

A Proxy Architecture for Reliable Multicast in Heterogeneous Environments

Yatin Chawathe, Steve A. Fink, Steven McCanne, Eric A. Brewer
Computer Science Division, University of California at Berkeley
{yatin, sfink, mccanne, brewer}@cs.berkeley.edu

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 paper, 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, application-aware agents — or *proxies* — use detailed knowledge of application semantics to hide the effects of heterogeneity from the rest of the system. We present a general architecture for proxy-based reliable multicast called the *Reliable Multicast proXy (RMX) model* and describe a prototype implementation of an RMX for a shared whiteboard application for hand-held PDAs.

1 Introduction

The Internet multicast backbone, or Mbone [14, 13], 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 [24, 31, 21, 29, 20] 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 inher-

ent 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 on layered media [40, 36, 10, 32] or proxy-based transcoding embedded within the network [44, 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” [17] and optimize the allocation of bandwidth across flows using intelligent rate adaptation [3, 5]. Moreover, the proxy can translate the underlying

System Characteristic	Low-end	High-end
Machine	Hand-held PDA	High-end Desktop machine
CPU Speed	16MHz	300 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**

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 [15, 42, 28]. 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 paper, 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 [12], 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. We have developed 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 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 [17] 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).

In the remainder of this paper, we develop the RMX architecture. We present a generic model for RMX, then show through a specific example – the PalmPilot PDA as a “thin” whiteboard client – how the RMX proxy can be easily specialized in an application-specific manner to optimize the ADU transformations and reliability requirements for the environment at hand. Next, we describe our prototype implementation and present some preliminary evaluation results. Finally, we summarize related and future work, and present our conclusions.

2 The Reliable Multicast proXY (RMX) Model

Based on the principles outlined in the previous section, we present a generic model for reliable multicast proxies. Figure 1 shows the different components of the RMX model. The RMX splits the session into two sub-sessions — the RM session and the “proxied” session. The *RM 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 *protocol agent* serves as the interface to the proxied session.

2.1 The Abstract Model

The RM 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 communication protocol, and recovers lost data by requesting the missing data units from other members of the session. Conceptually, the RM 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 RM 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 RM agent propagates the data to the multicast session.

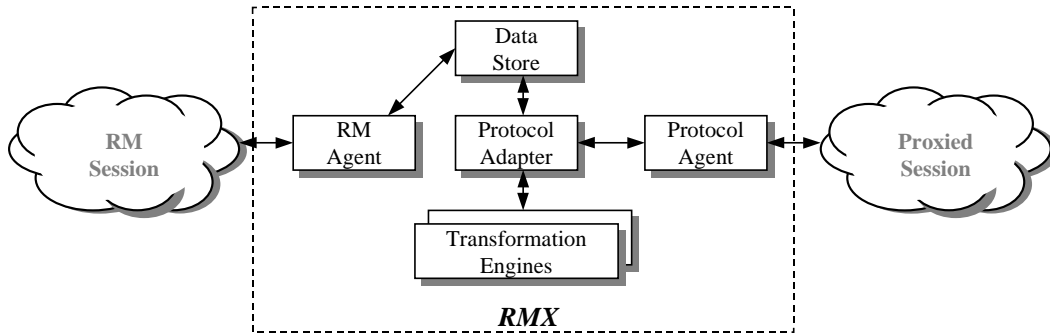


Figure 1: The RMX Model.

The data store is a “soft” copy of the reliable multicast data associated with the session. The RM 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 protocol agent provide the interface to the proxied session. The protocol 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 protocol 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.

2.2 The Protocol Adapter

The protocol adapter, which is interposed between the data store and the protocol 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.

2.2.1 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 2.2.2 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. Several end-to-end RM protocols throttle the source’s sending rate in response to network congestion [46, 33]. Though this works well in a homogeneous environment where all clients have essentially the same bandwidth, it breaks down in a heterogeneous setting. In the face of widely varying network connectivity, these traditional congestion control algorithms 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 two partitions. The proxy participates in both sessions and limits the data rate in the low-bandwidth session. Since the proxy itself can sustain the high-bandwidth flow of the original session, it does not affect the overall congestion control algorithm. 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.

