GROUPS MERGING AND GROUPS DISBANDING IN THE INTERNET

ROBIN
(B. Comp (Hons)., National University of Singapore, 2002)

A THESIS SUBMITTED
FOR THE DEGREE OF MASTER OF SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
NATIONAL UNIVERSITY OF SINGAPORE
2003

Summary

Although IP Multicast has been proposed for more than a decade, its deployment is still limited to network domains under single administrative control. In addition, IP Multicast communication model is largely limited to intra-group communication, where a host is allowed to become member of a group to send or receive from the group. IP Multicast does not provide a service for a whole group to become member of the group to facilitate inter-groups communication.

A novel method is proposed to provide groups merging and groups disbanding to support group-to-group communication in the Internet. The proposed solution is designed to work without modifying IP Multicast routing protocols or changing the existing Internet Group Management Protocol (IGMP). It does not relying on multicast support from routers. Therefore, it can be easily deployed in the Internet.

This solution facilitates more creative development of groupware applications. Some potential applications of the solution include: (1) managing group interactions in a dynamic Computer Supported Cooperative Work (CSCW) environment such as e-learning, (2) dynamic dissemination of notifications in publish/subscribe system, (3) supporting heterogeneous QoS management of multimedia streams encoded with a layered coding scheme.

An architecture to support groups merging and groups disbanding in the Internet called Application Layer Connection Management Framework has been proposed. This architecture consists of two main components: Connection Manager and Connection

Management System. Connection Manager is a distributed component for keeping track of group memberships in its immediate multicast-island, bridging each group in its immediate multicast-island to its respective group in other multicast-islands and managing request to merge/disband a group to/from another group. Connection Management System is formed by interconnected Connection Managers across multiple multicast-islands for exchanging control information and forwarding data.

Protocols for handling join message, join status expiration, bridge message, bridge status expiration, merge message and disband message have been designed. These protocols are explained with the help of flow diagrams.

A prototype has been implemented successfully to demonstrate the feasibility of the proposed solution. The implemented prototype has demonstrated the groups merging and groups disbanding capabilities in the Internet. Performance evaluation results have shown that the implemented prototype manages to perform reasonably well.

Acknowledgement

I would like to express my sincere thanks to my project supervisor, Dr. Pung Hung Keng, for his guidance, valuable discussions and encouragement throughout my research work towards this thesis. His valuable feedback has helped me to improve the thesis in many ways.

I would also like to thank my project team-mates. Special thanks go to Suthon Sae-Wong and Koh Kwang Yong for their willingness to take time to discuss research possibilities with me. I am thankful to Chaiwat Sriyuenyong, An Liming and Chen Xiaodong for their encouragement. I enjoyed all the discussions we had in various topics.

Finally, I would like to thank all the members of Network Systems and Services Laboratory.

Table of Contents

List of Tables

List of Figures

Abbreviation

API	Application Programming Interface
CM	Connection Manager
CMS	Connection Management System
CSCW	Computer Supported Cooperative Work
DPF	Dynamic Protocol Framework
IANA	Internet Assigned Numbers Authority
IEEE	Institute of Electrical and Electronics Engineers
IGMP	Internet Group Management Protocol
ISP	Internet Service Provider
LAN	Local Area Network
QMan	QoS Management Framework
QoS	Quality of Service
RTP	Real-time Transport Protocol
SLM	Service Locating Manager
SM	Stream Manager
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

CHAPTER 1
INTRODUCTION

Although IP Multicast has been proposed for more than a decade, its deployment is still limited to network domains under single administrative control. In addition, IP Multicast communication model is largely limited to intra-group communication, where a host is allowed to become member of a group to send or receive from the group. IP Multicast does not provide a service for a whole group to become member of the group to facilitate inter-groups communication.

