# Scalable Resilient Media Streaming

Suman Banerjee, Seungjoon Lee, Bobby Bhattacharjee, Aravind Srinivasan, Ryan Braud
Department of Computer Science
University of Maryland, College Park, MD 20742, USA
Emails: {suman, slee, bobby, srin}@cs.umd.edu, bigman@wam.umd.edu

CS-TR 4482 and UMIACS TR 2003-51

***Abstract*— We present a low-overhead media streaming system, called SRMS (Scalable Resilient Media Streaming) that can be used to scalably deliver streaming data to a large group of receivers. SRMS uses overlay multicast for data distribution to a large group of users. SRMS leverages a probabilistic loss recovery technique to provide high data delivery guarantees even under large network losses and overlay node failures. Through detailed analysis in this paper, we show that this loss recovery technique (and consequently SRMS) has efficient scaling properties — the overheads at each overlay node asymptotically decrease to zero with increasing group sizes.**

**We also present a detailed description of the SRMS architecture. The clients in the SRMS system are able to interoperate with existing media streaming servers that use RTP for data transport. One of the interesting features of SRMS is that it can simultaneously support clients with disparate access bandwidths. It enables the necessary bandwidth adaptations using standard Realtime Transport Protocol (RTP) mechanisms, e.g. RTP translators.**

**We have implemented and evaluated the SRMS system in detail on an emulated network as well as on a wide-area testbed with up to 128 clients. Our results show that clients using SRMS achieve high ($>97\%$) data delivery ratios with low overheads ($<5\%$) even for very high failure rates (upto five per minute).**

## I. Introduction

We present SRMS (Scalable Resilient Media Streaming): a system for scalable delivery of streaming media data to a large number of receivers using application-layer multicast. The design of SRMS is independent of any specific application-level multicast delivery protocol or media format. Further, SRMS incorporates a delivery protocol-independent loss recovery technique called Probabilistic Resilient Multicast (PRM [4]), which permits high data delivery ratios even under high network losses and node failures. Lastly, the SRMS architecture logically admits media transcoding for handling clients with disparate access bandwidths. In this paper, we present a full analysis of the SRMS resilience scheme (PRM), and analytically show that SRMS can achieve provably high data delivery ratios with overheads that asymptotically tend to zero. We also describe a full implementation of the SRMS architecture, including an implementation of PRM, and wide-area deployment with over one hundred simultaneous clients. We believe this paper presents the first reported implementation experiments that explicitly addresses the issues of data resilience with large application-layer multicast groups.

The data delivery mechanism of SRMS is based on overlay multicast (also known as application-layer multicast) [9], [12], [3], [35], [7], [25], [17]. Unlike native multicast where data packets are replicated at routers inside the network, in application-layer multicast data packets are replicated at end hosts. Logically, the end-hosts form an overlay network, and the goal of application-layer multicast is to construct and maintain an efficient overlay for data transmission. The eventual data delivery path in application-layer multicast is an overlay tree. While network-layer multicast makes the most efficient use of network resources, its limited deployment in the Internet makes application-layer multicast a more viable choice for group communication over the wide-area Internet.

The SRMS system can be implemented with any application-layer multicast protocol to construct the underlying data delivery paths. In our current implementation we chose the NICE application-layer multicast protocol [3]. Our choice was based on the following reasons: (1) NICE achieves good delivery ratios for a best-effort scheme [3], (2) NICE has a scalable construction and therefore is suitable for large application groups, and (3) the source-code for NICE is publicly available.

A key challenge in building a resilient media streaming system based on application-layer multicast is to provide fast data recovery when overlay node failures partition data delivery paths. Overlay nodes are processes on regular end-hosts which are potentially more susceptible to failures than the routers. Each such failure of a non-leaf overlay node causes data outage for nodes downstream until the data delivery tree is reconstructed. Losses due to overlay node failures are more significant than regular packet losses in the network and may cause data outage on the order of tens of seconds (e.g. the Narada application-layer multicast protocol [9] sets default timeouts between 30-60 seconds). Using PRM, the probabilistic loss recovery technique, SRMS is able to achieve high data delivery ratios even in scenarios with frequent overlay node failures.

### *Key Contributions*

The main contributions of this paper are:

- We describe SRMS, the first implementation of a resilient video delivery system based on application-layer multicast. Our implementation enables wide-area streaming of multimedia to large groups. Apart from good resilience properties, the proposed SRMS system also enables selective data rate adaptation for clients with disparate access bandwidths.
- Detailed analysis of the probabilistic loss recovery scheme (PRM). We present a full analysis of PRM and derive the necessary and sufficient conditions for PRM which

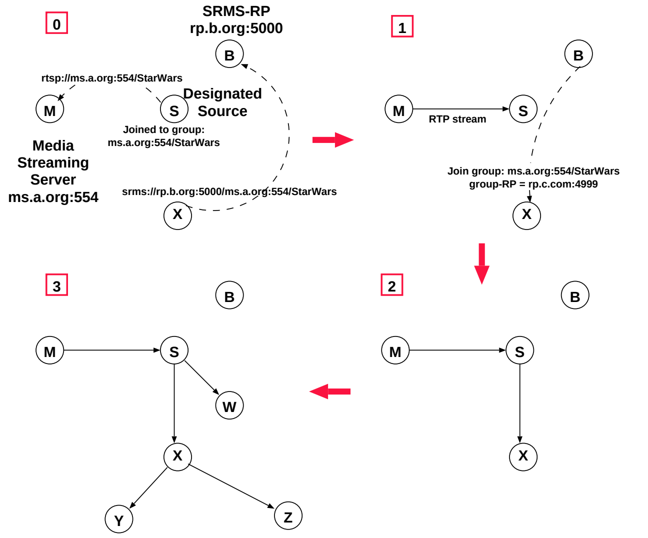


Fig. 1. Architectural overview of the SRMS System.

will allow the control overheads at the group members to asymptotically decrease to zero with increasing group sizes. These results are a significant improvement over the analysis presented in [4], where a simplified version of the scheme was analyzed (and the results were correspondingly weaker). We compare these new analytic results to prior work [4] in Section IV.

*Roadmap*

The rest of the paper is organized as follows: In Section II we present an architectural overview of the SRMS system. In Section III we summarize PRM [4], the probabilistic loss recovery technique used in SRMS. We present a detailed analysis of the full PRM scheme in Section IV. In Section V we describe the implementation of the system. We have performed detailed experiments with the SRMS system on a publicly available emulation network environment as well as a wide-area testbed and we report on experimental results in Section VI. In Section VII we describe some related work and present our conclusions in Section VIII.

## II. SRMS Architecture

A SRMS system comprises of the SRMS Rendezvous Point (SRMS-RP) and a set of receivers or clients. There is a separate multicast group for each media stream that is served by the SRMS system. In order to receive the relevant media stream, a receiver has to join the appropriate application-layer multicast group. One of the receivers in the group serves as the multicast source. The source is responsible for acquiring the media from the streaming server and forwarding it to the remaining group members. The media streaming server need not be aware of the multicast delivery tree to which the stream eventually gets forwarded to. This construction allows any media streaming server to interoperate with the SRMS system. In our implementation, the source uses the Real-time Streaming Protocol (RTSP) [29] to initiate reception of the media stream from the streaming server. RTSP is an application-level protocol that can be used to control the delivery of either a single or several time-synchronized media streams such as audio and video. RTSP does not typically deliver the data itself. In most cases, the Real-time Transport Protocol (RTP) [28], is used for data delivery. We defer the discussion of specific protocol issues to Section V.

In SRMS any client can potentially operate as the source to the multicast group. However, in a typical deployment, we expect a specific host, designated by the service provider, to serve as the source (which will be co-located with the media streaming server and/or the SRMS-RP). In absence of a designated source, the SRMS-RP will coordinate the choice of the source from among the existing clients. We do not explore the complexities of such a choice in the paper and focus only on the designated source case.

In Figure 1 we show a typical sequence of operations of the SRMS system. We define a new application protocol, `srms`, which is used for communication between the media clients and the SRMS-RP. A media stream is uniquely identified by a SRMS URL which consists of the `srms` protocol identifier; the hostname, port number pair of the SRMS-RP; the hostname, port number pair of the media streaming server and the media stream identifier. In our example the joining client, $X$, therefore, makes a request for the URL: `srms://rp.b.org:5000/ms.a.org:554/StarWars`. This identifies `rp.b.org:5000` as the SRMS-RP, `ms.a.org:554` as the media streaming server and the `StarWars` media stream. This is shown in Panel 0, Figure 1. (The syntax is intentionally similar to the Real-time Streaming Protocol (RTSP) [29] URL format.)

The SRMS-RP instructs $X$ to join the application-layer multicast group identified as `a.org:554/StarWars` (Panel 1). Application-layer multicast protocols typically have a group Rendezvous Point (group-RP) which is responsible for bootstrapping the join procedure. The SRMS-RP conveys this group-RP information to the client, $X$. Note that the SRMS-RP and the group-RP are two logically separate entities, but will likely be co-located on the same host. Decoupling these two entities allows us to decouple SRMS from undue dependence on any specific application-layer multicast protocol.

