# Source Filtering in IP Multicast Routing*

*Chang-Jung Kao, De-Nian Yang, and Wanjiun Liao*+
Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan
Email: wjliao@cc.ee.ntu.edu.tw

***Abstract*** *IP multicasting is an efficient group communication mechanism. It avoids transmitting packets from a sender to each of the receivers separately. With the current IP multicast mechanism, once having joined a group, a 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 in IP multicasting. Source filtering is the ability for an individual host 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 host does not want to receive. We investigate the issue of source filtering in the context of multicast routing protocols, and propose a solution to support source filtering in IP multicast routing. We have also conducted simulations to evaluate the performance of the proposed mechanism. The results show that, compared to multicasting without the capability of source filtering, our mechanism allows better bandwidth utilization and scalability, thus achieving a truly efficient use of resources for IP multicasting.*

***Keywords:*** *IP multicasting, source filtering*

## 1. Introduction

IP multicasting is an efficient group communication mechanism [1]. It avoids transmitting packets from a sender to each of the receivers separately. It is not necessary for a host to join a group in order to send data to the group. An individual host can freely join or leave a multicast group, without affecting the other group members. No restriction is placed on a group member in terms of its physical location and the number of groups it can participate in. Multicast data delivery on the Internet can be cooperatively provided through two mechanisms: local group management (e.g., IGMP [2-4] and RGMP [5]) and global multicast routing (e.g., DVMRP [6,7], MOSPF [8], CBT [9], and PIM [10,11]).

Source filtering is the ability for an individual host 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 host does not want to receive. This concept was first introduced in IGMP v3 [4]. With source filtering, a Designated Router (DR) will forward multicast datagrams of a source to an attached local network only when at least one host in the network wishes to receive packets from the source. Thus, it avoids network bandwidth wastage caused by delivering unnecessary packets to local networks.

To date, source filtering has been discussed only in the context of local group management (e.g., by IGMP v3), but not in multicast routing. Without being exchanged as routing information, the source filtering (SF) information provided by hosts in each local network is available only to their immediate neighboring DRs. Thus, even if all the hosts in a local network have 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 with IP multicasting, SF information should be propagated along with routing information across the global network.

This paper investigates the issue of source filtering in IP multicast routing. We propose a mechanism to support source filtering in multicast routing protocols, with the following objectives:

* This work is supported in part by the National Science Council, Taiwan, under Grant Number NSC 90-2213-E-002-114 and Grant Number NSC 89-E-FA06-2.
+ This author is also with the Graduate Institute of Communications Engineering, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan.

(1) Making truly efficient use of network bandwidth. A multicast router will receive a multicast packet from a particular source if and only if at least one host in its directly attached network or from any downstream routers (i.e., descendant routers in the original tree) wishes to receive from the source.
(2) Promptly reacting to SF state updates from local networks.
(3) Ensuring scalability, in terms of low overhead in computing (i.e., small forwarding table) and communications (i.e., low control overhead).
(4) Minimizing modification to existing multicast routing protocols to support source filtering.

Note that since IGMP v3 is the dominant local group management protocol supporting source filtering, we will follow its message format and its definition of source filtering in the proposed mechanism.

The rest of the paper is organized as follows. Section 2 presents the proposed mechanism to support source filtering in IP multicast routing protocols. Section 3 discusses the issue of scalability. Section 4 shows the simulation results. Finally, the conclusion is made in Section 5.

## 2. Source filtering in multicast routing protocols

Based on where the tree is rooted, existing multicast routing protocols can be classified into two types: source-based trees (e.g., DVMRP, MOSPF, PIM-DM) and shared trees (e.g., PIM-SM, CBT). The support of source filtering in source-based trees is straightforward. The formation of a source-based tree is data driven. The tree is constructed when a flow starts and is destroyed when the transmission is finished. For DVMRP [6,7] and PIM-DM [11], a tree is rooted at the flow source and spans all the group members. When no one in any directly attached network is interested in packets from a particular source, the DR just prunes itself from the tree. Thus, no packets from the source will be forwarded to the DR. For MOSPF, Group-Membership LSAs should be modified to carry the SF state of IGMP v3. Upon receiving the first multicast packet of a (source, group) pair, the router computes the shortest path tree based on the topology information and the source filtering states stored in its local database. While this scheme allows relatively straightforward provision of source filtering, the overhead is substantial. As far as communication overhead is concerned, a DR has to send several control messages per (S,G) pair to other routers whenever a DR's SF state is modified[1]. Thus, MOSPF requires larger routing table size because each router needs to record the SF states of all other routers in the same OSPF area.