A novel method is developed to merge and disband groups in the Internet, which makes inter-groups communication more pervasive. It is designed to work without modifying IP Multicast routing protocols or changing the existing Internet Group Management Protocol (IGMP). It can operate without depending on multicast support from routers. A prototype has been implemented successfully to demonstrate the feasibility of the proposed method. The proposed solution fills the necessary gap in the existing IP Multicast, where the lack of graceful groups merging and groups disbanding in the Internet presents an obstacle for more creative usage of group-to-group applications.

1.1 MOTIVATION

Even though the IP Multicast [1] has been available for more than a decade, it is yet to be widely deployed. IP Multicast requires each host to have access to a native multicast routing service. While intra-domain IP Multicast service (within network domains under single administrative control) is widely available, this is not the case for

inter-domain IP Multicast. Many Internet Service Providers (ISP) are still reluctant to provide multicast routing service [2] because (1) IP Multicast service is hard to maintain and manage; (2) pricing model for multicast service is not clear, should the source or the receivers be charged?

In addition to deployment problem, IP Multicast also suffers from scalability problem as it requires multicast routers to maintain forwarding state for every multicast tree. Forwarding state grows as the number of multicast group increases.

Today's group communication is largely limited to a host sending data to a group or receiving data from a group (intra-group communication). There is no communication service that allows a group sending data to another group or receiving data from another group (inter-group communication). The lack of support for group-to-group communication is largely due to the limitation of current IP Multicast which allows a host to become member of a group, but does not provide a service for a whole group to become members of the group.

Solution to provide groups merging[1] and groups disbanding capabilities has been proposed in previous work [3]. However, the groups merging and groups disbanding capabilities it provides has been limited to within a multicast-island. A multicast-island is defined here as a network of any size that supports IP Multicast. It can be as small as an Ethernet segment, a campus network or a wide area network. The boundary of an island is the furthest extent an IP Multicast packet can travel in the network.

[1] Throughout this thesis, the term "groups merging" is used to refer to the action of a group joining another group as a whole unit and the term "group disbanding" is used to refer to the action of a group leaving another group.

Figure 1-1 illustrates a typical non-multicast-supported network where there are members of group *X* and group *Y*. It is a non-multicast-supported network as the network consists of routers that are incapable of forwarding multicast packets. Group *X* has members x_i residing in subnet 2, while group *Y* has members y_i found in subnet 4. R_{ij} denotes a non-multicast router that bridges connection between subnet *i* and subnet *j*. Each subnet in the network forms a multicast-island.