In our example, the designated source for this media stream, $S$, is already joined to the application-layer multicast group for the requested media stream. It has also contacted the media streaming server, $M$, using the RTSP URL: `rtsp://a.org:554/StarWars` (Panel 1) and is subsequently receiving the media stream from the server. On receiving the media stream, $S$, multicasts it on the overlay tree of the corresponding application-layer multicast group. Therefore, when $X$ joins the group, it starts receiving the media stream from $S$. Subsequently when other clients, $W, Y$ and $Z$ request the same media stream, they eventually join the same application-layer multicast group and data forwarded by $S$ reaches all these clients via application-layer multicast.

## III. Summary of PRM

Probabilistic Resilient Multicast (PRM) [4] is a probabilistic loss recovery technique which is used to provide resilience in SRMS.[1] In this section, we present a summary of this scheme.

For a multicast data delivery system based on overlays, data losses at the receivers can happen due to the following two reasons: (1) network level data packet loss on some overlay hop (e.g. due to congestion), and (2) failure of an intermediate overlay node on the overlay distribution tree. Recovery from network level data losses is relatively easy and can be handled using retransmission-based or FEC-based mechanisms. However, for an application-layer multicast delivery system quick recovery of data lost due to failure of overlay nodes poses a significant challenge. Overlay nodes are regular processes on end-hosts and are potentially more failure-prone than network routers. A failure of an intermediate overlay node may cause data outage in the order of tens of seconds (e.g. the Narada application-layer multicast protocol [9] sets default timeouts for failure detection between 30-60 seconds). PRM uses an efficient probabilistic construction to recover from such losses. In this section we provide an overview of the PRM technique.

PRM consists of two components:

- A proactive component called *Randomized forwarding* in which each overlay node chooses a small number of other overlay nodes uniformly at random and forwards data to each of them with a low probability (e.g. 0.01-0.03). This randomized forwarding technique operates in conjunction with the usual data forwarding mechanisms along the tree edges, and may lead to a small number of duplicate packet deliveries. Such duplicates are detected and suppressed using sequence numbers. *In this paper we show that this randomized forwarding scheme scales well with increasing group sizes, i.e. the overheads of the scheme required to guarantee successful data delivery with a high probability asymptotically decreases (to zero) with asymptotic increase in group sizes.*
- A reactive mechanism called *Triggered NAKs* to handle data losses due to link errors and network congestion.

We briefly summarize each of these components in turn.

### A. Randomized forwarding

In randomized forwarding, each overlay node, with a small probability, proactively sends a few extra transmissions along randomly chosen overlay edges. Such a construction interconnects the data delivery tree with some cross edges and is responsible for fast data recovery in PRM.

We explain the details of proactive randomized forwarding using the example shown in Figure 2. In the original data delivery tree (Panel 0), each overlay node forwards data to its children along its tree edges. However, due to network losses on overlay links (e.g. $\langle A, D\rangle$ and $\langle B, F\rangle$) or failure of overlay nodes (e.g. $C$, $L$ and $Q$) a subset of existing overlay nodes do not receive the packet (e.g. $D, F, G, H, J, K$ and $M$). This is remedied as follows. When any overlay node receives the first copy of a data packet, it forwards the data along all other tree edges (Panel 1). It also chooses a small number ($r$) of other overlay nodes and forwards data to each of them with a small probability, $\beta$. For example node $E$ chooses to forward data to two other nodes using cross edges $F$ and $M$. Note that as a consequence of these additional edges some nodes may receive multiple copies of the same packet (e.g. node $T$ in Panel 1 receives the data along the tree edge $\langle B, T\rangle$ and cross edge $\langle P, T\rangle$). Therefore each overlay node needs to detect and suppress such duplicate packets. Each overlay node maintains a small duplicate suppression cache, which temporarily stores the set of data packets received over a small time window. Data packets that miss the latency deadline are dropped. Hence the size of the cache is limited by the latency deadline desired by the application. In practice, the duplicate suppression cache can be implemented using the playback buffer already maintained by streaming media applications. It is easy to see that each node on average sends or receives upto $1 + \beta r$ copies of the same packet. The overhead of this scheme is $\beta r$, where we choose $\beta$ to be a small value (e.g. 0.01) and $r$ to be between 1 and 3. In the analysis that we introduce in Section IV, we show that for increasing group sizes if the destinations of these cross edges are chosen uniformly at random, then each overlay node successfully receives the data packets with a high probability for even very low values of $\beta$.

### B. Triggered NAKs

This is the reactive component of PRM. It assumes that the application source identifies each data unit using monotonically increasing sequence numbers. An overlay node can detect missing data using gaps in the sequence numbers. This information is used to trigger NAK-based retransmissions. This technique has been applied for loss repair in RMTP [24].

## IV. Evaluation of PRM

A key component of the PRM scheme is the randomized forwarding technique which achieves high delivery ratios in spite of a large number of overlay node/link failures. In this section we present our analysis of this scheme.

We first informally explain the intuition as to why such a simple randomized forwarding scheme is so effective in achieving high data delivery ratios inspite of large number of failures on the overlay. Consider the example shown in Figure 3, where a large fraction of the nodes have failed in the shaded region. In particular, the root of the sub-tree, node $A$, has also failed. So if no forwarding is performed along cross edges, the entire shaded sub-tree is partitioned from the data delivery tree. No overlay node in this entire sub-tree would get data packets till the partition is repaired. However using randomized forwarding along cross edges a number of nodes from the unshaded region will have random edges into the shaded region as shown ($\langle M, X\rangle, \langle N, Y\rangle$ and $\langle P, Z\rangle$). The overlay nodes that receive data along such randomly chosen cross edges will subsequently forward data along regular tree edges and any chosen random edges. Since the cross edges are chosen uniformly at random, a large subtree will have a higher probability of cross edges being incident on it. Thus as the size of a partition increases, so does

[1] PRM is described in [4] which is currently unpublished and not publicly available. Therefore we present a summary of PRM in this paper for the sake of completeness.

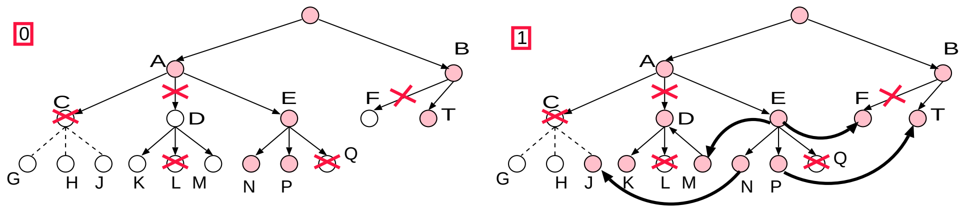


Fig. 2. The basic idea of the PRM scheme. The circles represent the overlay nodes. The crosses indicate link and node failures. The arrows indicate the direction of data flow. The curved edges indicate the chosen cross overlay links for randomized forwarding of data.

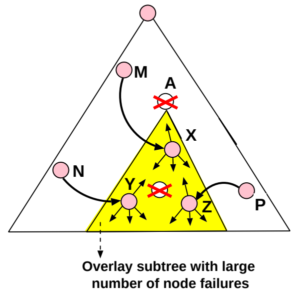


Fig. 3. Successful delivery with high probability even under high failure rate of overlay nodes.

its chance of repair using cross edges. Now we will formally state and prove the specific bounds of this scheme.

Recall from Section III that the per-node overhead of PRM is $\beta r$. We will now primarily be concerned with the case of low overhead; in particular, the case where the overhead is (much) smaller than 1. Thus, up to Section IV-A, we will consider the case where $r = 1$: here, each node does random forwarding to just one node, with a probability of $p = \beta$. However, all of our results will also hold for $r$ being arbitrary. In this paper, we prove that even if the probability of random forwarding $p$ is made an *arbitrarily small* positive constant (i.e., even if the data overhead $\beta r = pr = p$ is made negligible), the scheme can be designed so that almost all surviving overlay nodes get the data with high probability. In particular, the system scales: as the number $n$ of nodes increases, our probability of successful data delivery tends to 1.

We start with some notation and assumptions.

(A1) All nodes at the same level of the tree have the same number of children. The total number of nodes is denoted by $n$.

(A2) There are parameters $\epsilon$ and $\delta$ such that the probability of any given node failing is at most $\epsilon$, and the probability of any given link failing is at most $\delta$. We only require that $\epsilon$ and $\delta$ be bounded away from 1: e.g., we could have $\epsilon, \delta \leq 0.5$. (Indeed, a multicast tree composed of elements that may fail with more than $50\%$ probability, is in effect useless; in practice, we expect $\epsilon$ and $\delta$ to be close to zero.) The failure events are all independent.

