Multicast Session Announcements on top of SSM

Pavan Namburi
Department of Computer Science
University of Texas at Dallas
Richardson, Texas - 75080
Email: pavan@student.utdallas.edu

Kamil Sarac
Department of Computer Science
University of Texas at Dallas
Richardson, Texas - 75080
Email: ksarac@utdallas.edu

***Abstract*— Before having a multicast event, information about the event needs to be announced to prospective participants. In the current multicast service model, called Any Source Multicast (ASM), it is achieved by exchanging this information on a well-known multicast group address. Researchers have developed SSM, as an alternative multicast model in response to problems with ASM. And therefore, the expectation is that SSM will soon replace ASM in the inter-domain scale.**

Session announcements is inherently a many-to-many application. Support for this application on top of SSM is not a straight forward task. But, success of large scale multicast content distribution partly depends on the existence of a successful session announcements mechanism. In this paper we propose a mechanism to support multicast session announcements application on top of SSM.

I. Introduction

Before having a multicast event, information about the event needs to be announced to prospective receivers. Such an announcement is necessary (1) to inform receivers about the existence of the event, and (2) to convey important configuration information including the multicast addresses to be used, media types, encoding format, bandwidth requirements, the duration of the event, etc.

One approach for communicating this information is to list advertisements on a web page and expect prospective receivers to visit this page. Web has been successfully used to advertise on-line content (both unicast and multicast) and has been observed to be fairly scalable. However, with the proliferation of residential broadband Internet access and continuing deployment of multicast service in the Internet, we expect that both the number of multicast content providers as well as the volume of multicast content will substantially increase in the near future. It is important to realize that the success of large-scale multicast content distribution in the Internet partly depends on the existence of a successful advertisement mechanism. From this perspective, it is necessary to consider new and alternative mechanisms for multicast content advertisement.

An alternative approach is to communicate these advertisements using multicast. Currently, multicast is supported in the Internet using the Any Source Multicast (ASM) service model [6]. ASM supports both one-to-many and many-to-many multicast applications. However, due to the complexity of the required protocol architecture, the current ASM-based multicast service in the Internet is far from being a robust service in the inter-domain scale [3]. In response to this problem, researchers have developed an alternative multicast service model, called Source Specific Multicast (SSM) [11], [5]. SSM is mainly designed for single-source multicast applications. As we present in Section II-B, it also provides a limited support to a group of multi-source applications. SSM eliminates most of the problems for wide-scale deployment and the expectations are that SSM will soon replace ASM in the inter-domain scale [2]. From the session announcements application point of view, this creates a need to support this application on top of SSM. This is important because, as mentioned above, the success of large scale multicast content distribution partly depends on the availability of a robust session advertisement mechanism. However, due to its potential large scale and dynamism of announcement sources and receivers, supporting the session announcements application on top of SSM is not straightforward.

In this paper, we propose an architecture to support session announcements application on top of SSM. Our architecture depends on using dedicated session announcements servers (SASs). Each domain uses one (or several) SAS server(s) to support the application in the domain. In addition, SAS servers in different domains form a hierarchy to provide an inter-domain level support for the application. Moreover, these servers can also provide additional services for the application such as grouping announcements based on some classification schemes, searching for particular types of announcements, etc. Finally we propose a number of modifications to improve the scalability and delay properties of session announcement application.

The rest of this paper is organized as follows. In Section II we present previous work that covers session announcements mechanism in ASM and the currently known many-to-many support mechanisms in SSM. Section III presents our hierarchical architecture for supporting this application in the SSM service model. Section IV presents our improvements on session announcement mechanism and finally the paper is concluded in Section V.

II. Previous Work

A. Session Announcements in ASM

In ASM, session announcements are communicated on a well-known multicast channel, SAP.MCAST.NET (224.2.127.254). Using a session directory tool, called *sdr*, multicast users can create, send and receive announcements on the well-known announcement channel. When a user wants to create a session announcement, he/she uses the *sdr* tool to provide necessary information for the session announcement entry. Then, the *sdr* tool creates the announcement entry using the Session Description Protocol (SDP) [9] and periodically announces it using the Session Announcement Protocol (SAP) [10]. In addition, *sdr* listens to the SAP.MCAST.NET address for announcements by other users. When an announcement is received, *sdr* caches the information and presents an updated list to the user.