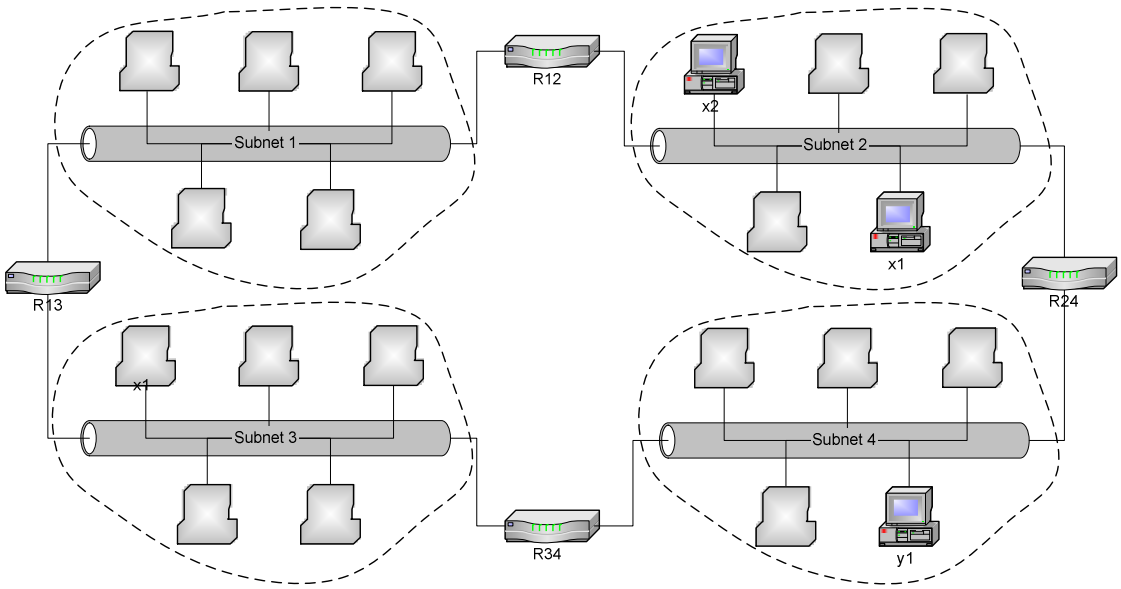

Figure 1-1: A typical non-multicast-supported network with members of group *X* and group *Y*

In this setup, let us suppose group *X* in subnet 2 wishes to receive data from group *Y* in subnet 4 in addition to its own data. There is no way to achieve this in the current IP Multicast as routers connecting subnet 2 and subnet 4 are not capable of forwarding multicast packets. Thus, leveraging current IP Multicast to achieve the effect of groups merging in non-multicast-supported network is practically impossible. Our proposed solution makes groups merging in non-multicast-supported network possible.

The limitations and constraints described above have hold back the development of creative group communication applications. Clearly, there is a need to provide groups

merging and groups disbanding capabilities beyond multicast-island to enable group-to-group communication in the Internet.

1.2 OBJECTIVE

The primary objective of this thesis is to extend groups merging and groups disbanding capabilities beyond multicast-island. We propose a complete solution to provide groups merging and groups disbanding capabilities, both within and beyond a multicast-island.

Our proposed solution provides a way for a group to join to another group as a whole unit and to leave from the group when needed. It is designed to work without modifying the existing IP Multicast routing protocols [4][5][6][7] or changing the existing Internet Group Management Protocol (IGMP) [8][9][10]. It can even operate without depending on multicast support from the router. Thus, it can be deployed easily in the Internet, bypassing the deployment issue of IP Multicast.

Our proposed solution can cooperate with modified router in the previous work or operate on its own to provide groups merging and groups disbanding capabilities within a multicast-island.

Our proposed solution extends groups merging and groups disbanding capabilities beyond multicast-island by leveraging on overlay multicast technology. Overlay multicast, which allows replication and forwarding of data packets to be done in a virtual layer build above the infrastructure network, implements IP Multicast in application layer without support from multicast routers.

Our proposed solution also enhances the reliability of groups merging and groups disbanding. This can be achieved through the usage of reliable protocol such as Transmission Control Protocol (TCP) [11] for overlay multicast.

1.3 GROUPS MERGING AND GROUPS DISBANDING SEMANTIC

Groups merging operation can be explained in terms of subset operation in set theory. Suppose there are two multicast group *A* and *B* with members a_i and b_i respectively. Initially, members of group *A* and group *B* receive only packets from its respective group as shown in Figure 1-2.

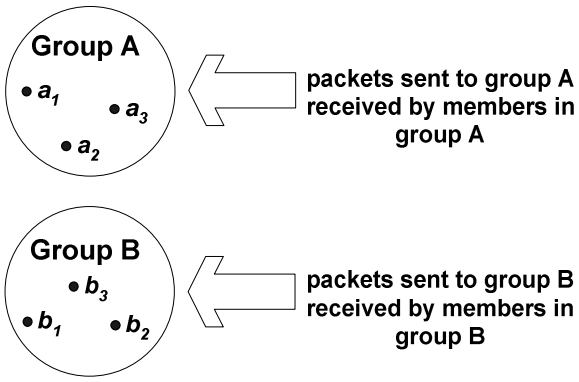

Figure 1-2: Initial state before groups merging operation

When group *A* is merged to group *B*, group *A* becomes a sub group of group *B*. In terms of set theory, this can be viewed as $A \subset B$. Members of group *A* receive multicast packets sent to group *B*. However, members of group *B* do not receive packets sent to group *A*. The effect of group *A* merges to group *B* is illustrated in Figure 1-3.