We next present a theorem which deals with the asymptotic regime where $n$ is large. We then discuss a "tree augmentation" technique and general optimality of our results in Section IV-A. We complement these in Section IV-B with simulation results for the "non-asymptotic" regime. of size 10,000.

*Theorem IV.1: Let the probability of random forwarding $p$ be an arbitrary positive constant (i.e., it can be arbitrarily small). Then, there is a constant $C > 0$ such that the following holds. Suppose every non-leaf node has at least $C \log n$ children. Then, with probability tending to 1 as $n$ increases, the following two claims hold simultaneously:*

(i) *All the non-leaf nodes that did not fail, successfully get the data.*

(ii) *At least an $(1-\epsilon)\cdot(1-\delta)\cdot(1-g(n))$ fraction of the leaf nodes that did not fail, successfully get the data; here $g(n)$ denotes a negligibly small quantity, which tends to 0 as $n$ increases.*

*Proof:* See Appendix. ■

To take a realistic example, we consider the case when the end-to-end losses on overlay links is 2%, and simultaneous number of overlay nodes failures is 1% of the group (for group of size 10,000 this translates to 100 simultaneous failures). Theorem IV.1 states that with a high probability all surviving non-leaf overlay nodes and at least 97% of the surviving leaf nodes continue to receive data packets using the existing overlay data paths and random edges.

It is, in fact, possible to increase the delivery to leaf-nodes to to increase the delivery to leaf-nodes to an arbitrarily high value (e.g. 99.98%) using a "tree augmentation" scheme described next.

### A. Tree-augmentation extensions to PRM

If we desire, for some confidence parameter $\phi$, that at least a $(1-\phi)$–fraction of the surviving leaves get the data with high probability (in addition to item (i) of Theorem IV.1), then it suffices to augment the tree as follows: each leaf connects to $1 + \lfloor \frac{\log(1/\phi)}{\log(1/\epsilon)} \rfloor$ randomly chosen nodes, and gets the data from any one of them that has received the packet. In fact, this amount of overhead is necessary for *any* protocol, if we require a $(1-\phi)$–fraction of the surviving leaves to get the data with high probability. In the Appendix that our protocol's lower bounds on the fraction of surviving leaves getting the data — both with and without the random augmentation for leaves — are optimal. Also in the Appendix we show that the logarithmic degree-requirement of Theorem IV.1 is both necessary and sufficient if we desire a low overhead. Specifically if the degrees, e.g., of the parents of the leaves are only some small constant times $\log n$, then in fact a large number of the leaves' parents will fail to get the data, with probability tending to 1 as $n$ increases.

**Comparison with prior work [4].** Theorem IV.1 shows that as long as the node-degrees are at least logarithmic in $n$, the tree is highly resilient to node- and link-failures, even with arbitrarily low data overhead. The prior work [4] only analyzes certain restricted versions of PRM, where randomly forwarded data can continue to be forwarded only by further random forwarding. This leads to much larger overheads than we achieve in Theorem IV.1. For instance, suppose we require at least a

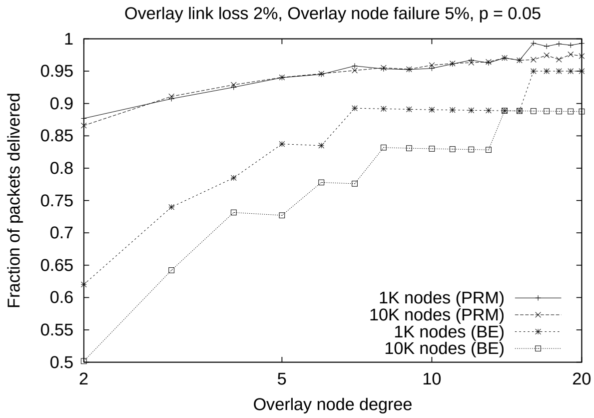


Fig. 4. Variation of data delivery ratio with overlay node degree.

$(1-\phi)$–fraction of the surviving nodes to get the data with high probability; then, the analyses of the restricted protocols in [4] require an overhead that grows proportional to $1/\phi$. Using the same example as before, the analysis in [4] requires an overhead greater than 300% to achieve successful delivery to 97% of the nodes, while the analysis presents in this paper illustrates that the overheads are, in fact, negligible.

The analysis presented in this paper shows that all non-leaf nodes and 97% of leaf nodes successfully get the data when the overlay link loss is 2% and 1% of the group fails. Through the tree augmentation extension to PRM (described in Section IV-A), we showed that it is possible to increase the delivery to leaf-nodes to an arbitrarily high value (e.g. 99.98%). If we require a $(1-\phi)$–fraction of the surviving nodes to get the data with high probability, the tree augmentation scheme in Section IV-A can achieve it by incurring $1 + \lfloor \frac{\log(1/\phi)}{\log(1/\epsilon)} \rfloor$, overhead. Therefore for the $1\%$ node failure rate case, the tree augmentation scheme will guarantee that 99.98% of all nodes (including leaf nodes) will successfully get the data packets with an overhead bounded by the constant two. In contrast, the analysis in [4] provides a overhead factor of $> 10,000$ to meet the same data delivery ratio.

Thus, by analyzing the full protocol, we are able to contribute the following improvements. First, our overhead analysis is a significant improvement over that of [4]; this is especially so in the realistic case where $\epsilon$ and $\delta$ are small and the overhead can be made arbitrarily small. Furthermore, our overhead of $1 + \lfloor \frac{\log(1/\phi)}{\log(1/\epsilon)} \rfloor$ in the tree augmentation algorithm is also optimal, as shown in the Appendix. Finally in the Appendix we prove that that a tree-degree that grows at least as fast as $\log n$ is both necessary and sufficient for high delivery ratios.

### B. Expected Scaling Behavior for Finite Groups

In the analysis of the PRM scheme (described above), we have explored its asymptotic behavior with corresponding increase in the group size. Through simulations we now show that for finite group sizes the randomized forwarding scheme achieves very high data delivery ratios for very low tree degrees. In these idealized simulations, we assume that there exists some application-layer multicast protocol which constructs and maintains a data delivery tree. When data packets are sent on this tree, a random subset of the overlay nodes fail simultaneously. The failed subset is chosen independently for each data packet sent. Additionally, data packets also experience network layer losses on overlay links. Consider a regular tree, where all non-leaf nodes have the same degree. From the analysis we can intuitively expect that as the degree of the tree increases, so does the data delivery ratio. In Figure 4 we illustrate how the data delivery ratio for the non-leaf nodes of an overlay tree improves with increase in degree. We considered two different tree sizes — 1000 nodes and $10,000$ nodes. In this example, we assume that for each overlay link experiences a loss rate of $2\%$ and the node failure rate is $5\%$ (which implies $50$ simultaneous failures for a 1000-node tree and $500$ simultaneous failures on a $10,000$-node tree). Such failure rates are very high by the usual Internet standards. The randomized forwarding probability $p$ is chosen to be $0.05$ (i.e. the data overhead is $5\%$). We can see that even under such adverse conditions, the randomized forwarding technique achieves data delivery ratio of about $93\%$ even with a tree degree of 5; the delivery ratio exceeds $95\%$ when the degree is made 10, quickly approaching 1 as the degree is increased further. The results for the leaf nodes in practice, are close to those of the non-leaf nodes.

## V. Implementation of the SRMS System

We now describe the implementation of the SRMS system. SRMS consists of the SRMS-RP and a set of receivers or clients. The SRMS-RP gets the initial media stream request from the clients using the `srms` protocol. On receiving each request, the SRMS-RP directs the client to the appropriate application-layer multicast group on which it can receive the media stream. The implementation of the SRMS-RP is relatively straightforward. Therefore in this section we focus on the client implementation.

### A. NICE and PRM

SRMS uses NICE application-layer multicast [3] as the underlying data delivery path. Additionally we also implement the PRM extensions as described in Section III. One of the key requirements of PRM is the ability to forward data to a few other overlay nodes, chosen uniformly at random. In practice, however, we observed that any node, $X$, the benefits of the random edges can be improved if the random choices are made from among a set of other nodes, $\mathcal{S}$, such that the number of losses shared between $X$ and nodes in $\mathcal{S}$ is minimal. Therefore in our implementation of PRM, we let each overlay node to periodically discover a set of random other nodes on the overlay and evaluates the number of losses that it shares with these random nodes. Each node subsequently performs random edge forwarding to those nodes with which it shares the minimum number of losses.

In order to locate a random overlay node, we implemented a random node discovery mechanism in NICE using a random walk on the overlay. The discovering node transmits a *Discover* message with a *time-to-live* (TTL) field to its parent on the tree. The message is randomly forwarded from neighbor to neighbor, without re-tracing its path along the tree and the TTL field is decremented at each hop. The node at which the TTL reaches zero is chosen as the random node. If alternately an overlay construction protocol like Narada [9] was used then no such additional node discovery mechanism would be needed. This is because in Narada each node maintains state information about all other nodes.