Before discussing the characteristics of session announcements application, it is worthwhile to briefly describe the existing protocol architecture for ASM to gauge its complexity. Currently, the ASM service model is implemented by using three protocols in the network. These are (1) Multicast Source Discovery Protocol (MSDP) [12], used to communicate the active source information to receivers in remote domains, (2) Multi-protocol BGP (MBGP) [4], used to communicate multicast reachability among remote domains, and (3) Protocol

Independent Multicast - Sparse Mode (PIM-SM) [7], used to create multicast forwarding trees between sources and receivers. When a source starts sending out a session announcement, the edge router at the source site unicast encapsulates the announcement in a PIM-Register message and forwards this message to a dedicated router called Rendezvous Point (RP). The RP may also act as an MSDP peer in the domain. This router communicates the source activity with remote MSDP peers to inform the remote group receivers about the new source. Then the receivers in remote domains use PIM-SM protocol to create a source specific tree toward the new source. And finally additional announcement packets originating from the source propagate on this tree toward the remote receivers. The forwarding trees established in the network are sensitive to group activity. If there is no activity in the group for a period of time, the state information in the routers expire and are removed from the forwarding tables.

Sdr-based session announcements application in the ASM service model presents some unique challenges. The *sdr* traffic is bursty and its announcement period is larger then the life time of the corresponding multicast forwarding states in the routers. Therefore, when an *sdr* tool sends out announcements, these announcements create a considerable amount of control traffic in the network. First, the active source information for the announcement originator site is communicated to remote receiver sites. Then, the routers in the network create a multicast forwarding tree between the receivers and the source site. Finally, this procedure is repeated every time when the source site sends out a new burst of announcements. As a result, the dynamic nature of the *sdr*-based session announcements mechanism makes it more difficult to support or significantly reduces the effectiveness of the application. In our previous work [13], we conducted a long term monitoring of the *sdr*-based multicast session announcements application. In this monitoring, we measured the effectiveness of the application in the ASM service model. Considering the importance of session announcement for multicast content distribution, our findings in this monitoring effort supports the common belief that the current ASM-based multicast service model is not robust enough to support this type of many-to-many multicast application.

B. Many-to-Many Multicast Support in SSM

As discussed in the previous section, ASM supports both one-to-many and many-to-many applications but does not perform well in the inter-domain. SSM, on the other hand, is mainly designed for one-to-many applications. The current approaches for supporting many-to-many applications on top of SSM include (1) using multiple SSM channels, one for each sender [11], (2) using an application layer relay mechanism in the application [11], and (3) using SSM proxies mechanism [14]. These approaches have efficiency issues if we want to use them for session announcements on top of SSM. In this application, the number of session announcing sites can be potentially large and their addresses may be unknown in advance. Therefore in the case of multiple SSM channels approach, it is not easy to use a separate SSM channel for each source to support the application. In a relay-based approach, on the other hand, each source sends its announcements to a relay node which then forwards these announcements to interested receivers on a well-known SSM channel. Even though this mechanism works fine for a small group of sources, using a single relay node for the entire multicast infrastructure is not scalable. Finally, the SSM proxies work aims to provide a generic support for multiple sender applications on top of SSM. In this work, a number of relay nodes form a mesh in the network. Whenever a relay node receives data from a sender, it forwards it to all other relays for delivery. The relay nodes then forward the data to their local receivers. Our work in this paper resembles SSM proxies work in that we use a number of relay nodes to provide global support for session announcements application. However, our approach uses a hierarchical configuration of the proxies (SAS servers), provides a detailed description of the relation among the proxies in different levels of the hierarchy and includes a detailed description of forwarding rules for session announcements.

III. SAS Architecture

In this section, we present an architecture to support multicast session announcements on top of SSM. First we present a relay-based mechanism to support the application in the intra-domain. Then, we extend our architecture to support this application in the inter-domain.

