

A Protocol to Improve the State Scalability of Source Specific Multicast

Jun-Hong Cui, Dario Maggiorini,
Jinkyu Kim, Khaled Boussetta, and Mario Gerla

Computer Science Department, University of California, Los Angeles, CA 90095

Abstract—Source Specific Multicast (SSM) is a viable solution for current multicast applications, since the driving applications to date are one to many, including Internet TV, distance learning, file distribution and streaming media, etc. It brings many benefits in billing, address allocation, and security. However, SSM still confronts the serious state scalability problem when there are a large number of simultaneous on-going multicast groups in the network. In this paper, we propose a protocol to improve the state scalability of Source Specific Multicast, which is called Aggregated Source Specific Multicast (ASSM). We design the detailed ASSM protocol and show that our solution can obtain significant multicast state and tree management overhead reduction while achieving transparency to end-users, compatibility with existing multicast technologies and low overhead.

I. INTRODUCTION

IP Multicast has existed for more than one decade since Stephen Deering established the IP multicast model (also called Any-Source Multicast (ASM)) in 1988 [5]. However, IP multicast is still far from being widely deployed in the Internet. There are many issues that have delayed its deployment, such as scalability, address allocation, billing, and security, etc.

Recently, some alternative service models have been proposed to solve these problems. Among them, Source Specific Multicast (SSM) [9] is dedicated to single source applications. The rationale of SSM is that almost 90% of multicast applications of immediate interest, such as file transfer and streaming multimedia, are single-source. Compared with Any-Source Multicast (ASM), SSM is a much simpler paradigm, but it alleviates many deployment problems in billing, address allocation, and security.

Like ASM, SSM utilizes a tree delivery structure, which is constructed by means of explicit-join signalling to the source. By doing so, very large multicast groups can be supported. However, a tree delivery structure requires all tree nodes to maintain per-group forwarding information, which increases linearly with the number of groups. The growing number of forwarding state entries requires more memory and entails slower forwarding processes since every packet forwarding action involves an address look-up. In other words, SSM still confronts the serious state scalability problem when there are a large number of simultaneous on-going multicast groups in the network.

The problem of state scalability has prompted recent research efforts in forwarding state reduction. Some architectures [8, 3] aim to completely eliminate multicast state at routers using network-transparent multicast, which pushes the complexity to the end-points of the network. However, these application-level multicast implementations are not very efficient since end hosts generally do not have the routing information available to routers, instead they must rely on end-to-end measurements to infer metrics to construct the multicast delivery tree. Some schemes attempt to reduce forwarding state at non-branching routers [13, 11, 4]. These non-branching-router dedicated approaches have limitations because they are designed only for net-

works with large numbers of sparse groups. Some other schemes try to achieve state reduction by forwarding state aggregation [10, 12]. Thaler and Handley analyze the aggregatability of forwarding state in [12] using an input/output filter model of multicast forwarding. Radoslavov et al. propose algorithms to aggregate forwarding state and study the bandwidth-memory tradeoff with simulations in [10]. Both these works attempt to aggregate routing state after the distribution trees have been established. Implementation of this per-router state aggregation is very complex. In addition, it is still an open question how much aggregation can be achieved or whether this approach is applicable in real systems.

To improve IP multicast state scalability, we have proposed a scheme to reduce multicast state, which we call *aggregated multicast* [6]. In this scheme, multiple multicast groups are forced to share one distribution tree, which we call an *aggregated tree*. In aggregated multicast, core routers need to keep state only per aggregated tree instead of per group. This can significantly reduce the total number of trees in the network and thus reduce forwarding state. The trade-off is that this approach may waste extra bandwidth to deliver multicast data to non-group-member nodes. Clearly, aggregated multicast applies to different multicast service models which employ a tree delivery structure, such as ASM and SSM. In our earlier work [6, 7], we introduced the basic concept of aggregated multicast, and gave an analysis of the trade-off between aggregation and bandwidth waste in general. In this paper, we propose a protocol to improve the state scalability of Source Specific Multicast using the idea of aggregated multicast, which we call *Aggregated Source Specific Multicast (ASSM)*. We design the detailed ASSM protocol. It is simple, transparent to end-users, and compatible with existing multicast technologies. Through simulations, we show that our solution can obtain significant multicast state reduction and tree management overhead decrease with low bandwidth waste.

