ABSTRACT

Title of Thesis:	IP ROUTING AND KEY MANAGEMENT FOR SECURE MULTICAST IN SATELLITE ATM NETWORKS
Degree candidate:	Ayan Roy-Chowdhury
Degree and year:	Master of Science, 2003
Thesis directed by:	Professor John S. Baras Department of Electrical and Computer Engineering

This thesis proposes a design for secure IP multicast in hybrid satellite networks. The emergence of IP multicast for Internet group communication has placed focus on communication satellites as an efficient way to extend IP multicast services for groups with distributed membership in wide-area networks. This poses interesting challenges for routing and security. Satellite networks can have both wired and wireless links and also different link-layer technologies like Ethernet and ATM. No proposed IP multicast routing protocol for wired networks offers an integrated solution for such networks. From the security viewpoint, in order to be commercially viable, the multicast traffic should be accessible only to legitimate receivers. Access control can be achieved by data encryption, with the correct decryption keys provided to authorized members.

This requires secure and efficient methods to generate, distribute and update the keys. Most current key management protocols do not scale well when applied to large dynamic groups operating in wide-area networks. This thesis attempts to solve the above problems and proposes a design for secure multicast in wide-area networks that have terrestrial Ethernet LANs interconnected by ATM-based satellite channels. The thesis reviews the multicast services offered by IP and ATM, and proposes a multicast routing framework for the hybrid satellite network. The thesis also looks at current group key management protocols, and designs a scheme for secure and scalable key management for the proposed multicast routing architecture. The frameworks are presented in detail, followed by analysis and simulation results.

IP ROUTING AND KEY MANAGEMENT FOR SECURE MULTICAST IN SATELLITE ATM NETWORKS

by

Ayan Roy-Chowdhury

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2003

Advisory Committee:

Professor John S. Baras, Chair
Professor Virgil D. Gligor
Professor Min Wu

DEDICATION

To Baba, Ma and Chordibhai.

ACKNOWLEDGMENTS

I am grateful to my advisor, Dr. John Baras, for his support, guidance and encouragement. I would like to thank Dr. Virgil Gligor and Dr. Min Wu for agreeing to serve on my committee and for reviewing this thesis.

My colleague, Nalini Bharatula, has helped me tremendously. I am indebted to her. I am also thankful to Vijay Bharadwaj, Gun Akkor, Dr. Majid Raissi-Dehkordi, Karthikeyan Chandrasekhar and Maria Striki for their help in analyzing the problems. Prabha Ramachandran and Gun Akkor were very helpful in critiquing the thesis document. Thanks also to Radostina Koleva for her valuable feedback on the presentation.

Thanks are due to the ISR and SEIL staff, notably Althia Kirlew and Trevor Vaughn, for their help in administrative matters.

I gratefully acknowledge the financial support that I receive from the Institute for Systems Research, through a Graduate Research Assistantship. The work reported in this thesis was made possible by a grant from Lockheed Martin Global Telecommunications, through Maryland Industrial Partnerships under contract number 251715.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

IP multicast routing [2] is a network layer mechanism that provides resource-efficient communication services for applications that send the same data to multiple recipients simultaneously. The source transmits a single copy of the data; an intermediate router makes a copy of each incoming multicast packet to retransmit on each outgoing link towards the destinations reachable from it. This makes efficient use of network bandwidth compared to sending multiple unicasts, where the source sends a copy of the packet separately to each receiver. Like broadcast, multicast allows simultaneous delivery to a set of clients, but multicast is selective in that the client set is a subset of the total nodes in the network. Applications that can benefit from use of multicast include webcasts, online stock updates, shared workspace, video- and voice-conferencing, distributed interactive simulation, file transfer, database access, and online gaming.

Satellite networks offer a natural method to extend the multicast services in wide-area networks where the sources and recipients are widely separated from one another. Satellites offer high bandwidth for broadband services, as many multicast applications are. Their broadcast nature allow the sources to reach multiple recipients simultaneously. For *geostationary* orbit satellites, the transmission from the source to recipients can be accomplished in a single hop, even if the recipients are geographically remote. The satellite networks are self-contained and require less infrastructure compared to terrestrial fiber-based networks, and hence can be set up rapidly. Satellites also offer an attractive option for interconnection of geographically distributed high-speed terrestrial networks. Satellites are hence expected to play a greater role in transmission of broadband multicast traffic in the future.

There is, however, little support today for IP multicast services over satellites. Most of the IP multicast routing protocols have been proposed for networks with homogeneous "tree" or "mesh" characteristics; they do not consider the satellite network architecture that can be hybrid in nature. Also, IP multicast implicitly assumes that Ethernet is used as the underlying access layer. Ethernet has native support for multicasting, therefore integrating IP multicasting with Ethernet multicast is relatively simple. However, the integration becomes much more complicated if we consider link layer technologies other than Ethernet. For example, ATM has no native support for multicast, and requires a fairly complex mechanism to support network layer multicast services over ATM links. Therefore, the design of IP multicast routing in a satellite network that supports a combination of Ethernet and ATM links is a fundamental issue that needs to be addressed. This is the routing problem we address in this paper.

The multicast model is "open" in nature - any host can join a multicast group and receive data. But in order for a multicast service to be commercially viable, it is important that access to the multicast data be tightly controlled so that only paying or authorized receivers can read the data. The multicast routing protocols do not give options to restrict receivers. Instead, access to the data can be controlled by means of encryption - the source encrypts the application content using a key; the decryption key is distributed to all authorized receivers. The mechanism of key distribution is challenging when the set of authorized receivers changes dynamically, with users joining and leaving the multicast group with time. Whenever the group membership changes, it is necessary to change the shared keys for the group. Hence there must exist an efficient system that generates and delivers the group keys to all members and updates the keys on membership changes, ensuring that at any point in time only authorized members have access to the decryption key to read the data in the group.

There have been several approaches to design efficient group key management[1] systems [3, 4, 5, 6, 7, 8]. The design problem becomes more challenging when we consider large groups of the order of thousands or even a million members, spread over a wide geographical area, as is the case for the wide-area satellite network that we consider. Hence in this work we also propose a framework for secure key management to ensure confidentiality of the multicast application data.

1.1 Contributions

This thesis makes the following technical contributions:

[1] The term *key management* refers to key generation, distribution and key updates in a group.

1. It proposes a design for routing that integrates IP with ATM for end-to-end multicast routing over a wide-area satellite network architecture, which has Ethernet-based terrestrial links and ATM-based satellite channels. For the design of the routing framework, the following issues are dealt with:

   - Analysis of IP multicast routing protocols and selection of a suitable protocol for the terrestrial networks.

   - Analysis of the support for IP multicast in ATM and its limitations; selection of a suitable mechanism for IP multicasting over ATM satellite links.

   - Integration of the IP multicast routing protocol with ATM multicast to create the end-to-end multicast tree.

   To demonstrate the viability of the routing framework, simulations of the framework are done and the simulation results are presented.

2. The thesis addresses the problem of scalable and secure key management in the satellite network. An analysis of various well-known key management protocols are done, and a framework is proposed for secure and scalable multicast key management for the satellite network. The proposed framework ensures confidentiality of the multicast application; it scales with a large number of users spread across wide regions; and efficiently handles the dynamics of group membership changes.

Simulation results are presented to demonstrate the feasibility of the key management framework.

1.2 Organization

The rest of the thesis is organized as follows. Chapter 2 covers the fundamental concepts of IP multicast and reviews some popular IP multicast protocols. Review of ATM multicasting is in chapter 3. Chapter 4 describes the network architecture and details the design of the proposed multicast routing framework. Simulation of the routing framework and the results of the simulation are given in chapter 5.

Some popular group key management protocols are analyzed in chapter 6. The proposed design of the key management framework is described in chapter 7. Simulation of the key management scheme and the results are given in chapter 8. We present our conclusions in chapter 9, including a highlight of additional issues and a discussion of future research directions.

Chapter 2

IP Multicast: Concepts and Routing Protocols

In this chapter, we first review the basic concepts of IP multicast and also discuss the support for IP multicast in satellite networks. We then look at the desirable features and challenges of multicast routing protocols. We review some of the popular intra-domain and inter-domain IP multicast routing protocols that have been proposed in the research community.

2.1 IP Multicast Fundamentals

The original IP multicast model, proposed in [2], is based on the notion of a *group*, identified by a unique *address*, and composed of a certain number of participants (senders and receivers). Here we review the basic concepts in IP multicast, based on the treatment in [9].

- IP Address Space: The IP address associated with a multicast group is assigned from the class D address space, which can range from 224.0.0.0 to 239.255.255.255. Some of these addresses are pre-assigned, while the others can be dynamically allocated at the time of group formation.

- Member Registration: The IP multicast protocols make use of the *Internet Group Management Protocol* (IGMP)[10] to find out about the participants in a group. All receivers in a multicast group are required to explicitly register the multicast address for which they wish to receive data, by sending join requests to their local IGMP-enabled multicast routers. When a receiver wants to leave a group, it sends an explicit leave request. The receivers can join and leave at any time during a multicast session. IP multicast hence "maps" a multicast address to a set of receivers.

  Registration is required only for receivers, but not for the senders to a group. The recipients can be anonymous; the sources need not know who the receivers are, also the receivers do not know each other.

- Multicast Tree: The join/leave requests of receivers are managed by IGMP-enabled routers in the local network. These requests, and the data packets sent by the sources, are forwarded by multicast-enabled routers. The multicast routers and the receivers together form the *multicast delivery tree*. The tree is an acyclic spanning tree; the exact structure of the tree is determined by the multicast routing algorithm used. The receivers are always at the leaves of the tree. The tree might have one or more root(s) or core(s), depending on the routing algorithm. The core(s), if present, is a(are) multicast router(s). Figure 2.1 shows a multicast group structure in a network.

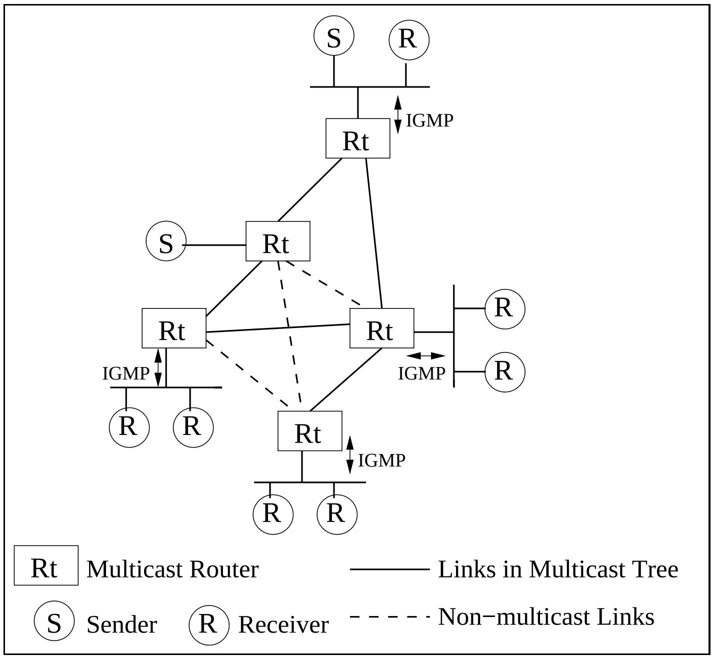


Figure 2.1: A Multicast Group

The multicast tree can be either a *shared tree*, i.e., a single common tree for a multicast group; or, *source-specific shortest path trees*, where every source for a multicast group has its own individual tree rooted at the source.

- Unidirectional or Bidirectional Forwarding: The multicast traffic in a group can be *unidirectional* or *bidirectional*. In unidirectional forwarding, the source(s) send the data packets to the core node; the data is then forwarded along the shared multicast tree to reach the set of receivers. Here the multicast data traffic always flows downstream, from the core to the leaves.

  In bidirectional forwarding, the multicast traffic from the source does not necessarily have to go through the core router(s) to reach the recipients in the tree. Bi-directional forwarding is hence a distributed approach compared to unidirectional forwarding.

- Managing the Multicast Tree: The management of the set of receivers in a multicast group depends on the routing protocol used. The routing protocol uses IGMP to detect changes in group membership, and accordingly adjusts the multicast tree. The routing protocols make use of one of the following three mechanisms to track membership changes:
    - Flooding: A receiver advertises its address to all the nodes in the domain. Flooding consists of forwarding a message on all outgoing interfaces, except the one it arrived from. Flooding is robust to link failures and packet loss, but it has heavy overhead in terms of duplicate packets. Flooding is suitable mainly for static multicast groups in which the membership does not change with time.
    - Centralized: A receiver advertises its membership only to the core of the multicast tree. The sources send to the core, which forwards to the receivers. Centralized schemes have minimal overhead in maintaining the multicast tree, but they suffer from the problem of single-point of failure. Also, the path from sources to receivers can be sub-optimal. Centralized schemes are suitable when the sources and receivers change frequently during a multicast session.
    - Distributed: A receiver advertises its address only to nodes in the multicast tree. The nodes are discovered through probe messages between a receiver

and its neighbors. Distributed schemes have higher overhead than centralized, but less than flooding.

In summary, support for IP multicast in wired networks requires the following mechanisms:

- Allocation of a class D address.
- Registration of the set of receivers.
- Setting up the multicast tree and dynamic membership management.
- Routing of traffic from the sources to the receivers along the multicast tree.

2.2 Wide-Area Multicast Routing via Satellites

Satellite networks have some inherent advantages in providing multicast service:

- Satellites can provide faster Internet access and higher throughput for applications due to their direct one-hop connectivity to the Internet backbone, bypassing congested multiple router-hops in terrestrial networks.
- Networks involving satellites can be set up faster compared to terrestrial networks, since the broadcast area of the satellite (the *satellite footprint*) can be quite large.

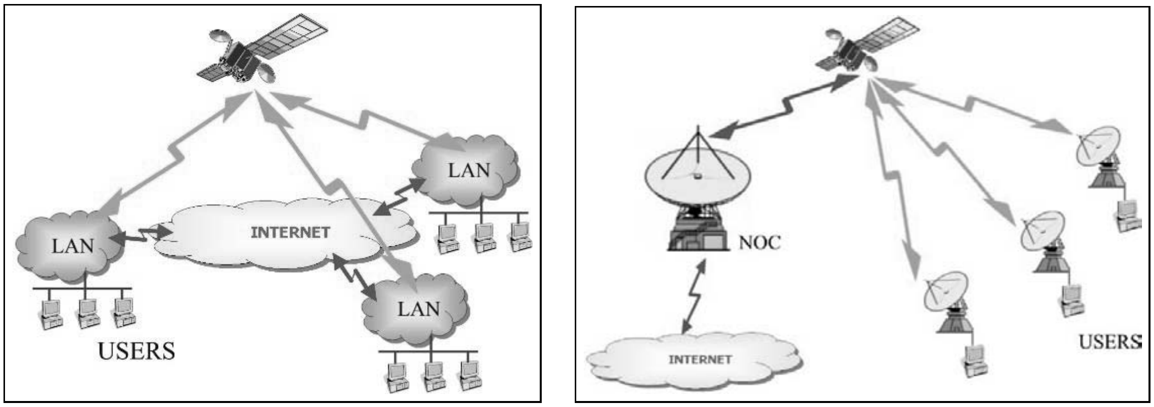
- The complexity in multicast routing protocols arise mainly from the necessity to route multicast packets over multiple hops, avoiding congested routes. This complexity can be avoided in a satellite network.

Terrestrial multicast networks are usually duplex, but satellite networks do not necessarily have multicast capability in the return path. A low cost (shared satellite or dial-up terrestrial modem) return link is often provided with limited capacity compared to the high-speed downlink [11]. The return channel is necessary for dynamic multicast groups, for allowing the users to join and leave the group during a multicast session.

There are two common topologies for support of multicast service in a satellite network [1]:

- a satellite can be deployed as a *backbone* for connecting local area networks (LANs) that are widely separated from one another. Each LAN has multiple terrestrial nodes and one or more satellite gateways that can uplink to and downlink from the satellite (figure 2.2(a)). The nodes in the LAN receive transmission from, and send to, the satellite via the gateway nodes. This topology is thus hierarchical in structure.

- The other topology is the *direct-to-home* (DTH), in which there are multiple independent terrestrial nodes, each with its own connectivity to the satellite. The connections can be unidirectional or bidirectional. The network has a star topology and user terminals have no access to other networks. The ground terminals access the terrestrial core network through a gateway node located at the Network Operations Center (NOC) (figure 2.2(b)).



(a) Satellite Backbone Deployment

(b) Satellite Direct-to-Home Deployment

Figure 2.2: Satellite Network Topologies[1]

Most deployed satellites do not perform on-board switching or processing; instead, they broadcast the data packets on all outgoing links. Future satellites are planned to be more sophisticated, supporting multiple spot-beams covering different geographical regions over a large area. These satellites will be able to do on-board switching and processing, and transmit the data packets only on the outgoing links that are necessary [12].

A geostationary satellite can connect large, widely-separated, terrestrial networks. The satellite will thus be a part of the multicast tree. If the networks in a multicast group are in different spot-beams, then the satellite will have to perform on-board switching for the multicast traffic. The challenge therefore is to design efficient routing protocols that would allow the satellite to do “selective” broadcast and send out the traffic only on the links that have receivers downstream. In the current Internet, multicast groups that span widely separated networks can be connected to each other through the use of multicast *tunnels*, e.g., the Multicast Backbone of the Internet (MBone) [13]. Managing a multicast group in this setting requires a complex setup with inter- and intra-domain multicast routing protocols, and the interfacing between the two. The relative simplicity of the satellite network can offer a simpler design for end-to-end multicast.

Most deployed satellites use their own link layer protocols. The amount of processing at the satellite is minimal. Since it is difficult to have a generic design based on proprietary protocols, one can look for standards that are closely matching. ATM is attractive since it supports very fast switching. It will also be more lightweight compared to IP routing. There have been proposals for satellites with ATM switching support. It is a challenging task to design a multicast routing framework that integrates terrestrial Ethernet networks with ATM satellite channels. Solutions using existing intra-domain protocols for the terrestrial networks, coupled with inter-domain protocols for managing the satellite connections between the networks, will not be efficient. Most protocols do not consider the broadcast nature of the satellite, or the multicast limitations imposed by ATM.

2.3 Challenges of Multicast Routing Protocols

The technical challenges faced by multicast routing protocols are [9]:

- Minimize the load on the network - avoid loops and traffic concentration on a link or subnetwork.

- Minimize the control message overhead required for setup and management of the multicast tree. Otherwise the protocol will not scale well to large groups.

- Provide basic support for reliable transmission, i.e., route changes have no adverse effects on the data delivery to receivers on the multicast tree.

- For the selection of optimal routes, consider different cost parameters like resource availability, bandwidth, link delay, end-to-end delay, etc.

- Minimize the state stored in the routers. Else the protocol will not scale to a large number of groups.

- Minimize processing at the nodes in the multicast tree.

- The protocol should be incrementally deployable and work well in an existing network, without requiring upgrades in all routers and the hosts.

2.4 Intra-domain Multicast Routing Protocols

Several protocols have been proposed for managing a multicast group within a domain. We survey some of the well-known ones, based on the treatment in [9, 14].

2.4.1 Multicast Extensions to Open Shortest Path First (MOSPF)

MOSPF [15] is the multicast extension of the Open Shortest Path First (OSPF) unicast routing protocol [16]. OSPF is a *link-state* routing protocol in which the routers advertise the state of their directly connected links.

To add support for multicast, a new type of link state advertisement, called "group membership LSA", has been added to OSPF. The group membership LSAs give detailed information on the routing topology and the receiver locations to every MOSPF router, which can hence compute the shortest path tree (SPT) from each multicast source to the set of receivers, without flooding the initial datagram from each source.

MOSPF requires heavy computation at each on-tree router for computing the SPT per source. For a network of N nodes, the number of computations increases as $\mathcal{O}(N^2)$ for every routing update. To improve scalability, the SPT can be computed on demand, when the first datagram from a source reaches an MOSPF router.

Another way to improve scalability in MOSPF is to partition the AS into *routing areas*, which are interconnected using a backbone network (figure 2.3). Multicasting within an area (*intra-area multicasting*) is done by computing the SPTs using group membership LSAs. Multicasting across areas (*inter-area multicasting*) is done via the backbone network. Inter-area multicasting is complicated due to a variety of reasons [14].

When the multicast group membership changes, MOSPF advertises changes in the

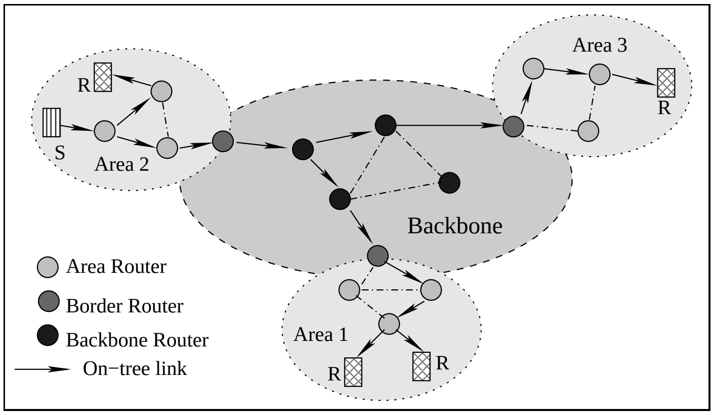


Figure 2.3: MOSPF Inter-Area Multicast

set of receivers to all the nodes in the area. This triggers a routing state update at every on-tree node, for each source. For a new active source, the multicast routers adjacent to it, need to compute the SPT that originates at the new source. Therefore if group membership changes frequently, MOSPF is slow to react, and incurs a heavy control message (LSA) overhead. Also, MOSPF needs to maintain routing state entry for every ⟨source, multicast group⟩, even if the source transmits infrequently. The protocol hence scales poorly to large groups. Partitioning the network into areas as above offers no significant advantage, whereas the complexity of multicast routing increases. For the above reasons, MOSPF is rarely used.

2.4.2 Distance Vector Multicast Routing Protocol (DVMRP)

DVMRP [17] is based on *distance vector routing*. DVMRP computes the multicast routing paths based on the unicast routing tables constructed by the unicast Routing Information Protocol (RIP)[18]. Hence, it is necessary to use RIP as the unicast protocol if DVMRP is to be used as for multicast routing.

For each multicast group, DVMRP version 3 [17] constructs source-based unidirectional multicast trees; the routing metric is the number of hops in the path. The multicast tree is constructed on-demand, when the initial data packet from the source arrives at a multicast router.

DVMRP uses “flood and prune” or Reverse Path Forwarding (RPF) [19] algorithm to construct the multicast tree. The incoming interface of each received multicast packet is checked against the interface used to unicast packets back to the source (RPF check)[1]. The initial multicast data packets are *flooded* to all the routers in the domain. The flooded packet reaches a router R in a leaf subnet (figure 2.4). If there are no group members present in the leaf subnet, R sends a “prune” message back towards the upstream router that forwarded the packet. The “prune” message indicates that data packets for the group from that particular source, should not be sent on the outgoing interface that leads to R . If an upstream router receives a prune message from all routers connected to all its outgoing interfaces, then it forwards a prune message up the tree.

The DVMRP multicast forwarding mechanism guarantees minimum end-to-end delay, since for each source an SPT is created. The algorithm is also robust to avoid routing loops. It is easier to implement compared to MOSPF. The computational complexity is also low in comparison. However, the flooding mechanism can incur a

[1] RPF check is done to avoid forwarding duplicate packets (due to loops); however, routing loops can occur in transient periods when the unicast routing tables are being updated.

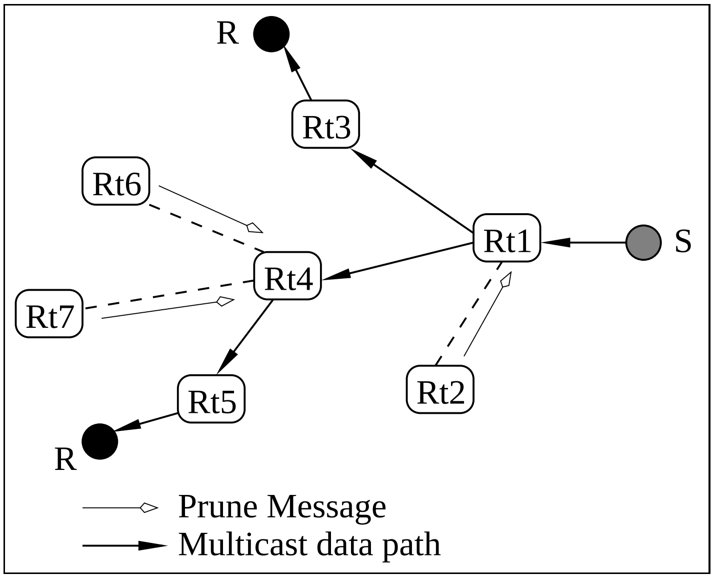


Figure 2.4: RPF Algorithm using Flood and Prune: Routers Rt3 and Rt5 have receivers downstream and accept the multicast data packets. Routers Rt2, Rt6 and Rt7 send prune messages to remove themselves from the SPT for source S.

heavy overhead in large networks with many sources. Also, DVMRP is a soft-state protocol requiring periodic refresh of the multicast prune state in each router, therefore the multicast packets need to be flooded periodically. DVMRP can also have heavy overhead in terms of storage, since each on-tree router needs to maintain state for every source per group. The routers that are not on the multicast tree also need to maintain prune state in case new members can be reached via them in the future. Hence for networks where most hosts are both receivers and sources, or if there are a large number of groups, each with many sources, DVMRP control can incur heavy consumption of network bandwidth and node memory [9].

2.4.3 Core-Based Tree (CBT)

CBT multicast routing protocol [20] uses a shared bidirectional tree for a group, in contrast to source-based unidirectional shortest path tree used in DVMRP.

CBT was developed to improve on DVMRP and MOSPF by addressing the scalability problems that arise due to periodic flooding to all nodes (as in DVMRP), and due to the need to maintain routing state per group and per source (MOSPF, DVMRP). This is done using the single shared tree, which requires less state information to be maintained at each multicast router per group. For example, in DVMRP, a router may need to maintain as many as n entries of the form (S_i, G) for $i \in 1, .., n$ where n is the number of senders in group G, and S_i is the i^{th} sender. On the other hand, in CBT, a router needs to maintain a single entry of the form $(*, G)$ irrespective of the number of senders [2].

CBT version 1 protocol (CBTv1)[21] is based on the use of multiple *cores*. A core is a fixed router in the network that acts as the center of the multicast group. Every multicast group has a *primary core* that is instrumental in setting up the multicast tree. Group members send “explicit” Join messages towards the primary core, creating a branch ending in the primary core, or ending in an existing branch of the tree. However, a single core might lead to long delays and inefficient utilization of resources for joining a group, particularly if the group members are widely dispersed. CBTv1

[2] Source-specific state can be used in CBT version 3, for backward compatibility with other protocols that might use the CBT domain as a transit domain [9]. However, source specific state is only set up on the tree branches spanning the border router and the core.

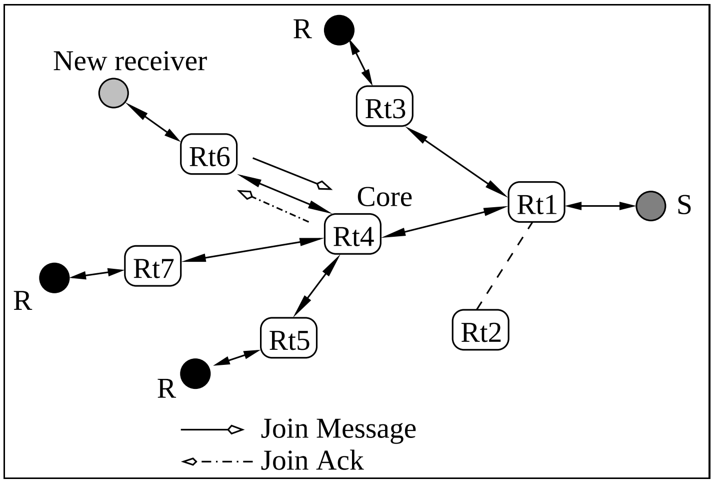


Figure 2.5: Core based tree in CBT. When a new receiver joins, a “Join” message is sent by the local router towards the core. A “Join Ack” is sent in response, creating bidirectional hard state in the nodes that constitute the branch of the tree to the new receiver.

therefore allows multiple *secondary cores* which act as primary cores within a local region; members in a local region join the secondary core, which in turn join the primary core. A secondary core has to join the primary core only once, irrespective of the number of members that join the secondary core. This reduces the control messages in the backbone network. However, using multiple cores can lead to stability problems, as explained below.

When a non-member source sends a packet, the packet is forwarded in the direction of the core until it reaches a node on the tree. The node forwards the packets on all the interfaces for the group, except the interface on which it arrived (bidirectional forwarding).

The primary drawback of CBT is that using a single shared tree leads to “traffic concentration” on a few links that are part of the shared tree. This can be avoided if source-based trees are used. Another drawback is that the sender and the receivers are not necessarily connected by the shortest path when using the shared tree. Therefore the delivery delay can be higher compared to using source-based shortest path trees.

CBTv1 using multiple cores is not robust since it can lead to loops. The Ordered Core Based Tree (OCBT) [22] was proposed as a solution to this problem. Hence, in CBT version 2 [23], only a single core is supported for robustness and easy implementation (figure 2.5).

2.4.4 Protocol Independent Multicast - Dense Mode (PIM-DM)

Protocol Independent Multicast [24] (PIM) has been proposed for multicast routing in an attempt to remove the deficiencies in other multicast routing protocols like DVMRP or CBT, while incorporating their positive features. As the name suggests, PIM is independent of the underlying unicast routing protocol. PIM comes in two flavors - *PIM Dense Mode* (PIM-DM) and *PIM Sparse Mode* (PIM-SM). We describe PIM-DM here, and PIM-SM in section 2.4.5.

PIM-DM [25] has been designed for networks that are densely populated with members of a multicast group. PIM-DM builds the multicast tree using “flood-and-prune” RPF, as in DVMRP. The primary difference between DVMRP and PIM-DM is that PIM-DM is independent of the unicast routing protocol; it simply requires that a unicast routing protocol exist to construct the unicast routing tables; PIM-DM uses the unicast routing tables to build the multicast tree. PIM-DM assumes that the unicast routes are symmetric. The packet forwarding on outgoing interfaces is also slightly different between PIM-DM and DVMRP. PIM-DM accepts additional overhead to simplify the RPF check. Else, the two protocols are very similar and the arguments for and against DVMRP apply to PIM-DM also.

2.4.5 Protocol Independent Multicast - Sparse Mode (PIM-SM)

PIM-SM [26] has been designed as a multicast routing protocol for a sparsely populated network. The definition of a region as *sparse* requires any of the following conditions to be true [14]:

- The number of networks/domains with members is smaller than the total number of networks/domains in a region.
- Group members are widely distributed.
- The overhead of flooding all the networks with data followed by pruning networks with no members in them is significantly high.

In addition, the groups are not necessarily small and hence dynamic alteration of the groups with a large number of members must be supported.

The features of PIM-SM design include [14]:

- low-latency data distribution if the application requires low end-to-end delay;

- independent of the underlying unicast routing protocol;
- inter-operability with other multicast routing protocols, like DVMRP or CBT;
- robustness - avoiding single point of failure, and to adapt gracefully to changes in network topology; and,
- scalability - the control message overhead should not exceed a certain percentage of the link bandwidth, irrespective of the size or distribution of the group.

To satisfy the above design requirements, PIM-SM supports both shared tree and shortest path trees. PIM-SM uses the concept of a central node for a multicast group, like CBT. The central node in PIM-SM is called the *Rendezvous Point* (RP). A unique RP for each group is determined based on the multicast group address. The selection of the RP is done by a router that is called the *Bootstrap Router* (BSR). The BSR is dynamically elected within a PIM domain.

