SSM Extensions: Network Layer Support for Multiple Senders in SSM

Kamil Sarac ksarac@utdallas.edu
Pavan Namburi pavan@student.utdallas.edu
Dept of Computer Sci, Univ of Texas at Dallas
Richardson, TX 75083

Kevin C. Almeroth almeroth@cs.ucsb.edu
Dept. of Computer Sci, UC Santa Barbara
Santa Barbara, CA 93106

***Abstract*— Source Specific Multicast (SSM) provides native support for single sender multicast applications. Current proposals for supporting multiple sender applications on top of SSM includes using multiple SSM channels or using an application layer relay mechanism on top of SSM. These approaches have potential scalability or single point of failure problems, respectively. In this paper, we propose network layer extensions to SSM. Our extensions provide a convenient mechanism for SSM group owners to include additional senders in their multicast applications. We present our SSM extensions model and provide a detailed discussion on the protocol operation under various scenarios. We also examine the protocol performance by comparing it to alternative approaches on several performance metrics. With SSM Extensions, we introduce a small number of changes into the existing infrastructure and in return, provide an effective and efficient mechanism to support multiple-sender applications on top of SSM.**

I. Introduction

The original IP multicast service model, now called Any Source Multicast (ASM) [1], has a number of problems preventing its wide-scale deployment in the Internet [2]. These problems include difficulties in protecting multicast groups from unauthorized senders, address allocation, and source discovery for multicast.

As a result a more simplified multicast service model, called Source Specific Multicast (SSM) [3], has been defined. In SSM, a multicast session is associated with a specific source and this source is the only host that is allowed to send to this group. This design choice has significantly simplified the required protocol architecture for multicast and easily eliminated the above mentioned problems [3]. Even though SSM was mainly designed for single-source multicast applications, it can also be used to support a group of multiple-sender applications such as online teaching or small to moderate size video conferencing applications. The main requirements of these applications is that the number of sources be small and their identities are usually known in advance.

Current proposals for supporting multiple-sender applications on top of SSM include (1) using multiple SSM channels, one for each source and (2) using an application layer relay mechanism. As we will discuss in more detail in the next section, each of these approaches has its own limitations/problems.

In this paper, we propose network layer extensions to SSM. By integrating multiple sender support into the network layer, we reduce the amount of forwarding states to a (close to) minimum number and eliminate potential single point of failure problems. Our intent in this work is not to achieve a general many-to-many multicast support as in the case of ASM. Instead, we provide a mechanism for SSM group owners to authorize additional senders in their sessions. The group owner uses a novel authorization mechanism to create necessary forwarding state entries for the additional senders at the on-tree routers. Then, packets originating from these new senders propagate on the forwarding tree of the group owner toward the group receivers. In other words, the existing forwarding tree of the group owner acts as a bi-directional tree for the packets originating from authorized senders. Similar to the shared trees used in Core-Based Trees [4], packets originating from authorized senders propagate bi-directionally on this shared tree. We present the basic authorization procedure that we use in our extensions and discuss the protocol functioning under a large number of different scenarios. In addition, we present an effective mechanism in reducing the amount of forwarding state entries in the network. Finally, we evaluate our approach by comparing it to the currently available alternatives. With SSM extensions, we introduce a small number of changes into the forwarding state entries and the forwarding rules in the routers, and in return, we provide an efficient and effective mechanism to support multiple-sender applications on top of SSM.

The rest of the paper is organized as follows. The next section gives an overview of the SSM service model. Section III presents the basic operation of SSM extensions. Section IV discusses the tree maintenance mechanism in the Extended-SSM model. Section V is on forwarding state aggregation. Section VI includes our evaluations. Section VII is on related work and Section VIII concludes the paper.

II. An Overview of SSM

SSM is mainly derived from the EXPRESS [5] work. In SSM, an IP datagram is sent by a source S, usually called the *group owner*, to an SSM multicast address G, an IP address in the 232.*.*.* range. The receivers can receive this datagram by subscribing to SSM group (S,G). Quite often SSM groups are also called as SSM *channels*. SSM channels are defined on a per-source basis and the channel (S1,G) is different from the channel (S2,G) where $S1 \neq S2$. This model eliminated

the above mentioned problems that ASM architecture suffers currently.

The authors of [5] propose two approaches for supporting multiple-source applications on top of SSM: (1) using separate SSM channels for each sender and (2) using a higher (application) layer session relay mechanism. In the first approach, each sender has its own SSM channel and all the receivers should join each of these channels separately. If a new sender wants to become active in a session, the channel information about this sender needs to be communicated to the current receivers. This approach uses shortest path trees and reduces the network delay. However, it increases the amount of forwarding state needed to support the application. In addition, receivers need a mechanism to discover new sources in the application, a task that may be difficult.