Providing source filtering to shared trees, however, is a challenge, because each such tree is shared by all sources of the group. The problem becomes even more complicated when considering the following issues:
(1) In a shared tree, a source may or may not be on-tree (i.e., the source may or may not be a group member). In addition, packets may be delivered unidirectionally or bi-directionally.
(2) Shared trees allow better scalability than source-based trees. However, it is a challenge for multicast routing protocols to enable 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 section III and IV, we will prove that our mechanism is more scalable than the source-based tree approach.

Our proposed mechanism is comprised of two parts: forwarding part and message exchange part. In the forwarding part, each router determines which SF states should be stored in each interface. This SF information is key to determining to which interfaces a received multicast packet should be forwarded. In the message exchange part, routers exchange messages to ensure the correctness of the SF tree, which filters unnecessary multicast packets to minimize network bandwidth wastage.

### 2.1 Forwarding

Upon receipt of a multicast packet from an interface, a router searches its forwarding table and determines how to route it. To support source filtering, the packet should be forwarded only to those outgoing interfaces where at least one descendant router downstream of the tree wishes to receive data from the packet's source. Thus, SF routers must be able to identify to which outgoing interfaces the

[1] For DVMRP and PIM-DM, the DR has to send a prune or graft message for each leaving or joining group whenever the SF states of the DR is changed. For MOSPF, the DR also needs to send Group-Membership LSAs to all the other routers in the same OSPF area. The overhead of DVMRP/PIM-DM may be large if the modification of SF states is correlated to many sources, especially as the filter mode of the DR is changed. The amount of MOSPF overhead is enormous because Group-Membership LSAs have to be flooded in the entire area.

packet should be forwarded.

One approach is associating each outgoing interface with multiple SF states, each of which records the state of each LAN (denoted as downstream LAN) attached to any descendant router. Each router then has the complete SF state of each downstream LAN. Upon receiving multicast packets, each router can unambiguously identify the outgoing interfaces to forward the packets. This approach, however, 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 scalable as the number of downstream LANs increases.

In fact, routers do not need to maintain one SF state per downstream LAN. Instead, we can associate each outgoing interface with one single SF state which summarizes the requirements of all the downstream LANs. An outgoing interface is *active* for an (S,G) pair only when at least one host in any LAN located downstream of the interface is interested in receiving source S's packets. This could significantly reduce the forwarding table size of each router and improve routing efficiency.

Forwarding mechanism

The SF router associates each interface with an SF state, the value of which may be the aggregation of the requirements of all the downstream routers, or the state obtained from its directly attached network. Each router maintains a forwarding table, each entry of which stores the SF state information for one interface. Each SF state includes a group address, a filter mode, and a source list. Upon receiving a source S's packet with group address G, the router searches its forwarding table and identifies the entries of the active outgoing interfaces for the (S,G) pair. An outgoing interface is active if either of the following conditions hold:

(1) The filter mode is an "include," and the source address of the received packet is in the associated source list.
(2) The filter mode is an "exclude," and the source address of the received packet is not in the associated source list.

## 2.2 Message exchange

### 2.2.1 Unidirectional shared tree

In an unidirectional shared tree, SF routers exchange messages to learn the SF states of group members so as to avoid transmitting unnecessary packets across the network. This exchange may be achieved by flooding. However, flooding is not scalable and suffers from high communication overhead, as in source-based trees like MOSPF. We attempt to improve scalability and to minimize the overhead. Thus, to exchange messages, each SF router first merges the SF states of all outgoing interface (including both interfaces downstream to child routers and interfaces to directly attached local networks) belonging to the same group. It then propagates the resultant state upstream to the parent router in the shared multicast tree.

Message exchange mechanism (I)

Considering all the interfaces of an SF router, we assume there are $m$ "include" filter modes with $m$ source lists, i.e., $I_1$, $I_2$.... $I_m$, respectively, and $n$ "exclude" filter modes with $n$ source lists, i.e., $E_1$, $E_2$.... $E_n$, respectively. For an SF state being forwarded upstream, its filter mode $M_U$ and source list $S_U$ are determined as follows:

(1) Let $I = I_1 \cup I_2 \cup ..... \cup I_m$, and $E = E_1 \cap E_2 \cap ..... \cap E_n$
(2) If $n = 0$, set $M_U$ = include, and $S_U = I$; otherwise, set $M_U$ = exclude, and $S_U = E\text{-}I$.