In PIM-SM, the routers responsible for managing group membership in the leaf subnets are called the *Designated Routers* (DRs). When any receiver wants to join the multicast group, its DR sends an explicit “join” request to the RP. The join message is processed by all the routers between the receiver and the RP; the routers save the state information for the group. Thus a branch of the multicast tree for the new member is set up (figure 2.6).

When a sender wants to multicast to a group, its DR initially encapsulates the data packets and unicasts them to the RP, which then forwards the de-capsulated data

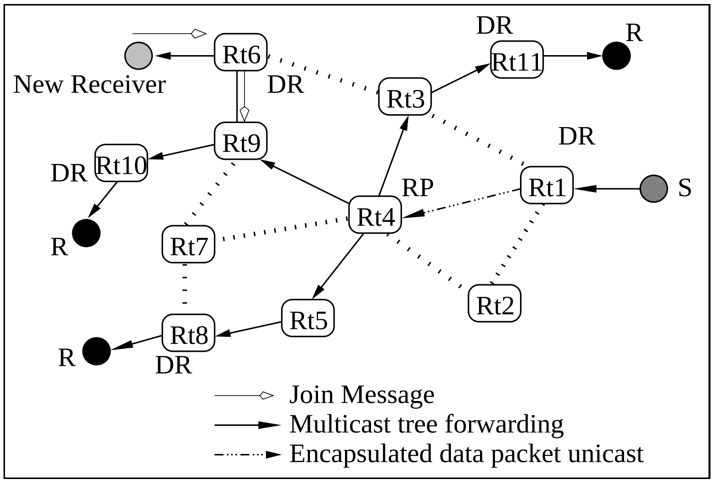


Figure 2.6: Shared RP Tree in PIM-SM. “Join” message for new receiver is sent by its DR towards the RP till it reaches a on-tree router. The DR for source S initially unicasts encapsulated packets to the RP, which de-capsulates the packets and forwards them to all receivers along the shared tree.

packets to the receivers along the shared multicast tree (figure 2.6). If the sender’s traffic increases beyond a pre-determined threshold, then the shortest path tree is created rooted at the sender. All the routers on the shared tree between the RP and the receivers send a “join” message towards the source and a ”prune” message towards the RP, thereby creating the source-rooted SPT (figure 2.7). The RP itself joins the SPT. Once the source-rooted tree is created, the source forwards the data packets along the SPT, and not the RP-rooted shared tree (RPT). The RP continues to receive a copy of the multicast data packet (in native format), and forwards the packet along the shared RP tree. This is done because there might still be receivers who are receiving from the shared tree. It also ensures that new receivers who join the group are able to receive data packets for the group till the time they switch to the SPT.

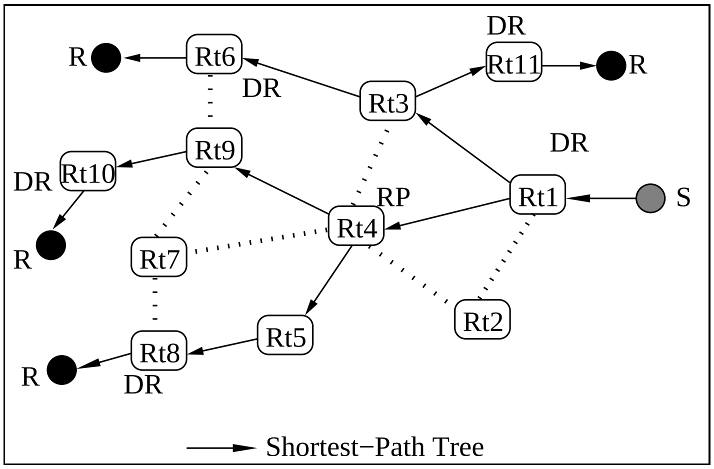


Figure 2.7: Source-specific shortest-path tree in PIM-SM. All the receivers switch to the shortest path tree when the data rate of the source exceeds a threshold. The RP also receives the data packets in native format from the shortest-path tree.

PIM-SM forwarding uses RPF check on the incoming interface to trace looping packets. The unicast routing information is derived from the unicast routing tables, independently of the unicast routing protocol that constructed them.

PIM-SM uses “semi-soft” states - the state information in each on-tree router has to be periodically refreshed (by sending join/prune message for each active entry in the PIM routing table). The periodic messages can reflect changes in topology, state or membership information. If the periodic update message is not received from a downstream router within the pre-set timeout period, the state entry is deleted from the upstream router’s local memory. Since the state information is periodically refreshed, PIM-SM does not need explicit *tear down* mechanism to remove state when a group ceases to exist.

PIM-SM and CBT share some similarities; both have been designed for sparse mode networks, and both use shared trees rooted at some central node. However, in PIM-SM the packets have to be first unicast to the RP, which then forwards them down the multicast tree - this is unidirectional forwarding, as opposed to CBT bidirectional forwarding. Also, PIM-SM can switch to the shortest path tree, which CBT lacks.

PIM-SM is a complex routing protocol; the amount of detail in the operation of the protocol is extensive. It creates large routing tables and requires significant memory at the routers to store the multicast state. The complexity of processing at the routers is also high. However, the protocol has many attractive features such as fast join to the multicast tree, low latency for high data rate sources, robustness to loops and node failures, that have led to its wide deployment.

2.4.6 Multicast Internet Protocol (MIP)

MIP [27] improves on some of the drawbacks that are faced in PIM-SM and CBT. Like PIM-SM, MIP is independent of the underlying unicast routing protocol, and it allows construction of both shared trees and shortest-path trees. But unlike PIM-SM, the multicast tree construction in MIP can be initiated by either the sender or the receiver or both. The two modes are interchangeable, and allows to construct a tree that is tailored according to the dynamics of the application and the group size.

MIP uses *diffusion* operations [28] to construct the multicast tree and manage the multicast group. This allows the multicast tree to be loop-free, even if the underlying unicast tables are inconsistent and contain routing loops. However, the diffusion mechanism is heavy in terms of control overhead. Hence it is not popular like PIM or CBT, where temporary loops are accepted for protocol simplicity. The loops also occur rarely, since the unicast routing tables do not change frequently in wired networks.

2.5 Inter-domain Multicast Routing Protocols

Several protocols have been proposed for managing a multicast group across different domains. Here we address some of the protocols that attempt to construct a multicast tree between domains, or branches of an existing intra-domain multicast tree that expand inter-domain. We do not consider the protocols that address constrained multicast routing, or policy routing. The descriptions given here are based on the surveys in [9, 14].

2.5.1 Hierarchical DVMRP (HDVMRP)

HDVMRP [29] aims to overcome the heavy overhead incurred by DVMRP when applied to wide-area networks consisting of many domains.

HDVMRP partitions a network into non-overlapping “regions” (which are different from autonomous systems). It organizes the network into a two-level hierarchy - the top-level consisting of non-overlapping regions and the lower level consisting of subnets within regions (figure 2.8). DVMRP is proposed as the inter-region multicast protocol. Any multicast protocol can be used for multicast within a region. The regions are interconnected through border routers that exchange information about the regions in the top-level only, and thus reduces the amount of information exchanged between the routers, and also reduces the number of entries in the routing tables.

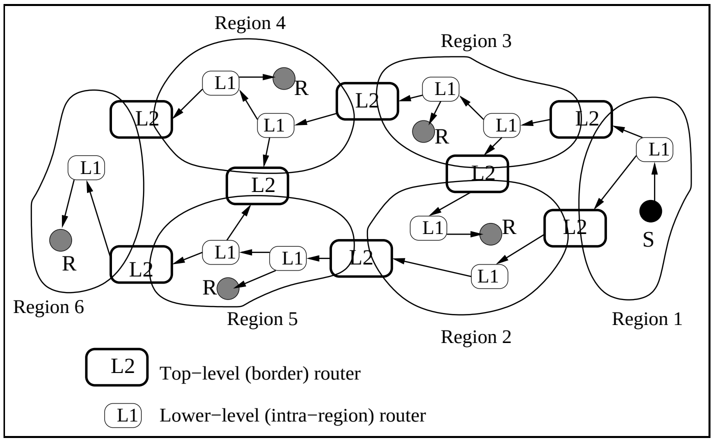

Region 4
Region 3
Region 6
Region 5
Region 2
Region 1
L2
L1
R
S
L2 Top−level (border) router
L1 Lower−level (intra−region) router

Figure 2.8: Inter-region Multicast Tree in HDVMRP

However, HDVMRP floods data packets to the border routers of all regions, and border routers that are not part of the group send prunes toward the source network to stop receiving packets. This implies a large overhead and maintenance of state per source, even when there is no interest for the group. HDVMRP also requires encapsulating the data packets for transit between the regions, which adds additional overhead.

2.5.2 Hierarchical PIM (HPIM)

HPIM [30] was designed to overcome the drawback in PIM that the placement of the RP can be sub-optimal for a sparsely distributed group in a large network.

HPIM uses a hierarchy of RPs for a group. Each candidate RP belongs to a certain level. An RP at a higher level has a wider coverage area. A receiver would send join messages to the lowest level RP (which is its local DR), which in turn would join an RP at the next higher level and so on, till the top-level RP is reached. Data flows in a bidirectional manner along the tree of RPs (figure 2.9).

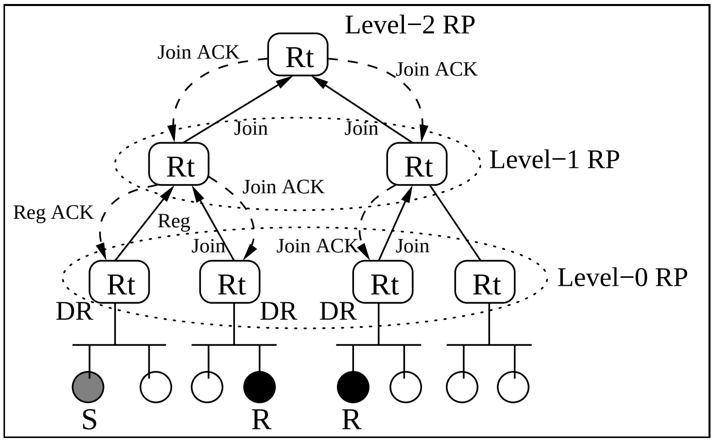


Figure 2.9: Hierarchical Multicast Tree in HPIM

The hierarchy of RPs helps in detecting loops and in decoupling control flow from the data flow. Even if control packets follow sub-optimal routes, data packets follow an improved route. However, it is difficult to come up with a hierarchical placement of RPs without extensive knowledge of the network topology and the receiver set. Also, the tree in HPIM does not perform well in terms of delays from the source to receivers, especially in the case of local groups.

2.5.3 PIM-DM/PIM-SM

The combination of PIM-DM and PIM-SM was an early proposal for inter-domain multicast routing - PIM-DM to be used for intra-domain routing, while PIM-SM will connect the domains. Thus, PIM-DM will maintain source-rooted trees at every domain, that will be connected by a shared tree (and source-rooted trees) constructed by PIM-SM. The RP set is advertised to all border routers in the inter-domain level, to provide a mapping between each multicast group address and the respective RP.

The approach cannot be applied to a large heterogeneous network since the mechanism to advertise RPs and the maintenance of soft state entries in PIM-SM will have heavy control overhead. The amount of state entries required to be maintained is also not feasible for an inter-domain protocol (one state entry for the shared tree, and then as many as the number of source-specific trees available).

2.5.4 Border Gateway Multicast Protocol (BGMP)

BGMP [31] has been proposed to address the issue of inter-domain multicast routing. BGMP is designed to inter-operate with any multicast routing protocol employed intra-domain, e.g., PIM-SM, CBT, DVMRP, etc.

BGMP associates each multicast group with a root or core and constructs a shared tree of domains, similar to PIM-SM or CBT. However, the root is an entire domain in BGMP, and not a single router. The selection of the root domain in BGMP is based on the multicast address prefix allocated by the Multicast Address-Set Claim (MASC) protocol [32]. BGMP also makes use of the Border Gateway Protocol (BGP) [33] which carries the multicast group prefixes between domain border routers.

Specific ranges of the class D address space are associated with various domains.

Each of these domains is selected as the shared tree root for all groups whose address is in its range. The association is done such that the root domain is usually chosen to be the domain of the group initiator under the assumption that this domain will source a significant portion of the multicast data.

Figure 2.10 shows the architecture of BGMP which consists of the following components:

1. Domains or autonomous systems

2. Border routers with two components: (1) BGMP component and (2) Multicast Interior Gateway Protocol (M-IGP) component. The M-IGP component can be any intra-domain multicast routing protocol.

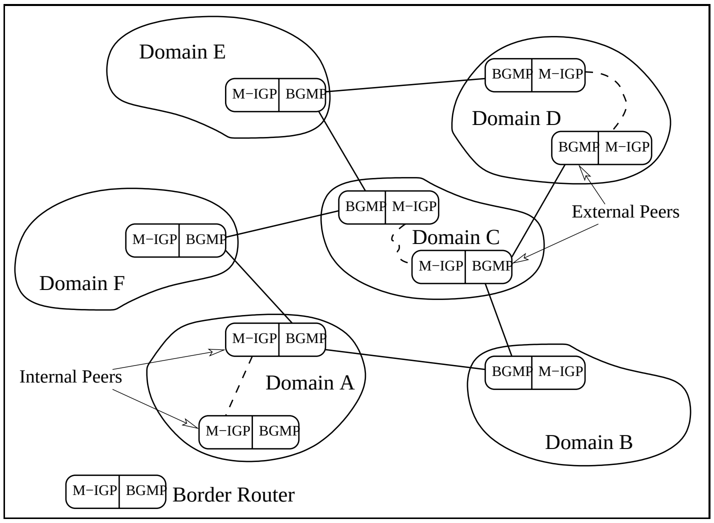


Figure 2.10: BGMP Inter-domain Multicast Architecture

BGMP runs on the border routers and and constructs a bi-directional shared tree that connects individual multicast trees built in a domain. The M-IGP component informs the BGMP component in the border routers about group membership in the domain. This triggers BGMP to send “Join” and “Prune” messages from border router to border router until the message reaches the root domain or a border router that is already on the shared tree.

In order to ensure reliable control message transfer, BGMP runs over TCP. BGMP routers have TCP peering sessions with each other to exchange control messages. The BGMP peers for a certain group are determined based on BGP.

Due to bi-directional forwarding, BGMP is not adequate for asymmetrical routing environments [9]. Moreover, BGMP can only support source-specific delivery criteria in limited cases, for keeping the protocol simple. To obtain a globally available multicast routing solution, the use of BGMP necessitates that inter-operability problems, specific to the M-IGP being used, be solved.

Chapter 3

ATM Support for IP Multicast

The IP multicast model is based on the premise that there exist technologies at the lower layers to natively support IP multicast service, e.g., Ethernet broadcast which does a simple mapping between IP class D addresses and Ethernet multicast addresses to support IP multicast.

ATM networks based on UNI 3.0/3.1 [34, 35] do not provide the native multicast support expected by IP; the specifications do not have the concept of abstract group address for multicasting as in IP. Therefore if a sender wants to multicast data to a group of recipients, it has to know apriori the ATM addresses of the set of recipients, and it needs to set up multicast connections rooted at itself, to the set of receivers before it can send the data packets. This is in contrast to IP, where the multicast model is receiver-initiated.

In this chapter we first look at the mechanisms provided by UNI 3.0/3.1 to support one-to-many communication. We then review the additions that have been made to support many-to-many communication, and finally look at the support for IP multicasting in ATM.

3.1 ATM Point-to-Multipoint VC

One-to-many traffic flow in ATM is done using a unidirectional *point-to-multipoint virtual connection* (p2mpVC) (figure 3.1), which is specified in UNI 3.0/3.1. The point-to-multipoint VC is initiated from the sender ATM endpoint by opening a point-to-point virtual connection (p2pVC) to the the first receiver ATM endpoint by explicit ATM signaling mechanism. The sender subsequently adds “branches” to the point-to-point VC, specifying the other receiver ATM addresses; the signaling ensures that branches are created in the intermediate ATM switches on the path from the sender to the set of receivers as appropriate. The sender is also responsible for connection tear down when it ceases data transmission.

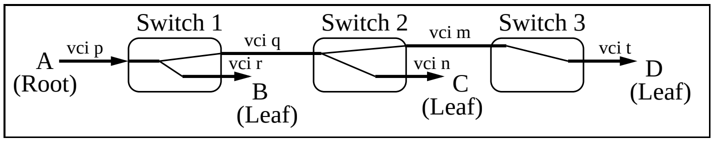


Figure 3.1: Point-to-Multipoint Virtual Connection

From the source’s perspective, the point-to-multipoint VC appears much like a point-to-point VC. The source transmits a single copy of each cell; cell replication happens at the ATM switches where branching occurs. Provided that each leaf node terminates the VC with the same ATM adaptation layer (AAL) service as used by the source, this point-to-multipoint VC effectively supports the unidirectional multipoint distribution of higher level AAL service data units (AAL_SDUs) [36].

In UNI 3.0/3.1, an ATM node who wants to receive cannot add itself to the p2mpVC. If the set of recipients changes during the lifetime of the connection, the source must explicitly add or remove any new or old recipients, by specifying the leaf node's actual unicast ATM address.

3.2 ATM Multipoint-to-Multipoint Communication Model

Emulating multipoint-to-multipoint service in ATM networks based on UNI 3.0/3.1 can be done using one of two methods:

1. a *VC mesh*, or,
2. a *multicast server* (MCS).

3.2.1 VC Mesh

The VC mesh is the simplest approach: each ATM sender creates its own unidirectional point-to-multipoint VC with the set of receivers as the leaf endpoints. Nodes that are both sources and receivers for a group will originate a single point-to-multipoint VC and then terminate a branch of one other VC for every other sender of the group. This results in a criss-crossing of VCs across the ATM network, hence the term *multicast mesh* of *VC mesh*. Figure 3.2 shows a VC mesh with four ATM nodes, each acting both as source and receiver.

The primary advantages of the VC mesh approach are as follows:

1. *Optimal data path performance*: cell replication load is distributed across all the

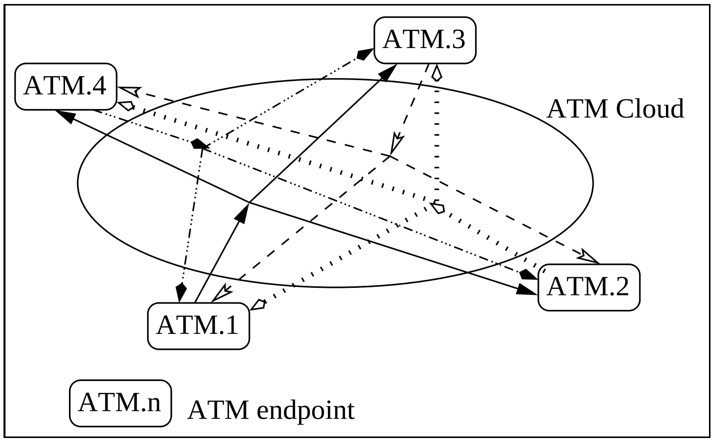


Figure 3.2: VC Mesh Architecture

switches in the network. Only switches on the multipoint distribution tree for a given source carry traffic from that source.

2. *Low latency*: the sender uses its own source-specific shortest path tree, without depending on any shared mechanism to distribute data on its behalf.

3. *Differential service*: since each sender uses a separate VC, it is possible to provide different quality of service for different senders to the same group [14].

The primary disadvantages of the VC mesh approach are:

1. *High usage of resources*: there are as many point-to-multipoint VCs as there are senders. The number of VCs increases linearly with the number of sources. For large number of sources, this leads to high network resource consumption.

2. *Heavy signaling load*: the signaling load placed on the ATM network by a group membership change is proportional to the number of active sources, since each

source has to update its point-to-multipoint VC to reflect the change in group membership.

3.2.2 Multicast Server (MCS)

The multicast server (MCS) architecture attempts to overcome the drawbacks of the VC mesh approach by using servers to forward multipoint-to-multipoint traffic.

The MCS attaches to the ATM network and acts as a proxy group member. It terminates point-to-point VCs from all the endpoints, either sources or receivers, and originates one point-to-multipoint VC which is sent out to the set of all group members. The basic function of the MCS is to reassemble AAL_SDUs from all the sources and retransmit them as an interleaved stream of AAL_SDUs out to the recipients. This is sometimes called the shared tree model, as traffic from all sources shares a point-to-multipoint distribution tree from the multicast server [36].

The paths out to the receivers must be established prior to packet transmission, and the multicast servers require an external mechanism to identify these receivers. Figure 3.3 shows the MCS architecture for one server. However, a single group might utilize more than one multicast server to forward the traffic.

The main advantages of the MCS architecture are:

1. *Low consumption of resources*: since the system has only one point-to-multipoint VC to the receivers, rooted at the MCS, this reduces consumption of VC resources compared to the VC mesh architecture in a similar network.

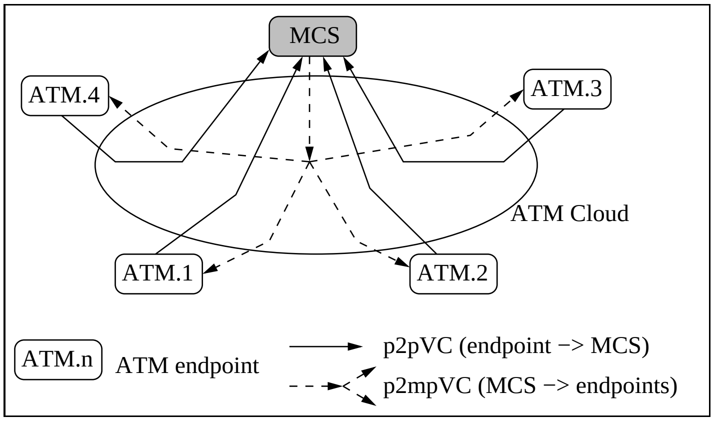


Figure 3.3: MCS Architecture

2. *Low signaling overhead*: if the group membership changes during the lifetime of a session, the amount of signaling traffic required to modify the distribution tree is much less compared to the VC mesh case. For example, if a new member joins, only two events occur: (i) the new member sets up its own point-to-point VC to the MCS, and, (ii) the MCS adds the new member as a leaf to its point-to-multipoint VC.

The major drawbacks of the MCS architecture are:

1. *Traffic concentration*: the MCS represents a single point of congestion for traffic from all sources, since every sender sends its data to the MCS; this increases the load on the server (or servers) and the links nearest to the multicast server itself. The MCS can potentially become a bottleneck for the group traffic. This can also have negative consequences for other customers attaching to the ATM network at or near the same switch as the multicast server.

2. *High latency*: the end-to-end latency experienced by each source's traffic is

	VC Mesh	MCS
Total VCs terminated at the group members	$n * m$	$n + m$
Point-to-Multipoint VCs	n	1
VCs terminated at each group member	n	2
Signaling requests generated due to a single membership change	n	2

Table 3.1: Cost of VC usage in VC mesh and MCS architectures [37]. m is the number of group members, n is the number of senders to the group.

potentially increased due to the longer path lengths and the AAL_SDU re-sequencing that must occur within the MCS server.

3. *Single point of failure*: If the multicast server stops, every source's traffic is lost.

4. *Reflected packets*: the MCS does not distinguish between source and receiver. Hence if a group member is also a source, it will receive copies of its own AAL_SDUs from the MCS point-to-multipoint VC, in addition to the AAL_SDUs from other sources. IP explicitly prohibits the underlying link interface from looping back packets. Hence protocols providing IP multicast over ATM must include additional mechanism *per AAL_SDU* to enable the detection and filtering out of such reflected packets before they reach the IP layer.

Based on [37], table 3.2.2 gives the VC cost in VC mesh approach and in the MCS approach.

IP Multicast Address	ATM Endpoint Address
Class D address$_1$	$\{$ATM.1, ATM.2, ..., ATM.n$\}$
Class D address$_2$	$\{$ATM.1, ATM.2, ..., ATM.n$\}$
.	.
.	.
.	.
Class D address$_N$	$\{$ATM.1, ATM.2, ..., ATM.n$\}$

Figure 3.4: IP-ATM address mapping table at MARS

3.3 IP Multicast Support in ATM: MARS Architecture

In order to make IP multicast work over ATM, the use of *Multicast Address Resolution Server* (MARS) [36] has been proposed. MARS is used to map IP multicast addresses to the ATM addresses of the endpoints belonging to the IP multicast group.

The MARS keeps a table of $\langle$Class D address, ATM address 1, ATM address 2, ..., ATM address n$\rangle$ mappings for every layer 3 multicast group that has one or more members (figure 3.4).

MARS satisfies the following requirements for IP multicast over ATM [36]:

- Provide a central registry that tracks which ATM addresses represent the current set of members to any given IP multicast group address.
- Provide a mechanism for IP/ATM endpoints to signal the central registry when they wish to join or leave an IP multicast group.

- Provide asynchronous updates to all relevant parties if and when changes to this registry occur.
- Allow for the use of multicast servers or VC meshes to support the traffic on particular IP multicast groups, in a manner transparent to each IP source.

The set of IP/ATM endpoints managed by a single MARS is known as a *cluster*. In the traditional model, the IP hosts are grouped into clusters or *Logical IP Subnets (LIS)*, and each such subnet has a MARS. The clusters are interconnected using IP multicast routers. Thus *inter-subnet* multicasting is still done using IP multicast routing protocols, while the *intra-subnet* multicasting is done using ATM with the help provided by MARS [14].

As described in [36], each IP/ATM interface logically attached to a particular cluster is considered to be a MARS client - a client of the MARS that supervises a given cluster. Interfaces within both hosts and routers are considered to be MARS clients.

Two types of VCs are used to carry control messages between a MARS and its MARS clients:

1. A transient point-to-point VC to the MARS carries query/response activity initiated by the MARS client. There is one such VC for every MARS client connected to the MARS.
2. For control messages propagated by the MARS, the MARS uses a semi-permanent point-to-multipoint VC that has all its MARS clients as leaf

nodes. This VC is known as the *ClusterControlVC (CCVC)*. Before a MARS client may use a given MARS, it must register with the MARS, allowing the MARS to add it as a new leaf of the CCVC. A registered client is also known as a cluster member.

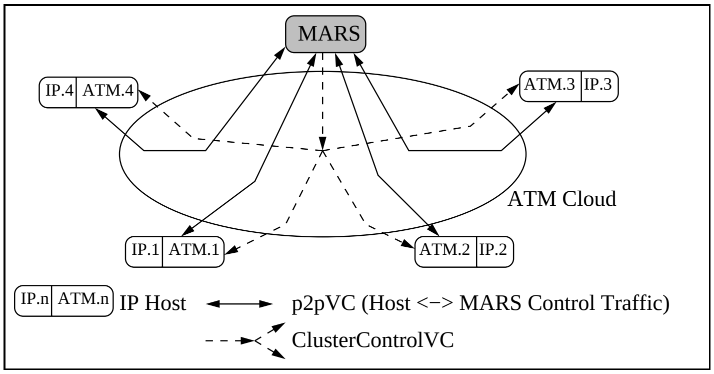


Figure 3.5: MARS Architecture

In addition, if ATM multicast for a group is done using multiple MCSs, MARS establishes a point-to-multipoint VC called the *ServerControlVC* to the MCSs.

Figure 3.5 shows the MARS architecture.

An ATM endpoint who wants to send to an IP multicast group, queries the MARS for the list of ATM addresses of the multicast group members. On receiving the list from the MARS in a reply message, the endpoint proceeds to send the multicast traffic to the endpoints. The actual transfer of the multicast traffic can be done using either the VC mesh or the MCS architecture.

The signaling mechanism and message exchanges for doing IP multicast over an ATM network using the MARS for address mapping, and VC mesh or MCS for point-to-multipoint data distribution, is described in detail in [38]. Figures 3.6 and 3.7

show the multicast architectures for VC mesh and MCS respectively using the MARS for address mapping.

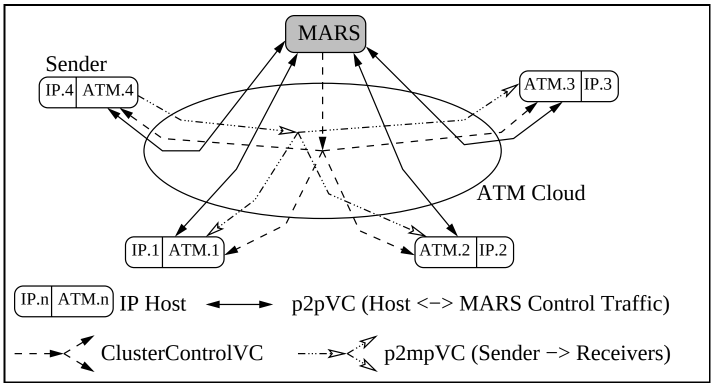


Figure 3.6: IP/ATM Multicast using MARS and VC Mesh

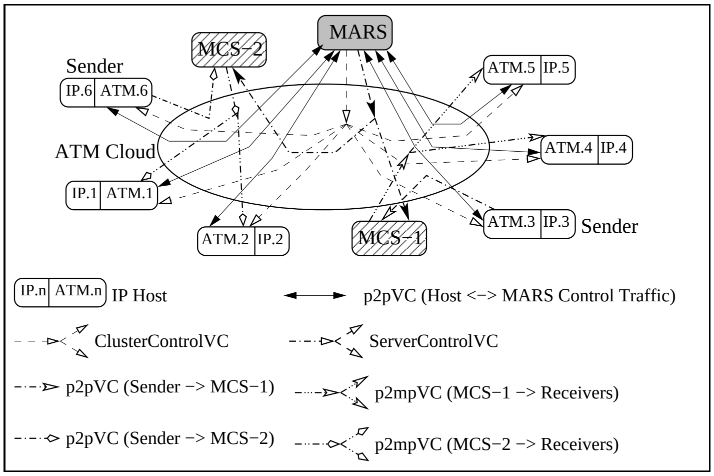


Figure 3.7: IP/ATM Multicast using MARS and MCS

Chapter 4

Framework for IP Multicast Routing in Satellite ATM Network

4.1 Satellite Network Architecture

The network architecture under consideration is shown in figure 4.1. The topology is of the satellite backbone type that is discussed in chapter 2.

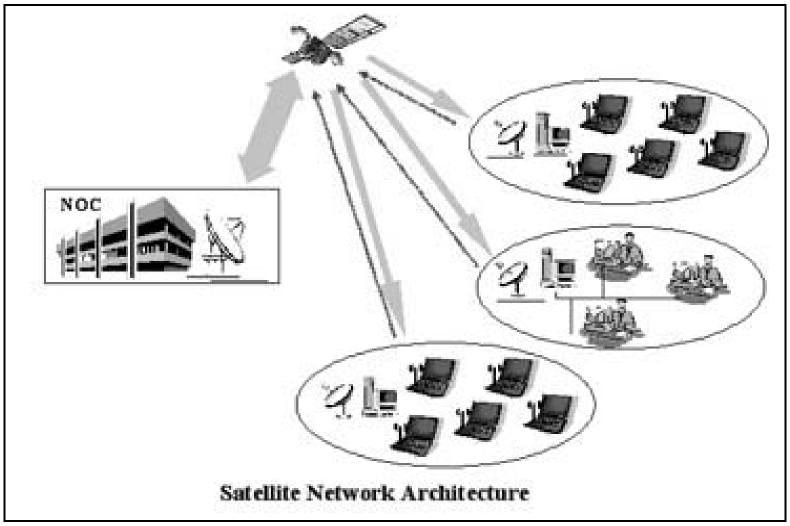


Figure 4.1: The Satellite Network Architecture

The architecture has a group of networks geographically separated and spread over a wide area. They constitute the “subnetworks” in the overall network. The subnetworks are connected to each other by satellite links using a geostationary satellite. The subnetworks are Ethernet-based, while the satellite links are ATM-based. The satellite is an ATM switch with no support for IP. There is a network operations center (NOC) from which the operation of the satellite is controlled, through a dedicated connection. The geostationary satellite links involve high delay, of the order of 250ms in a single-hop (for example, Spaceway [12]). The uplink bandwidth is also constrained to approximately 1.54 Mbps. These are important considerations when we design the multicast routing framework in section 4.2.

