Source Filtering in IP Multicast Routing

De-Nian Yang, Wanjiun Liao, and Chang-Jung Kao

Abstract—IP multicasting is an efficient group communication mechanism. It avoids transmitting packets from a sender to each of the receivers separately. Under the current IP multicast model, once having joined a group, the host will receive all data destined to the group from any source, irrespective of whether it wishes to receive them. This paper studies source filtering (SF) in IP multicasting. Source filtering allows individual hosts to specify the reception of packets sent to a multicast group only from a list of source addresses or to explicitly identify a list of the sources whose data the hosts do not want to receive. We investigate the issue of source filtering in the context of multicast routing protocols, and provide support of source filtering for shared-tree based IP multicast routing. We also discuss how source filtering improves the performance of existing work on IP multicast. Finally, we provide analysis and conduct simulations to evaluate the performance of the proposed SF mechanism. The results show that our mechanism allows better bandwidth utilization and scalability than those without the capability of source filtering, thus achieving a truly efficient use of resources for IP multicasting.

Index Terms—IP multicasting, source filtering.

I. INTRODUCTION

IP MULTICASTING is an efficient group communication mechanism [1]. It avoids transmitting packets from a sender to each of the receivers separately. The standard IP multicast model includes the features of IP-style semantics, open groups, and dynamic membership [2]. Each group can have many sources, and each source does not necessarily join a group to which it is sending. A host can freely join or leave a multicast group without affecting the other members. There is no restriction on group members with respect to their physical locations and the number of groups in which they can participate. Multicast data delivery in the Internet may be implemented with two mechanisms: local group management (e.g., IGMP [3]–[5], MLDv2 [6], RGMP [7]) and global multicast routing (e.g., DVMRP [8], PIM-DM [9], MOSPF [10], EXPRESS [11], Source-Specific Multicast [12], CBT [13], PIM-SM [14]–[16], and Simple Multicast [17]).

The group management protocols and multicast routing protocols discussed above force each receiver to receive data from all sources sending to the joined groups. A receiver, however, may only be of interest to some sources [5]–[7], [11], [12], [18]–[21]. For example, a host subscribing to an IP-TV program may just need data from the IP-TV server, and treat those from other sources as service disruption [11], [12]. In distributed interactive simulations, a receiver may only need data from certain sources at any instant, while those from other sources just waste network resources and add burden to the receiver [22], [23]. Players in a network game may only need updates from those with whom they are closely interacting [21].

One simple solution to packet filtering is to require the receivers themselves to do so. This approach, however, wastes both network resources and CPU processing cycles of the receivers [19], [21]. An alternative solution is to use different multicast addresses for receivers with different preferences. This approach, however, consumes too many addresses, which are regarded as scarce resources and should be carefully allocated [24]. Moreover, this approach needs large forwarding tables at routers because the number of forwarding states at a router is proportional to the number of multicast addresses used. Thus, it does not scale well.

Source filtering (SF) is the ability for individual hosts to specify the reception of packets sent to a multicast group only from a list of source addresses or to explicitly identify a list of the sources whose data the hosts do not want to receive. This concept is first introduced in IGMP v3 for IPv4 [5], and later extended to IPv6 [6]. With source filtering, a Designated Router (DR) forwards multicast datagrams from a source to the attached local network only when at least one host in the network wishes to receive packets from the source. Consequently, it avoids network bandwidth wastage caused by delivering unnecessary packets to local networks.

To date, source filtering has been discussed mostly in the context of local group management (e.g., by IGMP v3). Without being exchanged as routing information, source filtering (SF) information provided by hosts in each local network is available only to their immediate neighboring DRs. As a result, even though all hosts in a local network has notified their DRs of their unwillingness to receive multicast packets from a particular source, the source’s packets are still forwarded all the way to the DRs, where they are discarded. To avoid network bandwidth wastage in the global network and to make truly effective use of resources in IP multicasting, SF information should be distributed as routing information to the global network.

This paper studies source filtering for IP multicast routing. We propose a mechanism to support source filtering in shared tree protocols, with the following objectives:

1) Making truly efficient use of network bandwidth. A multicast router will receive a multicast packet from a particular source only if at least one host in its directly attached network or from any downstream routers in the tree wishes to receive from the source.

2) Promptly reacting to SF updates from local networks.
3) Ensuring scalability, in terms of low overheads in communications (i.e., few control messages) and computation (i.e., small forwarding tables). The scalability is regarded as a key issue to widely deploy multicast in the future [21], [25]–[30].

4) Minimizing modification to existing multicast routing protocols to support source filtering.

We compare source-based trees with source-filtering enabled shared trees in terms of control overhead and forwarding table size. In addition, we explain how source filtering can be deployed incrementally in shared trees. We also discuss how source filtering can improve the performance of existing work on IP multicast. Note that since IGMP v3 is the dominant local group management protocol supporting source filtering, we follow its message format and its definition of source filtering in the proposed mechanism.

The rest of the paper is organized as follows. Section II presents the proposed mechanism to support source filtering for IP multicast routing protocols. Section III discusses the issue of scalability. Section IV shows the simulation results. Section V discusses the implementation issues and related work. Finally, the paper is concluded in Section VI.

II. SOURCE FILTERING FOR SHARED TREE BASED MULTICAST ROUTING PROTOCOLS

The current IP multicast domain can be divided into three hierarchical levels of autonomy, as shown in Fig. 1: the local network level, the intra-domain level, and the inter-domain level. At the local network level, it consists of individual hosts, a designated router (DR), and other multicast routers. The components use local group management protocols such as IGMP to maintain dynamic group membership and source filtering states in local networks. At the intra-domain level, the routing information is exchanged among all routers in a domain, and each router runs the same intra-domain routing protocol in the domain. A border router sits at the boundary between a domain and the Internet. At the inter-domain level, the reachability information is exchanged among border routers and backbone routers using inter-domain routing protocols such as MBGP [31], MSDP [32], and BGMP [33]. The two levels of routing autonomy provide the following two advantages: (1) supporting heterogeneity in terms of network topology and multicast protocols in each domain and (2) hierarchically aggregating routing information in the Internet to improve scalability.

Existing multicast routing protocols can be classified into two types of trees: source-based trees (e.g., DVMRP, PIM-DM, MOSPF, EXPRESS) and shared trees (e.g., PIM-SM, Bi directional PIM-SM, CBT, Simple Multicast). Source-based tree protocols allow relatively straightforward solutions to supporting source filtering [34]. For DVMRP and PIM-DM, the trees are rooted at flow sources and span all group members. When no one in the directly attached network is interested in a particular source, the DR just prunes itself from the tree. Packets from the source will not be forwarded to the DR any more. For MOSPF, the SF states of IGMP v3 should be added to Group-Membership LSAs. These modified advertisements help SF routers collect SF states in their link state databases. Upon receiving a multicast packet from a source, group (S,G) pair for the first time, the router computes the corresponding shortest path tree based on the topology information and the source filtering states stored in its local database. Again, unnecessary packets will not be received. MOSPF, however, incurs high communication overheads because a DR has to send several control messages per (S,G) pair to other routers whenever an SF state of the DR is modified.1 In addition, each MOSPF router needs a large routing table to store the SF states of all other routers in the same OSPF area. For EXPRESS and PIM-SSM [15], source-based trees are established by sending explicit Join messages. Based on the received SF states of IGMP v3, the DR can determine which trees to join.

It is very challenging for shared tree protocols to provide source filtering because each such tree is shared by all sources of the group. The problem is even more complicated when one considers the following issues:

1) A source in a shared tree may or may not be on-tree (i.e., the source may or may not be a group member).
2) A shared tree may be unidirectional or bi-directional, i.e., packets may be delivered uni-directionally or bi-directionally.
3) A shared tree has better scalability than a source-based tree.

The challenge for shared tree protocols is enabling source filtering while achieving scalability.

The main contribution of this paper is to design a simple, low-overhead source filtering mechanism applicable to all shared tree multicast routing protocols. In the following, we will focus on the detail of our mechanism. In Sections III and IV, we will prove that our mechanism is more scalable than the source-based tree approach.