In the case of session relays, the application uses only one channel, the session relay (SR,G) channel. All senders send their data directly to the session relay node which then encapsulates and forwards it on (SR,G) toward the receivers. This approach reduces the amount of routing state in the network but requires the existence of a relay mechanism. An important issue is where to place the relay functionality in the network. Ordinary end users may not have the necessary computing or network resources. An alternative approach is to use dedicated relay servers in the network. But these servers may easily become a single point of failure affecting potentially a large number of SSM groups. In addition, due to traffic concentration at or around such servers, losses may occur and individual applications using the relay nodes may be negatively affected. In summary, the first approach has scalability problems in terms of the amount of forwarding state needed in the network and the second approach has the potential for single point of failure problems.

III. Extending SSM for Multiple Sender Support

In this section, we present an overview of the Extended-SSM model. The main idea in this work is to use the existing forwarding tree of an SSM group (S,G) as a shared bi-directional distribution tree for other senders in an application. Our work is based on authorizing a new sender N to send to an SSM group (S,G). This approach involves (1) running an off-line authorization protocol between S and N, and (2) informing on-tree routers as well as group receivers in (S,G) about the new sender N. During these two steps, on-tree routers create new forwarding state entries for (N,G) and configure these entries using the information from the forwarding state entries of (S,G). After the successful completion of these two steps, the newly *authorized* sender N can send its packets on (N,G) and these packets will be forwarded based on the new forwarding state entries created on the *existing* forwarding tree.

Figure 1 shows an example multicast forwarding tree for an SSM group (S,G) where S is the group owner and L, M and N are the group receivers. This tree is created by using the standard SSM join mechanism and initially only S is allowed to send data on the SSM group (S,G). Now, assume that receiver N wants to send data to the members in the group. For this, N uses an off-line authorization protocol to get permission from the session owner S. During this authorization

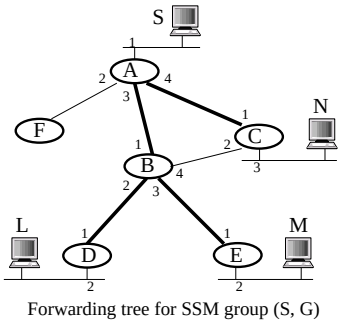

Fig. 1. A sample SSM session.

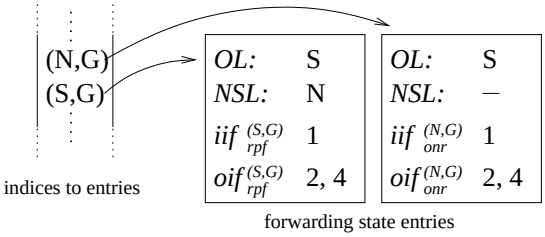

Fig. 2. Forwarding state entries at router B in Figure 1.

process, S will inform on-tree routers and group members about the new sender in the group. On-tree routers, by using the information in (S,G) forwarding state entries, will create necessary (N,G) forwarding state entries for this new sender and the receivers will be aware of the new sender in the group. Later on, when N starts sending its data to (N,G), this data will be forwarded based on the information in (N,G) forwarding state entries on the *existing* multicast tree of (S,G). In the next section, we present the tree maintenance and packet forwarding mechanisms that enable this operation.

IV. Tree Maintenance

In this section, we present the mechanism that we use to maintain forwarding trees in the Extended-SSM model. The initial tree construction procedure follows the original SSM model. Routers on the reverse shortest path between receivers and the group source in an SSM group (S,G) create forwarding state entries. More specifically, the edge router at a receiver site generates a join message and forwards this message toward the source S on the reverse shortest path. This operation is usually referred to as reverse path forwarding (RPF) [6]. Later on, packets originating from S and destined to G propagate on this forwarding tree toward the group receivers. In the Extended-SSM model, differences occur when we have a new *authorized* sender N in an SSM group (S,G). First, the multicast forwarding state entries that are maintained by on-tree routers include additional information as below:

• **Owner List (OL):** This list includes the identities of the owner(s) of the forwarding state entry. We require that in a forwarding state entry for a group (N,G), the Owner List can have at most two owners: (1) N, when the router receives an RPF Join message for (N,G) and (2) S, when S authorizes N

to send to the receivers of (S,G)[1].

• **New Sender List (NSL):** If the forwarding state entry is used by one of its owners, then this field includes the list of currently authorized senders, otherwise it is empty. Note that if this entry is used by an authorized sender, that sender may not be an owner of this entry. As an example, Figure 2 shows the forwarding state entries for (S,G) and (N,G) at router B of Figure 1 where N is an authorized sender of (S,G).

• iif_{onr} **and** oif_{onr}**:** These interfaces are used if/when the sender using the forwarding state entry is authorized by another sender. These entries are configured based on the iif_{rpf} and oif_{rpf} interfaces of the authorizing sender.