Each subnetwork connects to the satellite using one or more satellite gateways or satellite terminals. The network architecture forms a natural hierarchy. The logical grouping of the gateways connected by the satellite links form an overlay that interconnects the terrestrial subnetworks. The hosts in each subnetwork form a “lower level”, while the overlay can be looked upon as a higher level. Figure 4.2 gives a schematic of the logical grouping.

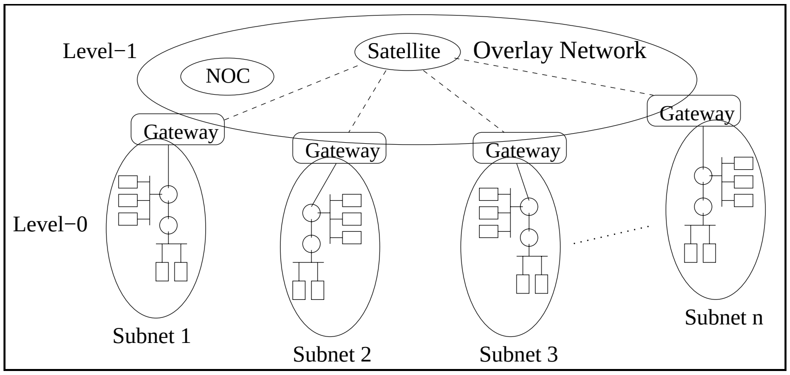


Figure 4.2: Logical Grouping in the Satellite Network Architecture

4.2 IP/ATM Multicast Routing Framework

The network architecture described in section 4.1 can be considered to be composed of terrestrial domains (the subnetworks) interconnected by satellite links. Therefore, the design of a framework for IP multicasting routing for this network involves two components:

- “Traditional” IP multicast routing in each Ethernet-based subnetwork. This is similar to the intra-domain IP multicast routing. Therefore it involves the selection of a suitable IP multicast routing protocol.

- IP multicast over ATM for inter-domain multicast routing. This requires the design of a suitable mechanism to multicast IP over the ATM-based satellite links.

4.2.1 Selection of Intra-domain Multicast Routing Protocol

The selection of a suitable IP multicast protocol for efficient and scalable intra-domain multicast routing within each subnetwork depends on the multicast group size and the dynamics of member joins and leaves. The terrestrial networks that we consider can be large with the members of a multicast group widely dispersed in each subnetwork. At the same time, the total number of group members in each subnetwork can be high, though a fraction of the total hosts in the subnet. We can therefore term the group as "sparse". PIM-SM has been proposed as a candidate protocol for multicast routing in sparse networks. Although PIM-SM is a complex multicast routing protocol, it has several features that make it attractive:

- It can efficiently manage a multicast group with low control overhead.
- It allows fast receiver joins to a multicast group due to the presence of the shared tree.
- Initial source transmission is also rapid and has low overhead due to the register mechanism.
- PIM-SM ensures low end-to-end latency for sources that require it by using source-specific trees.
- It can scale well if the number of group members increase.

We therefore select PIM-SM as the protocol for inter-domain multicast routing.

4.2.2 Selection of Inter-domain Multicast Routing Protocol

The inter-domain multicast in our network architecture involves sending IP packets over ATM connections. Our inter-domain architecture is a “one-hop” ATM network, with one switch (the satellite) that can reach all the nodes (the satellite gateways) simultaneously in a single broadcast.

None of the inter-domain protocols discussed in chapter 2 take into consideration the unique characteristics of the satellite medium. We wish to minimize the amount of control and data traffic that flow over the satellite links due to their high latency and constrained uplink bandwidth. BGMP, which is the popular inter-domain protocol, would create point-to-point TCP connections between the satellite gateways (BGMP peers). The root domain for every class D group will need to be one of the subnetworks; this therefore will mean unnecessary retransmissions - once to the root domain, and then from the root domain to all other domains, via the same overlay network. Also, since there will be point-to-point TCP connections between BGMP peers, the traffic will need to be replicated multiple times from the source border router to the receivers, which is a wasteful use of the satellite broadcast medium. The other inter-domain protocols also suffer from similar drawbacks when applied *as is* to our overlay network.

However, the VC mesh and MCS architectures can be well applied to the overlay network. The MCS architecture is ideally suited - the satellite can be the MCS, with each source sending only one copy of each cell on the uplink, which the satellite replicates and broadcasts using a point-to-multipoint VC to the receivers. However, the MCS architecture suffers from several drawbacks when applied to the network:

1. The network will have only one physical node that can act as the MCS. A single MCS can serve only one IP multicast group at a time, as it has no way to differentiate between traffic destined for different groups. The single MCS can be extended to serve multiple groups by creating multiple logical instances of the MCS, each with different ATM addresses (e.g. a different SEL value in the node's NSAPA [38]). But the SEL field is only 8 bits, therefore there can be at most 256 groups. This is a limitation for scalability that should be avoided.

2. To support even one group that can have multiple sources, the MCS needs to be able to do segmentation and re-assembly for every cell it receives, since AAL5 does not support cell level multiplexing of different AAL_SDUs on a single outgoing VC. This involves higher latency. Also, we assume that the satellite has very limited switching functionality, and does not do any extended processing.

3. A slightly more complex approach to support multiple groups using a single MCS would be to add minimal network layer processing into the MCS. This would require that every cell is re-assembled into the original IP multicast packet, the MCS checks the group address in each packet, and then the packet is again segmented into cells and sent out on the appropriate point-to-multipoint VC for the group. This will involve significantly higher latency due to the processing required, and necessitate sizeable buffers at the satellite, especially

when the sources have high data rate. Also, the processing at the MCS will be complex and will require it to support an IP stack. No satellite to date has support for IP processing in it, and we make no assumption to that effect.

Based on the above reasons, we do not design our framework using the MCS architecture for routing in the overlay. Instead, we select the VC mesh architecture. Although the VC mesh has higher resource consumption in comparison to the MCS, it is more scalable, has higher expected throughput and lower end-to-end latency (since the mesh lacks the intermediate AAL_SDU reassembly that must occur in MCSs), and makes no additional demand on the capabilities of the satellite, except that it be an ATM switch that supports UNI 3.0/3.1 signaling.

We describe in detail our framework in section 4.2.3. The framework is based on the technical description of PIM-SM and its message formats provided in [26], and on the description of ATM support for IP multicast and the signaling mechanism and message formats that are detailed in [38].

4.2.3 Description of the Multicast Routing Framework

4.2.3.1 IP Multicast Framework in each Subnet

- Each subnetwork is a PIM-SM domain and runs standard PIM-SM multicast protocol in the routers.

- Routers directly connected to the end hosts also run standard IGMP.

- One or more satellite terminals in a subnetwork are configured to act as Rendezvous Points (RPs) for all the multicast groups in the subnetwork. We term the subnet RPs the “local” RPs. The local RPs create the shared multicast tree for the multicast groups in their subnet.

- A router in each subnetwork is configured to act as the bootstrap router (BSR) for the subnetwork. Every subnetwork therefore has its own BSR.

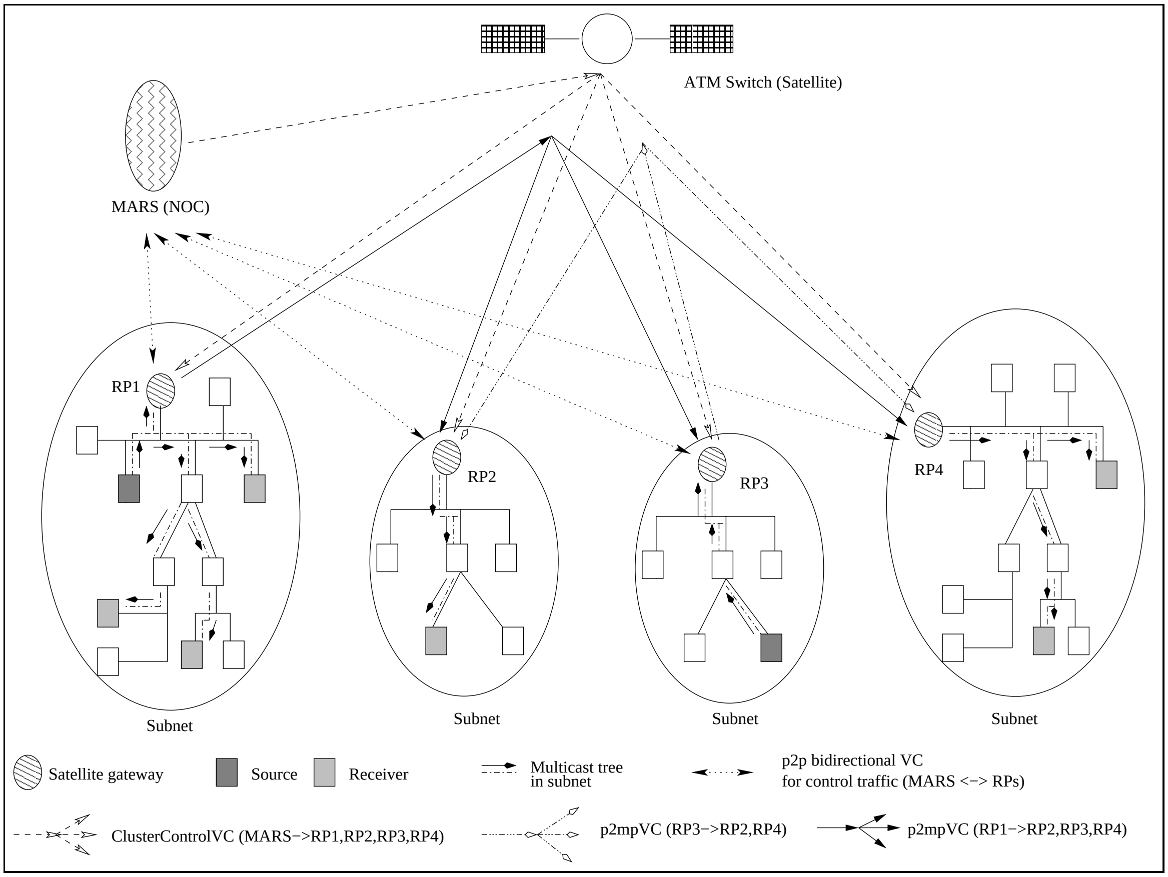


Figure 4.3: The IP/ATM Multicast Framework

4.2.3.2 ATM Multicast Framework over the Satellite Links

To facilitate the exchange of IP multicast data between subnetworks, we make use of the MARS with VC mesh architecture. The IP packets are carried as ATM cells over the point-to-multipoint virtual connections between the senders' RPs and receivers' RPs[1]. The framework is detailed below.

- A Multicast Address Resolution Server (MARS) is used to maintain a mapping of IP multicast addresses to ATM addresses. We define the MARS in our architecture to be located at the NOC.

- The satellite terminals have ATM interfaces with unique ATM addresses. These terminals are the ATM endpoints at the ATM level in the overlay network. The ATM interfaces of the satellite terminals together form an ATM cluster that is managed by the MARS. The ATM address of the MARS is known to all the ATM endpoints in the ATM cluster.

- All ATM connections go over the ATM switch located at the satellite.

- Many-to-many multicast is done over the ATM "cloud" using multiple point-to-multipoint VCs from each source RP to the set of receiver RPs per multicast group. This therefore implements the VC mesh architecture proposed

[1] The RP of a subnetwork that has the source is termed "sender RP" or "source RP", whereas the RP of the subnetworks that have the receivers are termed "receiver RPs". An RP might be both a source RP and a sender RP, and there can be multiple in each category for the same group.

in [38]. Multiple senders to the same multicast group, located in the same subnet, will share one point-to-multipoint VC to reach receivers in other subnets. Senders for different groups in the same subnet will use different point-to-multipoint VCs.

- Each receiver RP will terminate one branch of a point-to-multipoint VC for every external source RP to the group. If there are receivers for multiple groups in the subnetwork, the receiver RP will terminate branches of separate point-to-multipoint VCs per group and per external source RP.
- All satellite terminals that are configured to act as RPs, register their ATM addresses with the MARS on startup, following the procedure defined in [38]. A point-to-multipoint VC exists from the MARS to all the registered ATM endpoints in the subnets - this is the ClusterControlVC (CCVC) which is used by the MARS to advertise changes to group membership for all groups.

The multicast framework is given in figure 4.3. With the above framework, the operation of a multicast group is detailed in the following sections.

4.2.3.3 Creation of a Multicast Group When a Source Becomes Active

When a host in a subnetwork wants to send data to a multicast group that previously did not exist, the chain of events is as follows (refer to figure 4.4).

1. The source (host A) in subnet 1 sends the data to be multicast to its designated router (DR) for forwarding to the multicast group X.

2. The DR computes the (local) RP in subnet 1 for the multicast group X and unicasts a REGISTER message (encapsulated data packet) to the RP.

3. The RP de-capsulates the data packet and creates $(*, G)$ entry for group X in its multicast routing table.

4. The REGISTER message for the new group triggers the IP module at the RP to send a request to its ATM module to query the list of receivers for the group in other subnets.

5. The ATM module at the source RP sends a MARS_REQUEST message to the MARS.

6. The MARS, on receiving the request from its MARS client, searches the local database for the mapping ⟨IP_multicast_group, list of ATM endpoint addresses⟩. Since the group is new, no prior mapping exists in the MARS database. MARS therefore creates an entry for the multicast group in its address mapping table (and adds the ATM address of the source RP to the table entry for the group). MARS then sends a MARS_NAK to the source RP (or a MARS_MULTI message with the requesting ATM endpoint address as the only member address).

7. On receiving the MARS_NAK, the source ATM module waits a pre-determined delay period before sending a new MARS_REQUEST to the MARS.

8. When a host B in subnet 2 wants to receive data from group X, its DR sends a PIM JOIN$(*, X)$ message to the local RP for group X.

9. The RP in subnet 2 checks that it is not part of the multicast tree for group X. It therefore creates $(*, G)$ state for group X. It also triggers the IP module at the RP to send a request to its ATM module to register with the MARS for receiving external traffic for group X.

10. The ATM module, on receiving the request from the IP module, sends a MARS_JOIN message to the MARS for group X.

11. The MARS adds the ATM address of subnet 2 RP to the list of endpoints for group X.

12. The MARS_JOIN message is propagated by the MARS over the CCVC to all registered ATM endpoints. Thus the RP in subnet 1 is updated about the change in the group membership.

    This leads to some inefficiency since all endpoints will get the membership update information, but the information is useful only to the source RPs. We therefore propose that the MARS maintain a separate point-to-multipoint VC to only the source RPs, and inform them of changes to the group membership using MARS_MULTI message format. This would require additional database storage at the MARS to differentiate between the source RPs and the receiver RPs.

13. The ATM interface of the RP in subnet 1 gets the addresses of the receiver ATM endpoints from the the MARS_JOIN message. It then creates a point-to-multipoint VC over the satellite ATM switch to the set of ATM

endpoints following standard procedure as given in [38].

The ATM module at the source RP also sends a message to its IP module to inform the RP of the presence of receivers outside the subnet. The IP-ATM interface is therefore added to the outgoing interface (*oif*) list for the multicast group X in the local multicast tables.

14. Data flows in native IP format along the shared RP tree in subnet 1. The packets are received by the IP-ATM interface at the source RP, where they are segmented into ATM cells and multicast to the receiver RPs over the satellite point-to-multipoint VC.

15. The ATM cells are received by the IP-ATM interface of the RP in subnet 2, where they are reassembled into the corresponding IP packet and forwarded to the IP module. The IP module forwards the packet to the PIM-SM module based on the multicast destination address. PIM-SM adds the IP-ATM interface to the incoming interface list (iif list) for the multicast group, and forwards the packet on the outgoing interfaces (based on the oif list) to the receivers along the shared tree rooted at the RP in subnet 2. The IP multicast tree is thus set up spanning multiple subnets.

4.2.3.4 Source Join to an Existing Multicast Group

With reference to figure 4.5, host M in subnet 2 wishes to send data to multicast group X. Group X has sender A in subnet 1, and receivers in subnets 1, 2 3 and 4.

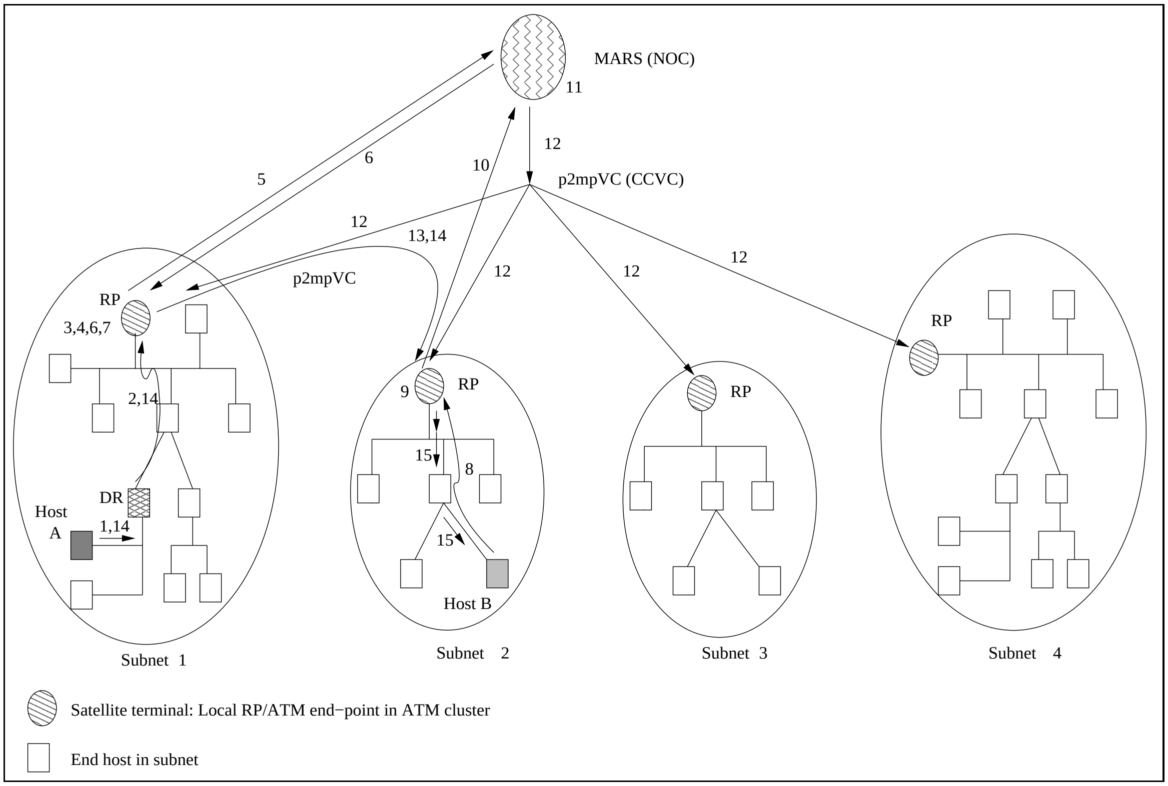


Figure 4.4: Creation of One Multicast Group Across Subnets

1. The DR of M sends the encapsulated data packet in a PIM REGISTER message to the RP for X in subnet 2 (RP2).

2. RP2 checks its IP multicast routing tables and finds that entry for group X is present, but there are no local sources[2]. The RP forwards the data along the shared RP tree in subnet 2. The REGISTER message also triggers the IP module to send a request to the local ATM module to query the MARS for the list of subnets who are receivers for data for group X. The ATM module hence sends a

[2] This can be done by checking the incoming interface (*iif*) list at the RP. It will contain only the IP-ATM interface, indicating that the current sources are external to the subnet.

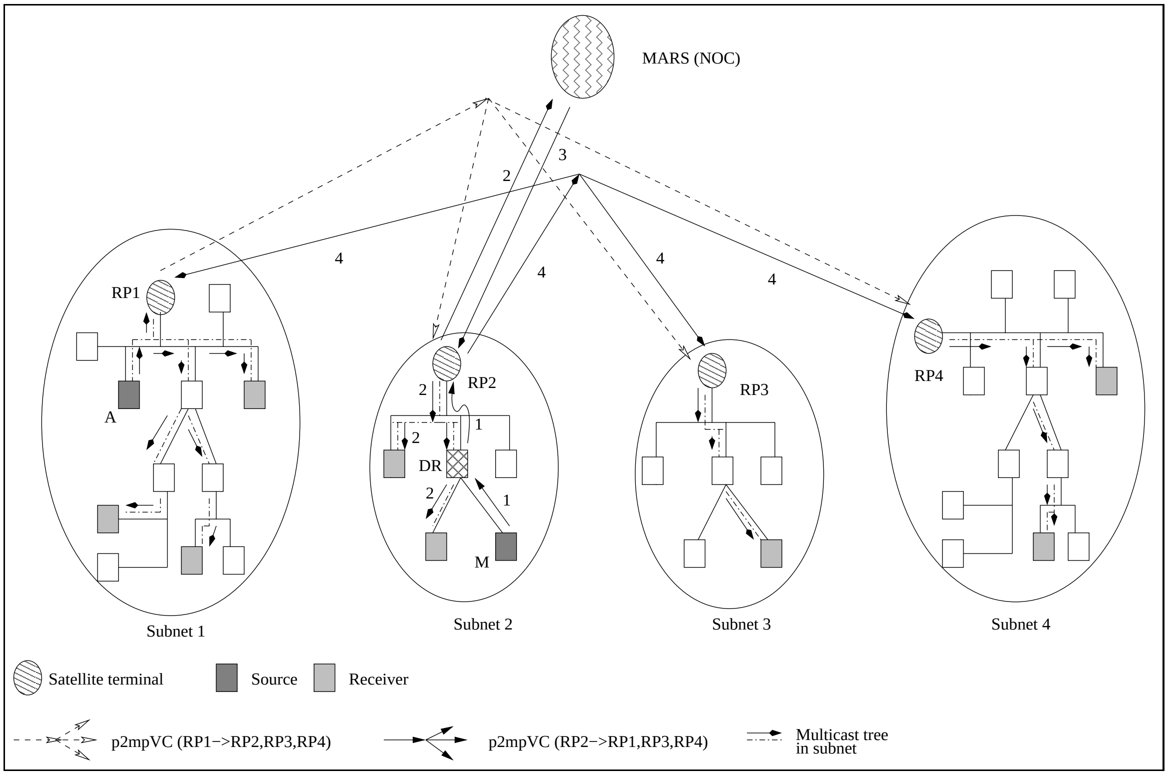


Figure 4.5: Source Join to Existing Multicast Group

MARS_REQUEST message to the MARS.

3. The MARS receives the MARS_REQUEST and responds with a MARS_MULTI message containing the list of ATM addresses for the endpoints of group X.

4. The ATM module in RP2 extracts the addresses of the endpoints for group X and creates a point-to-multipoint VC to all the endpoints over the satellite links. The IP module in RP2 is also informed of the presence of receivers outside the subnet. The IP-ATM interface is therefore added to the list of outgoing interfaces in the IP multicast state entry for group X in RP2.

Therefore there exists two point-to-multipoint VCs for group X, one for source A in subnet 1, and the other for source M in subnet 2. More point-to-multipoint VCs are set up if new sources in other subnets send to group X, thereby creating a VC mesh. However, multiple sources for group X in the same subnet will send data over one shared point-to-multipoint VC to receivers in other subnets.

4.2.3.5 Receiver Join to a Multicast Group

With reference to figure 4.6, assume host P in subnet 3 wants to receive data of group X, and it is the first receiver registering for group X in subnet 3.

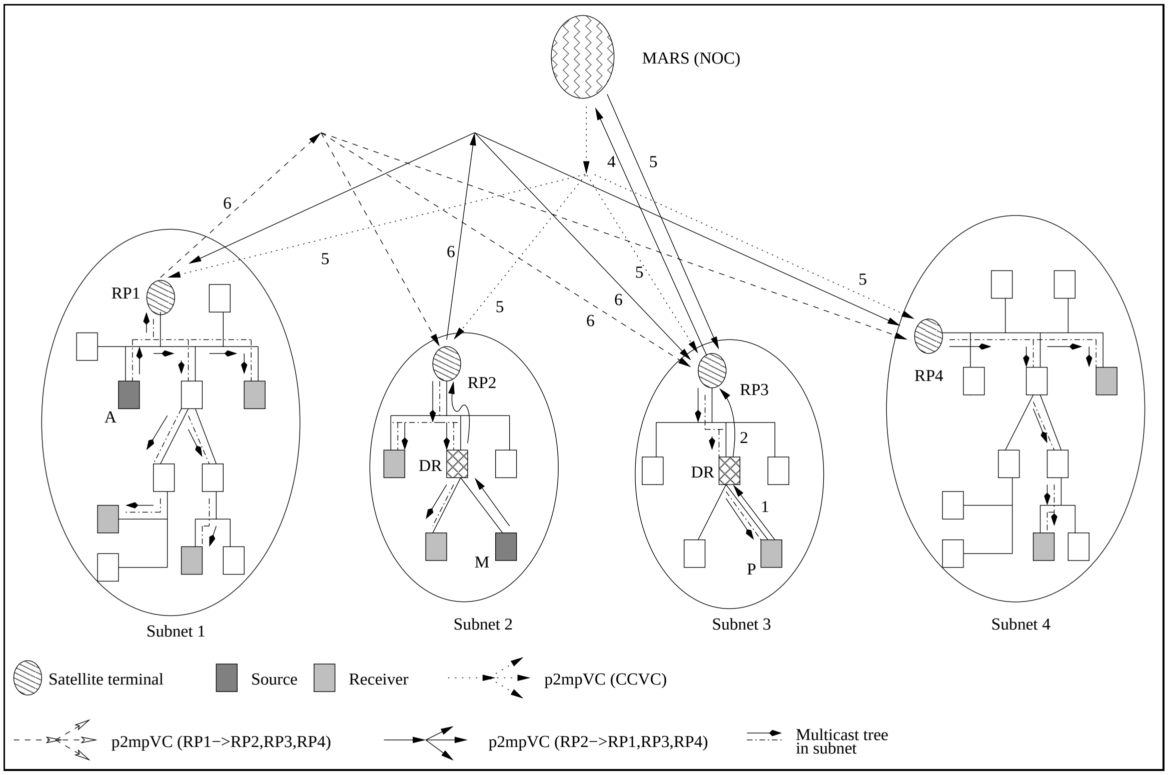


Figure 4.6: Receiver Join to Existing Multicast Group

1. Host P informs the DR of its intention to receive data of group X using IGMP.

2. The DR for P sends a $(*, X)$ JOIN towards the RP for group X in subnet 3 (RP3).

3. The JOIN message propagates hop-by-hop over the links in the subnet, setting up $(*, G)$ state for group X in each PIM router it passes through.

4. The JOIN message reaches RP3. RP3 checks its routing tables and finds no entry for group X. It creates a $(*, G)$ entry for X. The JOIN message also triggers the IP module in RP3 to signal the local ATM module for sending a join request to the MARS. The ATM module of RP3 therefore sends a MARS_JOIN request to the MARS.

5. MARS receives the MARS_JOIN from RP3. It adds the ATM address of RP3 to the list of endpoints for group X, and sends a MARS_MULTI message to the list of senders for group X using the point-to-multipoint VC as specified in section 4.2.3.3. The MARS also acknowledges the MARS_JOIN request from RP3 as specified in [38].

6. Each source RP receives the message from the MARS containing the updated list of endpoints for group X. Each source RP subsequently adds a branch to the point-to-multipoint VC it maintains for group X, with the branch being terminated at the ATM module of RP3. The multicast tree for X is thus extended into subnet 3.

Adding a new receiver to the multicast tree when it already exists (in the receiver's subnet) is done as in PIM-SM. Here there is no need for the RP to send any JOIN request to the MARS, since it is already a part of the multicast tree for that group.

4.2.3.6 Source Leave from a Multicast Group with One Source

Let host A in subnet 1 is the only source for multicast group X that has receivers in subnets 1, 2 and 3.

1. When host A wants to leave the multicast group X, it stops transmitting data with the class D address of X as destination.

2. The timers for group X in each PIM-SM router in the subnets time out and the multicast state for group X is removed from router memory.

3. Inactivity timers are also associated with the point-to-multipoint VC for group X rooted at the ATM module of the RP in subnet 1. Upon expiry of this timer, the point-to-multipoint VC is torn down.

4.2.3.7 Source Leave when Multiple Sources are Present

Let host A in subnet 1 and host M in subnet 2 be the two sources for multicast group X that has receivers in subnets 1, 2 and 3. When host A wants to leave the group X, the sequence of actions is identical to that outlined in 4.2.3.6.

In the case that there are two sources, A and B, for group X in subnet 1, the sequence of actions is different when only one host (for example, A), wants to leave. The differences are highlighted below.

1. Host A stops transmitting data to group X.

2. If source-specific tree (SPT) for A exists in subnet 1, the timers associated with the SPT for A time out and the SPT is torn down.

3. The shared tree for group X remains active since source B is active. Also, the SPT rooted at B remains active, if present.

4. The point-to-multipoint VC from subnet 1 to the other subnets for group X also remain active since the source B is active.

4.2.3.8 Receiver Leave from a Multicast Group

If there are multiple receivers present in a subnet for a multicast group X, and a subset of the receivers in the subnet decide to leave the group, then all the changes to the multicast tree are localized within the subnet, and follow standard PIM-SM procedure for pruning the multicast tree. The ATM multicast VC mesh between the subnets does not change.

However, if all the receivers in a subnet decide to leave, then the sequence of actions is different at the ATM level. For ease of description, we consider that there is only one receiver, host P in subnet 3, who decides to leave group X. The events are as follows.

1. Host P informs its DR about leaving group X, using IGMP report message.

2. The DR sends a PRUNE$(*, G)$ for group X towards the RP.

3. At each intermediate PIM router through which the PRUNE message passes, the

corresponding outgoing interface is removed for group X. Since the oif list for group X becomes empty, the multicast state for X is removed from router memory.

4. Eventually the PRUNE(*,G) reaches the RP (assuming the RP was in the path of the multicast tree to P). The RP removes the interface towards P from its list of outgoing interfaces. The oif list thus becomes NULL. This triggers the IP module at the RP to send a message to its ATM module.

5. The ATM module sends a MARS_LEAVE message to the MARS.

6. The MARS removes the receiver RP ATM address from the group information in its database. It then sends a MARS_MULTI message with the updated group membership information, to the sources for group X using the point-to-multipoint VC as specified in section 4.2.3.3.

7. The source RPs for X remove the connection to RP3 from the point-to-multipoint VC that each maintains for group X.

If sources for group X are also present in subnet 3, then the oif list at RP 3 for group X does not become NULL when P leaves, since the ATM interface is there in the oif list (for receivers in other subnets for the local source). The RP will therefore need to distinguish between the receivers who are local its subnet, and receivers who are elsewhere. Hence a MARS_LEAVE message will be triggered when the *oif list for local receivers* becomes NULL.

4.3 Issues with the Multicast Framework

Our design for IP multicast over satellite aims to maintain separation between the IP multicast in subnetworks, and the IP-over-ATM multicast between subnetworks. However, the following issues arise due to the framework:

- Since the interaction between the NOC and the satellite terminals is at the ATM level, and involves the ATM addresses of the RPs (satellite terminals) only, the NOC does not get to know the IP addresses of the actual senders and receivers. But the NOC gets the addresses of the subnets which are sending and receiving to a given multicast group (this it gets due to the MARS messages).