The proposed mechanism consists of two phases of operations: 1) a packet forwarding phase and 2) a message exchange phase. In the packet forwarding phase, each router determines how to forward packets based on the SF states stored for each of its interfaces. In the message exchange phase, routers exchange messages to ensure the correctness of the SF tables. Together, they filter out unnecessary multicast packets in shared trees to minimize bandwidth wastage.

1For DVMRP and PIM-DM, the DR has to send a prune or graft message to each leaving or joining group whenever the SF state of the DR is changed. For MOSPF, the DR also needs to send Group-Membership LSAs to other routers in the same OSPF area. The overhead of DVMRP/PIM-DM may be large if the SF updates are correlated to many sources, especially when the update is due to a change in the filter mode. MOSPF overhead is enormous because Group-Membership LSAs need to be flooded to the entire area.
A. Packet Forwarding Phase

In a unidirectional shared tree such as PIM-SM, a source tunnels packets to the Rendezvous Point (RP), from where the packets are distributed downstream along the tree to all receivers. In other words, packets are forwarded only in the downstream direction. For bi-directional shared trees such as CBT, packets from on-tree sources may be forwarded upstream, downstream, or in both directions, depending on where the source is located (and what the tree looks like). Consequently,

1) For unidirectional shared trees, a source packet must arrive at a router from the interface upstream to its parent (called the upstream interface in the rest of the paper), and the packet must be sent to the interfaces downstream to its children (called the downstream interface). Thus, the incoming interface for each packet in a group must be the upstream interface of a router, while the outgoing interfaces for each packet in a group must be the downstream interfaces of the router, plus the interfaces to the directly attached networks with group members.

2) For bi-directional shared tree, the incoming interface for each packet in a group may either be the downstream or upstream interface. Similarly, the outgoing interfaces for each packet in a group may be upstream, downstream, or in both directions, plus the interfaces to the directly attached networks with group members.

In summary, each router needs to associate each outgoing interface with an SF state for a group to determine how to route packets. Each packet is forwarded only to those “outgoing” interfaces through which the packet is needed. Thus, the first step for SF routers is to identify to which interfaces the packet should be forwarded.

One simple solution is to associate each outgoing interface of each router with multiple SF states, where each SF state corresponds to the state of each downstream LAN (i.e., a LAN attached to any downstream router). Each router then has the complete SF state for each downstream LAN. Based on the SF states, routers can unambiguously identify to which outgoing interfaces packets of a source should be forwarded. While simple, this solution does not scale well, because the memory size required by each SF router is proportional to the number of downstream LANs. Thus, it is not applicable to large shared trees.

In fact, it is not necessary for each router to maintain an SF state per downstream LAN. Instead, it can associate each of its outgoing interfaces with one single SF state which summarizes the requirements of all the downstream LANs. This can significantly reduce the forwarding table size of each router and improve routing efficiency.

1) Packet Forwarding Mechanism: Each SF router associates each of its outgoing interfaces with one SF state, and each state corresponds to either the aggregation of all downstream routers’ requirements, or the state obtained from its directly attached network. Each router maintains a forwarding table, each entry of which stores the SF state for one of its outgoing interfaces. Each SF state includes a group address, a filter mode, and a source list. Upon receiving packets of an (S,G) pair, the router searches its forwarding table and identifies the active outgoing interfaces for the pair. An outgoing interface is called active for an (S,G) pair if at least one of the group members interested in source S’s packets can be reached through the interface. In other words, an outgoing interface is active for an (S,G) pair if either condition holds:

1) The filter mode is an “include,” and the source S is in the associated source list.

2) The filter mode is an “exclude,” and the source S is not in the associated source list.

Fig. 2 gives an example forwarding table at an SF router, say \( R_1 \). Suppose that \( R_3 \) has three downstream interfaces, namely, \( D_1, D_2, \) and \( D_3, \) and two interfaces to the directly attached local networks, namely, \( L_1 \) and \( L_2 \). Each of its outgoing interfaces is associated with an SF state consisting of a filter mode and a source list. Upon receipt of a packet from source \( a \), \( R_3 \) determines the active outgoing interfaces for source \( a \). In Fig. 2, only \( D_1 \) and \( L_1, L_2 \) are active for source \( a \), and the packet will be forwarded only to these two interfaces. Without the capability of source filtering at the router, the packet would also be forwarded to \( D_2, D_3 \), and \( L_2 \), in which directions no hosts are interested in source \( a \).

One might doubt that SF may affect the forwarding speed of multicast packets. In fact, the extra time incurred by SF is negligible compared to the original forwarding time, argued as follows. Current high-speed IP routers support a processor per interface [35], which performs address lookups in a distributed manner. In addition, the number of sources in a source list should be far less than the number of destination addresses in the forwarding table, implying that address lookups of source lists are very fast. Moreover, recent work show that one address lookup in a forwarding table can be done in an order of \( 10^5 \) [36]–[39].

B. Message Exchange Phase

In this phase, routers exchange SF messages to avoid transmitting unnecessary packets across the network. To prevent a source from sending packets to a local network in which no one wishes to receive them, the SF information should be propagated toward the source. This information is forwarded hop by hop from the DR directly attached to the network, and through the interface to which packets from the source would be received. This process repeats until the source is reached, or the SF state can be merged at an on-tree router. For a unidirectional shared tree, a source’s packet must be received through the upstream interface of the tree. However, a source’s packet in a bi-directional shared tree can be received in either direction. Thus, each router must first determine the incoming interface for a source on the tree before the SF messages are exchanged among routers. In what follows, we will describe how to exchange messages for unidirectional shared trees, and extend the mechanism to bi-directional shared trees.

1) Unidirectional Shared Tree: In a unidirectional shared tree such as PIM-SM, SF routers exchange messages to learn
the SF states of group members. Flooding is one solution for this exchange, but incurs high communication overheads. To improve scalability and to minimize the overheads, we propose a new approach to exchanging messages. An SF router first aggregates the SF states of all outgoing interfaces belonging to the same group, and then forwards the aggregate state upstream to its parent.

a) SF state aggregation: Suppose that there are \( m \) “include” filter modes with \( m \) source lists, denoted by \( I_1, I_2, \ldots, I_m \), and \( n \) “exclude” filter modes with \( n \) source lists, denoted by \( E_1, E_2, \ldots, E_n \), at an SF router, considering all the outgoing interfaces for an group. Let \( M_I \) and \( S_I \) denote the resulting filter mode and source list after aggregation, respectively. They are determined as follows:

1. If \( n = 0 \), set \( M_I = \text{include} \), and \( S_I = I \); otherwise, \( M_I = \text{exclude} \), and \( S_I = E - I \).

IGMP v3 allows hosts to freely change their SF states. This leads to the SF states of each DR may be changed at any time, and thus its upstream routers need to reflect this change accordingly. One way to reflect this change is that as soon as an update in SF states is received through any interface, the router aggregates the SF states of all the outgoing interfaces belonging to the same group, and then forwards the aggregated state upstream to its parent. The operation repeats until it reaches a router in which the aggregated state is a subset of the SF state of the interface from which the aggregate message is received. This approach, however, has three drawbacks:

1. The computational complexity of aggregation performed by an SF router is proportional to the number of outgoing interfaces belonging to a group times the length of the source list for each outgoing interface. The computational overhead will be intolerant when the number of interfaces or source lists becomes very large.

2. Upon receipt of an SF update, the router aggregates the SF states of all the outgoing interfaces belonging to the same group and then forwards the aggregated state upstream, irrespective of whether the content remains intact.

3. If an aggregate remains unchanged (or only slightly different) before and after the receipt of an SF update (for example, the filter mode stays unchanged but the source list adds a few more new sources), it is inefficient to aggregate the SF states of all the outgoing interfaces for the group and then forward the entire result upstream.

In short, this approach is inefficient due to high computational overhead and high communication overhead. Again, it suffers from the scalability problem. To solve problem (2), we add a new entry to the forwarding table, called the aggregate entry, which stores the resulting SF state aggregated from all the outgoing interfaces. An aggregated state will not be forwarded upstream unless the aggregate entry is changed after the aggregation.

