. 1996. Internet Draft (work in progress).

[91] Rizzo, L. Effective erasure codes for reliable computer communication protocols. *Computer Communication Review 27*, 2 (Apr. 1997), 24–36.

[92] Saltzer, J. H., Reed, D. P., and Clark, D. D. End-to-end arguments in system design. *ACM Transactions on Computer Systems 2*, 4 (Nov. 1984).

[93] Schooler, E. M. A multicast user directory service for synchronous rendezvous. Computer science department, California Institute of Technology, Sept. 1996.

[94] SCHOOLER, E. M., AND CASNER, S. L. A packet-switched multimedia conferencing system. *ACM Special Interest Group on Office Information Systems Bulletin 10* (Jan. 1989), 12–22.

[95] SCHULZRINNE, H. 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.

[96] SCHULZRINNE, H. *RTP Profile for Audio and Video Conferences with Minimal Control*. Internet Engineering Task Force, Audio-Video Transport Working Group, Jan. 1996. RFC-1890.

[97] SCHULZRINNE, H., CASNER, S., FREDERICK, R., AND JACOBSON, V. *RTP: A Transport Protocol for Real-Time Applications*. Internet Engineering Task Force, Audio-Video Transport Working Group, Jan. 1996. RFC-1889.

[98] SHANNON, C. E. A mathematical theory of communication. *Bell Systems Technical Journal 27* (1948), 379–423.

[99] SHARMA, P., ESTRIN, D., FLOYD, S., AND JACOBSON, V. Scalable timers for soft state protocols. In *Proceedings IEEE Infocom '97* (Kobe, Japan, Apr. 1997).

[100] TENNENHOUSE, D. L., SMITH, J. M., SINCOSKIE, W. D., WETHERALL, D. J., AND MINDEN, G. J. A survey of active network research. *IEEE Communications Magazine 35* (Jan 1997), 80–86.

[101] TURLETTI, T. *INRIA Video Conferencing System (ivs)*. Institut National de Recherche en Informatique et an Automatique. Software on-line[13].

[102] TURLETTI, T. The INRIA videoconferencing system (IVS). *ConneXions 8*, 10 (Oct. 1994), 20–24.

[103] TURLETTI, T., AND BOLOT, J.-C. Issues with multicast video distribution in heterogeneous packet networks. In *Proceedings of the Sixth International Workshop on Packet Video* (Portland, OR, Sept. 1994).

[104] TURLETTI, T., AND HUITEMA, C. *RTP Payload Format for H.261 Video Streams*. Internet Engineering Task Force, Audio-Video Transport Working Group, Oct. 1996. RFC-2032.

[105] TURLETTI, T., AND HUITEMA, C. Videoconferencing in the Internet. *IEEE/ACM Transactions on Networking 4*, 3 (June 1996), 340–351.

[106] VAN RENESSE, R., BIRMAN, K. P., AND MAFFEIS, S. Horus, a flexible group communication system. *Communications of the ACM* (Apr. 1996).

[107] VISHWANATH, M., AND CHOU, P. An efficient algorithm for hierarchical compression of video. In *Proceedings of the IEEE International Conference on Image Processing* (Austin, TX, Nov. 1994).

[108] WAITZMAN, D., PARTRIDGE, C., AND DEERING, S. *Distance Vector Multicast Routing Protocol*. ARPANET Working Group Requests for Comment, DDN Network Information Center, SRI International, Menlo Park, CA, Nov. 1988. RFC-1075.

[109] WHETTEN, B., MONTGOMERY, T., AND KAPLAN, S. A high performance totally ordered multicast protocol. *Theory and Practice in Distributed Systems 938* (1995).

[110] WU, L., SHARMA, R., AND SMITH, B. Thin streams: An architecture for multicasting layered video. In *Proceedings of the Seventh International Workshop on Network and OS Support for Digital Audio and Video* (St. Louis, Missouri, May 1997), ACM, pp. 183–192.

[111] YAJNIK, M., KUROSE, K., AND TOWSLEY, D. Packet loss correlation in the MBone multicast network. In *Proceedings IEEE Global Internet '96* (London, England, Nov. 1996).

[112] YAVATKAR, R., AND MANOJ, L. Optimistic strategies for large-scale dissemination of multimedia information. In *Proceedings of ACM Multimedia '93* (Aug. 1993), ACM, pp. 1–8.

[113] ZHANG, L., DEERING, S., ESTRIN, D., SHENKER, S., AND ZAPPALA, D. RSVP: A new resource reservation protocol. *IEEE Network 7* (Sept. 1993), 8–18.

[114] ZIMMERMAN, H. OSI reference model — the ISO model of architecture for open systems interconnection. *IEEE Transactions on Communications COM-28*, 4 (Apr. 1980).

[115] ZIMMERMANN, P. *The Official PGP User's Guide*. MIT Press, 1995.

[13] http://www.inria.fr/rodeo/ivs.html