The remainder of this paper is organized as follows. Section II gives a short review of SSM and aggregated multicast. Section III describes the architecture of ASSM. Section IV designs the detailed ASSM protocol. Section V provides a simulation study. Then in Section VI, we have a discussion about the pros and cons of ASSM. Finally Section VII summarizes the contributions of our work.

II. BACKGROUND

A. Source Specific Multicast

Source Specific Multicast (SSM) [9] is motivated by the current large number of single source multicast applications. In fact, it is an extension of Any-Source Multicast (ASM). ASM is associated with the notion of *group*. Each group is identified by a single *address* (the IP class D address, from 224.0.0.0 to 239.255.255.255). In SSM, each “group” is actually identified by

a combination of a source address S and an IP address G in the 232/8 (232.0.0.0 to 232.255.255.255) range (which is designated as SSM destination address and reserved for use by source specific applications and protocols)— (S, G) . In SSM terminology, this network service is referred as a “channel”. A host requests receiving data from a channel (S, G) by explicitly specifying both S and G , not just G as in ASM. To differentiate from ASM, “join” and “leave” are called “subscribe” and “unsubscribe” respectively in SSM. And the subscribe/unsubscribe process is almost the same as join/leave in PIM-SM or CBT except that the subscribe message is propagated toward the source, not the core node. SSM alleviates many of the deployment problems of ASM in the following aspects:

- (1) SSM defines channels on a per-source basis. This eliminates the problem of global allocation of SSM destination addresses. And each source is independently responsible for resolving address collisions.
- (2) SSM requires only source-based forwarding trees. This avoids the need for complex shared tree routing infrastructure, making it viable for immediate deployment.
- (3) SSM’s single source ownership of the channel gives a basis on which to charge and whom to charge: ISP charges source for net resources and source charges customers for service. However, it is much more difficult to identify an entity to bill for the network costs in ASM.
- (4) SSM gives a better solution to the access control problem. When a receiver subscribes to an (S, G) channel, it only receives data sent by the source S . By contrast, in ASM, any host can transmit to a group. Hence, it is more difficult to spam an SSM channel than an ASM group.

B. Aggregated Multicast

B.1 Concept

Aggregated multicast [6] is proposed to reduce multicast state, and it is targeted to intra-domain multicast provisioning. The key idea of aggregated multicast is that, instead of constructing a tree for each individual multicast session in the core network (backbone), multiple multicast sessions are forced to share a single aggregated tree.

Fig. 1 illustrates a hierarchical inter-domain network peering. Domain A is a regional or national ISP’s backbone network, and domain D, X, and Y are customer networks of domain A at a certain location (say, Los Angeles) and Domain B and C can be other customer networks (say, in Boston) or some other ISP’s networks that peer with A.

A multicast session originates at domain D and has members in domain B and C. Routers D1, A1, A2, A3, B1 and C1 form the multicast tree at the inter-domain level while A1, A2, A3, Aa and Ab form an intra-domain sub-tree within domain A (there may be other routers involved in domain B and C). Consider a second multicast session that originates at domain D and also has members in domain B and C. For this session, a sub-tree with exactly the same set of nodes will be established to carry its traffic within domain A. Now if there is a third multicast session that originates at domain X and it also has members in domain B and C, then router X1 instead of D1 will be involved, but the sub-tree within domain A still involves the same set of nodes: A1, A2, A3, Aa, and Ab.

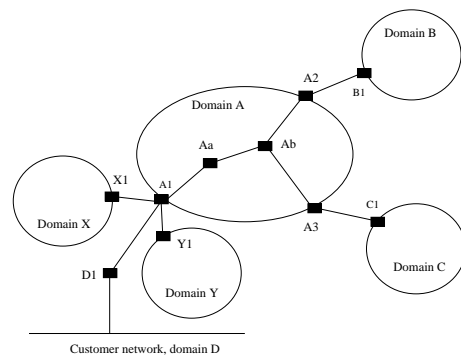


Fig. 1. Domain peering and a cross-domain multicast tree, tree nodes: D1, A1, Aa, Ab, A2, B1, A3, C1, covering group G_0 (D1, B1, C1).

