A Novel Architecture for Secure and Scalable Multicast over IP Network

Yawen Wei, Zhen Yu, and Yong Guan

Department of Electrical and Computer Engineering,
Iowa State University, Ames IA 50011, USA
{weiyawen,yuzhen,yguan}@iastate.edu

Abstract. Currently, multicast services can be implemented at the IP layer or the application layer. While IP multicast violates the stateless paradigm of Internet and incurs great difficulties to congestion and flow control, application-layer multicast is lack of scalability due to the unreliability and resource constraints of end-hosts. Moreover, security is a main weakness in Internet-wide group communications. We propose in this paper a novel architecture for secure and scalable multicast in the Internet. In our architecture, a *Multicast Agent* in each Autonomous System (AS) is responsible for delivering multicast packets at the AS-level, relaying packets to end-hosts, and generating and updating keys to secure group communications. The proposed membership management protocol enables no-delay to membership updating; the proposed inter-domain routing protocol reduces the worst-case link stress by one magnitude compared to state-of-the-art protocols, and bounds the extra bandwidth cost within one percent compared to traditional IP multicast.

Keywords: IP Multicast, Routing protocol, Security, Inter-domain, Source-encoding.

1 Introduction

Multicast is an important and efficient mechanism to support many applications such as multimedia teleconferencing, news distribution, software updates and network games. Previous research efforts have been devoted to implementing multicast service either at the IP-layer or at the application-layer, however, the protocols implemented at both layers have drawbacks and have never been Internet-widely deployed.

In application-layer multicast protocols [5,6,7,8,11,13,14,16,20,22,27], end-hosts are organized into tree-based or mesh-based overlays to forward packets. Since end-hosts are limited in bandwidth and often experience abrupt crashes or failures, the overlay they form always suffers from large end-to-end latency and low data delivery rate. Meanwhile, since end-hosts need to periodically measure link quality to add/drop certain overlay links to improve tree or mesh topology, expensive operation overhead will be incurred, especially for large groups. Therefore, application-layer protocols cannot become a practical solution for large-scale group communications in the Internet.

Y. Chen et al. (Eds.): SecureComm 2009, LNICST 19, pp. 417–436, 2009.

We thus pass the hope on network-layer protocols [3,4,9,10,12,15,17,21,24]. However, IP multicast also has some limitations and has not been deployed through the Internet either. First, IP multicast requires routers to maintain per-group state and violates the stateless paradigm of the Internet. Second, it raises great difficulties in providing reliability, flow and congestion control at higher layers. Finally, IP multicast lacks a strategic business model and a security architecture. In the current open usage model, any host may send packets to an existing multicast group. Besides, flooding and DoS attacks will render multicast service unreliable or unavailable, and make the accounting for providing multicast service infeasible.

In this paper, we propose a secure and scalable architecture for Internet-wide multicast applications. In the proposed architecture, a border router called *Multicast Agent (MA)* exists in each Autonomous System (AS). These MAes are in charge of delivering multicast packets at the inter-domain level, relaying multicast packets to end-hosts at the intra-domain level, and generating and updating keys to secure group communications. The security are achieved in a hierarchical manner: the packets delivered between MAes are encrypted by a *global key*, and the packets delivered between end-hosts in a local domain is encrypted using a *local key*.

In the proposed architecture, we first design a membership and key management protocol. In our protocol, the membership information is explicitly distributed using *augmented-packets*, rather than using the traditional way that membership information are periodically exchanged between neighboring domains. By our design, not only bandwidth is saved from the exchanging traffic but also the propagation latency is reduced.

To achieve efficient inter-domain routing, we design an inter-domain routing protocol using the source-encoding technique. The MA at the source domain constructs and encodes dissemination tree information into each multicast packet. The benefits of such source-encoding are as follows: (1) The source domain knows all in-group domains, thus service fee can be properly charge by ISPs based on the scalability of the multicast group. (2) The privilege of receiving/sending packets is restricted only to in-group members, so a more secure usage model can be enforced. (3) No state information needs to be maintained at intermediate routers, i.e., the stateless nature of Internet is maintained. (4) Since the source can specify the targeted recipients of each packet, hence, subgroup communications can be conveniently supported. The last feature is especially beneficial in some applications where the participating members have heterogeneous interests. For example, in the Commercial Mobile Alert System (CMAS) [1], the text alerts related to disaster, immanent and child abductions are required to send to geographically targeted subgroups of people's cell phones.

In our inter-domain routing protocol, instead of requiring all tree information to be encoded into the packet header, we decompose in-tree nodes into two hierarchical levels, and *shim header* and *shim payload* of a packet encodes the two levels respectively. Such hierarchical design can effectively mitigate the packet duplication problem. It is proved by simulations that our protocol can achieve

good scaling, e.g., the number of duplicated packets is around twenty on the most stressful link.

The rest of this paper is organized as follows. Section 2 provides an overview of our proposed multicast architecture. Section 3 describes the group membership management protocol. Section 4 describes the key management issues. Section 5 proposes an inter-domain multicast routing protocol. Section 6 evaluates the proposed multicast architecture through simulations and compares it with state-of-the-art multicast protocols. We discuss the related work in Section 7 and conclude the paper in Section 8.

2 Overview

In our proposed multicast architecture, we assume a border router called *Multicast Agent (MA)* exists in each AS. The MAes are in charge of delivering multicast packets at the AS-level, relaying multicast packets to end-hosts that are interested in sending/receiving these packets, and generating and updating keys to secure group communications. All multicast traffic in and out of an AS will be handled by the responsible MA (we will discuss multiple MAes within a local domain in the Discussion Section 3.3).

At the source domain, when the MA receives a multicast packet from an end-host, it first constructs an AS-level dissemination tree then inserts the tree information into the multicast packet. The encoded tree information can lead downstream MAs to correctly forward the packet. When a border router receives the packet, it forwards it to the local MA. The MA checks if any end-hosts in its domain are interested about this packet. If yes, it multicasts the packet within the local domain; otherwise, it does not perform the intra-domain multicasting. Then, the MA decodes the tree information in the packet and forwards the packet to border routers in other domains. In our architecture, the security are achieved in a hierarchical manner: the packets delivered at the inter-domain level are encrypted using a *global key* shared by MAes, while the packets within a local domain are encrypted using a *local key* shared by end-hosts and the local MA.

