Using SSM Proxies to Provide Efficient Multiple-Source Multicast Delivery

Daniel Zappala and Aaron Fabbri
Department of Computer Science, 1202 University of Oregon, Eugene OR 97403-1202
zappala|fabbri@cs.uoregon.edu

Abstract— **We consider the possibility that single-source multicast (SSM) will become a universal multicast service, enabling large-scale distribution of content from a few well-known sources to a general audience. Operating under this assumption, we explore the problem of building the traditional IP model of any-source multicast on top of SSM. Toward this end, we design an SSM proxy service that allows any sender to efficiently deliver content to a multicast group. We demonstrate the performance improvements this service offers over standard SSM and describe extensions for access control, dynamic proxy discovery, and multicast proxy distribution.**

I. Introduction

In the past several years, the Internet engineering and research communities have been exploring alternatives to the current IP multicast model. This interest has been fueled largely by deployment delays and the belief that current multicast routing protocols (PIM-SM [1], MSDP [2], and MBGP [3]) are too complex. One promising new approach is Source-Specific Multicast (SSM) [4]. With SSM, a multicast group is associated with a specific source, and that source is the only host that can transmit data to the multicast group. In contrast, the current IP multicast model allows any source to transmit to a group.

By restricting multicast in this manner, SSM has several advantages compared to traditional IP multicast. From a service standpoint, SSM can guarantee a content-provider that it will have exclusive access to its multicast groups. This means, for example, that unauthorized sources could not interrupt a multicast of a live TV event. From an engineering standpoint, SSM simplifies multicast address allocation and eliminates network-layer source discovery. Thus, SSM overcomes two significant barriers to deployment.

Due to this promise, it is conceivable that SSM will become a universal multicast service, enabling large-scale distribution of content from a few well-known sources to a general audience. However, SSM has been designed primarily for applications with one or a few sources who are known in advance to group members, and doesn't work as well for applications with a dynamic or large set of senders. Thus, if SSM becomes widely-supported at the expense of traditional IP multicast, then it becomes paramount to consider how we can build efficient multi-sender multicast applications on top of SSM. Looking at it another way, if we can efficiently support multi-sender applications using an SSM infrastructure, then due to SSM's simplification of address allocation it may be preferable to discontinue using current IP multicast protocols.

This work was supported in part by the National Science Foundation under grants ANI-9977524 and NCR-9714680.

In this paper we present a simple way to achieve this goal through the use of SSM proxies. In our basic approach, an SSM group creator may authorize other nodes to act as proxies for the group content. Group members join only a single SSM channel, rooted either at the group creator or one of the proxies. Any sender can then transmit data to the entire group by unicasting it to the nearest proxy. This proxy relays the data to other proxies, and each proxy delivers the data to its attached group members. All proxies operate at the application level, and utilize SSM as a basic network-layer service.

SSM proxies provide a multicast service that is complementary to standard SSM. Sessions with a dynamic set of senders experience no setup delay, and large numbers of senders can use proxies to achieve both low delay and low routing state. Receivers have added fault tolerance as they may use their choice of proxies for the content. SSM proxies are also backward-compatible with standard SSM, so a single delivery model may be used for all types of multicast applications.

II. SSM Background and Related Work

There has recently been a surge of interest in application-level multicast protocols [5], [6], some of which use proxies to distribute multicast content. However, these protocols make the opposite assumption of this paper, namely that multicast will *not* be generally available, and build application-level protocols on top of unicast. By assuming that SSM is generally available, we are able to more easily solve the problem of configuring proxies and efficiently delivering content.

A. SSM and Single-Source Multicast

SSM is based largely on Holbrook and Cheriton's Express protocol [7]. Both protocols define a multicast *channel* as the combination of a source address S and a group address G. A channel is identified at the network layer as (S, G), meaning the network allows only source S to send data on this channel. Another source S' may send data to the same group address, but this is considered to be a a separate channel (S', G). Likewise, membership is based on channels, so that a receiver joining only channel (S, G) receives data from S but not from S'. If a receiver wants to get data from both sources, it must explicitly join both channels.

One of the major advantages of SSM is that its definition of the channel concept obviates the need for any coordination when assigning multicast addresses. Each host can allocate multicast addresses independently because the group address only needs to be unique to the source. This is in contrast to the current suite of protocols needed to globally allocate addresses along a hierarchical administrative structure [8], [9]. Currently, the IP address range $232. * . * .*$ has been allocated by IANA for single-source multicast applications, so with SSM each host has at least 16 million separate channels it can use.