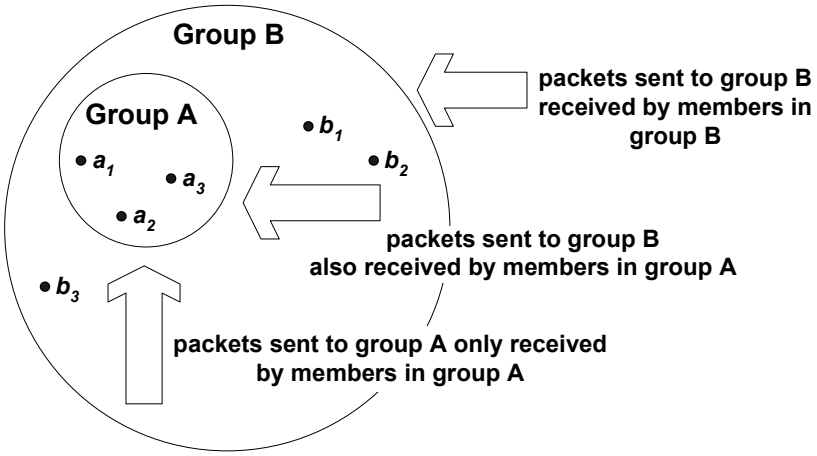

Figure 1-3: Result of group A merges to group B

In Figure 1-3, members of group *A* receive packets sent to group *B* but members of group *B* do not receive packets sent to group *A*. Sometimes, it is desired to have members of group *A* to receive packets sent to group *B* and at the same time members of group *B* can receive packets sent to group *A* as shown in Figure 1-4. This scenario can be achieved by using two groups merging operations: (1) group *A* merges to group *B* and (2) group *B* merges to group *A*.

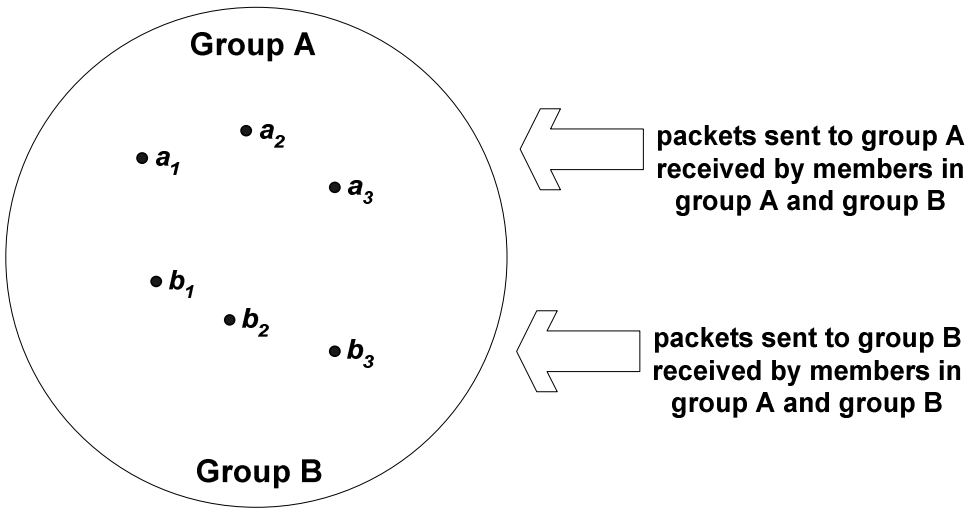

Figure 1-4: Both members of group A and group B can receive packets from each other

A cascading form of groups merging can be accomplished by treating a complex group, a group that contains subgroup, as a single group and let this group merges to another group.

Groups disbanding operation disbands one group from another group i.e. cancels the effect of groups merging operation. Continuing from the point after group *A* merges to group *B*; when group *A* is disbanded from group *B*, group *A* and group *B* return to their initial states i.e. members of group *A* and group *B* only receive packets from their respective group.