In the following, we will describe the group membership management protocol in Section 3. We then discuss the key management issues in Section 4. In Section 5, we propose the inter-domain routing protocol which is used to construct, decompose and encode/decode AS-level dissemination trees.

3 Group Membership Management

3.1 Intra-domain Management

To multicast a packet to end-hosts in a local domain, the MA should know which end-hosts are group members. We do not specify any intra-domain multicast protocols used in a local AS, because most existing protocols can scale well at the domain level. Specifically, if PIM-SM or CBT protocol is used, then the MA constructs a unicast tunnel to the local Rendezvous Point (RP) from which

it knows the membership information; it also relays multicast packets through the tunnel to/from the RP. If PIM-DM or MOSPF protocol is used, then the MA can participate as an active member and perform like a normal end-host to multicast to the group.

3.2 Inter-domain Management

The inter-domain level group membership is managed by MAes. Since any in-group MA may become the source of a multicast packet and need to know destination domains to construct a dissemination tree, the membership information should be available to every in-group MA. A simple approach to achieve such group-wide awareness is maintaining membership information at a central server. However, this will lead to large query traffic towards the server and cause it overloaded or even out-of-service. Therefore, we suggest every in-group MA keeps a copy of the member list and collaboratively updates the list when membership changes.

In this subsection, by *membership* we refer to the membership at the domain-level. In other words, only when the number of end-hosts in an AS domain rises above zero or decreases to zero, the MA of this domain joins or leaves the multicast group and becomes an in-group or out-group MA.

Augmented Packet. We first introduce the *Augmented Packet* which is a basic technique used in our inter-domain membership management. The format of an augmented packet is shown in Fig. 1 where a normal packet is augmented with a *membership payload*. In this membership payload, a header contains 32 bits. The first 8 bits (denoted by n_j) indicate the number of newly-joined MAes; the second 8 bits (denoted by n_l) indicate the number of leaving MAes; the last 16 bits are the checksum computed over the entire membership payload to ensure its integrity. Followed the header are the IDs of joining MAes then the IDs of leaving MAes.

The reason for introducing such augmented packet is to avoid the extensive traffic caused by sending a separate updating message for every member join or leave, especially if we consider the fact that millions of multicast groups may exist in the Internet simultaneously.

Now the question is which MA should be in charge of appending the membership payload to its multicast packet? A reasonable answer is that the first MA

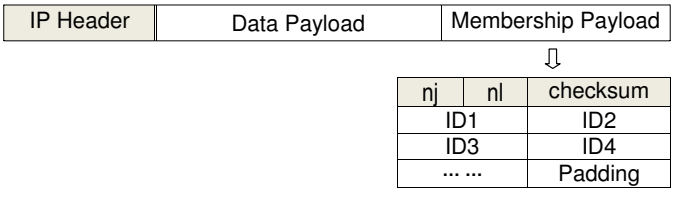


Fig. 1. Augmented packet format

that multicasts packet after membership changes should take the responsibility, because the delay in membership updating can be minimized. However, the next source MA is not directly available except for some application (for example, in IPTV, all data are originated from a source domain where the TV station resides). In our approach, we suggest MAes use self-learning algorithms to make predictions based on historical records on previous source domains. They can use those algorithms that are used to handle page replacements in virtual memory management, or they can predict the next source domain as the one that most recently or most frequently sends packets, or they can consider both frequency and recentness and apply aging algorithm for the prediction.

Membership Updates upon Member Joins. The detailed process for MA to join a multicast group is described in the following steps (Fig. 2).

- Step 0: If a MA (say, MA-1) wants to join a group, it should first contact the group registry server and get bootstrapped with a list of in-group MAes. (Here, we consider close usage model in which every member explicitly goes through registration process to obtain the privilege of sending/receiving packets from this group. Close usage model provides many benefits such as better control, traffic engineering and accounting.) The list obtained in bootstrap does not need to be complete, i.e., it may just contain several in-group MAes.
- Step 1: The new member MA-1 randomly selects a MA (say, MA-2) from its bootstrap list, and sends to it a join message.
- Step 2: MA-2 predicts the next source MA based on the record of source domains during a past period of time, and transmits the AS number of this MA (say, MA-3) together with a full list of all in-group MAes back to MA-1.
- Step 3: If MA-1 happens to have a multicast packet originated from its domain at this moment, it directly multicasts an augmented packet with its ID included in the membership payload, then goes to step 6. Otherwise, MA-1 will solicit help from MA-3 by sending to it a request, asking MA-3 to send an augmented-packet with MA-1's membership information.

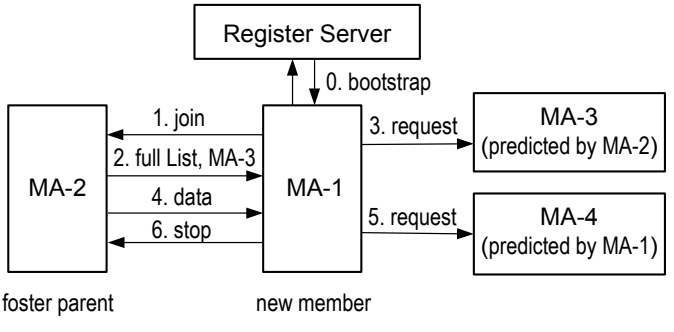


Fig. 2. A new MA joins a multicast group

- Step 4: Notice the next source prediction may not be accurate, i.e., MA-3 may not become the source during a period of time in the future. In this case, MA-1 will not receive any data for this group during this time since other in-group MAes are not aware of the existence of the new member. To mitigate this problem, we designate MA-2 as the *foster-parent* of MA-1. That is, when MA-2 receives any packet for this multicast group, it should relay the packet to MA-1 through a unicast tunnel between them.
- Step 5: If after a certain period of time t after joining, MA-1 has not received any packet containing its ID in the membership payload, it knows MA-3 has not sent out any augmented-packet yet. Then MA-1 will predict another MA (say, MA-4) based on its own historical record of the source domains, and send to it a request message. MA-1 repeats this step by periodically contacting different MAes until its membership can be group-widely notified. However, if more than a certain number of predications fail, MA-1 will multicast a separate message by itself to announce its membership.
- Step 6: In the end, MA-1 informs its foster-parent to remove it from the foster-children list.