- We assume that the BSR in each subnet independently advertises the list of RPs for different groups in respective domains. There is no synchronization between the BSRs. Consequently it is always possible that receivers send JOIN requests to groups for which there exist no senders in any subnet. We rely on the PIM-SM timer mechanism to delete the state for such groups from the PIM routers whenever such state is created.

- In our framework, it might be possible that the ATM interface is present both in the iif list (when there are sources in other subnets and local receivers) and also in the oif list (when there are local sources and remote receivers in other subnets). This is a valid state in our framework, and PIM-SM should not construe this as the existence of a loop.

Chapter 5

Routing Framework Simulation and Results

We have verified the validity and feasibility of our framework through simulations using Opnet Modeler, version 9.0[39]. The Opnet Modeler version 9.0 has support for PIM-SM, but it does not support ATM multicast. There is also no support for ATM point-to-multipoint connection.

5.1 Implementation Issues

We implemented the basic MARS architecture with VC mesh in the Opnet Modeler 9.0. The implementation issues in our framework design is discussed below.

1. PIM-SM - changes to RP Functionality: Our architecture requires modifications to the RP design in PIM-SM. The following are the important RP modifications:

    - RP action on receiving REGISTER, JOIN or LEAVE messages - this will trigger the IP module at the RP to signal the local ATM module for sending MARS messages.

- RP action on receiving REGISTER message - the RP has to keep track whether there are other sources present for the multicast group in its subnet.
- Addition to the outgoing interface list at the RP if there are local sources - add the IP-ATM interface to the oif list if there are receivers in other subnets.
- RP action on JOIN message - trigger a JOIN to the MARS if $(*, G)$ state does not exist.
- Additional RP action for PRUNE message - check the local oif list and trigger a LEAVE to the MARS if: (i) local oif list is empty; (ii) iif list includes the IP-ATM interface (which indicates the RP is a leaf on a point-to-multipoint VC for existing sources in other subnets).

2. Interaction between ATM and PIM-SM: The interaction between PIM-SM and ATM will occur for the following events:

    - REGISTER message for initial source (when $(*, G)$ state does not exist at RP).
    - JOIN message for initial receiver (when $(*, G)$ state does not exist at RP).
    - PRUNE message for last receiver (when IP-ATM interface is on the iif list for the group).
    - Signal from ATM interface to PIM-SM when a point-to-multipoint VC is created rooted at the ATM endpoint (for local sources). The signal will

make the RP add the ATM interface to the oif list.

- Signal from ATM module to PIM-SM when a point-to-multipoint VC is terminated at the local ATM module (for local receivers and external sources). RP will add the IP-ATM interface to the iif list.

5.2 Simulation Configuration

- In the network configuration for the simulation, there are 15 network domains spread over a region the size of the continental US; the domains are connected by a geostationary satellite. The MARS is located at the NOC.

  There are 50 nodes in each subnetwork, making a total of 750 nodes in the network. Each domain has one satellite gateway that acts as the PIM-SM RP for the multicast groups in its domain.

- Rendezvous Point: The RP is modeled by a ATM/Ethernet gateway router that has both ATM and Ethernet interfaces. We implemented the support module for MARS functionality at the IP Adaptation Layer, which is the interface between the IP layer and the ATM AAL layer in the node model.

- MARS: We selected an ATM server for simulating the MARS. We made modifications to the IP Adaptation Layer in the ATM server node model to include the support module for MARS functionality.

- Satellite Switch: For the satellite ATM switch, we selected a gateway router (similar to the RP) and modified the node model to allow the router to switch packets (segmented into ATM cells) between incoming and outgoing VCs at the IP Adaptation Layer, without sending the packets to the IP layer, i.e., without doing any IP routing. This is done to implement the point-to-multipoint VC functionality, which is not supported by the Opnet modeler v9.0.

- Multicast Router: For the Designated Routers in each domain, and other on-tree multicast-enabled routers, we selected Ethernet gateway routers. No modifications were made to the node model provided by Opnet.

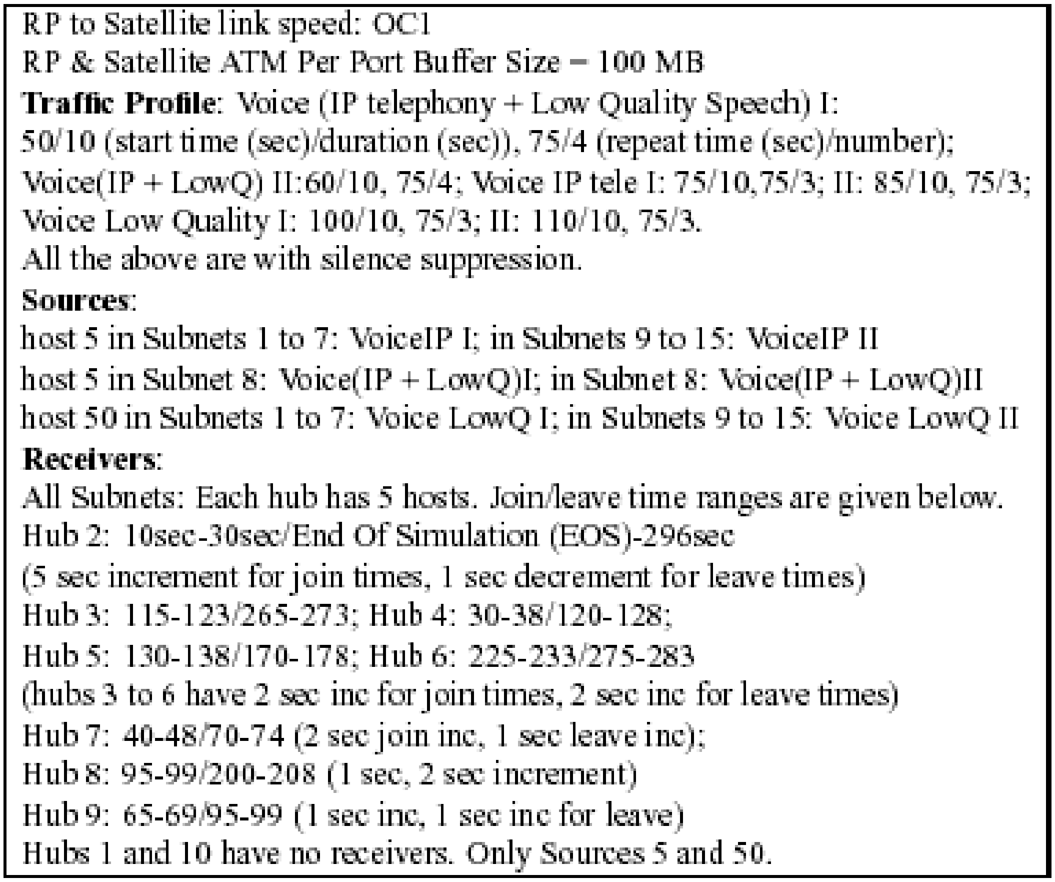
- End-host: The end-hosts in each subnetwork are advanced Ethernet workstation models; no modifications have been made to the node model provided by Opnet.

- The terrestrial links in each domain network are standard Ethernet links; the speeds range from 100BaseT for the connection from end-hosts to the leaf routers, and 10Gbps for connections between the routers. The satellite links are ATM links, with link delay of 0.13 seconds. We selected DS1 speed for the uplink, and OC1 for the downlink. Since we are concerned with best-effort IP multicast routing only, the channel errors are not considered.

5.3 Simulation Results

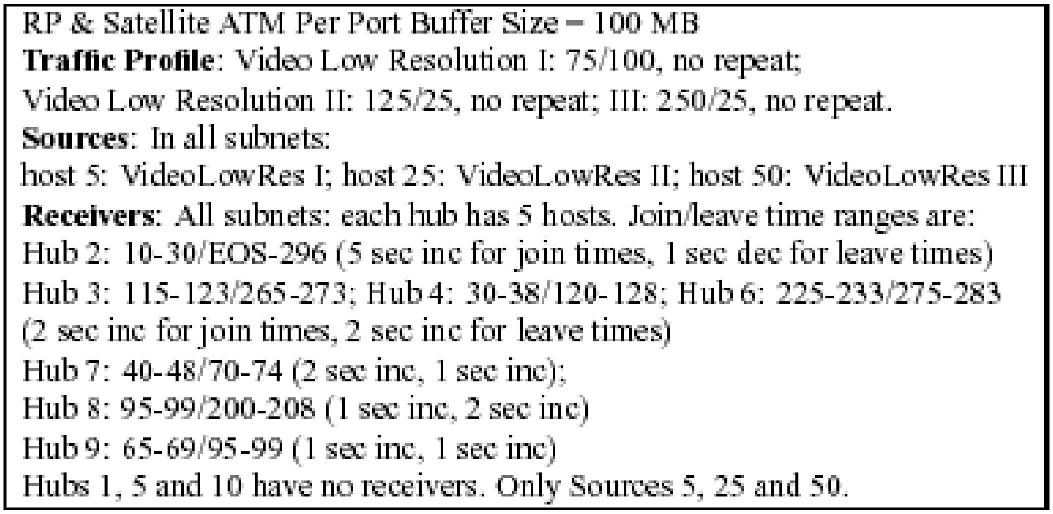
5.3.1 Many-to-Many Scenario Results

Simulations have been run for both many-to-many multicast, and one-to-many multicast, with each simulation being run for 300 seconds. For many-to-many multicast, simulations were done separately for voice and video traffic. The scenario for many-to-many multicast with voice traffic is given in figure 5.1. The scenario for video traffic is given in figure 5.2. To compare the performance of the multicast



RP to Satellite link speed: OC1
RP & Satellite ATM Per Port Buffer Size = 100 MB
Traffic Profile: Voice (IP telephony + Low Quality Speech) I:
50/10 (start time (sec)/duration (sec)), 75/4 (repeat time (sec)/number);
Voice(IP + LowQ) II:60/10, 75/4; Voice IP tele I: 75/10,75/3; II: 85/10, 75/3;
Voice Low Quality I: 100/10, 75/3; II: 110/10, 75/3.
All the above are with silence suppression.
Sources:
host 5 in Subnets 1 to 7: VoiceIP I; in Subnets 9 to 15: VoiceIP II
host 5 in Subnet 8: Voice(IP + LowQ)I; in Subnet 8: Voice(IP + LowQ)II
host 50 in Subnets 1 to 7: Voice LowQ I; in Subnets 9 to 15: Voice LowQ II
Receivers:
All Subnets: Each hub has 5 hosts. Join/leave time ranges are given below.
Hub 2: 10sec-30sec/End Of Simulation (EOS)-296sec
(5 sec increment for join times, 1 sec decrement for leave times)
Hub 3: 115-123/265-273; Hub 4: 30-38/120-128;
Hub 5: 130-138/170-178; Hub 6: 225-233/275-283
(hubs 3 to 6 have 2 sec inc for join times, 2 sec inc for leave times)
Hub 7: 40-48/70-74 (2 sec join inc, 1 sec leave inc);
Hub 8: 95-99/200-208 (1 sec, 2 sec increment)
Hub 9: 65-69/95-99 (1 sec inc, 1 sec inc for leave)
Hubs 1 and 10 have no receivers. Only Sources 5 and 50.

Figure 5.1: Multicast Routing: Many-to-Many Simulation Scenario for Voice



RP & Satellite ATM Per Port Buffer Size = 100 MB
Traffic Profile: Video Low Resolution I: 75/100, no repeat;
Video Low Resolution II: 125/25, no repeat; III: 250/25, no repeat.
Sources: In all subnets:
host 5: VideoLowRes I; host 25: VideoLowRes II; host 50: VideoLowRes III
Receivers: All subnets: each hub has 5 hosts. Join/leave time ranges are:
Hub 2: 10-30/EOS-296 (5 sec inc for join times, 1 sec dec for leave times)
Hub 3: 115-123/265-273; Hub 4: 30-38/120-128; Hub 6: 225-233/275-283
(2 sec inc for join times, 2 sec inc for leave times)
Hub 7: 40-48/70-74 (2 sec inc, 1 sec inc);
Hub 8: 95-99/200-208 (1 sec inc, 2 sec inc)
Hub 9: 65-69/95-99 (1 sec inc, 1 sec inc)
Hubs 1, 5 and 10 have no receivers. Only Sources 5, 25 and 50.

Figure 5.2: Multicast Routing: Many-to-Many Simulation Scenario for Video

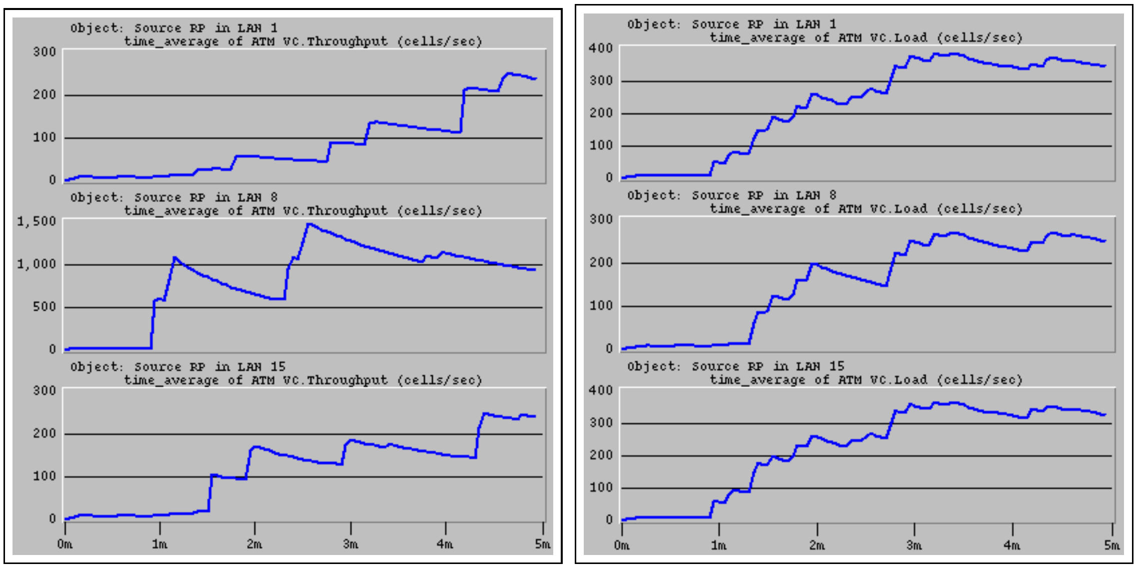
framework, we performed simulations of the above scenario using two more cases:

1. Default PIM-SM, with a single RP for a multicast group across all domains; the RP is located in one of the terrestrial subnetworks.

2. Default PIM-SM, with a single RP for a multicast group across all domains; the RP is located at the NOC.

The above scenarios are selected since the end-to-end multicast tree that we attempt to build in our framework can be done using default PIM-SM; the major issue then is the placement of the single RP, which is sub-optimal in both the above cases for our large network.

The results are given in figures 5.3 to 5.6. In all the graphs, the x-coordinate is the time of the simulation in minutes.



(a) Throughput on the uplinks

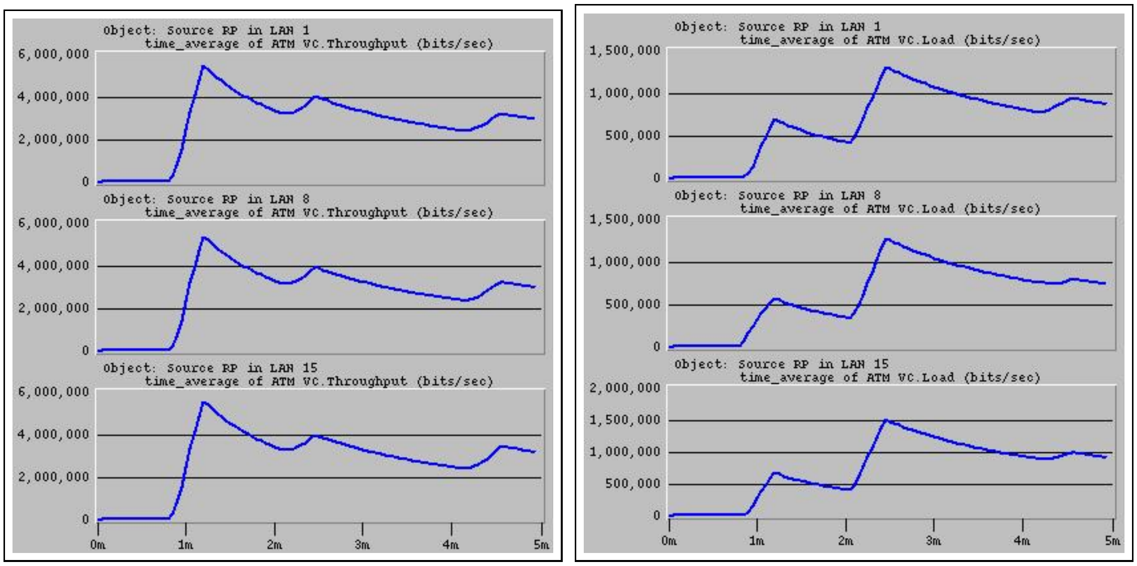
(b) VC load on the uplinks

Figure 5.3: Many-to-Many Multicast: Uplink Throughput and Load for Voice

The throughput and load obtained in the uplink VC for the three source RPs (in subnetworks 1, 8 and 15) are given in figures 5.3(a) and 5.3(b) respectively for voice traffic, and in figures 5.4(a) and 5.4(b) respectively for video traffic. Our concern is with the uplink, since the bandwidth is limited compared to the downlink (for example, 1.54Mbps compared to 92Mbps, respectively).

The total packet drop for our framework in comparison to using the default PIM-SM scenarios, is given in figure 5.5(a) for voice and figure 5.5(b) for video traffic[1].

[1] In all the comparison graphs, blue represents our framework, red is for the scenario where there is a single RP at the NOC, and green represents the scenario for a single RP in one subnetwork.



(a) Throughput on the uplinks

(b) VC load on the uplinks

Figure 5.4: Many-to-Many Multicast: Uplink Throughput and Load for Video

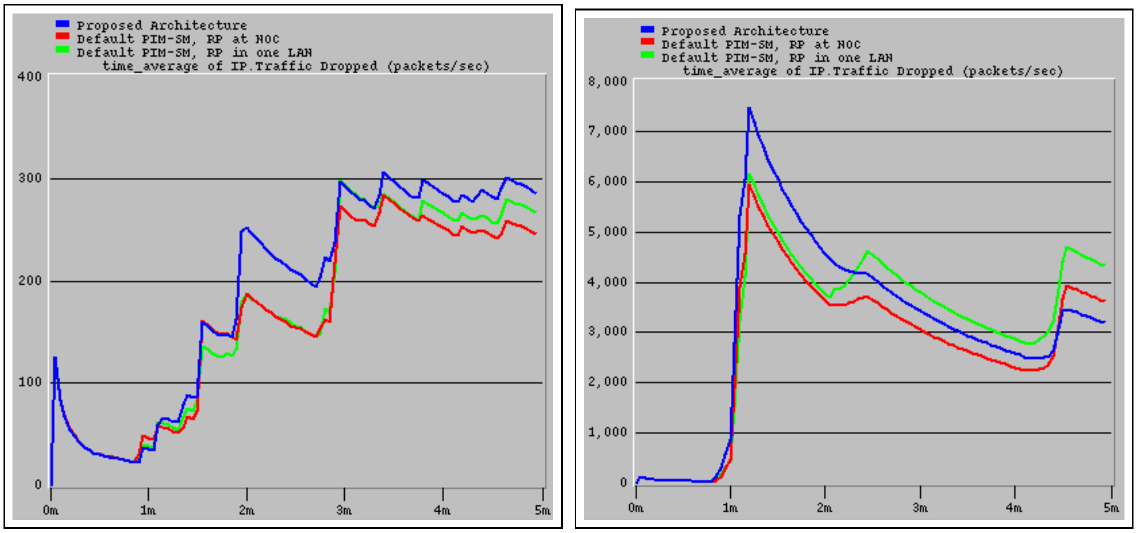
The end-to-end delay for voice and video applications are shown in figures 5.6(a) and 5.6(b) respectively. The perceived delay at the application is a very important criterion; our framework has less delay compared to the others, as the graphs show.

5.3.2 One-to-Many Scenario Results

We performed one-to-many simulations separately for voice and video traffic. The simulation scenario for voice traffic is detailed in figure 5.7.

The simulation scenario for video traffic is identical to voice traffic, except that the traffic type is Video Low Resolution I (75/100, no repeat), instead of voice.

To compare the performance of our multicast framework, we performed simulations of the above scenarios using the two default PIM-SM scenarios that are



(a) Voice Traffic

(b) Video Traffic

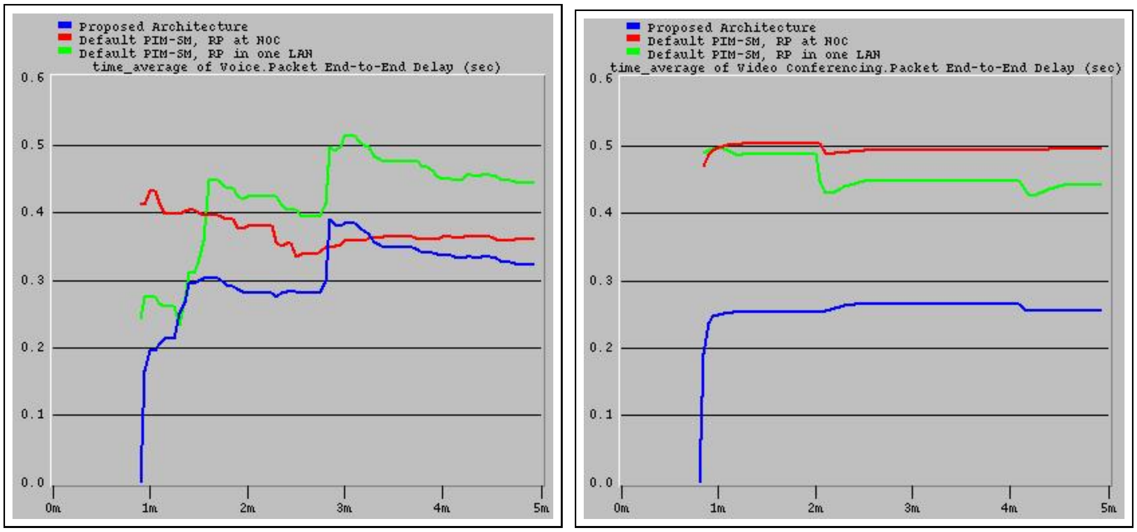
Figure 5.5: Many-to-Many Multicast: Total IP Packet Drop Comparison

described in section 5.3.1. The results are given in figures 5.8 to 5.12. In all the graphs, the x-coordinate represents the time of the simulation in minutes.

Figure 5.8 gives the profile of the traffic sent by the source, and the traffic received at selected group members, both in the subnet local to the source, and in remote subnets. The amount of traffic received by a host depends on the duration it is a member of the multicast group, hence some receivers get less than others.

The total IP packet drop for our framework in comparison to using default PIM-SM scenarios, are given in figures 5.9(a) and 5.9(b), for voice and video traffic, respectively.

When the IP multicast packets are segmented into ATM cells, they are assigned to Unspecified Bit Rate (UBR) service category. Figures 5.10(a) and 5.10(b) show the UBR cell loss ratio in the satellite links for the three scenarios, for voice and video



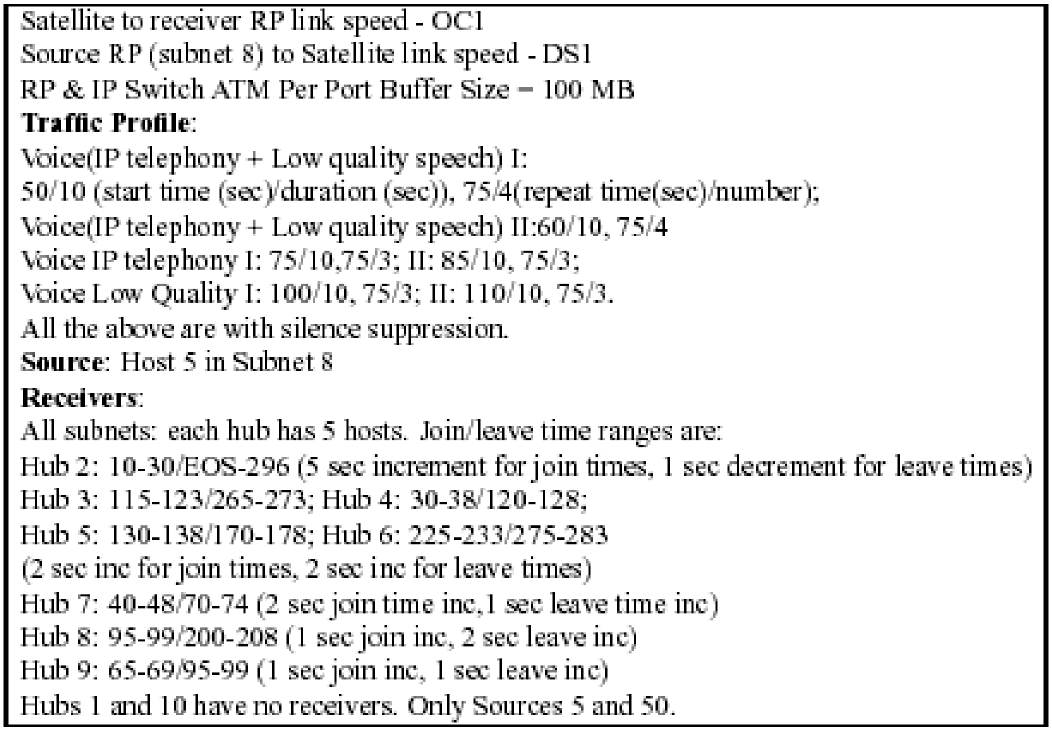
(a) End-to-end Delay for Voice

(b) End-to-end Delay for Video

Figure 5.6: Many-to-Many Multicast: Application Traffic End-to-end Delay

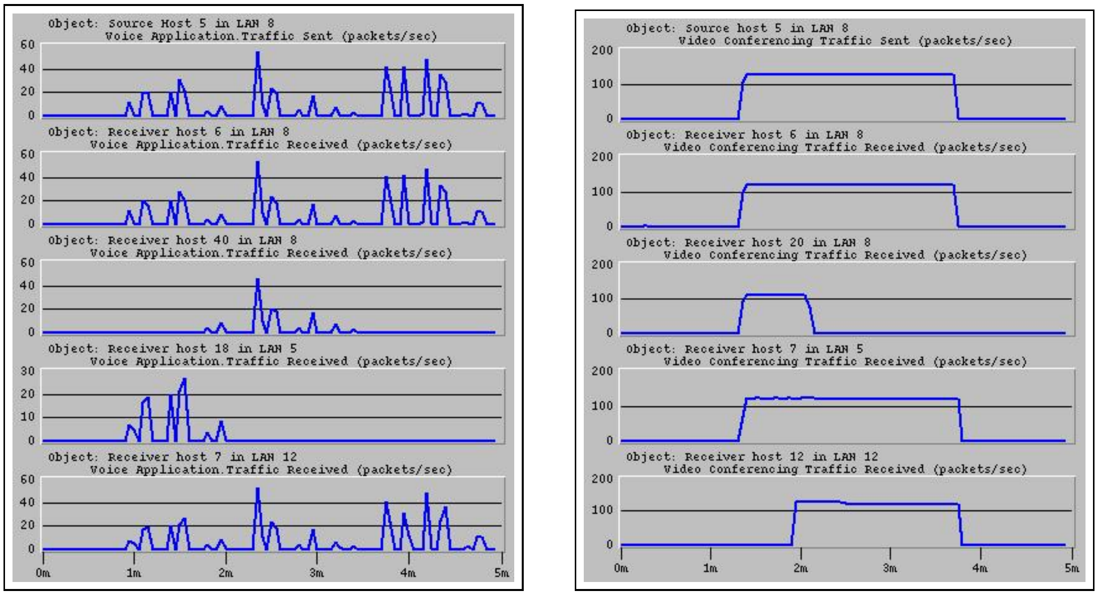
traffic respectively.

The packet end-to-end delay and the packet delay variation for voice application traffic are shown in figures 5.11(a) and 5.11(b) respectively, and in figures 5.12(a) and 5.12(b) respectively, for video traffic.



Satellite to receiver RP link speed - OC1
Source RP (subnet 8) to Satellite link speed - DS1
RP & IP Switch ATM Per Port Buffer Size = 100 MB
Traffic Profile:
Voice(IP telephony + Low quality speech) I:
50/10 (start time (sec)/duration (sec)), 75/4(repeat time(sec)/number);
Voice(IP telephony + Low quality speech) II:60/10, 75/4
Voice IP telephony I: 75/10,75/3; II: 85/10, 75/3;
Voice Low Quality I: 100/10, 75/3; II: 110/10, 75/3.
All the above are with silence suppression.
Source: Host 5 in Subnet 8
Receivers:
All subnets: each hub has 5 hosts. Join/leave time ranges are:
Hub 2: 10-30/EOS-296 (5 sec increment for join times, 1 sec decrement for leave times)
Hub 3: 115-123/265-273; Hub 4: 30-38/120-128;
Hub 5: 130-138/170-178; Hub 6: 225-233/275-283
(2 sec inc for join times, 2 sec inc for leave times)
Hub 7: 40-48/70-74 (2 sec join time inc,1 sec leave time inc)
Hub 8: 95-99/200-208 (1 sec join inc, 2 sec leave inc)
Hub 9: 65-69/95-99 (1 sec join inc, 1 sec leave inc)
Hubs 1 and 10 have no receivers. Only Sources 5 and 50.