To solve problem (3), we exchange SF updates with three IGMP messages:\(^2\) “Update,” “Allow,” and “Block.” As defined in IGMP v3, an Update message carries the complete SF state; both “Allow” and “Block” messages have no filter mode, but the source list only. Upon receiving an “Allow” message with a source list of \( S_a \) from an interface, the router interprets that some one wishes to receive the additional sources indicated by \( S_a \) through the interface. In contrast, upon receipt of a “Block” message with a source list of \( S_b \), the router considers no one is interested in source \( S_b \).

b) SF state exchanges: Some notations are defined as follows.

1. Before a router receives an SF update: let \( F_{I_{m}} \) and \( S_{I_{m}} \) denote the original filter mode and source list, respectively, for the interface through which the update will be received, and \( F_{E_{n}} \) and \( S_{E_{n}} \), the original filter mode and source list, respectively, in the aggregate entry.

2. After the router has received an update: let \( F_{I_{m}} \) and \( S_{I_{m}} \) denote the new filter mode and source list, respectively, for the interface through which the update have been received, and \( F_{E_{n}} \) and \( S_{E_{n}} \), the new filter mode and source list, respectively, in the aggregate entry.

The SF router works as follows.

1. Upon receipt of an “Allow” message with a source list of \( S_a \):
   a) If \( F_{I_{m}} \) is an include, it sets \( S_{I_{m}} = S_a \cup S_{I_{m}} \).
   b) If \( F_{E_{n}} \) is an exclude, it sets \( S_{E_{n}} = S_{E_{n}} - S_a \).
   c) If \( F_{E_{n}} \) is an include, it sets \( S_{E_{n}} = S_{E_{n}} \cup S_{E_{n}} \). Besides, if \( S_a - S_{E_{n}} \neq \emptyset \), the router forwards an “Allow” message with a source list of \( S_a - S_{E_{n}} \) toward the upstream router.
   d) If \( F_{E_{n}} \) is an exclude, it sets \( S_{E_{n}} = S_{E_{n}} - S_{a} \). Besides, if \( S_{E_{n}} \cap S_a \neq \emptyset \), the router forwards an “Allow” message with a source list of \( S_{E_{n}} \cap S_a \) toward the upstream router.

2. Upon receipt of a “Block” message with a source list of \( S_b \):
   a) If \( F_{I_{m}} \) is an include, it sets \( S_{I_{m}} = S_{I_{m}} - S_{b} \).
   b) If \( F_{E_{n}} \) is an exclude, it sets \( S_{E_{n}} = S_{E_{n}} \cup S_{b} \).
   c) If \( F_{E_{n}} \) is an include, it checks if, through any other interfaces, packets from source list \( S_{E_{n}} \cap S_b \) are of interest. If no one wishes to receive packets from a subset of \( S_{E_{n}} \cap S_b \) (denoted by \( S_{P} \)), the router sets \( S_{E_{n}} = S_{E_{n}} - S_{P} \), and forwards a “Block” message with a source list of \( S_{P} \) toward the upstream router.
   d) If \( F_{E_{n}} \) is an exclude, it checks if, through any other interfaces, packets from source list \( S_{E_{n}} \cap S_b \) are of interest. If no one wishes to receive packets from a subset of \( S_{E_{n}} \cap S_{E_{n}} \) (denoted by \( S_{P} \)), the router sets \( S_{E_{n}} = S_{E_{n}} \cup S_{P} \), and forwards a “Block” message with a source list of \( S_{P} \) toward the upstream router.

3. Upon receipt of a change in the SF state for an interface to a directly attached LAN, or upon receipt of an “Update” message with a filter mode of \( F_{I_{m}} \) and a source list of \( S_{I_{m}} \) from an immediate downstream router:
   a) If both \( F_{I_{m}} \) and \( F_{E_{n}} \) are includes, the router regards it as having received an “Allow” message with a source list of \( S_{I_{m}} \cap S_{I_{m}} \) and a “Block” message with a source list of \( S_{E_{n}} - S_{E_{n}} \).
   b) If both \( F_{I_{m}} \) and \( F_{E_{n}} \) are excludes, the router regards it as having received an “Allow” message with a source list of \( S_{I_{m}} - S_{E_{n}} \)
list of $S_{16} = S_{15}$ and a “Block” message with a source list of $S_{15} = S_{14}$.

(c) If $F_{16}$ and $F_{15}$ are exclude and include, respectively, the router sets $F_{16}^{\text{in}} = \text{exclude}$ and $S_{15}^{\text{in}} = S_{15}$. Besides, i) If $F_{16}$ is an include, the router checks if, through any other interfaces, packets from source list $S_{14} \cap S_{16}$ are of interest. If no one wishes to receive packets from a subset of $S_{14} \cap S_{16}$ (denoted by $S_{15}$), the router sets $F_{16}^{\text{an}} = \text{exclude}$ and $S_{15}^{\text{an}} = (S_{14} \setminus S_{16}) \cup S_{15}$, and then forwards upstream an “Update” message with a filter mode and a source list of $F_{16}^{\text{an}}$ and $S_{15}^{\text{an}}$, respectively. 

ii) If $F_{16}$ is an exclude, the router checks if, through any other interfaces, packets from source list $S_{14} \cap S_{16}$ are of interest. If no one wishes to receive packets from a subset of $S_{14} \cap S_{16}$ (denoted by $S_{15}$), the router sets $S_{15}^{\text{an}} = (S_{14} \setminus S_{16}) \cup S_{15}$, and forwards upstream an “Update” message with a filter mode and a source list of $F_{16}^{\text{an}}$ and $S_{15}^{\text{an}}$, respectively.

(d) If $F_{16}$ and $F_{15}$ are include and exclude, respectively, the router sets $F_{16}^{\text{in}} = \text{include}$, $S_{15}^{\text{in}} = S_{15}$, and aggregates the SF states of all the interfaces at the router to update the aggregate entry. The update message is forwarded to the upstream router, with a filter mode and a source list of $F_{16}^{\text{an}}$ and $S_{15}^{\text{an}}$, respectively.

On receiving a join or leave message, a router aggregates the SF states of all the outgoing interfaces for the group and determines $F_{\text{an}}$ and $S_{\text{an}}$. Join/leave messages can be regarded as Update messages to simplify the aggregating process. In other words, $F_{16}$ is set to an “include” and $S_{15}$ set to empty when the join message is received. Likewise, $F_{16}$ is set to an “include” and $S_{15}$ empty for the leave message.

Fig. 3 illustrates the mechanism described above. Assume that there are sources $a$ to $f$ in the system.

1) Upon receipt of an “Allow” message with source list $\{e\}$ through downstream interface $D_2$, router $R_1$ adds source $e$ to the source list of entry $D_2$, and checks the SF state in the aggregate entry. Since the filter mode of the aggregate entry is an exclude and the source list is $\{d\}$, leading to $S_{10} = S_{15} = \emptyset$, $R_1$ is not required to forward the new state upstream to its parent.

2) Upon receipt of a “Block” message with source list $\{f\}$ through downstream interface $D_2$, router $R_1$ checks if $D_1$, $D_3$, $L_1$, or $L_2$ wishes to receive packets from source $f$. Since the filter modes of both entries $D_3$ and $L_1$ are excludes, and their source lists do not include source $f$, both

---

<table>
<thead>
<tr>
<th>Group ID</th>
<th>Outgoing interface</th>
<th>Filter mode</th>
<th>Source list</th>
</tr>
</thead>
<tbody>
<tr>
<td>(*,G)</td>
<td>$D_1$</td>
<td>Include</td>
<td>a, b, c, e</td>
</tr>
<tr>
<td></td>
<td>$D_2$</td>
<td>Include</td>
<td>b, c, f</td>
</tr>
<tr>
<td></td>
<td>$D_3$</td>
<td>Exclude</td>
<td>a, d</td>
</tr>
<tr>
<td></td>
<td>$L_1$</td>
<td>Exclude</td>
<td>b, d</td>
</tr>
<tr>
<td></td>
<td>$L_2$</td>
<td>Include</td>
<td>c, e</td>
</tr>
<tr>
<td>Aggregate</td>
<td></td>
<td>Exclude</td>
<td>d</td>
</tr>
</tbody>
</table>

---

Fig. 3. An example of message exchanges.