Membership Updates upon Member Leaves. If an existing MA wants to leave a multicast group, it either multicasts an augmented-packet itself, or predicts the next source MA and informs it about its leaving. We notice that before a leaving MA is finally removed from the member list, the MA may continue receiving multicast packets, so it should perform as an in-group MA for a period of time to guarantee correct data delivery.

3.3 Discussion

In this subsection, we would like to discuss the consistency issue and multiple MAes issue associated with our inter-domain group membership management protocol.

Consistency. We notice that there may be some timing and delay in communicating MA joins/leaves at the inter-domain level. Consider a scenario where a users is channel surfing. The corresponding MA, MA-1, will first announce join then announce leave. Assume MA-1 contacts a predicted source MA, MA-2, and sends a join request to it. Later, MA-1 contacts another predicted source MA, MA-3, and announces its leave. Due to the possible timing/delay, some MAes may receive leave information by MA-3 prior to join information from MA-2, which may create a situation where not all MAes have a consistent view about MA-1's membership. Our fix to this problem is to use sequence numbers for each MA in sending out join/leave updates. Therefore, when in-group MAes receive conflicting group membership information, they can ignore the obsolete information and only keep the latest information to achieve consistency.

Multiple MAes. In our architecture, we assume one border router plays the role as the local MA. The selection can be based on which border router is the

exit point to most other ASes, or which has the smallest IP address in the local domain. However, the vulnerability and unreliability associated with a single MA is present. First, when the current MA crashes or reboots, another border router has to take over the responsibility and the handoff process will introduce some delay. Second, since all group traffic in/out of the domain is through a single MA, the traffic concentration problem is obvious given the inter-domain traffic is high in the backbone. We are considering to allow multiple MAes to share the workload of handling group traffic in a local domain. Some design challenges exist to guarantee a consistent view among multiple MAes of both the intra- and inter-domain membership, and achieve the smooth cooperations between them, and these will be our interested research subjects in the future.

4 Group Key Management

Given that security is one of the main weaknesses of IP multicast, the need to secure multicast packets is particularly apparent and crucial. Secure group communication systems mostly rely on a group key, which is a secret only known to group members and used to encrypt multicast messages. When group membership changes, a new group key should be established to guarantee forward security and backward security, that is, members who have left the group cannot decrypt messages in later sessions, and new members cannot decrypt messages in previous sessions. The challenging problem is to design key management schemes that can scale to large groups or groups with highly dynamic memberships. Previous key management schemes, including the key graph approach [25,26] and its extensions, require at least O(logN) computation and communication per rekeying operation, where N is the number of group members. In many Internet multicast applications such as the massive multiplayer games, the value of N, i.e. the number of participating players, can be several millions, which will make rekeying overhead particularly huge.

Instead of requiring all end-hosts use a single group key to secure their communications, in our architecture, we suggest to organize the end-hosts into a hierarchy that is consistent to the Internet topology. Namely, the end-hosts within an AS form a subgroup, and the domain's MA is the subgroup head. Within a local domain, multicast packets are encrypted using a *local key* shared by all in-group end-hosts and the local MA; at the inter-domain level, packets are encrypted using a *global key* shared by all in-group MAes. Therefore, if an end-host joins or leaves a group, only the local key needs to be changed, but the global key is maintained the same, which can greatly mitigate the scalability problem.

When a multicast packet enters/leaves a domain, decryption and re-encryption should be performed. Precisely, when a MA relays a multicast packet into its domain, it needs to decrypt the packet using the global key then re-encrypt it using the local key. To enhance the efficiency of the cryptographical operations, we can adopt techniques suggested by previous works such as [18]. Such that the local/global keys are not used to encrypt and decrypt a packet directly, instead, they are used to encrypt and decrypt a random *session key*, and the session key is

the real key that encrypts the packet. In this way, decrypting and re-encrypting a packet is reduced to decrypting and re-encrypting the session key.

In the following subsections, we discuss the key management at the intra- and inter-domain level in more detail.

4.1 Local Key Management

The MA serves as *key server* within each domain. The MA is responsible to distribute a local key for each group, and update the key whenever membership changes. We emphasize that when an end-host leaves or a new end-host joins the group, only the local key has to be updated, while the global key will remain the same. The MA shares a pair-wise secret key with every end-host. (1) When a new host joins, the MA generates a new local key, encrypts it using the shared key with the new host and unicasts to the host. Meanwhile, it encrypts the new local key using the old local key, and multicasts to previous hosts. (2) When an existing host leaves, the MA also generates a new local key, encrypts it using each of the shared keys with the remaining members, and multicasts one message containing all the encrypted keys to them. The computation overhead for member join and leave is O(1) and O(n) respectively, where n is the local group size. Although scalability may not be a severe issue within a local domain (compared to Internet-wide groups), we do not limit ourselves to adopt any key management algorithms that have less computation, communication or storage overhead.

4.2 Global Key Management

We assume each MA shares a pair-wise key with every other MA in the Internet. Since it is not reasonable to assume a single Internet-wide key server for all multicast groups, we require the global key for each group is maintained by in-group MAes. Precisely, for a group, one in-group MA is selected as the key server, and is responsible of distributing a new global key to the current in-group MAes, whenever a new MA joins or an existing MA leaves the group. We can adopt any re-keying algorithm for the key server to update the global key.