• iif_{rpf} **and** oif_{rpf}**:** These interfaces are used if/when the forwarding state entry is created/modified based on receiving an RPF Join message for the sender. In a forwarding state entry, if iif_{rpf} is non-null, this interface is always preferred over the one in iif_{onr} for accepting incoming packets from the sender using the entry.

The second difference in the Extended-SSM model is that the multicast forwarding tree corresponding to (S,G) acts as a bi-directional distribution tree for the authorized sender packets. Finally, depending on a number of session dynamics, such as receiver dynamics or authorization expiration, forwarding state entries for authorized senders may change dynamically. We discuss each of these scenarios below.

A. Sender Authorization

In this subsection, we discuss how to authorize a new sender to send to an SSM group. We assume that the new sender is a group member. If a non-member node wants to send to the group, for simplicity, we require that this sender first join the group and then initiate the authorization process. Consider a SSM group (S,G) and a group member N that wants to send data to the receivers in (S,G). For this, N gets authorization from the group owner S. This task includes two functional parts. In the first part, using an off-line authorization protocol, N gets authorization from the group owner S, and in the second part, S configures the routers on the forwarding tree of (S,G) to forward and informs the group receivers to accept packets coming from N on (N,G). During the authorization process, N sends an authorization request message, AUTH-REQ, to session owner S. On receiving the AUTH-REQ message, S first configures the on-tree routers and then sends an AUTH-OK to N granting it permission to send to the group.

In order to configure on-tree routers, S sends a NEWSENDER(N) (NS(N)) message to the group address and this message gets propagated all the way to the leaf routers on the RPF forwarding tree of (S,G)[2]. Before sending the messages, S sets IP Router Alert [7] option on these packets so that on-tree routers can intercept and act on these messages. In addition to forwarding the NS messages, each on-tree router creates a forwarding state entry for (N,G) and populates its content as shown in Figure 3. Moreover, each router modifies

[1]Having more than one authorizing node in the Owner List is possible but adds more complexity to forwarding. Therefore, we do not allow multiple authorizing senders in our protocol.

[2]The sender authorization in the Extended-SSM model is not transitive. As an example, if S is also an authorized sender for another SSM group (P,G), NS(N) messages of S will not be forwarded on RPF tree of (P,G).

R creates an entry for (N,G) and sets it as
  OL $\leftarrow$ S
  NSL $\leftarrow$ none
  $iif_{onr}^{(N,G)} \leftarrow iif_{rpf}^{(S,G)}$
  $oif_{onr}^{(N,G)} \leftarrow oif_{rpf}^{(S,G)}$

Fig. 3. Creating a forwarding state entry for an authorized sender.

/* R receives Join(N,G) on i and $OL^{(N,G)} = S$ */
R modifies (N,G) as
  $OL \leftarrow (OL \bigcup N)$
  $iif_{rpf}^{(N,G)} \leftarrow$ RPF interface toward N
  $oif_{rpf}^{(N,G)} \leftarrow i$
R sends Prune(N) on $iif_{onr}^{(N,G)}$, if $iif_{rpf}^{(N,G)} \neq iif_{onr}^{(N,G)}$

Fig. 4. State updates on receiving a Join(N,G) for an authorized sender N.

$NSL^{(S,G)}$ to include N as a new authorized sender. Finally, when NS messages arrive at edge routers on the (S,G) tree, they will inform the local group receivers about this new sender in the group. This can be done by introducing a new message type into Internet Group Management Protocol (IGMP) [8]. Note that the routers on the S-to-N forwarding path may not have the proper configuration information in their forwarding states for (N,G). That is, routers on S-to-N path should receive (N,G) packets on a different interface (i.e. $iif_{onr}^{(N,G)} \neq iif_{rpf}^{(S,G)}$). In addition, these routers should also forward (N,G) packets on $iif_{rpf}^{(S,G)}$ interface so that the packets propagate toward the root of the (S,G) tree and then other receivers on the (S,G) tree. Regarding the routers on S-to-N path, these routers may not necessarily receive the (N,G) data on the same interface as (S,G) data. For this, we use a data driven approach to correct the interface information on S-to-N routers. When an S-to-N router starts receiving (N,G) packets on an interface i, it will update its forwarding state entry and will set $iif_{onr}^{(N,G)} = i$ and $oif_{onr}^{(N,G)} = oif_{rpf}^{(S,G)} \bigcup iif_{rpf}^{(S,G)} - i$. Finally, in order to prevent source spoofing attacks by third party malicious nodes, we require edge routers to perform the standard reverse path forwarding (RPF) checks[6] in accepting and forwarding multicast packets from an end host in their local subnets.

B. New Receiver for (S,G)

When a receiver joins a SSM group (S,G), depending on the location of this receiver, a new branch is grafted onto the forwarding tree of (S,G). In the case of the Extended-SSM model, we need to (1) modify the forwarding state entries for authorized senders of (S,G) to reflect this join and (2) inform the receiver as well as the routers on the newly grafting branch about the existing authorized senders of (S,G).