D_3 and L_1 are active outgoing interfaces for source f. Consequently, the aggregate entry remains intact. Again, $R_1$ is not required to forward the new state to its parent.

Suppose that a host in a local network attached to $L_2$ changes its filter mode to an exclude and the source list to $\{a, e\}$ upon receipt of an IGMP General Query. Since the filter mode in the aggregate entry is an exclude, $R_1$ checks if source $e$ is to be received through any of its outgoing interfaces. $R_2$ finds that the source is of interest through $D_1$, $D_3$, and $L_1$, and source $d$ is of interest to $L_2$. Thus, the new source list in the aggregate entry is empty.

2) Bi-Directional Shared Tree:

a) Intra-domain protocols: For bi-directional shared trees, the incoming interface for packets of a source may be the upstream or downstream interface. Similarly, the outgoing interfaces may be upstream, downstream, or both, depending on where the source is located. Thus, the mechanism described for unidirectional trees in Section II-B-1 may not be directly applicable to bi-directional trees, for the sake of better efficiency of bandwidth utilization. Fig. 4 demonstrates what the problem is if it is applied directly. It shows a bi-directional shared tree with four routers: $R_1$, $R_2$, $R_3$, and $R_4$, where $R_1$ and $R_3$ are the DRs of $\text{LAN}_1$ and $\text{LAN}_2$, respectively. The entry for $\text{LAN}_1$ in $R_1$’s forwarding table has a state of exclude $\{S_2\}$; the entry for $\text{LAN}_2$ in $R_3$’s forwarding table has a state of exclude $\{S_1\}$. Using the unidirectional shared tree mechanism, all SF states are forwarded upstream only. Suppose that $R_4$ is not interested in source $S_1$. Since this SF information is eventually transmitted to $R_1$, $R_1$ will not forward $S_1$’s packets through $I_{12}$. SF states, however, will not be sent downstream. As a result, while $\text{LAN}_1$ has an SF state of “exclude $\{S_2\}$,” such information does not exist in the entries for interfaces $I_{21}$ in $R_2$ and $I_{32}$ in $R_3$. $S_2$’s packets will still be forwarded to $\text{LAN}_1$, even though no hosts in $\text{LAN}_1$ need them.

To avoid the problem addressed above, our mechanism is extended to deliver SF states bi-directionally (i.e., both upstream and downstream). One implementation for the extension is to
forward the aggregate entry of each interface to its adjacent router. While simple, this approach may contain unnecessary sources in the source list in the aggregate entry of an interface. For example, in Fig. 4, if the SF states of I_{21} and I_{23} are include \{S_2\} and include \{S_1\}, respectively, the aggregate entry of I_{24} will be include \{S_1, S_2\}. Since I_{24} is not the incoming interface of S_1 and S_2, the aggregated state should not be forwarded toward this direction (i.e., the right branch of router R_2).

In short, this implementation will transmit too many redundant SF states on the links which do not lead to the source. To solve this problem, each router should identify the incoming interface for a source, and send the SF states only toward the direction to the source.

In [18], the authors proposed a method to assign an ID to each router on a multicast tree. Each child router’s ID consists of its parent ID appended by a number. For example, the ID of the root is 1, and the ID of its first child router is 11. This approach allows each router to determine the incoming interface for a source if the mapping between the source’s IP address and source’s ID is available. However, it is very costly to maintain such ID structure and the mapping. Many routers on such a tree may be involved in modifying their IDs by message exchanges to reflect dynamic membership even though the tree topology is just changed slightly. In this paper, we adopt another approach which associates each SF router with the domain topology information (such as the link state database in OSPF), and then determines the shortest path tree which is rooted at the core and covers all routers in the domain. From the shortest path tree, each router can easily identify the incoming interface for any on-tree source, and forward the SF state only through the identified incoming interface. In short, the number of extra states disseminated is reduced at the expense of a larger forwarding table (to store topology information) at each router. This approach performs well for small-scale groups. As the group size increases, more users will join the group, which may result in more redundant SF updates being merged at successive hops. Consequently, the advantage of considering topology information will wane as group size becomes large. We will come back to this issue again in Section IV. Note that even without the topology information, our mechanism still works, except that the SF messages forwarded may contain some unnecessary sources in the source list. Section II-B-2-b suggests another solution to identifying incoming interfaces for sources without the need of topology information.

So far, we have just focused on the processing of on-tree sources. Off-tree sources are treated differently in different bi-directional shared tree protocols. For Bi-directional PIM-SM and Simple Multicast, packets of off-tree sources are unicast to the core of the tree. Once the packets have reached any on-tree router, the router distributes them to the tree both in the upstream and downstream directions, exactly in the same way as for on-tree sources. In other words, packets of any source will arrive only through the interface in the shortest path to the source. For CBT, on the other hand, packets of an off-tree source may come through two possible incoming interfaces, depending on whether the source is on-tree. Packets from an off-tree source are tunneled to the core of the tree, from where the packets are forwarded downstream to all the receivers. Thus, packets of any off-tree source arrive from the upstream interface of a CBT router; while packets of any on-tree source may come through either the upstream interface or any downstream interface, depending on where the source is located on the tree.

To modify the mechanism employed by unidirectional trees for Bi-directional PIM-SM and Simple Multicast, the messages of “Update,” “Allow,” and “Block” should be sent only toward the direction to each source. For example, in Fig. 4, if the SF state of the interface to LN_1 is changed to exclude \{\}, R_1 will forward Allow \{S_2\} only to R_2, and R_3 will forward Allow \{S_2\} only to R_3. For CBT, the state messages should be forwarded in both directions to the source and the core, due to having different incoming interfaces for on-tree and off-tree sources.

b) Inter-domain protocols: MBGP/PIM-SM/MSDP is a near-term working solution for inter-domain multicasting [2]. PIM-SM is operated inside each domain. The RP of each domain distributes MSDP Source Active (SA) messages initiated by the source domain to the RPs of neighboring domains. After receiving an SA message, an RP will join the respective source through the path advertised by MBGP if any receiver has joined in its shared tree. The receivers may switch to the PIM-SM source-based tree when the number of packets exceeds a threshold, exactly as in intra-domain PIM-SM, except that this time an inter-domain source-based tree is created. MBGP/PIM-SM/MSDP, however, does not consider the willingness of receivers inside the domain in packet receiving when an SA message is received, but just joins the respective source in a different domain. SF states can help MBGP/PIM-SM/MSDP prune unnecessary branches of cross-domain PIM trees because an RP will join a source in another domain only if at least one of the receivers in the group in its domain is interested in the source.

The long-term solution for inter-domain multicasting is BGP [33]. It supports two levels of routing autonomy and provides the following two advantages: 1) supporting heterogeneity in each domain in terms of network topology and multicast protocols and 2) hierarchically aggregating routing information in the Internet to improve scalability. Consequently, it avoids the following scalability problems when providing inter-domain multicasting by MSDP/PIM-SM.

1) Since Source Active messages are flooded to the entire network, they may cause a lot of overheads when the number of sources is large.

2) PIM-SM trees are used to form inter-domain multicast source-based trees. The size of forwarding tables in border/backbone routers and the number of control mes-
sages exchanged are proportional to the number of sources in a group.