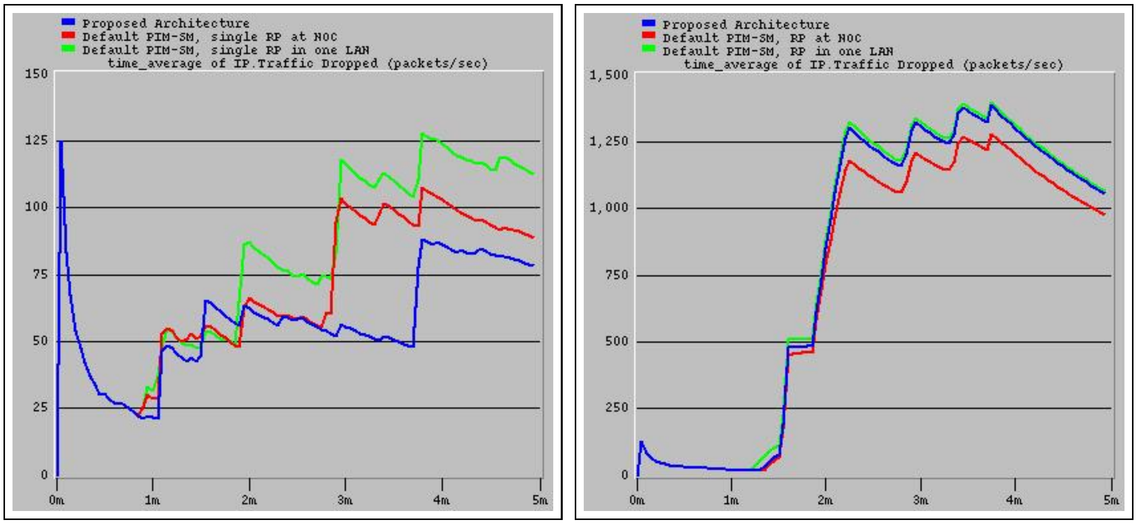
Figure 5.7: Multicast Routing: One-to-Many Simulation Scenario for Voice



(a) Voice Traffic

(b) Video Traffic

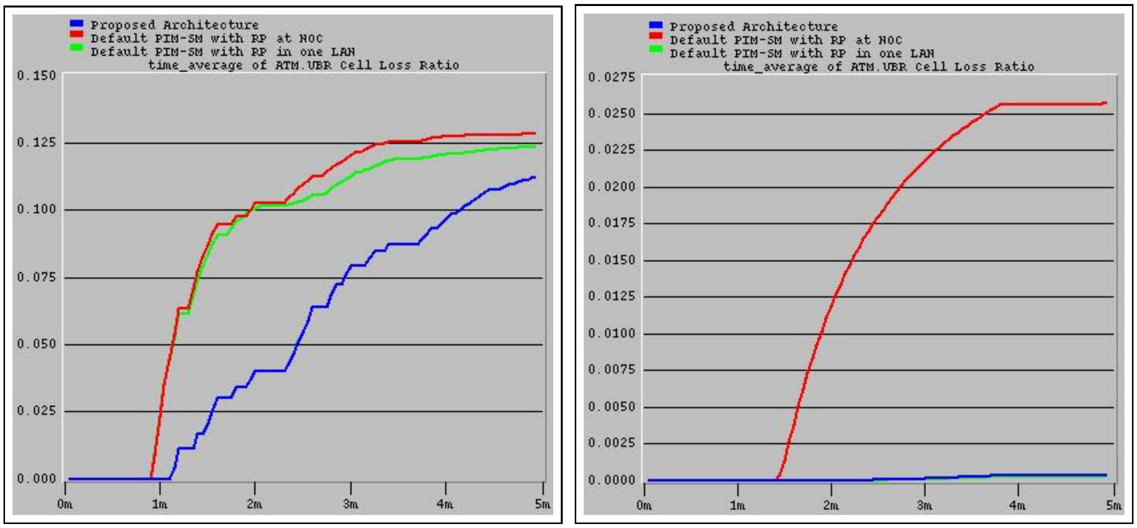
Figure 5.8: One-to-Many Multicast: Traffic Sent and Received



(a) IP Packet Drop for Voice

(b) IP Packet Drop for Video

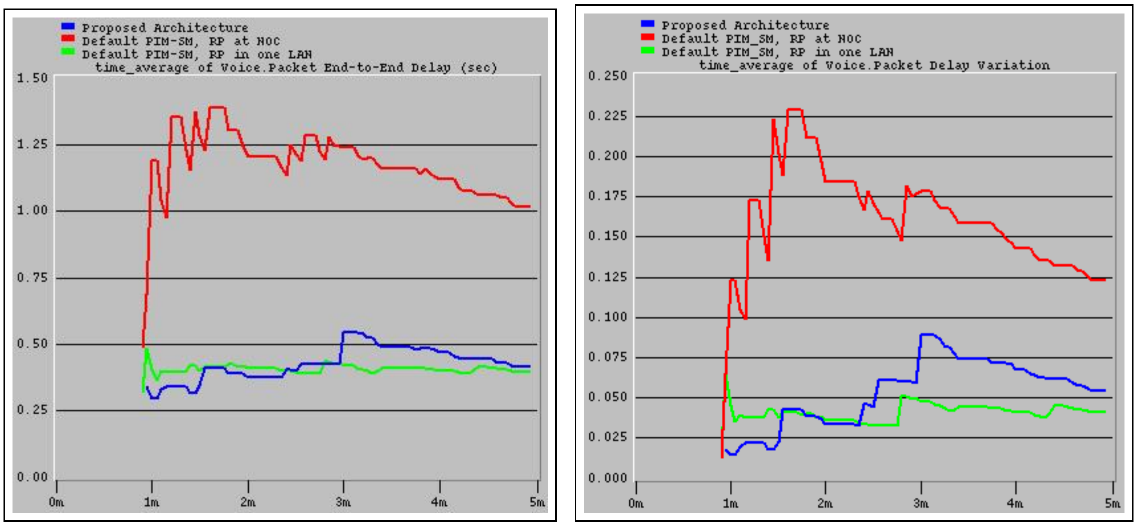
Figure 5.9: One-to-Many Multicast: Total IP Packet Drop Comparison



(a) UBR Cell Loss Ratio for Voice Traffic

(b) UBR Cell Loss Ratio for Video Traffic

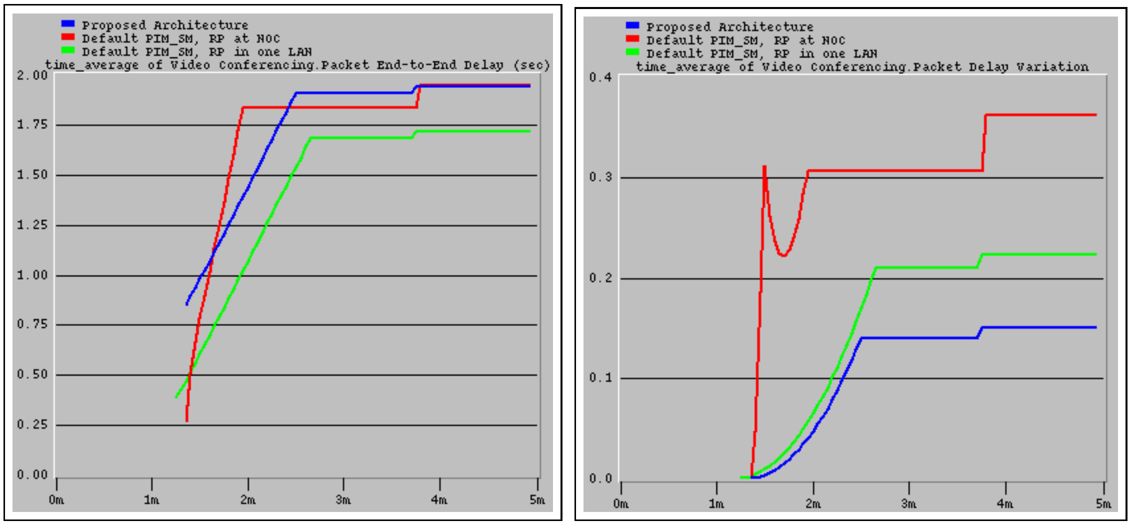
Figure 5.10: One-to-Many Multicast: UBR Cell Loss Ratio



(a) End-to-end Delay

(b) Packet Delay Variation

Figure 5.11: One-to-Many Multicast: End-to-end Delay for Voice Application



(a) End-to-end Delay

(b) Packet Delay Variation

Figure 5.12: One-to-Many Multicast: End-to-end Delay for Video Application

Chapter 6

Review of Group Key Management Protocols

The design of a secure multicast routing architecture requires design of both a routing framework and a scheme for secure data transfer. In the previous chapters, we have developed the routing framework. Our goal now is to design a scheme for secure data transfer in the network under consideration. Therefore, in the following chapters, we develop a framework for key management in the proposed routing framework. The key management framework is essential for the encryption of the multicast traffic, to ensure data confidentialtiy.

In this chapter we first review the basic concepts involved in group key management. We then describe and analyze some of the well-known group key management schemes that have been proposed in the literature.

6.1 Features of Group Key Management Systems

6.1.1 Security Requirements

The desirable security properties of a group key management system are:

- Group Key Confidentiality: All group keys should be known only to authorized group members; different members might know only subsets of the set of keys in the group. It should be computationally infeasible for a non-member to discover any group key.

- Backward Access Control: New users should not be able to decrypt past group communication using the group keys they receive upon joining the group.

- Forward Access Control: Users upon losing their group privileges should not be able to decrypt future group messages using the old group keys they have in their possession.

- Key Independence: A passive adversary who knows a proper subset of the group keys $\hat{K} \subset K$ cannot discover any other group key $\bar{K} \in (K - \hat{K})$.

6.1.2 Cost Metrics

The following resources are important when analyzing the overhead of key management systems:

- Communication cost: The propagation delay in sending key management information to the group members should be kept low to minimize the latency involved in the initial key generation and distribution phase, and in subsequent key updates. Delay is the most important criterion in time-sensitive applications like live video-conferencing and online stock market updates.

The bandwidth consumption in distributing key management information to the group members is also very important. The key management system should minimize the amount of information sent to the members, since bandwidth can be limited in many networks.

The number of message exchanges required to establish the group keys contribute to both the overall delay and bandwidth consumption, and therefore should be minimized in efficient schemes.

- Storage cost: If there is a group key controller, it has to store all the member-specific keys. For groups with large number of members, the storage requirements at the server can be quite large.

  Each group member will need some storage at the local node for its member-specific keys and the session key. Although the number of keys stored at the member is much less than at the controller, the member node might have limited storage capabilities, for example PDAs.

- Computation cost: The group members and the group key controller (if any) have to do computation to generate the member keys and the group keys. With the rapid increase in processing speeds of workstations, computation costs are becoming less important as a benchmark. However, there are some schemes that involve prohibitively heavy computation at the member nodes for large groups, and so this overhead should still be considered. Also, resource-constrained devices like PDAs would perform significantly worse compared to high-end machines for key generation, especially if public-key cryptography is involved [40]; the computation cost is an important criterion for these devices.

6.2 Security Terminology

We list some of the common terms and notations that we use to describe existing key management protocols.

1. The entities in a system who take part in receiving from and sending data to a group are called the group members. The size of the member set is usually denoted by n.

2. Some schemes have a trusted third party who generates and disseminates the keys to the group members. It is known as the *Group Controller* (GC) or the *Key Server*. It is usually not a member of the group.

3. The key used to encrypt group data traffic for a particular session is termed the *session key* (SK). It is also called the *traffic encrypting key* (TEK).

4. Some protocols require an additional set of keys which are mainly used to transport the TEK securely to the members. These keys are called the *key encrypting key* (KEKs).

5. Encryption of a message m using key K is denoted by $E_K(m)$. m_K refers to the encrypted message (also called ciphertext C): $C = m_K = E_K(m)$. Likewise, decryption is denoted by $D_K(C)$.

6. Transmission of an encrypted message from entity A to B is denoted by:

$$A \xrightarrow{m_K} B$$

7. In public key cryptography, the key pair is denoted by $\{K, K^{-1}\}$, where K^{-1} is the private key corresponding to the public key K.

8. The size of a public key is k_p bits, while a symmetric key is k_s bits.

6.3 Centralized Key Distribution vs. Distributed Key Management

A significant amount of research has been done in designing key generation and distribution protocols. Most of the protocols designed fall in two categories: centralized key distribution schemes and distributed key generation schemes.

6.3.1 Centralized Key Distribution

There is a centralized key controller to whom all members send join and leave requests. The key controller is fully trusted and is responsible for key generation and distribution to the group members, and for key updates, triggered periodically or on membership changes.

Centralized schemes provide a higher degree of security and are more efficient. Their major weakness is the dependence on a central entity, which can be a single point of failure. Centralized schemes require a degree of infrastructure to be available for the protocols to work (viz., availability of the key controller), which is not feasible in several group communication situations, such as an ad hoc network in a military battlefield. Examples of centralized schemes are [3, 4, 41, 7].

6.3.2 Distributed Key Generation

Distributed key generation schemes do not place key generation responsibilities on any one node. All the group members (or a chosen subset), contribute shares in a round of message exchanges to generate a common group key. Subsequent joins and leaves require further message exchanges to update the group key.

Distributed schemes are resilient to single-node failures. They are suited for hostile group environments where trust is at a premium, with each participating member being assured that the common key is generated with its contribution. They also do not require any infrastructure in most cases. The major drawback in distributed schemes is the communication overhead involved in key generation. For large groups, the amount of message exchanges for key generation and updates can be prohibitively high. Also, in certain group scenarios like IP multicast, a group member need not be aware of other members that have joined the group. This is contrary to the premise in distributed key generation that all members participating in the key setup are aware of everyone else, and can send messages in order to the others. Distributed schemes can also lead to deadlock situations, for example when the contribution from a key generation participant does not reach the other members due to channel errors. Examples of distributed schemes are [6, 5].

6.4 Review of Key Management Protocols

The schemes that have been proposed in the literature are too many to detail here. We describe in brief some of the fundamental ideas presented in centralized and distributed key management, and point to other similar well-known protocols.

6.4.1 Key Predistribution Systems

In [3] the authors proposed Key Predistribution Systems (KPS), a centralized scheme for distributing keys to groups of users. The scheme requires multiple trusted managing centers that a member contacts when it wants to join the system. The trusted centers generate keys for all possible groups, and distribute to the joining entity a list of all keys for the groups that has the entity as a member. Subsequently when a group of users want to establish secure group communication, each member reads out the common key from its list according to the identities of all the members. The scheme involves a one-time, two-message communication overhead to generate and distribute the keys to the members. However, the scheme assumes each key center is aware of which groups the entity would like to join in the future. To accommodate the possibility that groups might change with dynamic joins and leaves, and the possibility that a joining entity can potentially be interested in forming all the groups that are possible with the other entities, a trusted center will need to generate a huge number of keys for every member. The storage requirements at the trusted centers and the members can become prohibitively high for large groups. The member storage required is $(2^{n-1} - 1)$ for a system with n entities, while the storage at the controller is $2^n - (n+1)$ keys. Even for a modest system with 100 nodes, each entity might need to store $6 * 10^{29}$ keys; considering 64 bit symmetric keys (DES [42]), the total storage requirement is of the order of $4 * 10^{18}$ TB.

6.4.2 Broadcast Encryption

Broadcast Encryption (BE)[4] is similar to KPS. BE requires a centralized key server which generates and distributes the keys to the entities in the system. The most basic scheme requires $\sum_{i=0}^{r} \left(\begin{array}{c} n \\ \\ r \end{array} \right)$ keys storage at each user for r possible groups. The authors improve on the storage requirements by relaxing the security constraints, and by increasing the number of messages sent from the center to the entities. Their $k - resilient$ scheme requires every user to store $\mathcal{O}\left(k\, log\, k\, log\, n\right)$ keys and the center to broadcast $\mathcal{O}\left(k^2\, log^2 k\, log\, n\right)$ messages. The (k,p)*-random resilient* scheme described in [4] requires $\mathcal{O}\left(log\, k\, log(\frac{1}{p})\right)$ key storage and $\mathcal{O}\left(k\, log^2 k\, log\, (\frac{1}{p})\right)$ messages broadcast. Calculations with representative group sizes show that neither of the schemes can scale very well. Improvements on the above have been proposed in [43], but at the cost of a significant relaxation in security (the improvement comes by allowing a higher threshold of unauthorized users to decrypt the data). Another *threshold broadcast encryption* scheme was proposed in [44] based on *"k out of n" secret sharing*. It requires any k out of n participants to pool their shares (which they were given *apriori* by a central controller) to reconstruct the secret. Apart from requiring collaboration between participants (who might not know each other as in IP multicast), the storage requirement can be very high for large groups. Also, the scheme is suited for one-to-many traffic only, with the key controller being the source knowing the entire secret.

6.4.3 Secure Lock

Secure Lock[45] is a secure broadcasting scheme proposed for one-to-many traffic. The "lock" is constructed using the Chinese Remainder Theorem (CRT), the lock being the common solution of the CRT. We discuss the mechanism in brief.

- The key controller shares a unique secret key with each user. For every encrypted message $C = E_{\hat{e}}\left(m\right)$, the message-decrypting key $\hat{d}$ is enciphered separately for each receiver using the receiver's shared secret, yielding ciphertext $R_i = E_{ek_i}\left(\hat{d}\right)$ for user i; the common solution X for all R_i is computed using CRT:

  $$X \ = \ R_i \, mod \, N_i \ \ \text{for all users} \, i$$

  where N_i, $i \in \{1, ...n\}$ are the public relatively prime integers.

- The center broadcasts $\langle X, CKD = E_{\hat{e}}\left(\hat{d}\right), C\rangle$, to all users.

- The users compute $R_i = (XmodN_i)$, and $\hat{d}$ by decrypting R_i using their secret. $\hat{d}$ validity is checked by: $\hat{d} \overset{?}{=} D_{\hat{d}}(CKD)$. $\hat{d}$ is then used to decrypt C to get the original message M.

Secure lock therefore requires each user to store two keys - the long-term shared secret with the source, which is obtained by a two-message exchange with the key controller during initial setup, and the current session key. However, the number of key encryptions done at the source increases linearly with the number of members. Even if the computational burden is not heavy, this system is strictly one-to-many, since only the key controller can know the shared secret with every receiver. The key storage required at the source is also very high. To adapt the system for multiple sources would require every receiver to share long-term keys with every source. The scheme would then face storage problems similar to KPS or BE.

6.4.4 Conditional Access Systems

Conditional Access Systems (CAS)[46] is popular for data confidentiality in satellite broadcasts. The CAS system is one-to-many and shares similarities with Secure Lock. The source shares long-term secrets with every receiver (e.g. subscribers in a cable network receiving their long-term passwords in smart cards). Data transmission is “encrypted” using ephemeral keys, the decryption key being enciphered individually for every receiver using its long-term secret. The key information is sent along with the encrypted data. Decryption is somewhat similar to the Secure Lock case.

CAS suffers from the same kind of inefficiency as Secure Lock, with the source having to perform individual encryption for every receiver. As stated earlier, it is a one-to-many system.

6.4.5 Group Key Management Protocol

Group Key Management Protocol (GKMP)[41] has been proposed for key management in groups with multiple sources and receivers. The scheme uses a centralized controller called the Group Security Controller, or GSC, that generates and distributes the group key (figure 6.1). In GKMP each user shares a unique long-term key with the GSC. The GSC generates the session key and sends it by unicast to each member, encrypted with the member's long-term secret. The storage required at each member is only 2 keys - the group session key, and the KEK, which is the member's long-term secret. The GSC needs to store $(n + 1)$ keys.

The system is simple, but the communication overhead for the initial setup, and key updates on member leaves, is high for large groups. For n members, the GSC needs to exchange $2n$ messages for the initial setup. On a member leave, $(n - 1)$ messages are sent from the GSC to the remaining members. However, the cost is only 3 messages on a join. The scheme therefore scales poorly and is suited only for very small systems.

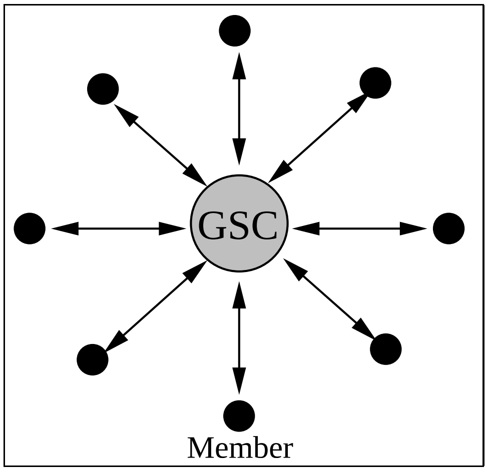


Figure 6.1: GKMP Framework

6.4.6 Key Agreement based on Hidden Fractional Keys

In the distributed key agreement area, several protocols have been proposed. Key agreement for secure multicast using *hidden fractional keys* (HFK) has been proposed in [6], with extensions in [47]. These protocols require a trusted third party to distribute the “initial pads” and the “blinding factor” to all the members participating in the group key generation. Subsequently the members go through a round of message exchanges, with each member making its contribution, the *fractional key*, to the shared pool, the key being hidden using the member’s initial pad. Once all the participants have made their contributions, each member can individually compute the key (or the keying material) by removing the blinding factor, which is the combined effect of all the members’ pads. The protocol is elegant with no member’s fractional key being exposed to the other members, even though the final key has every participant’s contribution; every member arrives at the same final result securely. A schematic is given in figure

6.2.

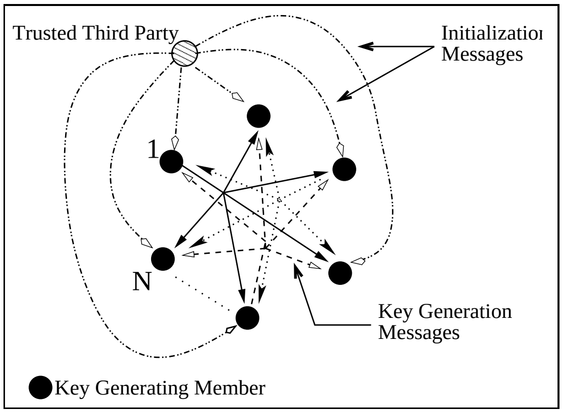


Figure 6.2: Message Exchanges for Key Agreement using Fractional Keys

The computation and storage requirements are low in HFK; for example, simple X-OR can be used for hiding the keys. Each member stores only the session key, its fractional key, its initial pad and the blinding factor. However, the method requires a trusted third party for initialization. The third party can be removed with one of the participants taking the role of the *group initiator*; but then the scheme adds n additional rounds of initial message exchanges to distribute the members' initial pads and the blinding factor. The exchanges have to be *ordered*, with each member knowing its immediate neighbors. That might not be feasible in a IP multicast scenario, where members might not be aware of each other. Where communication is expensive, the number of message exchanges between the participants can be costly. For the trusted third party case, the communication cost is n messages from the third party to the n participants; and an additional $n(n-1)$ message exchanges between the participants to distribute the HFKs. For the distributed initialization, the message exchange is

$(2n-1)+n(n-1)$. (This cost can however be amortized by broadcasting the messages; the original protocol was suggested for the broadcast setting.) Also, the protocol does not handle membership changes well; if a member becomes unavailable, then the scheme has to go through n message exchanges to recover the member's contribution; otherwise, future key updates based on the participating members' existing fractional keys is impossible and the protocol has to be restarted. An attempt has been made to improve on the last problem in [47], but the improvement is neither efficient nor fully secure.

6.4.7 Group Diffie-Hellman Protocols

A suite of protocols have been proposed in [5, 48] for fully distributed group key agreement between a set of participating entities without any trusted third party or any security assumptions about the participating nodes.

The multi-party group Diffie-Hellman comes in many flavors - GDH.1, GDH.2 (and its authentication extensions), GDH.3[5, 48] and TGDH[49]. Here we describe the simplest, GDH.1. The protocol has two stages: *upflow* and *downflow*.

- In the upflow stage, in round i $(i \in [1, ..n-1])$, member M_i collects the contributions from members M_j, $j \in \{1, ..., i-1\}$, and computes $g^{N_1, ...N_i}$ on the $g^{N_1, ...N_{i-1}}$ received from M_{i-1}. M_i then sends to M_{i+1}:

$$M_1 \xrightarrow{\{g^{\Pi\left(a_k|k\in[1,j]\right)}|j\in[1,i]\}} M_{i+1}$$

- In the final transaction in the upflow stage, M_n computes the group key:

  $$K_n = (g^{a_1, \ldots a_{n-1}})^{a_n}$$

  In the upflow stage, each member performs one exponentiation, and a total of (n-1) messages are exchanged, with the message between M_i and M_{i+1} containing i values.

- After computing K_n, M_n initiates the downflow stage:

  $$M_{n-i} \xleftarrow{\{g^{\prod(a_k | k \ni [1,j])} | j \in [1,i]\}} M_{n-i+1}$$

  Each M_i performs i exponentiations - one to compute K_n from the values received from M_{i+1}, and $(i-1)$ to provide intermediate values to members $M_j,\ j \in [1, ..., (i-1)]$. Hence a downflow message from M_{i+1} to M_i has i values, there being a total of $(n-1)$ such messages.

A schematic for the message exchanges is given in figure 6.3. Thus in summary, GDH.1 protocol requires $2(n-1)$ rounds with $2(n-1)$ messages being exchanged, the combined size of which being n(n-1). Also, member M_i ($i \in [1, ..n-1]$) needs to perform $(i+1)$ exponentiations, and n for M_n. The total exponentiations in one key generation is $\frac{(n+3)n}{2} - 1$.

The protocol is elegant and allows a group of entities to set up a common group key without any infrastructure. The entities do not need to trust one another. However, it scales very poorly. The number of message exchanges and the size of the messages become very high for large groups. The messages also have to be *ordered*, requiring the entities to be aware of their immediate neighbors. But most importantly, the computational burden on each entity is prohibitive for large groups. Exponentiation is an expensive operation of cubic complexity; in a group of 1000 nodes, the total exponentiations required in GDH.1 is of the order of $5 * 10^5$, with each node performing 1001 exponentiations, thereby introducing high latency. One can obtain the cost metrics for GDH.2 and TGDH from [49] - the cost of exponentiations is very high even for small group sizes (for example, 140 members). The family of protocols is therefore unsuitable for very large dynamic groups.

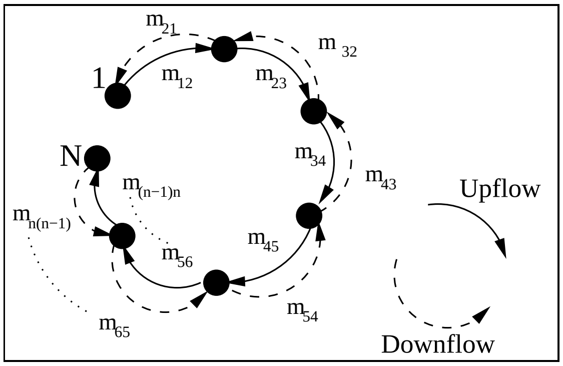


Figure 6.3: Key Agreement in Group Diffie-Hellman

Several other protocols based on the Diffie-Hellman discrete logarithm problem have been proposed in [50, 51, 52]; all are susceptible to similar inefficiency problems in large groups.

6.4.8 Tree Based Key Distribution Protocols

A family of protocols have been proposed for key generation and distribution based on logical key trees. The original idea of using rooted trees for key distribution was independently proposed in [7, 53]. We briefly review the basic centralized tree based key management in this section.

The group controller (GC) constructs a logical key tree hierarchy (LKH) with the group members as the leaves of the tree. The internal nodes of the tree are the key encrypting keys (KEK) which are used to securely transport key updates to the group. The root of the tree is the session key or traffic encrypting key (TEK). The key corresponding to a leaf node is the long-term secret that the corresponding member shares with the GC. A leaf node knows all the keys on the path from its leaf to the root, and no other. Figure 6.4 illustrates a binary key tree for 8 members.

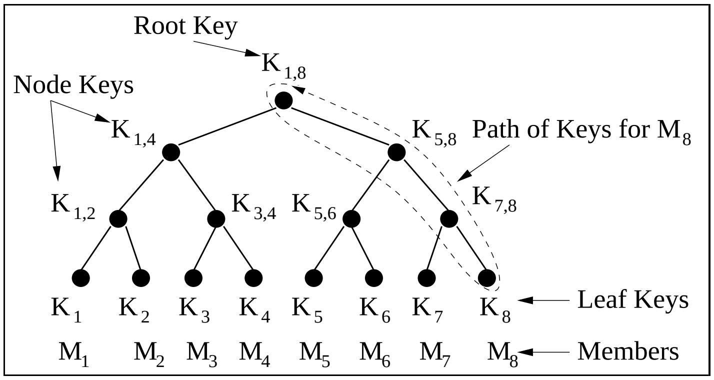


Figure 6.4: Binary Logical Key Tree of 8 Nodes

The tree structure creates a natural hierarchy of the members in a group[6]. The GC can place the members logically, corresponding to the network setup and/or application requirements; choose appropriate keys for the members, and hence selectively update the keys of the group as needed.

When a member joins or leaves, the GC needs to update only a subset of the keys in the tree selectively. Figure 6.5 shows the keys updated by the GC when member M_5

joins the group in figure 6.4.

- The GC first updates the root key to $\hat{K}_{1,8}$ and securely transmits it to the current members using the old root $K_{1,8}$ key:

$$GC \xrightarrow{\{\hat{K}_{1,8}\}_{K_{1,8}}} M_1, M_2, M_3, M_4, M_6, M_7, M_8$$

- The GC also updates the internal node keys $K_{5,8}, K_{5,6}$ on the path from the root to the joining member M_5 and transmits them securely to the relevant group members:

$$GC \xrightarrow{\{\hat{K}_{5,8}\}_{K_{5,8}}} M_6, M_7, M_8$$
$$GC \xrightarrow{\{\hat{K}_{5,6}\}_{K_6}} M_6$$

- Finally, the GC transmits all the keys in the path from the root node to M_5 using M_5's long-term secret K_{GC,M_5} (assuming it is pre-established):

$$GC \xrightarrow{\{\hat{K}_{1,8}, \hat{K}_{5,8}, \hat{K}_{5,6}, K_5\}_{K_{GC,M_5}}} M_5$$

When a member leaves the group or is revoked, all the keys known to the member have to be invalidated and new keys generated as needed. For simultaneous leave of multiple members, the GC has to identify all the invalid keys and the minimum number of valid keys that are required to transmit the updated keys to the existing members. For example, figure 6.5 shows the list of keys that need to be replaced when M_2 is revoked. A total of 3 keys need update - the root key $K_{1,8}$, and the internal KEKs $K_{1,4}, K_{1,2}$ that were known to M_2. GC sends the following messages in order to

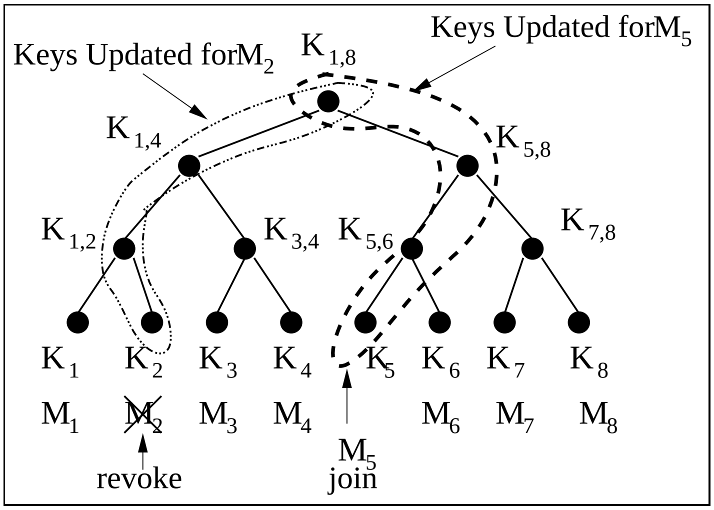


Figure 6.5: Key Update in a Binary Logical Key Tree

transmit the updated keys to the existing members:

$$\begin{array}{lcl} GC & \xrightarrow{\{\hat{K}_{1,2}\}_{K_1}} & M_1 \\ GC & \xrightarrow{\{\hat{K}_{1,4}\}_{\hat{K}_{1,2}},\{\hat{K}_{1,4}\}_{K_{3,4}}} & M_1, M_3, M_4 \\ GC & \xrightarrow{\{\hat{K}_{1,8}\}_{\hat{K}_{1,4}},\{\hat{K}_{1,8}\}_{K_{5,8}}} & M_1, M_3, M_4, M_5, M_6, M_7, M_8 \end{array}$$

In a d-ary key tree protocol, the total number of keys required to be stored at the GC is $\frac{d^{h+1}-1}{d-1} = \frac{nd-1}{d-1}$, and at each member is $(h+1)$, where h is the height of the key tree, $h = log_d(n)$. For the initial tree setup, the GC has to perform $\frac{d(n-1)}{d-1} + n$ key encryption operations, which can be sent in an equivalent number of messages, or can be broadcast in a single message (the latter is preferable in terms of rounds, though the message size will be larger). On a member join, the GC has to update h keys and generate a new key for the new leaf, and send the updated keys to affected members only. The GC requires $dh + 1$ rekey message components, which can be sent in two messages - one to the existing members, and the other to the joining member. On a member leave, the number of keys updated is h, requiring $dh - 1$ encryption cost at the GC and one message transmission to the members of size $dh - 1$ keys. The key tree protocols have communication and computation complexity of $\mathcal{O}\left(log_d n\right)$. The storage required at each member is also $\mathcal{O}\left(log_d n\right)$.

