A Proxy-based Framework for Reliable Multicast in Heterogeneous Environments

Yatin Chawathe
yatin@cs.berkeley.edu
Computer Science Division
University of California, Berkeley

Dissertation Proposal (Qualifying Exam)
Committee: Profs. Steven McCanne (Chair), Eric Brewer (Advisor), Randy Katz, and Marti Hearst

Abstract

IP Multicast has proven to be an effective communication primitive for best effort, large-scale, multi-point audio/video conferencing applications. While the best-effort transport of real-time digital audio/video is a relatively straightforward and well understood problem, many other applications like multicast-based shared whiteboards and shared text editors are more challenging to design because their underlying media require reliable transport, i.e., a "reliable multicast" protocol. The design of scalable end-to-end reliable multicast protocols has unfortunately proven to be an especially hard problem, exacerbated by the enormous degree of network and system heterogeneity present in the Internet. In this work, we propose to tackle the heterogeneity problem with a hybrid model for reliable multicast that relies in part on end-to-end loss recovery mechanisms and in part on intelligent and application-aware adaptation carried out within the network. In our framework, a network of application-aware agents – or Reliable Multicast proXies (RMXs) – use detailed knowledge of application semantics to adapt to the effects of heterogeneity in the environment. We identify three key issues that must be addressed by this framework. First, to allow the RM applications to adapt to a wide range of network conditions and client capabilities, the RMXs must intelligently perform bandwidth and data adaptation to best suit the clients' and applications' requirements. Second, to allow applications to customize the RMX behavior based on the concepts of Application Level Framing (ALF), we must define an application API for controlling and manipulating RMXs. Finally, we define and evaluate a software architecture for dynamically deploying RMXs in the wide area.

As a preliminary demonstration of the efficacy of this approach, we have implemented a prototype RMX for a shared whiteboard application for hand-held PDAs. We rely on an implementation of the Scalable Reliable Multicast (SRM) protocol, *libsrm* that we have built at Berkeley. We have built a number of applications on top of this reliable multicast library and plan to evaluate the usefulness of RMXs for these applications.

1 Introduction

The Internet multicast backbone, or MBone [16, 15], forms the conduit for the "IP multicast forwarding service," an extension of the traditional, best-effort Internet datagram model for efficient group-oriented communication. In IP Multicast, each source's data flow is delivered efficiently to all interested receivers according to a multicast routing tree. For large-scale group communication, the bandwidth savings afforded by multicast are enormous, and consequently, a large and growing number of multimedia conferencing tools [27, 37, 23, 36, 22] have been developed that exploit multicast and the MBone.

Though multicast applications reap enormous performance benefits from the underlying multicast service, they are fundamentally challenged by the heterogeneity that is inherent in the disparate technologies that comprise the Internet, both within the end systems and across the network infrastructure. Table 1 shows the high variance in client and network capabilities today. End devices range from simple palm-top personal digital assistants (PDAs) to powerful high-end desktop PCs, while network link characteristics can vary by many orders of magnitude in terms of delay, capacity, and error rate. Although technology continually advances the low end of the heterogeneity spectrum, the gap between low-end and high-end systems will inevitably exist far into the future. Hence, any software system designed to function well across such a wide range of characteristics must adapt to the needs of its environment.

When network heterogeneity convolves with the multicast communication model, a communication source is potentially confronted with a wide range of path characteristics to each receiver, e.g., different delays, link rates, and packet losses. Consequently, that source cannot easily modulate its data stream in a uniform fashion to best match the resource constraints in the network. For example, if the source sends at the most constrained bit rate among all paths to all receivers, then many high-bandwidth receivers experience performance below the network's capability, whereas if the source sends at the maximum possible bit-rate, then low-bandwidth paths become congested and receivers behind these congested links suffer. A source cannot simply transmit a stream at a uniform rate and simultaneously satisfy the conflicting requirements of a heterogeneous set of receivers.

A number of promising works have addressed the problem of multicast heterogeneity in the particular case of real-time audio/video data, and each of these solutions generally falls into one of two categories: end-to-end adaptation based

System Characteristic	Low-end	High-end
Machine	Hand-held PDA	High-end Desktop machine
CPU Speed	16MHz	450 MHz
Screen Resolution	160x160 2-bit gray-scale	1600x1200 24-bit true-color
Memory Capacity	2 MB physical 64 KB address space	128 MB physical 4 GB address space
Network Bandwidth	28.8 modem connection	100 Mb/s Ethernet
Network Latency	200-400 ms wireless [2]	1 ms ethernet

Table 1: **End-client and Network Heterogeneity**

on layered media [50, 41, 10, 38] or proxy-based transcoding embedded within the network [55, 5]. In the former approach, a source encodes its signal in a layered representation and stripes these layers across multiple multicast groups. In turn, receivers individually tune their reception rates by adjusting the number of groups they receive. As a result, heterogeneity is accommodated since each receiver sustains the maximum rate that the network supports.

In the proxy model, media gateways are situated at strategic points within the network and actively transform media streams to mitigate bandwidth heterogeneity and client diversity. By placing a proxy between the source and sink of data, we can accommodate network bandwidth variation through format "distillation" [19] and optimize the allocation of bandwidth across flows using intelligent rate adaptation [3, 5]. Moreover, the proxy can translate the underlying media representations to enable communication among otherwise incompatible clients.

Unfortunately, not all applications are amenable either to layered representation or to transformational compression. Moreover, unlike audio and video, where media streams are ephemeral and packet loss can be gracefully accommodated by momentarily degrading quality, many applications like group whiteboards or shared text editors rely upon "persistent state" and thus require that all data eventually reaches all interested receivers, i.e., such applications require a "reliable multicast" (RM) transport [17, 52, 35]. Coping with network heterogeneity in these cases is more challenging compared to the unreliable case because the goals for reliability imply that information cannot be discarded to create a heterogeneous set of transmission rates. In other words, the source is fated to run at an average rate at or below the most constrained receiver's rate.

In this work, we propose a twofold solution to this problem by (1) relaxing the semantics of reliability, and (2) decoupling the members of the reliable multicast session through a proxy-based communication model. In relaxing the semantics of reliability, we lift the constraint that all receivers advance uniformly with a sender's data stream. To this end, we leverage the Application Level Framing (ALF) protocol architecture [13], which says that application performance can be substantially enhanced by reflecting the application's semantics into the design of its network protocol. Thus, to accommodate network heterogeneity for reliable multicast, we allow each receiver to define its own level of reliability and to decide how and to what degree individual application data units (ADUs) might be transformed and compressed thereby admitting a scenario where receivers "tap" into the multicast session at a variety of rates. To support these semantics, the end-client must be able to interact with a network infrastructure that supports receiver-directed reliability and programmable transformation. We thus adopt a proxy architecture, where computational and protocol bridging elements are embedded within the network, and end-clients interact with these components to customize their transport decisions in a fine-grained, application-specific fashion.

Although significant work has been carried out with respect to proxy architectures for web access and real-time media gateways, to our knowledge, the proxy concept has yet to be applied to the rate-adaptation problem for reliable multicast. This work proposes a general software architecture, based on Reliable Multicast proXies (RMX), which allows heterogeneity to be accommodated in the context of reliable multicast. This framework is based on the following design principles:

- The proxies utilize a divide-and-conquer strategy to split a large and complex heteregeneous problem into many smaller and simpler homogeneous sub-problems.
- The proxy components exploit application-specific information to optimize the client/network adaptation process.
- The transport protocol is tuned for specific environments by making explicit use of knowledge from the session and application layers. This form of cross-layer optimization enables better performance and smarter adaptation.
- We leverage the semantics of the data when creating data adaptation algorithms. For example, lossy compression is a powerful form of dynamic data adaptation [19] that can give much better results than general lossless compression schemes by discarding data which would not be usable by a low-capability client (e.g., image resolution can be reduced for a smaller screen size).

This is accomplished by logically partitioning the multicast session into a number of clouds of "pseudo-homogeneous" members. Each cloud has a representative RMX that acts as the communication gateway between the cloud and the rest of the session. Figure 1 shows a typical RMX configuration. Such a configuration effectively localizes the problems that are exacerbated by heterogeneity, such as congestion control, loss recovery, bandwidth allocation, etc. This framework requires the following key issues to be addressed:

Mechanism We define an architecture for reliable multicast using proxy components spread across the network. We describe an abstract model for an RMX, and use this model to demonstrate the ability of the framework to adapt to network and client heterogeneity.