When a receiver joins the group (S,G), the join request initiated by the edge router at the receiver site is forwarded toward the session source S. When an on-tree router R receives the join request on interface i, R updates the forwarding state entries of (S,G) and (N,G) for each authorized sender N of (S,G) as $oif_{rpf}^{(S,G)} \leftarrow (oif_{rpf}^{(S,G)} \bigcup i)$, and $oif_{onr}^{(N,G)} \leftarrow (oif_{onr}^{(N,G)} \bigcup i)$ respectively. In addition, the grafting router

R sends a NEWSENDER' (NS') message on the grafting interface i to inform the new receiver as well as the on-tree routers toward this receiver about the current authorized senders in (S,G). For this, it spoofs the IP address of S and uses it as the IP source for this message and addresses it to G. Note that the spoofing is necessary so that on-tree routers on this new branch can forward this message based on the forwarding state for (S,G). Similar to NS messages, NS' messages are used by on-tree routers to create necessary forwarding states for authorized senders of (S,G) and inform receivers about them. Later on, when this receiver leaves the group (S,G), router R will prune the interface i from forwarding entries of (S,G) and (N,G).

Due to their soft state nature, multicast forwarding states are maintained using periodic join messages. When an on-tree router receives a join message, it cannot always detect whether this message is to graft a new branch to the tree or it is a join refresh message coming from an existing downstream neighbor on the tree. Therefore, in order to prevent redundant forwarding of NS' messages, we require that when an on-tree router receives a NS'(N) message, it will check if it already knows about the authorized sender N. If so, it will suppress this message and will not forward it anymore. Otherwise, it will forward the message on the tree. This way, NS' messages will always be forwarded on the newly grafted branches toward new receivers but will be suppressed on already existing branches of the tree. Finally, NS messages originating from the session owner S are always forwarded toward the receivers.

C. Effect of Topology Changes

In general fault tolerance in multicast is achieved by using soft-state approach in forwarding tree creation. That is routers maintain the forwarding state entries for a short period of time and receivers send refreshing join messages to maintain the states in the routers. Therefore when a router in the multicast tree fails, this failure results in a routing topology change in the network. As a result, when a change occurs in the underlying unicast topology, due to the refreshing join messages, the multicast forwarding tree adjusts itself to reflect this topology change shortly.

In the extended SSM model, such changes are handled similar to new receiver joins: an on-tree router of an SSM group (S,G) receives a join message on an interface and sends NS' message on the interface for currently authorized sender(s) of (S,G). Downstream routers that do not have any information for the authorized sender will update their local state and also forward the NS' message on. When this message arrives a router that has information for the authorized sender, the router will stop forwarding it. Topology changes that cause a change in the multicast forwarding path between an authorized sender N and the group owner S will also be handled as discussed in Section IV-A.

D. Expiration/Revokation of an Authorization

Sender authorization in the Extended-SSM model uses a soft state approach. When an authorized sender N finishes its transmission, the session owner S stops sending refreshing NS(N) messages to (S,G) group address. After a timeout period, authorization expires at the routers and routers remove N from *NSL* list of (S,G) entry and remove (N,G) entry from their forwarding table. On the other hand, when a session owner S wants to explicitly remove authorization privileges of an authorized sender N, it sends an REVOKE-AUTH(N) message on the group address (S,G). On receiving this message, on-tree routers follow similar steps as above.

/* R receives NS(M) from N */
R creates an entry for (M,G) and sets it as
OL $\leftarrow$ N
NSL $\leftarrow$ none
$iif_{onr}^{(M,G)} \leftarrow iif_{rpf}^{(N,G)}$
$oif_{onr}^{(M,G)} \leftarrow oif_{rpf}^{(N,G)}$

Fig. 5. Creating a forwarding entry for on receiving NS(M).

/* R receives NS(N) from S */
R modifies (N,G) as
$OL \leftarrow (OL \bigcup S)$
$iif_{onr}^{(N,G)} \leftarrow iif_{rpf}^{(S,G)}$
$oif_{onr}^{(N,G)} \leftarrow oif_{rpf}^{(S,G)}$

Fig. 6. State updates on receiving a NS(N) from S.

E. Protocol Operation under Unexpected Scenarios

In our discussion so far, we presented the protocol functioning when a new sender gets authorization from a group owner and then transmits its data on the forwarding tree of the group owner. Even though we expect this to be the operational behavior most of the time, there are other cases that we need to provide support for.

