Scalable Resilient Media Streaming *

Suman Banerjee, Seungjoon Lee, Ryan Braud, Bobby Bhattacharjee, Aravind Srinivasan

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. SRMS leverages a probabilistic loss recovery technique to provide high data delivery guarantees even under large network losses and overlay node failures. 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 Real-time 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 a very dynamic network (up to five membership changes per minute).

Categories and Subject Descriptors: C.2.2 Network Protocols — Applications

General Terms: Design, Experimentation

Keywords: media streaming, overlay network, multicast, resilience

1. 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-layer multicast delivery protocol or media format. SRMS incorporates a protocol-independent loss recovery technique called Probabilistic Resilient Multicast (PRM) [3], which permits high data delivery ratios even under high network losses and node failures. The SRMS architecture logically admits media transcoding for handling clients with disparate access bandwidths. We describe a full implementation of the SRMS architecture, including wide-area deployment with over one hundred simultaneous clients. We believe this paper presents the first implementation experience 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) [7, 9, 2, 6]. Logically, the end-hosts form an overlay network, and the eventual data delivery path in application-layer multicast is an overlay tree. 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 [2]. Our choice was based on the following: (1) NICE achieves good best-effort delivery ratios [2], (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. Although data packets can be lost in the network level (e.g. due to congestion), this sort of loss recovery is relatively easy and can be handled using retransmissions or FEC. In contrast, 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. This outage duration can be on the order of tens of seconds (e.g. the Narada application-layer multicast protocol [7] sets default timeouts between 30-60 seconds).

In this paper, we describe SRMS, the first implementation of a *resilient* video delivery system based on application-layer multicast. Using PRM, the probabilistic loss recovery technique, SRMS is able to achieve high data delivery ratios even with frequent overlay node failures. 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.

Roadmap. In Section 2 we present an architectural overview of the SRMS system. In Section 3 we describe the implementation of the system. In Section 4 we report the experiment results with SRMS on an emulated network as well as a wide-area testbed. In Section 5 we describe some related work and present our conclusions in Section 6.

* S. Banerjee is with the Department of Computer Sciences, University of Wisconsin, Madison, WI 53706 USA. S. Lee, R. Braud, B. Bhattacharjee, and A. Srinivasan are with the Department of Computer Science, University of Maryland, College Park, MD 20742, USA. B. Bhattacharjee and A. Srinivasan are also with the Institute for Advanced Computer Studies, University of Maryland. S. Banerjee, S. Lee, R. Braud, and B. Bhattacharjee were supported in part by NSF award ANI 0092806. A. Srinivasan was supported in part by NSF Award CCR-0208005.

NOSSDAV'04, June 16–18, 2004, Cork, Ireland.

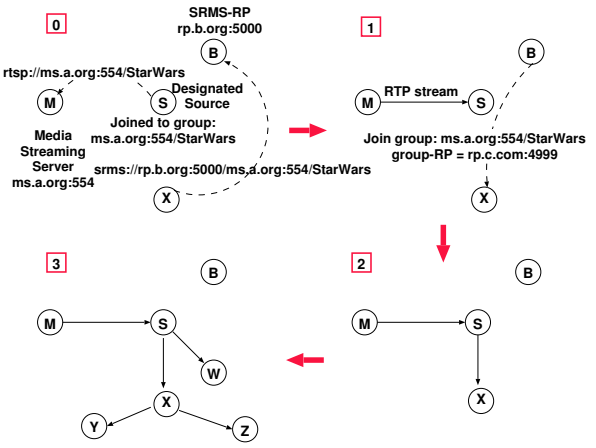

Figure 1: Architectural overview of the SRMS System.

2. SRMS ARCHITECTURE

A SRMS system comprises of the SRMS Rendezvous Point (SRMS-RP) and a set of receivers. Each media stream served by the SRMS system requires a separate multicast group. 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. This construction allows any media streaming server to interoperate with the SRMS system. We defer the discussion of specific protocol issues to Section 3.

In SRMS any client can potentially operate as a source to the multicast group. However, in a typical deployment, we expect the service provider to designate a specific host as the source (which will be co-located with the media streaming server and the SRMS-RP). Otherwise, the SRMS-RP will select a source among the existing clients. In this paper, we focus on the first scenario.