In conventional IP multicast, all the nodes in the above example that are involved within domain A must maintain separate state for each of the three groups individually though their multicast trees actually have the same “shape”. Alternatively, in the aggregated multicast, we can set up a pre-defined tree (or establish one on demand) that covers nodes A1, A2 and A3 using a single multicast group address (within domain A). This tree is called an **aggregated tree** (AT) and it is shared by more than one multicast groups (three groups in the above example). Data from a specific group enters the transit domain at the traffic injecting node. It is then distributed over the aggregated tree and forwarded by traffic exiting nodes to neighboring networks. This way, core routers Aa and Ab only need to maintain a single forwarding entry for the aggregated tree regardless how many groups are sharing it. Thus, aggregated multicast can reduce the required multicast state since a core router does not need to maintain state for individual groups. And the management overhead for the distribution trees is also reduced because much fewer aggregated trees are established.

B.2 Group-Tree Matching in Aggregated Multicast

The problem of matching groups to aggregated trees hides several subtleties. The set of the group members and the tree leaves are not always identical. A match is a **perfect** or **non-leaky match** for a group if all the tree leaves are traffic injecting or exiting nodes for the group, thus traffic will not “leak” to any nodes that are not group members. For example, the aggregated tree with nodes (A1, A2, A3, Aa, Ab) in Fig. 1 is a perfect match for multicast group G_0 which has members (D1, B1, C1). A match may also be a **leaky match**. For example, if the above aggregated tree is also used for group G_1 which only involves member nodes (D1, B1), then it is a leaky match since traffic for G_1 will be delivered to node A3 (and will be discarded there since A3 does not have any state for that group). A disadvantage of leaky match is that some bandwidth is wasted to deliver data to nodes that are not members for the group (e.g., deliver multicast packets to node A3 in this example). Leaky match may be unavoidable since usually it is not possible to establish aggregated trees for all possible group combinations. Thus, if we want to achieve more multicast group aggregation, some bandwidth waste might be introduced. Therefore, there is a trade-off between aggregation and bandwidth overhead.

III. AN OVERVIEW OF ASSM ARCHITECTURE

In this section, we describe the architecture of ASSM. We intend to propose a protocol to improve the state scalability of SSM, aiming to obtain significant multicast state and tree management overhead reduction while achieving transparency to end-users, compatibility with existing multicast technologies and low overhead. At the same time we try to require no modifications on core routers.

For SSM, a typical scenario is that multicast traffic distribution is performed between a source and many subscribers (or end users) in different ISP networks, which are connected by means of wide area networks, or *transit domains* (TDs). Aggregated multicast will be implemented in a domain, especially a transit domain, to reduce multicast state. We call a domain which implements aggregated multicast a *Multicast Aggregation Domain* (MAD). To facilitate our discussion, in this paper, we demonstrate ASSM implemented in a TD, that is, a MAD is a TD in the rest of this paper. Multicast aggregation is performed inside a MAD (transparent to other domains) with multicast channels aggregated at incoming edge routers and de-aggregated at outgoing edge routers. Moreover, the multicast routing protocol in an ISP is independent of that in the MAD. An ISP can use any source specific multicast routing protocol. In the MAD, the underlying multicast routing protocol is PIM-SSM (PIM-SM supporting SSM [2]).

In a MAD, multiple multicast channels from ISPs are aggregated into a single aggregated tree. The addresses of aggregated trees in the MAD are transparent to the global Internet, and thus are independent of the multicast addresses ISPs use. To deliver packets from ISPs to the MAD, some address “translation” scheme is needed in the incoming edge routers. However, we can not change the multicast channel address in the packet header, since the outgoing edge routers will not be able to distinguish the multiple multicast channels in order to redistribute packets to ISP networks. To solve this problem, one possibility is to use IP encapsulation, which will introduce some overhead due to additional processing time and bandwidth waste. However, currently, it is the most affordable solution, because almost all widely-spread operating systems support native IP-over-IP tunnelling. Another possibility is to use MPLS, which will reduce processing time and bandwidth waste significantly. To be specific, in this paper, we illustrate ASSM using IP encapsulation.