If the key server itself wishes to leave, it has the authority to designate another in-group MA to be the key server. The designation can be based on reliability, bandwidth, membership length and local group size. For instance, the MA with the largest number of participating end-hosts is less possible than other MAes to quit the group, thus it can be selected as the next key server. For a newly-joined MA whose membership have not been notified to other in-group MAes (thus not known to the key server either), the multicast packets relayed from its foster-parent to the new member can be encrypted and decrypted using their pair-wise key.

5 Inter-domain Multicast Protocol

In this section, we propose an efficient inter-domain multicast protocol. We first discuss how a dissemination tree is constructed, decomposed and encoded at

AS Number	AS Path	Next Router	Interface
4515	34225 41692 3491 4515	193.138.164.1	m0
6356	34225 1299 6830 22773 22318 6356	193.138.164.1	m1
... ...	... ...	... ...	... ...

Fig. 3. Routing table at MA34225

the source MA, and then we describe how the tree information is decoded and updated at downstream MAes.

5.1 Preliminary Work

Before we introduce our inter-domain routing protocol, we first take a look at a previous work related to our protocol, i.e., the Free Riding Multicast (FRM) [21] protocol. In FRM, the border router of a source AS computes a dissemination tree from the union of unicast paths, then puts the tree information into the fixed-size *shim header* of each multicast packet. If the tree is very large, then multiple packets have to be transmitted to carry the encoded tree information. The packet duplication problem can be very severe, e.g., it is reported that when the dissemination tree spans on all AS domains in the Internet, the worst-case physical link has to transmit about 150 duplicated packets.

Although two approximation methods were suggested to mitigate packet duplication problem associated with FRM, there are some practical issues with the suggested methods. (1) The first method is to omit customer ASes at tree leaves when encoding the dissemination tree. However, this requires the border router at the source AS to know customer-provider relationships between other ASes. Although some techniques [23] can help guess AS relationships by exploring the AS graph, the guess cannot be validated because the customer information is proprietary information of an ISP. Therefore, the guesses would be wrong and packets would not be able to be delivered to some valid in-group members at the leaves. (2) The second approximation method is to replace all the tree links connecting a node by one *aggregated link*, if the number of tree edges from the node is a large fraction of its total edges. It is claimed that an AS domain A forwards packets to its neighbor B only when B-X (X is a neighbor of B) lies on the path from A to some destination domain. However, this cannot ensure A-B is an in-tree link because the path containing B-X do not necessarily pass domain A. The consequence is that some packets will be sent on non-tree links.

In the following, we propose a novel inter-domain multicast protocol that can effectively mitigate packet duplications and achieve efficient data delivery.

5.2 Construction of Hierarchical Dissemination Tree

For a multicast group, the source MA can learn all in-group MAes from the membership management protocol we have described in Section 3. To construct a dissemination tree, the source MA looks up its *MA routing table* to find out

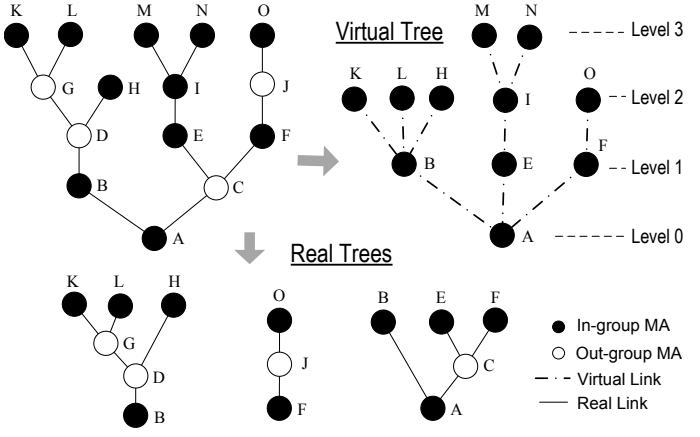


Fig. 4. Decomposition of the flat tree

the unicast path leading to each in-group domain (an example is shown in Fig. 3). The routing table is indexed by AS numbers and each entry provides the AS paths, next-hop, interface, and other path attribute information. This table can be easily constructed using BGP RIB and updated using BGP routing updates. If multiple policy-permitted paths exist leading to a same AS, then the best-quality path can be selected. For instance, the path with the smallest AS-hops or the shortest geographical distance [19] can be selected.

After consulting this table, the source MA can construct a flat tree by the union of unicast pathes leading to all destination domains. Then, it decomposes the flat tree into one *virtual tree* and multiple *real trees*. Fig. 4 presents an example of the decomposition. The virtual tree consists of *virtual links* connecting in-group MAes, and the real trees consist of *real links* connecting out-group MAes. The source MA puts the information about the virtual tree and the real tree rooted at itself into the packet. The real tree information can guide out-group MAes to forward the packet properly until it reaches in-group MAes at level 1 in the virtual tree. Then, each level-1 MA replaces the real tree in the packet with the one rooted at itself, which leads the packet to level-2 MAes, and so forth. Finally, the packet will traverse through the whole tree and visit every in-group MA.

5.3 Shim Header and Shim Payload

Now the problem is how the source MA encodes and attaches tree information to each multicast packet. Since real-tree information is used by out-group MAes and virtual-tree information is used by in-group MAes, we can encode them into *shim header* and *shim payload* of a packet, respectively (Fig. 5).

We adopt the technique of bloom filter to encode tree information into shim header and shim payload. Bloom filter is a space-efficient probabilistic data structure that can support membership queries. It uses k independent hash functions to map every member to k different positions in a m-bit vector. When using

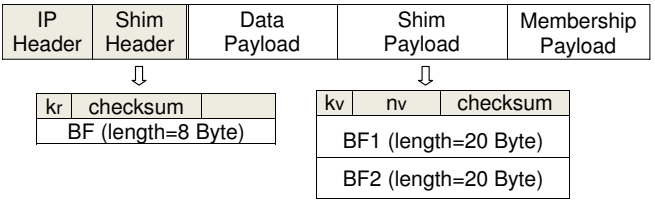


Fig. 5. Shim header and shim payload formats

bloom filter, false negative is guaranteed to be zero, but false positives is nonzero and will increase with the number of members hashed into the filter. Given the highest acceptable false positive, the maximum number of members one bloom filter can afford can be determined. We assume all MAes in the Internet share a same set of hash functions, therefore, an MA can use the set of functions to decode the links from filters constructed by other MAes.