In Figure 1 we show a typical sequence of operations of the SRMS system. We define a new application protocol, `srms`, 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. (The syntax is intentionally similar to the Real-time Streaming Protocol (RTSP) [19] URL format.)

In Panel 0, Figure 1, a client, X, requests the following URL: `srms://rp.b.org:5000/ms.a.org:554/StarWars`. This identifies SRMS-RP at `rp.b.org:5000`, the streaming server at `ms.a.org:554`, and the `StarWars` media stream. The SRMS-RP instructs X to join the application-layer multicast group identified as `ms.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 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. This logical separation 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, has already joined the corresponding application-layer multicast group. It has also contacted the media streaming server, M, using the URL: `rtsp://ms.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 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.

3. IMPLEMENTATION OF SRMS SYSTEM

We now describe the implementation of the SRMS system. SRMS consists of the SRMS-RP and a set of receivers. On receiving a media stream request using the `srms` protocol, the SRMS-RP directs the client to the appropriate application-layer multicast group on which it can receive the media stream. While SRMS can be implemented using any application-layer multicast protocol, for reasons explained in Section 1 we use the NICE protocol enhanced by PRM to construct the application-layer multicast data paths. The implementation of the SRMS-RP is relatively straightforward. Therefore we focus on the client implementation in this paper.

3.1 PRM-enhanced NICE

PRM [3] is a loss recovery technique for resilient application-layer multicast. PRM uses *randomized forwarding*, which is a novel proactive technique that quickly recovers lost data due to overlay node failures. In PRM, 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). Such a construction interconnects the data delivery tree with some cross edges and is responsible for fast data recovery. Randomized forwarding operates in conjunction with the usual data forwarding mechanisms along the tree edges, and may lead to a small number of duplicate packet deliveries. Due to our choice of parameters, these duplicates contribute to less than 5% data overheads in SRMS in comparison to a non-resilient, best-effort scheme. Clients in the group detect and suppress such duplicates using sequence numbers. Additionally, clients in PRM can also detect gaps in sequence numbers and initiate Negative Acknowledgment (NAK)-based recovery, a well-known reactive technique [17]. Such NAKs enable recovery from network losses only, and randomized forwarding is essential for clients to recover data losses due to node failures in the multicast tree. More detailed description of PRM can be found in [3].

Integration of NICE and PRM

Our current SRMS implementation uses NICE application-layer multicast [2] for the underlying data delivery path. We also added the PRM extensions to NICE, which is less than 500 lines of C code. One of the key requirements of PRM is the ability to forward data to a few other overlay nodes, chosen uniformly at random. Therefore in our PRM extension, we let each overlay node periodically discover a set of random other nodes on the overlay as described in [3] [1]. To implement triggered NAKs, each overlay node 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 detects missing packets using the gaps in the received sequence and sends NAKs to the previous node to request the appropriate retransmissions.

[1]If an overlay construction protocol like Narada [7] was used, no such discovery mechanism would be needed because in Narada each node maintains state information about all other nodes.

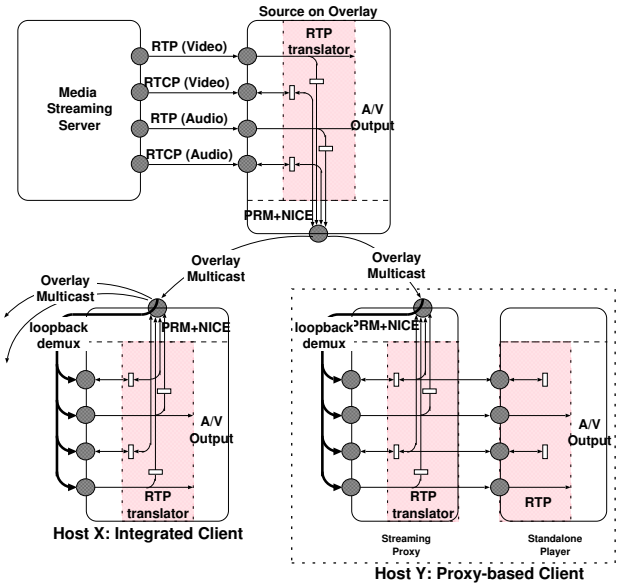

Figure 2: RTP and RTCP paths in SRMS.