A. Intra-Domain Support

For intra-domain support, we use a dedicated Session Announcement Server (SAS) within each domain. Each network operator that provides SSM service in his/her domain maintains an SAS server. This server acts as a relay node for disseminating multicast session announcements to local receivers on a well-known SSM channel. Individual end systems within the domain can use the session directory tool *sdr* to create and send their announcements directly to the SAS server via unicast. The SAS server caches and periodically forwards these announcements on a well-known SSM channel until a specified time (until the announced event starts/ends). On the other hand, all local users, who are interested in learning about future multicast events can use the *sdr* tool to join the well-known SSM channel and learn this information from the server.

One issue at this point is how to learn the address of an SAS server in a domain. A possibility is to define a naming convention (e.g. sas.foo.com for a domain foo.com) for SAS servers in each domain and then use Domain Name Service to resolve this name to the IP address. Another possibility would be to supply the SAS address information as part of the host configuration either manually or through Dynamic Host Configuration Protocol (DHCP). In addition to this the multicast address to be used has to be identified. Similar to the *sdr*-based multicast session announcements (SAP.MCAST.NET), SAS servers can use a reserved multicast address, SAS.MCAST.NET for the application. Therefore, an SAS server sas.foo.com forwards announcements on (sas.foo.com,sas.mcast.net) SSM channel and the local users join this channel to receive the announcements.

A potential problem with this approach is that the SAS server may become a single point of failure for the application. When a SAS server fails, this failure affects the session announcement application for everyone in the domain. In order to minimize the effect of such potential failures we can use multiple SAS servers in the domain. In this case, end users chooses one of these servers to send and receive session announcements. Server selection can be done at host configuration time or can use support from DNS. If an end user experiences a problem with the current SAS server, then he/she can switch to another one in the domain. In addition, each SAS server uses two SSM channels for the application: (1) sas1.mcast.net, for communicating session announcements to end hosts and (2) sas2.mcast.net, to communicate its local announcements to other SAS servers in the domain. Therefore, each SAS server joins the second SSM channel (sas2.mcast.net) of every other SAS server in the domain to receive the complete list of local announcements generated in

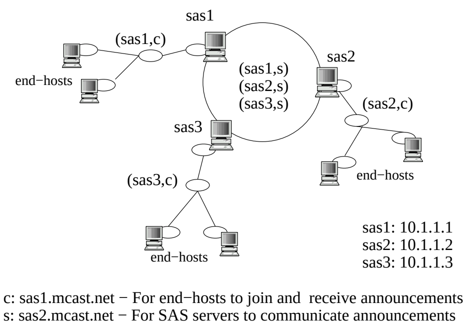

sas1
(sas1,c)
end−hosts
(sas1,s)
(sas2,s)
(sas3,s)
sas2
(sas2,c)
sas3
(sas3,c)
end−hosts
end−hosts
sas1: 10.1.1.1
sas2: 10.1.1.2
sas3: 10.1.1.3
c: sas1.mcast.net − For end−hosts to join and receive announcements
s: sas2.mcast.net − For SAS servers to communicate announcements

Fig. 1. Using multiple servers in a domain.

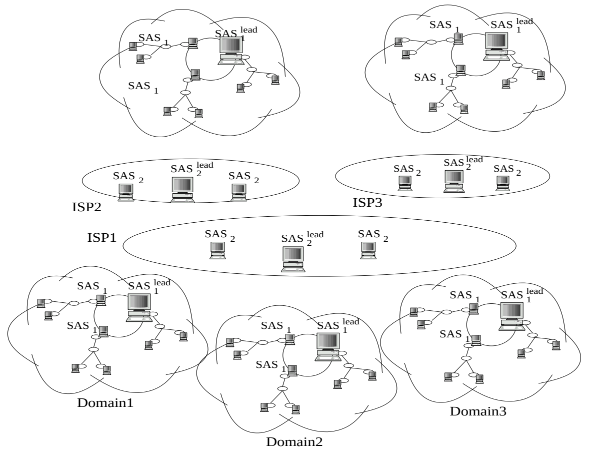


Fig. 2. A hierarchical architecture for inter-domain support.

the domain. Finally, each SAS server forwards its announcements to the receivers joined to their own sas1.mcast.net SSM channel. As a result, end hosts will be able to receive all the available local announcements independent of which local SAS server they use. Figure 1 shows an example domain having three SAS servers to support session announcements application on top of SSM.

