

The Case for a Multicast Session Layer

Andrew Swan Lawrence A. Rowe

Computer Science Division

University of California, Berkeley

{aswan,rowe}@cs.berkeley.edu

Abstract

The past decade has seen a great deal of effort invested in developing multicast technology for the Internet. Unfortunately, early excitement about multicast has given way to frustration with the challenges of designing and deploying a scalable multicast service. The reaction to the slow deployment of multicast has been an explosion in different implementation techniques designed for specific applications. None of these new services have been widely deployed. As a result, multicast technology is complex and confusing to application writers. To address this complexity, we argue for development of a multicast session layer. By decoupling the service model and programming interface from the underlying implementation, this new layer simultaneously simplifies the development of multicast applications and eases the deployment of new multicast technology. We outline the case for a multicast session layer and discuss principles for its design.

1 Introduction

Most observers recognize the potential of using multicast to deliver data efficiently to multiple destinations. Sadly, these advantages have not been realized in the Internet because no protocol has emerged that satisfies the requirements of both application writers and network operators. The first widely deployed protocol was IP Multicast. While early implementations based on *flooding and pruning* were developed by researchers in the early 1990's, this protocol did not scale as usage increased. New protocols were developed to address these shortcomings but none addressed all the scaling issues. Consequently, in the past few years researchers have defined new service models and protocols to exploit the advantages of multicast.

Some multicast protocols are well-matched to a specific application (e.g., Single Source Multicast is ideal for large-scale 1-to-many applications such as streaming radio or television) but are not general enough to implement all multicast applications. Another example is the use of application-level multicast to implement peer-to-peer and small-scale video conferencing applications. This proliferation of narrowly focused protocols has confused application developers and inhibited deployment and use of

multicast technology.

An application developer today has several protocols to choose between when implementing a multicast application:

- Any Source Multicast (ASM) offers a general service model that can be used to build a range of applications but suffers from serious implementation problems that have limited its deployment.
- Single Source Multicast (SSM) addresses many of the implementation problems in ASM and, as a result, is likely to be widely deployed. However, as the name suggests, the SSM service model provides for only one sender per session, so applications that require multiple senders must either use a different protocol or implement extra functionality on top of SSM.
- Application-level Multicast (ALM) avoids the deployment problems of ASM and SSM by moving multicast forwarding from the network to end hosts. As a result, ALM offers more flexibility than ASM or SSM but is less efficient and does not scale well.

Few applications are well-suited for just one of these protocols. Instead, applications demand a rich service model that can be mapped to different network-level implementation techniques as necessary. We propose that a new multicast session layer protocol be defined to support this functionality. A session layer protocol sits on top of lower layer protocols (e.g., UDP at the transport layer and IP Multicast at the network layer) to provide more sophisticated services to applications than the underlying protocols.

An obvious benefit of adding this new layer is that it simplifies the job of writing a multicast application. The application developer no longer needs to understand the details of and tradeoffs between the different techniques available. Moreover, a single application may require the use of different network-layer protocols depending on how it is used or the application may combine different protocols. The rules and algorithms to choose between and combine different protocols will likely be complex. Localizing them in a reusable session-layer protocol simultaneously simplifies the task of writing a new application

while offering a richer service. Another benefit of adding a new multicast session layer is that new network-layer multicast protocols and technologies can be seamlessly deployed by updating the session layer, eliminating the need to modify existing applications.

Based on the past decade of experience with IP Multicast, we believe a major shortcoming of the ASM service model is that it places many demands on the network. Yet, the service model offers applications no way to express their requirements which may influence how the multicast service is implemented. As a result, network-level multicast protocols are unnecessarily complex and inefficient. We propose that the interface to the new session layer permit, or even require, applications to convey their service requirements and important session parameters to guide the implementation.