2.2.2 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 progres-

sive data formats such as progressive JPEG [47] (or any of a multitude of research codecs based on sub-band transforms [37, 41] 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.

2.2.3 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 [15] works well in a high-bandwidth LAN, and Lorax [26] 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) [15], Pretty Good Multicast (PGM) [42], Reliable Multicast Transport Protocol (RMTP) [28], 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 3.4.1 while discussing our example prototype.

2.3 Locating an RMX

Having established the mechanisms that allow the RMX framework to carry out its goals, we now turn to the problem of how an RMX client locates an RMX point of contact inside the network. Additionally, the RMX service must be highly available and support a large collection of simultaneous clients. It should automatically configure itself to the clients' needs and recover from system failures in a graceful manner.

These problems, for the most part, are orthogonal to the design of the RMX model itself. Fortunately, two research projects at UC Berkeley have developed frameworks based on clusters of work-stations [6] for scalable, available, fault tolerant infrastructure services. One such framework is the SNS (Scalable Network Service) architecture [11, 18] developed by the GloMop research project at UC Berkeley. The SNS framework consists of a *front end* that provides the interface to the rest of the system. *Workers* are the task engines that satisfy the actual requests. Figure 2 shows the components of the framework as tailored to our application. Clients connect to a front end that provides a level of indirection for locating the RMX. The RMX is implemented as a "worker" in the infrastructure framework. A client announces its interest in a session to the front end, which locates an appropriate proxy (if one already exists) or starts up a new RMX worker on behalf of the client and returns its location to the client.

The MASH research project at UC Berkeley is investigating an alternative approach to the problem of infrastructure services that relies on IP multicast as the rendezvous mechanism between clients and services. The Active Service (AS1) Framework [4] uses a collection of *host managers* that listen to a well-known multicast channel. Clients broadcast their interest in a session on this channel and one of the host managers traps these requests and starts a new proxy if required.

3 A Prototype for an RMX

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.

3.1 Shared Whiteboard Proxy for PDAs

The original electronic shared whiteboard application, *wb* [29], 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* [43]. 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

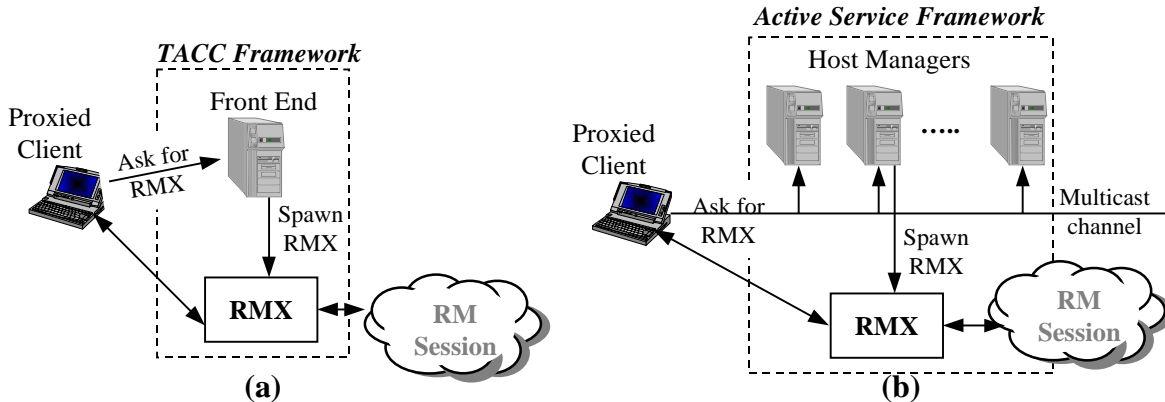


Figure 2: **The Architecture of the RMX Infrastructure:** (a) The SNS Architecture, (b) The AS1 Architecture.