First, consider a router R that is on the RPF tree of an SSM group (S,G). Assume that N is an authorized sender for (S,G) and therefore R has a forwarding state entry for (N,G) which was created due to an NS(N) message of S. Now assume that the router R receives an RPF Join request for (N,G) on an interface i. In this situation, R performs several modifications on the (N,G) state entry as shown in Figure 4. According to this figure, the router R updates the $OL^{(N,G)}$ to include N and updates the $iif_{rpf}^{(N,G)}$ and $oif_{rpf}^{(N,G)}$ entries and sends a Prune(N) message to upstream router on the (S,G) forwarding tree if $iif_{rpf}^{(N,G)} \neq iif_{onr}^{(N,G)}$. On receiving the Prune message, the upstream router stops forwarding (S,G) packets towards R. From this point on, the router R expects to receive (N,G) data on $iif_{rpf}^{(N,G)}$ and forwards them on interfaces in $(oif_{rpf}^{(N,G)} \bigcup oif_{onr}^{(N,G)})$. At the end of the join procedure, one possibility is that R may receive a NS(M) (or NS'(M)) message from N on (N,G) indicating that the group (N,G) has an authorized sender M. In this case, R creates a new forwarding state entry for (M,G) as shown in Figure 5 and includes M to $NSL^{(N,G)}$. Finally, R forwards NS(M) request to its down stream neighbors on (N,G).

The second case is on authorizing the owner of an existing SSM group. This refers to a situation where a router R that is on both (S,G) and (N,G) RPF forwarding trees receives a

NS(N) message from S on (S,G). This may correspond to a scenario where S wants its group members to receive data from another SSM group (N,G). In this situation, R modifies the forwarding state entry of (N,G) as shown in Figure 6. From this point on, R continues to accept (N,G) packets on $iif_{rpf}^{(N,G)}$ but now forwards them on interfaces in ($oif_{rpf}^{(N,G)} \bigcup oif_{onr}^{(N,G)}$).

At the end of this authorization, R is on RPF trees of both (S,G) and (N,G) groups and N is an authorized sender in (S,G). At this point, the session owner N of (N,G) may want to authorize a new sender M to send on (N,G) forwarding tree. In this situation, the router R will receive an NS(M) message and create a new forwarding state entry for (M,G) and populate it using information from (N,G) similar to the case shown in Figure 3. Note that during this updates, R uses only *RPF* interface information of (N,G) to populate *owner* interfaces in (M,G) and does not use the *owner* interface information of (N,G) in the process. This is due to the non-transitivity of sender authorization.

V. Forwarding State Aggregation

In this section, we present an effective approach to aggregate forwarding states in the routers. The goal of this aggregation is to reduce the memory requirements on the routers. At the end of this section, we discuss a second aggregation task which aims to reduce forwarding state table lookup time.

Our main observation is that when a router R on the RPF tree of an SSM group (S,G) receives a NS(N) message from S, if R is not on S-to-N direct path and it is not on the RPF tree of (N,G) (i.e. it does not have a (N,G) forwarding state yet), then it creates a new forwarding state entry for (N,G) and gets $iif_{onr}^{(N,G)}$ and $oif_{onr}^{(N,G)}$ interface information from $iif_{rpf}^{(S,G)}$ and $oif_{rpf}^{(S,G)}$ respectively. This means that R expects to receive (N,G) packets on the same interface as (S,G) packets and it expects to forward them on the same interfaces as (S,G) packets toward the group receivers. At this point, we can easily combine these two forwarding state entries into one entry and have R use this entry for both (S,G) and (N,G) packets as shown in Figure 7. When an authorized sender N of (S,G) uses the same forwarding state entry with S (when $N \notin OL^{(N,G)}$ and $N \in NSL^{(N,G)}$), R forwards (N,G) packets *only on the RPF interfaces of this forwarding state entry.* On the other hand, if R is on S-to-N path, since $iif_{rpf}^{(S,G)} \neq iif_{onr}^{(N,G)}$, R keeps these states separately. In a recent study on the characteristics of multicast trees, Chalmers and Almeroth report that the average length of a multicast path between a source and a destination node is around 15 hops[9]. Therefore, it can be easily seen that when we combine the forwarding state entries in the above way, the reduction in the total number of state entries in the network is quite significant.

So far in this section, we have seen state aggregation under the normal operation mode. However, as we discussed in Section IV-E there are a number of additional cases that we need to pay attention. Following on the example in Figure 7, we present the forwarding state maintenance procedure for each of these cases below:

• **R becomes on S-to-N path:** Due to a topology change the router R may become part of the S-to-N direct path. In this situation, router R creates a new forwarding state entry for (N,G) as presented in Section IV-A.

• **R receives NS(S) from P of (P,G):** Assume that R is also on RPF tree of some (P,G) SSM group and P wants to authorize S to forward on (P,G). In this case, the router R adds *P* to $OL^{(S,G)}$ and populates the $iif_{onr}^{(S,G)}$ and $oif_{onr}^{(S,G)}$ lists. From this point on, when R receives (S,G) data, it forwards them on interfaces in ($oif_{rpf}^{(S,G)} \bigcup oif_{onr}^{(S,G)}$) but when it receives (N,G) data, it forwards them on $oif_{rpf}^{(S,G)}$ only (remember that both (S,G) and (N,G) are pointing to the same forwarding entry).