Control We discuss means to allow applications to control and manipulate RMXs in order to tailor them according to the principles of ALF. We propose a scheme for adaptive bandwidth allocation that takes into account receiver and application interest in the RMX decisions.

Deployment We investigate the issues related to dynamic placement of RMXs at appropriate locations in the network. Ratnasamy *et al.* [45] have proposed a scheme called the Group Formation Protocol (GFP) for dynamically grouping receivers within a multicast session that appear to see correlated losses from the source. We use this as a basis for figuring out optimal placement of RMXs.

Applications We build upon this framework to support a range of reliable multicast applications such as shared electronic whiteboards, software distribution, information dissemination and web caching. We will evaluate the performance of the framework in the context of these applications.

The RMX framework is designed to allow the use of different reliable multicast protocols in the various clouds in the session. However, in order to best exploit the principles of Application Level Framing, we plan to concentrate our initial efforts on Scalable Reliable Multicast (SRM) [17] which was specifically designed to accomodate application semantics in its transport protocol. We build upon *libsrm* [42], an implementation of SRM developed at Berkeley.

Our preliminary exploration into the rich space of the RMX infrastructure is described in [12]. In that paper, we discuss some initial work on the RMX abstract model and present a prototype implementation of an RMX for enabling reliable multicast sessions such as shared electronic whiteboards for impoverished devices like the PalmPilot PDA.

The rest of this proposal is organized as follows. Section 2 describes our architecture for the RMX infrastructure. In Section 3 we discuss an abstract model for an RMX and the various functions that the RMX performs. Section 4 presents a prototype RMX implementation. We describe our strategies for controlling and customizing RMXs in Section 5 and discuss how the ALF philosophy affects the RMX design. We talk about deployment issues in Section 6. Finally, Section 8 lists some related work, and Section 9 presents our research agenda for this dissertation.

2 An Architecture for Heterogeneous Reliable Multicast

In the unicast world, TCP has proven to be the protocol of choice for most applications that require reliable unicast communication. TCP was designed to be a generic transport protocol for reliable unicast communication: it provides a simple ordered reliable data stream. Although several researchers have proposed similar generic transport protocols for reliable multicast communication [52, 35, 31], Floyd *et al.* [17] demonstrate how different multicast applications have widely different requirements for reliability, and hence propose a scheme for reliable multicast that can take into account these diverse application needs. Some applications may require ordered delivery while other may not care while still others may care about complete reliability for only a subset of the application's data. Although it is possible to design a protocol taking into account the worst-case requirements, such a protocol would not be optimized for specific applications. To overcome this weakness of "one size fits all" type of protocols, Clark and Tennenhouse [13] proposed a protocol architecture called Application Level Framing (ALF) that explicitly includes the application's data semantics in the design of the transport protocol. ALF claims that the best way to design a protocol that can meet the diverse reliability requirements of applications is to directly involve the application in the reliability and loss recovery mechanisms of the protocol. The application is best suited to make the most intelligent decisions with respect to what portions of the data space are required to be reliable and what sort of ordering constraints are needed.

The Scalable Reliable Multicast (SRM) protocol [17] leverages these ALF principles in its design of a reliability protocol for multicast. It builds upon the basic IP multicast delivery model—best-effort delivery to a dynamic group of receivers with possible duplication and reordering of packets—and builds reliability on an end-to-end basis. By allowing a receiver to selectively repair portions of the data stream, it effectively accounts for the application's semantics in the design of its network protocol. While conceptually the design of a receiver-reliable transport protocol seems straightforward, realizing a toolkit/library for such an application-specific protocol can run into problems with using traditional transport primitives. Primitives such as sequence numbers hide the structure of the application's data space from the transport protocol. In order to allow the transport protocol to be fine-tuned to the needs of the application, we use a richer naming scheme—SNAP—that is more amenable to ALF [43]. This is discussed in more detail in Section 5.1.

While SRM provides the basic building blocks for an end-to-end reliable multicast framework, it has no support for dealing with heterogeneity in the network and in end-client devices or applications. Dealing with congestion and bandwidth management is essential to the operational success of a wide-area reliable multicast protocol. The SRM toolkit (*libsrm*) built at Berkeley [42] provides a basic mechanism for a fixed bandwidth constraint over the aggregate session. This works only as long as the bandwidth constraint is low enough to suit the most poorly connected client. Obviously, this solution does not work well when the session consists of a range of network and client characteristics.

Unicast congestion control over the wide area even in the face of heterogeneity is easier to manage than that for multicast. In the unicast case, there is a single path from the source to its destination and a feedback loop between the sender and the receiver that can be used to detect and react to congestion. The problem gets much harder in the multicast arena where there can be multiple senders, multiple communication paths, and competing congestion on the

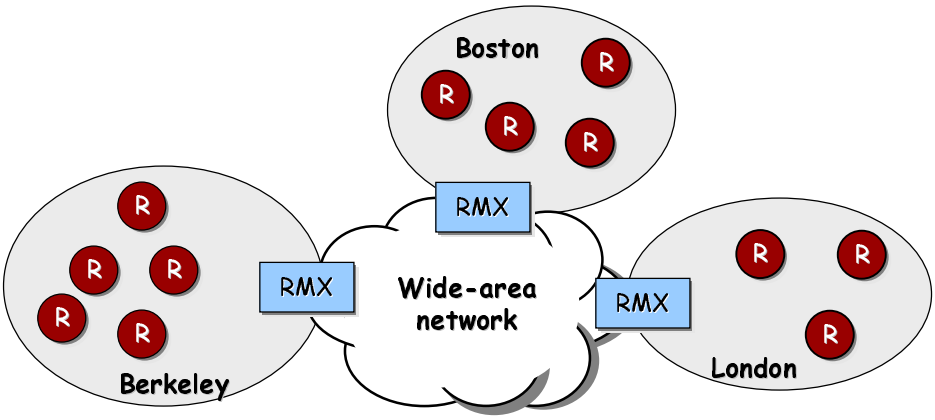

Figure 1: **The RMX Architecture**.

paths to the different receivers. This multiplicity of paths and the possibility of multiple congestion points along independent sections of the paths imposes great difficulty on the design of an end-to-end congestion control scheme for reliable multicast. A good congestion control algorithm should try to match the load imposed by the end-systems to the resources available within the network and to dynamically adapt the load to changes in network characteristics. Researchers have proposed various congestion-control mechanisms for multicast that can co-exist with TCP [57, 59]. However, all of these schemes suffer from limitations: most are single-source-based schemes, and none of the schemes address the issue of bandwidth heterogeneity along the multicast distribution tree. As outlined in Section 1, this heterogeneity compounds the problems of multicast congestion control.

A crucial requirement for tackling the heterogeneity problem in multicast sessions is some scheme for exploiting the structure and topology of the underlying multicast distribution tree. By explicit knowledge of the underlying topology we can deal with the heterogeneity in the network with a *divide and conquer* approach: creating multiple groups of colocated receivers arranged in a hierarchy to essentially localize the effects of congestion and network heterogeneity. In this way, one large heterogeneous and difficult problem is split into multiple smaller homogeneous sub-problems. By topologically clustering the entire multicast session into clouds of colocated homogeneous members, we can reduce the wide-area heterogeneous congestion control problem into a simpler problem of rate adaptation within homogeneous clusters.

Figure 1 shows a picture of an architecture for wide-area reliable multicast using proxies to mitigate the effects of heterogeneity. The entire multicast session is split into a number of clouds of essentially homogeneous participants based on topological proximity. Each cloud contains a representative proxy agent that deals with the heterogeneity issues across clouds. The RMX may be a designated receiver from the group of session participants, or a specialized agent that is strategically placed in the network to service this group of participants.

The various RMXs that are spread across the network may communicate with each other over a global multicast channel, or may be connected to each other via unicast links. The networking research community has studied the effects and behavior of unicast congestion control in protocols such as TCP, and these are well-understood and robust systems. We can leverage this work in the design of a usable wide-area reliable multicast architecture by relying on a robust unicast protocol for communicating across RMXs over the wide-area and relying on simple local-area multicast congestion control schemes within each cloud.