IGMP v3 allows individual hosts to freely change their SF states. As a result, the DR may have its SF states changed at any time, and thus requires its upstream routers to reflect this change accordingly. One solution to this problem is that as soon as an SF state update is received from an interface, a router merges the table entries of all the other interfaces (i.e., all the outgoing interfaces) to generate a new state, and then forwards the new state to its immediate upstream (i.e., parent) router. The operation repeats until a router is reached, where the updated SF state is a subset of the SF state associated with the interface from which the updated SF state is received. This approach, however, suffers from three drawbacks:

(1) The computational complexity of the merge operation performed by an SF router is proportional to the number of interfaces times the size of the source lists of the interfaces. Hence, if a router needs to generate a new SF message by merging all the table entries whenever an update is received, the computational overhead becomes very large as the number of interfaces increases or as the source lists become large.
(2) Generated by merging the table entries of all the interfaces belonging to the same group, a new state message must be propagated upstream, irrespective of whether the new state remains unchanged.
(3) If the new state is only slightly different from the old state (for example, the filter mode stays unchanged but the source list adds a few more new sources), it is inefficient to perform a merge to all entries and then forward the entire resultant state upstream.

In short, this approach is inefficient due to its high computational overhead (due to (1)) and high communication overhead (due to (2) and (3)). Again, it suffers from the problem of scalability. In the following, we will describe how our mechanism works. To solve problem (2), we introduce a new entry in the forwarding table, called merge entry, storing the resultant state after merging the SF states of all the outgoing interfaces. It is unnecessary to forward the new state upstream whenever the merge entry remains unchanged after merging. To solve problem (3), we use three IGMP messages[2]: "Update," "Allow," and "Block." Defined in IGMP v3, the Update message is used to deliver an entire SF state; both "Allow" and "Block" messages have no filter mode, but a source list only. Upon receiving an "Allow" message with a source list of $S_a$ from an interface, a router interprets that the system wishes to receive the additional sources indicated by $S_a$ from the interface. On the contrary, upon receipt of a "Block" message with a source list of $S_b$, a router considers the system no longer interested in receiving $S_b$'s packets.

Message exchange mechanism (II)

Some notations are defined as follows.

(1) Before a router receives an SF state update from an interface: let $F_{mo}$ and $S_{mo}$ be the filter mode and the source list, respectively, in the merge entry, and $F_{io}$ and $S_{io}$, the filter mode and the source list, respectively, associated with the interface from which the update is received.

(2) After the router has received an update: let $F_{mn}$ and $S_{mn}$ be the filter mode and the source list, respectively, in the merge entry, and $F_{in}$ and $S_{in}$, the filter mode and the source list, respectively, associated with the interface from which the update is received.

The SF router works as follows.

(1) Upon receipt of an "Allow" message with a source list of $S_a$
   (a) If $F_{io}$ is include, it sets $S_{in} = S_a \cup S_{io}$.
   (b) If $F_{io}$ is exclude, it sets $S_{in} = S_{io} - S_a$.
   (c) If $F_{mo}$ is include, it sets $S_{mn} = S_a \cup S_{mo}$. Besides, if $S_a - S_{mo} \neq \varnothing$, the router forwards an "Allow" message with a source list of $S_a - S_{mo}$ toward the upstream router.
   (d) If $F_{mo}$ is exclude, it sets $S_{mn} = S_{mo} - S_a$. Besides, if $S_{mo} \cap S_a \neq \varnothing$, the router forwards an "Allow" message with a source list of $S_{mo} \cap S_a$ toward the upstream router.