Fig. 5 shows the format of shim header and shim payload. Shim header consists of 32 control bits and a 8-byte bloom filter to contain real tree links. The control bits include 4 bits (denoted by k_r) that indicate the number of hash functions, 16 bits checksum that are computed over the entire shim header to ensure the integrity, and the remaining 12 bits for future use. Shim payload consists of 32 control bits and multiple fixed-length bloom filters (20 bytes in our design) . The control bits include 4 bits (denoted by k_v) that indicate the number of hash functions, 12 bits (denoted by n_v) that indicate the number of bloom filters, and 16 bits of checksum that are computed over the entire shim payload to ensure its integrity.

5.4 Tree Encoding on Source MA

In this subsection, we give an example that illustrates the tree encoding process at a source MA. As we can see in Fig. 6, source node A sends packet P1 to an in-group node (node B) and packet P2 to an out-group node (node C). In packet P1, the shim header contains link A:B, which is both a real link and a virtual link, and the shim payload contains virtual links B:K, B:L and B:H. In packet P2, the shim header contains links A:C, C:E and C:F, and the shim payload contains two bloom filters corresponding to two subtrees, one with virtual link F:O and the other with virtual links E:I, I:M and I:N. In this example, the whole dissemination tree is first decomposed into one virtual tree and three real trees, then the virtual tree is decomposed into three sub virtual trees. Given the maximum number a bloom filter can afford, the decomposition of the virtual tree is to ensure no bloom filter contain more links than the threshold.

5.5 Tree Decoding and Updating on Transit MAes

We now discuss the checking and updating process at downstream MAes. If the MA is an out-group MA, it simply checks all its AS-neighbor-links to find out

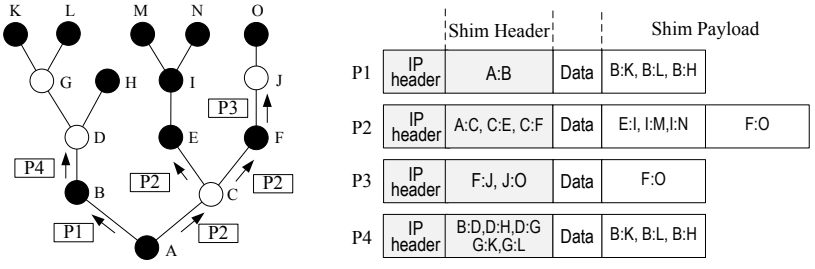


Fig. 6. Example of tree encoding, decoding and updating

the present ones in the shim header, then forwards the packet accordingly. No updating on the shim header or payload is needed. If the MA is an in-group MA, it first checks the shim payload and decodes the present virtual links, then it rewrites the shim head to encode the links of the real tree rooted at itself, and removes some bloom filters in the shim payload. As the example shown in Fig. 6, a transit MA node, node F, receives packet P2 and sends packet P3. It rewrites the shim head to contain links F:J and J:O, then it removes the second bloom filter in the shim payload, because none of the virtual links E:I, I:M and I:N in that filter will be of future use as the packet traverses deeper into the tree through node F.

When an in-group MA checks the bloom filters in the shim payload, it does not need to check the virtual links connecting itself and every other in-group MA, instead, it only needs to check those connecting itself and its children MAes in the virtual tree rooted at itself. We can prove the correctness of such checking by the following proposition.

Preposition 1. In a multicast group G, node i's child in a virtual tree rooted at node j ($j \neq i$) must be i's child in the virtual tree rooted at itself.

Proof. We prove by contradiction. We denote the virtual tree constructed for multicast group G and rooted at node t by T_t^G. Assume in the virtual tree T_j^G, there exists a child k of node i, such that k is not i's child in the virtual tree T_i^G, which means another node s must lie on the shortest path from node i to node k in T_i^G, denoted by L_{ik}. Since L_{ik} must be a part of L_{jk} in T_j^G (otherwise L_{jk} will not be the shortest), node s will also lie between node i and k in the path L_{jk} in T_j^G, which implies that node k is not a child of i in T_j^G. Contradicted.

5.6 Discussions

We now discuss some properties of our proposed inter-domain routing protocol.

IP Fragments. During the packet delivery process, IP fragmentation may take place at an intermediate router. Our protocol deals with such situations by doing the followings: first, both the IP header and the shim header from the original IP

datagram should be copied to new datagrams. Therefore, the shim header is per-packet based, which enables out-group MAes to solely look at a packet's shim header to forward it correctly. Second, the shim payload should be fragmented and inserted into multiple packets. Since in-group MAes are the destinations of all the fragmented packets, IP reassembly will be performed at every in-group MA and the shim payload will be recovered.

Bandwidth Consumption. The bandwidth consumption is minimized in our protocol by adopting two important techniques.

First, the hierarchical decomposition effectively alleviates packet duplications. In the basic source-encoding approaches, the entire flat tree is encoded into the shim header of a packet. Since the shim header has very limited length and cannot accommodate too many links, multiple shim headers have to be constructed, which directly causes duplicated packets. In our protocol, the virtual tree is inserted into a packet's payload which can contain up to 65KB data. Only a small real tree needs to be inserted into the shim header. Fig. 6 showcases the benefits. The shim header of packet P2 only contains three links A:C, C:E and C:F in our protocol; without tree decomposition, the shim header would have to contain eight links A:C, C:E, C:F, E:I, I:M, I:N, F:J and J:O.

Second, we encode virtual links into multiple bloom filters instead of a single large one, which further reduce bandwidth consumptions. As a packet traverse deeper into the dissemination tree, in-group MAes can continuously remove some "useless" filters that contain virtual links not present in its subtrees. Therefore, packet size can be reduced and bandwidth can be saved. This advantage can be seen clearly in Fig. 6, where node F removes the second bloom filter in the shim payload, since none of the virtual links E:I, I:M and I:N will be of future use.