To implement triggered NAKs, each overlay node, $x$, piggybacks a bit-mask with each forwarded data packet indicating which of the prior sequence numbers it has correctly received. The recipient of the data packet, $y$, detects missing packets using the gaps in the received sequence and sends NAKs to $x$ to request the appropriate retransmissions. $x$ can either be a parent of $y$ in the data delivery tree, or a random node forwarding along a cross edge.

The PRM extensions to NICE required less than 500 lines of C code.

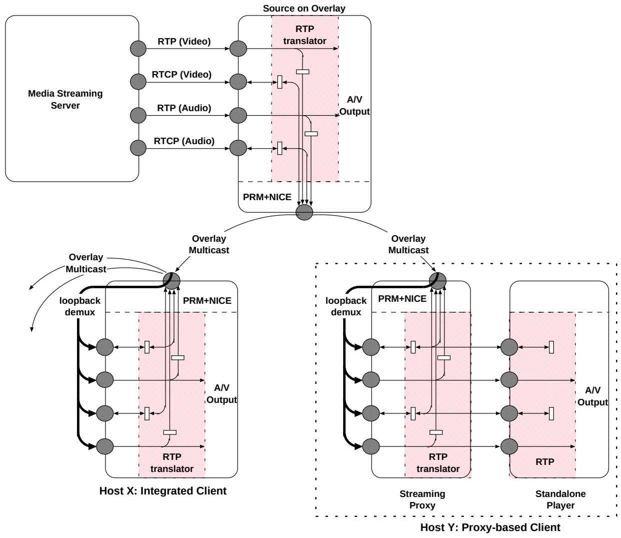


Fig. 5. RTP and RTCP paths in SRMS.

### B. Media Transport

Like most streaming media systems, SRMS uses the Real-time Transport (RTP) protocol to transport encoded media. RTP consists of a data protocol and a control protocol. The data component carries encoded media in the payload, timing and synchronization information and the source identifier (known as the "synchronization source" or SSRC). The control component is called Real-time Transfer Control Protocol (RTCP) [28] and performs a variety of related control operations, e.g. quality of service feedback from receivers, synchronization of different media streams, etc.

RTP enables application sources to perform quality of service adaptations by defining mechanisms for receivers to send appropriate feedback. All RTP data packets carry sequence numbers (generated by the data source) and receivers use gaps in the received sequence numbers to infer loss rates on the path. Periodically receivers send back quality of service feedback, e.g. loss rates, using Receiver Report (RR) RTCP packets. The source application can use this feedback to appropriately adapt the data rate. In general, if a receiver encounters high losses on the data path, the source can reduce the media quality using aggressive quantization or can transcode the media to a higher compressed format. Efficient implementations of media transcoding and compression can be found in [2], [30].

If network-layer multicast is used for streaming media to a group of clients, data rate adaptations by the source would affect all clients. However, the use of application-layer multicast in SRMS provides a new opportunity where the system can perform selective data rate adaptation based on loss rates and access bandwidths of individual clients. To do this, we treat each client on the overlay data delivery path as a potential *RTP translator*. According to RFC 1889 [28], a translator is an entity which forwards RTP packets of a stream without changing the source identifier that generated the data.

If there is no bandwidth mismatch between upstream and downstream nodes of an overlay hop, no data transcoding operation is necessary and the RTP data packets can be forwarded without any change. However, some of the RTCP packets carry control information which apply only to that specific overlay hop only e.g. the Receiver Report (RR) packets. Therefore these RTCP packets are not forwarded to the entire data delivery tree using the overlay multicast operation, but are terminated and re-generated as needed, on each overlay hop. We defer this aspect of RTP translation to Section V-E.

We now describe the implementation of the designated source and the media clients.

### C. Designated Source

In conformance with the RTP standards the media streaming server sends the content using two separate RTP streams, one for audio and one for video. Each RTP stream has an accompanying RTCP stream for exchange of control information. Thus it uses four separate ports, as specified in RFC 1889 [28]. In the SRMS system the designated source receives these streams in four different ports. These RTP/RTCP packets are delivered by the network stack to the designated source. The RTP data packets (transcoded if necessary) and the re-generated RTCP packets are multiplexed onto a single overlay multicast port and forwarded to clients along the overlay delivery tree.

### D. SRMS-client

The SRMS-client has three logical components:

- Overlay-multicast: This is the PRM-enhanced NICE application-layer multicast protocol.
- RTP translator: The translator performs any necessary data rate adaptations before forwarding on the overlay hops.
- Audio/Visual Output: This component is responsible for the playback of the media. We use the player code from the `MPEG4IP` tool publicly available from `http://sourceforge.mpeg4ip.net`.

Each client receive the RTP packets through the single overlay multicast port. The overlay multicast code delivers the packet to the appropriate RTP or RTCP port internally (as shown in Figure 5).

The SRMS-client can be implemented in two different ways. In the first approach, we call *integrated client*, all the three components are implemented in a single process (Host $X$ in Figure 5). This is the most efficient implementation of the client. In particular, there is no redundant RTP code (unlike in the alternative approach described next). Additionally it is possible to closely integrate the different components. For example, the duplicate detection and suppression buffers of PRM and the playback buffer of the audio/video output component can be shared and there is no redundant data movement.

In the alternative approach, we call *proxy-based client*, there are two processes that together serve as a single client (Host $Y$ in Figure 5). One process implements the overlay multicast and the RTP translator functionalities. This process, called the streaming proxy, demultiplexes the RTP and RTCP packets from the overlay multicast port, performs necessary RTP translations and forwards the data along its downstream overlay hops. Additionally it forwards a copy of the received RTP and RTCP packets to the other process, the stand-alone media player. A number of such media players (e.g. MPEG4IP) are available in the proprietary and in the free software domain. These media players typically interact with media streaming servers using RTP, and implement RTP functionality themselves. Therefore, RTP functionality in the proxy-based clients is replicated in the two processes. While such a construction is comparatively inefficient, it has its advantages. It decouples the implementation of the media player from the rest of the SRMS-system. Thus, the user has the flexibility of using off-the-shelf media player binaries as part of the SRMS system.

A key component of handling RTP packets in the clients, is the process of RTP translation. We now present a brief summary of how this is handled in the clients and its consequent interaction with PRM.

### E. RTP Translation

Prior to forwarding the data to each downstream neighbor the RTP component optionally transcoded the media to a different (lower data rate) encoding format. Such transcoding is performed at the granularity of overlay hops. As a consequence clients are not constrained to the minimum bandwidth on the entire network. Instead each client is able to receive data at the maximum permissible bandwidth on the path to the source.

RTP translation is a relatively expensive operation in terms of processing and need not be performed by all clients. In fact only the clients that have disparate available bandwidth in their upstream and downstream hops need to perform the transcoding. A good description of such an application-level gateway for video streams can be found in [2].

The summarize the key aspects of the RTP translation process are as follows:

- If the packetization interval or frame rate is changed or the sampling frequency of the data packets is altered, then the timestamp in the packets needs to be adjusted.
- If RTP packets are merged or split then the sequence number needs to be appropriately altered. Additionally the packet counts in the sender/receiver report (SR/RR) RTCP packets should also be updated to reflect these changes. Similarly if the translator also changes the data encoding then the octet counts also need to be appropriately adjusted.
- All SDES CNAME packets and BYE packets (both part of RTCP) should be forwarded unchanged. The SSRC field should not be altered.

Each overlay hop in RTP is treated as an independent RTP session and independent data transcoding can be performed on them. As a consequence the entire group of clients are logically split into connected subsets of the data delivery paths. All clients within the same subset receive media with the same data encoding format and data rate. We call each such connected subset, a *media domain*.

*Interaction of RTP translation and PRM*

In PRM randomized forwarding, each overlay node chooses another overlay node at random and forwards a data packet to the latter with a low probability. Consider the case where an overlay node, $X$, forwards a data packet to another node, $Y$, along a random edge, and $X$ and $Y$ belong to two different media domains. In such a case the media encoding in the payload of the forwarded packet may potentially be inconsistent with the media encoding at $Y$ due to intermediate RTP translations.

There are two simple solutions that reduce such wasteful data forwarding along random edges.

- *Use a single media encoding format:* The media encoding format is not changed during RTP translations. Instead an aggressive quantizer is used on the input stream to produce a lower quality, lower bit rate stream with the same format. In such a scenario all media domains will use the same media encoding format and will be compatible with each other. This approach is relatively simple, but it trades off media quality to reduce data rates.
- *Restrict random forwarding to compatible media domains:* In this alternative approach we propose a simple extension to PRM as follows. Each overlay node randomly discovers other overlay nodes only within its own media domain, or other media domains with a compatible media encoding. To enforce this constraint we have to modify the behavior of the *Discover* message used by an overlay node to randomly discover some other overlay nodes. For this the application-level *TTL* of a *Discover* message is not decremented when it passes through an overlay node that has an incompatible media encoding with the source node. This ensures that the *TTL* of the *Discover* message reaches zero only at another overlay node with a compatible media encoding. Such a technique also requires interaction between the overlay multicast component (PRM enhancements in this case) and the RTP translator, and an API which supports such an interaction.