The feasibility of the RMX architecture depends upon the ability to group session participants into topologically sensitive clusters. This requires information about the underlying multicast distribution tree that the IP service model deliberately hides from higher transport and application layers. We describe a Group Formation Protocol [45] for inferring the distribution tree structure and its application to our architecture in more detail in Section 6.

3 An Abstract Model for an RMX

Based on the principles outlined earlier, we present a generic model for reliable multicast proxies. Figure 2 shows the different components of the RMX model. The RMX splits the session into multiple proxied session. Essentially, the RMX separates the cloud of clients that it serves—the *proxied* session—from the rest of the *global* session. The *Global Session Agent* serves as the interface to the main multicast session. The *protocol adapter* is the core of the RMX, and uses the *transformation engines* to assist in converting the *data store* between the formats of the main session and the proxied session. Finally, the *proxy agent* serves as the interface to the proxied session.

The global agent is the proxy's interface to the reliable multicast session. It participates in the RM session on behalf of the RMX clients, handles the details of the commu-

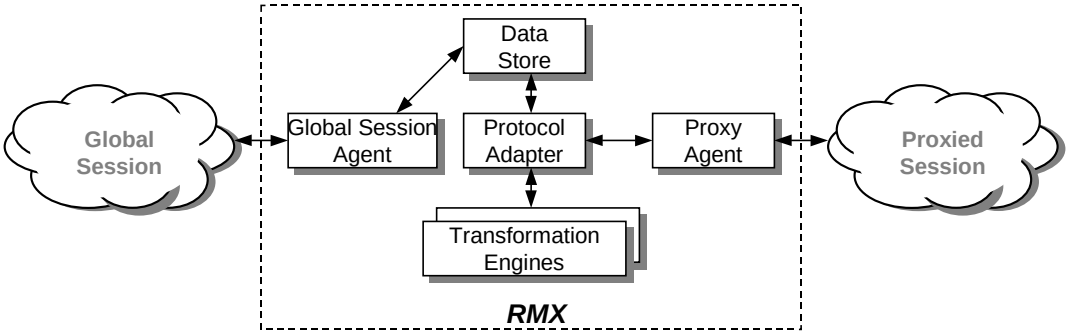

Figure 2: **The RMX Model**.

nication protocol, and recovers lost data by requesting the missing data units from other members of the session. Conceptually, the global agent builds a data store of all "objects" that are part of the reliable session. The data store is updated whenever data is received either from the RM session or from the proxied session. When the global agent receives a data object, it adds it to the data store. If the data store is updated with data from the proxied session, the global agent propagates the data to the multicast session.

The data store is a "soft" copy of the reliable multicast data associated with the session. The global agent uses the loss recovery mechanisms built into the protocol to construct the data store. In the event that the store is lost due to a system crash, it can be regenerated by recovering the lost data from other agents in the reliable multicast session.

The protocol adapter and proxy agent provide the interface to the proxied session. The proxy agent implements the actual communication protocol to the clients. This protocol may be another instance of a reliable multicast session using the same or some other RM protocol, or a totally different communication protocol such as TCP. The design of the proxy agent is driven by ALF principles and depends largely on the characteristics of the proxied clients and network. For example, clients that do not have multicast support can use a unicast protocol agent that provides a tunnel between the multicast session and the client. On the other hand, an RMX agent might simply carry out congestion control by limiting its transmission rate according to application-specific policies. In this case, two instances of the same RM protocol run on both sides of the proxy and another RM agent communicates with the proxied session.

The protocol adapter is the most sophisticated component of the RMX model. Not only does it provide the requisite functionality for heterogeneous environments, but it also relies heavily on ALF to achieve reasonable performance.

3.1 The Protocol Adapter

The protocol adapter, which is interposed between the data store and the proxy agent, orchestrates all data transformations to best adapt the multicast communication for the environment at hand. The adapter relies upon three core forms of dynamic adaptation: rate adaptation, data transformation, and protocol conversion.

3.1.1 Rate Adaptation

A rate-limiting adapter reduces the rate at which data flows from the RM session to the proxied session and vice versa. Clients that are connected to the MBone via low-bandwidth links can use this form of RMX to participate in RM sessions without getting overwhelmed by data arriving at a faster rate than they can handle. Section 3.1.2 describes how application semantics can assist in rate adaptation by controlling the data that is transmitted across the network.

Rate-limiting adapters also provide a mechanism for connecting low-bandwidth clients to an RM session that uses rate-based congestion control. In the face of widely varying network connectivity, traditional congestion control algorithms [57, 39] effectively limit the overall bandwidth of the session to that of the slowest client. But, by interposing a rate-limiting proxy between the low- and high-bandwidth clients, we alleviate this problem, effectively splitting the RM session into independent partitions. The proxy participates in both sessions and limits the data rate in the low-bandwidth session. The proxy may use an independent algorithm in the low-connectivity region, and because the sessions are decoupled in this fashion, low-bandwidth clients do not adversely impact the reception rates of the well-connected session participants.

3.1.2 Data Transformation

Since the protocol adapter is tightly coupled to the application, it can exploit application-level knowledge to transform data objects while shuttling them between the data store and the proxied session. Data transformation serves two important purposes. First, it allows the proxy to adapt the data according to the clients' device characteristics as clients may be incapable of handling certain data types. For example, many PDAs do not support standard image formats such as JPEG and GIF and instead use simple bitmap representations. The protocol adapter can convert these more complex data types into representations that an unsophisticated client can easily understand.

Second, active data transformation allows the system to carry out rate adaptation through compression, which can either be lossy or lossless depending on the nature of the underlying data. Images and video data are prime candidates for lossy compression, since much of the color information and resolution can be reduced or discarded, often without degrading the information conveyed by the image. This form of lossy compression is particularly helpful when the client devices are physically incapable of handling color or high resolution, and such information would be discarded at the client in any case. For data that cannot tolerate any loss, the protocol adapter uses lossless compression. An even better form of dynamic data adaptation involves the use of progressive data formats such as progressive JPEG [58] (or any of a multitude of research codecs based on sub-band transforms [47, 51] or hierarchical vector quantization [10]); with such formats, the adapter initially generates a low quality image for the client and gradually fills in higher quality information in the background.

The protocol adapter uses specialized transformation engines to perform these conversions. These engines can often be built from off-the-shelf code such as image conversion and compression algorithms and data compression routines.

3.1.3 Protocol Conversion

While the data transformation stage described above modifies the representation of individual objects or groups of objects to meet bandwidth constraints, the protocol conversion stage, in contrast, bridges together diverse protocol families running in different sub-sessions across the network. Our premise is that the different regions of a diverse network environment might be best served by an equally diverse range of reliability mechanisms and each such region should be optimized by locally deploying the most suitable protocol, e.g., hop-wise ARQ might be appropriate to effectively accommodate the high loss-rates of a series of radio links, while SRM [17] works well in a high-bandwidth LAN, and Lorax [31] is better for a wide area topology arranged as a tree. To this end, the RMX framework allows us to seamlessly integrate a diverse set of protocols running across a disjoint set of network clouds.

RMX supports two different variants of protocol conversion: transport-level conversion and application-level conversion. In transport-level conversion, the protocol adapter acts as a bridge between two different transport protocols, such as a reliable multicast protocol like SRM and some other protocol, say TCP. This allows multicast-incapable clients (e.g., behind an ISDN or modem line) to access a reliable multicast session. Though one could argue that thin clients such as PDAs should include multicast in their network stacks, the fact is that many simply do not, and instead, we rely upon our protocol adapter to provide a unicast tunnel to such clients.

Transport-layer protocol adaptation is not limited to conversion between multicast and unicast protocols as the RMX can also mediate among different flavors of reliable multicast. By exploiting application-specific knowledge, the protocol adapter can provide interoperability across the wide range of reliable multicast protocols that are in use in research and commercial communities, e.g., Scalable Reliable Multicast (SRM) [17], Pretty Good Multicast (PGM) [52], Reliable Multicast Transport Protocol (RMTP) [35], etc.

In contrast to transport-layer conversion, application-layer conversion modifies the actual application objects to mitigate fundamental semantic discontinuities across diverse applications. In this case, the entire application-level data is transformed from one format to another. Examples of such adaptation include the following:

- Consider two desktop applications designed to implement a shared whiteboard. These applications, if designed without a common standard, will use completely different protocols and data formats for communication within the session. We can build an RMX to bridge the gap between these two applications. Such a proxy must maintain two data stores, one for each application format, and the protocol adapter must intelligently map objects and operations in one data store to the other.