By using one message for merging to or disbanding from another group, a graceful form of groups merging to share information and groups disbanding to return to the form of separate groups is achieved.

1.4 POTENTIAL APPLICATION

The provision of groups merging and groups disbanding in the Internet allows developers to create more innovative group communication applications. Potential applications of the solution include: (1) managing group interactions in a dynamic Computer Supported Cooperative Work (CSCW) environment such as e-learning, (2) dynamic dissemination of notifications in publish/subscribe system, (3) supporting heterogeneous QoS management of multimedia streams encoded with a layered coding scheme.

1.4.1 Managing Group Interactions in E-learning Environment

In real-time collaborative e-learning activities such as group discussions, groups merging and groups disbanding can be leveraged to manage interactions between different groups.

Let’s suppose that there is an online discussion session on transport protocol topic. Students are divided into two groups, one group discusses about Transmission Control Protocol (TCP) and another group discusses about User Datagram Protocol (UDP).

After the students have discussed within their own group for some time, the course lecturer wants to allow interaction between groups to compare TCP and UDP. This can be achieved easily by leveraging on groups merging operation.

When the discussion to compare TCP and UDP is done, the lecturer can then disband the merged group into their separated groups.

1.4.2 Dissemination of Notifications in Publish/Subscribe System

Publish/subscribe paradigm is a simple interaction model consisting of information providers that publish events and information consumers who subscribe to events of interest. A publish/subscribe system notifies subscribers as quickly as possible upon the occurrence of relevant events.

In a subject-based publish/subscribe system, each event is classified based on topics. Information providers are required to label each event with a topic and information consumers subscribe to all events within a particular topic.

Since the introduction of IP Multicast, several subject-based publish/subscribe systems that leverage on the IP Multicast to disseminate information have been proposed [12][13]. In these systems, a multicast group is defined for each topic. Events that match a particular topic are multicast to the multicast group assigned for the topic.

Using traditional IP Multicast for disseminating notifications runs into problem in the area of complex subscriptions where one topic may be part of another topic. For

example, suppose there is an online news subscription system which allows subscriber to subscribe to different news topic. The news topic hierarchy is shown in Figure 1-5.

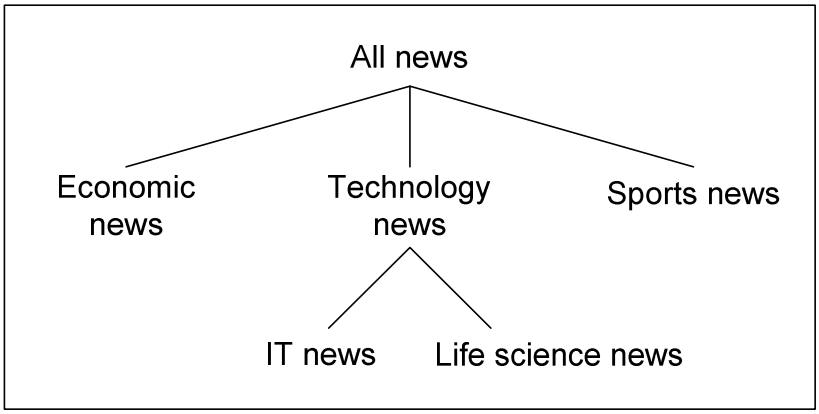

Figure 1-5: News topic hierarchy

When an article on LINUX, which matches topic “All news”, “Technology news” and “IT news”, is published to a publish/subscribe system leveraging on traditional IP Multicast, three copies of the same LINUX article have to be sent, each for multicast group of topic “All news”, “Technology news” and “IT news” as shown in Figure 1-6.