3.2 RTP and Designated Source

To transport encoded media, SRMS uses the Real-time Transport Protocol (RTP) [18]. The data component of RTP carries encoded media in the payload, timing and synchronization information and the source identifier. The control component of RTP is called Real-time Transfer Control Protocol (RTCP) [18] and performs a variety of related control operations (e.g. quality of service feedback from receivers, synchronization of different media streams).

In conformance with the RTP standards [18], the media streaming server sends the content using two separate RTP streams, one for audio and one for video. Also, each RTP stream has an accompanying RTCP stream. In the SRMS system the designated source receives these streams in four different ports. Then, it multiplexes the RTP data packets (transcoded if necessary) and the re-generated RTCP packets onto a single overlay multicast port, which are forwarded to receivers along the overlay delivery tree (Figure 2).

3.3 SRMS-client

A SRMS-client has three logical components (Figure 2):

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

Each client receives the RTP and RTCP packets through the single overlay multicast port. The overlay multicast code delivers the packet to the appropriate RTP or RTCP port internally.

The SRMS-client can be implemented in two different ways. An *integrated client* implements all the three components in a single process (Host X in Figure 2). 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 can 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.

On the other hand, a *proxy-based client* needs two processes that together serve as a single client (Host Y in Figure 2). One process called *streaming proxy* implements the overlay multicast and the RTP translator functionalities. Additionally it forwards a copy of the received RTP and RTCP packets to the stand-alone media player. In this scenario, RTP functionality is replicated in both the player and the proxy. While such a construction is comparatively inefficient, it decouples the implementation of the media player from the rest of the SRMS-system. This implementation gives a user the flexibility to use off-the-shelf media player binaries as part of the SRMS system.

3.4 RTP Translation

RTP enables application sources to perform quality of service adaptations by defining mechanisms for receivers to send appropriate feedback. Based on sequence numbers in received data packets, receivers infer loss rates on the data path, and periodically send Receiver Report (RR) RTCP packets to the source. The source application can use this feedback to appropriately adapt the data rate. Efficient implementations of media transcoding and compression can be found in [1, 20].

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 for more subtle rate adaptation. In SRMS, we treat each client on the overlay data delivery path as a potential *RTP translator* [18], and data transcoding is performed at the granularity of overlay hops based on bandwidths available to individual clients. As a result clients are not constrained to the minimum bandwidth on the entire network, but are able to receive data at the maximum permissible bandwidth on the path to the source. Each overlay hop in SRMS is treated as an independent RTP session. Some RTCP packets (e.g. RR packets) carry control information meaningful only to such individual RTP sessions, and they are not forwarded to the entire data delivery tree.

Interaction of RTP translation and PRM

In PRM randomized forwarding, the media encoding in the payload of the forwarded packet may potentially be inconsistent with the media encoding at the receiver due to intermediate RTP translations. We describe solutions to such a case as well as data rate adaptation mechanisms in SRMS in a Technical Report [4].

4. EXPERIMENTAL RESULTS

In this section we focus on the results from our implementation of the SRMS system. In contrast to the simulation study in [3], the results presented here demonstrate the performance of PRM when integrated with real media streaming applications on a more realistic environment.

4.1 Experiment Testbed

Our experiments were performed in two different environments [23]: (1) a publicly available emulated network (Emulab in University of Utah), and (2) a public wide-area testbed (MIT's RON testbed which forms a distributed version of the Emulab). In all our experiments, we streamed multimedia data from the Darwin streaming media server [2]. The server streamed a four minute MPEG4 encoded

[2]publicly available at *http://www.apple.com/quicktime/products/qtss/*

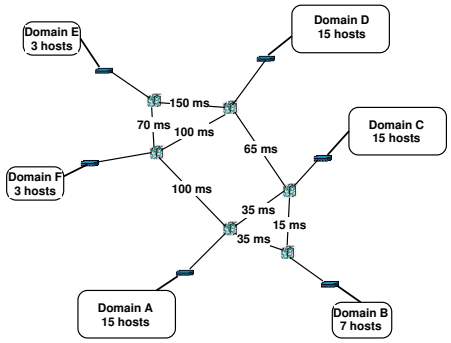

Figure 3: The network topology used in the emulated network environment. Each backbone link is marked with the average latency. The losses on the links connecting the access gateways and the backbone were chosen randomly between 1% and 2%. Within each domain the latencies between pairs of hosts were randomly assigned between 5 and 10 ms, and the corresponding losses were 0.2% to 0.5%.