Processing Delay. In our protocol, there are three procedures where delay may be introduced.

First, the processing delay can be introduced at the source MA by constructing shim header and shim payload. However, we notice a MA can pre-construct the dissemination tree and cache the shim header and payload information. For example, a MA will cache for groups for which it has a large number of end-host users, because it is very probable for it to become the source MA in the future.

Second, since an out-group/in-group MA should check the shim header/payload for present real/virtual links, there is a delay associated with the look-ups in bloom filters. Fortunately, bloom filters can be implemented using very efficient hardware like TCAM [28], such that the to-be-checked links can be hashed into multiple rows and accessed in parallel to achieve high efficiency.

Third, delay may be incurred at an in-group MA by the rewriting operation of the shim header of a packet. To mitigate this delay, a MA can pre-compute and cache the bloom filter containing the real links associated with each virtual link, then it can rapidly construct the shim header by simply XORing these filters. As an example, node B in Fig. 6 can cache three bloom filters: filter-1 contains B:D and D:H corresponding to the virtual link B:H, filter-2 contains B:D, D:G and

G:K corresponding to virtual link B:K, and filter-3 contains B:D, D:G and G:L corresponding to the virtual link B:L. After receiving packet P1, node B checks the presence of virtual link B:H, B:K and B:L. So it XORs filter-1, filter-2 and filter-3 and inserts the result into the shim header of packet P4. Currently, less than 30,000 AS domains exist in the Internet, which means the memory cost for caching the bloom filters associated with virtual links will be no more than 240KB at all MAes.

6 Simulation Result

In this section, we conduct simulations to evaluate the performance of the proposed multicast architecture. We mainly focus on the network cost of our inter-domain multicast protocol and compare it with other state-of-the-art protocols. The protocols we use to compare with our protocol include the followings. (1) IP multicast: the dissemination tree is composed of shortest reverse paths from the source AS to destination ASes. (2) Per-AS unicast: the source AS sends a separate unicast packet to each destination AS. (3) FRM: the dissemination tree is constructed by the union of unicast paths from the source AS to destination ASes, and the whole tree is encoded into shim header of every packet. (4) AS-level overlay: the dissemination tree is constructed using our proposed protocol, but the packets are unicast between different MAes.

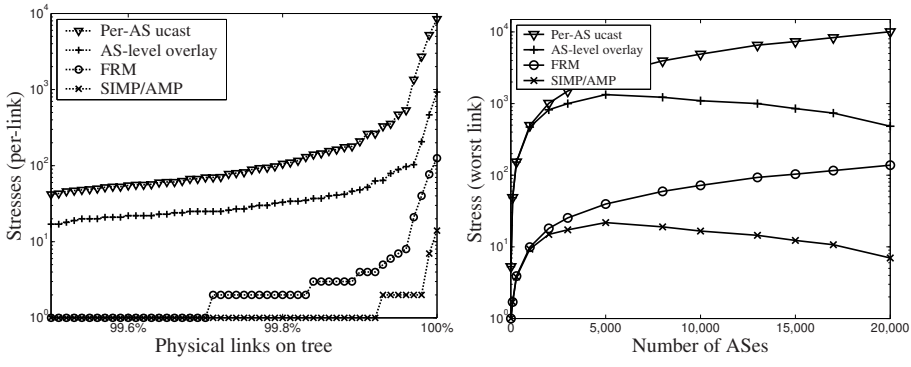
We use the following metrics to measure the network costs associated with different multicast protocols. (1) *Link stress* is defined as the number of *duplicate* packets transmitted on a physical link. By duplicate packets we mean the packets that have identical *application* payload, though they may have different protocol-related headers or payloads. Obviously, the stress on all physical links is one in IP multicast. (2) *Protocol overhead* is defined as the extra bandwidth consumed by the protocol-related data in the packets. In IP multicast and per-AS unicast, the protocol overhead is zero; in FRM, AS-level overlay and our protocol, the protocol overhead is not zero because the tree information is present in the packets' headers/payloads. (3) *Bandwidth cost*: This metric evaluates the total bandwidth consumption to multicast one packet to all receivers. Essentially, this metric reflects the combined impacts of the link stresses and protocol overheads.

Our simulations are conducted using real BGP data from RIS [2]. RIS is a RIPE NCC project that collects and stores routing data from the Internet. We download one day's files of BGP data collected by the Remote Route Collectors (RRCs) in MRT format. After removing the incomplete measurements and IPv6 paths, we select twenty IPv4 full tables as the basis of our experiments. The results are averaged over 200 runs using the 20 BGP tables with 10 runs per table. In our further work, we will implement our protocol on real border routers for better understanding of the protocol's behavior in dynamic real-world environments.

6.1 Link Stress

Fig. 7(a) compares the CDF of link stresses of our protocol and other multicast protocols in a typical run when the multicast group consists of 10,000 in-group domains. The per-AS unicast and AS-level overlay have very high link stresses, with the worst-case link stress reaches four and three orders of magnitude respectively. In FRM, about 99.6% links see one transmission, but the worst link stress is over one hundred. Since the worst case always happens on links between the root and its children domains in the dissemination tree, the congestion of these links will impact many downstream members. Our protocol effectively reduces the stresses on all physical links: more than 99.9% links see exactly one transmission, and the worst-case stress can be reduced to only 14.

Fig. 7(b) plots the worst link stress for our protocol and other multicast protocols for different group sizes. We select in-group AS domains randomly and increase group size from 10 domains to 20,000 domains. In per-AS unicast protocol, the worst stress increases linearly with the group size. In AS-level overlay, the worst link stress first increases then decreases to a few hundreds. This is because the duplicate transmissions are incurred by the unicast between overlay nodes. As the group size increases, an overlay node will have more children, resulting in more stresses on physical links leading to these children domains. However, when the group gets even larger, in-group ASes get closer and the unicast paths between them become shorter, thus, less links are shared between the unicast paths and the link stresses drop accordingly. In FRM, the worst link stress gradually increases with the group size, with the highest one around 100. In our protocol, the max worst link stress is only 21 and happens when group size is around 5,000. Since the duplicate packets are caused by multiple shim headers that encode real tree links, hence, the fact that the size of real trees first increases then decreases with the group sizes directly causes the same tendency on link stresses.