This paper presents the case for the development of a multicast session layer and suggests principles for the design of its interface. Section 2 briefly summarizes the evolution of multicast protocols (i.e., ASM, SSM, and ALM) and discusses the strengths and weaknesses of each protocol. Section 3 describes our assumptions about the network-level multicast facilities that will be available to implement the session layer. Section 4 surveys some multicast applications and discusses their characteristics and appropriate implementations. Lastly, Section 5 discusses some issues surrounding the design of the multicast session layer.

2 Multicast Protocol Background

The basis for ASM is the host group service model developed and implemented by Deering[5]. In the host group model, any host can dynamically join or leave a multicast session and any host can send to a session at any time. The host group model places the burden of choosing the implementation for a particular multicast session on the network itself. However, the interface between applications and the network is, by design, very narrow. As a result, the best the network can do is to observe an application's behavior and guess what implementation is best based on its observations.

While this architecture is simple and elegant for application writers, it has proven to be impractical. For example, in a multicast session with multiple senders, the multicast infrastructure must decide whether to construct a single shared distribution tree for all senders or multiple trees rooted at individual senders (or some combination thereof).¹ Ideally the decision of whether to use a shared tree or a source-rooted tree will take into account global parameters such as the total number of senders in a ses-

¹A multicast distribution tree is a logical structure over which packets are forwarded to reach receivers in a multicast session. The nodes of the tree are typically routers or hosts arranged in an overlay.

sion and the application's sensitivity to latency. However, a scalable multicast routing protocol that implements the ASM service model must not be centralized so this decision must be made by individual routers. Each router observes only the local behavior of a multicast session so no router has the complete information needed to make an optimal decision.

As another example, the ASM service model requires no signalling from an application (i.e., session set-up) before sending data to a multicast session. At the same time, the network must discover new sources and promptly include them in a distribution tree. There are two basic approaches to the problem of source discovery. One approach is to identify a unique Rendezvous Point (RP) for each multicast group and send all packets from all senders to the RP. This approach was rejected by network operators since it forces them to rely on a third party for correct multicast operation when the RP is located in a network controlled by a different organization.

Another approach to sender discovery allows each network to maintain an RP and to announce new senders to remote networks (i.e., to the RPs in other networks). This design is the foundation of the Multicast Source Discovery Protocol (MSDP)[6]. Engineering robust protocols dictates the use of soft state in which state is discarded unless it is periodically refreshed. But the absence of signalling in ASM leaves no way to maintain an up-to-date list of senders in a multicast session. Instead, the network must react quickly to data packets from previously unknown senders. As a result, if an application sends data less frequently than the timeout interval in the underlying routing protocol, the protocol will either work poorly or require extra effort to serve such applications. This *bursty source* problem has hampered MSDP since its conception.

Another problem with ASM is that of multicast address allocation. Each ASM session must be allocated a unique group address out of a pool of only 2^{28} available addresses. This constraint places a severe limit on the total number of sessions that can be simultaneously active given the size of the Internet. The lack of a reasonable scheme for global multicast address allocation compounds this problem.

To address the problems of source discovery and address allocation, Holbrook[10] and others proposed SSM. SSM provides an alternative multicast service model in which sessions are constrained to have just one sender. By including the address of the sender in the multicast session address, the network no longer needs to solve the source discovery problem. Of course, the burden is simply moved to the application since because addresses must now be given explicitly when joining a session. The address allocation problem is also solved because SSM offers a large pool of multicast sessions (2^{24}) per source host. As a result, the total number of simultaneous SSM sessions is much larger and each host can autonomously allocate addresses for sessions whose source is on that host.

In his thesis, Holbrook outlined several techniques to provide a richer service model (i.e., multiple sources) on top of a network-level SSM service[11]. A recent project by Hoerdt, Beck, and Pansiot has developed a detailed protocol for one of those techniques[9]. Their goal is to emulate the existing ASM interface and unlike our proposal, uses a single implementation for all applications.