(2) Upon receipt of a "Block" message with a source list of $S_b$
   (a) If $F_{io}$ is include, it sets $S_{in} = S_{io} - S_b$.
   (b) If $F_{io}$ is exclude, it sets $S_{in} = S_{io} \cup S_b$.
   (c) If $F_{mo}$ is include, it checks if any other interfaces are interested in receiving packets from source list $S_{mo} \cap S_b$. If no one wishes to receive packets from $S_p$, a subset of $S_{mo} \cap S_b$, the router sets $S_{mn} = S_{mo} - S_p$, and then forwards a "Block" message with a source list of $S_p$ toward the upstream router.
   (d) If $F_{mo}$ is exclude, it checks if any other interfaces are interested in receiving packets from source list $S_b - S_{mo}$. If no one wishes to receive packets from $S_p$, a subset of $S_b - S_{mo}$, the router sets $S_{mn} = S_{mo} \cup S_p$, and then forwards a "Block" message with a source list of $S_p$ toward the upstream router.

(3) Upon receipt of a change to the SF state associated with an interface from a directly attached LAN, or upon receipt of an "Update" message with a filter mode of $F_u$ and a source list of $S_u$ from an immediate downstream router,
   (a) If both $F_u$ and $F_{io}$ are include, the router regards it as having received an "Allow" message with a source list of $S_u - S_{io}$ and a "Block" message with a source list of $S_{io} - S_u$.
   (b) If both $F_u$ and $F_{io}$ are excludes, the router regards it as having received an "Allow" message with a source list of $S_{io} - S_u$ and a "Block" message with a source list of $S_u - S_{io}$.
   (c) If $F_u$ and $F_{io}$ are exclude and include, respectively, the router sets $F_{in}$ = exclude and $S_{in} = S_u$. Besides,
      (i) If $F_{mo}$ is include, the router checks if other interfaces are interested in receiving packets from source list $S_u \cap S_{io}$. If no one wishes to receive packets from $S_p$, a subset of $S_u \cap S_{io}$, the router sets $F_{mn}$ = exclude and $S_{mn} = (S_u - S_{mo}) \cup S_p$, and then forwards upstream an "Update" message with a filter mode and a source list of $F_{mn}$ and $S_{mn}$, respectively.
      (ii) If $F_{mo}$ is exclude, the router checks if other interfaces are interested in receiving packets from source list $S_u \cap S_{io}$. If no one wishes to receive packets from $S_p$, a subset of $S_u \cap S_{io}$, the router sets $S_{mn} = (S_u \cap S_{mo}) \cup S_p$), and then forwards upstream an

[2] IGMP v3 defines six messages for source filtering, namely, Is_Include, Is_Exclude, To_Include, To_Exclude, Allow, and Block. Here we borrow the terms of "Allow" and "Block" directly from IGMP v3, but use "Update" to represent both "To_Include" and "To_Exclude." Note that we do not need "Is_Include" and "Is_Exclude" in our mechanism, since these two messages are used when IGMP v3 probes the presence of membership in a LAN.

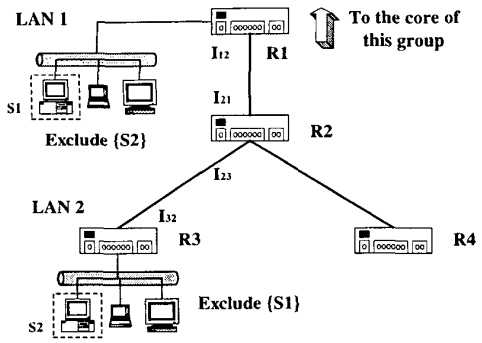


**Figure 1. A bi-directional shared tree**

"Update" message with a filter mode and a source list of $F_{mn}$ and $S_{mn}$, respectively.

(d) If $F_u$ and $F_{io}$ are include and exclude, respectively, the router sets $F_{in}$ = include, $S_{in}$ = $S_u$, and then merges the SF states of all the interfaces in the router to update the merge entry. The update message is forwarded to the upstream router, with a filter mode and a source list of $F_{mn}$ and $S_{mn}$, respectively.

To determine $F_{mn}$ and $S_{mn}$ on receiving a join or leave message, a router merges the SF states of all its interfaces. An approach to avoiding the complicated merging process is to regard join/leave messages as Update messages. Thus, it can set $F_{io}$ to an "include" and $S_{io}$ to an empty when receiving a join message. Similarly, it sets $F_u$ to an "include" and $S_u$, an empty in a leave message.