*Data-rate adaptation in SRMS:* The design of SRMS allows flexible use of different media translation and data adaptation mechanisms. In our prototype implementation, we use a simplified variant of the first option proposed above — use of a single media encoding format. We have implemented a very simple *packet dropping* mechanism as the adaptation technique. If the upstream client detects that packet losses on the overlay hop is above a configurable threshold ($\pi_{high}$), it selectively

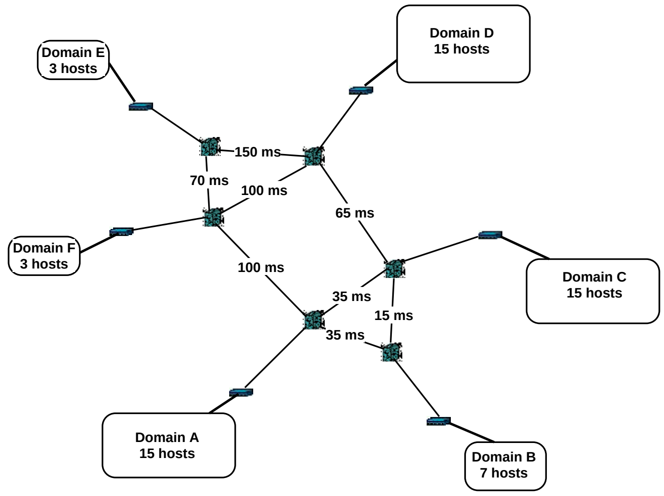


Fig. 6. The network topology used in the emulated network environment. The latencies of the backbone links are marked. The losses on the links connecting the access gateways and the backbone were chosen randomly between 1% and 2%. Within each domain the latencies between pairs of hosts were randomly assigned between 5 to 10 ms, and the corresponding losses were 0.2% to 0.5%.

drops a proportional fraction of the data packets. If the loss rate is below another configurable threshold, ($\pi_{low}$), it reduces the selective packet drop rate by a constant factor. The data packet loss rates are available from the receiver report (RR) messages. More sophisticated schemes e.g. Binomial congestion control [5], Generalized-AIMD [33], TFRC [10] etc. can also be used in this adaptation to choose the packet drop rates. Prior work [16] has shown that such packet-dropping based bandwidth adjustments is an effective way to perform data rate adaptation, without significantly impacting the media quality.

However the design goal in SRMS is to provide the mechanism that enables adaptation and allow the exact adaptation and data translation scheme to be chosen according to application goals. Apart from packet dropping techniques, transcoding based adaptations also provide a viable and efficient option. An example of efficient media transcoding technique [2] has been defined in prior literature.

## VI. Experimental Results

In this section we focus on the results from our implementation of the SRMS system. Prior work [4] had investigated the resilience achieved by PRM through simulation studies. These results, in the context of application-layer multicast protocols, can be summarized as follows: A number of different schemes can achieve high data delivery ratios when the failure rates of overlay nodes are insignificant. However, only PRM achieves high data delivery ratio even when the failure rate of overlay nodes is relatively high. In a comparison of PRM with FEC-based schemes the authors in [4] had shown that not only PRM achieves significantly higher data delivery ratios under moderate overlay node failure rates, but it is able to do so with orders of magnitude less overhead.

### A. Experiment Testbed

Our experiments were performed on a publicly available emulated network environment, as well as a public wide-area testbed. In all our experiments, we streamed multimedia data from the Darwin streaming media server (publicly available at `http://www.apple.com/quicktime/products/qtss/`). We used a four minute MPEG4 encoded movie and streamed it cyclically to the clients using SRMS. Different experiments reported in this section were between 15 minutes and one hour in duration. The Darwin media streaming server and the designated source of the multicast group were co-located in the same host for all these experiments. The bandwidth of the media stream was about 250 Kbps.

For the emulated experiments, we set up the network topology as follows. We modeled a group of clients distributed geographically in different parts of the world. We performed ping based latency measurements and used it to assign the latencies between these geographic domains. There were multiple hosts in each domain and the latencies between pairs of hosts in each domain was randomly chosen from between 5 ms and 10 ms. As an example, the end-to-end latency between a host in domain C and another host in domain F is (5 to 10) ms + 35 ms + 100 ms + (5 to 10) ms = 145 to 155 ms We assumed a larger loss rate at the access gateways of each domain (between 1% and 2% packet losses) and comparatively lower losses inside each domain (between 0.2% and 0.5%). This topology is shown in Figure 6. The machines in the emulated environments were 650 and 800 MHz Pentium machines running versions of Linux or FreeBSD.

Our wide-area testbed consists of 32 hosts. Out of these 4 hosts were in Europe, 2 in Asia, 1 in Canada, and the remaining in different locations in the USA. The machines of the wide-area testbed consisted of Celeron 733 MHz, and different Pentium machines with processor speeds between 300 MHz and 1.7 GHz, running versions of Linux and FreeBSD.

### B. Experiment Scenarios

We have evaluated the performance of the SRMS system for a range of group sizes (upto 128), join-leave patterns and system parameters. In these experiments, all departures of clients were modeled as "ungraceful leaves." This is equivalent to a host failure, where the departing member is unable to send a *Leave* message to the group.

In the experiments reported in these section, we first let a set of end-hosts join the multicast group. Subsequently end-hosts join and leave the multicast group, which was varied for different experiments. The join and the leave rate for members are chosen to be equal so that the average size of the group remained nearly constant. We studied the various data delivery properties of SRMS under these dynamic conditions. The Darwin server continuously streams the movie to the group. Since the clients were running at remote hosts, we disabled the actual media playback in the clients for these experiments. We logged all RTP data packets received at the clients and used the RTP sequence numbers to detect packet losses.

In all results reported in this section, we use the notation PRM-$b(r, \beta)$ to indicate an SRMS configuration where the parameters of PRM are set as follows: $b$ denotes the size of the bitmask used for NAK-based retransmissions, $r$ denotes the number of randomly chosen neighbors, and $\beta$ denotes the probability of forwarding to each of these random neighbors.

| Scheme | Failure and Join rate (per min) 1.2 | 4.8 |
|---|---|---|
| BE | 0.81 | 0.72 |
| PRM-128 (3,0.01) | 0.98 | 0.98 |
| PRM-256 (3,0.01) | 0.99 | 0.98 |

TABLE I

COMPARISON OF DATA DELIVERY RATIO FOR DIFFERENT OVERLAY NODE FAILURE RATES. THE AVERAGE NUMBER OF OVERLAY NODES IN THE EXPERIMENT WAS 64 AND THE EXPERIMENT DURATION WAS 30 MINUTES.

### C. *Experiments on the Emulated Network Environment*

We first describe results from experiments performed on the emulated network environment using the network topology is shown in Figure 6.

*Resilience:* In Table I we show the average data delivery ratios achieved using the SRMS system for experiments (30 minutes in duration each) in which there were an average of 64 clients. The best-effort based delivers 72% and 81% of the data for the two different failure rates of 1.2 per minute and 4.8 per minute respectively. In contrast the PRM-based system achieves between 98% and 99% data delivery in all cases. We can also observe that a longer bitmask (for NAK-based retransmissions) leads to better performance (we will discuss this aspect later in this section).

We now present a more detailed snapshot of data delivery ratio in Figure 7. This plot corresponds to experiments with node failure and join rate of 4.8 per minute. Each SRMS scheme used the same join-leave pattern. In this scenario, four intermediate overlay nodes departed from the group between time 850 and 860 seconds. As can be observed, the effect of these departures is quite severe for the best-effort case and the data delivery ratio decreases to less than 10%. In contrast the PRM-based scheme maintains a high data delivery ratio ($> 95\%$) for all receivers at all times.

In Figure 8 we plot the cumulative distribution of the maximum data outage period experienced by the different clients in the same experiment (with 4.8 failures and joins per minute). The PRM-based scheme with a bitmask size of 256 performs extremely well — about 98% of the clients have a *maximum* data outage period of less than 10 seconds. As noted before, using a longer bitmask helps improve the data delivery ratio. This is a significant improvement over the best-effort case, where more than 90% of the group experience data outages of 30 seconds or more.

Note that in all these experiments, each client chose three other random clients (i.e. $r = 3$) and forwarded data to them with probability, $\beta = 0.01$. This implied that the additional data overheads for the PRM-based schemes, in these experiments, was 3%.

*Control overheads:* For all these experiments we also measured the control overheads incurred by the system. The control overheads at each node was essentially insignificant compared to the data rate. For the 64 node experiments, the total control overheads was about 2.9 control packets per second on average at each overlay node. Out of these, on average about 0.96 control packets per second was due to overlay tree construction and maintenance using the NICE application-layer multicast protocol. Most of the remaining control packets were due to NAKs. The volume of NAKs would depend on the source data rate and network loss rates. In these experiments, NAKs accounted for about 1.78 control packets per second at each overlay node. The *Discover* messages and the corresponding responses accounted for about 0.2 control packets per second at each overlay node.