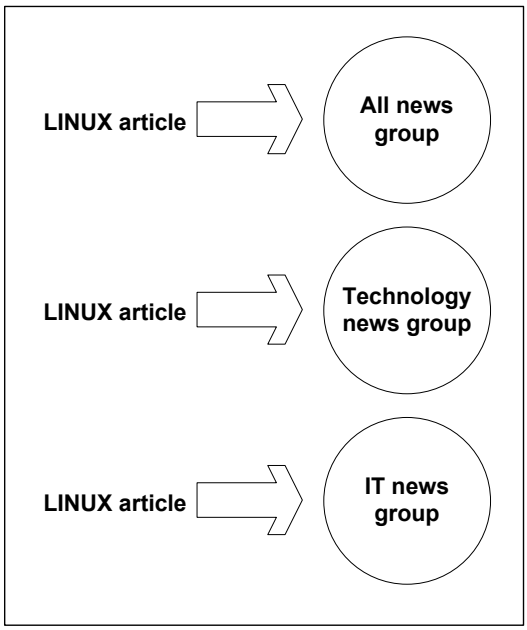

Figure 1-6: Multiple copies of same article are sent when traditional IP Multicast is used

The usage of groups merging can help publish/subscribe systems to avoid sending redundant copies of notifications. Using the groups merging capability, multicast group of topic “IT news” can be merged to multicast group for “Technology news” which in turn merged to multicast group for “All news”. The system only needs to send the LINUX article to the multicast group of “All news”. Because of the groups merging effect, this LINUX article is replicated to “Technology news” and “IT news” multicast groups (as shown in Figure 1-7) only at the edge networks where subscribers reside.

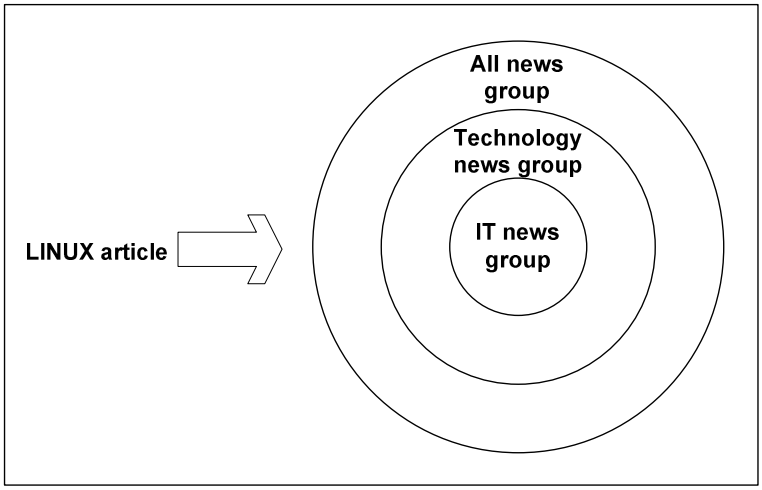

Figure 1-7: Only one copy of the LINUX article is sent when groups merging capability is utilised

1.4.3 Supporting Heterogeneous QoS Management of Multimedia Streams

Groups merging and groups disbanding operations can be leveraged upon to serve heterogeneous QoS based on layered coding scheme [14][15][16]. Layered coding is a signal representation technique, in which the source data is partitioned into base layer and enhancement layers. The base layer contains essential information for reconstruction of the signal by the receiver. The enhancement layers contain information that improves the quality of reception.

Suppose a video stream is encoded with a layered coding scheme into three layers: *base*, *enhancement1* and *enhancement2*. Each of these layers is streamed to a multicast group as illustrated in Figure 1-8.