### 2.2.2 Bi-directional shared tree

In a bi-directional shared tree, the interface[3] of a router from which a source's packet is received may not be the upstream interface of the router (i.e., the interface to the parent router in the shared tree). Similarly, the outgoing interfaces may not be the downstream interfaces (i.e., the interfaces to the child routers in the shared tree). Thus, the mechanism described for unidirectional trees in Sec. 2.2.1 may not be directly applicable to bi-directional trees, for the sake of efficiency of bandwidth utilization. The reason is explained as follows. Fig. 1 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 $LAN_1$ and $LAN_2$, respectively. The entry for $LAN_1$ in $R_1$'s forwarding table has a state of exclude $\{S_2\}$; the entry for $LAN_2$ in $R_3$'s forwarding table has a state of exclude $\{S_1\}$. With the unidirectional shared tree mechanism, SF states are forwarded upstream only. Suppose that R4 also does not want to receive multicast packets from source $S_1$. $R_1$ maintains this SF information and thus will not forward $S_1$'s packets to $I_{12}$. $LAN_1$ has an SF state of "exclude $\{S_2\}$." However, since SF states are not propagated downstream, the table entries for interfaces $I_{21}$ and $I_{32}$ in $R_2$ and $R_3$, respectively, do not maintain the states about $LAN_1$. As a result, $S_2$'s packets are still forwarded to $LAN_1$, even though no hosts in $LAN_1$ wish to receive packets from $S_2$.

To avoid the problem described above, our mechanism should be extended to allow the delivery of SF states bi-directionally (i.e., both upstream and downstream). One implementation is forwarding SF states toward all interfaces. While simple, this approach transmits many redundant SF state messages on the links of the tree which do not lead to the source. To remedy this problem, we can consider not including a source in the source list of the SF state associated with an interface unless the source's packet is received from that interface. A solution is to associate each SF router with the domain topology information, as in OSPF, and then determine the shortest path tree rooted at the core of all the routers. With the shortest path tree, a router can easily identify the incoming interface of any on-tree source, and propagates the SF state only toward the identified incoming interface. In short, the number of extra states is reduced at the expense of a larger forwarding table (to store the topology information) in each router. Note that the second approach performs better only when the group size is small. This is because as group size increases, more users join the group, which in turn increases the number of redundant updated SF states merged in the successive hops. Thus, the advantage of integrating topology information wanes as group size becomes large. We will examine this issue again later in the simulation section.

Off-tree source senders are treated differently in bi-directional PIM-SM [12] and CBT. With Bi-directional PIM-SM, packets are unicast to the core of the tree. Once a packet has reached any on-tree router, the router distributes it to all the recipients both in the upstream and downstream directions. Thus, any on-tree router receives packets only from the interface of the shortest path to the source, irrespective of whether the source is located. As a result, on-tree routers cannot tell where the packet source is on-tree or off-tree. With CBT, however, packets from an off-tree source are tunneled to the core of the tree, from where the packets are forwarded downstream to all the recipients. Thus, packets from any off-tree source in a CBT router must come from the upstream interface of the router. However, packets from any on-tree source may come from either the upstream interface or any

[3] This interface is called the incoming interface of the source.

downstream interface.

To modify the mechanism employed by unidirectional trees for bi-directional PIM-SM, the messages of "Update," "Allow," and "Block" are forwarded only toward sources. For example, in Fig. 1, if the SF state of the interface to $LAN_1$ is changed to exclude {}, $R_1$ will forward Allow $\{S_2\}$ only to $R_2$, and $R_2$ will forward Allow $\{S_2\}$ only to $R_3$. In CBT, the state messages should be forwarded in the both directions of the source and the core, due to having different incoming interfaces for on-tree and off-tree senders.

## 3. Scalability of source filtering in IP multicasting

Scalability is an important criterion to evaluate the effectiveness of an IP multicast routing protocol. Thus, the mechanism to support source filtering in IP multicasting should be scalable, with the following objectives:

(1) A forwarding table in an SF router should be small, because the forwarding table size determines the memory size (i.e., the cost) of the router and forwarding efficiency, which in turn determines the number of multicast groups the router can support.

(2) The communication overhead and the computational overhead incurred by the mechanism should be as small as possible, because the overhead increases as the number of multicast groups increases. Thus, with a small overhead, the mechanism is applicable even for a sizeable network.

In the following, we will investigate the scalability from the perspectives of using both source-based and shared multicast trees with the support of source filtering. Previous work related to the scalability of source-based tree and shared tree may be found in [13-15].