- A second scenario is a proxy for communicating with computationally impoverished clients. Such a client (say, a PDA) may be too limited to handle the full complexities of the application data. Hence the proxy must convert the entire data store to a much simpler representation before relaying it to the client. We elaborate on this in section 4.3.1 while discussing our example prototype.

4 A Prototype: Shared Whiteboard Proxy for PDAs

We now use a specific example to describe the design and implementation of a prototype RMX system, while demonstrating our use of Application Level Framing to tailor the RMX model to a specific application. We use a shared whiteboard proxy as our motivating example. The proxy is used to enable whiteboard applications for hand-held devices such as PDAs.

The original electronic shared whiteboard application, *wb* [36], was developed at the Lawrence Berkeley Laboratory. Based on their experiences with *wb*, researchers at UC Berkeley have built a second-generation whiteboard tool, *mediaboard* [54]. This application allows a diverse set of media to be created and displayed interactively by a group of users sharing a multicast session. A mediaboard session consists of a shared presentation space that is divided into a number of canvas pages. It supports traditional whiteboard data types such as line drawings and text, and adds support for other media such as images and postscript files.

The mediaboard has been designed with desktop and laptop PCs as the main usage platform. We explore the extension of this application to small hand-held devices or personal digital assistants (PDAs). Most PDAs are too limited in their capabilities to be able to handle the complexities of the mediaboard protocol on their own. For example, while the 3COM PalmPilot PDA [1] has a 64 kilobyte code size limit, the binary for the desktop version of *mediaboard* is several megabytes in size. Given these technical limitations of PDAs, it is not feasible to create a stand-alone mediaboard client on current generation PDAs.

We were able to implement the client mediaboard within the severe limitations of our PDA platform by making extensive use of the ALF principles. The RMX handles most of the complexity of the reliable multicast protocol, requiring little more from the PDA than a simple drawing canvas.

4.1 The PDA Client

The mediaboard client on a PDA must be able to support the standard whiteboard features such as creating, cutting, pasting, and moving objects in the shared presentation space. It should allow the user to browse through existing pages without having to communicate with the proxy every time the user switches to a new page. Moreover, given the physical limitations of the PDA screen, it is important that the client be able to pan around the current page and zoom in and out to different levels of granularity. We used the 3COM PalmPilot [1] as our testbed. Figure 3 shows screenshots of the desktop and PDA versions of the mediaboard application.

4.2 The mediaboard Proxy

In this section, we analyze the individual components of the RMX model and demonstrate how we specialize them to the requirements of this application. Most PDA clients, including the PalmPilot, do not support multicast; hence the proxy agent for the mediaboard RMX must map the unicast world of the client to the multicast session. To preserve reliability, we use TCP for communication between the clients and the proxy. For every client connected to the proxy, the proxy agent maintains a connection object which encapsulates the per-client state at the proxy. It contains up-to-date information about the client's device characteristics, the current page, and the current zoom level. The protocol adapter uses this information to assist it in the adaptation process.

The global agent participates in the mediaboard session on behalf of all clients. It is built using the SRM framework that was developed for the desktop mediaboard application. The global agent joins the multicast group for the mediaboard session and uses the desktop mediaboard protocol to communicate with the rest of the session. It deals with losses that occur in the session, and uses the reliability machinery in the protocol to request lost data objects and repair them [17]. Each data object in the mediaboard protocol is a "command" that performs a certain action on the shared drawing space. Commands are associated with a specific page and client in the session.

When the global agent receives mediaboard commands from the multicast session, it adds them to the data store. The data store is organized hierarchically in order to separate the data associated with the various pages and clients in the session. Similarly, when the proxy agent receives data from the PDA client, it hands the data over to the protocol adapter which in turn adds mediaboard commands to the data store. The global agent picks these commands up from the store and sends them to the rest of the session.

The protocol adapter implements the details of the mediaboard protocol and provides the interface to a simplified protocol that is used for communication with the PDA.

4.3 The Protocol Adapter

The protocol adapter for the mediaboard proxy implements all three aspects of adaptation discussed in section 3.1.

4.3.1 Protocol Conversion

The proxy agent uses TCP to communicate with the PDA clients. The protocol adapter provides a bridge between the TCP and SRM sessions. Moreover, to ensure that the client implementation is as straightforward as possible, the protocol adapter handles all the complexities of the mediaboard. The client, instead, receives only a sequence of simple draw operations (*draw-ops*). The protocol adapter transforms the entire data store of mediaboard commands into a "pseudo-canvas" by executing each command and storing its result in the canvas. The draw-ops on the pseudo-canvas are what is transmitted to the PDA. For example, to eliminate any unnecessary state at the client, all undo operations are performed entirely by the protocol adapter and are converted into appropriate draw-ops before sending them to the client.

Since a client may join a mediaboard session at any time in the life of the session, the protocol adapter must be able to replay all past events that have happened on the pseudo-canvas. Hence, the canvas caches a history of the effects of all mediaboard commands in memory. When a new client joins the session, it can replay this history.

4.3.2 Data Transformation

In addition to converting mediaboard commands into simpler draw-ops, the protocol adapter also converts individual data objects according to the requirements of the clients. The PalmPilot can handle simple draw operations such as lines, circles, rectangles, text, etc. However more complex objects such as images and postscript are too difficult for the PDA to digest on its own. We look at each of these in the following sections.

Image and Postscript conversion: The mediaboard uses the Web standard formats GIF and JPEG for images, which the PalmPilot cannot understand. Implementing decoders for these formats on the PDA is too complex and time-consuming. Instead, we rely on decoders in the proxy. Internally, the PalmPilot uses a simple bitmap representation for images. The proxy converts mediaboard images directly to the PDA's native representation before sending them. Similarly, the proxy must convert postscript data either to images in the PDA's native format or into plain text that can be easily displayed by the client.

The protocol adapter uses specialized image transformation engines to assist it in the conversion. We have implemented an image converter using code developed by Paul Haeberli [21]. The image converter is optimized for the PalmPilot's screen characteristics. In addition to format conversion, it performs lossy compression by scaling down the images according to the zoom level on the client, the screen resolution of the client, and the color depth of the client's screen. The processing steps consist of image resizing, sharpening, adding noise, and dithering.

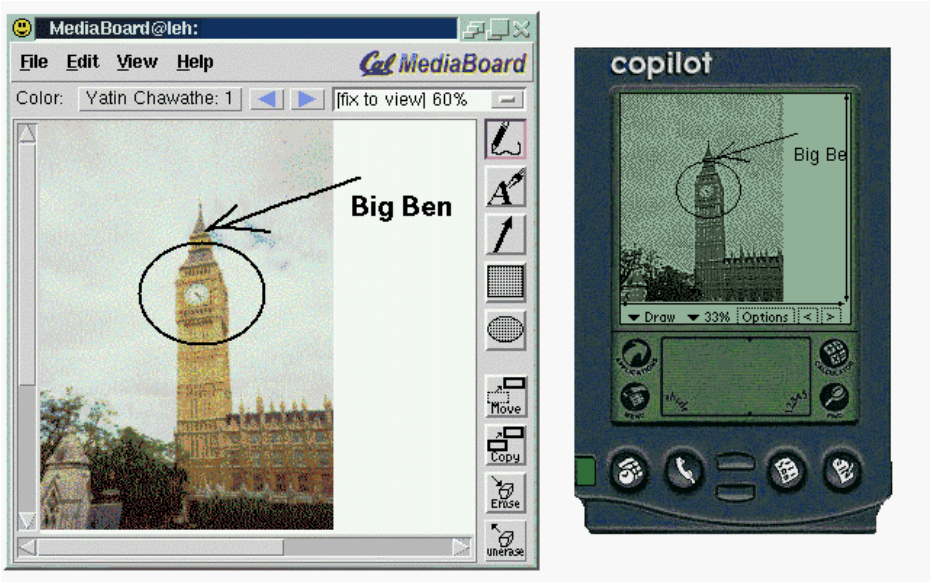

Figure 3: **The desktop and PDA** *mediaboard* **applications.**

Other data types: The protocol adapter also assists the client for seemingly simpler data types such as arrows and fonts. Drawing an arrow requires trigonometric calculations using floating point numbers. The PalmPilot has no built-in floating point hardware and emulation software is either not installed or too slow. Hence the protocol adapter computes the arrow coordinates and sends them as part of the draw-op to the client. Similarly, since the client cannot understand the X Windows-based fonts that are used by the mediaboard protocol for text objects, the protocol adapter converts these font names into reasonable native PDA fonts.

