Towards Dynamic Sender Access Control for Bi-directional Multicast Trees

Ning Wang & George Pavlou
Center for Communication Systems Research
University of Surrey
Guildford GU2 7XH
United Kingdom
{N.Wang, G.Pavlou}@eim.surrey.ac.uk

Abstract-**Bi-directional shared tree is an efficient routing scheme for many-to-many multicast applications (e.g. multi-party videoconferencing, interactive distance lecturing and Internet games etc). Given the open-group IP multicast service model, it is important to perform sender access control so as to prevent group members from receiving irrelevant data, and also protect the multicast tree from various Denial-of-Service (*DoS*) attacks. In comparison to source based and uni-directional shared trees where the data source can be authorized or authenticated at the single root or rendezvous point, in bi-directional routing this is a much more difficult problem since hosts can send data to all group members directly from any point in the tree. In this paper we propose a dynamic sender access control mechanism for bi-directional multicast trees so that irrelevant data is policed and discarded as it reaches any on-tree router. We show through simulation that the overhead of our mechanism is relatively small in terms of required state information in routers so that the proposed approach scales well for large groups.**

1. INTRODUCTION

IP multicast supports efficient communication services for applications in which an information source sends data to a selected group of receivers simultaneously. Although some IP multicast applications have been available on the experimental Multicast Backbone (*MBone*) for several years, large-scale development has not yet been achieved. The key problem is that group management is not strong enough to control both senders and receivers due to the adopted "open group" strategy, i.e., any information source can send data to any group member at any time. *IGMPv2* [8] is used to manage group members when they join or leave the multicast session but there are no control mechanisms for group members to avoid receiving data from particular information sources or prevent particular hosts from receiving sensitive information.

Since the model of IP multicast does not provide efficient authorization and authentication for group members and senders, protecting multicast data and end users and guaranteeing service availability has become an important challenge. Numerous research has focused on how to protect multicast data from being accessed by unauthorized hosts [2, 5, 12, 14, 16]. Most of these solutions are based on the application level encryption/decryption, hence efficient key distribution and management becomes an important issue. However, application level mechanisms cannot protect the routing infrastructure against *Denial-of-Service* (*DoS*) attacks such as flooding. This serious problem is magnified in multicast applications because data can be duplicated in multicast-capable routers on their way to receivers. The corresponding mechanism to prevent this should be deployed in the network rather than application layer. Relevant research efforts on network layer are far less than those on application layer. A key research work in the former area is [14]. Realizing that many multicast applications are based on one-to-many communication such as Internet TV/radio, data distribution etc., Holbrook et al proposed the *EXPRESS* routing scheme [10], from which the Single Source Multicast (*SSM*) is subsequently evolved. Although *SSM* is not a security multicast architecture, it provides significantly improved group management in comparison to the IP multicast service model. A single multicast tree is built rooted at the well-known source for delivering data to all interested subscribers. Under such a scenario, centralized group authorization and authentication can be achieved at the root of the source-based tree for client/server based multicast applications. Currently *IGMPv3* [4] is under the development to support source specific joins in *SSM*.

On the other hand, it should be noted that there exist many other interactive applications based on many-to-many multicast, such as multi-party videoconferencing system, distance lecturing and Internet games etc. For this type of applications, bi-directional multicast trees such as Core Based Tree (*CBT*) [1], Simple Multicast [13] and Bi-directional *PIM* (still under development) [9], are ideal routing schemes for efficiently delivering data flows between multiple hosts. However, since there is no single point for centralized group membership control, sender authentication and authorization becomes a new challenge. Typically, if a malicious host wants to perform denial-of-service attack, it can flood bogus data from any point of the bi-directional multicast tree. This problem of sender access control in bi-directional trees was extensively discussed in the *IETF* 43rd meeting [3]. One possible proposed solution to this problem is to periodically "push" the entire sender access list to all the on-tree routers. This policy does not scale well especially when many multicast groups or groups with many senders are involved. In this paper we propose an efficient and scalable sender access control mechanism for bi-directional trees so that data from unauthorized hosts is discarded once it reaches any on-tree router. In this way not only group members never receive irrelevant data, but also the multicast tree is immune to flooding attacks.