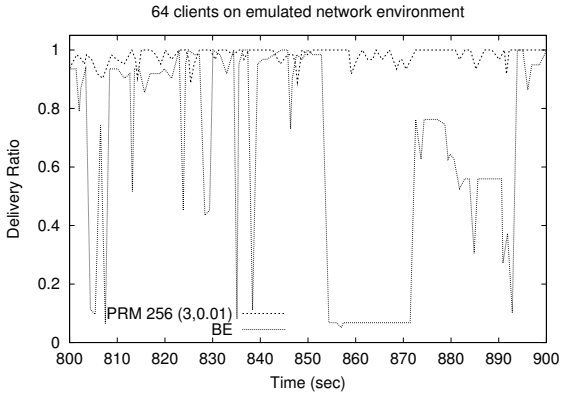

Figure 4: Data delivery ratio achieved for a group of 64 clients as they join and leave the multicast group. Overlay node failure and join rate was 4.8 per minute. Between time 850 and 860 seconds four intermediate overlay nodes on the data delivery tree left the multicast group.

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 1: Comparison of data delivery ratio for different overlay node failure rates. The average number of overlay nodes was 64 and the experiment duration was 30 minutes.

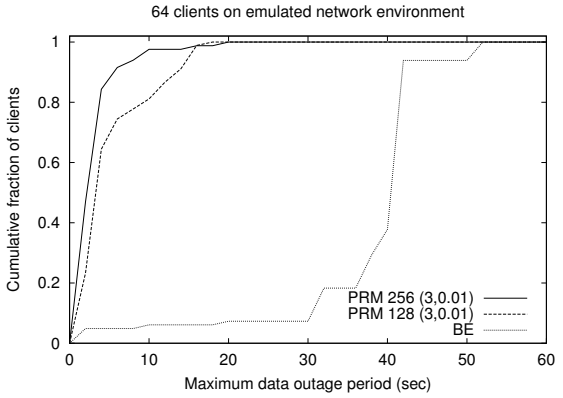

Figure 5: The cumulative distribution of the largest data outage period seen by all the clients for a 64 client experiment with overlay node failure and join rate of 4.8 per minute.

movie to the clients using SRMS. Each experiment lasted between 15 minutes and one hour. 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 average bandwidth of the media stream was about 250 Kbps.

For the emulated experiments, we modeled a group of clients distributed geographically in different parts of the world as shown in Figure 3. The choice of parameters were based on ping measurements we performed on the Internet. 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. More details about the emulated and wide-area testbeds can be found in [23].

4.2 Experiment Scenarios

We have evaluated the performance of the SRMS system for a range of different system parameters, e.g. group sizes up to 128, with different join-leave patterns. In these experiments, all departures of clients were modeled as "ungraceful leaves," 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 rates for members are chosen to be equal so that the average size of the group remained nearly constant. 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 β denotes the probability of forwarding to each of these random neighbors.

In these results we compare the performance of SRMS-based resilient media streaming to a baseline overlay media streaming system without resilience (e.g. the basic NICE application-layer multicast protocol [2]).

4.3 Experiments on Emulated Network

We first describe results from experiments performed on the emulated network environment using the topology in Figure 3.

Resilience

We measure resilience by data delivery ratio — the fraction of packets successfully delivered to receivers out of all packets transmitted by the source. In Table 1 we show the average delivery ratio of SRMS. For example, with 4.8 node failures and joins per minute, SRMS delivers 98% of data packets while the best-effort scheme achieves 72% data delivery. We can also observe that a longer bitmask (for NAK-based retransmissions) leads to better performance.