B. Inter-Domain Support

The above mechanism supports exchanging session announcements within a domain. In order to communicate these announcements in the inter-domain scale, we need to connect SAS servers in different domains to each other. One way is to have each SAS server join the SSM group of every other SAS servers in other domains. But, as the number of multicast-enabled domains increases in the Internet, this approach becomes unscalable.

An alternative approach is to use a hierarchy of SAS servers in the inter-domain. This hierarchy can be in accordance with the underlying network provider hierarchy or it can be based on geographical proximity. In this paper we develop a two level hierarchy among SAS servers but extending it to higher levels is straightforward. In addition, for ease of reference, we use a subscript i to indicate that a server SAS_i belongs to level i in the hierarchy.

In this model, a number of topologically or geographically close-by domains form a cluster. In each cluster, we use additional SAS server(s) in the second (upper) level of the hierarchy. First, each domain having multiple SAS_1 servers chooses one as the leader, called SAS_1^{lead}. This leader then exchanges session announcements with an SAS_2, sending the local ones to it and receiving the remote ones from it. Then, each cluster chooses a leader, SAS_2^{lead}, for the second level of the hierarchy and this leader exchanges session announcements with the leaders of other clusters to support session announcements globally. Contrary to SAS_1 servers running in stub domains, SAS_2 servers running in backbone networks do not cache any announcements but directly relay them to their receivers. We expect the number of SAS_2^{lead} servers to be relatively small and well-known to each other. When a backbone provider installs an SAS_2^{lead} server, he learns the host names of the current set of SAS_2^{lead} servers operating in other backbone networks and initializes his SAS_2^{lead} server with this information. Later on, each SAS_2^{lead} server periodically sends the list of SAS_2^{lead} servers to each other to help maintain the complete list. In Figure 2 shows an example hierarchical deployment scenario for SAS servers.

In the hierarchical model, each SAS server uses one or more SSM channels to forward session announcements to others:
• *c*: used to forward session announcements to local receivers. For a SAS_1 server, the receivers of (SAS_1,*c*) SSM channel are the end systems in the domain and for an SAS_2 server, the receivers of (SAS_2,*c*) are SAS_1^{lead} servers in local domains.
• *s*: used to forward announcements to other SAS servers (siblings) in the same domain. For a SAS^{lead} server, we represent this channel as $\bar{s}$.

In addition, SAS_2^{lead} servers use a third channel *r* to forward cluster-local session announcements to other SAS_2^{lead} servers in remote domains. Moreover, end systems and SAS_1^{lead} servers use unicast to forward announcements to their parent servers. As a result, session announcements originating from any end system in any domain can be communicated to all potential receivers in any other domain in the multicast infrastructure.

In the above presentation, we build an overlay network architecture by installing SAS servers in each stub domain and connecting them to the ones in transit/backbone domains and form a global announcement infrastructure. We assume that these servers are owned and maintained by the network operators of the corresponding domains. However, such an infrastructure can be built and operated by a third party aiming to provide on-line advertisement services for commercial purpose [1]. Note that this alternative deployment scenario simplifies issues related to organization of the global SAS infrastructure. Finally, a potential alternative to our SAS-based advertisement architecture is to have individual content providers to install their own announcement servers in different locations in the Internet. However, this approach creates more state overhead in the network (less scalable) and requires more management overhead (more costly) and is therefore not desirable. A similar trade-off has been observed for web/content caching and resulted in birth of several commercial companies that built a global caching infrastructure to provide web/content caching services on top of the Internet [1].