The tree based protocols scale very well to large groups. Their primary drawback is the use of a centralized GC; protocols that do away with the GC have been suggested, at the cost of higher complexity[54]. Various modifications to the original protocol have been made that try to reduce the communication and computational complexity. Canetti et al. [8] have proposed an optimization to the original LKH that halves the size of the key update message; the optimization is called LKH+. Computation of the optimized tree structure based on the probabilities of member joins and leaves have been discussed in [6, 55, 56]. We can do away with the GC sending key updates to the members on a join; protocols that allow the members to independently update the keys on the path to root (while maintaining overall tree consistency) have been developed [57, 58]. We incorporate these ideas in our framework and describe them in detail in chapter 7.

Tables 6.1 and 6.2 gives a comparison of the key management protocols presented in chapter 6.

	KPS	BE	Secure Lock	GKMP
Group setup Comm.(bits)	$n(2^{n-1}-1)k$	$n(\sum_{i=0}^{r}\begin{pmatrix} n \\ r \end{pmatrix})$	nk	$2nk$
Rounds	1	1	n	1
Member add Comm.(bits)	0	0	0	$3k$
Rounds	0	0	0	2
Member evict Comm.(bits)	0	0	0	$(n-1)k$
Rounds	0	0	0	$n-1$
Storage Controller	$(2^n-(n+1))k$	$(\sum_{i=0}^{r}\begin{pmatrix} n \\ r \end{pmatrix})k$	$n+1$	$n+1$
Member	$\lceil(2^{n-1}-1)k\rceil$	$\lceil rk\rceil$	2	2

Table 6.1: Comparison of Centralized Key Management schemes. Here we take a uniform key length of k. r is the maximum number of possible groups in Broadcast Encryption. The other symbols have been explained in the text. For storage, we consider only long-term keys stored. In Secure Lock, the key information is piggybacked with the data, and does not require any extra round, except the initial long-term channel setup.

	HFK	GDH.1	LKH
Group setup			
Comm.(bits)	n^2k	$n(n-1)k$	$(n + \frac{d(n-1)}{d-1})k$
Rounds	n^2	$2(n-1)$	$\mathcal{O}(n)$
Member add			
Comm.(bits)	$\mathcal{O}(n^2k)$	$\mathcal{O}(n^2k)$	$(dh+1)k$
Rounds	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$dh+1$
Member evict			
Comm.(bits)	$\mathcal{O}(n^2k)$	$\mathcal{O}(n^2k)$	$(dh-1)k$
Rounds	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$dh-1$
Storage			
Controller	–	–	$\frac{nd-1}{d-1}$
Member	3	2	$h+1$

Table 6.2: Comparison of Distributed Key Management schemes and LKH. Here we take a uniform key length of k. d is the degree of the logical key tree and h is the height of the tree. n is the total number of nodes in the group. The above table does not show the computation cost, which is a major drawback in GDH.

Chapter 7

Multicast Key Management in Satellite ATM Network

We describe the proposed key management framework in this chapter. The key management framework builds on the multicast network architecture that has been proposed in chapter 4.

7.1 Trust Model and Security Assumptions

The network entities that are relevant in the security framework are the MARS, the RPs and key server in each subnetwork and the end-hosts. This section describes the trust placed in each entity, and other security assumptions that are made about the model.

- MARS: The MARS is owned and controlled by the network provider. We assume that the application service providers who lease the network services from the network provider prefer to keep their application data confidential from the network provider. However, the network provider needs to know which domains/subnetworks are utilizing the network for transmission/reception for billing purposes, etc. In the security framework we therefore model the MARS as the trusted third party for the following functions:

- The MARS performs access control, based on group policy, for different subnetworks that want to join or send to a given multicast group. For this, the MARS authenticates the join/send requests that come from the RPs servicing their respective subnetworks. The mechanisms for establishment of group access control policy and authentication of the RPs are assumed to be in place for the data security architecture.
- The MARS maintains the database of multicast group membership at the subnetwork level. The MARS periodically sends the group membership information to all the RPs that are subscribed to the group.
- In addition, the MARS acts as a Certificate Authority (CA) for verifying the public keys of the RPs when needed. This is only in the case where we assume that the bootstrapping of the secure channel between two RPs is done online, using public keys.

The MARS is not trusted with the multicast traffic. The MARS should not receive the application data (unless it explicitly subscribes as a member to the multicast group). If the MARS "listens" to the group traffic without subscribing to the group, it should not be able to read or modify the multicast traffic.

- RP: In the security framework, the RP is trusted for the following functions:
  - The RP securely transmits the multicast traffic to all the group members in its subnet.
  - The RP securely transmits the multicast traffic, generated by any source local to its subnet, across the satellite links to other subnetworks that have receivers for the multicast group. It is assumed that the RP performs suitable source authentication check on the data before forwarding it onto the local tree or to other subnetworks.
  - The RP securely receives data from other subnetwork RPs, if it has group members in its local multicast tree. It is assumed that the receiving RP performs suitable source authentication check on the received data before forwarding it onto the local tree.

  The RP is not trusted to read the multicast traffic, even though it is trusted to receive and forward the data. This requires that the RP should not be able to decrypt the application data. We place this limitation since the RP is located at the satellite gateway in each subnetwork, and it is owned by the network provider.

  The RP transmitting data to other subnetworks does not perform access control on the receiving subnetworks; access control is performed by the MARS. We assume that messages from the MARS to the RPs are integrity-protected. The RP sending data to other subnetworks, therefore accepts from the MARS messages, the list of subnetwork RPs as routers with valid access permissions for sending and receiving group multicast traffic.

- End-Host: The end-hosts are trusted to securely encrypt or decrypt the multicast traffic. We assume that the end-hosts perform source authentication checks on the received traffic before they accept the data.

- Subnetwork Key Controller: In addition to the above network entities, the framework requires a key server in each subnet, which is termed the *Subnetwork Key Controller* (SKC). The SKC is responsible for managing group keys in its subnet. It is trusted for the following functions:

  – The SKC performs access control operations when a subnetwork host wants to join a multicast group as member or wants to send data to a multicast group.

  – The SKC performs key generation and distribution and periodic key updates for all multicast groups that have members in its local subnet. The key management done by the SKC is limited to the group members in its subnet, and does not affect members outside.

Each end-host is assumed to *apriori* establish a *secure channel* to the SKC for receiving the key information. The secure channel is established based on either a *shared secret* that is known only to the SKC and the particular member, or a public-private key pair. The SKC can be co-located with the RP in its subnet, but we make no assumption about this. In the design, the SKC is considered to be a separate entity. The SKC and the RP in each subnet establish a secure channel between them; the SKC uses the secure channel to send the group session key to the RP.

In addition to the above, we make the assumption that the IP/ATM multicast routing is secure, i.e., all routers in the multicast tree are trusted to correctly forward the multicast traffic. The routing messages between the routers are properly authenticated. The routers also authenticate all the hosts and do access control checks on them before they are allowed to join the multicast tree or are allowed to send to a multicast group.

7.2 Tiered Tree Based Key Management

The primary metric that we consider for our design is the communication overhead in the network. As mentioned previously in 6.1.2, the propagation delay in the geostationary satellite links is high, of the order of 250ms in one hop. The uplink bandwidth is limited to 1.5Mbps. Also, geostationary satellites operating in the Ka-band are highly susceptible to atmospheric conditions such as rain fade [59]. We therefore need a key management scheme that minimizes the communication over the satellite links, to reduce the delay in group initialization or key updates, and also to minimize the possibility of error conditions where the group keys do not reach all the members due to channel conditions. The processing power or memory capacity in current computers is significant so that computation or storage problems are not critical issues.

The hierarchical structure of the network creates two distinct levels in the network - the terrestrial subnetworks, and the satellite connections between the subnetworks forming an “overlay”.

We divide the key management into two tiers - one at the subnetwork level, while the other at the level of the satellite overlay. A schematic is given in 7.1. The key generation and distribution in each subnetwork is independent of one another, and also of the key generation and distribution in the overlay; we add mechanisms so that the encrypted data can be transferred securely across the different key management areas. The key management in each logical group is based on centralized key trees. The key management therefore has two trees: a global *RP Tree* for managing the keys between the subnet RPs in the overlay; and the local *SN Tree* for managing the keys amongst the hosts in each subnet. We term this framework, *Tiered Tree Based Key Management*. The concept of dividing a system into subgroups for scalable key management was

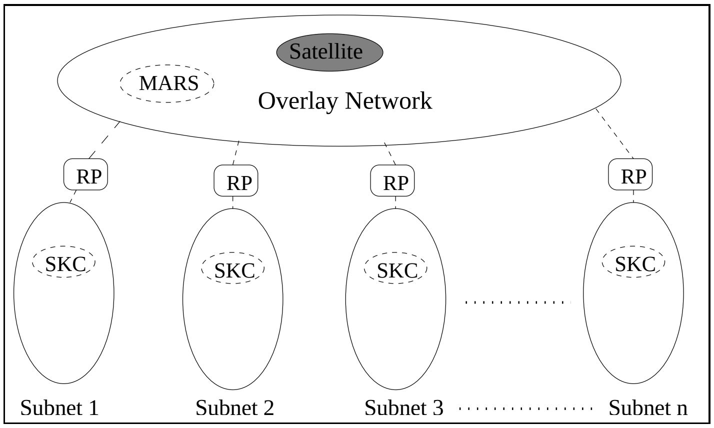


Figure 7.1: Logical Grouping in Tiered Tree Framework

originally proposed in Iolus[60]. The paper considered peer subgroups being relatively independent of one another, each having its own multicast group with its own address. Iolus has a top-level subgroup managed by a *group security controller* (GSC); the key management in each subgroup is done by *group security intermediaries* (GSI). The GSIs are subgroup representatives of the GSC, and therefore there is a dependency between them. Iolus considers a hierarchy of GSIs, with the GSIs at one level joining the subgroups of the GSIs at the next higher level or the subgroup of the GSC. This way a *secure distribution tree* is built. Key management using a hierarchy of logical key trees has also been explored in [61], but it does not consider the underlying network characteristics.

We now describe the operational details of our framework.

7.2.1 Key Management in the Overlay: RP Tree

Figure 7.2 illustrates the key trees for the overlay and each subnetwork in our framework. The members of a multicast group in the overlay network are the RPs. The key management is centralized and based on the logical key hierarchy concept; we term the logical key tree in the overlay, the RP Tree. The RPs in different subnetworks are located at the leaves of the RP tree. The root of the RP tree is one of the RPs in the group, as explained below.

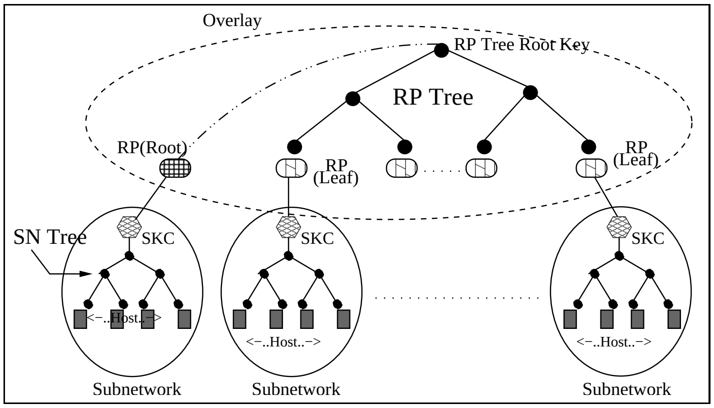


Figure 7.2: RP Tree and SN Tree

7.2.1.1 RP Tree Setup

Additions are made to the MARS message exchanges protocol [36] to setup the RP tree.

Sender RP Request: When a RP has hosts downstream who wants to send to group G, the RP sends a request message to the MARS for the list of group members. The request message will contain the joining group IP address, the ATM address of the requesting RP, and also the IP address and public key of the RP in the data portion.

$$RP \xrightarrow{\{IP_G, IP_{RP}, K_{RP}\}, \{h(m)\}_{K_{RP}^{-1}}} MARS$$

where:

- IP_G and IP_{RP} refer to the IP addresses of group G and RP respectively
- K_{RP} is the public key of the RP
- $\{h(m)\}_{K_{RP}^{-1}}$ is the signature on message $m = \{IP_G, IP_{RP}, K_{RP}\}$, signed by

using a suitable hash function $h()$ and the private key K_{RP}^{-1} of the RP

We assume all messages carry proper authentication data (for example, signatures as above) and are omitted from subsequent messages. We add fields to the MARS message structures for implementing the key management functionality.

If the MARS database has a non-empty entry of RPs subscribed to G, the MARS adds the requesting RP to the entry, and returns the details of the entry to the requesting RP in a reply message. The reply message is broadcast to all RPs in G present in the MARS entry at that time. The message has the IP address and public key of each valid RP, and the address of the RP tree controller:

$$MARS \xrightarrow{\{IP_G,\{IP_{Root},K_{Root}\},\prod\{IP_{RP_j},K_{RP_j}|j\in\{1,..,l\}\}\}} RP_i$$

$$\forall i \in \{1, .., l\} \text{ s.t. } RP_i \in G \text{ and } l \leq q$$

where we assume there are l RPs subscribed to the group, IP_{Root}, K_{Root} are the IP address and public key of the root RP respectively, and there are in all q RPs in the network (i.e., q subnetworks). The message is the MARS_MULTI message, with above fields added for security functionality.

If MARS has no entry for G (i.e., the requesting RP is the first to join G at the MARS), then MARS creates a new entry for G, adds the requesting RP ATM address to the entry, and sends a negative acknowledgment in reply. The following new fields are added to the MARS database entry for each group.

1. For each RP, a tag to indicate whether sender or receiver RP or both;

IP Multicast Address	ATM Endpoint Address	Sender/ Receiver	Join Time
Class D address$_1$	ATM.1	S	JoinTime.1
-	..	..	..
-	ATM.n	R	JoinTime.n
.	.	.	.
.	.	.	.
.	.	.	.
Class D address$_N$	ATM.1	R	JoinTime.1
-	..	..	..
-	ATM.n	S	JoinTime.n

Figure 7.3: IP-ATM address mapping table at MARS with security enhancements

2. For each sender RP, the joining time (required for selection of RP tree root).

Figure 7.3 shows the MARS database updated with the information required for security. The MARS database entry can also include the access control policy, privilege list, etc. for each group; we assume that access control is done by the MARS before adding each RP to the group entry in its database.

Receiver RP Join: When a RP has hosts in its local subnetwork requesting to join a group G as receivers, the RP sends a join request to the MARS. The security-relevant fields in the message are similar to the one above. The MARS adds the joining RP's IP address, public key to the database entry at MARS for group G. If the entry does not exist, then a new entry is created. Subsequently the MARS broadcasts the list of RP group members in a regular membership update message to all the sender RPs subscribed to G. The reply message format is the same as in the sender RP case.

Selection of the RP Tree Root: The root of the RP tree is selected to be the sender RP that is the *earliest to join* the group amongst the sender RPs in the MARS group entry. The selection is done by the MARS based on the join time in the requests it receives. The address and public key information of the root RP becomes known to all the group RPs from the MARS message they receive. This is important so that group RPs can verify the source of the key information messages that they receive from the root. In case the root RP leaves the group, the MARS checks the joining times of the remaining sender RPs, selects the earliest-to-join, and broadcasts a new message to the group. The RPs update their local group security information upon receiving the MARS message.

Tree Setup at the Root: When a sender RPs receives the MARS message, it checks whether it is the root. If so, it proceeds to set up the logical key tree in its local node. We suggest two efficient schemes to set up the key tree in section 7.2.5. The information about the leaves of the key tree are obtained from the MARS message itself - the IP addresses of all RPs in G. Once the root RP (henceforth referred to as "root") has generated all the keys in the tree, it proceeds to send the keys to the appropriate leaf RPs. In cases where there is more than one sender RP, all sender RPs except the root are added as leaves to the RP tree by the root.

Secure Channel between Root and Leaf RPs: To send data securely between the root RP and any leaf RP, first a secure channel has to be established between the two. The secure channel can be established either *offline* or *online*. In the offline case, we can assume that there exist *apriori* long-term secrets between the root RP and the leaf RPs. The root RP for any given group can change over time, and any RP is a potential root. Hence prior establishment of long-term secrets would require every RP to share a secret with every other - this has $\mathcal{O}\left(n^2\right)$ complexity. Since the number of subnetworks are much less than the actual number of hosts, and will not exceed several hundred in a typical network, this will require each RP to store several thousand symmetric keys beforehand. Since the satellite gateways where the RPs are located are a part of the network owned by the network provider, this assumption is also not unreasonable.

In case the secure channel is set up online, one can use public keys. The public keys of all the RPs in the group can be obtained from the MARS message, as shown in section 7.2.1.1. In the initial communication, the root RP encrypts the leaf node symmetric key (the long-term secret shared between the root RP and the leaf RP) using the public key of the corresponding leaf RP; the keys of the next higher level in the tree are encrypted using the symmetric leaf key, and so on. The long-term secret is cached at both RPs and all subsequent communication from the root RP to a leaf RP uses the long-term secret shared between the two for establishing the secure channel. Here the initial communication and key processing cost for setup is higher than in the offline case, but the total number of keys that need to be stored at either a root RP or a leaf RP is potentially much less in comparison. A root RP for a group needs to store as many long-term keys as there are (or has been) leaf RPs for that group; a leaf RP needs to store as many long-term keys as the number of groups for which it is (or has been) an RP tree member.

Use of public keys requires access to a Certificate Authority (CA) for verifying the association between a node identity and its advertised public key. The CA is a trusted third party to which all the entities have easy access. In the satellite network, the MARS is a central point to which all the RPs have access. In our security design, the MARS is trusted with performing access control on the RPs joining a group. Therefore, we make the MARS the CA for verifying the public keys of RP nodes, if needed in the bootstrapping of the secure channel between the root RP and the leaf RPs, in the case the secure channel is set up online.

Key Information Transmission: Once the RP tree has been setup at the root, the root creates one message containing all the keys of the RP tree, encrypted as appropriate, and broadcasts the message over the satellite links to all the other RPs in the group. Upon reception, the leaf RP decrypts its relevant key information using its private key, and obtains all the keys on the path from its leaf to the root of the tree. The key corresponding to the tree root is now used as the session key.

7.2.1.2 Tree Update on Member Join, Leave

When a RP wants to join an existing group as a member, it sends a join request to the MARS as described above. The MARS adds the RP to the group entry. When a leaf RP leaves a group due to absence of any sender or receiver downstream in its subnetwork RP tree, it sends a leave request to the MARS for the group. The leave message contains, in addition to the standard MARS fields, the IP address of the RP.

$$RP \xrightarrow{\{IP_G, IP_{RP}, K_{RP}\}} MARS$$

The MARS checks whether the leaving RP is the RP tree root and removes the RP information from the group entry. The join or leave message is retransmitted to the existing group members to update them about change in the group membership.

When the root RP sends a leave message, the chain of events is different. MARS removes the root from the group entry; runs an algorithm that selects a new root RP based on the earliest-to-join parameter; creates a new update message and immediately sends the update to the remaining group members. The new root, upon receiving the update message, proceeds to create a new RP tree as explained above. Till the new tree is created, the group information is secured using the existing session key. The drawback is that the old root RP can still receive all the information, but it prevents “blackout periods”.

The above assumes that there are multiple sender RPs, which is the case when the group has many sources spread across different subnetworks. However, a group can have only one sender RP (the root) in situations where there is only one source host, or all the sources are concentrated in the same subnet. In such a case, the root RP leaving implies there are no source hosts left, and the group should cease to exist. The MARS on getting the leave message cannot locate a new root. Then it does not send out a new update message. The group entry will be erased from the MARS database on a timeout, and also at each of the receiver RPs.

7.2.1.3 Tree Removal on Group Termination

When the remaining sender RP (who is also the root), leaves the group, the group terminates as described above. The sender RP simply removes the key management information in its local node.

When a group has no receiver RP remaining, the root gets this information from the MARS message. It then destroys the RP tree in its local node and stops sending information over the satellite links. The group might still have sources and receivers in the root RP's local subnet; key management for the group continues in the SN tree as described in 7.2.2.

7.2.2 Key Management in the Subnet: SN Tree

The key server in each subnet, known as the Subnetwork Key Controller (SKC), manages the subnetwork key tree (SN tree). We assume that the security module in all hosts and the RP are aware of the address of the SKC. We also assume that each host in the subnetwork and the RP have established secure channels to the SKC. Since the SKC in a subnet is unchanging, the secure channel is long-term and needs to be set up only once.

7.2.2.1 SN Tree Setup

When an end-host wants to join a multicast group G as a receiver, or intends to send to a multicast group as a sender, it first sends a *join request* message to the SKC specifying the IP address of G.

$$a_{ij} \xrightarrow{\{IP_G\}} SKC_i$$

where: a_{ij} is the j^{th} host in the i^{th} subnetwork and SKC_i is the key controller in subnetwork i.

In the subnet, the SKC does not need to differentiate between a sending host and a receiving host.

When the SKC receives a join request, it checks its local database for an entry for the group. If none exists, the SKC creates an entry and the corresponding key tree. The SKC also generates a *datahiding key* (DK) for the group. The datahiding key for group G has to be identical across subnetworks; the SKC in a subnetwork has to contact the SKCs in other subnetworks with members in G to agree on the datahiding key for G. The datahiding key is long-term; once created, it does not change for the lifetime of group G. The SKC assigns the joining host to a leaf in the tree. It then encrypts all the keys in the path from the leaf node to the root and the datahiding key using the long-term secret it shares with the joining host; it also encrypts only the session key for the RP. The SKC then forms a *key information* message containing the encrypted keys, and transmits the key information message to the host and the local RP.

$$SKC_i \xrightarrow{\{IP_G, \{K_0, .., K_{h_j}, DK_G\}_{K_{SKC_i, a_{ij}}}, \{SK_{G_i}\}_{K_{SKC_i, RP_i}}\}} a_{ij}, RP_i$$

where

- $K_0, .., K_{h_j}$ is the set of SN tree keys from the root to the leaf corresponding to host a_{ij};
- DK_G is the datahiding key for group G;
- SK_{G_i} is the current session key for group G in subnetwork i ($K_0 \equiv SK_{G_i}$);
- $K_{SKC_i, a_{ij}}$ is the shared secret between SKC_i and host a_{ij}, and
- K_{SKC_i, RP_i} is the shared secret between SKC_i and RP_i in subnetwork i.

The host decrypts the tree keys and group datahiding key and stores them in local memory. The RP decrypts the session key, creates an entry for the group in local memory, and stores the session key in the entry.

The key information message as described above is for only one joining host. When there are existing group members, or multiple members joining simultaneously, the message will contain all the relevant tree keys encrypted for all affected members.

7.2.2.2 Tree Update on Member Join

When one host sends a join request for group G to the SKC, the controller assigns the host to a empty leaf node in the SN tree. In case the tree is full, then a new branch is created and the member added as a leaf to the branch. All the keys in the path from the member to the root are updated, and a message sent to the existing group members informing them of the update. The local RP is informed about the update in the session key. (We present improvements in section 7.2.5 where the SKC does not need to send the updated keys to the existing members; affected members update the keys themselves on receiving a update notification from the SKC). Subsequently, the SKC encrypts all the keys in the path from the joining member leaf to the root, and the datahiding key, and sends it to the joining member.

For multiple members joining simultaneously, the sequence is similar, with the added processing at the SKC to find the minimum number of valid KEKs to send the update information.

7.2.2.3 Tree Update on Member Leave

When a member leaves, all the keys on the path from the member leaf to the root are invalidated. The SKC generates new keys in replacement, and sends the fresh keys to all affected members, and the RP. In this case the existing members have to be explicitly informed about the updated keys. For bulk member revocation, the SKC has to identify all the invalid keys, and find the minimal number of valid keys that are required to transmit the updated keys.

In case of either member join or leave, the datahiding key is not changed. Once created at the time of establishing the SN tree for group G, the datahiding key remains unchanged till the group terminates.

7.2.2.4 Group Termination

When all the members in a subnetwork have left group G, the SKC destroys the key tree and frees the local memory. But it saves the long-term shared secrets for every registered member for subsequent use in other groups. The RP also removes state for the local group when it tears down the inactive multicast tree.

7.2.3 Synchronization of Group Information at the RP

The RP is a part of the RP tree and it also has to store the subnetwork session key provided by the SKC. At all times, the RP maintains integrated state information for a group.

When the RP is a leaf of the RP tree, the group entry in local memory specifies it is the leaf, and contains the path keys to the root of the RP tree, and the subnetwork session key. If a leaf RP becomes a root, then a root entry is created. The subnetwork session key is transferred from the leaf entry to the root entry. The RP sets up the RP tree and stores all the keys in the root entry, then deletes the leaf entry. However, a root RP for group G does not become a leaf RP for G at any time when it is continuously subscribed to G.

7.2.4 Secure Data Transmission in a Group

Multicast traffic can be transmitted securely when the SN trees and the RP tree have been established. The sequence is described here.

1. Source host a_{ij} in subnetwork i encrypts the data m for group G twice: first using the datahiding key DK_G to produce ciphertext $C = E_{DK_G}(m)$. The encrypted data is re-encrypted using the subnetwork session key SK_{G_i} to produce ciphertext $\hat{C} = E_{SK_{G_i}}(C)$.

2. a_{ij} sends the doubly-encrypted data to the local multicast tree and the RP:

$$a_{ij} \xrightarrow{\hat{C}} a_{ik}, RP_i$$

$$\forall a_{ik} \in G \text{ in subnetwork } i, k \neq j$$

3. The group members aik in the local multicast tree decrypt $\hat{C}$ to retrieve the multicast traffic: $C = D_{SK_{G_i}}\left(\hat{C}\right)$, $m = D_{DK_G}(C)$.

4. The RP decrypts $\hat{C}$ to obtain C. It cannot decrypt C to get m, since it does not know DK_G. The RP re-encrypts C with the RP tree session key $SK_{G_{RP}}$ and transmits the ciphertext $\hat{C'} = E_{SK_{G_{RP}}}(C)$ to the other subnetworks over the satellite link.

$$RP_i \xrightarrow{\hat{C'}} RP_j$$

$$\forall RP_j \in G, j \neq i$$

5. RP_j in subnetwork j receives the encrypted transmission. It decrypts $\hat{C'}$ to obtain $C = D_{SK_{G_{RP}}}\left(\hat{C'}\right)$. RP_j cannot decrypt C since it does not know DK_G. It re-encrypts C using the local subnetwork session key SK_{G_j} for G to generate ciphertext $\hat{C''} = E_{SK_{G_j}}(C)$; RP_j sends $\hat{C''}$ along the multicast tree in its subnet.

$$RP_j \xrightarrow{\hat{C''}} a_{jk}$$

$$\forall a_{jk} \in G \text{ in subnetwork } j$$

6. Each host a_{jk} in subnetwork j subscribed to G receives $\hat{C}''$. It decrypts the ciphertext using SK_{G_j} to obtain C. a_{jk} decrypts C using the datahiding key DK_G to obtain m: $m = D_{DK_G}(C)$.

Thus data flows securely end-to-end across the network.

7.2.5 Algorithms for Managing the Key Tree

Different centralized key management techniques can be applied to our framework, both in the overlay and in the subnetworks. For scalable key management we have proposed use of logical key trees. In the family of logical key tree protocols, there are several that can be applied, apart from the basic LKH. Here we discuss two that we consider to be very good candidates to reduce the overhead of key management.

7.2.5.1 One-Way Function Tree

One-way Function Tree algorithm (OFT) [57] uses one-way functions to compute the key tree. The keys are computed up the tree, from the leaves to the root. The algorithm uses binary trees. The group controller maintains a binary tree, each node x of which is associated with two cryptographic keys, a *node key* K_x and a *blinded node key* $K'_x = g(K_x)$. The blinded node key is computed from the node key using a one-way function g. It is computationally infeasible for an adversary with limited processing power to obtain K'_x from K_x. The interior node keys are defined by the rule

$$K_x = f\left(g\left(K_{left(x)}\right), g\left(K_{right(x)}\right)\right)$$

where $left(x)$ and $right(x)$ denote the left and right children of node x.

The *system invariant* for the OFT algorithm states that each member knows the unblinded node keys on the path from its node to the root, and the blinded node keys that are siblings to its path to the root, and no other blinded or unblinded keys. Each member maintains the unblinded node key of its associated leaf, and the blinded node keys for all the siblings of the nodes along the path from its leaf to the root. This enables the member to compute the unblinded keys along her path to the root, including the root key. If one of the node keys changes, the member can recompute the keys on the path to the root, when informed of the updated node key value. The algorithm assures consistency in operation; each member arrives at the same view of the path to the root that is consistent with the view of the key tree maintained at the controller. The algorithms for member addition and deletion are detailed in [57].

OFT reduces the communication overhead in member joins and leaves, compared to the basic LKH algorithm.

7.2.5.2 ELK Protocol

The ELK protocol[58] uses centralized key trees for key distribution, and is somewhat similar to the OFT algorithm. To update a node key K, ELK uses contributions from the two child keys of K, K_L and K_R. The left child key K_L contributes k_1 bits:

	LKH	OFT	ELK (Full)
Group setup			
Communication (bits)	$(3n-2)k$	$(3n-2)k$	$(3n-2)k$
Adding a member			
Communication (bits)	$2hk+k$	$hk+k$	0
Adding l members			
Communication (bits)	$2s_lk+lk$	s_lk+lk	0
Evicting a member			
Communication (bits)	$2hk-k$	$hk+k$	$(h-1)(k_1+k_2)$
Evicting l members			
Communication (bits)	$(2s_l-l)k+lk$	s_lk+lk	$(s_l-l)(k_1+k_2)$
Memory requirement			
Controller storage	$(2n-1)k$	$(2n-1)k$	$(2n-1)k$
Member storage	$(h+1)k$	$(h+1)k$	$(h+1)k$

Table 7.1: Comparison of LKH, OFT and ELK for binary tree. n is the number of group members; $h = log_d(n)$ is the height of the key tree; s_l is the size of the *Common Ancestor Tree* when l leaves change. k is key size in bits. C_E, C_r and C_g are respectively the computation cost of one evaluation of encryption function E, generating a key from a cryptographically secure random source, and one evaluation of g.

$C_L = \text{PRF}^{\langle n \to k_1 \rangle}_{K^{\alpha}_{L}}(K)$. Similarly, the right child key K_R contributes k_2 bits, where $k_1 + k_2 \leq k$ (k is the length of a key in bits): $C_R = \text{PRF}^{\langle n \to k_2 \rangle}_{K^{\alpha}_{R}}(K)$. PRF is a pseudorandom function. A new key of length $k_1 + k_2$ is formed by concatenation: $C_{LR} = C_L | C_R$. The new node key K' is computed by applying a pseudorandom function, with C_{LR} as the key, to K: $K' = \text{PRF}_{C_{LR}}(K)$. ELK uses small key updates, termed *hint*, to update the keys on join events. Each member can do so independently and therefore there is no requirement for a broadcast from the controller. The protocol for member joins and leaves is detailed in [58].

ELK improves over the basic key tree protocol in that the controller does not need to broadcast key update messages to the existing group members on a join. This also leads to perfectly reliable and efficient member joins. The size of the broadcast message on member leave is also significantly smaller in ELK. This improvement in communication cost comes at the expense of higher computation at the member nodes. Table 7.1 gives a comparison of OFT and ELK with the basic LKH.

Chapter 8

Key Management Framework Analysis and Simulation

8.1 Security Analysis

8.1.1 Passive Adversary

SN Tree: We first consider a passive adversary A, who is never a group member, and look at its difficulty in computing any group key. We assume A eavesdrops on all traffic in an arbitrary subnetwork and receives all the encrypted key information and data packets. A cannot decrypt the data packets, since it does not know either the subnetwork session key or the datahiding key. A brute-force attack to find the group key takes $\Omega\left(2^k\right)$ operations where k is the length of the group key. A cannot do better than this, since it does not know any of the key encrypting keys in the tree. It cannot obtain any KEK from the key information messages because it does not know any key to decrypt even a single key information message. The framework is thus secure against a passive adversary in the subnet.