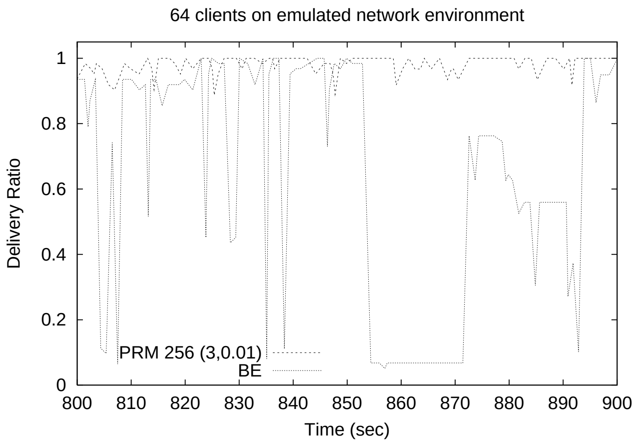


Fig. 7. Data delivery ratio achieved for a group of 64 clients as they join and leave the multicast group. Overlay node failure and join rate was 4.8 per minute. Between time 850 and 860 seconds four intermediate overlay nodes on the data delivery tree left the multicast group.

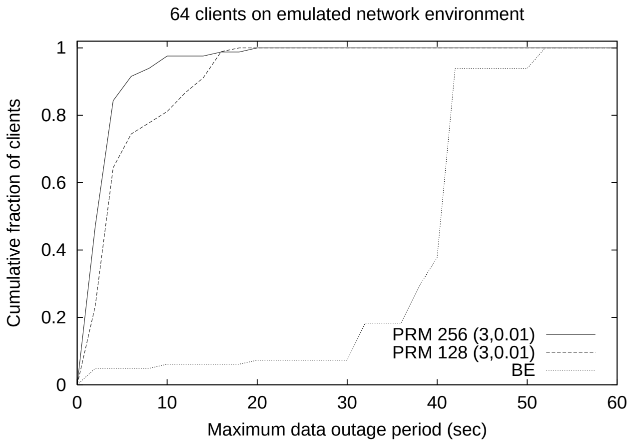


Fig. 8. The cumulative distribution of the largest data outage period seen by all the clients for a 64 client experiment with overlay node failure and join rate of 4.8 per minute.

### D. *Wide-area Experiments*

The wide-area experiment was performed using 60 clients on the wide-area testbed. We ran two or three clients on each of the hosts. The host with the Darwin server and the designated host was located in the USA, and distribution of one-way latencies from the source to the other clients varied between less than 1 ms to 225 ms. To limit the load imposed on this wide-area testbed, we had also reduced the data rate sent out from the Darwin server to about 32 Kbps. We correspondingly reduced the bitmask size (used for NAK based retransmissions) to smaller values. As before, the overlay node failure and join rate was 4.8 per minute.

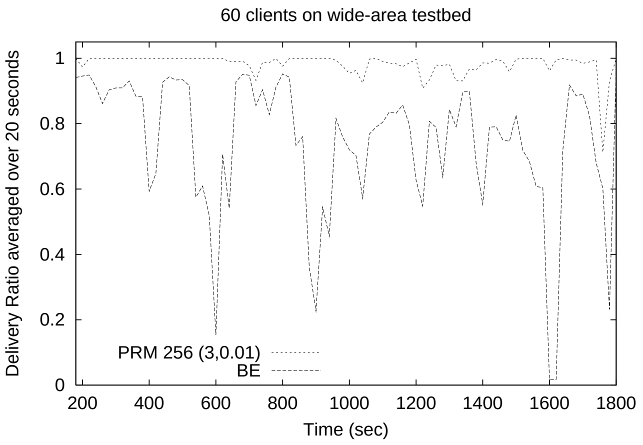


Fig. 9. Data delivery ratio achieved by a group of 60 clients (on average) on the wide-area testbed for a 30 minute experiment. The media stream was started three minutes into the experiment. The data delivery ratio is averaged over each 20 second interval for clarity. The overlay node failure and join rate was 4.8 per minute each.

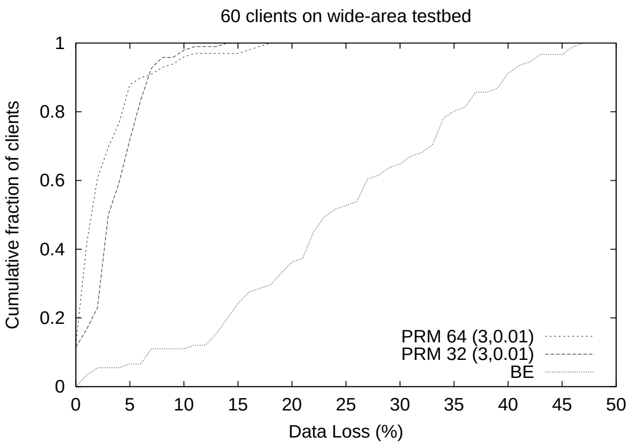


Fig. 10. Cumulative distribution of data losses for a 30 minute experiment on a group with 60 clients (on average) performed on the wide-area testbed. The overlay node failure and join rate was 4.8 per minute each.

In Figure 9 we show the data delivery ratio achieved over the entire duration of this experiment. To improve the legibility of this plot, we averaged the data delivery ratio over 20 second intervals. We can observe that SRMS achieves a high data delivery ratio for nearly the entire 30 minute duration, while the best-effort based data delivery suffers significant losses.

In Figure 10 we show the cumulative distribution of data that was lost at the different clients for the same experiment. It can be observed that for the PRM-based system (with a 64 bit bitmask) about 20% of the clients do not experience *any* data loss on the wide-area testbed. About 90% of the clients experience a loss of less than 5%. The additional data overheads for both the PRM-based schemes were 3%. This is a significant improvement over the best-effort based SRMS system, in which about 50% of the clients experience more than 20% data loss.

## VII. Related Work

A large number of research efforts (IVS [32], Rendez Vous [2], vic, vat [3], rat [4], CUSeeMe [5]etc.) have addressed real-time media streaming in the last decade. Media streaming to a group of users in these systems typically relied on network-layer multicast (or MBone) support. A number of commercial efforts e.g. Real Networks, Windows Media Player, Fast Forward Networks, handle media streaming to groups of users using proprietary protocols. In contrast SRMS is an overlay multicast based media-streaming system that is the first of its kind to provide strong resilience guarantees. It is an open-source software, built using some other open-source softwares available today.

A large number of research proposals have addressed reliable delivery for multicast data, most notably in the context of network-layer multicast. A comparative survey of these protocols is given in [18] and [31]. In SRM [11] receivers send NAKs to the source to indicate missing data packets. Each such NAK is multicast to the entire group and is used to suppress NAKs from other receivers that did not get the same packet. In this approach, however, a few receivers behind a lossy link can incur a high NAK overhead on the entire multicast group.

Tree-based protocols provide another alternative solution for reliable and resilient multicast. In this approach the receivers are organized into an acknowledgment tree structure with the source as the root. This structure is scalable because the acknowledgments are aggregated along the tree in a bottom-up fashion and also allows local recovery and repair of data losses. Protocols like RMTP [24], TMTP [34], STORM [26], LVMR [20] and Lorax [19] construct this structure using TTL-scoped network-layer multicast as a primitive. In contrast, LMS [23] uses an additional mechanism, called directed subcast, to construct its data recovery structure. Our work differs from of all these above approaches in two key aspects. First, unlike all these protocols that employ network-layer multicast service for data distribution our scheme is based upon an application-layer multicast delivery service.

More recently, a number of projects have addressed the problem of constructing efficient data delivery paths for application-layer multicast [9], [12], [3], [35], [15], [7], [25], [17]. Of these, the Narada protocol [8] has been used to deliver media streams to a set of clients. However the protocol itself does not address the issue of resilience and recovery. The Overcast protocol [15] is defined specifically to provide reliable multicast services using overlays. Each overlay hop in Overcast uses TCP for data transfer and such a construction is not suitable for streaming media applications with real-time requirements. In fact none of these protocols explicitly address the issue of resilience which is essential to media streaming applications.