2. SENDER ACCESS CONTROL OVERVIEW

Compared with uni-directional shared trees such as *PIM-SM* [6] in which source filtering can be performed at the Rendezvous Point (*RP*) where the registrations of all the senders are processed and authorized, in bi-directional trees this is much more difficult since data from any source will be directly forwarded to the whole tree once it hits the first on-tree router. As already mentioned, the simplest solution for this is to periodically broadcast the entire access control list down to all the routers on the bi-directional tree for deciding whether or not to accept the data e.g. [13] - we name this mechanism *Full Policy Maintenance* (*FPM*). However, this method is only feasible when a few small-sized groups with limited number of senders are considered. For large-scale multicast applications for which this solution is not feasible, three questions need to be answered [3]: (1) how to efficiently distribute the list where necessary; (2) how to find "edge" routers that act as the trust boundary; and (3) how to avoid constant lookups for new sources.

It should be noted that most interactive multicast applications are highly dynamic in nature, with frequent join/leaving of information sources and group members, including those that are also capable of sending data. Hence the corresponding control policy should also be dynamically managed. Here we propose an efficient sender-initiated distribution mechanism of the access list during the phase of multicast tree construction. Compared with *FPM*, The key idea of our approach is that each on-tree router only adds its *downstream* senders to the local Sender Access Control List (*SACL*) during their join procedure, and the senders in the access list are activated when receiving the notification from the core. In fact only the core has the right to decide whether or not to accept the sources and it also maintains the entire *SACL* for all the currently authorized senders. Packets coming from any unauthorized host (even if it has already been on the tree) will be discarded immediately as they arrive at any on-tree router. To achieve this, all senders must first register with the core before they can send data to the group. As their registration packet hits an on-tree router, the unicast address of the sender is added to the *SACL* of each router on the way. In this scenario, the access policy for a particular sender is actually deployed on the branch from the first on-tree router where the registration is received along to the core router itself. Here we define the interface from which this registration packet is received as the *downstream interface* and the one used to deliver unicast data to the core as the *upstream interface*. The format of each *SACL* entry is (*G*, *S*, *I*) where *G* indicates the multicast group address, *S* identifies the sender and *I* is the downstream interface from which the corresponding registration packet was received. If the core has approved the join, it will send a type of "activating packet" back to the source, and once each on-tree router receives this packet, it will activate the source in its own *SACL* so that the new source will be able to send data onto the bi-directional tree from then on. Source authentication entries kept in each *SACL* are maintained in soft state for flexibility purpose, and this requires that information sources should periodically send refreshing packets up to the core to keep their states alive in the upstream routers. This action is especially necessary when a source is temporarily not sending group data. Once data packets have been received from a particular registered sender, the on-tree router may assume that this source is still alive and will automatically refresh the state for it. The difference between our dynamic access control scheme and *FPM* is illustrated in Figure 1, parts (a) and (b) respectively. It may be noticed that only the on-tree routers (in gray color) having received the sending request from the new source *h* need to maintain the policy for *h*. This is more scalable compared with the mechanism in which all on-tree routers keep the entire sender list. However, this requires that the sender should send data to the bi-directional tree *only* from the designated ingress router (router *A* in Figure 1). If any link between router *A* and the core *C* fails, the corresponding authentication entry will time out and become obsolete, hence host *h* has to find alternative path to perform re-registration so as to continue sending data to group members.