• **R receives a Join(N,G) on an interface i:** In this case, the router R needs to create a new entry for (N,G) as shown in Figure 8. According to this figure, R includes both S and N into the owner list and copies RPF interface information from (S,G) (now the old (N,G) entry) into owner interface fields in the new forwarding state entry of (N,G).

Another issue is the possibility of merging forwarding states of (S,G) and (N,G) where N is an authorized sender of S. Such a merge is possible when $S \in OL^{(N,G)}$ but $N \notin OL^{(N,G)}$ and $iif_{rpf}^{(N,G)}$ and $oif_{rpf}^{(N,G)}$ are both empty. This can happen when the router R prunes the RPF tree state information for (N,G). Finally, a second aggregation aims at reducing the size of the lookup table (entries in the *indices to entries* list in Figure

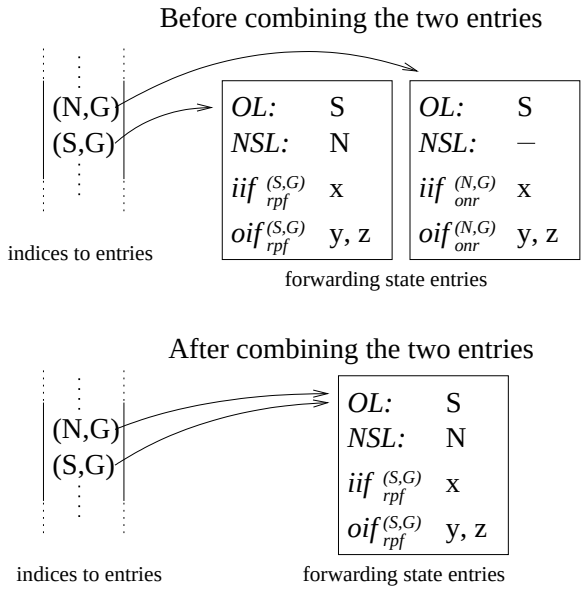

Fig. 7. Combining two state entries into one.

R receives Join(N,G) on i /*$N \notin OL^{(N,G)}$ */
R creates (N,G) as
$OL \leftarrow$ N, S
$iif_{rpf}^{(N,G)} \leftarrow$ RPF interface toward N
$oif_{rpf}^{(N,G)} \leftarrow i$
$iif_{onr}^{(N,G)} \leftarrow iif_{rpf}^{(S,G)}$
$oif_{onr}^{(N,G)} \leftarrow oif_{rpf}^{(S,G)}$
R sends Prune(N) on $iif_{onr}^{(N,G)}$, if $iif_{rpf}^{(N,G)} \neq iif_{onr}^{(N,G)}$

Fig. 8. Splitting (S,G) state on receiving Join(N,G) for authorized sender N.

7). An aggregation in this regard will essentially reduce the time for exact match plus longest prefix match lookup which needs to be done at high speed [10],[11],[12]. Currently, we are working on extending our work to enable this aggregation.

VI. Evaluations

In this section, we present preliminary evaluations on SSM extensions (SSM-Ext). For this, we compare SSM-Ext with multi-channel SSM (MC-SSM) and relay-based SSM (RB-SSM) approaches based on their efficiency and overhead in supporting multiple sender applications on top of SSM. Regarding forwarding efficiency, MC-SSM approach uses reverse shortest path forwarding trees, and therefore, provides an efficient mechanism for forwarding packets. In contrast, both RB-SSM and SSM-Ext approaches use shared trees for forwarding, and in general, shared trees are not as efficient.

As for the operational overhead, MC-SSM approach requires receivers to create a new multicast forwarding tree for each sender in the application. Each forwarding tree introduces control message overhead (includes the join messages from the receivers toward the source of the tree) and forwarding state overhead in the network. In addition, each receiver needs to keep track of the current senders in the application. RB-SSM approach, on the other hand, does not introduce state overhead for each sender in the application but requires an application layer relay mechanism. The relay node can potentially be a single point of failure for the applications using it. Even though the underlying multicast forwarding service works properly, failure of the relay node disturbs the applications significantly. On the other hand, in case of SSM-Ext, failure of the owner will disturb only those sessions for which it is the owner. Moreover on failure of the owner, part of the tree still receives messages depending on the topology of the network and position of the owner. Finally, in SSM-Ext approach, contrary to MC-SSM, each new sender incurs only a small amount of forwarding state overhead in the network. In addition, contrary to RB-SSM approach, SSM-Ext proposes a new network layer multicast routing approach eliminating the need for using an application layer support mechanism.