IP encapsulation/decapsulation is done in the edge routers connecting ISPs and the MAD. These edge routers are called *Multicast Aggregation* routers (MA router for short hand). Correspondingly, we call MA routers with incoming traffic (to the MAD) as Source MA routers, and those with outgoing traffic as Destination MA routers. Source MA routers collect multicast packets coming from the ISP networks and distribute them on the right aggregated tree. And the Destination MA routers receive traffic from the MAD and forward into the ISP networks. An example environment for ASSM is shown in Fig. 2.

For the ISP networks, an MA router behaves just like a normal multicast router, subscribing/unsubscribing to a multicast channel following the ISP employed multicast routing protocol. In the MAD, MA routers act as a key role of multicast aggregation. When a multicast packet is delivered to a Source MA router, it will be encapsulated in a multicast address directed to the correspondent aggregated tree and then transmitted. As a multicast

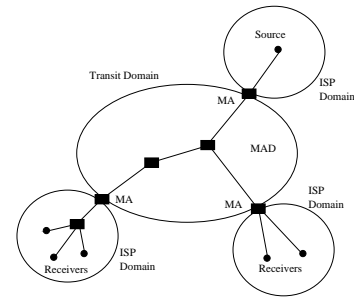


Fig. 2. An example environment for ASSM

packet reach a Destination MA router, it will be decapsulated, and distributed on the ISP networks.

IV. THE ASSM PROTOCOL

The ASSM protocol is based on transport level message communication between MA routers, which is transparent to core routers in the MAD. Core routers only need to process PIM-SSM routing messages and transmit data. MA routers are responsible for components of multicast aggregation: managing groups and trees, collecting group member change, manipulating group-tree matching, etc. It should be noted that, in this section, the terms “group” and “channel”, “subscribe” and “join”, “unsubscribe” and “leave” are interchangeable unless specified.

In order to distinguish different types of messages, we add an additional prefix to a message name. For SSM, the subscription and unsubscription messages are denoted by *M-JOIN* and *M-LEAVE* separately. For the messages defined for ASSM protocol, we add a prefix *A*, such as *A-JOIN*, *A-ACK*, *A-LEAVE*, and *A-MOVE*.

A. Source MA Routers

Since PIM-SSM is employed in the MAD, each aggregated tree has only one Source MA router and multiple Destination MA routers. In order to do aggregation, the Source MA router needs to maintain group membership, leaf nodes of aggregated trees and group-tree mapping. Based on a group-tree matching algorithm (which will be described in Section IV-F), the Source MA router can determine which tree to use for each group.

B. Source Advertisement

A multicast source is declared over the Internet by sending a global multicast directory service an SDR advertising message (containing the source and the group addresses (S, G)). Any end-user who is willing to subscribe to a channel will query the directory service in order to know the source and group addresses.

C. Local Hosts Subscribing To A Channel

Fig. 3 plots the subscription procedure of a local host R .

When a local host R located in any ISP wants to subscribe to a multicast channel, identified by (S, G) , it will send an SSM *M-JOIN*(S, G) message toward the source S .

If the channel (S, G) is already distributed inside the ISP domain, then the local protocol implementation will provide a function to intercept the request and add the new receiver to the corresponding multicast tree (like branch grafting in PIM-SSM). Otherwise, the message will reach the MA router M_r .

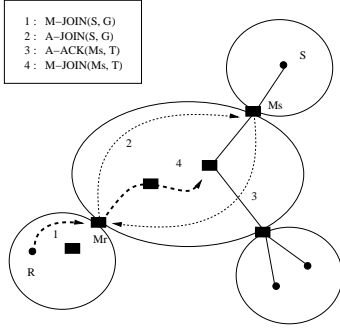


Fig. 3. The subscription procedure of R to (S, G)

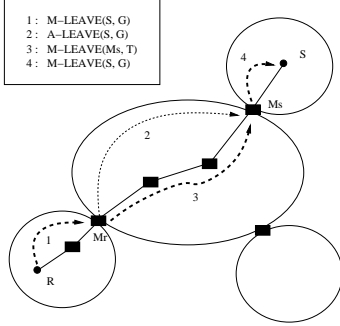


Fig. 4. The unsubscription procedure of R from (S, G)

M_r needs to know which aggregated tree is distributing channel (S, G). In order to obtain this information, it sends a transport level message $A-JOIN(S, G)$ to S. Such message will never reach S. Instead, it will be intercepted by the MA router M_s which connects the ISP of the source and the MAD.