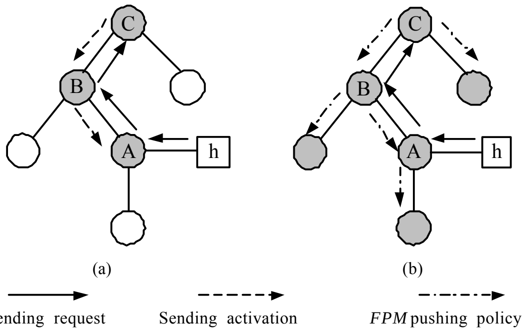

Fig. 1 Sender access control policy comparison

When a router receives a data packet from one of its downstream interfaces, it will first check if there exists such an entry for the data source in its local *SACL* list. If the router cannot find a matching entry that contains the unicast address of the source, the data packet is discarded. Otherwise if the corresponding entry has been found, the router will continue to verify if this packet comes from the same interface as the one recorded in the *SACL* entry. Only if the data packet has passed these two mechanisms of authentication, it will be forwarded to the upstream interface and other interfaces with the group state, i.e., interfaces where receivers are attached. On the other hand, when a data packet comes from the upstream interface, the router will always forward it to all the other interfaces with group state and need not perform any authentication. Although the router cannot judge if this data packet is from a registered sender, since it comes from the upstream router, there exist only two possibilities: either the upstream router has the *SACL* entry for the data source or the upstream router has received the packet from its own parent router in the tree. The extreme case is that none of the intermediate ancestral routers have such an entry and then we have to backtrack to the core. Since the core has recorded entries for all the registered senders and it never forwards any unauthenticated packet on its downstream interfaces, we can safely conclude that each on-tree router can trust its parent, and hence packets received from the upstream interface are always from valid senders. However, this scenario ignores the case of routers attached to multi-access networks which needs special attention; we will discuss the corresponding operations in section 3.3.

3. Dynamic Policy Maintenance

3.1 SACL Construction and Activation

As we have mentioned, all the data sources must register with the core before they can send any message onto the bi-directional tree. For each on-tree router, its *SACL* is updated when the registration packet of a new sender is received, and the individual entry is activated when its corresponding activating notification is received from the core.

If a host wants to both send and receive data, it must join the multicast group and become a Send-Receive capable member (SR-member, *SRM*). Otherwise if the host only wants to send data to the group (i.e. without receiving any), it may choose to act as a Send-Only member (SO-member, *SOM*) or a Non-Member Sender (*NMS*). In the former case, the host must join the bi-directional tree to directly send the data, and its Designated Router (*DR*) will forward the packets on the upstream interface as well as other interfaces with the group state. If the host does not want to join the tree and chooses to act as a non-member sender, it must encapsulate the data and unicast it towards the core. Once the packet hits the first on-tree router and passes the corresponding authentication, it is forwarded to all the other interfaces with group state.

(1) SR-member join

When the Designated Router (*DR*) receives a group *G* membership report from a SR-member *S* on the *LAN*, it will send a join request towards the core. Here we note that group membership report cannot be suppressed by the *DR* if it is submitted by a send-capable member. Once a router receives this join-request packet from one of its interfaces, say *A*, the (*G, S, A*) entry is added to its *SACL*. Then in a similar fashion to other bi-directional tree protocols, if the router has not been on the shared tree, a (*, *G*) state is created with the interface leading to the core as the upstream interface and *A* is set to the downstream interface. At the same time, interface *A* is also added to the interface list with group state so that data from other sources can be forwarded to *S* via *A*. If the router has already been on the tree, but *A* is not in the interface list with group state, then it is added to the list. Thereafter, the router just forwards the join-request towards the core via its upstream interface. Once the router receives the activating notification from the core, the (*G, S, A*) entry is activated so that *S* is able to send data.

(2) SO-member join

Similar to SR-member joins, the *DR* of a SO-member also sends a join-request up to the core and when the router receives this request from its interface *A*, the (*G, S, A*) entry is added to the local *SACL*. If the router is not yet on the tree, (*, *G*) state will be created but interface *A* is not added to the interface list with group state. This is because *A* does not need to forward data to a send-only member.

(3) Non-Member Sender (NMS) registration

Here we use the terminology "registration" instead of "join request", since this host is not a group member and need not be on the tree to send group data. The registration packet for an *NMS* is unicast towards the core and when it hits the first router with (*, *G*) state, (*G, S, A*) will be created in the local *SACL* of all the on tree routers on the way to the core. Though obvious, it should be noted that if a router is not on the tree, it does not maintain *SACL* for the group.

Finally, if a receive-only member (also known as a *group member* in conventional multicast routing) wants to join the group, the join request only creates (*, *G*) state if the router is not on the tree, but no new *SACL* entries are created.