### 3.1 Source-Based Trees

As mentioned earlier, source-based multicast trees, by default, support the capability of 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 are required to store SF states, with an entry for each host if IGMP v3 is employed. While simple and effective in enabling SF, the memory size of a router adopting DVMRP or PIM-DM is:

$$C_{memory} = \sum_{i=1}^{G} \sum_{j=1}^{S_i} I_j^i \times E \qquad (1)$$

where $G$ is the number of groups in the network, $S_i$ is the number of sources in group $i$, $I_j^i$ is the number of interfaces related to (group, source) = (i, j) on the source-based tree, and $E$ is the size of a forwarding entry in bytes. The memory size is the same as the case without source filtering. Besides, the communication overhead of the entire network when a DR modifies the SF states of a group is:

$$C_{communication} = \sum_{j=1}^{S} N_j^i \times M \qquad (2)$$

where $S$ is the number of sources which appears in any source list in any router, and $N_j^i$ is the number of routers reponsible to forward a prune or a graft message to its parent router, and $M$ is the message size of a prune message or graft message in bytes.

For MOSPF, in addition to the topology information used by OSPF, the memory size required by each router is:

$$C_{memory} = \sum_{i=1}^{G} \sum_{j=1}^{N_i} SF_j^i \qquad (3)$$

where $G$ is the number of groups in the network, $N_i$ is the number of DR participating in group $i$, and $SF_j^i$ is the memory size required to record the SF states of router $j$ about group $i$. Besides, the communication overhead in bytes generated in the network whenever a DR modifies its SF states of a group is:

$$C_{communication} = L \times SF \qquad (4)$$

where $L$ is the number of links involved in broadcasting the new SF state, and $SF$ is the size of the message with the new SF state.

### 3.2 Shared Trees

In source-based tree multicasting, a tree is constructed for each (source, group) pair. In the shared tree approach, a tree is shared by all sources of the group. Due to maintaining table entries on a per group basis, the shared tree approach demands a smaller forwarding table and gives better scalability. The memory size required by a shared tree with SF router is:

$$C_{memory} = \sum_{i=1}^{G} \sum_{j=1}^{I_i} SF_j^i \qquad (5)$$

where $SF_i^j$ is the size of the SF states of group $i$ for interface j, and $I_i$ is the number of interfaces in a shared tree router for group $i$. Besides, the communication overhead in bytes generated in the network whenever a DR changes its SF states of a group is:

$$C_{communication} = \sum_{i=1}^{N} \sum_{j=1}^{I_i} SF_j^i \qquad (6)$$

where $SF_i^j$ is the size of the SF states which router $i$ forwards from interface $j$, and $N$ is the number of routers receiving an update from adjacent routers.

## 4. Performance evaluation

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

### 4.1 Simulation setup

We use GT-ITM to generate a network topology with 100 routers and employ Doar-Lesile [16] with $(\alpha, \beta)$ = (0.3, 0.3) to determine the probability of a link between two routers. We consider 1000 hosts distributed on LANs connected by 50 routers (DRs). The time each host stays in a group is Pareto distributed with a mean of 20 min. Located at the tree leaf, each DR has an SF state changed at a rate of y/180, 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 group size (i.e., number of hosts in a group), to measure bandwidth efficiency and scalability. Only one group is simulated because routing, forwarding, and SF operations of a group do not affect the corresponding operations of other groups. In our simulation, a sender may or may not be on-tree. To transmit packets to a unidirectional shared tree, the sender unicasts 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 the recipients.

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

We measure the following items:

(1) Utilization efficiency: we measure the average number of hops each multicast packet traverses in 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 the bandwidth consumed by the mechanism.

(2) Scalability: we measure protocol overhead (i.e., communications overhead) of shared trees with and without SF. Again, we observe the average number of hops each control packet traverses, and normalize the total number of hops traversed by all control packets by the entire tree size. 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 topology information, and compare the number of hops traversed by all control packets.