RP Tree: We assume A has the capability of listening on the satellite traffic and receives all the traffic in a complete session, i.e., A can be a passive eavesdropping RP. A still cannot decrypt the encrypted traffic, since it does not know the RP session key. It cannot obtain the session key from the RP tree key messages, because it does not have any of the keys used to decrypt the key messages. Hence here also A can only perform a brute force attack of $\Omega\left(2^k\right)$ operations.

MARS: One of the requirements for the design is that the NOC should not be able to read the multicast traffic. The MARS is located at the NOC, and plays a very important role in setting up the secure group. As such, it is important to analyze whether the MARS (and thereby, the NOC) can read the multicast traffic. If the MARS is a passive adversary, then under normal operation of the network, the multicast traffic will not reach it at all. This is because the point-to-multipoint VC that is created from a source RP to the set of receiver RPs will not include the MARS. Since we make the assumption that the underlying routing infrastructure is trusted, the point-to-multipoint VC from any source RP will not have a branch to the MARS, which therefore will not receive any multicast traffic in normal network operation.

8.1.2 Active Adversary

Let B be an active adversary, who has been a group member during some previous time period, and analyze its degree of computational difficulty in reading the group data traffic when it is not a member of the group.

SN Tree: In the tree key management protocol, when B joins the group in any subnet, it cannot derive any previous group key by doing better than exhaustive search, i.e., $\Omega\left(2^k\right)$ operations. This is because even if B has listened to and stored past group traffic, it does not get any of the decryption keys for the previous enciphered messages. The only keys it gets are the ones that are sent to it by the SKC, and precisely these keys have been updated at the time of its join.

Consider the case where B leaves the group and tries to read the group traffic after it has left. B has with it the set of keys on its key path, and the datahiding key. However, it cannot read the group traffic at a later time, since the key controller updates all the keys on the key path that B knows, including the session key, and securely transmits the updated keys to the other members using long-term keys that B does not know. B therefore cannot find the updated keys in the tree. Hence it needs to again perform a brute force attack to obtain the new session key. The datahiding key does not change, and B knows the datahiding key. However, this does not help B since it first needs to decrypt using the session key to obtain the ciphertext that is encrypted with the datahiding key.

RP Tree: Consider the scenario where B is a RP who was a member of the group at some previous time. Before B had joined the RP tree, it could not decrypt the data traffic since it did not know the group key at a previous time instant. After B joins the RP tree and collects the keys in its key path, it leaves. But once it has left, the root of the tree (assuming B was not the root), updates all the keys known to B, including the RP session key, in the RP tree. B cannot obtain the updated keys from the key message since it does not know the decryption keys used to send the updated keys to the other RPs. Therefore for B to read the traffic after it leaves, it needs to obtain the RP session key by a computationally infeasible exhaustive search. Thus the framework is secure against active adversaries.

The only time when B, as an RP, could read the data after leaving, is if B was the root of the RP tree. Then for the interval of time it takes the new root to setup a new tree, the group traffic would continue to be encrypted using the old RP session key, allowing B access to it.

Note that B as an RP could have obtained only the ciphertext of the data, encrypted with the datahiding key. The purpose of the datahiding key is precisely to prevent the RPs from reading the actual data traffic, because our trust model does not allow us to trust the routers in the tree. The datahiding key would also prevent the MARS from reading the traffic.

MARS: What would happen if we consider the MARS to be an active adversary? We note that the MARS can easily join any multicast group - it can simply add its ATM address to the list of addresses for the multicast group, and sends to the source RPs. The point-to-multipoint VCs created by the source RPs will therefore include a branch to the MARS. Consequently the MARS will be able to receive all the key traffic on the RP tree, and all the encrypted multicast traffic. But even under this situation, the MARS will not be able to read the multicast data. This is because the multicast traffic is first encrypted with the datahiding key, to which no RP nor the MARS has access. Therefore even if the MARS is able to partially decrypt the multicast traffic using the

RP tree key, it will not be able to decrypt further. Hence the data is secure even if the MARS deliberately adds itself to the multicast groups to receive the data. However, it is to be noted that under the assumption that the routing framework is secure, the MARS would operate normally and this scenario will not arise.

Our tiered tree framework therefore allows secure transmission of multicast traffic across subnetworks, allowing only privileged group members to receive the data. The framework also prevents other entities in the multicast distribution tree from reading the traffic.

8.2 Cost Analysis

We compute the cost for communication and storage for the basic key tree scheme: LKH in the overlay and in each subnet.

Notation

- n is the total number of members in the group.
- n_1 is the number of RPs, n_2 is the number of members in each subnet.

  $n_1 * n_2 = n$.
- d_1, h_1 are respectively the degree and height of the RP tree, $h_1 = log_{d_1}(n_1)$.
- d_2, h_2 are respectively the degree and height of the SN tree, $h_2 = log_{d_2}(n_2)$.
- k_p is the length of a public key.

	RP Root	SKC
Tree setup	$(n_1 - 1)\, k_p + \frac{d_1(n_1-1)}{d_1-1} k_s$	$\left(n_2 + \frac{d_2(n_2-1)}{d_2-1} + 1\right) k_s$
Member join to existing group in subnet	0	$(d_2 h_2 + 1)\, k_s + k_s$
Adding a subnet to existing group	$(d_1 h_1 + 1)\, k_s + k_p$	$\lceil \left(n_2 + \frac{d_2(n_2-1)}{d_2-1} + 1\right) k_s \rceil$
Evicting a member from subnet	0	$(d_2 h_2 - 1)\, k_s$
Evicting a subnet	$(d_1 h_1 - 1)\, k_s$	0

Table 8.1: Communication Cost in Tiered Tree Based Key Management with LKH algorithm.

- k_s is the length of a symmetric key.

The results are derived by applying the cost metrics of the basic LKH to the RP tree and the SN tree, and by aggregating the two. Table 8.1 shows the communication overhead for the RP tree and SN tree individually, while 8.2 gives the total communication cost in the network.

In every case above, the RP tree root takes advantage of the broadcast capabilities of the network to send the key messages in one round. In the subnetworks, the SKC sends the messages to the multicast tree and therefore takes one round for updates (and the additional unicasts to the joining members for joins). The communication cost for

	Total Cost
Tree setup	$(n_1-1)\,k_p+\frac{d_1(n_1-1)}{d_1-1}k_s+n_1\left(\left(n_2+\frac{d_2(n_2-1)}{d_2-1}+1\right)k_s\right)$
Member join to existing group in subnet	$(d_2h_2+2)\,k_s$
Adding a subnet to existing group	$k_p+(d_1h_1+1)\,k_s+\lceil\left(n_2+\frac{d_2(n_2-1)}{d_2-1}+1\right)k_s\rceil$
Removing a member from subnet	$(d_2h_2-1)\,k_s$
Removing a subnet	$(d_1h_1-1)\,k_s$

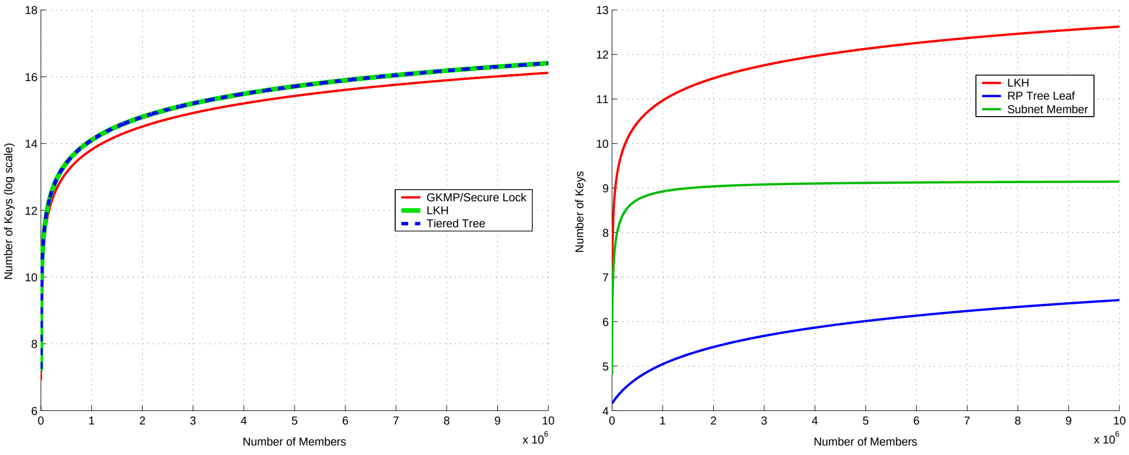
Table 8.2: Total Communication Cost in Tiered Tree Based Key Management with LKH algorithm.

RP root	SKC	RP	Member
$\lceil\frac{(d_1n_1-1)}{d_1-1}k_s+n_1k_p\rceil$	$\lceil\frac{(d_2n_2-1)}{d_2-1}k_s+2\rceil$	$\lceil h_1+2\rceil$	$\lceil h_2+2\rceil$

Table 8.3: Storage Cost in Tiered Tree Based Key Management with LKH algorithm.

multiple members addition or revocation depends to a great degree on the placement of the members in the tree. Since the placement is not determinate, we leave out the communication costs for the case of multiple members. The figures for the communication cost are only approximate. In most of the calculations, we do not rigorously consider the fact that the root of the RP tree itself is a group member; hence all the RP tree key update messages are sent to only (n_1-1) members.

Table 8.3 gives the total storage cost in the framework, using basic LKH algorithm. The two additional keys at the SKC is due to the datahiding key, and the shared secret with the local RP. The single additional key storage at the RP is due to the subnetwork session key, while at the member is due to the datahiding key. The expressions consider that the RP root stores the public keys of all subscribed RPs, though the public keys are not needed except for the initial setup. One can compare tables 8.2 and 8.3 to table 6.1 to analyze the advantages of our tiered key management framework, even when we consider basic LKH and not any of its optimizations. Figures 8.1 to 8.3 show plots comparing the different protocols to Tiered Tree using basic LKH. We consider group size varying from 10^3 to 10^7; the number of subnetworks considered in Tiered Tree range from 20 to 500; the number of members in a subnetwork therefore range from 50



(a) Storage required at the controller. For Tiered Tree, we consider the total storage for RP root and all SKCs.

(b) Storage required in individual member nodes. For Tiered Tree, we consider the storage both at the RP leaf and in subnetwork members.

Figure 8.1: Comparison of Key Management Schemes: Total Storage Requirement

to $20 * 10^3$, with members distributed uniformly across subnetworks. We consider quaternary trees for LKH and Tiered Tree. We do not consider the probability of member join and leave in our computations. In several cases, the plots of LKH and Tiered Tree overlap, as do those of HFK and GDH, and Secure Lock and GKMP. We could not plot the storage requirements for KPS or Broadcast Encryption (basic scheme); they blow up even for 10^3 members.

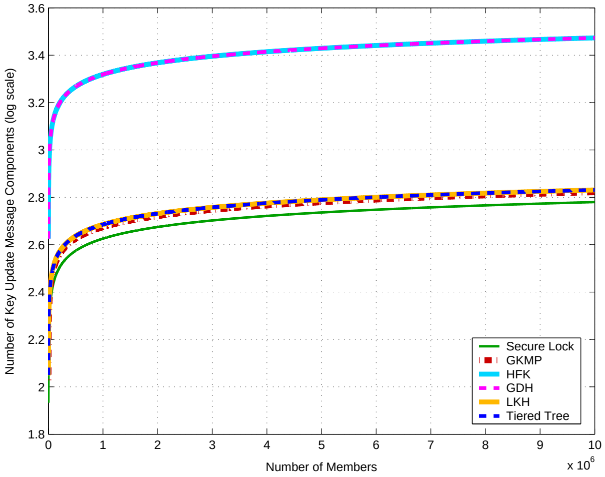
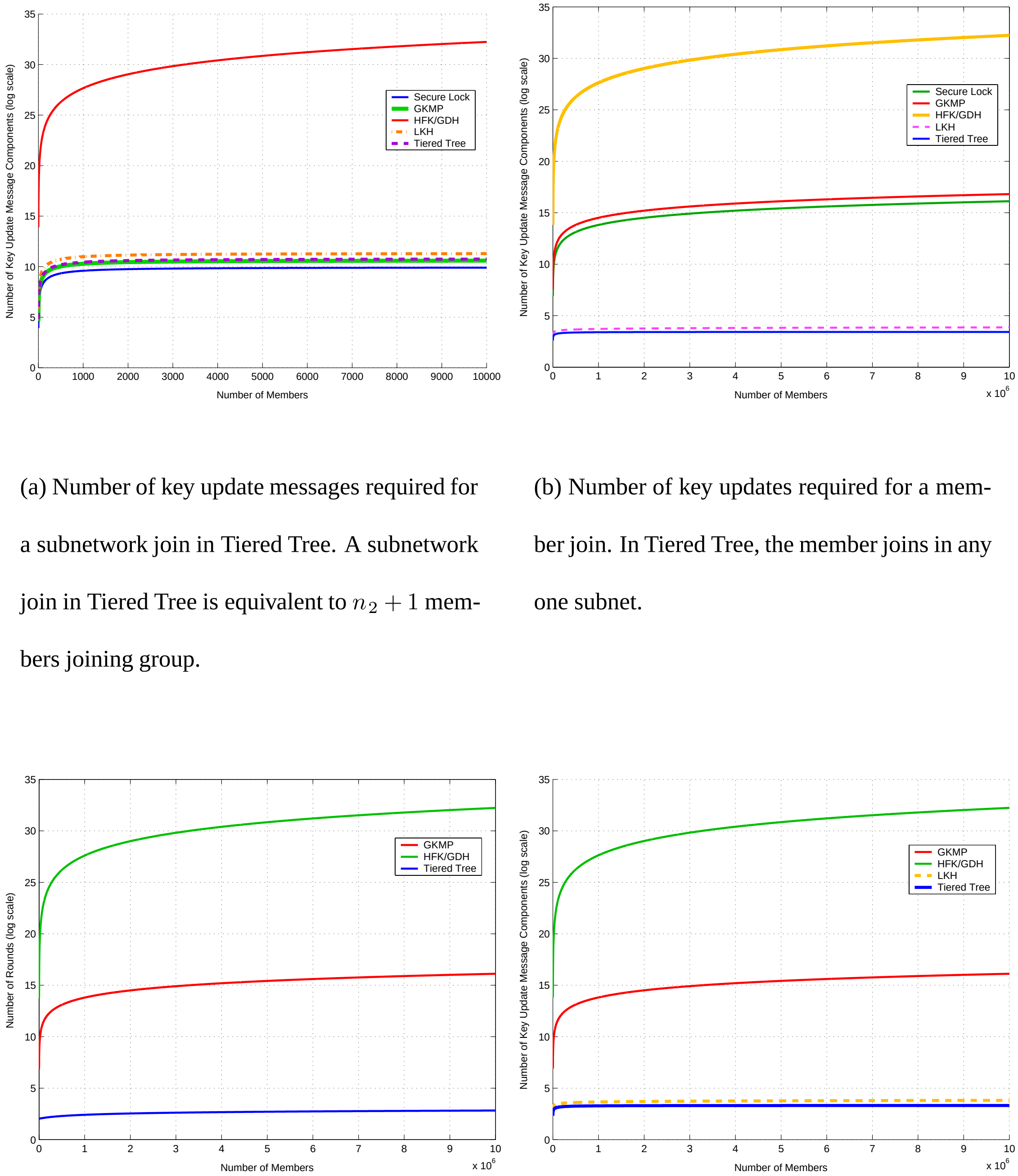


Figure 8.2: Comparison of Key Management Schemes: Total Number of Messages Required for Setup.



(a) Number of key update messages required for a subnetwork join in Tiered Tree. A subnetwork join in Tiered Tree is equivalent to $n_2 + 1$ members joining group.

(b) Number of key updates required for a member join. In Tiered Tree, the member joins in any one subnet.

(c) Number of key update messages required for a subnetwork leave in Tiered Tree. A subnetwork leave in Tiered Tree is equivalent to $n_2 + 1$ members leaving group.

(d) Number of key updates required for a member leave. In Tiered Tree, the member leaves from any one subnet.

Figure 8.3: Comparison of Key Management Schemes: Total Key Updates for Join and Leave

8.3 Simulation

We have verified the validity and feasibility of our framework through simulations using OPNET Modeler, version 9.0[39]. We used the multicast simulation setup from chapter 5 and added our security features to it.

- The network configuration for the security framework simulation has 31 subnetworks; there are 49 nodes in each subnetwork, making a total of 1519 nodes in the network.

- The security module in each RP is located at the IP Adaptation Layer. The security module has functionality to manage the key tree if the RP is selected as a root; else it stores the keys as a leaf entry. Provisions are made to merge the leaf entry with the root entry if the role of a RP changes from leaf to root.

  For every multicast data packet received by a RP, it checks whether it has the correct RP tree key and subnetwork session key, for performing decryption and re-encryption, respectively. If both keys are present, the RP forwards the packet, else the packet is dropped.

- The key management server in each subnetwork is modeled by a Ethernet server from the Opnet library, with the key management functionality added to it. The key management module is located at the Transport Protocol Adaptation Layer - UDP Interface, which is a sub-layer of the application layer. All the group keys are managed and stored by the key management module.

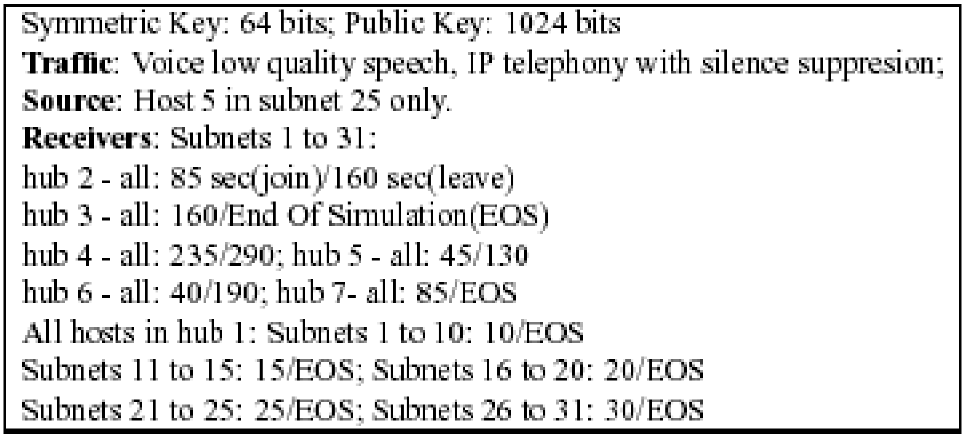
- For the hosts who take part in the multicast group, we selected Ethernet workstations. The security module in each end-host is added at the Transport Protocol Adaptation Layer - UDP Interface. The end-hosts contact the key server before they join a multicast group, or send to a multicast group. The keys obtained from the key server are processed by the security module and stored there. Upon traffic reception, every data packet is processed by the security module, which checks if the session key and the datahiding key for the group are correct. If not, the packet is dropped.

8.3.1 Results

We ran simulations for both one-to-many traffic and many-to-many traffic. In each case, we considered 64 bit symmetric keys and public key size of 1024 bits. For subnetwork key management, we assumed that a shared secret already exists between the SKC and all the hosts, and also the RP. The public keys are used for the initial encryption in the RP tree; subsequent messages in the RP tree are encrypted using 64 bit symmetric keys.

8.3.1.1 One-to-Many Traffic Scenario

In the one-to-many case, a single multicast group has a single source - host 5 in subnetwork 25. Each subnetwork has 48 receivers for the multicast group; therefore there are 1488 receivers in all. The receivers join and leave the group dynamically, as given in the scenario details in figure 8.4. We ran the simulation for 300 seconds.



Symmetric Key: 64 bits; Public Key: 1024 bits
Traffic: Voice low quality speech, IP telephony with silence suppresion;
Source: Host 5 in subnet 25 only.
Receivers: Subnets 1 to 31:
hub 2 - all: 85 sec(join)/160 sec(leave)
hub 3 - all: 160/End Of Simulation(EOS)
hub 4 - all: 235/290; hub 5 - all: 45/130
hub 6 - all: 40/190; hub 7- all: 85/EOS
All hosts in hub 1: Subnets 1 to 10: 10/EOS
Subnets 11 to 15: 15/EOS; Subnets 16 to 20: 20/EOS
Subnets 21 to 25: 25/EOS; Subnets 26 to 31: 30/EOS

Figure 8.4: Key Management: One-to-Many Simulation Scenario

The total overhead in terms of number of key information packets and bytes transferred is given in figure 8.5 (in all the graphs, the x-coordinate is the simulation time in minutes). The packets per second metric actually show the number of key components; all the components are broadcast in a single message in the RP tree. In the SN tree, since we do not have too many receivers and the links are fast Ethernet links, we unicast the key management messages from the SKC to the members and the RP.

The root of the RP tree is the RP in the subnet of the source, i.e., RP in subnet 25. The total RP tree traffic sent by the RP in subnet 5 is shown in figure 8.6(a). The figure indicates that at the level of the overlay, there is very little dynamism. This is because in all the subnets, at least one receiver remains as a group member throughout the duration of the sender traffic. Figure 8.6(b) shows the total key traffic sent by the SKC in subnet 5. As can be seen, within the subnet the dynamism of joins and leaves is much higher.

Our tiered framework effectively “hides” the dynamics of member joins and leaves

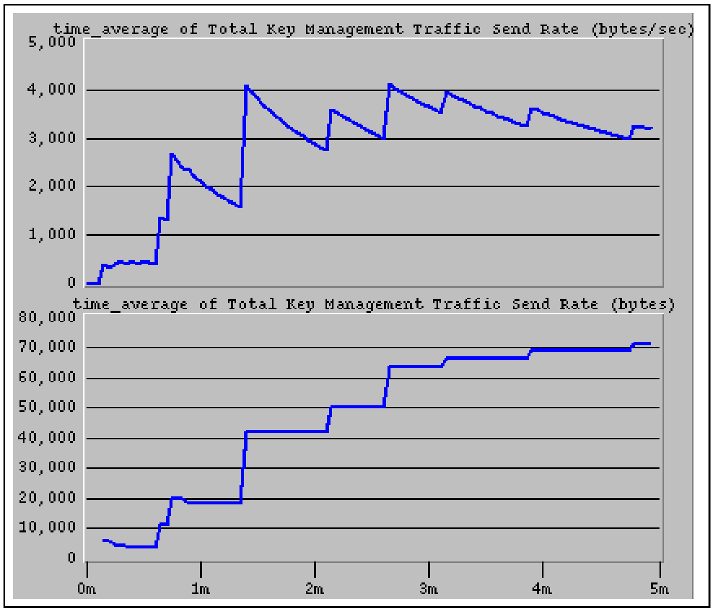
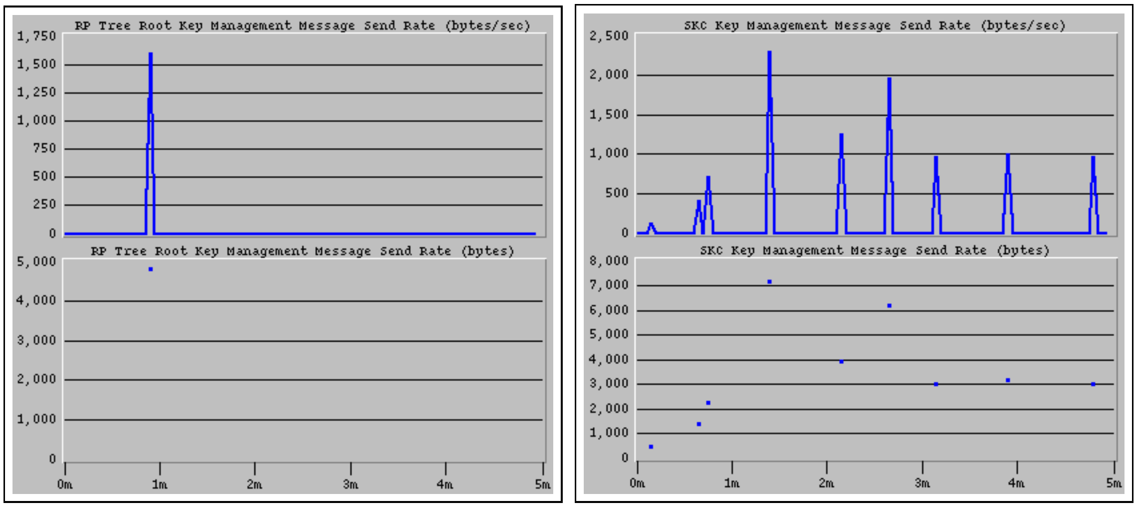


Figure 8.5: Tiered Tree Framework - One-to-many: Total Key Management Traffic Sent

at the subnet level from affecting other subnets. This is made very clear by figure 8.7; while the RP of subnet 5 receives frequent key information updates from the local SKC, it does not affect the RP tree. The savings in the satellite links due to using a tiered tree compared to a single tree is given in figure 8.8, which shows the comparison between the total key traffic and the key traffic on the RP tree. In the tiered framework, the security traffic in the satellite overlay is the traffic on the RP tree. In the absence of the tiered framework, the security traffic in the satellite overlay would have been the total key traffic shown in the graphs.

8.3.1.2 Many-to-Many Traffic Scenario

There are three IP multicast groups in the network, each spread across 31 subnetworks. Each group has 10 sources in 10 subnetworks, one source in each subnetwork, as



(a) Total RP Tree Key Traffic Sent

(b) Total SN Tree Key Traffic in Subnet 25

Figure 8.6: Tiered Tree Framework - One-to-Many: Traffic in RP Tree and SN Tree

detailed in figure 8.9. Each group has 35 receivers in each of the 21 subnetworks that have no sources for the group, and 34 receivers in each of the 10 subnetworks that have sources for the group. Therefore each group has a total of 1075 receivers.

The simulation was run for 300 seconds.

Figure 8.10 gives the total key management overhead for many-to-many traffic, for all the three groups (in all the graphs, the horizontal scale is the simulation time in minutes).

The RPs that were selected by MARS as the root of the RP trees for the three groups are:

- RP of subnet 5 for group 224.25.25.25 (group A),
- RP of subnet 11 for group 224.0.1.1 (group B), and,

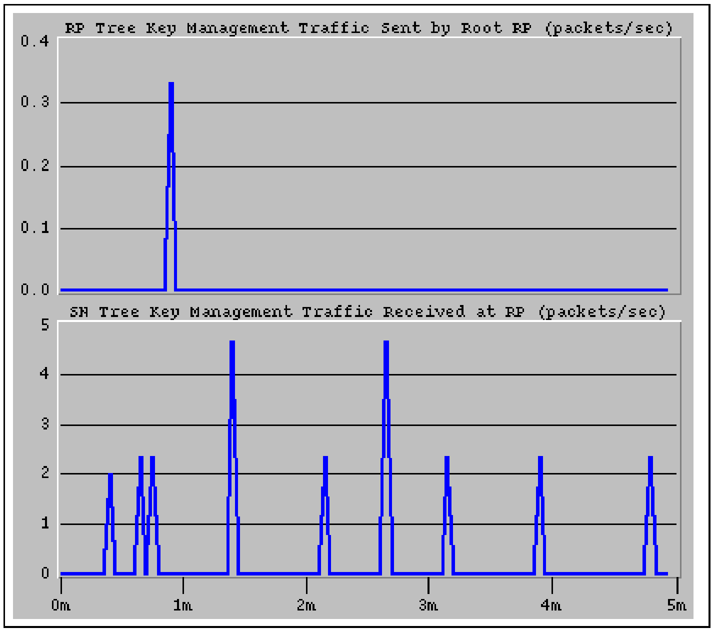
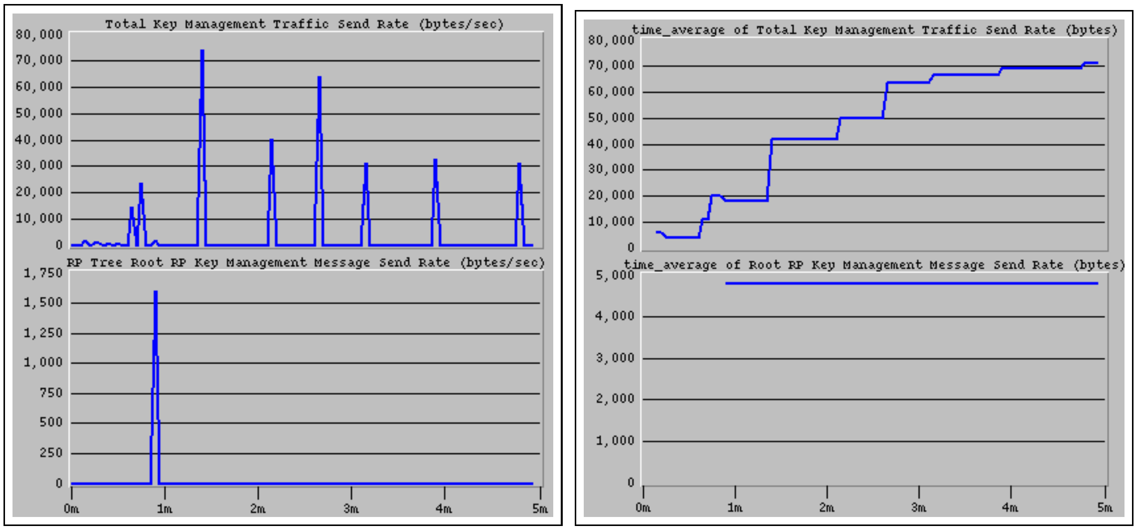


Figure 8.7: Tiered Tree Framework - One-to-Many: Total Key Traffic Received and Sent by Root RP

- RP of subnet 23 for group 224.0.5.5 (group C)

Note that the above RPs are leaves in the RP trees for the groups for which they are not the RP tree root. Thus in our framework, the key management in the overlay can be distributed among different RPs for different groups. Figure 8.11 shows the total key information traffic sent by the three root RPs for the three multicast groups, compared to the total key information traffic received by them from their local SKCs. Note that the total key information traffic received by the RPs from the local SKC is the traffic for all the three multicast groups, and not only the group for which the RP is the root RP. The RP is a leaf RP for the other two groups. From figure 8.11, we can see that even though the group dynamics are high, the amount of message exchanges are very