The forwarding behavior of an on-tree router under send access control mechanism is as follows: If group data comes from downstream interfaces, the router will authenticate the information source by looking up the local *SACL* and if the sender has its entry in the list and comes from the right interface, the data packet is forwarded to the upstream interface and other interfaces with group state. If the corresponding *SACL* check fails, the data is discarded at once. On the other hand, if the data packets comes from the upstream interface, it is forwarded to all the other interfaces with group state because the router's parent is always trusted by its children.

3.2 An Example

A simple network model is shown in Figure 2. We assume that node *A* is the core router and all the Designated Routers of potential members of group *G* should send join request to this node. Hosts *H1-H5* are attached to the individual routers as depicted below.

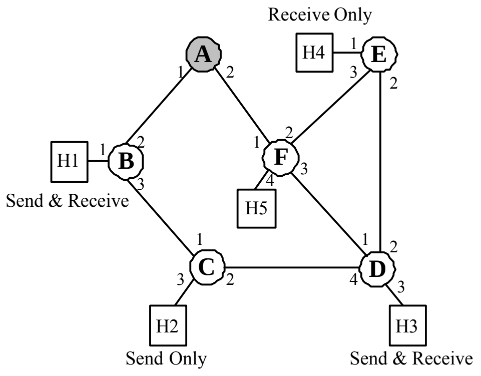

Fig. 2 Network model

Initially when *H1* wants to join the group, its *DR* (router *B*) will create (**, G*) state and send the join request to the core *A*. Since *H1* is a SR-member that can both send and receive data to/from the group, each of the routers that the join request has passed will add this sender into its local *SACL*. Hence both router *B* and *A* have the *SACL* entry (*G, H1, 1*), since they both receive the join request from interface 1. Host *H2* only wants to send data to group *G*, so it may choose to join as a SO-member or just act as a *NMS*. In the first case, its *DR* (router *C*) will create (**, G*) state indicating that this router is an on-tree node and then add *H2* to its *SACL*. Thereafter, router *C* will send a join request indicating *H2* is a SO-member towards the core, when *B* receives this request, it will also add *H2* to its local *SACL* and then forward the join-request packet to *A*. Since *H2* does not want to receive the data from the group, link *CB* becomes a send-only branch. To achieve this, router *B* will not add *B3* to the interface list with group state. If *H2* chooses to act as the Non-Member Sender, router *C* will not create (**, G*) state or *SACL* for the group but send a registration packet towards *A*. As this packet hits an on-tree router, say, *B* in our example, *H2* will be added to the local *SACL* of all the routers on the way. When sending group data, router *C* just encapsulates the data destined to the core *A*. When the data reaches *B* and passes the *SACL* authentication, it will be forwarded to interfaces *B1* and *B2* in order to reach *H1* and the core respectively. After *H3* and *H4* join the group, the resulting shared tree is shown in Figure 3, with the *SACLs* of each on-tree router also shown. It should be noted that *H4* is a receive-only member, and hence Routers *E, F* and *A* do not need to add it to their local *SACLs*. Suppose router *F* has received group data from *H3* on interface *F3*, it will check in its local *SACL* whether *H3* is an authorized sender. When the data passes the address and interface authentications, it is forwarded to both interfaces *F1* and *F2*. When group data is received on the upstream interface *F1*, since its parent *A* is a trusted router (in fact the data source should be either *H1* or *H2*), the data is forwarded to *F2* and *F3* immediately without any authentication. However, if the non-registered host *H5* wants to send messages to the group, data won't be forwarded onto the bi-directional tree due to the *SACL* authentication failure at router *F*.