Zooming: Since most PDA screens are extremely small, we support zooming to multiple levels on the client canvas. This enables the user to view the session data at different levels of refinement. The client can handle scaling of simple objects (lines, rectangles and ellipses) on its own. For scaling complex objects, it relies on the proxy. Whenever the user switches zoom levels on the client, it communicates this state change to the protocol agent on the proxy. The protocol adapter is notified of this change, and it recomputes new font mappings for the new zoom level. In addition, the client may request the proxy to send some or all of the displayed images and postscript at the new zoom level. The protocol adapter recomputes the new bitmap representations at the new zoom level and sends them over to the client.

4.3.3 Intelligent Rate Limiting

Since the proxy has complete knowledge of the client's state, the protocol adapter can perform intelligent forwarding of data from the mediaboard session to the client. Lossy image compression is one such mechanism that we use.

By eliminating redundant draw-ops before sending data to the client, we further reduce the number of bytes that must be sent over the low-bandwidth link to the client. For example, if an object has been placed on the canvas and later deleted, the canvas will refrain from sending any information to the client about that object. Similarly, if an object has been moved multiple times, all move operations are combined into a single draw-op before sending it to the client.

Lastly, the protocol agent keeps track of the current page that each client is viewing. The protocol adapter sends only the data associated with that page to the client. All other data is kept buffered in the pseudo-canvas until the client actually switches to a new page. At that time, the protocol adapter collects all new data on that page, packages it into draw-ops, and sends them to the client.

5 Control: Configuring and Manipulating RMXs

Our prototype RMX is very tailored to a specific application: a shared electronic whiteboard for PDAs. Although it uses ALF principles to involve the application in its decisions, it does so in a very ad hoc manner and it is not easy to generalize the prototype to other applications. Application Level Framing presents two conflicting requirements for the design of a generic framework. On the one hand, we need to specialize the RMX according to specific application needs, but on the other hand, we do not wish to implement a new RMX for every new application and client-type. What is required is a common well-defined interface between the application and the proxy that can be used to specify application policies to the RMX. We rely on two separate mechanisms to

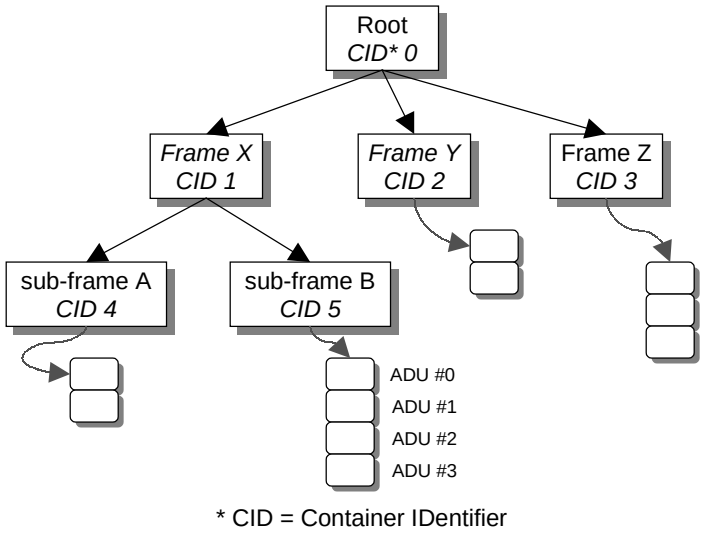

Figure 4: **The SNAP Hierarchical Namespace**.

achieve this:

- A common naming scheme for objects in the reliable multicast data space that is consistent for both the application and the proxy.
- An ability to customize specific components of the RMX by injecting specialization code into the RMX.

We now look at these two mechanisms in detail.

5.1 Hierarchical Data Naming

The ALF principles require that an application be able to control the behavior of the reliable multicast transport protocol. However, if we use traditional schemes for naming protocol data units, such as sequence numbers like in TCP, we would lose a lot of semantic information and structure that an application may desire in order to intelligently control the transport protocol. For example, if the transport library detects that it is missing data associated with sequence numbers 4567 through 6012 and queries the application as to whether it should repair the missing data, the application has no idea what those sequence numbers correspond to in its own data space. Similarly if the transport library has a flat sequence space, it becomes harder for the application to specify flexible ordering contraints. For example, an application that is trying to retrieve many images that are part of a web page may require ordering within each image, but no ordering constraints across images. With a flat sequence space in the transport library, the structure of the various images in the application's data space is lost in the transport layer and it becomes difficult to implement such flexible ordering schemes.

These examples argue for a sematically richer and more structured naming scheme in the transport protocol. If the application can embed its own semantic information into the structure of the transport protocol's name space, then it becomes much easier for the receiving application to control the protocol behavior. In [43], Raman *et al.* proposed a Scalable Naming and Announcement Protocol (SNAP) for the SRM framework. SNAP defines a hierarchical *namespace* that is used to identify data objects between the application and the transport protocol. Structuring data hierarchically allows the application to define different delivery and reliability semantics for different portions of the namespace. Figure 4 shows how a SNAP hierarchy is organized.

An *Application Data Unit (ADU)* is the smallest unit of data that the application can meaningfully handle. The application hands ADUs to the transport protocol and expects them to be delivered atomically to the receivers. ADUs can be of arbitrary size. Large ADUs may be split into multiple packets by the transport protocol. However, it is the job of the protocol to reassemble the ADUs at the receiver before handing them up to the application. Every ADU is mapped into a *container*. A SNAP namespace consists of a hierarchy of containers. Applications map all of their data into the hierarchy. The transport layer assigns each container within a namespace a unique identifier. Applications can attach their own labels to containers. The labels can be arbitrary application level tags such as URL-like strings or <*attribute,value*> tuples that describe the data. Such labels allow the receiving application to determine the semantics of the container hierarchy. A container can refer to other containers or ADUs or both. Within a container ADUs are organized into a flat sequence space.

As an example, a whiteboard [36] source may create a namespace consisting of a root container and many second-level containers, one for each page in the session. Operations on a particular page in the session are ADUs within the corresponding container. The application can associate the page label with the container identifier that is generated by the transport layer.

Each data source in the group creates its own indepen-

dent namespace. This simplifies consistency issues that may arise with multiple sources trying to update the same namespace. Containers within each namespace are the unit of selective reliability: a receiving application can choose to apply different reliability and ordering semantics to different containers. Applications can also use the naming scheme to instruct the RMX to prioritize containers or ADUs within containers based on receiver interest or capabilities. Effectively, the common naming scheme allows us to split the policy decisions for adaptation from the mechanisms and allow the applications to specify different policies via the hierarchy of containers and ADUs.

5.2 Code Injection: Customizing the RMX

Although a common naming scheme between the application and the RMX is a step towards a generic RMX framework, we still need application-specific code in the RMX to assist in the adaptation process. For example, applications may wish to specify transformation routines to convert data objects on the fly inside the RMX before they are downloaded to the clients. These transformations depend heavily on the individual applications under consideration as well as the client devices that the RMX is serving. In order to allow for such application-specific code modules, the RMX must provide a well-defined interface for injecting specialization "code" into the RMX in the form of applets. These applets may be written in a full programming language script such as Java or Tcl. The applets can customize individual RMX components according to the specific needs of the end application or client device.

Dynamic code uploading introduces a number of hard issues such as code safety, resource sharing, etc. A number of researchers are trying to address these issues [14, 53]. For the purpose of this research, we will not focus on these issues.

5.3 Dynamic Bandwidth Adaptation

We now look at how we would apply these mechanisms to the design of a generic RMX for dynamic bandwidth adaptation across a range of applications. Given the heterogeneity across the network, we need to adapt to the varying bandwidth requirements across the different clouds of participants. Since some participants may be interested in high data rate streams of content, while others may not be able to handle the high rates, the RMXs need to intelligently throttle the bandwidth across the bottleneck links and potentially adapt the content (by transcoding it down to lower rates, etc.).