Another response to the problems with ASM is Francis’s work on a system he initially called Yallcast and later renamed Yoid[7]. Yoid aims to provide a general content distribution service. A major part of Yoid is a scalable application-level multicast service that can exploit network-level multicast where it is available but does not require it. However, Yoid is also designed to use permanent storage at forwarding nodes so that large data sets (e.g., Usenet news, WWW mirrors, etc.) can be distributed even if all interested parties cannot participate in a multicast session simultaneously. Our goals are more modest — to provide a working multicast service that can be used by a range of multicast applications. Although the initial document outlining the design of Yoid[7] has a number of new and promising ideas, the project apparently was abandoned before these ideas were pursued.

Several other ALM schemes have been proposed recently[12, 2, 13, 14]. These protocols are designed to use only unicast and hence cannot exploit the advantages of network-level multicast where it is available.

3 Network Model

This section discusses our assumptions about how network-level multicast will evolve in the future and what services will be available to applications as a result.

We assume that the unresolved challenges of interdomain ASM will prevent wide-scale deployment. However, within a limited area (i.e., on a single LAN or within a larger network operated by a single organization), interdomain source discovery is not an issue and the problem of address allocation is mitigated. As a result, we expect ASM may be deployed within small, isolated parts of the Internet.

As for SSM, we assume it will eventually be widely available for interdomain multicast. The protocols needed to support SSM are already widely available in routers. The major barrier to widespread SSM deployment today appears to be support in host operating systems. Although support for SSM in switches is another nagging problem², we do not believe it will inhibit the eventual deployment of SSM. SSM has addressed the major architectural problems with ASM and as a result, we expect

²Without special support in switches, multicast is flooded to all ports on a switch. Some switches support IP multicast by “snooping” on IP-level multicast signalling (i.e., IGMP). But, as IGMP has grown more complex, IGMP snooping is less feasible, particularly in low-end switches. Other solutions have been proposed (e.g., CGMP, GMRP) but none is yet widely used.

it will be widely deployed.

Finally, we assume neither SSM nor ASM will be ubiquitous. That is, although some form of network-level multicast may be available in many parts of the Internet, there will always be some endpoints that have neither SSM nor ASM available. Ideally, our new multicast session layer should accommodate these endpoints, with the understanding that application-level forwarding, which will be required to reach these nodes, implies less efficient transmission and extra overhead. Including these endpoints addresses the chicken-and-egg problem with multicast deployment. Applications can be deployed before the network supports multicast but when network-level multicast is deployed, applications benefit from the improved efficiency immediately and transparently.

4 Applications

Many attributes distinguish multicast applications, including the total number of participants per session, the number of senders and receivers, and performance requirements such as latency, throughput, and packet loss. This section surveys some multicast applications and their characteristics to illustrate the diversity of requirements.

One obvious multicast application is distribution of the same content to many simultaneous receivers. The content might be bulk data transmitted reliably via digital fountain coding and framing[3] or time sensitive continuous media (i.e., audio/video). In either case, a session contains a single sender and a potentially very large set of receivers. We refer to these applications as *1-to-many* applications.

SSM is ideal for, and indeed was motivated by, such applications. Even if a global ASM service was available, it is a poor fit for a 1-to-many application for two reasons. First, it requires global multicast address allocation as discussed above. And second, ASM offers a content publisher no protection against a rogue sender intent on disrupting a session by sending its own, unwanted traffic.

Another natural application of multicast is *multiparty collaboration*. Multiparty collaboration encompasses several multicast applications with different needs. Video conferencing is an important collaboration application, but a richer environment may also include other tools such as a shared workspace application (e.g., shared whiteboard or text editor) or a distributed presentation application (e.g., PowerPoint, synchronized web browsing, or a distributed visualization system). These applications have slightly different requirements — audio and video conferencing demand low latency for effective communication. The other tools cited have looser bounds on latency but often require reliable transmission.