Another advantage of SSM is that it simplifies the network core, with more complex services implemented at the network edge. With SSM, routers provide only unidirectional, source-specific multicast trees, and all other functionality – address allocation, source discovery, and multiple-source sessions – is implemented at the application level. To participate in SSM sessions, a host uses either IGMPv3 [10] or Cisco's URL-intercept protocol [11].

B. Multiple-Source Multicast

For sessions with multiple senders, Holbrook and Cheriton propose two solutions. In the first, an application uses only one channel, called the session relay. All receivers join this channel to receive data from any sender. Sources then relay their data through the session relay to reach the group members. The advantages of this approach are that routing state is kept to a minimum and the relay can control access to the channel. The disadvantage is that the delay from a sender to a group member may be much larger than if it used the shortest path.

The second approach described by Holbrook and Cheriton is to create a separate channel for each source. This improves the delay from a source to the receivers, but increases the amount of routing state needed to support the application. In addition, receivers need some method for discovering sources for the session, either statically (such as through a web page) or dynamically (such as announcements through a session relay).

III. Using SSM Proxies for Multiple-Source Applications

We are able to bridge the performance gap between a session relay and separate channels by using SSM proxies. A multicast group establishes a set of proxies, each of which is the root of an SSM channel. Group members join the channel for one proxy, and senders likewise transmit their data to one proxy. The proxies then use both application-level protocols and SSM channels to deliver the data to all the group members. Our use of proxies is based on previous work designing a multicast routing protocol with multiple network-level cores [12].

A. Basic Protocol

Each SSM session advertises a primary channel, denoted by (S_1, G_1) and a set of proxies $P_1, P_2 ... P_p$, where $P_1 = S_1$. The primary channel is used for backward-compatibility with standard SSM hosts. Standard SSM receivers join this channel and receive all session data. Likewise, standard SSM senders use this channel as a session relay. Proxies are used to reduce

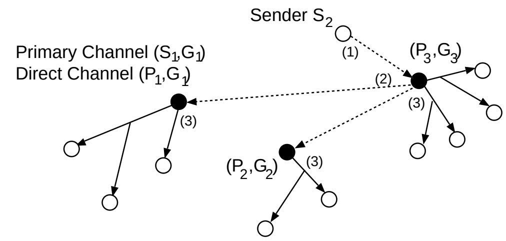

Fig. 1. Basic SSM Proxy Protocol

delay for proxy-enabled hosts that would otherwise use the session relay mechanism. A proxy can be any host and does not necessarily need to be a sender.

Fig. 1 illustrates the basic parts of the SSM proxy protocol. A host joins an SSM session by choosing its nearest proxy P_i and joining that proxy's *direct channel*, denoted (P_i, G_i). The host receives data from all sources over this channel, which is a standard SSM channel rooted at the proxy. Sending data occurs in three steps, as designated in the figure. First, a source chooses the nearest proxy and unicasts its data to this proxy. Next, the proxy relays this data by unicasting it to all other proxies. Note that the primary sender S_1 is equivalent to proxy P_1, so it also receives this data. Finally, each proxy, including the one that originally received the data, delivers the data to the group members on its direct channel. All of the unicast communication can use either UDP or TCP, depending on the application.

One additional protocol mechanism is needed to ensure efficient delivery among the proxies. In some cases, there may be proxies that have not been chosen by any members, for example when there are more proxies than members or proxies that are remotely located. When this is the case, any data sent to the proxy wastes resources, since there are no members who have joined the proxy's direct channel. To avoid this overhead, a so-called *dangling proxy* sends a unicast message to each of the other proxies, notifying them that it has ceased to relay data. The other proxies will not send data to the *dangling proxy* until it re-activates itself.

We assume for now that all proxies are known in advance to all hosts, such as by announcing the proxies and their direct channels on a web page. We discuss dynamic proxy configuration in Section IV. Using static proxy configuration is comparable to how SSM is currently defined, except that only proxies, and not sources, need to be identified. Choosing the nearest proxy can be simplified by separating them on the announcement web page according to region. Alternatively, a host can ping several of the proxies and choose the one with the shortest response time.

Note that neither the senders nor the receivers have to interact with more than one proxy. This has several advantages. First, it allows SSM proxies to remain backwardly-compatible with standard SSM hosts. Standard SSM hosts can send or receive all data through the primary channel. Second, this feature eliminates the case where a sender must replicate its data in order to send it to the proxies. We explicitly want to avoid this scenario so that we may support senders with limited available bandwidth. We also want members to join to only one proxy so that they have control over choosing the best proxy. Finally, our protocol design makes it easier to optimize delivery between proxies.