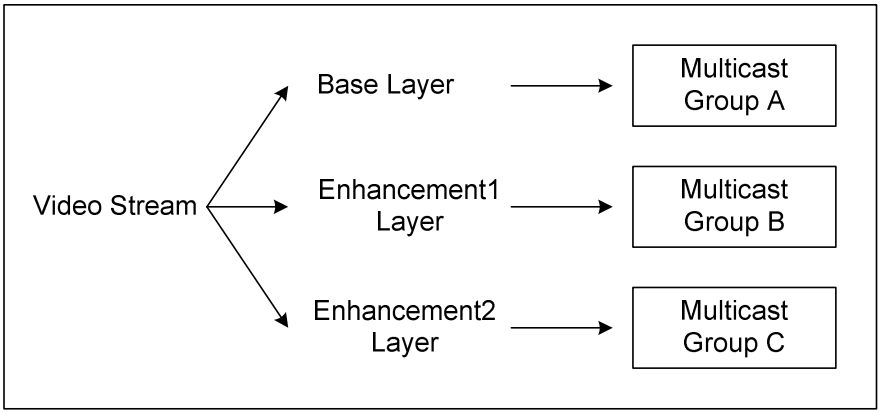

Figure 1-8 Video stream encoded with a layered coding scheme

Further assume that there are three classes of QoS: low, medium and high. Instead of having to join the different multicast groups to receive video coding of the various layers, the hosts can leverage on the merging capability by merging the appropriate multicast groups to the groups they have joined initially. To receive high quality video, hosts subscribe to multicast group F, which is created by merging multicast group A, group B and group C to it, as shown in Figure 1-9.

By leveraging on groups merging and groups disbanding capabilities, dynamic QoS adaptation can be provided. In Figure 1-9, hosts subscribe to multicast group E to receive medium quality video stream can upgrade to high quality simply by merging multicast group E to multicast group C. Similarly, they can downgrade to low quality video stream by disbanding multicast group E from multicast group B.

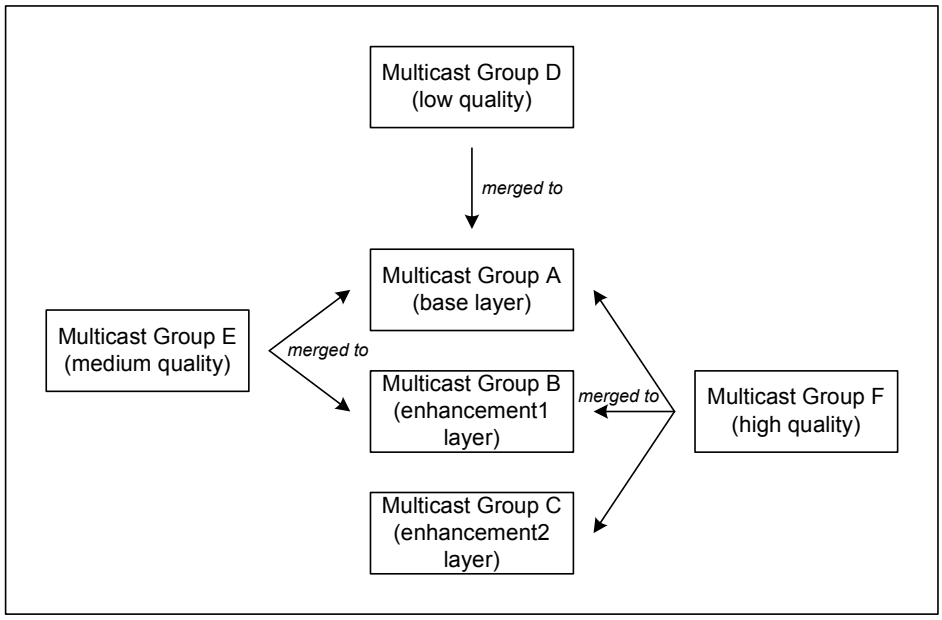

Figure 1-9 Serving heterogeneous QoS by leveraging on groups merging capability

1.5 THESIS OUTLINE

The remaining of the thesis is organised as follows.

Chapter 2 provides background on IP Multicast, Internet Group Management Protocol and OCTOPUS middleware. In addition, related works on groups merging and disbanding are discussed.

Chapter 3 explains the concepts of groups merging and groups disbanding. First, definition of group is given followed by an overview of the solution to provide groups merging and disbanding in various network environments.