IV. Improving on SAP

In the current ASM model, SAP-based session announcements have significant scalability problems: (1) due to the disaccord between the multicast forwarding state lifetime and SAP announcement periodicity, SAP incurs a significant overhead on the underlying multicast network infrastructure, and (2) due to the global nature of the application and the mechanisms used to limit resource (bandwidth) usage, SAP introduces delays in receiving a complete set of announcements for the end systems. In this section, we present our

Server	Channels It Sends On	Info It Sends	Channels It Joins
SAS_2^{lead}	(SAS_2^{lead}, c) $(SAS_2^{lead}, \overline{s})$ (SAS_2^{lead}, r)	$C \bigcup S \bigcup P$ $C \bigcup P$ $C \bigcup S$	(SAS_2^{lead}, r) of remote SAS_2^{lead} servers (SAS_2, s) of sibling SAS_2 servers
SAS_2	(SAS_2, c) (SAS_2, s)	$C \bigcup S$ C	$(SAS_2^{lead}, \overline{s})$ of SAS_2^{lead} server (SAS_2, s) of sibling SAS_2 servers
SAS_1^{lead}	(SAS_1^{lead}, c) $(SAS_1^{lead}, \overline{s})$	$C \bigcup S \bigcup P$ $C \bigcup P$	(SAS_2, c) of its parent SAS_2 server (SAS_1, s) of its sibling SAS_1 servers
SAS_1	(SAS_1, c) (SAS_1, s)	$C \bigcup S$ C	(SAS_2, c) of its parent SAS_2 server $(SAS_1^{lead}, \overline{s})$ of SAS_1^{lead} server (SAS_1, s) of its sibling SAS_1 servers

TABLE I
CHANNELS SAS SERVERS JOIN AND SEND ON.

approaches to help with these scalability issues.

A. Reducing the Network Overhead

Multicast forwarding states maintained in routers have a soft state life time (180 seconds by default). If an on-tree router does not receive any data packets for a particular multicast group during this time, it drops the corresponding multicast forwarding state and forgets about the group. If the data forwarding periodicity of a source is larger than 180 seconds, each time the source wants to forward a packet, it initiates a new forwarding tree establishment process in the network. This periodic tree establishment introduces significant overhead for the network and substantially reduces the robustness of the application.

SAP [10] regulates the periodicity of session announcements. This way, it attempts to control the overall bandwidth requirement of the application on a single SAP channel. The default bandwidth limit suggested by the protocol is 4000 bps. Each announcement originator is expected to listen to other announcements and determine the total number of current announcements and use this information to compute the periodicity of its own announcements. More specifically, for a given bandwidth limit, *limit,* (in bps) and announcement size, *ad_size*, the announcement interval, *interval,* is computed as

$$interval = max(300, (8 * no\_of\_ads * ad\_size)/limit). \quad (1)$$

Then an *offset* is calculated based on the *interval* as

$$offset = random(interval * 2/3) - (interval/3). \quad (2)$$

Finally, the next announcement time t_n is computed as

$$t_n = t_{current} + interval + offset. \quad (3)$$

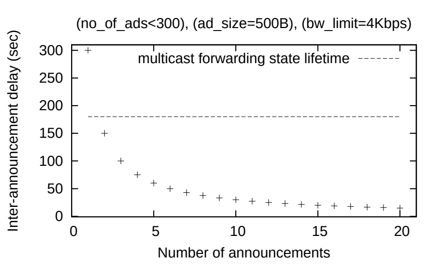
According to Equation 1, the interval between two consecutive session announcements is at least 300 seconds. However, this is larger than the lifetime of multicast forwarding states kept by routers. As a result, each time the source sends out an announcement, it initiates a new forwarding tree in the network. As we mentioned previously, this incurs more overhead to the network and affects the robustness of the application. In addition, even if the source site has several announcements, to the best of our knowledge, the *sdr* tool sends out these announcements as a burst, with a similar time interval (300 seconds minimum) between two consecutive bursts.