To the best of our knowledge the SRMS system is the first application-layer multicast based scheme that addresses resilience. Second, all the network-layer multicast based schemes described employ completely reactive mechanisms for providing data reliability and therefore incurs moderate or high delivery latencies. As we show in this paper, proactive mechanisms,

[2]Rendez Vous is available at www.lyonnet.org/IVStng
[3]Both vic and vat are available at www-nrg.ee.lbl.gov
[4]rat is available at www-mice.cs.ucl.ac.uk/multimedia/software/rat/index.html
[5]CUSeeMe is currently available commercially at www.fvc.com

| Scheme | Data delivery | Recovery mechanism | Overheads | Recovery latency |
|---|---|---|---|---|
| SRM [11] | Network multicast | Reactive NAKs with global scope | High (for high network losses) | High |
| STORM [26], Lorax [19] | Network multicast | Reactive NAKs on ack tree | Low | Moderate |
| LMS [23] | Network multicast and directed subcast | Reactive NAKs on ack-tree | Low | Moderate |
| RMTP [24] LVMR [20] | Network multicast | Reactive/periodic ACKs with local scope | Low | Moderate |
| TMTP [34] | Network multicast | Reactive NAKs and periodic ACKs with local scope | Low | Moderate |
| Parity-based [22] (APES [27]) | Network multicast (and directed subcast) | Reactive NAKs and FEC-based repairs | Moderate | Moderate |
| FEC-based [14], [22], [6], [21] | Network multicast or App-layer multicast [6] | Proactive FECs | High | Low |
| Overcast [15] | App-layer multicast | Reactive ACKs (TCP) | Low | Moderate |
| SRMS | App-layer multicast | Proactive randomized forwarding and reactive NAKs | Low | Low |

TABLE II

COMPARISON OF DIFFERENT RELIABILITY/RESILIENCE MECHANISMS FOR MULTICAST DATA.

e.g. randomized forwarding, can be used to significantly improve resilience for applications that require low latency data delivery.

SRMS is not the only system to provide improved reliability performance for multicast data. There exists some well-known forward error correcting code based approaches that are also proactive in nature. For example, Huitema [14] had proposed the use of packet level FECs for reliable multicast. Nonnenmacher et. al. [22] studied and demonstrated that additional benefits can be achieved when an FEC-based technique is combined with automatic retransmission requests. APES uses a related approach for data recovery [27]. Digital Fountain [6] and RPB/RBS [21] are two other efficient FEC-based approaches that provide significantly improved performance. All these FEC based approaches can recover from network losses. However, they alone are not sufficient for resilient multicast data delivery when overlays are used. Overlay nodes are processes on regular end-hosts and are more prone to failures than network routers. FEC-based approaches are not sufficient to recover from losses due to temporary losses on the data path, especially when low-latency delivery is required. However SRMS differs from all these other schemes by providing a proactive component that allows the receivers to recover from losses due to overlay node failures. In Table II we summarize the recovery characteristics of all these schemes including SRMS.

SRMS also defines a framework in which various bandwidth adaptation techniques can be applied within the context of media streaming to user groups. Different bandwidth adaptation schemes defined in prior literature are, therefore, complementary to our work. The MeGa Gateway [2] within the Active Services framework [1] describe the implementation experience of a media transcoding system. LVMR [20] defines an alternate approach to bandwidth adaptation based on layered encoding of video, specially in the context of network-layer multicast, where clients can subscribe to a subset of video layers in accordance with their processing power and access bandwidths.

[6] Although FEC-based schemes can be implemented over application-layer multicast, as this paper shows, it alone is not sufficient to achieve high delivery ratios even under moderate frequency of membership changes on the overlay.

## VIII. CONCLUSIONS

In this paper, we have described SRMS, an application-layer multicast based system for resilient media streaming to a large group of clients. The system is implemented in a way such that it can interoperate with existing tools for media streaming and playback.

Being based on the PRM loss resilience scheme, SRMS is able to achieve very high data delivery ratios inspite of network losses and overlay node failures. Through our detailed analysis we prove that PRM (and consequently SRMS) scales well to large groups. In particular the additional data overheads required to achieve high data delivery ratios asymptotically go to zero as the size of the group increases. In this paper we have also derived necessary and sufficient conditions that enable PRM to have such asymptotic scaling properties. These results are especially interesting since as prior work [4] shows, some of the other existing error-recovery techniques, such as FEC, alone do not provide adequate data recovery for application-layer multicast based data distribution.

Another interesting component of SRMS is its architecture that allows flexible implementation of data rate adaptation to suit application needs. We use existing techniques and protocols to enable selective data rate adaptation based on the network conditions and access bandwidths of individual clients.

We have studied the performance of SRMS through detailed experiments on public emulated network environments and wide-area testbeds. Our results show that SRMS provides good data resilience ($> 97\%$ delivery ratio) even under adverse conditions with less than 5% overheads.

## References


[1] E. Amir, S. McCanne, and R. Katz. An active service framework and its application to real-time multimedia transcoding. In *ACM Sigcomm*, Sept. 1998.

[2] E. Amir, S. McCanne, and H. Zhang. An application level video gateway. In *ACM Multimedia*, Nov. 1995.

[3] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable application layer multicast. In *Proc. ACM Sigcomm*, Aug. 2002.

[4] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan. Resilient multicast using overlays. In *Proc. of ACM SIGMETRICS*, 2003.

[5] D. Bansal and H. Balakrishnan. Binomial congestion control algorithms. In *IEEE Infocom*, Apr. 2001.

[6] J. Byers, M. Luby, and M. Mitzenmacher. A digital fountain approach to asynchronous reliable multicast. *IEEE Journal on Selected Areas in Communications*, 20(8), Oct. 2002.

[7] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron. SCRIBE: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in communications*, 20(8), Oct. 2002. To appear.

[8] Y.-H. Chu, S. G. Rao, S. Seshan, and H. Zhang. Enabling Conferencing Applications on the Internet using an Overlay Multicast Architecture. In *Proceedings of ACM SIGCOMM*, Aug. 2001.

[9] Y.-H. Chu, S. G. Rao, and H. Zhang. A Case for End System Multicast. In *Proceedings of ACM SIGMETRICS*, June 2000.

[10] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. In *ACM Sigcomm*, Aug. 2000.

[11] S. Floyd, V. Jacobson, C. Liu, S. McCanne, and L. Zhang. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Transactions on Networking*, 5(6), Dec. 1997.

[12] P. Francis. Yoid: Extending the Multicast Internet Architecture, 1999. White paper http://www.aciri.org/yoid/.

[13] W. Hoeffding. Probability inequalities for sums of bounded random variables. *American Statistical Association Journal*, 58:13–30, 1963.

[14] C. Huitema. The case for packet level FEC. In *Proc. 5th International Workshop on Protocols for High Speed Networks*, Oct. 1996.

[15] J. Jannotti, D. Gifford, K. Johnson, M. Kaashoek, and J. O'Toole. Overcast: Reliable Multicasting with an Overlay Network. In *Proceedings of the 4th Symposium on Operating Systems Design and Implementation*, Oct. 2000.

[16] R. Keller, S. Choi, D. Decasper, M. Dasen, G. Fankhauser, and B. Plattner. An active router architecture for multicast video distribution. In *IEEE Infocom*, Apr. 2000.

[17] J. Leibeherr and M. Nahas. Application-layer Multicast with Delaunay Triangulations. In *IEEE Globecom*, Nov. 2001.

[18] B. Levine and J. Garcia-Luna-Aceves. A comparison of reliable multicast protocols. *Multimedia Systems Journal*, 6(5), Aug. 1998.

[19] B. Levine, D. Lavo, and J. Garcia-Luna-Aceves. The case for concurrent reliable multicasting using shared ack trees. In *Proc. ACM Multimedia*, Nov. 1996.

[20] X. Li, S. Paul, P. Pancha, and M. Ammar. Layered video multicast with retransmissions (LVRM): Evaluation of error recovery schemes. In *Proc. NOSSDAV*, 1997.

[21] A. Mahanti, D. Eager, M. Vernon, and D. Sundaram-Stukel. Scalable on-demand media streaming with packet loss recovery. In *ACM Sigcomm*, Aug. 2001.

[22] J. Nonnenmacher, E. Biersack, and D. Towsley. Parity-based loss recovery for reliable multicast transmission. *IEEE/ACM Transactions on Networking*, 6(4), Aug. 1998.

[23] C. Papadopoulos, G. Parulkar, and G. Varghese. An error control scheme for large-scale multicast applications. In *Proc. Infocom*, 1998.

[24] S. Paul, K. Sabnani, J. Lin, and S. Bhattacharyya. Reliable multicast transport protocol (rmtp). *IEEE Journal on Selected Areas in Communications*, 15(3), Apr. 1997.

[25] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Application-level multicast using content-addressable networks. In *Proceedings of 3rd International Workshop on Networked Group Communication*, Nov. 2001.

[26] X. Rex Xu, A. Myers, H. Zhang, and R. Yavatkar. Resilient multicast support for continuous-media applications. In *Proceedings of NOSSDAV*, 1997.

[27] D. Rubenstein, S. Kasera, D. Towsley, and J. Kurose. Improving reliable multicast using active parity encoding services (APES). In *Proc. Infocom*, 1999.

[28] H. Schulzrinne, G. Gokus, S. Casner, R. Frederick, and V. Jacobson. RTP: A transport protocol for real-time applications. *RFC 1889*, Jan. 1996.

[29] H. Schulzrinne, A. Rao, and R. Lanphier. Real time streaming protocol: RTSP. *RFC 2326*, Apr. 1998.