Application semantics are key to being able to fine-tune the bandwidth adaptation process to best suit the individual client/application needs. Depending upon the application content, various schemes can be used to dynamically adjust the data rate and/or prioritize the data objects to be transmitted across the bottleneck links. Layering of the data stream is one approach [50, 41, 10, 38]. The data stream is split into a number of layers, each layer adding a level of refinement to the previous layers. The basic layer is sufficient to recreate a low quality version of the original data; the refinement layers add more detail. Another approach to receiver-driven bandwidth adaptation is a consensus-driven scheme [3]. In such schemes, receivers vote on which data objects are more important than others; the RMX gathers these votes and prioritizes the data objects according to the aggregate votes. In addition to bandwidth adaptation, the RMX can be tuned to specific network or application conditions. For example, an RMX serving a network cloud that is known to have a high error rate (e.g. a wireless network) can be programmed to adjust its data stream to adapt to these specific conditions. Forward error correction is one example of a scheme that can be used in high error-rate situations, rather than strict ARQ-based schemes, to allow receivers to reliably receive the data without having to wait entire round-trip times. Another example of a scheme that works well in high error-rate conditions is the Digital Fountain coding scheme [9]. In this scheme, the sender injects a stream of distinct encoding packets into the network; the key property of a digital fountain is that the source data can be reconstructed at a receiver intact from any subset of the encoding packets that is equal in length to the source data.

We concentrate on consensus-driven schemes for bandwidth adaptation in RMXs. The Hierarchical Data Naming scheme gives us a convenient way to allow the application to hook into the bandwidth adaptation process by allowing end-clients to mark specific containers and/or ADUs are important or as having higher priority than others. Each end-participant can periodically send in its votes for the priorities to associate with specific data objects or groups of data objects within the session. The RMX aggregates these votes and assigns priorities to the containers and ADUs within the SRM namespace. It uses these priorities to schedule individual ADUs to be transmitted to the session participants. Higher priority ADUs are given a larger portion of the available bandwidth, while lower priority ADUs may be suppressed or delayed. In addition, the RMX can be programmed to transcode specific ADUs to reduce their size or to convert them into progressive representations (such as progressive JPEG [24] for still images). An object that is converted into a progressive format can be split up into multiple sub-objects, one for each layer of the progressive representation. The lowest layer is assigned a high priority so that it reaches the end-clients immediately while the enhancement layers slowly trickle down at lower rates.

6 Deployment of RMXs

A crucial aspect of the RMX architecture is the ability of the framework to dynamically deploy RMXs in the appropriate places in the network. Our initial work on the framework assumes static organization of RMXs. The prototype has been built on top of AS1 [4], a framework for supporting network-based services for transcoding, archiving, proxying, etc. of multimedia content. We look at the Active Service approach initially presented in [4] and extend that work for our architecture. We also look at a dynamic protocol for topological clustering of clients in a multicast session—the Group Formation Protocol (GFP) [45]. We plan on investigating the usefulness of GFP for our architecture.

6.1 Active Services

A number of researchers have proposed the use of agents deployed in the network for providing a variety of services to users, for example, web proxies, audio/video transcoding gateways, firewalls, etc. The RMX is an example of a network agent that operates on the reliable multicast data stream and manipulates it to support various client and application classes. Although these agents are embedded within the network infrastructure, an important feature of a large class of such agents is that they are created, configured, manipulated, and controlled at the application layer by employing application-specific protocols.

The existing IP service model does not provide any mechanisms for deploying such agents at appropriate places in the network. In [4], Amir *et al.* have proposed a programmable service architecture built on top of the existing IP infrastructure that allows users to install and run agents at strategic locations in the network. Our prototype RMX relies on this Active Service framework for deploying mediaboard proxies. The framework is implemented on top of a cluster of nodes within the network. Clients can request for specific *service agents* (*servents*) to be launched within the cluster to handle a mediaboard session. The Active Service framework hides the issues of scalability, fault tolerance, and robustness from the servents; the servents only need to deal with providing the actual service.

6.2 The Group Formation Protocol

Although the Active Service framework provides the basic mechanisms for deploying RMXs in the network, it does not provide the right primitives for grouping homogeneous clients within a reliable multicast session into topologically sensitive clusters as described in Section 2. Moreover, once the session has been partitioned into groups, we need to pick appropriate locations within each group to place RMXs.

Once again, the IP service model is a "hindrance" to solving this problem. The IP service model was explicitly designed to hide the topology of the underlying network from the end-points. However, to effectively organize the clients within an RM session into groups that can adapt to bandwidth congestion along bottleneck links, that is exactly what is required. The question then arises: is it at all possible to discover topological information in an end-to-end manner given that the existing IP service model deliberately hides topology. In [44], Ratnasamy *et al.* prove that it is indeed possible and present an inference algorithm that determines the logical topology of the multicast routing tree. The algorithm gathers loss statistics for the receivers in the multicast session and estimates the multicast distribution tree based on losses along the shared paths between receivers. However, the algorithm's assumption of global knowledge of losses at all receivers precludes its inclusion in a practical protocol framework.

In [45], Ratnasamy *et al.* build upon their initial tree-inference work and present a more practical protocol building block—a distributed Group Formation Protocol (GFP)—to produce a topologically-sensitive protocol primitive. Using GFP, participants in a multicast session self-organize into a multi-level hierarchy of groups where the hierarchy is congruent with the multicast tree topology from the source of the session.

6.3 Using GFP for the RMX framework

In order to properly deploy RMXs in the network, it is crucial that all colocated homogeneous receivers that share the same loss sub-tree be coordinated in their actions such as tuning to the same RMX. Moreover, it is important that the RMXs be placed intelligently and dynamically at the right points in the network. We can leverage GFP to solve these problems. By building a hierarchy of groups that correspond to the different colocated homogeneous sub-trees, we can ensure coordination. Also, GFP picks a representative member for each group; the representative is typically the member that is closest to the source in that group. This member is an ideal place for deploying RMXs.

However, in its current state, GFP generates topology information based on source-rooted trees for single source sessions. We plan on investigating the extension of GFP for multi-party sessions. Using GFP as a basic building block, we plan to implement a dynamic deployment algorithm for RMXs that exploits the topology information generated by GFP.

7 RMX Applications

We plan on implementing bandwidth adaptation RMXs that can serve a variety of reliable multicast applications; we plan to build a deployment framework that can dynamically place these RMXs at strategic locations in the network. In order to test the flexibility of the framework, we have included a range of applications varying from interactive drawing tools to bulk data transfer applications.

Shared Electronic Whiteboards We have already built a prototype RMX for shared electronic whiteboards across a range of client devices. We plan on extending this work to support bandwidth adaptation and congestion management for large scale wide-area whiteboard sessions.

Information Dissemination PointCast-like applications that deliver stock quotes, news headlines, etc. to clients are examples of a periodic data stream that needs to be delivered reliably to session participants. A key distinction of such applications is that information is in some sense "real-time"; once new information is available, the old data becomes obsolete and no longer needs to be reliable.

Software Distribution This is an example of a bulk data transfer application. The requirements for bandwidth adaptation for such applications are very different from the previous two examples. Most software consists of binary images that cannot be transformed or transcoded in a lossy manner to a lower bit-rate to adapt to lower bandwidth capacities. However, at the same time, they do not have strict time constraints like interactive applications.

We plan on using these applications as a testbed for the efficacy of the RMX architecture.

8 Related Work

The notion of proxies as intermediaries between clients and servers is not new. Numerous proxy mechanisms have been proposed for HTTP [26]. The HTTP proxy mechanism was originally designed for implementing security firewalls. It has since been used in a number of creative applications, including Kanji transcoding [48], Kanji-to-GIF transformation [60], application-level stream transducing [8, 49], and personalized agent services for web browsing [6]. It has been used to hide the effects of error-prone and low-bandwidth wireless links [19, 34]. Bruce Zenel [61] applies the proxy mechanism to the mobile environment: *filters* on an intermediary host drop, delay, or transform data moving between mobile and fixed hosts. However, the filters are part of the application, complicating their reuse and making it awkward to support legacy applications. Proxies have been used as caching and pre-fetching agents [7, 40] to hide latencies in fetching data from across the network. In the context of multicast, [5] is a proxy framework for real-time audio/video data. The InfoPad project [25] used an extreme approach with proxies: move all intelligence into the infrastructure and use the PDA simply as a dumb terminal.