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 application is built on top of an implementation of the Scalable Reliable Multicast (SRM) framework presented in [15].

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. However, such a client does enable a variety of interesting applications:

- *Meeting support:* Mediaboard-equipped PDAs can be used as collaboration tools to annotate or write on a shared screen that may be projected into the room using an overhead projector or a LiveBoard [48].
- *Smart cell phones:* Smart phones, such as the Nokia 9000 [35] are another interesting example of PDAs. Such “smart” phones can enhance communication between people: for example, a person trying to locate a friend’s house could use a shared map on a small whiteboard on the phone’s screen to interact with the friend.

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.

3.2 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.

3.3 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 protocol agent for the mediaboard proxy 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 protocol 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 RM 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 RM 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 [15]. 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 RM 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 protocol agent receives data from the PDA client, it hands the data over to the protocol



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

adapter which in turn adds mediaboard commands to the data store. The RM 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.

3.4 The Protocol Adapter

The protocol adapter for the mediaboard proxy implements all three aspects of adaptation discussed in section 2.2.

3.4.1 Protocol Conversion

The protocol 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.

3.4.2 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 bit map 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 [19]. 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.

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.

3.4.3 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.

4 Implementation Status

We used the MASH toolkit [30] as our development platform. This is a Tcl/C++-based programming framework for multimedia networking applications developed by the MASH research project at UC Berkeley. The reliable multicast transport protocol that we used was SRM [15]. The RM agent is sub-classed from the SRM objects that are part of the MASH toolkit, while the protocol adapter is derived from the mediaboard objects in the toolkit.

Data size (bytes)	Mediaboard Protocol	Simplified PDA Protocol
Image	52651	4704
Free-hand drawing	11004	812 (max compression) 10580 (max interactivity)
Arrow	84	76

Table 2: Examples of bandwidth savings with ALF-based RMX-PDA protocol

By completely exposing the mediaboard protocol to the RMX, we were able to fully optimize it for the PalmPilot. Table 2 shows the bandwidth savings that are possible with ALF-based adaptation. As expected, through lossy compression, the proxy dramatically reduces the number of bytes that need to be transmitted to the PDA by over a factor of 10 at the PDA's typical zoom level of 33%¹. Freehand drawings show an interesting tradeoff between bandwidth utilization and interactivity. For maximum interactivity, the desktop mediaboard protocol sends each line segment of a free-hand sketch as a separate packet as soon as it is generated. To avoid this overhead, the RMX intelligently groups these individual line segments and sends a coalesced draw-op to the PalmPilot. The example in Table 2 consists of a total of 131 individual line segments. The RMX can once again achieve savings of up to a factor of 10, albeit at some loss of interactive drawing. A final data point is arrows, where the actual data transmitted to the PalmPilot is more than the original data in the mediaboard command – the RMX computes the arrowheads for the client and sends the coordinates as part of the draw-op. Yet, the packet size is smaller for the PDA protocol simply due to the elimination of costly SRM headers.

As part of our evaluation, we would like to measure the effects of the RMX on the retransmission algorithms in the reliable multicast session. With high degrees of heterogeneity, we expect to see a drop in goodput² as the bandwidth-gap between the well-connected and poorly-connected clients increases. As the source continues to transmit at a high rate, packets will be dropped in low-bandwidth areas, thus resulting in retransmissions and a drop in goodput. By placing an RMX between the regions of poor connectivity and the rest of the session, the goodput in the main session can be kept high, while using the techniques outlined in sections 2 and 3 to limit the data rate in the low-bandwidth regions. However, due to bugs in the implementation of the SRM library that we used, we were unable to perform these experiments. We hope to evaluate the effectiveness of the RMX once these errors have been fixed.