### 4.2 Simulation results

#### 4.2.1 Efficiency of bandwidth utilization

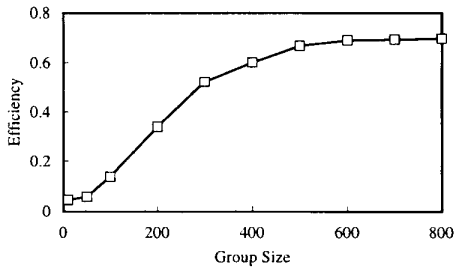


**Figure 2. Efficiency of unidirectional shared trees**

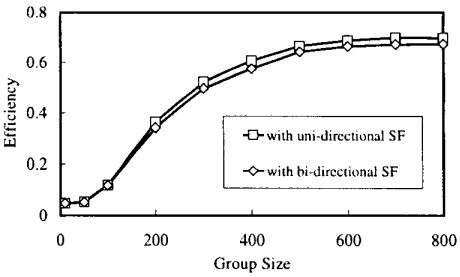


**Figure 3. Efficiency of bi-directional shared trees**

Fig. 2 shows the bandwidth utilization of unidirectional shared trees, with and without SF. We measure the ratio of hop count traversed by each multicast packet in the approach with SF to that without SF, varying group size (i.e., the number of hosts in the group) from 10 to 800. As group size increases, the number of hosts

wishing to receive a packet increases, and thus 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 much less bandwidth, thanks to forwarding packets toward effective outgoing interfaces only. When the group size is less than 100, the approach with SF consumes 20% less bandwidth than that without SF. In other words, more than 80% of the data packets transmitted in shared trees can be eliminated when using the proposed approach.

Fig. 3 depicts the bandwidth utilization of bi-directional shared trees, with and without SF. As in the case of unidirectional shared trees, the approach with SF uses bandwidth more efficiently as compared to the one without SF. We also measure the utilization of the bi-directional shared tree but using the SF mechanism for unidirectional shared trees, which forwards SF states only toward upstream interfaces. The result shows that compared to the unidirectional one, the bi-directional approach is more efficient in bandwidth utilization, allowing SF states to be delivered bi-directionally to avoid bandwidth wastage.

### 4.2.2 Scalability

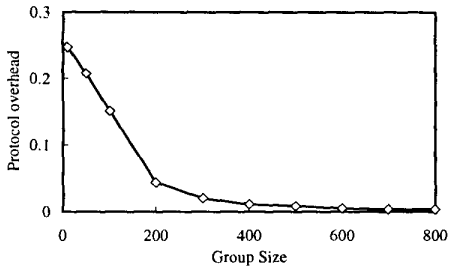


**Figure 4. Protocol overhead**

Fig. 4 depicts the protocol overhead of shared trees with SF. We measure the ratio of the average number of hops traversed by each control packet to the entire tree size, varying group size from 10 to 800. The average number of hops traversed by each control packet decreases as group size increases. This is because the source range covered by an SF state in a router increases as group size increases, rendering a higher probability to suppress the newly received control messages. As group size exceeds 200, the control overhead becomes very small. It proves that the computational and communication overhead generated by SF messages is limited to a few hops, instead of expanding to the entire multicast tree.

When group size is small, say less than 50, control packets may need to traverse across a large part of the tree (up to 25% as shown in Fig. 4). However, the approach with SF has a significant bandwidth advantage over that without SF when the group size is small, say less than 50, as shown in Fig. 3 (up to 5%). In short, the approach with SF can save 90% of bandwidth spent on transmitting data packets at the expense of 20% of bandwidth spent on exchanging control packets. Normally control packets are much smaller than data packets. Thus, the approach with SF performs better than that without SF. Note that we have also simulated the scenario with the bi-directional shared tree, and the results are consistent with the unidirectional one shown in Fig. 4. To conserve space, only the results of unidirectional shared trees are included.

Fig. 5 compares the scalability of the forwarding table size of a source-based tree to a shared tree with the proposed SF mechanism. We average the table size of all corresponding routers in 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 senders be proportional to group size, and allow each host to freely determine packet sources. The results show that when 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, even though each router is required to store one SF state per interface, the probability of efficiently suppressing SF states increases as the number of sources increases. Thus, it allows a much smaller table size as compared to the source-based one.

Fig. 6 compares the protocol overhead, considering the cases with and without topology information in the bi-directional shared tree. 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 we can save up to 30% bandwidth when using the topology information. The advantage of using the topology information diminishes when group size becomes large because the redundant messages forwarded are more likely to be merged.