Partitioning of application complexity between the client and infrastructure has been used in other situations. A related project, TopGun Wingman [18], uses an infrastructure proxy to support a simplified web-browser on the PalmPilot. [56] have proposed the use of a simplified document format (HDML) to reduce the complexity of PDA application. The Rover system [28] provides a distributed object model that presents a queued RPC mechanism for disconnected operation and object migration. For example, simple UI code can be migrated to a mobile client, where it uses queued RPC to communicate with the rest of the application running on the server.

Content layering is another scheme that has been widely used to tackle heterogeneity in multicast environments [50, 41, 10, 38]. Layering typically involves encoding the source data into multiple layers; the base layer provides an approximate representation of the original data, each additional layer provides more information about the original data.

Various flavors of reliable multicast protocols have been proposed in the research community. Pragmatic General Multicast (PGM) [52] relaxes the reliability requirements by using a sliding window-based model where reliability is guaranteed only as long as the data is within the current window. If applications desire greater reliability, they build on top of this basic mechanism. Tree-based protocols such as RMTP [35] organize group members into a hierarchical tree structure and aggregate acknowledgments at midpoints in the network. Each branch in the tree has a designated receiver to receive acknowledgments from its children and aggregate them upwards to the sender. Raman *et al.* are investigating a general framework for a reliable multicast transport [42] that take into account application semantics to optimize the protocol performance and behavior.

In addition to ARQ-based schemes for reliable multicast, a number of specialized encoding schemes have been proposed that can avoid the need for requests for retransmission and the consequent round-trip latency that is introduced. Foward error correcting codes are an example of such schemes. Rizzo *et al.* [46] describe a Reliable Multicast data Distribution Protocol (RMDP) that relies on FEC techniques to adapt to client and network heterogeneity. Digital Fountain [9] is another scheme that encodes data in a form that it can be reconstructed from any subset of the encoding packets that is equal in length to the source data. In combination with a clever layering strategy, this scheme can be used to service a heterogeneous set of receivers, with each receiver being able to receive the encoded fountain at whatever rate best suits its network capacity.

In the arena of multicast congestion control, a number of schemes have been proposed [57, 59]. Most of these schemes, however, deal only with a single-source session and do not handle bandwidth heterogeneity very gracefully. Recently, Balakrishnan *et al.* have proposed an architecture for generic congestion management that provides congestion control service to the transport layer: a host's different communication streams use a Congestion Manager service to determine when they can send and at what rate. However, their work does not yet address the issues related to multicast congestion control. Amir *et al.* [3] present a receiver-driven scheme for dynamic bandwidth adaptation that relies on receiver interest to drive the bandwidth allocation process.

A few other researchers have proposed the use of intelligent computing in the network to assist in the design of reliable multicast protocols. Active Reliable Multicast (ARM) [30] uses the concept of "active routers" that can perform customized computation on behalf of the end-points. They provide best-effort soft-state storage and perform the following functions: data caching for local retransmission, NACK fusion/supression, and partial multicasting for scoped retransmission.

Although the Active Networks initiative [53] provides a network service model where all routers in the network can act as computation engines on behalf of the end-clients to implement services such as RMXs, we can get a lot of the benefits associated with Active Networks without having to deploy a brand new network architecture by resorting to application-level service frameworks such as those proposed by [4, 11, 20]. These systems provide an application-level interface to deploy network-based services for performing intelligent computing on behalf of end-clients.

Many researchers have proposed solutions for the problem of finding appropriate locations for deploying these network services, Brian Levine [32] use IGMP MTRACE packets to allow receivers to obtain their path to the source of a multicast group; receivers use the multicast path information to determine how to achieve local error recovery and effective congestion control. Self-organizing Transcoders (SOT) [29] are a scheme for dynamic adaptation of continuous-media applications to varying network conditions by allowing groups of co-located receivers that experience losses due to a bottleneck link to elect a representative transcoder for local repair. [33] describe an algorithm for finding the optimal placement of multiple web proxies among a set of potential sites under a given traffic pattern.

9 Research Agenda

Phase 1 (0-6 months)

- Finish implementation of MBv2, a shared electronic whiteboard, built on top of *libsrm*.
- Design and implement a dynamic bandwidth adaptation RMX for MBv2. Extend the RMX for other applications such as Information Dissemination and Software Distribution.
- Target INFOCOM 2000 (submission deadline: July 1999)

Phase 2 (6-12 months)

- Design and implement the wide-area deployment architecture. Implement GFP (or equivalent protocol) and build the RMX architecture on top of it.
- Preliminary evaluation of the RMX architecture.
- Target SIGCOMM 2000 (submission deadline: January 2000)

Phase 3 (12-18 months)

- Evaluate wide-area performance of the RMX architecture and the dynamic deployment algorithms.

10 Summary

We have proposed to design, build, and evaluate an architecture for reliable multicast applications that can adapt to the heterogeneity that is inherent in the wide-area Internet. We identify three types of heterogeneity that we will address: network heterogeneity (bandwidth and latency variations from Gigabit ethernet to low capacity wireless links), client heterogeneity (ranging from powerful desktops and workstations to impoverished PDAs), and protocol heterogeneity (wide range of reliable multicast protocols, multicast availability, etc.).

Our proposed solution relies on intelligent adaptation agents in the network (Reliable Multicast proXies or RMXs) that can help mitigate the heterogeneity problems by acting as intermediaries between the source and the consumer of the data and dynamically adapting the content and/or the rate of the data to best suit the clients' needs and interests. We will look at three key issues associated with this architecture:

Mechanisms: We will define an abstract model for an RMX and use this model to demonstrate the framework's ability to adapt to heterogeneity.

Control: We will build a control framework for dynamically manipulating the RMXs and configuring them to current client and network conditions. We rely on a flexible hierarchical naming scheme (SNAP) to identify objects within the application's name space. We use dynamic code injection to specialize individual RMX components for custom operation.

Deployment: We will build upon the Group Formation Protocol and the Actice Service ideas to design a deployment framework for dynamically placing RMXs within the network. Using GFP as a building block, we will define a protocol for dynamically clustering receivers into topologically sensitive groups and electing appropriate locations in the network to place RMXs.

We plan on implementing three different kinds of applications on top of this framework: an interactive conferencing tool (shared whiteboard), a bulk data transfer application (software distribution), and a periodic information application (information dissemination: stock quotes, news headlines, etc.). We plan to use these applications to evaluate the efficacy of our approach.

References

[1] 3COM CORPORATION. 3COM PalmPilot. http://www.3com.com/palm/index.html.

[2] AMIR, E., AND BALAKRISHNAN, H. An Evaluation of the Metricom Ricochet Wireless Network. Class report, UC Berkeley, May 1996.

[3] AMIR, E., AND KATZ, S. M. R. Receiver-driven Bandwidth Adaptation for Light-weight Sessions. In *Proceedings of ACM Multimedia '97* (Seattle, WA, Nov. 1997).

[4] AMIR, E., MCCANNE, S., AND KATZ, R. An Active Service Framework and its Application to Real-time Multimedia Transcoding. In *Proceedings of ACM SIGCOMM '98* (Vancouver, British Columbia, Canada, Sept. 1998).

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

[6] BARRETT, R., MAGLIO, P., AND KELLEM, D. How to Personalize the Web. In *Proceedings of CHI '97* (Atlanta, GA, Mar. 1997).

[7] BORMAN, C. M., DANZIG, P. B., HARDY, D. R., MANBER, U., AND SCHWARTZ, M. F. The harvest information discovery and access system. *Computer Networks and ISDN Systems 28* (1995), 119–125.

[8] BROOKS, C., MAZER, M. S., MEEKS, S., AND MILLER, J. Application-specific Proxy Servers as HTTP Stream Transducers. In *Proceedings of WWW-4* (Boston, MA, Dec. 1995). http://www.w3.org/pub/Conferences/WWW4/ Papers/56.

[9] BYERS, J., LUBY, M., MITZENMACHER, M., AND REGE, A. A Digital Fountain Approach to Reliable Distribution of Bulk Data. In *Proceedings of ACM SIGCOMM '98* (Vancouver, British Columbia, Canada, Sept. 1998).

[10] 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), Association for Computing Machinery.