BGMP uses a bi-directional shared tree approach. Like Bi-directional PIM-SM, it treats packets of any source in the same way once the packets have reached the tree, irrespective of whether the source is on-tree. Packets of any source must arrive through the interface in the shortest path to the source. Nevertheless, it is more complicated to provide the support of source filtering for BGMP, because border routers do not have the network topology information at the inter-domain level. As a result, it is hard to determine the shortest path tree spanning all border/backbone routers in order to identify the incoming interface for a source as in Bi-directional PIM-SM. We suggest another approach to learning the incoming interface for a source in a BGMP tree. Once packets of a source are received through an interface, the router purges the source from the source lists of all other interfaces. As such, even without the help of the data from a source to identify the incoming interface, unnecessary sources in a source list can still be limited. The reason is that “exclude” is more likely to be the filter mode of the aggregate entry of an interface because it occurs once any interface has a filter mode of “exclude.” An unnecessary source will exist in the source list of the aggregate entry of an interface only if all the other interfaces are not interested in the source. Fortunately, this rarely happens in a large group according to our simulation result in Section IV. In other words, unnecessary sources in source lists are more likely to be merged as the group size increases.

III. SCALABILITY OF SOURCE FILTERING IN IP MULTICASTING

Scalability is an important criterion to evaluate the effectiveness of IP multicast routing protocols. A scalable mechanism should address the following issues.

1) The forwarding table in an SF router should be kept small, because the table size determines both the cost of a router and the forwarding efficiency, which in turn determines the number of multicast groups the router can support.

2) Both communication and computational overheads incurred by the mechanism should also be kept small, because the overheads increase as the number of multicast groups increases.

In this section, we will discuss the scalability of supporting source filtering for both source-based and shared multicast trees. Previous work related to the scalability of various multicast routing protocols without SF can be found in [40]–[43].

A. Source-Based Trees

Source-based multicast trees, by default, provide support for source filtering. If a DR detects a host in an attached LAN is interested in a source, it joins the corresponding tree; otherwise, it leaves the group. Thus, only DRs need to store SF states, with one entry per host if IGMP v3 is employed. While simple and effective in enabling SF, dense-mode protocols, such as DVMRP and PIM-DM, create the states in routers regardless of whether the router is on the tree. Hence, the average size of memory required by a router adopting a dense-mode protocol is given by

$$M_{DVMPR/PIM-DM} = G \times S \times E_{FT}, \quad (1)$$

where $G$ is the number of multicast groups in the network, $S$ is the average number of sources in a group, and $E_{FT}$ is the size of a forwarding entry. The communication overhead of the entire network when a DR modifies the SF states of a group is given by

$$C_{DVMPR/PIM-DM} = \sum_{j=1}^{S_{MOD}} N_j \times L_m, \quad (2)$$

where $S_{MOD}$ is the average number of source based trees in which a DR needs to join or leave after modifying its SF states of the group, $N_j$ is the number of routers in the $j$th source-based tree involved in forwarding prune or graft messages to their parents, and $L_m$ is the message size of a prune or graft message in bytes. The communication overhead also depends on the frequency of sending refresh messages to maintain soft states. In this paper, we do not consider this factor. The work in [44] shows how the frequency is determined for different link speeds and how the amount of network states should be created at a router.

For MOSPF with SF, each router has to memorize the SF states of all DRs in order to build source-based trees. The average size of memory required by a router using MOSPF with SF is given by

$$M_{MOSPF} = G \times N_{DR} \times E_{SF} + G_{SBT} \times S_{SBT}, \quad (3)$$

where $G$ is the number of groups in the network, $N_{DR}$ is the average number of DRs participating in a group, $E_{SF}$ is the size of an SF state for a group, including a filter mode and a source list, and $G_{SBT} \times S_{SBT}$ is the memory size required to store such information for the average number of source-based trees in which a router is participating. The first term corresponds to the size of an entry for the group in the routing table used to build a source-based tree, while the second term is the size of an entry of the forwarding table. The communication overhead in bytes generated in the network whenever a DR modifies its SF states of a group is given by

$$C_{MOSPF} = N_L \times L_{SF}, \quad (4)$$

where $N_L$ is the number of links involved in broadcasting a new SF state, which is proportional to the number of links in the network, and $L_{SF}$ is the message size of an SF state.

For EXPRESS/SSM, the average size of memory required by a router is given by

$$M_{SSM} = G_{SBT} \times S_{SBT} \times E_{FT}, \quad (5)$$

where $G_{SBT} \times S_{SBT}$ is the memory size corresponding to storing the information for the average number of source-based trees in which a router is participating, and $E_{FT}$ is the size of a forwarding entry. Compared with dense-mode protocols,
EXPress/SSM has the advantage that routers only need to record the states of the groups in which it participates. Compared with MOSPF with SF, it does not need to store the states of DRs because source-based trees are formed by explicit Join messages. The communication overhead is similar to (2), where \( N_j \) is the number of routers in the jth source-based tree involved in forwarding join or leave messages to their parents, and \( L_{SF} \) is the size of a join or leave message in bytes.

B. Shared Trees

For shared tree protocols, the average size of memory required by a router is given by

\[
M_{ST} = G_{ST} \times I_{ST} \times E_{SF},
\]

where \( G_{ST} \) is the average number of shared trees on which a router is located, \( I_{ST} \) is the average number of interfaces belonging to a shared tree, and \( E_{SF} \) is the size of a forwarding entry. The communication overhead in bytes generated in the network whenever a DR changes its SF states of a group is given by

\[
C_{ST} = \sum_{i=1}^{N} \sum_{j=1}^{I_i} M_{SF}^{(ij)},
\]

where \( M_{SF}^{(ij)} \) is the size of the SF message forwarded by router \( i \) through interface \( j \), \( I_i \) is the number of outgoing interfaces belonging to the group, and \( N \) is the number of routers sending an update to adjacent routers. We note that \( M_{SF}^{(ij)} \) is less than \( M_{SF} \) in MOSPF because here part of the SF information is suppressed by the router receiving the SF message.

C. Comparison of Source-Based Trees and Shared Trees With SF

From the previous section, the memory size required by each router for DVMRP, MOSPF, EXPRESS/SSM, and shared trees with SF are \( O(G \times S) \), \( O(G \times S \times N) \), \( O(G \times S) \), and \( O(G \times S) \), respectively, where \( G \) is the number of groups, \( S \) the number of sources, and \( N \) the number of DRs. Compared with DVMRP and EXPRESS/SSM, shared trees with SF use far less memory size than DVMRP so long as some DRs contain “excludes” as their filter modes and have only a few sources in the source lists. In MOSPF, all MOSPF routers are required to store SF states, irrespective of whether the routers are on the multicast tree. For a shared tree with SF, on the other hand, each router only stores the SF states for the groups it has joined. Therefore, the routers on shared trees with SF require the least memory.

The communication overheads of each group generated for the entire network by DVMRP, MOSPF, EXPRESS/SSM, and shared trees with SF are \( O(T \times S) \), \( O(E \times S) \), \( O(T \times S) \), and \( O(T \times S) \), respectively, where \( T \) is the number of edges on a multicast tree, \( S \) is the number of sources, and \( E \) is the number of links in the network. Compared with MOSPF, which floods updates to the entire area, shared tree protocols with SF generate less communication overheads, since messages are distributed only to part of the multicast tree.

In Section II, we mentioned that the communication overheads caused by bi-directional shared trees with SF can be further reduced if the topology information is added to update messages to be forwarded. Since each router needs to store topology information, the memory size required by each router in this case is \( O(G \times S + V + E) \), where \( V \) is the number of routers, and \( E \) is the number of links in the network. This overhead is far less than MOSPF’s, and also less than DVMRP’s in small and medium domains with large group numbers. However, as we will see in the next section, the advantage of using topology information will wane when the group size becomes very large, since redundant messages forwarded are more likely to be merged.

Our mechanism performs aggregation to the source lists of all interfaces in a router whenever an SF update is received. The computational overhead increases as the length of the source list increases. We suggest three approaches to reducing the size of the source lists, which in turn reduces the memory size and computational overhead, as follows:

1) If the filter mode of a table entry is an “include” and the source list is lengthy, a router can modify the SF state as “exclude { }”. Only when the number of packets from superfluous sources exceeds a threshold will the router add the sources to the source list.

2) If the source lists contain many sources from the same domain, the router can replace those sources with a single domain address.