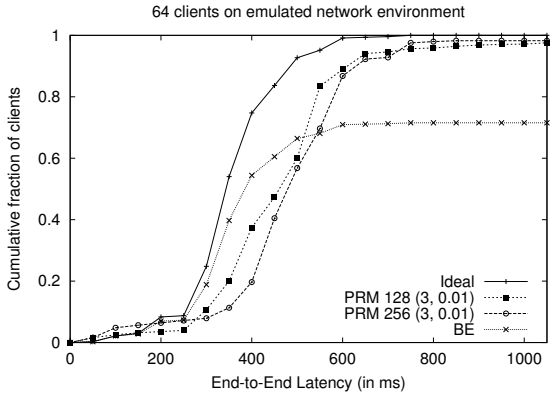

Figure 6: Distribution of latency experienced at the clients for a group of 64 nodes (on average) with overlay node failure rate and join rate of 4.8 per minute. Ideal marks the latency distribution for a hypothetical scenario when the overlay paths to the clients are instantaneously repaired following a failure or a join, i.e. there are no data losses on the overlay path.

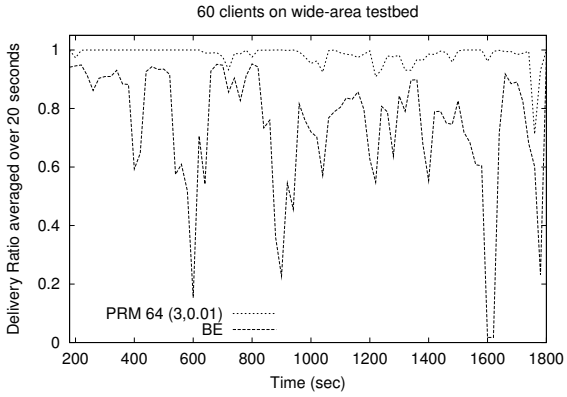

Figure 7: Data delivery ratio achieved by a group of 60 clients (on average) on the wide-area testbed for a 30 minute experiment. The media stream was started three minutes into the experiment. The data delivery ratio is averaged over each 20 second interval for clarity. The overlay node failure and join rate was 4.8 per minute each.

We now present a detailed 100-second snapshot of data delivery ratio from a specific experiment (with node failure and join rate of 4.8 per minute) in Figure 4. Both SRMS and best-effort schemes used the same join-leave pattern. Unlike the best-effort scheme, the PRM-based scheme maintains a high data delivery ratio ($> 95\%$) for all receivers at all times. For example, four intermediate overlay nodes departed from the group between time 850 and 860 seconds. The effect of these departures was quite severe for the best-effort case and the data delivery ratio decreases to less than 10%, while the performance of SRMS was largely unaffected.

In Figure 5 we plot the cumulative distribution of the maximum data outage period experienced by the different clients in the same experiment as in Figure 4. 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. This is a significant improvement over the best-effort case, where more than 90% of them experience data outages of 30 seconds or more.

The data overheads for SRMS in all these experiments, due to data duplication in randomized forwarding, was 3%. This follows from the parameter choice of PRM: each client chose three other random clients ($r = 3$) and forwarded data to them with probability, $\beta = 0.01$.

Latency

In Figure 6 we plot the distribution of overlay latency experienced by different clients in the same experiment (with 4.8 failures and joins per minute). The best-effort schemes delivers all data using the single overlay path along the overlay tree. In contrast, path lengths in SRMS are longer — some of the clients receive the data along potentially longer paths (traversing random overlay hops). For example, 64% of the clients observe data latencies of up to 500 ms in the best-effort case, while about 58% of the clients observe the same latency bound in the case of PRM schemes. However, as shown in Table 1, this marginally higher overlay latency in SRMS allows significantly higher data delivery ratio (98% for PRM, 72% for best-effort).

Lastly, we report that the control overheads (typically less than 2 Kbps at each overlay node) was essentially insignificant compared to the data rate.

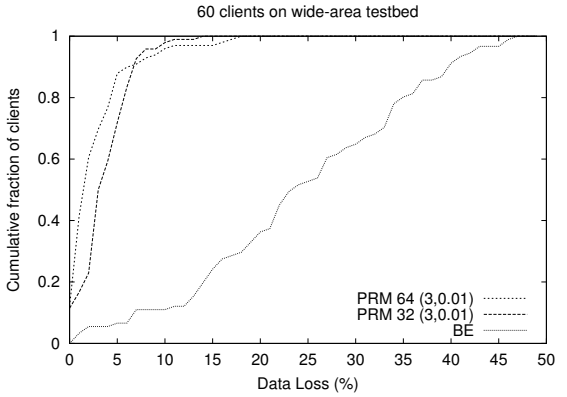

Figure 8: Cumulative distribution of data losses for a 30 minute experiment on a group with 60 clients (on average) performed on the wide-area testbed. The overlay node failure and join rate was 4.8 per minute each.