(a) Stresses on physical links (b) Worst stress vs. group size

Fig. 7. Link stress distribution and worst-case Links stress

6.2 Protocol Overhead

Fig. 8 plots the total protocol overhead involved in our protocol and FRM for different group sizes. Since this metric measures overhead due to shim headers and payloads, the overhead increases with the to-be-encoded tree size for both schemes. However, the growing speed is quite different. In FRM, the protocol overhead grows almost linearly with the group size, which is expected because the shim header encodes all tree links. In our protocol, the protocol overhead grows much more slowly, and when in-group domains is 20,000, a total of 0.5MB bandwidth is consumed. We attribute the conservation on protocol overhead when using our protocol by two major reasons: first, the tree decomposition hides the real tree links from virtual tree, resulting in less information to be encoded in shim headers; second, the shim payload updating at in-group domains enables further reducing of shim payloads.

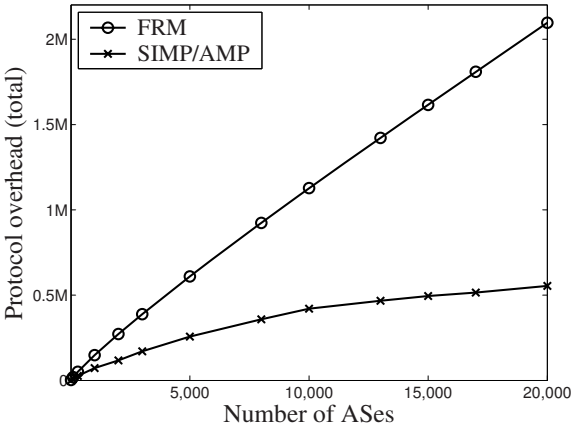


Fig. 8. Protocol overhead vs. group size

6.3 Bandwidth Cost

Fig. 9 shows the bandwidth cost against group sizes. We use 1 KB as the size of the data packet, and normalize the bandwidth costs of different multicast protocols with respect to IP multicast. We have repeated this study with different packet sizes and observed similar ratios for all protocols, which implies that the bandwidth consumption is largely due to packet duplications rather than the protocol overheads. This also explains the similar shapes of the curves in this figure compared to the curves in Fig. 7(b). The two upper curves correspond to per-AS unicast and AS-level overlay, which show that they introduce at least 100% and 10% more bandwidth cost respectively. The other two curves correspond to FRM and our protocol. We see our protocol performs the best again, and incurs extra bandwidth cost no more than 1% in the worst case compared to traditional IP multicast.

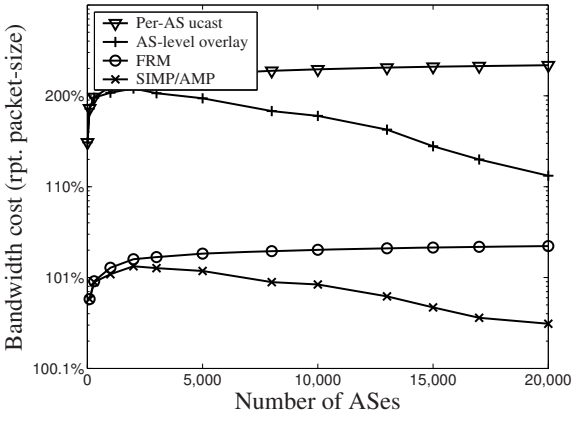


Fig. 9. Bandwidth cost vs. group size

7 Related Work

Numerous protocols have been proposed to provide multicast service at the network layer. Intra-domain multicast protocols include DVMRP [24], PIM-DM [3] and MOSPF [17], etc. Inter-domain multicast protocols include CBT [4], PIM-SM [10], BGMP [15], PIM-SSM [12], etc. In DVMRP/PIM-DM [24,3], the multicast data is first broadcast to all routers, then every router that receives unwanted multicast data sends a pruning packet to its parent. MOSPF [17] protocol is an extension to OSPF protocol. Every router refers to the link state database and the group membership knowledge, and constructs shortest-path tree from any source to all receivers. CBT [4] and PIM-SM [10] constructs a shared tree rooted at a group-specific Rendezvous Point (RP). BGMP [15] constructs bidirectional shared tree that is rooted at the home domain whose address allocation includes the group's address. However, all of above protocols cannot become efficient solutions for Internet-wide multicast services because of their scalability limitations. PIM-SSM [12] protocol bypasses the discovery of RP, and constructs shortest path trees rooted at the single source domain. This protocol can be used only for single-source multicast model.

FRM [21] uses the source-encoding forwarding technique, where the source router forms the dissemination tree and inserts the tree information into the header of multicast packet. Their protocol has severe packet duplications especially when the group size is large. While our protocol utilizes hierarchical decomposition and shim payload (instead of shim header) to accommodate large trees, we can effectively reduces the number of duplicated packets.

Many application-layer protocols have been proposed in recent years. They can be classified as tree-based [5,11,13,20,27] or mesh-based [7,8,16] protocols, depending on whether the dissemination tree is maintained directly, or a mesh is maintained and the tree is constructed over the mesh on demand. The proxy based overlays [6,14,22] can provide more reliable and efficient multicast service,

where the proxies are application servers deployed throughout the Internet. They self-organize into overlays to disseminate multicasting packets, and relay packets to attached end-hosts.