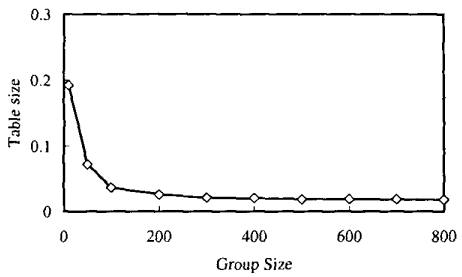


**Figure 5. Table size**

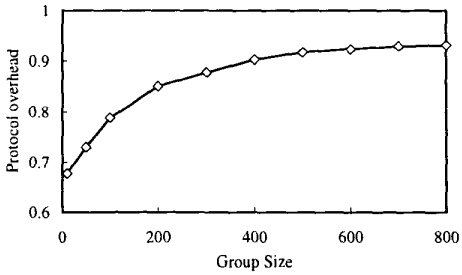


**Figure 6. Protocol overhead**

## 6. Conclusion

In this paper, we have proposed a new mechanism to support the capability of source filtering in IP multicast routing protocols. We have also conducted computer simulations to evaluate the performance of the proposed mechanism. The results demonstrate that, as compared to multicasting without the capability of source filtering, our mechanism allows better bandwidth utilization and scalability in terms of control overhead and forwarding table size, thus achieving a truly efficient use of resources for IP multicasting.

## References

[1] V. P. Kompella, J. C. Pasquale, and G. C. Polyzos, "Multicast Routing for Multimedia Communication," *IEEE/ACM Trans. on Networking*, Vol. 1, No. 3, pp. 286-292, June 1993.

[2] S. Deering, "Host extensions for IP multicasting," *IETF RFC 1112*, Aug. 1989.

[3] W. Fenner, "Internet Group Management Protocol, Version 2," *IETF RFC 2236*, Nov. 1997.

[4] B. Cain, S. Deering, B. Fenner and A. Thyagarajan. "Internet Group Management Protocol, Version3," *IETF Internet draft, draft-ietf-idmr-igmp-v3-07*, Mar. 2001.

[5] W. Liao and D. N. Yang, "Receiver-initiated Group Membership Protocol (RGMP): a New Group Management Protocol for IP Multicasting," *IEEE ICNP '99*, Nov. 1999.

[6] S. Deering, C. Partrige, and D. Waitzman, "Distance Vector Multicast Routing Protocol," *IETF RFC 1075*, Nov. 1988.

[7] T. Pusateri, "Distance Vector Multicast Routing Protocol," *IETF Internet Draft*, draft-ietf-idmr-dvmrp-v3-10, August 2000.

[8] J. Moy, "Multicast Routing Extensions for OSPF," *Communication of the ACM*, vol. 37, no. 8, pp 61-66, Aug. 1994.

[9] A. Ballardie, J. Crowcroft, and P. Francis, "Core Based Tree (CBT) – An Architecture for Scalable Inter-Domain Routing Protocol," *ACM SIGCOMM*, Sept. 1995, pp. 85-95.

[10] S. Deering et al., "An Architecture for Wide-Area Multicast-Routing," *ACM SIGCOMM '94*, pp. 126-135, Oct. 1994.

[11] S. Deering et al., "Protocl Independent Multicast Version 2 Dense Mode Specification," *IETF Internet Draft*, draft-ietf-pim-v2-dm, Nov. 1998.

[12] M. Handley, I. Kouvelas, T. Speakman, L. Vicisano, "Bi-directional Protocol Independent Multicast (BIDIR-PIM)," *IETF Internet Draft*, draft-farinacci-bidir-pim-02, Mar. 2001.

[13] T. Billhartz, J. B. Cain, E. Farrey-Goudreau, D. Fieg, and S. G. Batsell, "Performance and Resource Cost Comparisons for the CBT and PIM Multicast Routing Protocols," *IEEE Journal on Selected Area on Communications*, Vol. 15, No. 3, Apr. 1997.

[14] L. Wei and D. Estrin, "Multicast Routing in Dense Mode and Sparse Modes: Simulation Study of Tradeoffs and Dynamics," *IEEE International Conference on Computer Communications and Networks*, 1995, pp. 150-157.

[15] M. Sola, M. Ohta, T. Maeno, "Scalability of Internet Multicast Protocols", *Proceedings of INET'98*, http://www.isoc.org/inet98/proceedings//6d/6d_3.htm, July 1998.

[16] M. Doar and I. Leslie, "How bad is naïve multicasting routing?" *IEEE INFOCOMM '93*, pp. 82-89, 1993