5 Related Work

The notion of proxies as intermediaries between clients and servers is not new. Numerous proxy mechanisms have been proposed for HTTP [23]. 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 [38], Kanji-to-GIF transformation [49], application-level stream transducing [9, 39], and personalized agent services for web browsing [7]. It has been used to hide the effects of error-prone and low-bandwidth wireless links [17, 27]. Bruce Zenel [50] 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 [8, 34] 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 [22] used an extreme approach with proxies: move all intelligence into the infrastructure and use the PDA simply as a dumb terminal.

¹The PalmPilot screen is approximately one-third the size of desktop mediaboards.

²Goodput is the ratio of useful bytes to total bytes transmitted.

Partitioning of application complexity between the client and infrastructure has been used in other situations. A related project, TopGun Wingman [16], uses an infrastructure proxy to support a simplified web-browser on the PalmPilot. [45] have proposed the use of a simplified document format (HDML) to reduce the complexity of PDA application. The Rover system [25] 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.

6 Future Work

Our prototype RMX examines the case of client device heterogeneity. We plan on implementing an RMX for an effective rate-limiting protocol that deals with network heterogeneity. The Computer Science Division at UC Berkeley is experimenting with conducting classes over the MBone. Such MBone broadcasts are limited by wide-area MBone bandwidth. We would like to provide high bandwidth content to students within the campus, while, at the same time, multicasting lower bandwidth, lower quality data to the rest of the MBone. A rate-limiting RMX can be used to complement the similar functionality provided by audio and video gateways for audio/video data.

An interesting issue that arises with RMXs in the network is the problem of placing them intelligently and dynamically throughout the network. Our current prototype relies on the existence of a well-known service cluster that supports the RMX. We plan to investigate more dynamic placement algorithms for such agents.

Currently the prototype implementation does not support all data types associated with the mediaboard. We plan on implementing transformation engines for handling postscript data. The prototype client is rudimentary, primarily due to the limitations of the drawing APIs. We plan to extend the drawing functions to allow us to present a more realistic drawing canvas to the user.

We need to analyze our current implementation to extract out the core reusable abstractions. The transformation engines, the RM agent and the data store in the prototype are reasonably modular, but the rest of the components may have to be re-organized for reuse. This will allow us to construct a framework for other heterogeneous, reliable groupware applications.

7 Summary

In this paper, we presented a solution for adapting reliable multicast sessions to heterogeneous environments. We have designed an abstract model for reliable multicast proxies based upon application level framing that allows us to specialize the protocol framework for the environment at hand. To demonstrate the efficacy of our framework, we developed a prototype RMX for a real application (i.e., the MASH mediaboard) running an impoverished hardware client (i.e., the PalmPilot). Our prototype demonstrates the power of ALF to enable the proxy to optimize performance of the end-client by a tight integration at all levels between the proxy and the client.

Acknowledgments

We would like to thank Teck-Lee Tung and Suchitra Raman for helping us navigate through the SRM and MASH mediaboard protocol and implementation. Ian Goldberg gave us a number of pointers to programming the PalmPilot and updated the PalmPilot compiler to effectively handle C++. Finally, we would like to thank all our colleagues for their inputs and suggestions for the drafts of this paper.

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] ANDERSON, T. E., CULLER, D. E., PATTERSON, D. A., AND THE NOW TEAM. A Case for Networks of Workstations: NOW. In *Principles of Distributed Computing* (Aug. 1994).
- [7] BARRETT, R., MAGLIO, P., AND KELLEM, D. How to Personalize the Web. In *Proceedings of CHI '97* (Atlanta, GA, Mar. 1997).
- [8] 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.
- [9] 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>.
- [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] 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).
- [13] 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).
- [14] DEERING, S. E. *Multicast Routing in a Datagram Internetwork*. PhD thesis, Stanford University, Dec. 1991.
- [15] 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.
- [16] FOX, A., ET AL. TopGun Wingman: A Web-browser for the 3COM PalmPilot. <http://www.isaac.cs.berkeley.edu/pilot/wingman/>, Dec. 1997.