[30] B. Smith. Fast software processing of motion JPEG video. In *ACM Multimedia*, Oct. 1994.

[31] D. Towsley, J. Kurose, and S. Pingali. A comparison of sender-initiated and receiver-initiated reliable multicast protocols. *IEEE Journal on Selected Areas on Communication*, 15(3), Apr. 1997.

[32] T. Turletti and C. Huitema. Videoconferencing in the internet. *IEEE/ACM Transactions on Networking*, 4(3), June 1996.

[33] Y. R. Yang and S. Lam. General aimd congestion control. In *International Conference on Network Protocols*, Nov. 2000.

[34] R. Yavatkar, J. Griffioen, and M. Sudan. A reliable dissemination protocol for interactive collaborative applications. In *ACM Multimedia*, Nov. 1995.

[35] B. Zhang, S. Jamin, and L. Zhang. Host multicast: A framework for delivering multicast to end users. In *Proc. IEEE Infocom*, June 2002.


## IX. Appendix: Proof sketch for Theorem IV.1

We now outline the main ideas of the proof. In several places below, we will employ the union bound: for any collection of events $E_1, E_2, \ldots, E_m$,

$$\Pr[E_1 \vee E_2 \vee \cdots \vee E_m] \leq \sum_{i=1}^{m} \Pr[E_i].$$

Let the overlay tree have some depth $d$; for $i = 0, 1, \ldots, d-1$, suppose all nodes at depth $i$ have some $D_i$ children. (Recall that the root is at depth 0, its children at depth 1, etc.) Recall the failure model (A2) of Section IV. Suppose node $u$ does not fail. Then, define $C(u)$ to be the connected component (subtree) containing $u$, after the node- and link-failures occur. For each connected component $C$, we choose as a *leader*, an arbitrary element of $C$ of *smallest depth*. Suppose the probability of random forwarding $p$ has been chosen (as an arbitrarily small positive constant), and that each value $D_i$ is at least $C \log n$ where $C$ is a sufficiently large constant, as required by Theorem IV.1. We first show that the following conditions hold simultaneously with high probability:

(P1) the number of surviving nodes at every depth $i$ is at least $(1-2\epsilon) \cdot D_0 D_1 \cdots D_{i-1}$; and

(P2) for all $i \leq d-1$ and for all surviving nodes $u$ at depth $i$, the number of descendants of $u$ that lie in $C(u)$, is at least $((1-\epsilon)^2(1-\delta))^{d-i} \cdot D_i D_{i+1} \cdots D_{d-1}$.

Claim (P1) is proved using a simple application of the Hoeffding bound [13]. Claim (P2) is proven using an iterative application of the Hoeffding bound; the intuition is as follows. Call a node *good* if it, as well as the link connecting it to its parent, survive. The expected number of good children of $u$ is $D_i(1-\epsilon) \cdot (1-\delta)$; since $D_i$ grows at least logarithmically in $n$, one can use the Hoeffding bound to show that with high probability, at least $D_i(1-\epsilon)^2 \cdot (1-\delta)$ children of $u$ are good. Iterating this argument down the tree and applying a union bound over all $u$, we prove (P2).

The heart of the proof is as follows; we first give a proof sketch for part (i) of Theorem IV.1. To show that every surviving non-leaf node gets the data with high probability, it suffices to show that for every non-leaf *leader* $u$, some node in $C(u)$ gets the data with high probability. (Once this happens, the data gets reliably transmitted across the links of $C(u)$ with probability 1.) For the sake of simplicity of our mathematical expressions, we assume here that all non-leaf nodes have the same number of children $D$ (this condition is only required for notational convenience here). Let $\exp(x)$ denote $e^x$. We show the following claim by induction on $i$, where $a$ is a positive constant, and $\psi$ is a strictly positive quantity which depends only on $\epsilon$ and $\delta$:

*Let $u$ be an arbitrary non-leaf leader at depth $i$. Then, conditional on (P1) and (P2), the probability of no node in $C(u)$ getting the data is at most $\exp(-a(D\psi)^{d-i})$.*

The base case of the induction is for $i = 0$ which corresponds to the case where $u$ is the root. Since the root never fails, the claim holds trivially. Next suppose $i \geq 1$. Assuming the claim for all $i' < i$, we complete the induction for $i$ as follows. If (P1) and (P2) hold, it can be shown that the cardinality of the set $\mathcal{S}$ of surviving nodes in connected components whose leaders are at depth strictly smaller than $i$, is at least

$$\begin{aligned} n_1 &= \Omega(D^d((1-\epsilon)^2(1-\delta))^{d-i+1}) \\ &= \Omega(n \cdot ((1-\epsilon)^2(1-\delta))^{d-i+1}). \end{aligned}$$

By choosing $a$ appropriately, we can apply the induction hypothesis and a union bound to show that the probability of even one such connected component (whose leader is at depth smaller than $i$) not getting the data, is much smaller than $\exp(-a(D\psi)^{d-i})$. Next, assuming (P2), the size of $C(u)$ is at least

$$n_2 = ((1-\epsilon)^2(1-\delta))^{d-i} \cdot D^{d-i}.$$

Thus, the probability that no random forward from $\mathcal{S}$ arrived into $C(u)$, is at most

$$(1-p/n)^{\Omega(n_1 n_2)} \leq \exp(-\Omega(p \cdot D^{d-i} \cdot ((1-\epsilon)^2(1-\delta))^{2(d-i)+1},$$

which can be bounded by $\exp(-\Omega((D\psi)^{d-i}))$ for a suitable choice of $\psi$. These ideas help complete the induction proof; a union bound over all surviving non-leaf nodes (using the facts $d - i \geq 1$ and that $D$ grows at least logarithmically in $n$) then shows that all of them get the data with high probability.

Having shown the above, we can handle the surviving leaves, to prove part (ii) of Theorem IV.1. Consider the surviving leaves; the fraction of these such that their parent, as well as the link to their parent, survive, can be shown to be at least $(1-\epsilon) \cdot (1-\delta) \cdot (1-g(n))$ with high probability using the Hoeffding bound, where $g(n)$ tends to 0 as $n$ increases. Now, we have argued above that all surviving non-leaves get the data with high probability; thus, all surviving leaves that remain connected to their (surviving) parent, will receive the data with high probability. This completes the proof sketch for part (ii) of Theorem IV.1.

The degree lower bound of a suitable constant times $\log n$ can be shown to be asymptotically necessary, as follows. Suppose, for some small constant $b > 0$, that all parents of leaves have degree at most $b \log n$. Then, we give a proof sketch a few lines below that with probability tending to 1 as $n$ increases, a substantial number of *surviving* parents-of-leaves will have the following property: they get disconnected from all of their neighbors in the tree. Then, if the overhead needs to be kept small, most of these isolated parents-of-leaves will not get the data, with high probability. Here is a proof sketch. Suppose the number of parents-of-leaves that survive is some value $t$. For any one of them, the probability that it gets disconnected from all of its neighbors can be as high as

$$\begin{aligned} (\epsilon + \delta - \epsilon\delta)^{b \log n} &\geq \epsilon^{b \log n} \\ &= n^{-b \log(1/\epsilon)}. \end{aligned}$$

One can then show that with high probability, the number of these completely isolated nodes is $\Omega(t \cdot n^{-b \log(1/\epsilon)})$, which is large if $b$ is small enough (e.g., if $b \leq 1/(2 \log(1/\epsilon))$). Furthermore, if we require low overhead, the random forwards will with high probability not reach most of these nodes, thus leading to several surviving nodes not receiving the data.

Next, consider the tree augmentation scheme of Section IV-A; we now sketch why its overhead of $\Omega(\frac{\log(1/\phi)}{\log(1/\epsilon)})$ is necessary. Suppose the average overhead of a node is at most $\alpha \doteq c \cdot \frac{\log(1/\phi)}{\log(1/\epsilon)}$, for some small constant $c$. Then, since most of the nodes are at the leaf level, most leaves get connected to at most $3\alpha$ other nodes, including the random forwards. For a given such leaf, the probability that all of these $3\alpha$ interconnections go to failed nodes, is

$$\epsilon^{3\alpha} = \phi^{3c},$$

which is much larger than $\phi$ if $c \ll 1/3$ (recall that $\phi < 1$). Thus, the fraction of successful nodes that do not get the data, will with high probability be much more than $\phi$ in such a situation. Finally, a similar proof shows that The "$(1-\epsilon) \cdot (1-\delta) \cdot (1-g(n))$ fraction" bound of part (ii) of Theorem IV.1 is optimal if we desire low overhead. The idea is that under low overhead, with high probability the only connections (including the random forwards) for most leaves will be to their parents. Now, each surviving leaf has its connection to its parent in tact with probability $(1-\epsilon) \cdot (1-\delta)$: both the parent, and the link to the parent, must survive. Thus, with high probability, only about a $(1-\epsilon) \cdot (1-\delta)$–fraction of the successful leaves will receive the data with high probability, if we aim to keep the overhead low.