[11] CHAWATHE, Y., AND BREWER, E. System Support for Scalable and Fault Tolerant Internet Services. In *Proceedings of Middleware '98* (Lake District, U.K., Sept. 1998).

[12] CHAWATHE, Y., FINK, S., MCCANNE, S., AND BREWER, E. A Proxy Architecture for Reliable Multicast in Heterogeneous Environments. In *Proceedings of ACM Multimedia '98* (Bristol, U.K., Sept. 1998).

[13] CLARK, D. D., AND TENNENHOUSE, D. L. Architectural Considerations for a New Generation of Protocols. In *Proceedings of ACM SIGCOMM '90* (Philadelphia, MA, Sept. 1990).

[14] CYBENKO, G., GRAY, B., KOTZ, D., RUS, D., ET AL. D'Agents: A Mobile Agent System. Submitted for publication, Feb. 1999.

[15] 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).

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

[17] Floyd, S., Jacobson, V., Liu, C., McCanne, S., and Zhang, L. A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing. In *Proceedings of ACM SIGCOMM '95* (Boston, MA, Aug. 1995), pp. 342–356.

[18] Fox, A., et al. TopGun Wingman: A Web-browser for the 3COM PalmPilot. http://www.isaac.cs.berkeley.edu/pilot/wingman/, Dec. 1997.

[19] Fox, A., Gribble, S., Brewer, E., and Amir, E. Adapting to Network and Client Variability via On-demand Dynamic Distillation. In *Proceedings of ASPLOS-VII* (Cambridge, MA, Oct. 1996).

[20] Fox, A., Gribble, S., Chawathe, Y., Brewer, E., and Gauthier, P. Cluster-based Scalable Network Services. In *Proceedings of SOSP '97* (St. Malo, France, Oct. 1997), pp. 78–91.

[21] Haeberli, P. An Image Convertor for 2 bits/pixel Grayscale Images. Private communication, 1997.

[22] Handley, M. *Session DiRectory*. University College London. Software available at ftp://cs.ucl.ac.uk/mice/sdr/.

[23] 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), Association for Computing Machinery.

[24] Independent JPEG Group Software Library. Available at: ftp://ftp.uu.net/graphics/jpeg/.

[25] InfoPad. UC Berkeley, http://infopad.eecs.berkeley.edu/.

[26] Internet Engineering Task Force. *HyperText Transfer Protocol – HTTP 1.1*, Mar. 1997. RFC-2068.

[27] Jacobson, V., and McCanne, S. *Visual Audio Tool*. Lawrence Berkeley Laboratory. Software available at ftp://ftp.ee.lbl.gov/conferencing/vat.

[28] Joseph, A., et al. A Toolkit for Mobile Information Access. In *Proceedings of 15th ACM Symposium on Operating Principles* (Copper Mountain Resort, CO, Dec. 1995).

[29] Kouvelas, I., Hardman, V., and Crowcroft, J. Network Adaptive Continuous-Media Applications through Self Organised Transcoding. In *Proceedings of the Network and Operating Systems Support for Digital Audio and Video* (Cambridge, U.K., July 1998).

[30] Lehman, L.-W. H., Garland, S. J., and Tennenhouse, D. L. Active Reliable Multicast. In *Proceedings of INFOCOM '98* (Mar. 1998).

[31] Levine, B., Lavo, D., and Garcia-Luna-Aceves, J. The Case for Concurrent Reliable Multicasting Using Shared Ack Trees. In *Proceedings of ACM Multimedia '96* (Boston, MA, Nov. 1996).

[32] Levine, B. N., Paul, S., and Garcia-Luna-Aceves, J. Organizing Multicast Receivers Deterministically According to Packet-Loss Correlation. In *Proceedings of ACM Multimedia '98* (Bristol, U.K., Sept. 1998).

[33] Li, B., Golin, M. J., Italiano, G. F., Deng, X., and Sohraby, K. On the Optimal Placement of Web Proxies in the Internet. In *Proceedings IEEE Infocom '99* (New York, NY, Mar. 1999).

[34] Liljeberg, M., et al. Enhanced Services for World Wide Web in Mobile WAN Environments. Tech. Rep. C-1996-28, University of Helsinki CS, Apr. 1996.

[35] 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.

[36] McCanne, S. A Distributed Whiteboard for Network Conferencing. Class report, UC Berkeley, May 1992.

[37] McCanne, S., and Jacobson, V. *vic*: A Flexible Framework for Packet Video. In *Proceedings of ACM Multimedia '95* (San Francisco, CA, Nov. 1995), pp. 511–522.

[38] McCanne, S., Jacobson, V., and Vetterli, M. Receiver-driven Layered Multicast. In *Proceedings of ACM SIGCOMM '96* (Stanford, CA, Aug. 1996), pp. 117–130.

[39] Montgomery, T. A Loss-tolerant Rate Controller for Reliable Multicast. Tech. Rep. NASA-IVV-97-011, West Virginal University and GlobalCast Communications Inc., Aug. 1997.

[40] National Laboratory for Applied Network Research. The Squid Internet Object Cache. http://squid.nlanr.net/.

[41] 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), Association for Computing Machinery, pp. 269–278.

[42] Raman, S. An Application-controlled Framework for Reliable Multicast Transport. Submitted for publication, Feb. 1999.

[43] Raman, S., and McCanne, S. Scalable Data Naming for Application Level Framing in Reliable Multicast. In *Proceedings of ACM Multimedia '98* (Bristol, U.K., Sept. 1998).

[44] Ratnasamy, S., and McCanne, S. Inference of Multicast Routing Trees and Bottleneck Bandwidths using End-to-end Measurements. In *Proceedings IEEE Infocom '99* (New York, NY, Mar. 1999).

[45] Ratnasamy, S., and McCanne, S. Scaling End-to-end Multicast Transports with a Topologically-sensitive Group Formation Protocol. Submitted for publication, Feb. 1999.

[46] Rizzo, L., and Vicisano, L. A Reliable Multicast Data Dristribution Protocol Based on Software FEC Techniques. In *Proceedings of HPCS '97* (Greece, June 1997).

[47] Said, A., and Pearlman, W. A. A New Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees. *IEEE Transactions on Circuits and Systems for Video Technology* (1996). Submitted for publication.

[48] Sato, Y. DeleGate Server. Documentation available at http://www.aubg.edu:8080/cii/src/delegate3.0.17/doc/Manual.txt.

[49] Schickler, M. A., Mazer, M. S., and Brooks, C. Pan-browser Support for Annotations and Other Meta-information on the World Wide Web. In *Proceedings of WWW-5* (Paris, France, May 1996). http://www5conf.inria.fr/fich_html/papers/P15/Overview.html.

[50] Shacham, N. Multipoint Communication by Hierarchically Encoded Data. In *Proceedings IEEE Infocom '92* (1992), pp. 2107–2114.

[51] Shapiro, J. M. Embedded Image Coding Using Zerotrees of Wavelet Coefficients. *IEEE Transactions on Signal Processing 41*, 12 (Dec. 1993), 3445–3462.

[52] Speakman, T., Farinacci, D., Lin, S., and Tweedly, A. *Pragmatic General Multicast (PGM) Reliable Transport Protocol*. CISCO Systems, 1998. Internet Draft.

[53] 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.

[54] Tung, T.-L. Mediaboard: A Shared Whiteboard Application for the MBone. Master's thesis, University of California, Berkeley, Dec. 1997.

[55] 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).

[56] Unwired Planet. Handheld Device Markup Language. http://www.uplanet.com/tech/products/hdml.html.

[57] Vicisano, L., Rizzo, L., and Crowcroft, J. TCP-like Congestion Control for Layered Multicast Data Transfer. In *Proceedings of INFOCOM '98* (Mar. 1998).

[58] Wallace, G. K. The jpeg still picture compression standard. *Communications of the Association for Computing Machinery 34*, 4 (1991), 31–44.

[59] Whetten, B., and Conlan, J. A Rate-based Congestion Control Scheme for Reliable Multicast. GlobalCast Communications, Oct. 1998.

[60] Yee, K. P. Shoduoka Mediator Service. http://www.shoduoka.com/.

[61] Zenel, B., and Duchamp, D. A General-purpose Proxy Filtering Mechanism Applied to the Mobile Environment. In *Proceedings of MobiCom '97* (Budapest, Hungary, Oct. 1997).