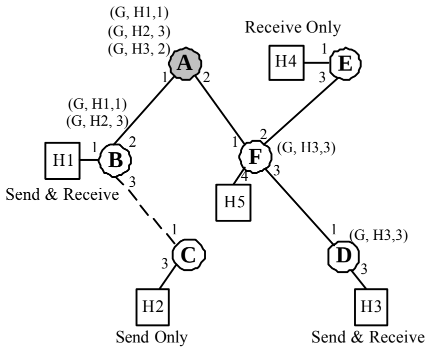

Fig. 3 Bi-directional tree with *SACL*

3.3 Operations on Multi-access Networks

We need special considerations for protecting group members from unauthorized sources attached to multi-access networks such as *LANs*. As we have mentioned, if an on-tree router receives packets from its upstream interface, it will always forward them to all the other interfaces with group state, since these packets have been assumed to come from an authorized information source. However this may not be the case if the upstream interface of an on-tree router is attached to a broadcast network. When an unauthorized host on the multi-access *LAN* wants to send data to the multicast group, a mechanism must be provided to prevent this data from being delivered to all the downstream group members. To achieve this, once the Designated Router (*DR*) on the *LAN* receives an "unauthenticated" packet from its downstream interface, i.e. it cannot find a matching access entry for the data source in its *SACL*, it will discard the packet, and at the same time will broadcast a "forbidding" packet containing the unicast address of the unauthorized host to the *LAN* from its downstream interface. Once the downstream router receives this packet on its upstream interface, it will stop forwarding the data with this unicast address that originates from an unregistered host attached to the *LAN*. Hence all the downstream session members will only receive little amount of useless data for a very short period of time.

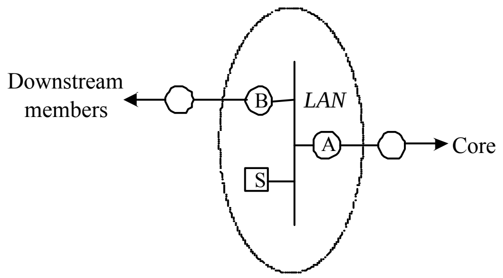

Fig. 4 Access control operation on *LANs*

In Fig. 4, suppose that the unauthorized host *S* sends data to the group. When the *DR* (router *A*) cannot find the corresponding entry in its local *SACL*, it immediately discards the packet and then sends a "forbidding" packet containing the address of *S* onto the *LAN*. Once the downstream router *B* receives the forbidding packet, it will stop forwarding data coming from host *S*.

4. *SACL* SCALABILITY ANALYSIS

In this section we discuss scalability issues regarding router memory consumption. It is obvious that the *maximum* memory space needed in maintaining *SACL* is *O(ks)* where *k* is the number of multicast groups and *s* is the number of senders in the group. Typically this is exactly the size of *SACL* in the core router. However, since on-tree routers need not keep the access policy for all sources but only for

downstream senders, the average size of *SACL* in each on-tree router is significantly smaller.
We can regard the bi-directional shared tree as a hierarchical structure with the core at the top level, i.e., level 0. Since each of the on-tree routers adds its downstream senders to its local *SACL*, then the *SACL* size *S* of router *i* in the shared tree *T* can be expressed as follows:

$$S(i) = \sum_{(i,j)\in T} (S(j))$$

and the average *SACL* size per on-tree router is:

$$\overline{S} = \frac{\sum_{i=0}^{H}\sum_{j=1}^{L_i} S(j)}{\sum_{i=1}^{n} Y'_i}$$

where H is the number of hops from the farthest on-tree router (or maximum level) and L_i is the number of routers on level *i*, while

$$Y'_i = \begin{cases} 1 & \text{if router } i \text{ is included in the shared tree} \\ 0 & \text{otherwise} \end{cases}$$

To ensure that the scalability issues are fairly evaluated throughout our simulation, random graphs with low average degrees, which represent the topologies of common point-to-point networks, e.g., *NSFNET*, are constructed. Here we adopt the commonly used Waxman's random graph generation algorithm [15] that has been implemented in *GT-ITM* for constructing our network models.
First we study the relationship between average *SACL* size and total number of senders. In the simulation we generate a random network with 100 routers and 50 receivers. The core router in the network is also randomly selected. The number of senders varies from 10 to 50 in steps of 10 while the group size is fixed at 50.
Here we study three typical situations regarding the type of sending hosts:

(1) All senders are also receivers (*AM*);
(2) 50% senders are also receivers (*HM*);
(3) None of the senders are receivers (*NM*).

All send-only hosts choose to act as Non-Member Senders (*NMS*) without joining the bi-directional tree.