Regardless of the specific tools being used, collaboration applications can be used in any of several different

scenarios. A small collaborative session might contain just a few participants, perhaps as few as two. We refer to these applications as *small scale n-way* applications. In a small session, network-level multicast is not very compelling, particularly if the participants are physically distant from each other. The overhead induced by using application-level forwarding in such a session (in both bandwidth consumption and latency) is minimal. In addition, by using application-level forwarding an application can avoid nagging problems with network-level multicast deployment. Moving the forwarding state into the end hosts also reduces the amount of state that each multicast router must maintain.

In addition to small N-way conferences, multiparty collaboration tools are used in larger settings. For instance, the Access Grid[1] (AG) is a suite of collaboration tools that support room-level and desktop interaction in groups with as many as several hundred endpoints. This is an example of a *large scale n-way* application. In such a large session, the benefits of network-level multicast are obvious. Given the problems with wide-area ASM deployment, SSM is a logical choice for the AG. However, AG applications have multiple sources so applications must either establish separate SSM trees rooted at each sender or share an SSM tree between multiple senders by “back-hauling” data from each sender to a designated tree root.

The decision about how many trees to use and how to share them depends on the dynamics of the session. For instance, in an interactive meeting with several active participants, it makes sense to use individual trees rooted at each sender to minimize latency. Alternatively, consider a lecture in which one site is the primary source but participants at other sites occasionally ask questions. In this case, remote sites participate infrequently and only briefly. In addition, there is minimal interaction between different sites so some extra latency can be tolerated. These facts suggest that in this scenario it is more convenient for remote participants to send directly to the primary sender who can then resend to all participants via the multicast distribution tree that is already established.

The underlying network topology also influences the decision. If two sites are close to each other (topologically) then the cost in added latency and bandwidth consumption of using a single distribution tree and forwarding all traffic to one site is minimal. On the other hand, if the sites are far apart (e.g., if they are on separate continents), it is appropriate to use separate distribution trees for each geographic (or topological) region.

Regardless of how trees are shared, each receiver must dynamically locate all sending nodes in order to join the appropriate SSM trees. At first glance, this problem of locating sending nodes may seem like a major challenge. However, in the AG case, endpoints join existing conferences through a centralized “venue server” which can easily keep track of all the participants currently in the

session. Hence, in this particular case, there is no need for the network or multicast session layer to locate sources — the application already knows where the sources are and just needs a way to convey that information to the session layer.

Large scale 1-to-many applications and multiparty collaboration have been the most successful multicast applications on the Internet thus far but there are numerous other applications well-suited to multicast that have not used it, primarily due to the lack of deployment.

In multiplayer games and distributed interactive simulations (DIS), participants in a shared environment must constantly update their neighbors as they move around or alter the environment. These applications typically use a centralized server to collect updates and then redistribute them to participants. This architecture simplifies the implementation since a centralized server can easily serialize events and resolve conflicts. Multicast can greatly improve the scalability and robustness of these applications when it is possible to process events in a completely distributed fashion[15].

Another example of a multicast application is a service that broadcasts a continuous stream of information (e.g., sports scores, stock quotes, etc.) to multiple simultaneous clients. Such “push” technology was trendy several years ago via services such as PointCast. Although PointCast’s technology was never widely deployed, similar services are growing rapidly today. They could be optimized to greatly reduce the load on content servers as well as bandwidth consumption by using multicast.

The applications outlined above illustrate some of the implementation techniques available for multicast applications and the factors that influence the choice. Our goal is to simplify application development and improve the utility of multicast protocols by moving the decision process as well as the actual implementation of the multicast service into the multicast session layer.

The benefits offered by this new layer are many. For example, the programming interface for multicast applications changes from today’s balkanized world of ASM, SSM, and ALM to a single interface that offers a rich service to a broad range of applications. Furthermore, common functionality such as sharing a single SSM tree between multiple senders and application-level forwarding to reach receivers without multicast connectivity is localized in a single protocol module that can be improved and updated independent of applications.

5 Implementation Issues

In this section we discuss some general issues surrounding the design and implementation of a multicast session layer.