M_s will check if it has state for channel (S, G). If yes, it will look up the aggregated tree used for (S, G), say, (M_s, T_0) (each aggregated tree is actually a normal SSM tree, which is identified by the combination of an MA router and an assigned tree address). By running a group-tree matching algorithm (which will be described in Section IV-F), M_s may obtain a better aggregated tree for (S, G), denoted by (M_s, T) . In this case, the other subscribers of channel (S, G) should be notified to switch trees, which is described in Section IV-E. Then M_s sends back an $A-ACK(M_s, T)$ message to M_r . If M_r has already subscribed to aggregated tree (M_s, T) , no further operations are needed except M_r records the mapping from (S, G) to (M_s, T) . For the other case, if M_r has not yet subscribed to (M_s, T) , it simply grafts to the tree using an $M-JOIN(M_s, T)$.

In the above, we assume that M_s has the state of (S, G). If this is not true, M_s will allocate an aggregated tree address, say (M_s, T) , for (S, G), then it sends back an $A-ACK(M_s, T)$ message to M_r . On receiving the $A-ACK$ message, M_r sets up the aggregated tree (M_s, T) by sending $M-JOIN(M_s, T)$. On the ISP side, M_s needs to subscribe to channel (S, G) through a classical SSM $M-JOIN(S, G)$ message toward S.

D. Local Hosts Unsubscribing From A Channel

Fig. 4 shows the unsubscription procedure of local host R.

When an MA router M_r receives an unsubscription message $M-LEAVE(S, G)$ from a local host R, it will check if there are

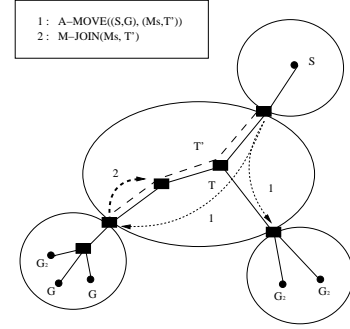


Fig. 5. The tree switching process. Channel (S, G) will be distributed by T' , since only the left side ISP has receivers for such group.

other subscribers for channel (S, G). When there are no other subscribers, M_r is required to send an $A-LEAVE(S, G)$ message to M_s and delete the state (S, G) and the mapping from (S, G) to (M_s, T) (assume (M_s, T) is used to deliver data for channel (S, G)). Moreover, M_r may be no longer interested in receiving (M_s, T) when there are no subscribers to any group inside the aggregated tree. In this case, M_r is also required to unsubscribe from (M_s, T) by sending an $M-LEAVE(M_s, T)$ message.

As in the subscription procedure, an $A-LEAVE(S, G)$ message might cause tree switching, because it is possible that the previous aggregated tree is not good for channel (S, G), since the membership is changed.

In ASSM, Source MA routers keep track of group membership. When a Source MA router detects the number of subscribers for a channel drops to 0, it will unsubscribe from (S, G) in the local ISP.

E. Tree Switching

The tree switching is an operation performed by a Source MA router in order to improve aggregation. As we mentioned above, both local host joining and leaving might trigger this operation.

To minimize additional messages, in ASSM, the tree switching is simply to ask a member to unsubscribe from its original aggregated tree and subscribe to another tree. Considering the possible data loss during the switching procedure, it is suggested to subscribe to the targeted tree first and then unsubscribe from the old tree.

In order to switch a multicast channel (S, G) from aggregated tree (M_s, T) to aggregated tree (M_s, T') , we define message $A-MOVE((S, G), (M_s, T'))$. M_s will send this message via the tree (M_s, T) to all receivers notifying of the change. It should be noted that while all other ASSM protocol messages are unicast, this one is multicast.