B. Advantages

Compared to using a session relay, SSM proxies reduce delay because a source can reach local group members directly through a nearby proxy. This is shown in Fig. 1, where sender S_2 sends data directly to local members via proxy P_3, whereas with session relay all data would go through P_1. Clearly, there will be some cases where delay increases when using proxies, but the overall effect is to reduce both the maximum and average delay across all members of the group. The more proxies a session uses, the greater the chance that delay will be close to that of a separate SSM channel.

Using proxies also reduces state compared to using separate SSM channels. Recall that a router must keep a separate routing entry for each channel. Thus if a separate channel is needed for every sender, some routers will have one entry for every source. By using proxies, we reduce this in the worst case to one entry for every proxy. In fact, the worst-case scenario for SSM proxies is rather unlikely to occur as long as members choose their nearest proxy. In this case, the channels will not overlap and routers will need only one entry per group, the same as with session relay.

SSM proxies also eliminate the setup delay incurred when the set of senders is dynamic. When separate channels are used for each sender, a new sender's identity first needs to be advertised through the session relay, and then members need to join its new channel. With SSM proxies, this setup delay is completely eliminated. A new sender simply transmits data to its nearest proxy, and this data is immediately relayed to group members.

Finally, using SSM proxies improves the fault tolerance of SSM delivery. If a member is unsatisfied with its performance, or if its proxy fails, it can simply join the direct channel for a different proxy. If the primary sender fails, the session can continue by using the other proxies.

C. Performance Evaluation

We evaluate the performance of SSM proxies by comparing them to standard SSM using either session relay or separate channels for each sender. For our experiments, we use a static model in which we generate a random graph and choose nodes to act as members and senders. We use both flat and transit-stub random graphs, and vary the number of members, senders, and proxies. For each experiment we measure delay and bandwidth cost, using shortest-path trees as the baseline for comparison across different graphs and groups. We run each experiment until the width of the 95% confidence interval for all metrics is within 10% of the mean value.

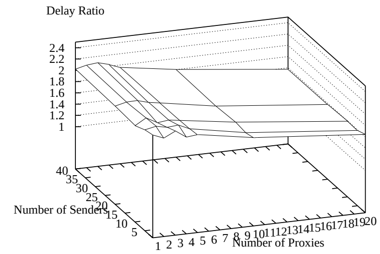

Fig. 2. Avg Delay for SSM Proxies: Random Placement, Flat Network 40 Members

Note that this experiment setup allows us to compare SSM proxies to SSM with either a session relay or with separate channels for each sender. Using a session relay is equivalent to using a single proxy located at the primary sender, and using separate channels is equivalent to using shortest-path trees.

C.1 Experiment Results: Delay

We conducted experiments over both flat and transit-stub networks, choosing the proxies either randomly or from among the senders. For the latter strategy, the number of proxies is configured as a *maximum*, and may be less than this amount if there are fewer senders.

In all our experiments, delay generally decreases as more proxies are used. Fig. 2 shows the average delay ratio for SSM proxies using random placement on a flat network with 40 members. Results with smaller numbers of members are similar. With a single proxy, delay can be twice that of using separate SSM channels, and actually increases slightly when a second proxy is added. As more proxies are added, the delay decreases because there is a greater chance that a sender and receiver will both use proxies near to the shortest path. Note that these results are not directly comparable with a session relay, because proxies are not necessarily placed where senders are located.

A direct comparison with session relay can be made by placing proxies only at senders; in this case using a single proxy is equivalent to using an SSM session relay. Fig. 3 shows the average delay ratio for SSM proxies for this case, using a flat network and a group of 40 members. Here the delay ratio decreases more rapidly, depending on the number of senders. This experiment demonstrates that it is beneficial for a session to dynamically configure proxies as more senders arrive. The number of proxies can be controlled, if desired, using mechanisms discussed in the next section.

Our results are better for hierarchical graphs than for flat graphs, as illustrated in Fig. 4. With hierarchical graphs there is a better chance that proxies are aligned with the shortest path between a source and receiver. We suspect even more improvement will be seen by placing proxies at a domain's border router rather than at the sender.