4.4 Wide-area Experiments

The wide-area experiment was performed using 60 clients distributed across 22 hosts on MIT's Resilient Overlay Networks testbed. The Darwin server and the designated source was co-located in a single host (located in the US) and the 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 reduced the data rate sent out from the Darwin server to about 32 Kbps. We also used correspondingly smaller bitmasks for NAK based retransmissions. As before, the overlay node failure and join rate was 4.8 per minute.

In Figure 7 we show the data delivery ratio achieved over the entire duration of this experiment. 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 8 we show the cumulative distribution of data that was lost at the different clients for the same experiment. We can observe that for the PRM-based system (with a 64 bit bit-

mask) 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 system, in which about 50% of the clients experience more than 20% data loss.

5. RELATED WORK

A large number of research efforts (IVS [22], Rendez Vous [3], vic, vat [4], rat [5], CUSeeMe [6], 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 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.

On the other hand, a number of projects have addressed the problem of constructing efficient data delivery paths for application-layer multicast [7, 9, 2, 6]. Of these, the Narada protocol [7] 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 in the way PRM and SRMS does. The Overcast protocol [11] 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.

A large number of research proposals have addressed reliable delivery for multicast data, most notably in the context of network-layer multicast [8, 17, 14, 13]. In contrast to the PRM approach used in SRMS, all these techniques employ reactive mechanisms for providing data reliability and therefore incurs moderate or high delivery latencies. A comparative survey of these protocols is given in [12] and [21]. On the other hand, proactive schemes using redundant data encoding (e.g. FECs) have been used to provide multicast data reliability [10, 16, 5, 15]. 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. We presented a detailed comparison of these reliability mechanisms (including PRM) in [3].

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. To the best of our knowledge the SRMS system is the first application-layer multicast based media streaming application that explicitly addresses resilience. Second, all the network-layer multicast based schemes employ completely reactive mechanisms for providing data reliability and therefore incurs moderate or high delivery latencies. In contrast, our proactive mechanism, (i.e. randomized forwarding) significantly improves resilience for applications that require low latency data delivery. SRMS also defines a framework in which various bandwidth adaptation techniques [1, 14] can be applied within the context of media streaming to user groups.

6. 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 such that it can interoperate with existing tools for media streaming and playback. Based on the PRM loss resilience scheme, SRMS is able to achieve very high data delivery ratios in spite of network losses and overlay node failures. 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.

[3]Rendez Vous is available at www.lyonnet.org/IVStng

[4]Both vic and vat are available at www-nrg.ee.lbl.gov

[5]http://www-mice.cs.ucl.ac.uk/multimedia/software/rat/index.html

[6]CUSeeMe is currently available commercially at www.fvc.com

7. REFERENCES

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

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

[3] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan. Resilient multicast using overlays. *ACM Sigmetrics*, June 2003.

[4] S. Banerjee, S. Lee, B. Bhattacharjee, A. Srinivasan, and R. Braud. Scalable resilient media streaming. *CS-TR 4482, University of Maryland, College Park. http://www.cs.umd.edu/projects/nice/papers/cs-tr-4482.pdf*, May 2003.

[5] 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.

[6] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron. SCRIBE: A large-scale and decentralized application-level multicast infrastructure. *IEEE JSAC*, 20(8), Oct. 2002.

[7] 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.

[8] 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.

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

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

[11] J. Jannotti, D. Gifford, K. Johnson, M. Kaashoek, and J. O'Toole. Overcast: Reliable Multicasting with an Overlay Network. In *Proc. OSDI*, Oct. 2000.

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

[13] 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.

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

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

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

[17] 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.

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

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

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

[21] 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.

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

[23] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An integrated experimental environment for distributed systems and networks. In *Proc. of the Fifth Symposium on Operating Systems Design an d Implementation*, pages 255–270, Boston, MA, Dec. 2002. USENIX Association.