Fig. 5 illustrates the tree switching process. Each Destination MA router receiving $A-MOVE((S, G), (M_s, T'))$ may be required to (i) subscribe to (M_s, T') if it is not in the tree; (ii) unsubscribe from (M_s, T) if there are no subscribers to any group inside the aggregated tree (M_s, T) . M_s also needs to update its internal group-tree mapping. As a result, the channel (S, G) will be aggregated to the multicast channels transmitted by (M_s, T') . It should be noted that the original tree (M_s, T) may become empty, then its information will be deleted from M_s ; the targeted tree may not exist, in this case, the tree has to be set up and the corresponding tree information will be installed at M_s .

F. Group-Tree Matching

In ASSM, to decide a good compromise point between aggregation and overhead, groups are aggregated based on a group-tree matching algorithm, and more precisely, on an overhead function that we apply on each aggregated tree. Both *A-JOIN* and *A-LEAVE* messages will trigger the matching algorithm.

F.1 Overhead function

We denote an overhead function associated to an aggregated tree as

$$u : A \rightarrow V, \quad (1)$$

where A is the set of all possible aggregated trees and V is a value set.

Since we are trying not to deal with core routers, we need to use overhead functions that will take into account only the information in MA routers. One example overhead function is defined as

$$u(T) = 1 - \frac{\sum_{G \in g(T)} n(G)}{|g(T)| \times n(T)} \quad (2)$$

where $g(T)$ is the group set which use tree T and $n(G)$ denotes the number of Destination MA routers which want to receive data from G . $|g(T)|$ is the size of $g(T)$ and $n(T)$ is the number of MA subscribers to T . If we assume all multicast sessions have same bandwidth request, this function can be interpreted as the fraction of wasted data, the data which is delivered to unwanted MA routers in “leaky match” (defined in Section II-B.2). In other words, $u(T)$ is the relative bandwidth waste transmitted by T to MA routers.

We use function (2) as an approximation of the total bandwidth waste on T , though it will overestimate the waste, since we assume each core router has the same waste as MA router. In the real implementation, we will be able to determine good control thresholds which reflect the actual bandwidth waste.

F.2 A Matching Algorithm

Here, we present a very simple group-tree matching algorithm based on function (2).

As membership changes for a group G (because join or leave), the Source MA router M_s computes the $u(T)$ function for the corresponding aggregated tree. When it goes beyond the given threshold, the group G will be switched to another aggregated tree. The targeted tree is chosen in a greedy way, so that the sum of the overhead functions for all the aggregated trees rooted in M_s is minimized.

V. SIMULATION STUDY

In this section, we provide a performance evaluation of the ASSM protocol through simulations.

We conduct our simulations in a network abstracted from a real network topology, Abilene backbone [1], which has 12 core routers. Since there are no edge routers in the backbone, we attach an additional node as an edge router to each core router.

Given the absence of large scale real multicast traces, we are forced to develop membership models that exhibits a locality and correlated group preferences. In our simulation, we use one of

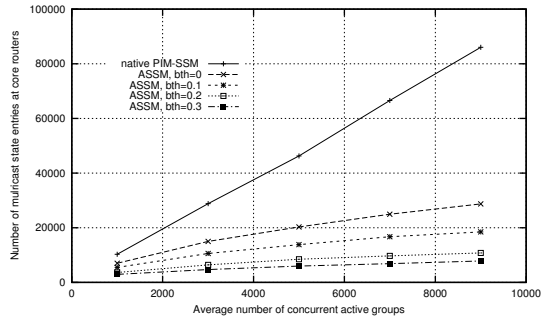


Fig. 6. Number of multicast state entries at core routers vs number of concurrent active groups at simulation steady state.

our previous group models developed in [7]: the random node-weighted model. In this model, each node is assigned a weight, which is the probability of the node to participate in multicast sessions. In the target network, core routers will not be members for any multicast group and thus are assigned weight 0. Any other edge router is assigned a weight 0.2 or 0.8 according to the real-time traffic of its corresponding core router. For a router, more traffic means more participation in the network communication, thus there is higher probability for it to join a group.