In our architecture, we require that SAS servers adapt a slight modification to this approach. Instead of sending announcements as a burst, SAS servers spread the announcements over an *interval* and send them as a stream of announcements with the inter-announcement time (*iat*) between two subsequent announcements being

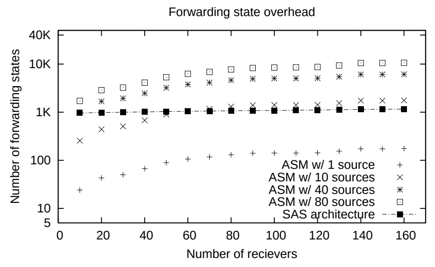
$$iat = interval/no\_of\_ads. \quad (4)$$

With this modification, as the number of announcements increases, the frequency of sending out an announcement from the SAS server will also increase. In return, this will help refresh the soft state lifetime of the forwarding states in the routers and prevent their premature expiry. Figure 3-a shows the relation between the number of announcements and inter-announcement delay for an average announcement size of 500 bytes and announcement bandwidth limit of 4000 bps. According to this figure, with a careful inter-announcement timing, the multicast forwarding states can be maintained by having as few as 2 announcements at an SAS server. Since the SAS servers will function as a proxy for a domain, we expect that these servers will easily have several dozen announcements to send periodically and therefore will greatly help in avoiding the scalability problem that SAP-based session announcement application suffers currently.

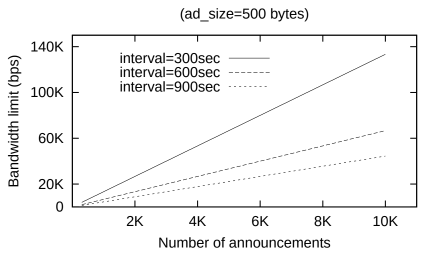
A second issue is the multicast state overhead incurred by the application. In ASM, depending of the configuration of multicast routers in the network (PIM-SM Shortest-Path-Tree Switch Over threshold configuration [7]), each session announcing source may potentially have its own shortest path multicast tree between itself and the receivers. As the number of announcement sites increases, this will cause a significant increase in the amount of forwarding state kept in the network. On the other hand, in our SAS-based infrastructure, only a small number of the servers (SAS_2 servers) use SSM channels with global scopes and incur state overhead in the backbone of the network. SAS_1 servers, on the other hand, use local (SAS_1, c) SSM channels to forward the complete set of announcements (both local and remote ones) to their local receivers. Since the scope of these SSM channels is domain-local, they do not create any scalability problems in terms of the overall forwarding state overhead in the network. Figure 3-b compares the amount of forwarding state incurred by the current approach and our SAS-based approach. In this figure, we present results of our simulations that were designed to compare the forwarding state overheads on a simple two-tier (transit-stub) network topology. We created a synthetic two-tier network topology with 615 nodes (120 stub domains each with 5 nodes on average and 5 transit domains each with 3 backbone nodes on average) using the Georgia Tech Internet Topology Modeler (GT-ITM) tool [15]. We used ns-2 network simulator [8] to simulate ASM-based and SAS-based session announcement applications with increasing number of session announcement sources and receivers; and counted the number of forwarding states created in the network. According to the figure, in ASM, as the number of session announcing sources increases, the amount of forwarding states in the



(a) Announcement periodicity.



(b) Comparison of state overhead.



(c) Bandwidth limit vs number of ads.

Fig. 3. Evaluations

network increases significantly. However, in our SAS-based approach, all the announcements are communicated over the proposed SAS-based overlay infrastructure which imposes a fairly constant and relatively small amount of forwarding state overhead in the network.

B. Reducing the Delay in Receiving Announcements

According to SAP specification, as the number of announcements exchanged in a SAP channel increases, the interval for each announcement will also increase. From the receivers point of view, this will increase the delay to receive a complete set of announcements exchanged within the channel. In order to reduce this delay, the SAP protocol specification recommends caching available announcements by using intermediate proxies. *Sdr* tools running at end systems would then contact these proxies to download the currently available announcements and use them to populate their announcement caches. Currently, such a proxy mechanism does not exist and our architecture attempts to provide this service to end systems. Before a new receiver joins the local announcement channel (SAS_1, c), it first contacts the SAS_1 server and gets the current set of announcements to populate its announcement cache. After this initialization, end systems listen to periodic announcements on the (SAS_1, c) channel to keep their announcements up to date.