In order to quantitatively compare the network overhead between MC-SSM and SSM-Ext (i.e. control message and forwarding state overhead), we ran simulations using ns-2 network simulator[13]. We generated 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[14]. In the simulations, we counted the number of forwarding states needed to support a many-to-many multicast applications as the number of sources increase from 1 to 20. Figure 9 compares the state overhead of both techniques on our sample network topology. In the case of MC-SSM, since each source has its own forwarding tree, the amount of required forwarding state in the network increases significantly. On the other hand, due to forwarding state aggregation, each new sender in SSM-Ext increases the number of the required states by a small amount. In addition, we counted the amount of control message overhead for both approaches and found out that they have a very similar trend as shown in Figure 10. This figure shows the total hop counts required to successfully establish the forwarding trees for the new (authorized or additional) sources in the application.

We also compared the delay overheads of SSM-Ext with RB-SSM and MC-SSM. The simulations for these evaluations were made on the same network topology as described earlier. In particular we compared the delay of tree setup of SSM-Ext with that of MC-SSM. In the simulations, we used multicast trees with different receiver sizes, i.e. 5, 10, 15 and 20 receiver trees. In the case of SSM-Ext, the tree setup delay includes the time for a current group receiver getting authorization from the group owner and the group owner sending a NS message to the other receivers on the current forwarding tree. In the case of MC-SSM, the delay includes the time for the same group receiver announcing itself as a new sender and the owner relaying this information to other receivers on the existing forwarding tree and then the receivers joining the group of this new sender in the network. We generated three different trees for each receiver size and measured the tree setup delay for each tree. In the first part of the Table I, we present the average delay values for different receiver size trees we used in the simulation. We observe that the time required for tree setup for MC-SSM is more than that of SSM-Ext. The reason being that in MC-SSM, each of the receivers should join each of the senders, whereas in SSM-Ext there is no need for the receivers to explicitly join each sender. Instead, as the group owner sets up the forwarding states in the existing shared tree. In case of RB-SSM, the tree used is a shared tree and there is no requirement of a tree setup.

In the final analysis, we compare SSM-Ext and RB-SSM with respect to the packet propagation delay. In these simulations also as earlier, we use multicast trees with different number of receivers, i.e. trees with 5, 10, 15 and 20 receivers each. For SSM-Ext, the delay for packet propagation is the time taken from the moment the sender sends a packet to the group to the time when the last receiver receives it, along the bidirectional shared tree. This is the delay for the packet to be received by all the receivers in the multicast group. In case of RB-SSM, the packet propagates from the sender to the relay node, after which the relay node forwards the packet on the multicast tree rooted at the relay node. The packet propagation delay in this case is calculated as the time taken since the sender sends the packet, to the time when all the receivers receive the packet. The second part of Table I shows the comparison of the average packet propagation delays for the different trees. The average packet propagation delay is computed as the average of individual delays for each receiver to receive the packet. In case of SSM-Ext as the tree is a bidirectional tree we see that the average packet delay is less than that of the RB-SSM. The reason being that as the packet propagates to the owner, it is also delivered to some of the receivers on the way.

VII. Related Work

SSM extensions borrow some of its ideas from Core Based Trees (CBT) multicast routing protocol [4] where the join messages are sent toward a core router and the constructed forwarding tree works as a bi-directional distribution tree.

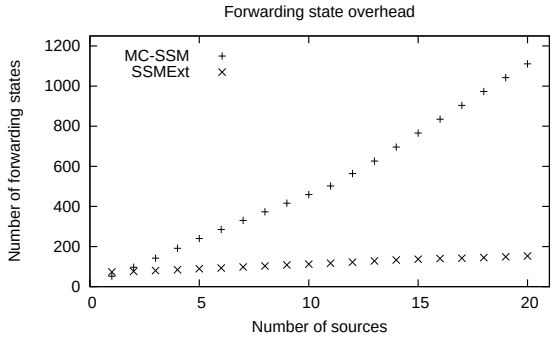

Fig. 9. Forwarding state overhead.

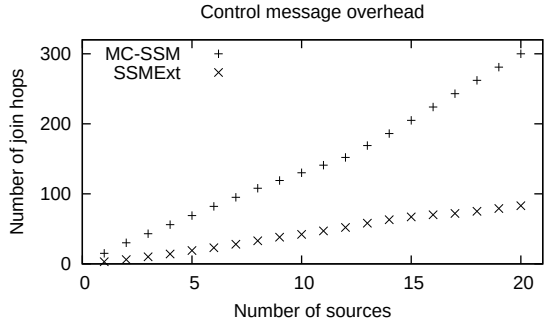

Fig. 10. Control message overhead.

Average Tree Setup Delay		
Tree Size	**SSM-Ext**	**MC-SSM**
5 receivers	4.78 sec	8.63 sec
10 receivers	5.02 sec	8.72 sec
15 receivers	5.53 sec	9.19 sec
20 receivers	5.53 sec	9.36 sec
Average Packet Delay		
Tree Size	**SSM-Ext**	**RB-SSM**
5 receivers	3.65 sec	4.53 sec
10 receivers	4.01 sec	4.55 sec
15 receivers	3.56 sec	4.05 sec
20 receivers	3.55 sec	3.95 sec

TABLE I

TREE SETUP AND PACKET PROPAGATION DELAYS.