In our simulation experiments, multicast session requests arrive as a Poisson process with arrival rate λ . Sessions’ life time has an exponential distribution with average μ^{-1} . At steady state, the average number of sessions is $\bar{N} = \lambda/\mu$. Performance data is collected after steady state is reached (e.g. after $T = 10/\mu$). The group-tree matching algorithm used in our experiments is described in Section IV-F.2, but we use total bandwidth waste instead of relative bandwidth waste (mentioned in Section IV-F.1) in order to illustrate real bandwidth overhead.

We design experiments to evaluate the state reduction, join/leave latency, and tree maintenance overhead of ASSM by comparing with PIM-SSM without aggregation (native PIM-SSM). In our experiments, we vary the bandwidth overhead threshold (represented as **bth**) from 0 to 0.3 for ASSM. Fig. 6 plots the change of multicast state at core routers with the number of concurrent active groups. We can see that ASSM scales with the average number of concurrent groups: the number of multicast state entries at core routers increases with the number of groups for both native PIM-SSM and ASSM, but the increase for ASSM is much less than that of native PIM-SSM (even for perfect match ($bth = 0$), the number of state entries is only 28700 instead of 86000 when there are 9000 groups), which means much less state need to maintain at core routers. In addition, we can also see that as bandwidth overhead threshold is lifted, the number of multicast state entries reduces, which means more aggregation is achieved if more bandwidth is sacrificed. When $bth = 0.3$ and the average number of groups is 9000, the state reduction ratio is almost 90% (from 86000 for PIM-SSM to 7820 for ASSM).

We measure the number of M-JOIN and M-LEAVE messages to examine the join/leave latency. The results for M-JOIN messages are illustrated in Fig. 7. The figure shows a similar trend to Fig. 6: the more active groups and the more wasted bandwidth, the less M-JOIN messages. And the number of M-JOIN messages decreases dramatically for ASSM. In Fig. 7, when $bth = 0.3$ and the average number of groups is 9000, the reduc-

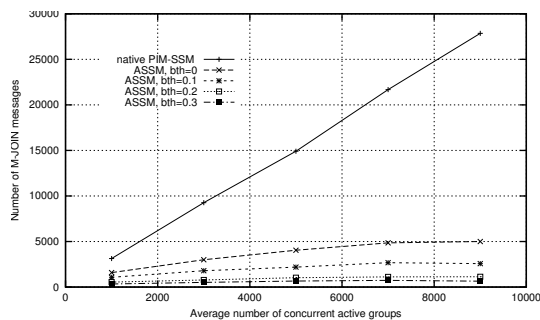


Fig. 7. Number of M-JOIN messages vs number of concurrent active groups in 50 time units of simulation steady state.

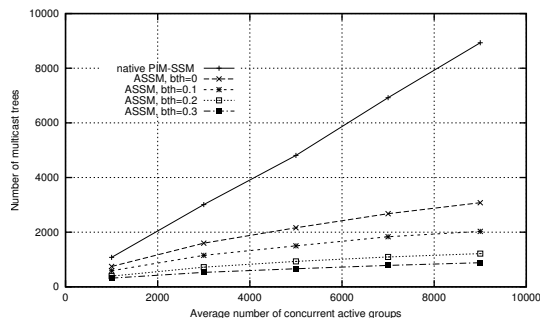


Fig. 8. Number of trees vs number of concurrent active groups at simulation steady state.

tion of M-JOIN messages is around 97% (from 27850 for native PIM-SSM to 650 for ASSM). For M-LEAVE messages, we get almost the same figure as Fig. 7, since an M-LEAVE message always corresponds to as an M-Join message in a long enough simulation period. Less M-JOIN/M-LEAVE messages mean smaller join/leave latency.

In the ASSM protocol description, we did not discuss the issue of tree maintenance, which is managed by PIM-SSM itself. In PIM-SSM, each tree is refreshed by join message sent periodically. Omitting the difference between tree shapes, which depends on the network topology and group model, we refer the number of trees as an approximate metric for tree maintenance overhead. The simulation results are shown in Fig. 8, where the curves present similar trends to those for state reduction and join/leave latency: ASSM scales very well to the number of groups, and the more bandwidth we sacrifice, the more tree maintenance overhead reduction we can achieve.

VI. DISCUSSIONS

Through the design and simulation study of the ASSM protocol, we can see that there are some trade-offs between the gains and overheads.