3) When the filter mode of a table entry is an “exclude,” a router can remove those sources at low data rates from the source list. The tradeoff between the size of forwarding tables at routers and the amount of unnecessary packets to be received has been discussed in [45].

These three approaches allow routers to adaptively reduce the size of forwarding tables in a conservative way. They achieve better scalability at the expense of receiving some unnecessary packets. Later in the simulation section, we will demonstrate that even without using these approaches to reducing the size of the source lists in the forwarding table, the shared tree approach with SF still has better scalability than source-based tree, in terms of smaller forwarding tables.

The forwarding states of the shared tree with SF can be further reduced when used with existing mechanisms for forwarding state reduction [25], [28]. Forwarding states of these approaches only need to be created in routers with more than one downstream interface.

IV. PERFORMANCE EVALUATION

This section describes the simulations conducted to verify the arguments we made in the previous sections.

A. Simulation Setup

We generate a network topology consisting of 100 routers based on GT-ITM and employ Doar-Lesile [46] with \((\alpha, \beta) = (0.3, 0.2)\) to determine the probability for a link formed between two routers. We consider 1000 hosts distributed in LANs connected by 50 routers (DRs). The time each host stays in a group is a Pareto distribution with a mean of 20 min. Located at the tree leaf, each DR has an SF state which is changed at a rate of \( y/180 \) per minute, where \( y \) is the total number of hosts participating in the same group in an attached LAN. Each host changes its SF state with the following pattern: a probability of 0.5 to increase...
or decrease the length of the source list and of 0.5 to change the filter mode and the source list.

We simulate one shared multicast tree only, varying the group size (i.e., number of hosts in a group) from 10 to 800, to measure bandwidth efficiency and scalability. The reason we simulate only one group is that routing, forwarding and SF operations for a group do not affect the corresponding operations for the other groups. In our simulation, a sender might or might not be on-tree. For a unidirectional shared tree, the sender unidcasts the packets to the core of the tree, from where the packets are multicast downstream to all recipients. For a bi-directional shared tree, packets are sent to any on-tree router, from where the packets are multicast to all recipients.

There are data packets (multicast datagrams) and control packets (state messages) transmitted on the tree. Routers exchange control packets and update their SF states every 30 minutes. Furthermore, routers update their states in response to an SF change requested by hosts in an attached LAN. Each host can be a data source, sending packets using exponentially distributed inter-arrival times with an average of 12 minutes.

We measure the following performance metrics:

1) Utilization efficiency: we measure the average number of hops each multicast packet traverses on shared trees with and without source filtering. We assume that all data packets are of the same size. Thus, the more the number of tree links traversed, the more bandwidth is consumed.

2) Scalability: we measure protocol overheads (i.e., communications overhead) of shared trees with and without SF. Again, we measure the average number of hops each control packet traverses, and normalize the total number of hops traversed by all control packets by the tree size (i.e., total number of tree links). We also compare the table size required by the source-based tree approach to the shared tree with SF approach. We regard the total size of source lists in the table as the forwarding table size, because the table size is proportional to the number of sources in the source lists. Finally, we discuss the advantage of including the topology information, and compare the number of hops traversed by all control packets.

B. Simulation Results

1) Efficiency of Bandwidth Utilization: Fig. 5 shows the bandwidth efficiency of unidirectional shared trees with SF. The efficiency here is defined as the ratio of hop count traversed by each multicast packet for the approach with SF to that without SF, varying group size from 10 to 800. When the group size increases, the number of hosts wishing to receive packets increases, and the number of links traversed by the packet increases. Without SF, every multicast packet is forwarded to the entire tree. Thus, the number of hops each packet traverses is equal to the tree size. Tree size increases as group size increases. The approach with SF consumes far less bandwidth, thanks to forwarding packets only to active outgoing interfaces. When the group size is less than 100, the approach with SF consumes 20% less bandwidth than that without SF. In other words, SF helps eliminate more than 80% of the data packets transmitted on shared trees.

Fig. 6 depicts the bandwidth efficiency of bi-directional shared trees with SF. As in the case of unidirectional shared trees, SF allows more efficient use of network bandwidth. We also measure the utilization of the bi-directional shared tree but use the SF mechanism for unidirectional shared trees, which forwards SF states only to upstream interfaces. The results show that both approaches are more efficient in bandwidth utilization than the approach without SF, but the bi-directional one has even better result.

2) Scalability: Fig. 7 depicts the protocol overheads of shared trees with SF. We measure the ratio of the average number of hops traversed by each control packet to the tree size, varying group size from 10 to 800. The average number of hops traversed by each control packet decreases as the group size increases. This is because the length of source list in an SF table at a router increases as the group size increases, making control messages highly likely to be suppressed. As the group size exceeds 200, the control overhead becomes very small. This proves that the computational and communication overheads generated by SF messages are limited to a few hops, instead of expanding to the entire tree.
When the group size is small, say less than 50, the control packets of SF may be distributed to a large part of the tree (up to 25% as shown in Fig. 7). On the other hand, SF can improve the bandwidth utilization of shared trees during data forwarding, especially when the group size is small, say less than 50, as shown in Fig. 6 (up to 5%). In short, SF can save 90% of bandwidth spent on transmitting data packets at the expense of 20% of bandwidth spent on exchanging control packets. Typically, control packets are far smaller than data packets in size. Thus, SF can significantly improve the bandwidth efficiency of shared trees. Note that we have also simulated with bi-directional shared trees, and the results are consistent with the unidirectional ones shown in Fig. 7. To conserve space, only the results of unidirectional-shared trees are included in this paper.

Fig. 8 compares the scalability of a source-based tree to that of a shared tree with the proposed SF mechanism in terms of forwarding table size. We average the table size of all corresponding routers for each approach, and calculate the ratio of the average table size of the shared tree with SF approach to the source-based one. We let the number of sources be proportional to group size, and allow each host to freely determine sources of interest. The results show that when the group size exceeds 100, a shared tree with SF needs only 5% of the table size required by a source-based tree. The reason is that the table size of a source-based tree is proportional to the total number of sources (i.e., the number of trees). The table size increases dramatically as the number of sources becomes large. In the shared tree with SF approach, while each router needs to store one SF state per interface, the probability of SF states being efficiently suppressed increases as the number of sources increases. Thus, it needs much smaller table size than the source-based one.

Fig. 9 compares the protocol overheads of bi-directional shared trees with and without topology information. We measure the ratio of the number of hops traversed by all control messages without including the topology information to that with topology information. This curve shows that up to 30% bandwidth can be saved with topology information. The advantage of using the topology information wanes when the group size becomes large because redundant messages forwarded are more likely to be merged.

V. RELATED WORK AND DISCUSSION

This section discusses the implementation issues of SF for multicast routing, presents the related work, and describes how the SF mechanism help improve the performance of related work on IP multicasting.

A. Implementation Issues

To disseminate SF states, we may piggyback the states on the control packets of multicast routing protocols such as PIM-SM and CBT, instead of creating new messages. For example, SF states can be carried by Join_Request messages in CBT and PIM-Join messages in PIM-SM during the shared tree construction. Moreover, our mechanism needs periodic exchanges of SF states between neighboring routers to refresh their states. Again, this can be achieved with soft state mechanisms used by its multicast routing companion. In CBT, each child periodically sends an Echo_Request message to its parent, and the parent sends an Echo_Reply message in reply to indicate reachability. Since CBT is a bi-directional shared tree protocol, a child can piggyback the SF state of its upstream interface on an Echo_Request, and the parent can also attach the SF state of the downstream interface to an Echo_Reply for exchange. Likewise, a child router in PIM-SM can add SF states to periodic Join/Prune messages. Note that the prune list, which is designed to avoid sending duplicated packets in a shared tree, can be eliminated when SF is provided with PIM-SM.

SF can be deployed incrementally in multicast routing protocols. To accommodate existing SF-incapable routers, tunneling can be used. All SF-capable routers form a virtual topology to transmit SF messages. (Note that data are still forwarded in ordinary multicast trees.) For unidirectional shared trees, each SF-capable router needs to know the identity of its virtual upstream router on the shared tree, to which SF messages are tunneled. The SF states of several virtual downstream routers may arrive through the same downstream interface. In this case, we can aggregate the states of all virtual downstream routers to obtain the SF state of a downstream interface. For bi-directional shared trees, in addition to the virtual upstream router, each SF-capable router must also know the identities of all virtual downstream routers.