- [17] 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).
- [18] 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.
- [19] HAEBERLI, P. An Image Converter for 2 bits/pixel Grayscale Images. Private communication, 1997.
- [20] HANDLEY, M. *Session DiRectory*. University College London. Software available at <ftp://cs.ucl.ac.uk/mice/sdr/>.
- [21] 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.
- [22] INFOPAD. UC Berkeley, <http://infopad.eecs.berkeley.edu/>.
- [23] INTERNET ENGINEERING TASK FORCE. *HyperText Transfer Protocol – HTTP 1.1*, Mar. 1997. RFC-2068.
- [24] JACOBSON, V., AND MCCANNE, S. *Visual Audio Tool*. Lawrence Berkeley Laboratory. Software available at <ftp://ftp.ee.lbl.gov/conferencing/vat>.
- [25] 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).
- [26] 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).
- [27] 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.
- [28] 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.
- [29] MCCANNE, S. A Distributed Whiteboard for Network Conferencing. Class report, UC Berkeley, May 1992.
- [30] MCCANNE, S., ET AL. Toward a Common Infrastructure for Multimedia-Networking Middleware. In *Proceedings of the Seventh International Workshop on Network and OS Support for Digital Audio and Video* (St. Louis, Missouri, May 1997), Association for Computing Machinery.
- [31] 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.
- [32] MCCANNE, S., JACOBSON, V., AND VETTERLI, M. Receiver-driven Layered Multicast. In *Proceedings of ACM SIGCOMM '96* (Stanford, CA, Aug. 1996), pp. 117–130.
- [33] MONTGOMERY, T. A Loss-tolerant Rate Controller for Reliable Multicast. Tech. Rep. NASA-IVV-97-011, West Virginia University and GlobalCast Communications Inc., Aug. 1997.
- [34] NATIONAL LABORATORY FOR APPLIED NETWORK RESEARCH. The Squid Internet Object Cache. <http://squid.nlanr.net/>.
- [35] NOKIA SYSTEMS. Communicator 9000 Press Release. Available at <http://www.forum.nokia.com/nf/products/communicators/9000/index.html>.
- [36] PASQUALE, J. C., POLYZOS, G. C., ANDERSON, E. W., AND KOMPPELLA, V. P. Filter Propagation in Dissemination 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.
- [37] 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.
- [38] SATO, Y. DeleGate Server. Documentation available at <http://www.aubg.edu:8080/cii/src/delegate.3.0.17/doc/Manual.txt>.
- [39] 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.
- [40] SHACHAM, N. Multipoint Communication by Hierarchically Encoded Data. In *Proceedings IEEE Infocom '92* (1992), pp. 2107–2114.
- [41] SHAPIRO, J. M. Embedded Image Coding Using Zerotrees of Wavelet Coefficients. *IEEE Transactions on Signal Processing* 41, 12 (Dec. 1993), 3445–3462.
- [42] SPEAKMAN, T., FARINACCI, D., LIN, S., AND TWEEDLY, A. *Pragmatic General Multicast (PGM) Reliable Transport Protocol*. CISCO Systems, 1998. Internet Draft.
- [43] TUNG, T.-L. Mediaboard: A Shared Whiteboard Application for the Mbone. Master's thesis, University of California, Berkeley, Dec. 1997.
- [44] 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).
- [45] UNWIRED PLANET. Handheld Device Markup Language. <http://www.uplanet.com/tech/products/hdml.html>.
- [46] VICISANO, L., RIZZO, L., AND CROWCROFT, J. TCP-like Congestion Control for Layered Multicast Data Transfer. In *Proceedings of INFOCOM '98* (Mar. 1998).
- [47] WALLACE, G. K. The jpeg still picture compression standard. *Communications of the Association for Computing Machinery* 34, 4 (1991), 31–44.
- [48] XEROX LIVEWORKS. The LiveBoard Interactive Meeting System. <http://www.liveworks.com/liveboard/index.html>.
- [49] YEE, K. P. Shoduoka Mediator Service. <http://www.shoduoka.com/>.
- [50] 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).