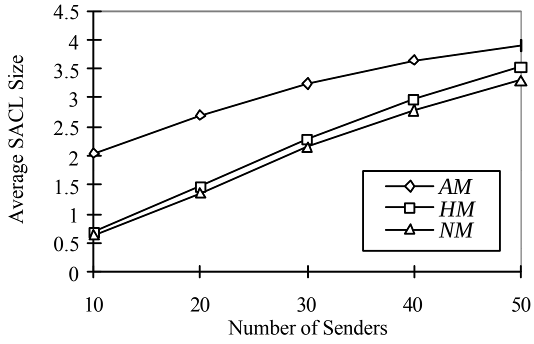

Fig. 5 *SACL* size *vs*. number of senders (I)

From Figure 5 we can see that the average *SACL* number of entries grows as the number of senders increases. However, it can be observed that even when the number of senders reaches the upper limit 50, the average *SACL* size is still quite small (less than 4 entries on average). This is in significant contrast with the "Full Policy Maintenance" (*FPM*) strategy. Further comparison between the two methods is presented later in Table 1. From Figure 5 we can also see that if all the senders are also receivers (case *AM*), this results in larger average *SACL* size. On the other end, if none of the senders is a receiver (case *NM*), the corresponding *SACL* size is smaller. This behavior is expected because given the fixed number of receivers on the bi-directional tree as well as the sender group, the larger the proportion of senders coming from the receiver set, the larger the average *SACL* size.

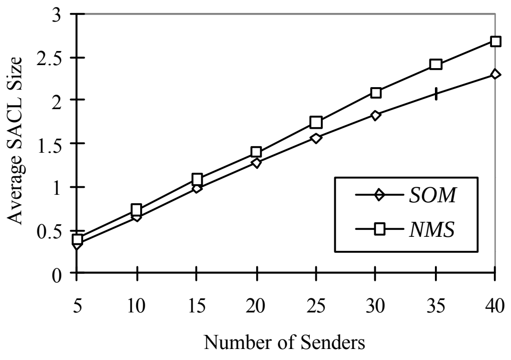

Fig. 6 *SACL* size *vs*. number of senders (II)

Next we study the effect on *SACL* size of the senders' choice acting as a Send-Only Member (*SOM*) or a Non-Member Sender (*NMS*). As we have mentioned, a host only wishing to send data to the group can decide to act as a *SOM* or *NMS*. Figure 6 illustrates the relationship between the *SACL* size and total number of senders. The group size is fixed at 50 and the number of senders varies from 5 to 40 in steps of 5. It should be noted that in this simulation all group members are receive-only hosts and do not send any data to the group. From the figure we can see that the *SACL* size also grows with the increase of the number of senders. Moreover, if all the hosts join the bi-directional tree and act as Send-Only Members (*SOM*), the average *SACL* size is relatively smaller. The reason for this is obvious: If the hosts choose to take the role of *SOM*, this will make the bi-directional tree expand for including the *DRs* of these senders. Since the number of on-tree routers grows while the total number of senders remains the same, the resulting average *SACL* size will become smaller. On the other hand, if all the hosts act as Non-Member-Senders, the figure of the shared tree will not change and no more on-tree routers are involved.
We continue to study the relationship between the average *SACL* size and the group size (number of receivers) with the number of senders fixed at 20. We still let these senders

choose to act as a *SOM* or *NMS* respectively. From Figure 7 we can see that the *SACL* size decreases with the growth of the group size in both cases. On the other hand, *SOM* join results in smaller average *SACL* size compared with *NMS*. The gap is more significant when there are fewer receivers. This is because if senders choose to act as *SOM*, they have to join the tree and generate many send-only branches, i.e., more routers are involved in the bi-directional tree. If the hosts just send data without becoming group members, the shared tree won't span to any of these senders, so the number of on-tree routers is independent of the number of senders. When the group size is small (e.g., 5 receivers), the size of the bi-directional tree will be significantly increased to include all the senders if they join as *SOMs*. This explains why the gap is more obvious when a small set of receivers is involved.