The SF mechanism needs to work with a multicast routing protocol. When a router or a link fails, we assume the multicast routing protocol companion can rapidly adapt to the new topology. Since SF adopts the soft state mechanism, the state of the interface adjacent to the failed router or link expires after...
a period of time. Therefore, it does not need additional mechanisms to deal with failures.

In CBT, a DR on a multi-access link is elected as the upstream router for all groups. Hence, a router can forward SF messages to the other routers on a multi-access link just as it does to any routing messages in CBT. Similarly, in PIM-SM, the next hop of a router with the best metric to RP is the upstream router. Join suppression must be disabled in order to let the upstream router learn the identities of the downstream ones.

Until now, we have not yet discussed how each receiver learns the addresses of possible sources. This problem is similar to how each receiver learns the addresses of desired multicast groups. In the Mbone, this problem is solved by using Session Announcement Protocol (SAP) [47] that periodically multicasts the detailed information of each group using the format of Session Description Protocol (SDP) [48]. Source addresses can be added to SAP messages. Encrypting source addresses may be required so that only authorized receivers know the addresses. Each receiver should also acquire the source addresses from the certificate authority if some multicast security mechanism is implemented.

B. Related Work

Fenner et al. [49] proposed a mechanism, called IGMP proxy, which extends IGMPv3’s operations in a subnet to a domain with a simple topology. IGMP proxy maintains a bi-directional spanning tree covering all routers in a domain, and eliminates the need to have multicast routing protocols. Each parent plays the role of a DR in IGMPv3 (i.e., sending a query periodically), while each child acts as a host. A router periodically queries its children via IGMPv3 to obtain their SF states. The router then merges the SF states obtained and responds to its parent. When a packet arrives through the upstream interface, a router will forward the packet to the downstream routers which need it. Meanwhile, a packet from a downstream interface will be forwarded to both upstream and downstream routers which require it. The main advantages of IGMP proxy are simple and fully compatible to IGMPv3. However, it is not applicable to large networks because multicast traffic from all groups would be concentrated on a single tree, leading to the traffic to be distributed unevenly and even congestion. Moreover, IGMP v3 Report messages are merged only from downstream to upstream. Routers may still forward unnecessary data upstream until the core is reached (the same problem as shown in Fig. 4). In other words, IGMP proxy is based on a unidirectional shared tree mechanism to deal with bi-directional shared trees. Besides, an IGMP Report message may contain unnecessary sources because this approach does not take the incoming interface for a source into account during packet forwarding. The proposed SF mechanism does not have these problems because we consider both cases as described in Section II-B-2.

Under the current multicast service model, sources cannot learn group membership from network layer protocols. Consequently, sources may transmit data to a tree even though there are no receivers in the group. For dense mode protocols, this is not an issue because multicast trees are driven by data. For sparse mode protocols, however, transmitting data without considering group membership may waste network bandwidth because multicast trees are formed by explicit message exchanges. In this case, sources should not transmit data until there is at least one receiver in the group. Fenner et al. [50] designed a protocol, called Multicast Source Notification of Internet Protocol (MSNIP), to solve this problem for source specific multicast by PIM-SSM. Since PIM-SSM uses source based trees, the DR of a source knows if there is any receiver in the source specific multicast group. The main idea of MSNIP is that according to the forwarding table, the attached DR will inform the source when to start or stop transmitting data. MSNIP, however, does not provide support for shared tree protocols. The proposed SF mechanism can help extend MSNIP to work with shared trees. With the SF information in RPs/cores, MSNIP can regard RP/cores in shared trees as DRs in the original MSNIP, enabling the RP/cores to notify all sources when to start or stop sending data.

He [51] adopted the MSNIP concept in IGMP proxy to prevent a router from sending unnecessary data to the upstream router. Sources send MSNIP Source Interest Solicitation messages to the attached DRs and ask for notification. Each DR will collect messages from all the attached sources and send a Source Interest Solicitation message to its upstream router. Each router will combine solicitations from all its downstream routers and pass them to its upstream router hop by hop. Eventually, the RP will have solicitations from all sources. Based on IGMPv3 reports from all downstream routers, the RP can determine which sources are of interest to the group, and forward Source Group Membership Reports downstream, asking the corresponding sources to start or stop sending data. Since an RP has to memorize solicitations from all sources, this mechanism may suffer from the scalability problem. Compared with this approach, SF with shared trees is more scalable, because the SF information is forwarded to each source, not only to the core.

Levine and Garcia-Luna-Aceves [18] proposed a “stream” concept in shared tree protocols. Each source can initiate multiple streams to a group, and each receiver can subscribe to (i.e., join) any streams of interest. Since data of a stream are forwarded only to the outgoing interfaces through which some receivers are subscribing to the stream, the mechanism can be considered as providing source filtering. Besides, this mechanism allows hosts receiving only part of data from a source. In other words, it provides finer granularity of “include” filter mode than IGMP v3, but does not offer the option of “exclude.” This mechanism has the following problems. It enables this functionality at the expense of high overheads to maintain router IDs, as described in Section II-B-2. In addition, their states cannot be aggregated. Thus, it may limit the number of streams supportable for each group.

C. Unified Solution for Single Sender and Multi-Peer Services

The IETF proposes the Multicast Address Allocation Architecture (MAAA) to use multicast addresses more efficiently. While MAAA is feasible, its complexity and scalability are still an open issue. Simple Multicast (SM) [17], EXPRESS [11], and Source-Specific Multicast (SSM) [12] provide solutions to relieving the requirement of global uniqueness of multicast address allocation in the MAAA model. They couple a Class-D address with a core address or a source address to form a multicast address, i.e., (C,G) or (S,G), and eliminate the needs of

\footnote{Simple Multicast is a shared tree protocol. The mechanism discussed in previous sections can be used to provide source filtering in SM.}
address allocation and the group-and-core mapping as in CBT or PIM-SM [52].

EXPRESS is a multicast routing protocol which provides support for both single-sender and multi-peer services. For sessions with multiple senders, EXPRESS creates a separate channel (i.e., a tree) for each sender, or use a common channel to relay data from all senders. Comparing these two options for multi-peer service, we find that the former is less scalable because it will create lots of forwarding states for each group, while the latter causes high end-to-end delay because data will be sent to the core of the relaying channel first, from where the data are forwarded to all receivers. Zappala et al. [53] proposed to use multiple relaying channels to solve the latter problem of EXPRESS. Each receiver first subscribes to a nearby channel. Each sender sends data to a nearby proxy, i.e., the core of a channel, from where the data are relayed to other proxys. Each proxy then in turn forwards the data to all receivers. This approach can reduce the end-to-end delay. However, it operates at the application layer and might become the bottleneck in multicast forwarding. Furthermore, from the simulation result in [53], this mechanism consumes a lot more bandwidth because data have to be relayed between proxys.

Simple Multicast (SM), which is designed for multi-peer service, can also be extended to single-sender service by using the source address as the core address in the (C,G) pair. However, SM alone cannot prevent other sources from sending data to a single-sender group because it uses shared trees to transmit packets. This problem can be easily solved when SF is used in conjunction with SM. Each router knows whether a multicast group is single-sender or multi-peer from the class-D address of the group. For a single sender group, “include” is the only permitted filter mode in LAN [54]. In a routing domain, each router using SM with SF sets the filter mode as “include” and puts the sender’s address in the source list. Since this is the only permitted SF state on the whole tree, message exchanges described in Section II-B can be eliminated.

Comparing these two unified solutions for both single sender and multi-peer services, we find

1) both solutions grow the same source-based tree rooted at the sender for the single sender service, and

2) for multi-peer service, SM with SF causes less end-to-end delay because SM is a bi-directional shared tree so that data newly received from an off-tree sender will be forwarded to the entire tree.