As the number of announcements (both local and remote) increases, depending on the local bandwidth limitations for the application, the interval for each individual announcement also increases. In order to keep this delay bounded, we relax the bandwidth limit for local session announcement channels, i.e. (SAS_1, c) channels. Note that since the (SAS_1, c) channels are domain-local, the required bandwidth does not accumulate for the overall multicast infrastructure. Figure 3-c shows the relation between the number of announcements and the required bandwidth availability for the application. According to this figure, a bandwidth limit of 50 Kbps can support as much as 10,000 announcements with a reasonable announcement interval (900 seconds or 15 minutes).

V. Conclusions

In this paper, we have proposed a new mechanism to support multicast session announcements application in the SSM service model. These announcements are used to inform multicast users about the availability of future multicast events in the network. Our approach uses a hierarchy of relay nodes to provide both intra- and inter-domain support for communicating session announcements. We argued that due to the dynamic nature of this application, the existing approaches can only provide a limited local support for the application. Our work essentially provides a convenient mechanism to extend the scope of session announcements to a global scale in the SSM context. We have identified and examined several issues related to our hierarchical architecture. We have also proposed several modifications on the Session Announcement Protocol (SAP) to improve the overhead and delay properties of session announcement application.

Supporting many-to-many applications on top of SSM is important as the current ASM service model is not robust and will soon be replaced by SSM. Also, for the success of multicast content distribution, there is a need of an effective session announcement mechanism on top of SSM. Our work in this paper targets this problem and presents a hierarchical architecture to support session announcements on top of SSM. Even though our discussion in this paper builds around multicast, we believe that our architecture provides a robust and scalable mechanism for advertising both unicast- and multicast-based digital content on the Internet.

References

[1] *Akamai Technologies, Inc.* http://www.akamai.com/.
[2] *MBoned Mailing List.* http://antc.uoregon.edu/MBONED/.
[3] K. Almeroth, S. Bhattacharyya, and C. Diot. Challenges of itegrating ASM and SSM IP multicast protocol architectures. In *International Workshop on Digital Communications: Evolutionary Trends of the Internet (IWDC'01)*, Taormina, ITALY, September 2001.
[4] T. Bates, R. Chandra, D. Katz, and Y. Rekhter. Multiprotocol extensions for BGP-4. Internet Engineering Task Force (IETF), RFC 2283, February 1998.
[5] S. Bhattacharyya, C. Diot, L. Giuliano, Rockell R., J. Meylor, D. Meyer, G. Shepherd, and B. Haberman. An overview of source-specific multicast (SSM). Internet Engineering Task Force (IETF), draft-ietf-ssm-overview-*.txt, November 2002.
[6] S. Deering and D. Cheriton. Multicast routing in datagram internetworks and extended LANs. *ACM Transactions on Computer Systems*, pages 85–111, May 1990.
[7] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, and L. Wei. Protocol independent multicast sparse-mode (PIM-SM): Protocol specification. Internet Engineering Task Force (IETF), RFC 2362, June 1998.
[8] K. Fall and K. Varadhan. *ns Notes and Documentation*. UC Berkeley, LBL, USC/ISI, and Xerox PARC. Available at http://www.isi.edu/nsnam/ns.
[9] M. Handley and V. Jacobson. SDP: Session description protocol. Internet Engineering Task Force (IETF), RFC 2327, April 1998.
[10] M. Handley, C. Perkins, and E. Whelan. SAP: Session announcement protocol. Internet Engineering Task Force (IETF), RFC 2974, October 2000.
[11] H. Holbrook and D. Cheriton. IP multicast channels: EXPRESS support for large-scale single-source applications. In *ACM Sigcomm*, Cambridge, Massachusetts, USA, August 1999.
[12] D. Meyer and B. Fenner. Multicast source discovery protocol (MSDP). Internet Engineering Task Force (IETF), draft-ietf-mboned-msdp-*.txt, November 2002. work in progress.
[13] Kamil Sarac. Multicast monitoring: Supporting a robust multicast service in the internet. In *PhD Dissertation Thesis*, UC Santa Barbara, June 2002.
[14] D. Zappala and A. Fabbri. Using SSM proxies to provide efficient multiple-source multicast delivery. In *IEEE Globecom*, San Antonio, TX, USA, November 2001.
[15] E. Zegura, Calvert K., and Doar M. Modeling internet topology. Technical report, College of Computing, Georgia Institute of Technology, 1999.