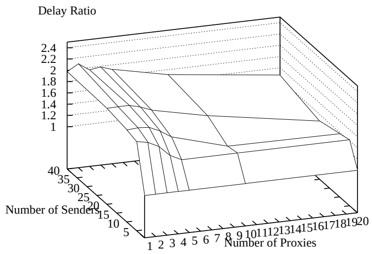

Fig. 3. Avg Delay for SSM Proxies: Sender Placement, Flat Network, 40 Members

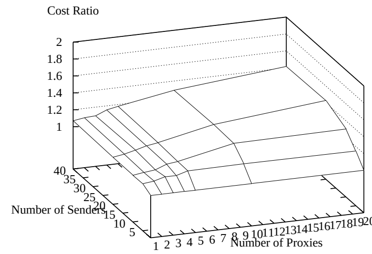

Fig. 5. Avg Cost for SSM Proxies: Sender Placement, Flat Network 40 Members

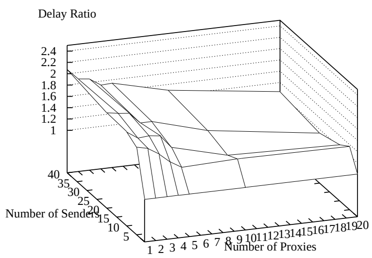

Fig. 4. Avg Delay for SSM Proxies: Sender Placement, Transit-Stub Network, 40 Members

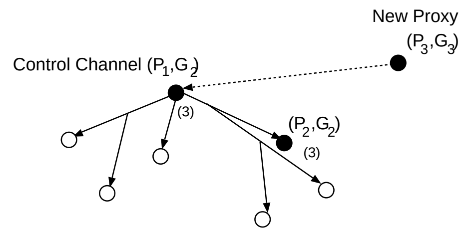

Fig. 6. Dynamic Proxy Configuration

C.2 Experiment Results: Bandwidth Cost

We used the same experiments described above to measure the bandwidth cost of SSM proxies. The bandwidth cost represents the resources used to send packets from a sender to a group; thus if three packets are sent to three different proxies, it may be possible for one link to be counted three times.

Fig. 5 shows the average cost ratio for a flat network with 40 members and a variable number of senders and proxies. As this figure shows, the average cost ratio increases when compared to the session relay approach, to a maximum of about 1.4. With a transit network the increase is a little steeper, with a maximum of about 1.8.

The bandwidth cost increases as more proxies are used because unicast distribution among proxies is rather inefficient. In section IV we show how this cost can be reduced by using multicast between the proxies. As it is, these results eliminate a major factor in bandwidth consumption, which is delivering data to dangling proxies. Our experiments show that when delivery is not optimized in this manner, the cost ratio can increase to 6 when the number of senders and proxies are both large but the number of members is small.

IV. Extensions

A. Dynamic Proxy Configuration

An application may want to avoid configuring proxies in advance and instead choose senders to act as proxies as they become active. To support dynamic proxies, the sender S_1 associated with the primary channel acts as the group manager. This sender maintains a set of authorized proxies and periodically multicasts the set on a *control channel* denoted by (S_1, G_2). The period of this advertisement can be relatively large in order to reduce overhead; delaying an update simply means that new members will stay on the primary channel longer.

The control channel is advertised via the usual SSM mechanism (i.e. a web page), and all proxies, sources, and receivers join the control channel in order to obtain the updated proxy set. Hosts that wish to act as proxies contact S_1 via unicast in order to be authorized and added or removed from the set. Fig. 6 shows an example of proxy P_3 asking to be authorized as a proxy, with other proxies and members listening on the control channel.

In the event that the primary sender fails, proxy P_2 takes over the role of the primary sender for the purpose of authorizing new proxies. Since the ordering of the proxy list is fixed, failure recovery is deterministic.

B. Access Control

The basic SSM proxy protocol implements the traditional IP model of any-source multicast, and thereby avoids setup delays. There are some applications for which this is the correct choice to make, but others will want to maintain control over which sources can transmit data to a session.

To support access control, the group creator maintains a list of authorized senders and transmits this list periodically to the proxies. Unlike the proxy list, the access control list is only needed by proxies and must be updated more frequently in order to reduce setup delay. For these reasons, we sug-

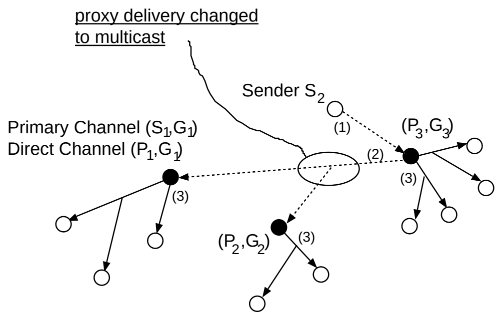

Fig. 7. SSM Proxies using Multicast Distribution

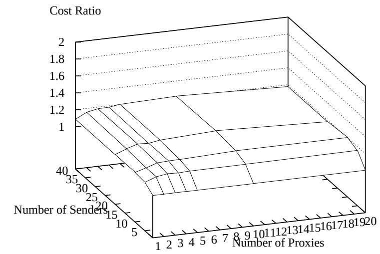

Fig. 8. Avg Cost for SSM Proxies: Multicast Delivery, Sender Placement, Flat Network, 40 Members

gest that the primary sender manage a separate *access channel* (S_1, G_3), which is joined only by proxies. The primary sender S_1 periodically multicasts the set of authorized senders on this channel, plus immediately multicasts the new set whenever it changes.