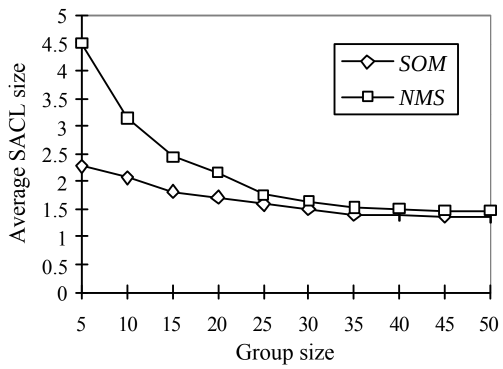

Fig. 7 Average *SACL* Size *vs*. Group Size

Finally we give the comparison between our approach and the "Full Policy Maintenance" (*FPM*) strategy regarding router's memory consumption. Table 1 gives the relationship of *SACL* size and total number of senders (*S*). From the table we can see that the length of the access list in each on-tree router using the *FPM* mechanism is exactly the number of active senders. This imposes a large overhead on routers in comparison to our scheme.

S	10	20	30	40
FPM	10	20	30	40
SOM	0.65	1.27	1.82	2.3
NMS	0.73	1.4	2.09	2.73

Table 1 Comparison to *FPM*

5. SUMMARY

In this paper we proposed an efficient mechanism of sender access control for bi-directional multicast trees. Each on-tree router dynamically maintains access policy for its downstream senders. Under such type of control, data packets from unauthorized hosts are discarded once they hit any on-tree router. Group members won't receive any irrelevant data, and service availability will be guaranteed since the multicast tree will be protected from denial-of-service attacks such as data flooding. Simulation results show that the overhead of our scheme is lightweight in terms of required state information in routers so that good scalability can be achieved. Our proposed mechanism constitutes in fact a control protocol that can be used in parallel to existing bi-directional multicast routing protocols such as *CBT*, Simple Multicast and bi-directional *PIM*. In the future we plan to focus on hierarchical sender access control mechanisms regarding inter-domain wide multicast applications with large sender and receiver group size.

REFERENCES

[1] T. Ballardie, P. Francis, J. Crowcroft, "*Core Based Trees (CBT): An Architecture for Scalable Multicast routing*", Proc. *ACM SIGCOMM' 93*, pp85-95

[2] A. Ballardie, "*Scalable Multicast Key Distribution*", *RFC* 1949, May 1996

[3] B. Cain, "*Source Access Control for Bidirectional trees*", 43rd *IETF* meeting, December, 1998

[4] B. Cain *et al*, "*Internet Group Management Protocol, Version 3*", Internet draft, draft -ietf-idmr-igmp-v3-*.txt, Feb. 1999, work in progress

[5] R. Cantti *et al*, "*Multicast Security: A Taxonomy and Some Efficient Constructions*", proc. *IEEE INFOCOM' 99*

[6] S. Deering *et al*, "*The PIM Architecture for Wide-Area Multicast Routing*", *IEEE/ACM* Transactions on Networking, Vol. 4, No. 2, Apr. 1996, pp 153-162

[7] C. Diot *et al*, "*Deployment Issues for the IP Multicast Service and Architecture*", *IEEE* Network, Jan./Feb. 2000, pp 78-88

[8] W. Fenner, "*Internet Group management Protocol, version 2*", *RFC* 2236, Nov. 1997

[9] M. Handley *et al*, "*Bi-directional Protocol Independent Multicast (BIDIR-PIM)*", Internet Draft, draft -ietf-pim-bidir-*.txt, Nov. 2000, work in progress

[10] H. W. Holbrook, D. R. Cheriton, "*IP Multicast Channels: EXPRESS Support for Large-scale Single-source Applications*", Proc. *ACM SIGCOMM' 99*

[11] S. Kummar *et al*, "*The MASC/BGMP Architecture for Inter-domain Multicast Routing*", Proc. *ACM SIGCOMM' 99*

[12] S. Mittra, "*Iolus: A Framework for Scalable Secure Multicasting*", proc. *ACM SIGCOMM*'97

[13] R. Perlman *et al*, "*Simple Multicast: A Design for Simple, Low-overhead Multicast*" Internet Draft, draft-perlman-simple-mulitcast-*.txt, Oct. 1999, work in progress

[14] C. Shields *et al*, "*KHIP-A Scalable Protocol for Secure Multicast Routing*", Proc. *ACM SIGCOMM' 99*

[15] B. M. Waxman, "*Routing of multipoint connections*", *IEEE JSAC* 6(9) 1988, pp 1617-1622

[16] C. Wong *et al*, "*Secure Group Communications Using Key Graphs*", Proc. *ACM SIGCOMM' 98*