5.1 Application Hints and Requirements

As alluded to above, we believe that the pragmatic interface to the multicast session layer should be very expressive. It should allow applications to give hints about expected properties of each multicast session. For instance, if an application specifies that there will be one sender, a single SSM tree is clearly an appropriate implementation. On the other hand, if the application indicates that there will be many nodes, all of which will both send and receive data, then the session layer should consider sharing trees in some fashion.

The session layer can, of course, observe an application and infer some of this information. (For instance, PIM-SM uses shared trees by default and if it notices one sender generating a lot of traffic, it may switch the distribution just for that sender to a tree rooted directly at the source.) However, since applications generally have detailed information that can be used at lower layers, it may greatly simplify the lower layers to pass that information explicitly rather than forcing it to be deduced. In any case, having applications explicitly indicate what they know about a session does not preclude the session protocol from using introspection to refine its behavior. Rather, it lets the session layer make a more informed decision about the initial implementation.

The interface should also allow applications to express their requirements. For example, an audio application used interactively in a collaboration environment might specify that low latency is important which would argue for using per-source trees rather than shared trees. On the other hand, a bulk data distribution application (e.g., database replication) might prefer a tree with higher throughput even at the cost of increased latency. Unlike many session parameters outlined above (e.g., the number and locations of senders), an application's requirements cannot be inferred by simply observing the application's behavior. As a result, adding the ability for an application to specify its preferences can enable the session layer to make optimizations that are not possible with the existing, narrow ASM interface.

5.2 Naming

The addition of a new layer provides the opportunity to define a new scheme for naming or addressing multicast sessions. Much like network-layer IP addresses abstract the details of underlying link-layer addressing, the naming at the session layer can hide the details of network-layer multicast addresses from higher layers. By doing so, the session layer can easily incorporate new network-level technologies, even if they use different addressing schemes (e.g., IPv6).

The scheme for naming sessions must be somewhat coupled to the implementation of the session layer protocol since the protocol must be able to map a session name

(i.e., a parameter to a `join_session()` interface routine) to transport-layer parameters for joining the session.

We believe that a good design is for the session name to explicitly include an application-level rendezvous which a new participant must contact as the first step in joining a session. The rendezvous can then inform the joining host of either a network-level multicast group to join or another member of the session to provide application-level forwarding. Although this is a simple transaction, there are still two major drawbacks to using a fixed (per-session) rendezvous: the rendezvous is a single point of failure and it may be a performance bottleneck.

Multiple rendezvous hosts may be specified to mitigate the availability problem. Performance issues may be addressed by using network-level multicast whenever possible for the initial exchange with the rendezvous, falling back to unicast only if network-level multicast is unavailable. Of course, this does not address the case of a large group in which many of the participants are forced to contact the rendezvous via unicast. We have no specific solution to this problem but note that this problem is not addressed by the application-level multicast protocols cited in Section 2. If this turns out to be an important problem, a caching hierarchy could be deployed to reduce the load at the rendezvous.

With this design, session names include transport-layer parameters for contacting the rendezvous as well as an identifier to distinguish different sessions managed by that rendezvous. An obvious syntax for encoding this information is a URL. For instance, consider the following URL:

```
mcast://rendezvous.domain.com:2222/foo
```

To join this group, a host might have the session name (`foo`) to generate a group address and attempt to join an SSM tree with that address rooted at `rendezvous.domain.com`, on UDP port 2222. The rendezvous periodically transmits over that SSM session the addresses of network-level multicast sessions that are part of the larger session and the addresses of other members in the session that can forward the session traffic to new members if necessary. If the joining host cannot join the rendezvous SSM session, it falls back to contacting the rendezvous directly via unicast. In either case, after finding a way to join the session the host no longer needs to remain in contact with the rendezvous. The process outlined above is merely a strawman — there are many ways in which it can be optimized which we plan to study.