C. Multicast Proxy Distribution

To reduce the bandwidth cost of the SSM proxy protocol, we can replace the unicast proxy distribution with multicast, as illustrated in Figure 7. Each proxy maintains a separate SSM channel, called a *relay channel*, for relaying data to the other proxies. The identifier for each proxy's relay channel is distributed through either static or dynamic configuration. Each proxy then joins the relay channel for each of the other proxies, provided it is not dangling. Senders continue to unicast data to their nearest proxy, as with the basic protocol. The proxy then sends a copy of the packet on its relay channel so that it reaches the other proxies. Finally, each proxy sends the packet on its direct channel.

We implemented this extension and repeated our experiments from Section III-C to gauge its effectiveness. This change has no effect on the delay experienced by members, but does decrease the bandwidth cost of proxy distribution. Fig. 8 shows the cost ratio for a flat network with 40 members and a variable number of senders and proxies. Using multicast reduces the cost such that the ratio stays relatively flat and never exceeds 1.2. Recall that with unicast, the ratio increased up to 1.4 (see Fig. 5). Similar results are seen with transit networks, where the maximum cost is reduced from 1.8 to 1.4. The overhead introduced by using multicast consists of an extra SSM channel for each proxy.

V. Conclusions and Future Work

We have demonstrated how SSM proxies can use application-level protocols to implement traditional IP multicast on top of an SSM. This enables an SSM infrastructure to efficiently support applications with a dynamic and potentially large set of senders. We have shown how the basic proxy protocol may be augmented with dynamic proxy configuration, access control, and multicast proxy distribution.

Several areas remain open for future research. We plan to study proxy placement within hierarchical networks by using more accurate models of internetwork topologies and more precise placement policies. We are also interested in exploring how we can use SSM proxies to enhance reliability and congestion control mechanisms for multicast, similar to RMX [5]. Finally, the basic SSM proxy protocol is also a useful platform for investigating alternative content distribution architectures. Nearly any group communication protocol (e.g. Narada [6] or RMX) could be used for proxy distribution. We plan to investigate various communication algorithms to determine which are best adapted to our approach.

References

[1] Stephen Deering, Deborah Estrin, Dino Farinacci, Van Jacobson, Ching-Gung Liu, and Liming Wei, "An Architecture for Wide-Area Multicast Routing," in *ACM SIGCOMM*, August 1994.

[2] Dino Farinacci, Yakov Rekhter, David Meyer, Peter Lothberg, Hank Kilmer, and Jeremy Hall, "Multicast Source Discovery Protocol (MSDP)," Internet Draft: work in progress, July 2000.

[3] T. Bates, Y. Rekhter, R. CHandra, and D. Katz, "Multiprotocol Extensions for BGP-4," RFC 2858, June 2000.

[4] H. Holbrook and B. Cain, "Source-Specific Multicast for IP," Internet Draft: work in progress, November 2000.

[5] Yatin Chawathe, Steven McCanne, and Eric A. Brewer, "RMX: Reliable Multicast for Heterogeneous Networks," in *IEEE INFOCOM*, 2000.

[6] Yang hua Chu, Sanjay G. Rao, and Hui Zhang, "A Case For End System Multicast," in *ACM Sigmetrics*, June 2000.

[7] D.R. Cheriton and H.W. Holbrook, "EXPRESS Multicast: Making Multicast Economically Viable," in *ACM SIGCOMM*, August 1999.

[8] S. Kumar, P. Radoslavov, D. Thaler, C. Alaettinoglu, D.Estrin, and M. Handley, "The MASC/BGMP Architecture for Inter-domain Multicast Routing," in *ACM SIGCOMM*, August 1998.

[9] D. Thaler, M. Handley, and D. Estrin, "The Internet Multicast Address Allocation Architecture," Internet Draft: work in progress, December 2000.

[10] Brad Cain, Steve Deering, Bill Fenner, Isidor Kouvelas, and Ajit Thyagarajan, "Internet Group Management Protocol Version 3 (IGMP V.3)," Internet Draft: work in progress, January 2001.

[11] Cisco Systems, "Source Specific Multicast with IGMPv3, IGMPv3lite, and URD," Cisco documentation.

[12] Daniel Zappala and Aaron Fabbri, "An Evaluation of Shared Multicast Trees with Multiple Active Cores," in *International Conference on Networking, ICN'01*, July 2001.