(1) Multicast state and tree management overhead reduction vs. bandwidth waste. Obviously, we can achieve better aggregation (in terms of more state reduction and less tree management overhead) if we can sacrifice more bandwidth. Though the real performance depends on many factors, such as network topology, group distribution, and member dynamics, a supporting point for ASSM is that bandwidth in transit domains is not a critical problem, since the deployment of optical networks in many transit domains may offer very high capacity bandwidth. In addition, from

our simulation results, we can see that significant state reduction (up to 90%) achieved while at reasonable bandwidth waste (less than 30%).

(2) A balanced view of join/leave latency. In ASSM, when a host subscribes to a channel, if the MA router which intercepts its subscription message is already on the expected tree, the join latency will be much shorter than in traditional SSM. In the other case, due to the two-step subscription process (detailed in Section IV-C), the join latency will be a little bit longer. Thus, if we can achieve enough aggregation, the average join/leave latency will be significantly reduced. We have observed this in our simulation results.

(3) Simplicity vs optimal solutions. The group-tree matching algorithm in MA routers determines the aggregation of multicast groups. To minimize bandwidth waste while maximizing state and tree management overhead reduction, we need to use a complex algorithm which considers the whole network topology and all the tree structures. However, in our ASSM design, simplicity is one of our main concerns. A suboptimal method can also help us to choose appropriate control thresholds.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, we propose ASSM to solve the state scalability problem of SSM. We design the detailed ASSM protocol and show that it is simple, transparent to end-users (and even to the core routers in transit domains), and compatible with current multicast technologies. We also analyze some trade-offs in the protocol design. Through simulation results, we can see that ASSM can achieve significant multicast state and tree management overhead reduction and join/leave latency decrease at low bandwidth waste.

We are now in the process of implementing ASSM protocol in Unix systems, and we will set up a testbed for real application scenarios, which will allow us to better evaluate the performance of ASSM.

REFERENCES

- [1] Abilene network topology. <http://www.ucaid.edu/abilene/>.
- [2] S. Bhattacharyya and et al. An overview of Source-Specific Multicast (SSM) deployment. *Internet draft: draft-ietf-ssm-overview-01.txt*, Aug. 2001.
- [3] Y. Chu, S. Rao, and H. Zhang. A case for end system multicast. *Proceedings of ACM Sigmetrics*, June 2000.
- [4] L. H. M. Costa, S. Fdida, and O. C. M. Duarte. Hop-by-hop multicast routing protocol. *Proceedings of SIGCOMM'01*, Aug. 2001.
- [5] S. Deering. Multicast routing in a datagram internetwork. *Ph.D thesis*, Dec. 1991.
- [6] A. Fei, J.-H. Cui, M. Gerla, and M. Faloutsos. Aggregated Multicast: an approach to reduce multicast state. *Proceedings of Sixth Global Internet Symposium (GI2001)*, Nov. 2001.
- [7] A. Fei, J.-H. Cui, M. Gerla, and M. Faloutsos. Aggregated Multicast with inter-group tree sharing. *Proceedings of NGC2001*, Nov. 2001.
- [8] P. Francis. Yoid: extending the internet multicast architecture. <http://www.aciri.org/yoid/docs/index.html>.
- [9] H. Holbrook and B. Cain. Source-Specific Multicast for IP. *Internet draft: draft-holbrook-ssm-arch-03.txt*, Nov. 2001.
- [10] P. I. Radoslavov, D. Estrin, and R. Govindan. Exploiting the bandwidth-memory tradeoff in multicast state aggregation. Technical report, USC Dept. of CS Technical Report 99-697 (Second Revision), July 1999.
- [11] I. Stoica, T. Ng, and H. Zhang. REUNITE: A recursive unicast approach to multicast. In *Proceedings of IEEE INFOCOM'00*, Tel Aviv, Israel, Mar. 2000.
- [12] D. Thaler and M. Handley. On the aggregatability of multicast forwarding state. *Proceedings of IEEE INFOCOM*, Mar. 2000.
- [13] J. Tian and G. Neufeld. Forwarding state reduction for sparse mode multicast communications. *Proceedings of IEEE INFOCOM*, Mar. 1998.