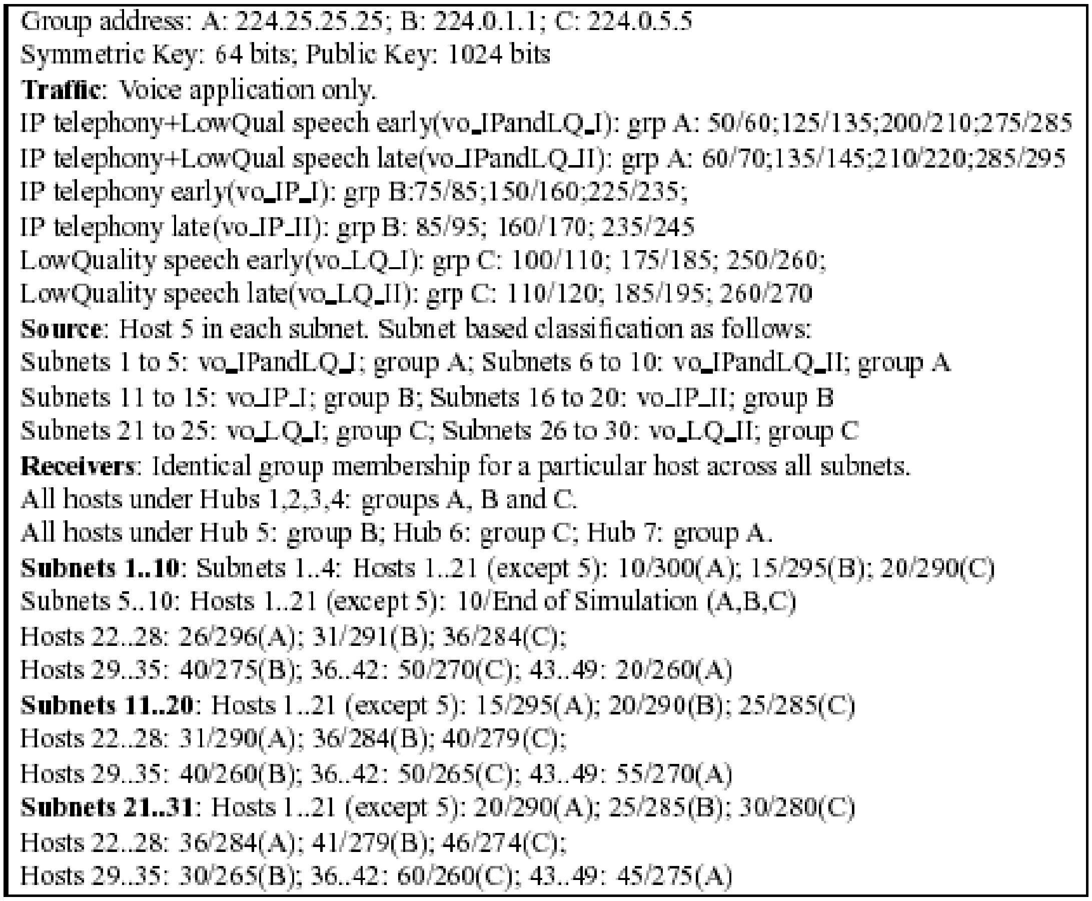


(a) Total Key Traffic Sent vs. RP Tree Traffic (bytes/sec)

(b) Total Key Traffic Sent vs. RP Tree Traffic (byte average)

Figure 8.8: Tiered Tree Framework - One-to-Many: Savings in Tiered Tree Key Management

few in the RP tree. This is because the RPs remain subscribed to a group as long as there is at least one member in its local subnetwork sending to or receiving from the group; the frequency of joins and leaves in the subnetwork is transparent to the RP tree. This is precisely our intention, to minimize the cost of message exchanges over the satellite links. The figure also illustrates another important point of our key management scheme, namely, scalability. The effect of frequent member joins and leaves in one subnetwork remains localized within the subnetwork, and does not affect the group dynamics in other subnetworks. Therefore subnetworks where the group membership is relatively long-term is free of the overhead of frequent key update



Group address: A: 224.25.25.25; B: 224.0.1.1; C: 224.0.5.5
Symmetric Key: 64 bits; Public Key: 1024 bits
Traffic: Voice application only.
IP telephony+LowQual speech early(vo_IPandLQ_I): grp A: 50/60;125/135;200/210;275/285
IP telephony+LowQual speech late(vo_IPandLQ_II): grp A: 60/70;135/145;210/220;285/295
IP telephony early(vo_IP_I): grp B:75/85;150/160;225/235;
IP telephony late(vo_IP_II): grp B: 85/95; 160/170; 235/245
LowQuality speech early(vo_LQ_I): grp C: 100/110; 175/185; 250/260;
LowQuality speech late(vo_LQ_II): grp C: 110/120; 185/195; 260/270
Source: Host 5 in each subnet. Subnet based classification as follows:
Subnets 1 to 5: vo_IPandLQ_I; group A; Subnets 6 to 10: vo_IPandLQ_II; group A
Subnets 11 to 15: vo_IP_I; group B; Subnets 16 to 20: vo_IP_II; group B
Subnets 21 to 25: vo_LQ_I; group C; Subnets 26 to 30: vo_LQ_II; group C
Receivers: Identical group membership for a particular host across all subnets.
All hosts under Hubs 1,2,3,4: groups A, B and C.
All hosts under Hub 5: group B; Hub 6: group C; Hub 7: group A.
Subnets 1..10: Subnets 1..4: Hosts 1..21 (except 5): 10/300(A); 15/295(B); 20/290(C)
Subnets 5..10: Hosts 1..21 (except 5): 10/End of Simulation (A,B,C)
Hosts 22..28: 26/296(A); 31/291(B); 36/284(C);
Hosts 29..35: 40/275(B); 36..42: 50/270(C); 43..49: 20/260(A)
Subnets 11..20: Hosts 1..21 (except 5): 15/295(A); 20/290(B); 25/285(C)
Hosts 22..28: 31/290(A); 36/284(B); 40/279(C);
Hosts 29..35: 40/260(B); 36..42: 50/265(C); 43..49: 55/270(A)
Subnets 21..31: Hosts 1..21 (except 5): 20/290(A); 25/285(B); 30/280(C)
Hosts 22..28: 36/284(A); 41/279(B); 46/274(C);
Hosts 29..35: 30/265(B); 36..42: 60/260(C); 43..49: 45/275(A)

Figure 8.9: Key Management: Many-to-Many Simulation Scenario

messages due to volatility in membership elsewhere. The scheme can thus scale to large number of members spread across multiple subnetworks. The savings in terms of bytes of key information sent per second is illustrated in figure 8.12, which compares the total key information sent for all the groups in the RP trees and all the SN trees, to the total key information sent on the RP trees only. As the graph shows, the resource savings on the satellite links is substantial using the tiered tree scheme.

For completeness, we show the key information sent and received by randomly selected hosts in the network. Graph 8.13(a) show the total key requests sent by hosts 1

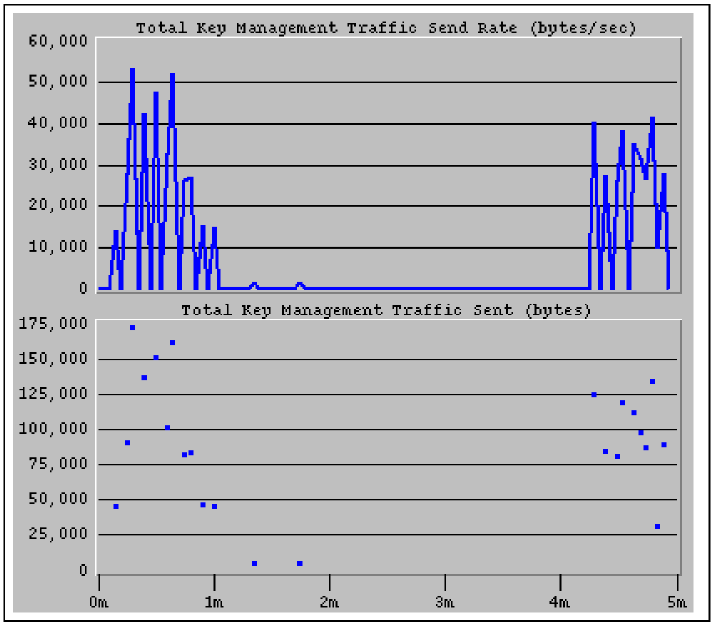


Figure 8.10: Tiered Tree Framework - Many-to-Many: Total Key Management Overhead for All Three Multicast Groups

and 45 in subnet 1, compared to the total key information received by them from their local SKC. Host 1 is a member of all three groups in the scenario, and remains a group member for the entire duration of group existence. Host 45 is a member of only group A, and its membership is for the shortest duration amongst all group A members in the subnetwork. Hence host 1 receives significantly more traffic than host 45. This indirectly demonstrates that our scheme is secure, i.e., a group member receives key traffic only as long as it is subscribed to the group, and does not receive any meaningful key traffic when it is not a member.

Graph 8.13(b) show the total key requests sent by three hosts in the same subnet 25 who belong to different groups. Host 25 receives traffic for all three groups, but in comparison to other subnetwork hosts who subscribe to all three groups, it remains a

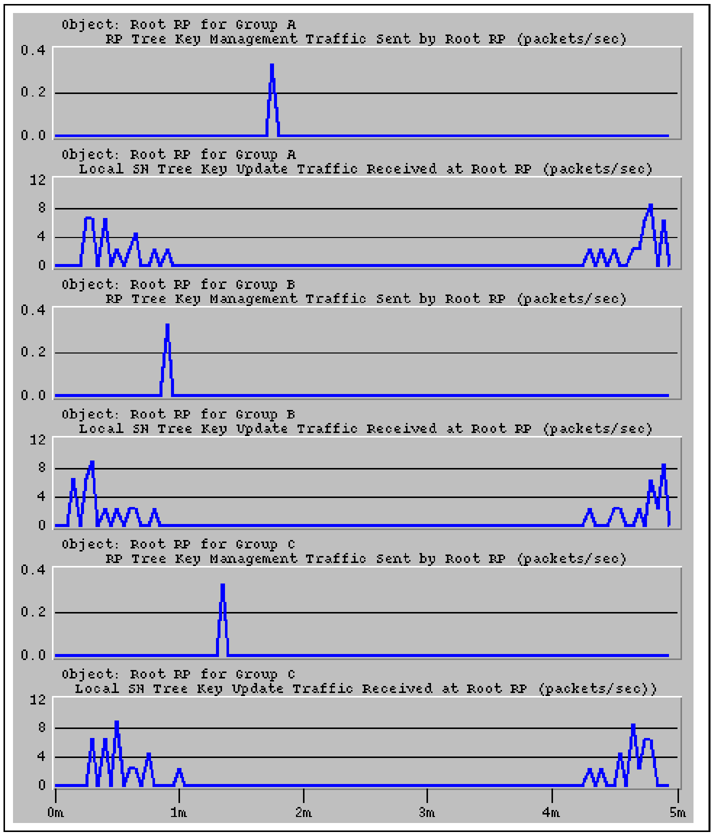


Figure 8.11: Tiered Tree Framework - Many-to-Many: RP Tree Traffic Sent vs. SN Tree Traffic Received by Root RPs

group member for the different groups for the shortest period of time. Host 35 receives for group B only, and host 40 is a member of group C only. The amount of key information received by each depends on their join/leave times, and also on the dynamics of other member joins and leaves for their respective groups.

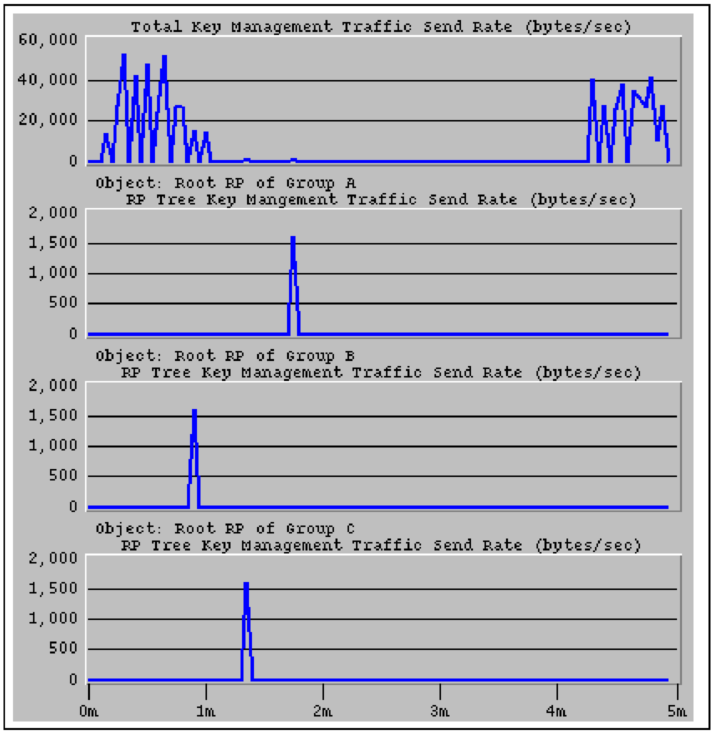
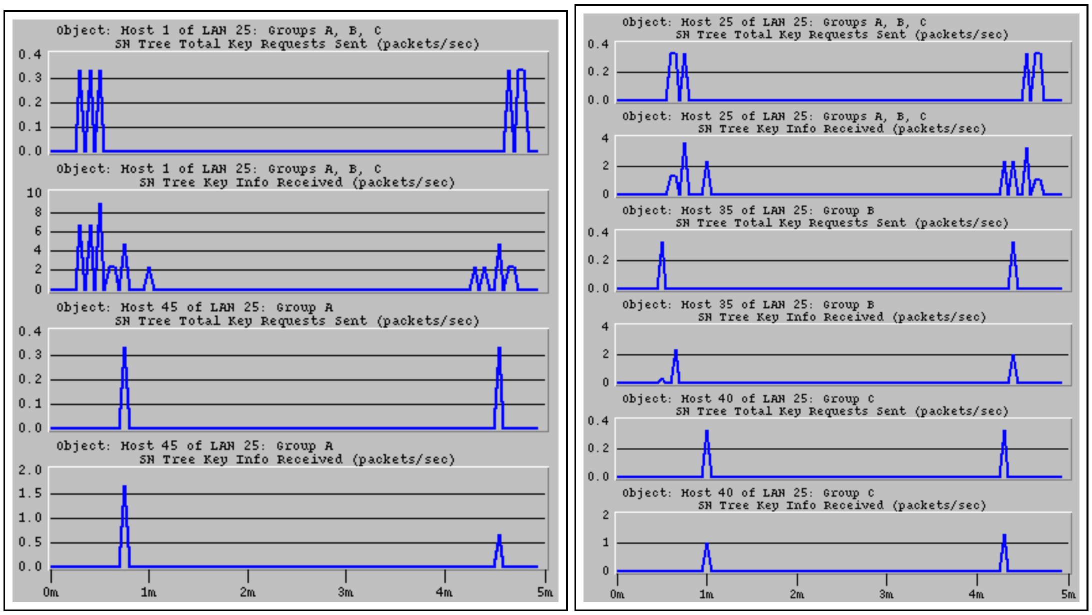


Figure 8.12: Tiered Tree Framework - Many-to-Many: Total Key Traffic vs. RP Tree Traffic for 3 Groups



(a) SN Tree Traffic for Hosts 1 and 45 in Subnetwork 25

(b) SN Tree Traffic for Hosts 25, 35 and 40 in Subnetwork 25

Figure 8.13: Tiered Tree Framework - Many-to-Many: Key Management Traffic for Selected Group Members in one LAN

Chapter 9

Conclusions and Future Work

In this chapter, we first mention some notable features of the routing framework and the key management protocol. We follow up the discussion with an overall conclusion combining our routing and key management frameworks. In the final section, we outline the problems that would require additional work in the future.

9.1 Comments on the Routing Framework

The routing framework proposed here avoids the problem of sub-optimal placement of RPs which would happen in such a large network if standard PIM-SM is used. This has the advantage that the amount of multicast control traffic over the satellite channels is reduced significantly. If standard PIM-SM is used, with the RP for a multicast group located in a remote subnetwork or the NOC, then every REGISTER message would have to be over the satellite channels, even if there is no receiver in other locations. This would be wasteful use of the satellite bandwidth, and also introduce additional delay. Also, the data traffic would have to flow to the RP since the shared RP tree would remain active always. This would happen even if there are no receivers in any remote location. Our framework solves this problem very effectively by localizing the PIM-SM control messages and data traffic to the subnetworks. The amount of MARS control traffic sent over the satellite links is much less, and done once when the group is set up or torn down, instead of for every source. Also, the data traffic is sent over the links if and only if there are receivers in other locations.

9.2 Comments on the Key Management Framework

It is interesting to note some of the characteristics of the tiered key management framework.

- The framework is essentially a generic design; different types of key management algorithms can be applied in each logical grouping. Our focus is very large groups; hence we considered tree based algorithms because of their scalability and robustness for large groups sizes. However, tree based algorithms can be inefficient if the group is small. If the subnetworks in a group are limited and remain static, then GKMP might be a good candidate. Likewise, if the total members in a subnetwork are small, then we can use GKMP or HFK in a subnet, for example.

- Our framework "hides" the dynamism of member joins and leaves in a subnetwork from other parts of the network. Thus it satisfies the *1-affects-n* property[60] of key management.

- One issue in our design is the generation of the datahiding key for a group. This requires the SKCs of all subnetworks in the group to be in agreement about the datahiding key. We have not considered the key management for the datahiding key, since that is a one time message exchange. A simple mechanism for this to happen is for the SKC in the root RP subnetwork to generate the key and send it to the SKCs in the other subscribed subnetworks; the generating SKC can know of the other subnetworks in a message from the root RP. This would require additional message exchanges between the root RP and the local SKC, and between the generating SKC and other subscribed SKCs. The SKCs should also be aware of each other's address and have secure channels established between them, but this can be done at the time of network setup.

  Note that we need the datahiding key not to prevent unauthorized hosts from reading the multicast traffic, but to prevent the RPs from reading the traffic. Since we already trust the RPs to forward data securely, in many scenarios we might also trust the RPs with the un-encrypted contents. In such cases, the datahiding key is not needed.

- Comparing the costs in our scheme using LKH trees, to the single tree LKH protocol, we see that there is no major difference in setup, join in terms of communication overhead, or in storage. A case can hence be made to use a single LKH tree, which would be a less complex design. However, the different subnetworks might be independent domains, such as company networks, and might follow different security policies. Reconciling the security policies across the subnetworks to build a single LKH might be a harder task than our tiered framework. Also, a single LKH would suffer from the *1-affects-n* scalability problem; the probability of updates in the keys stored at a member would be much higher due to the dynamics of member joins and leaves overall. For a member joining/leaving in one subnetwork, the keys would be updated at a member in a remote subnetwork. The key management communication over the satellite links would be much more frequent.

- Another point to note is that our framework “fuses” key management at the application layer with key management at the network layer. In the hosts and the SKC, the security module is a part of the application layer. However, in the RPs the multicast traffic does not go up to the application layer; the RPs operate on the multicast IP packets, and therefore the security module is located at the network layer. As our design and simulations show, the above can co-exist well and seamlessly perform secure data transmission.

9.3 Conclusions

In this work we have proposed a framework for IP multicast routing in a wide-area satellite network that has terrestrial Ethernet-based networks connected via ATM-based satellite links, and added a key management framework to the proposed network architecture for secure data transfer.

We selected PIM-SM for the intra-domain multicast routing in the terrestrial networks; and IP-over-ATM multicast using MARS and VC mesh for inter-domain multicast routing over the satellite channels. We have proposed modifications to the protocols to adapt them to our network. Specifically, we have introduced the concept of active peer RPs for the same PIM-SM multicast group, one RP per subnetwork. We have also made additions to the RP functionality to allow seamless end-to-end multicast in a group spread across different areas. Our additions are lightweight, and do not involve any major change to existing RP functions. We have also used the MARS with VC mesh concept to do inter-domain multicasting, which differs from the "traditional" use of MARS for intra-domain multicasting. We have performed simulations of our framework, and have shown that it performs well, and compares favorably to other models. Our framework makes optimal use of the expensive satellite links, and the satellite broadcast capability, and removes the drawback that arises in PIM-SM due to the sub-optimal placement of the RP.

For the design of the key management framework, we have analyzed the issues involved, discussed existing protocols and shown that most of them do not scale to large groups that will have dynamic member joins and leaves. Consequently we have designed a framework for key management for large groups in our satellite network architecture. Our design is scalable and efficient and very well suited for the unique network architecture that we consider.

9.4 Future Work

We have not considered channel errors in the multicast framework design, since the work is limited to the network layer and below. However, channel errors are very important in geostationary satellite networks. Countering the effect of channel errors requires mechanisms for reliable transmission to be added to the multicast framework. We are therefore working on the design of reliable transport protocols for the multicast traffic in the hybrid satellite network.

The design of the key management framework has not explicitly detailed how the datahiding key is distributed across the subnetworks. Since the datahiding key is long-term, one choice is to do this offline. However, we are looking at mechanisms that would efficiently distribute the datahiding key online, and update it online if needed.

Ensuring data confidentiality is one aspect of secure multicast; authenticating the source of the data is another important aspect to protect against attacks due to unauthorized messages. We have not considered source authentication in our security design. Several efficient schemes for multicast source authentication have been proposed in the research community. [62] will be well-suited for our network, with the modifications that have been proposed in [63] for ad hoc networks. Source authentication with the modifications for broadcast networks remains to be investigated in our framework.

BIBLIOGRAPHY

[1] G. Akkor, M. Hadjitheodosiou, and J. S. Baras. ”IP Multicast via Satellite: A Survey”. Technical Report CSHCN TR 2003-1, Center for Satellite and Hybrid Communication Networks, University of Maryland College Park, 2003.

[2] S. E. Deering. *”Host Extensions for IP Multicasting”*. Internet RFC 1112, August 1989.

[3] T. Matsumoto and H. Imai. ”On the KEY PREDISTRIBUTION SYSTEM: A Partical Solution to the Key Distribution Problem”. *Advances in Cryptology - CRYPTO ’87, Lecture Notes in Computer Science, LNCS,* 293:185–193, 1988.

[4] A. Fiat and M. Naor. ”Broadcast Encryption”. *Advances in Cryptology - CRYPTO ’93, Lecture Notes in Computer Science, LNCS,* 773:480–491, 1994.

[5] M. Steiner, G. Tsudik, and M. Waidner. ”Diffie-Hellman Key Distribution Extended to Group Communication”. *Proceedings of the 3rd ACM Conference on Computer and Communications Security - CCS ’96,* March 1996.

[6] R. Poovendran. *”Key Management for Secure Multicast Communication”*. PhD thesis, University of Maryland College Park, 1999.

[7] C. K. Wong, M. Gouda, and S. S. Lam. ”Secure Group Communications Using Key Graphs”. *IEEE/ACM Transactions on Networking*, 8(1):16–30, February 2000.

[8] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas. ”Multicast Security: A Taxonomy and Some Efficient Constructions”. *Proceedings of INFOCOMM ’99*, March 1999.

[9] M. Ramalho. ”Intra- and Inter-domain Multicast Routing Protocols: A Survey and Taxonomy”. *IEEE Communications Surveys and Tutorials*, 3(1), First Quarter 2000.

[10] B. Cain, S. Deering, I. Kouvelas, B. Fenner, and A. Thyagarajan. *”Internet Group Management Protocol, Version 3”*. Internet RFC 3376, October 2002.

[11] M. Koyabe and G. Fairhurst. ”Wide Area Multicast Internet Access via Satellite”. Technical report, University of Aberdeen, http://www.erg.abdn.ac.uk/users/gorry/ipmulticast/docs/2000-MK-ECSC-5.p%df, 2000.

[12] E. J. Fitzpatrick. ”Spaceway System Summary”. *Space Communications*, (13):7–23, 1995.

[13] ”Introduction to the MBone”. http://www-itg.lbl.gov/mbone/.

[14] S. Paul. *”Multicasting on the Internet and Its Applications”*. Kluwer Academic Publishers, USA, 1998.

[15] J. Moy. *”Multicast Extensions to OSPF (MOSPF)”*. Internet RFC 1584, 1994.

[16] J. Moy. *”OSPF, version 2”*. Internet RFC 1583, 1994.

[17] T. Pusateri. *”Distance Vector Multicast Routing Protocol”*. Internet Draft, draft-ietf-idmr-dvmrp-v3-05.txt, October 1997.

[18] C. Hedrick. *”Routing Information Protocol”*. Internet RFC 1058, June 1988.

[19] Y.K. Dalal and R.M. Metcalfe. ”Reverse Path Forwarding of Broadcast Packets”. *Communications of the ACM*, 21(12):1040–1048, December 1978.

[20] A. Ballardie, P. Francis, and J. Crowcroft. ”Core Based Trees (CBT): An Architecture for Scalable Inter-Domain Multicast Routing”. *Proceedings of ACM SIGCOMM ’93*, September 1993.

[21] A. Ballardie, S. Reeve, and N. Jain. *”Core Based Tree (CBT) Multicast - Protocol Specification”*. Internet Draft, 1996.

[22] C. Shields and J. J. Garcia-Luna-Aveces. ”The Ordered Core Based Tree Protocol”. *Proceedings of INFOCOM 1997*, 1997.

[23] A. Ballardie, S. Reeve, and N. Jain. *”Core Based Trees (CBT, v2) Multicast Routing - Protocol Specification”*. Internet Draft, 1996. work in progress.

[24] S.E. Deering, D. Estrin, D. Farinacci, V. Jacobson, C-G Liu, and L. Wei. ”The PIM Architecture for Wide-Area Multicast Routing”. *IEEE/ACM Transactions on Networking*, 4(2):153–162, April 1996.

[25] S.E. Deering et al. *”Protocol Independent Multicast (PIM), Dense Mode Protocol: Specification”*. Internet Draft, 1994. work in progress.

[26] B. Fenner, M. Handley, H. Holbrook, and I. Kouvelas. *”Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)”*. Internet Draft, draft-ietf-pim-sm-v2-new-05.ps, March 2002.

[27] M. Parsa and J. J. Garcia-Luna-Aveces. ”A Protocol for Scalable Loop-Free Multicast Routing”. *IEEE Journal on Selected Areas in Communications*, 15(3):316–331, 1997.

[28] J. J. Garcia-Luna-Aveces. ”Loop-Free Routing Using Diffusing Computations”. *IEEE/ACM Transactions on Networking*, 1(1):130–141, 1993.

[29] A.S. Thyagarajan and S.E. Deering. ”Hierarchical Distance-Vector Routing for the MBone”. *Proceedings of ACM SIGCOMM ’95*, October 1995.

[30] M. Handley, J. Crowcroft, and I. Wakeman. *”Hierarchical Protocol Independent Multicast”*. University College London, November 1995. work in progress.

[31] D. Thaler, D. Estrin, and D. Mayer. *”Border Gateway Multicast Protocol (BGMP): Protocol Specification”*. IETF Draft, 1999. work in progress.

[32] M. Handley and D. Estrin D. Thaler. *”The Internet Multicast Address Allocation Architecture”*. IETF Draft, 1997. work in progress.

[33] T. Bates, R. Chandra, D. Katz, and Y. Rekhter. *”Multiprotocol Extensions for BGP-4”*. Internet Draft, draft-ietf-idr-bgp4-multiprotocol-01.txt, September 1997.

[34] ATM Forum. *”ATM User-Network Interface Specification Version 3.0”*. Prentice Hall, Englewood Cliffs, NJ, September 1993.

[35] ATM Forum. *”ATM User-Network Interface Specification Version 3.1”*. Prentice Hall, Englewood Cliffs, NJ, June 1995.

[36] G. Armitage. ”IP Multicasting over ATM Networks”. *IEEE Journal on Selected Areas in Communications*, 15(3):445–457, 1997.

[37] R. Talpade and M. Ammar. ”Experience with Architectures for Supporting IP Multicast over ATM”. Technical report, Bellcore, http://www.cc.gatech.edu/computing/Telecomm/people/PhD/taddy/ipmcatmd/a%bs.ps.gz, August 1996.

[38] G. Armitage. *”Support for Multicast over UNI 3.0/3.1 based ATM Networks”*. Internet RFC 2022, November 1996.

[39] Opnet Modeler. http://www.opnet.com/products/modeler/home.html.

[40] D. S. Wong, H. H. Fuentes, and A. H. Chan. ”The Performance Measurement of Cryptographic Primitives on Palm Devices”. *Proceedings of the 17th Annual Computer Security Applications Conference (ACSAC 2001)*, December 2001.

[41] H. Harney and C. Muckenhirn. *”Group Key Management Protocol (GKMP) Architecture”*. Internet RFC 2094, July 1997.

[42] Data Encryption Standard. http://csrc.nist.gov/csrc/fedstandards.html.

[43] M. Abdalla, Y. Shavitt, and A. Wool. ”Key Management For Restricted Multicast Using Broadcast Encryption”. *IEEE/ACM Transactions on Networking*, 8(4):443–454, August 2000.

[44] S. Berkovits. ”How To Broadcast A Secret”. *Advances in Cryptology - EUROCRYPT ’91, Lecture Notes in Computer Science, LNCS*, 547:535–541, 1991.

[45] G. Chiou and W. Chen. ”Secure Broadcasting Using the Secure Lock”. *IEEE Transactions on Software Engineering*, 15(8), August 1989.

[46] K. S. Kim, S. J. Kim, and D. H. Won. ”Conditional Access System Using Smart Card”. *Proc. of JCCI’96, The 6th Joint Conference on Communication and Information*, pages 180–183, 1996.

[47] A. Mani. ”Authenticated Key Agreement in Dynamic Groups”. Master’s thesis, Univeristy of Maryland College Park, 2002.

[48] M. Steiner, G. Tsudik, and M. Waidner. ”Key Agreement in Dynamic Peer Groups”. *IEEE Transactions on Parallel and Distributed Systems*, 11(8):769–780, August 2000.

[49] Y. Kim, A. Perrig, and G. Tsudik. ”Simple and Fault-Tolerant Key Agreement for Dynamic Collaborative Groups”. *Proceedings of the 7th ACM Conference on Computer and Communications Security - CCS ’00*, 2000.

[50] I. Ingemarsson, D. Tang, and C. Wong. ”A Conference Key Distribution System”. *IEEE Transactions on Information Theory*, 28(5):714–720, september 1982.

[51] M. Burmester and Y. Desmedt. ”A Secure and Efficient Conference Key Distribution System”. *Advances in Cryptology - EUROCRYPT ’94, Lecture Notes in Computer Science*, 1994.

[52] D. Steer, L. Strawczynski, W. Diffie, and M. Wiener. ”A Secure Audio Teleconference System”. *Advances in Cryptology - CRYPTO ’88, Lecture Notes in Computer Science, LNCS*, 403:520–528, 1990.

[53] D. Wallner, E. Harder, and R. Agee. *”Key Management for Multicast: Issues and Architectures”*. Internet RFC 2627, June 1999.

[54] O. Rodeh, K. Birman, and D. Dolev. ”Optimized Group Rekey for Group Communication Systems”. *Proceedings of Network and Distributed System Security Symposium (NDSS’00)*, February 2000.

[55] A. Selcuk, C. McCubbin, and D. Sidhu. *”Probabilistic Optimization of LKH-based Multicast Key Distribution Scheme”*. Internet Draft, draft-selcuk-probabilistic-lkh-01.txt.

[56] G. Noubir, F. Zhu, and A. H. Chan. ”Key Management for Simultaneous Join/Leave in Secure Multicast”. *ISIT 2002*, 2002.

[57] D. A. McGrew and A. T. Sherman. ”Key Establishment in Large Dynamic Groups Using One-way Function Trees”. *IEEE Transactions on Software Engineering*, 29(5):444–458, May 2003.

[58] A. Perrig, D. Song, and J.D. Tygar. ”ELK, a New Protocol for Efficient Large-Group Key Distribution”. *Proceedings of IEEE Security and Privacy Symposium S&P2001*, May 2001.

[59] F. Gargione, T. Iida, F. Valdoni, and F. Vatalaro. ”Services, Technologies, and Systems at Ka Band and Beyond - A Survey”. *IEEE Journal on Selected Areas in Communications*, 17(2), February 1999.

[60] S. Mittra. ”Iolus: A Framework for Scalable Secure Multicasting”. *ACM SIGCOMM*, September 1997.

[61] M. Striki and J. S. Baras. ”Key Distribution Protocols for Multicast Group Communication in MANETs”. Technical Report CSHCN TR 2003-10, Center for Satellite and Hybrid Communication Networks, University of Maryland College Park, 2003.

[62] A. Perrig, R. Canetti, D. Song, and J. D. Tygar. ”The TESLA Broadcast Authentication Protocol”. *RSA Cryptobytes*, Summer 2002.

[63] P. Ramachandran. ”Source Authentication for Multicast in Mobile Ad hoc Networks”. Master’s thesis, University of Maryland College Park, 2003.