5.3 Anycast

Another application of multicast is *anycast*. In the ASM service model, host group addresses provide a level of indirection that, in conjunction with network-based source discovery can be used for applications such as service discovery[4]. Although such applications could use a multicast session layer, these applications would be better

served by defining a separate service layer for anycast. ASM is one implementation that could be used for anycast. As with multicast, separating the programming interface for anycast from its implementation would allow new the underlying services to evolve independently of applications. For instance, it would be expedient to deploy anycast applications using ASM as the implementation but network-level anycast technology could later replace ASM. In this way, no applications use ASM directly so it could later be deprecated in favor of SSM and other newer technologies without requiring extensive changes to existing applications.

5.4 Other Services

The introduction of a multicast session layer offers the opportunity to include richer services than those provided by lower layers. A new service should be added only if it cannot be implemented at a higher layer or if there are significant benefits (e.g., in performance) to including it in the lower layer and it can be implemented efficiently enough that its inclusion does not penalize applications that do not need it.

Examples of services that could be considered for inclusion in the multicast session layer include data reliability and encryption/authentication. These areas need further study.

6 Summary

This paper argues that a multicast session layer should be introduced. The goal of this layer is to provide a higher-level abstraction to a collection of multicast services that will improve the quality of multicast applications, simplify application development, and satisfy the practical constraints of multicast deployment. We are in the process of building a prototype of the session layer protocol outlined in this paper. We hope to perform wide-area experiments soon.

Acknowledgements

This work was funded by a donation from Hewlett-Packard Laboratories and grants from the National Science Foundation Internet Technologies Program (grant 9907994) and Instrumentation Equipment Program (grant 9512332).

References

- [1] Access grid web page.
<http://www.accessgrid.org>.
- [2] Suman Banerjee, Bobby Bhattacharjee, and Srinivasan Parthasarathy. A protocol for scalable application layer multicast. Technical Report CS-TR 4278, University of Maryland, College Park, 2001.
- [3] John Byers, Michael Luby, Michael Mitzenmacher, and Ashu Rege. A digital fountain approach to reliable distribution of bulk data. In *Proceedings of SIGCOMM '98*, Vancouver, Canada, September 1998.
- [4] Steven E. Czerwinski, Ben Y. Zhao, Todd D. Hodes, Anthony D. Joseph, and Randy H. Katz. An architecture for a secure service discovery service. In *Mobile Computing and Networking*, 1999.
- [5] Stephen E. Deering. *Multicast Routing in a Datagram Internetwork*. PhD thesis, Stanford University, December 1991.
- [6] Bill Fenner and David Meyer. *Multicast Source Discovery Protocol (MSDP)*, May 2003. Internet Draft, work in progress.
- [7] Paul Francis. Yoid: Extending the internet multicast architecture. Unpublished draft, April 2000.
- [8] Mickaël Hoerdt, Frederic Beck, and Jean-Jacques Pansiot. *Multi-source communications over SSM networks*, June 2003. Internet Draft, work in progress.
- [9] Hugh Holbrook and David Cheriton. Explicitly requested source-specific multicast: EXPRESS support for large-scale single-source applications. In *Proceedings of SIGCOMM '99*, Cambridge, MA, 1999.
- [10] Hugh W. Holbrook. *A Channel Model for Multicast*. PhD thesis, Stanford University, August 2001.
- [11] Yang hua Chu, Sanjay G. Rao, and Hui Zhang. A case for end system multicast. In *Proceedings of SIGMETRICS 2000*, 2000.
- [12] Mingyan Liu, Rajesh R. Talpade, and Anthony McAuley. AMRoute: Adhoc multicast routing protocol. Technical Report 99-1, University of Maryland Center for Satellite and Hybrid Communications Networks, 1999.
- [13] Dimitrios Pendarakis, Sherlia Shi, Dinesh Verma, and Marcel Waldvogel. ALMI: An application level multicast infrastructure. In *Proceedings of Usenix Symposium on Internet Technologies 2001*, 2001.
- [14] J. Mark Pullen, Michael Myjak, and Christina Bouwens. *RFC 2502: Limitations of Internet Protocol Suite for Distributed Simulation in the Large Multicast Environment*, February 1999.