D. Authorization

The current IP multicast model proposed by Deering does not cover the issue about authorization of senders and receivers. In other words, anyone can freely join or send data to a multicast group. The lack of authorization may defer multicast from being deployed widely and commercially [20], [55]. Several key management mechanisms [56]–[58] have been proposed to prevent unauthorized receivers from acquiring the key and decrypting the ciphertext of a multicast group. These protocols, however, cannot prohibit unauthorized hosts from sending to or receiving data from a multicast group. Besides, a new key needs to be distributed to all members when a member joins or leaves the group.

6The range of addresses 232/8 has been assigned to SSM for providing single-sender service in the standard service model. Therefore, applications using Class-D addresses within this range will be classified as single-sender service.

to maintain join and leave privacy. Administratively Scoped IP Multicast [59] enables senders to multicast data only to receivers in an administrative domain. However, it does not work for some scenarios in which authorized receivers are distributed over different administrative domains.

Authorization can be classified into:

1) Only the hosts authorized by the sender or the network administrator can be the receivers. In EXPRESS, a channel key is distributed to each on-tree router. Thus, only the Join messages associated with the right channel key will be accepted by on-tree routers. Xcast [60] includes the addresses of all authorized receivers in the destination field of a packet. Thus, packets will be forwarded only to the authorized receivers

2) Only the senders authorized by all receivers can forward data to the multicast tree. This paper falls in this category.

3) Only the senders authorized by the group administrator can forward data to the multicast tree.

The category (3) is studied in [61]. It uses the core of a bi-directional shared tree as the authorization server of a group. The routers between the core and a permitted source have to store the address of the source. One disadvantage of this approach is that routers near the core need to store a large number of addresses of permitted sources. With SF, this problem can be easily overcome. An authorization server periodically multicasts the addresses of permitted sources to the group, while each DR excludes unauthorized sources from its SF state. The main advantage of this integration is that no routers in a shared tree have to store the addresses of permitted sources. Furthermore, even without sender authorization, SF can help block the denial-of-service attack out of the tree. When a host detects that it is being attacked, it can just modify the SF state to prevent system failure. Network administrators can use existing network management tools like SNMP to configure the SF states of on-tree routers.

E. Forwarding State Aggregation

Fei et al. proposed a method, called Aggregated Multicast [26], using an aggregate shared tree in a domain in lieu of multiple sub-trees from multiple inter-domain multicast groups if the shared tree can cover all of these sub-trees. When a multicast packet arrives at an ingress border router, the router substitutes the original destination address with the multicast address of the aggregate tree. Thanks to aggregation, this approach can reduce the number of forwarding states at routers inside the domain. However, it cannot avoid leaking data of a group to other groups connected by the same aggregate tree. This problem occurs when the sub-tree of the leaking group is only part of the aggregate tree so that data will be leaked to unnecessary branches of the tree. Worse, when dynamic group membership is considered, an aggregate tree may grow over time and more branches may be added to the tree, which may cause more data leaking to other groups connected by the aggregate tree. In [26], this problem is solved as follows. An aggregate tree is allowed to grow or be trimmed for dynamic membership. However, a group will be switched to another new aggregate tree when the amount of data leaking to the current aggregate tree exceeds a threshold. This switching process, however, may affect the QoS of existing
members in the group [26]. We can solve this problem by cooperating aggregate trees with SF. Data will not leak to unnecessary branches of aggregate trees because SF can help prune the unnecessary branches of the trees as it does for any shared trees.

Song [30] proposed a similar approach to aggregating forwarding states. Again, SF can help [30] solve the problem of injecting unnecessary data to unnecessary branches of the aggregate tree. In addition, SF can also help [30] eliminate the needs of the complicated selection algorithm of dynamic filtering points [30].

Preference clustering [21] is an approach to clustering group members with similar preferences to sources. It provides a mechanism to cluster multiple groups with similar sources and receivers on a shared tree, further reducing the number of addresses used and the size of the forwarding tables. Again, SF can help this approach avoid leaking unnecessary data to the aggregate tree, as it does for [26] and [30].

In summary, compared to existing work, SF is more scalable. Besides, it will work with all existing intra and inter-domain multicast routing protocols and improve their performance. SF enables SM to serve both single-source and multi-peer services and receivers can be forwarded to the multicast tree. Finally, SF can improve the performance of many mechanisms for multicast forwarding state aggregation.

VI. CONCLUSION

In this paper, we have proposed a new mechanism to support source filtering in IP multicast routing protocols. The details of the operations have been described and the performance of the proposed mechanism has been evaluated via simulations. The results show that, as compared to multicasting without the capability of source filtering, our mechanism allows better bandwidth utilization and scalability in terms of lower control overheads and smaller forwarding table size. Thus, it achieves truly efficient use of resources for IP multicasting. Moreover, we have addressed the implementation issues, and discussed how to improve the performance of related work by source filtering.

REFERENCES

[41] L. Wei and D. Estrin, “Multicast routing in dense mode and sparse
modes: simulation study of tradeoffs and dynamics,” in IEEE Interna-
tional Conference on Computer Communications and Networks, 1995,


performance considering multicast routing protocol,” in IEEE ICNP,
2000, pp. 95–104.


[45] P. I. Radoslavov, E. Estrin, and R. Govindan, “Exploiting the Band-
width-Memory Tradeoff in Multicast State Aggregation,” USC Com-

IEEE INFOCOM, 1993, pp. 82–89.

[47] M. Handley, C. Perkins, and E. Whelan, “Session Announcement Pro-

2327, Apr. 1998.

[49] B. Fenner et al., “IGMP-Based Multicast Forwarding (IGMP Prox-
ying),” IETF Internet Draft, draft-ietf-magma-igmp-proxy-00.txt, Nov.

cation of Interest Protocol (MSNIP),” IETF Internet Draft, draft-ietf-

[51] H. He, “MSNIP Extension for IGMP Proxying,” IETF Internet Draft,

[52] H. Lin and S. Lai, “Core placement for the core based tree multi-
1053–1949.

[53] D. Zappala and A. Fabbri, “Using SSM proxies to provide efficient

[54] H. W. Holbrook and B. Cain, “Using IGMPv3 and MLDv2 for Source-
Specific Multicast,” IETF Internet Draft, draft-holbrook-idmr-igmpv3-


[58] I. Chang et al., “Key management for secure internet multicast using
Boolean function minimization techniques,” in IEEE INFOCOM,


1656–1661.

De-Nian Yang was born in Taiwan in 1977. He received his BS degree from the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, in 1999. Exempted from the normal entrance exam to the graduate school, he is currently Ph.D. candidate in the EE department of National Taiwan University. He received the best student paper award in ICME 2000. His research interests include broadband Internet, QoS, and multicasting.

Wanjuan Liao received the BS and MS degrees from National Chiao Tung University, Taiwan, in 1990 and 1992, respectively, and the Ph.D. degree in Electrical Engineering from the University of Southern California, Los Angeles, California, USA, in 1997. She joined the Department of Electrical Engineering, National Taiwan University (NTU), Taipei, Taiwan, as an Assistant Professor in 1997. Since August 2005, she has been a full professor. Her research interests include wireless networks, multimedia networks, and broadband access networks.

Dr. Liao is actively involved in the international research community, and has served on the program committees of many international conferences. She was an editor of Computer Networks, and is currently an editor of IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS and IEEE TRANSACTIONS ON MULTIMEDIA. Dr. Liao has received many research awards. Two papers she co-authored with her students received the Best Student Paper Award at the First IEEE International Conference on Multimedia and Expo (ICME) in 2000, and the Best Paper Award at the First International Conference on Communication, Circuits and Systems (ICCCAS) in 2002. Dr. Liao was elected as one of Ten Outstanding Young Women in Taiwan in 2000 and was listed in the Marquis Who’s Who in 2001–2005.

Chang-Jung Kao was born in Taiwan in 1978. He received the BS and MS degrees from the Department of the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, in 2000 and 2002, respectively. His research interests include wireless networks and multimedia communications.