The Extended-SSM model resembles to CBT in that the root router at an SSM group owner site can be seen as a core from packet forwarding point of view. However, one important difference is that in Extended-SSM, we create explicit forwarding state entries for each additional (authorized) senders in the group and CBT does not have an authorization notion for the senders. The forwarding state creation/maintenance process in the Extended-SSM model has been influenced by the Hop-by-Hop multicast routing protocol (HBH)[15]. In this protocol, the group source actively exchanges control messages with on-tree routers to build/modify a forwarding tree in the network. Similar to HBH, in the Extended-SSM model, a group owner sends NEWSENDER messages to the group address to create/modify the necessary forwarding state entries for the additional (authorized) senders. Finally, our work is closely related to the SSM Proxies work[16] which proposes use of multiple proxies to provide multiple-sender support on top of SSM. Similar to application layer relay mechanism, this approach is vulnerable to service disruptions due to potential proxy failures.

VIII. CONCLUSIONS

In this paper, we have presented network layer extensions to Source Specific Multicast (SSM) service model. With these extensions, we introduce a small amount of changes to the SSM forwarding state entries and SSM forwarding rules in the routers, and in return, we provide an efficient and effective mechanism to support multiple-sender applications on top of SSM. First, we have discussed the existing mechanisms for supporting multiple-sender applications on top of SSM and showed their limitations/problems. Then, we presented our basic approach and examined its behavior under a large number of different scenarios. As part of our future work, we are planning to work on security issues, the details of sender authorization protocol and extending our work to enable aggregation on the lookup table size (*indices to entries* table in Figure 7). We are also planning to develop a prototype implementation for SSM extensions and perform various experiments with it in a small test bed network environment.

REFERENCES

[1] S. Deering and D. Cheriton, "Multicast routing in datagram internetworks and extended LANs," *ACM Transactions on Computer Systems*, pp. 85–111, May 1990.

[2] C. Diot, B. Levine, B. Lyles, H. Kassem, and D. Balensiefen, "Deployment issues for the IP multicast service and architecture," *IEEE Network*, vol. 14, no. 1, pp. 10–20, January/February 2000.

[3] H. Holbrook and B. Cain, "Source-specific multicast for IP," Internet Engineering Task Force (IETF), draft-holbrook-ssm-arch-*.txt, March 2001.

[4] T. Ballardie, P. Francis, and J. Crowcroft, "Core based trees (CBT): An architecture for scalable multicast routing," in *ACM Sigcomm*, San Francisco, California, USA, September 1993, pp. 85–95.

[5] H. Holbrook and D. Cheriton, "IP multicast channels: EXPRESS support for large-scale single-source applications," in *ACM Sigcomm*, Cambridge, Massachusetts, USA, August 1999.

[6] Y. Dalal and R. Metcalfe, "Reverse path forwarding of broadcast packets," *Communications of the ACM*, vol. 21, no. 12, pp. 1040–1048, 1978.

[7] D. Katz, "IP router alert option," Internet Engineering Task Force (IETF), RFC 2113, February 1997.

[8] B. Cain, S. Deering, I. Kouvelas, B. Fenner, and A. Thyagarajan, "Internet group management protocol, version 3," Internet Engineering Task Force (IETF), RFC 3376, October 2002.

[9] R. Chalmers and K. Almeroth, "On the topology of multicast trees," *IEEE/ACM Transactions on Networking*, January 2003.

[10] P. Radoslavov, D. Estrin, and R. Govindan, "Exploiting the bandwidth-memory tradeoff in multicast state aggregation," USC - Dept. of CS, Tech. Rep., July 1999.

[11] J. Cui, D. Maggiorini, J. Kim, K. Boussetta, and M. Gerla, "A protocol to improve the state scalability of source specific multicast," in *Proceedings of IEEE GLOBECOM*, Taiwan, November 2002.

[12] D. Thaler and M. Handley, "On the aggregatability of multicast forwarding state," in *Proceedings of INFOCOM 2000*, 2000, pp. 1654–1663.

[13] 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.

[14] E. Zegura, C. K., and D. M., "Modeling internet topology," College of Computing, Georgia Institute of Technology, Tech. Rep., 1999.

[15] L. Costa, S. Fdida, and O. Duarte, "Hop by hop multicast routing protocol," in *ACM Sigcomm*, San Diego, California, USA, August 2001.

[16] D. Zappala and A. Fabbri, "Using SSM proxies to provide efficient multiple-source multicast delivery," in *IEEE Globecom*, San Antonio, TX, USA, November 2001.