Our protocol can be viewed as a variant of proxy based overlay where the MA in each domain plays the role of multicast proxy. However, both the membership management and the routing method in our protocol are completely different from those in traditional proxy overlays. (1) To manage membership information, proxy overlays propagate updating information to all members through periodically neighboring exchanging. We do not adopt such techniques in our protocol because of the following reasons: first, MA servers are much more reliable than end-hosts, it is not worthwhile to detect the rare abrupt failures of members using periodic refreshing messages at the expense of high bandwidth consumptions. Second, the constraints on the exchange frequency between border routers introduce latency to the propagations of membership information. In our protocol, we use foster-parent technique to reduce the join-delay, and use augmented-packets to minimize the communication overhead. (2) For the routing mechanisms, proxy overlays construct mesh and measure link qualities periodically, our protocol explores the knowledge about the domain-level unicast paths available at BGP border routers, and constructs source-optimal dissemination trees directly.

8 Conclusion

In this paper, we proposed a secure and scalable multicast architecture over IP Network. In our architecture, the AS-level group membership are explicitly maintained by in-group Multicast Agents (MA), the inter-domain routing is based on source-encoded information in multicast packet, and the multicast packets are encrypted using two-level keys: a global key at the inter-domain level and a local key at the intra-domain level. Our future work involves implementing the proposed multicast architecture in real Internet environments.

Acknowledgments

This work was partially supported by NSF under grants No. CNS-0644238, CNS-0626822, and CNS-0831470. We appreciate anonymous reviewers for their valuable suggestions and comments.

References

1. Rules for Delivery of CM Alerts to the Public During Emergencies (April 2008), `http://hraunfoss.fcc.gov/edocs_public/attachmatch/FCC-08-99A1.pdf`
2. Routing Information Service (October 2007), `http://www.ripe.net/projects/ris/index.html`
3. Adams, A., Nicholas, J., Siadak, W.: Protocol Independent Multicast - Dense Mode (PIM-DM) Protocol specification (Revised). Internet Draft (October 2003)

4. Ballardie, T., Francis, P., Crowcroft, J.: Core based trees (CBT) an architecture for scalable inter-domain multicast routing. Technical report, San Francisco, CA (September 1993)
5. Banerjee, S., Bhattacharjee, B., Kommareddy, C.: Scalable application layer multicast. In: Proceedings of ACM SIGCOMM (September 2002)
6. Banerjee, S., Kommareddy, C., Kar, K., Bhattacharjee, B., Khuller, S.: Construction of an efficient overlay multicast infrastructure for real-time applications. In: Proceedings of IEEE INFOCOM (April 2003)
7. Chawathe, Y.: Scattercast: An Architecture for Internet Broadcast Distribution as an Infrastructure Service, Ph.D. Thesis, University of California, Berkeley (December 2000)
8. Chu, Y., Rao, S.G., Zhang, H.: A case for end system multicast. In: Proceedings of ACM SIGMETRICS (June 2000)
9. Deering, S., Cheriton, D.: Multicast routing in datagram internetworks and extended LANs. ACM Transactions on Computer Systems 8(2), 85–110 (1990)
10. Fenner, B., Handley, M., Holbrook, H., Kouvelas, I.: Protocol Independent Multicast sparse mode (PIM-SM): Protocol specification (October 2003); Internet Draft
11. Francis, P.: Yoid: your own internet distribution (March 2001), `http://www.isi.edu/div7/yoid/`
12. Fenner, B., Handley, M., Holbrook, H., Kouvelas, I.: Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised). Internet Draft (March 2001)
13. Helder, D.A., Jamin, S.: End-host multicast communication using switch-tree protocols. In: Proceedings of the Workshop on Global and PeertoPeer Computing on Large Scale Distributed Systems (GP2PC) (May 2002)
14. Jannotti, J., Gifford, D., Johnson, K., Kaashoek, M., OToole, J.: Overcast: reliable multicasting with an overlay network. In: Proceedings of the Symposium on Operating Systems Design and Implementation (October 2000)
15. Kumar, K., Radolavov, P., Thaler, D., Alaettinoglu, D., Estrin, D., Handley, M.: The MASC/BGMP architecture for inter-domain multicast routing. In: Proceedings of SIGCOMM, Vancouver, Canada (September 1998)
16. Liebeherr, J., Beam, T.: HyperCast: a protocol for maintaining multicast group members in a logical hypercube topology. Networked Group Communication, 72–89 (1999)
17. Moy, J.: RFC 1585: MOSPF. Analisys and Experience. Proteon Inc. (March 1994)
18. Mittra, S.: Iolus: A framework for scalable secure multicasting. In: ACM SIGCOMM, pp. 277–288 (1997)
19. Oliveira, R., Lad, M., Zhang, B., Zhang, L.: Geographically Informed Inter-domain Routing. In: Proceeding of IEEE International Conference on Network Protocols (ICNP) (October 2007)
20. Pendarakis, D., Shi, S., Verma, D., Waldvogel, M.: ALMI: An Application Level Multicast Infrastructure. In: Proceedings of 3rd Usenix Symposium on Internet Technologies & Systems (USITS) (March 2001)
21. Ratnasamy, S., Ermolinskiy, A., Shenker, S.: Revisiting IP Multicast. In: Proceeding of SIGCOMM 2006, Pisa, Italy, September 2006, pp. 11–15 (2006)
22. Shi, S., Turner, J.: Routing in overlay multicast networks. In: Proceedings of IEEE INFOCOM (June 2002)
23. Subramanian, L., Agarwal, S., Rexford, J., Katz, R.H.: Characterizing the Internet Hierarchy from Multiple Vantage Points. In: Proceedings of IEEE INFOCOM (June 2002)

24. Waitzman, D., Partridge, C., Deering, S.: Distance Vector Multicast Routing Protocol. ARPANETWorking Group Requests for Comment, DDN Network Information Center (November 1988); RFC-1075
25. Wallner, D., Harder, E., Agee, R.: Key management for multicast: Issues and architectures. IETF Request For Comments, RFC 2627 (June 1999)
26. Wong, C.K., Gouda, M.G., Lam, S.S.: Secure group communications using key graphs. In: ACM SIGCOMM, pp. 68-79 (1998)
27. Zhang, B., Jamin, S., Zhang, L.: Universal IP multicast delivery. In: Proceedings of the International Workshop on Networked Group Communication (NGC) (October 2002)
28. Content Addressable Memory Cypress Semiconductor, `http://www.cypress.com`