Chapter 4 describes the architecture of application level connection management framework. Main components of the architecture together with their functionalities are explained.

Chapter 5 elaborates connection management protocols. Various message types and their usages are described. Detail protocols for various operations are also explained with flowcharts.

Chapter 6 shows the implementation of our prototype. Functional evaluation and performance evaluation for several scenarios are described.

Chapter 7 discusses scalability, reliability, security, placement of Connection Manager and partial groups merging issues and presents solutions to address the issues.

Finally, Chapter 8 presents some concluding remarks and highlights direction of future development.

CHAPTER 2
BACKGROUND AND RELATED WORKS

This chapter gives some background on IP Multicast, Internet Group Management Protocol and OCTOPUS middleware. In addition, related works on groups merging and groups disbanding are discussed.

2.1 BACKGROUND

2.1.1 IP Multicast

Multicast is used to deliver data to multiple receivers from a single source or multiple sources. It provides one-to-many and many-to-many communication; unlike unicast which allows only one-to-one communication. Multicast delivers traffic to multiple receivers efficiently in such a way that packets are not sent several times on a given link.

There are two levels of multicast transmission: local multicast transmission and IP Multicast transmission. Local multicast leverages on multicast capabilities of the physical layer, e.g. Ethernet (IEEE 802.3) to deliver data to multiple receivers on a local area network (LAN). IP Multicast is totally different. It requires the use of multicast routers that are capable of managing multicast delivery tree and responsible for forwarding multicast traffic across LANs.

IP Multicast group is identified by a single multicast IP address. Internet Assigned Numbers Authority (IANA) has assigned Class D IP address space for IP Multicast. IP Multicast group addresses fall in the range of 224.0.0.0 to 239.255.255.255.

IP Multicast group model is an open model. Any host can listen to a multicast group and send to a multicast group without any authorisation. A source can send to a multicast group regardless of whether it is a member of the group or not. Hosts that are interested in receiving a particular multicast group data must join the group to receive data sent to this group.

2.1.2 Internet Group Management Protocol

Internet Group Management Protocol (IGMP) is used by multicast routers to discover the presence of group members on their directly attached LANs.

In IGMP Version 1 [8], there are two types of messages: Membership Report and Membership Query. When a host wants to join a multicast group, it sends out an IGMP Membership Report corresponding to the group that it wishes to receive to its local multicast router. The multicast router periodically sends out an IGMP Membership Query to discover active groups on its directly attached LANs. Hosts respond to IGMP Membership Query by generating IGMP Membership Report for each group that they are interested to join. When IGMP Membership Report is not received for a particular group after three consecutive IGMP Membership Queries, the multicast router times out the group and stops forwarding traffic directed toward that group.

IGMP Version 2 [9] is similar to Version 1. The main difference is the addition of IGMP Membership Leave Report and IGMP Group Specific Membership Query. The hosts can communicate to their local multicast router their intention to leave the group by sending out an IGMP Membership Leave Report. The multicast router then sends out an IGMP Group Specific Membership Query to check whether there are any remaining hosts interested in receiving the traffic from this particular group. If there are no replies, the multicast router times out the group and stops forwarding traffic

directed toward that group. This can greatly reduce the leave latency compared to IGMP Version 1.

IGMP Version 3 [10] adds support for source filtering which allows a host to specify which sender to include or exclude from the list of sources that it is willing to receive multicast traffic from. This enables hosts to specify which source it wants to receive multicast traffic from or to indicate which source it does not want to receive multicast traffic from.

2.1.3 OCTOPUS

OCTOPUS [17][18] is designed to provide middleware level support for the development of multimedia applications and services in the Internet. The main strength of OCTOPUS lies in its ability to segregate control and management of connection, membership, quality of service and streams. It also provides automatic discovery of services in a network and robustness in the provision of services by leveraging on JAVA/JINI [19] technology.

This thesis provides groups merging and groups disbanding capability as Connection Manager within the OCTOPUS architecture (Figure 2-1).

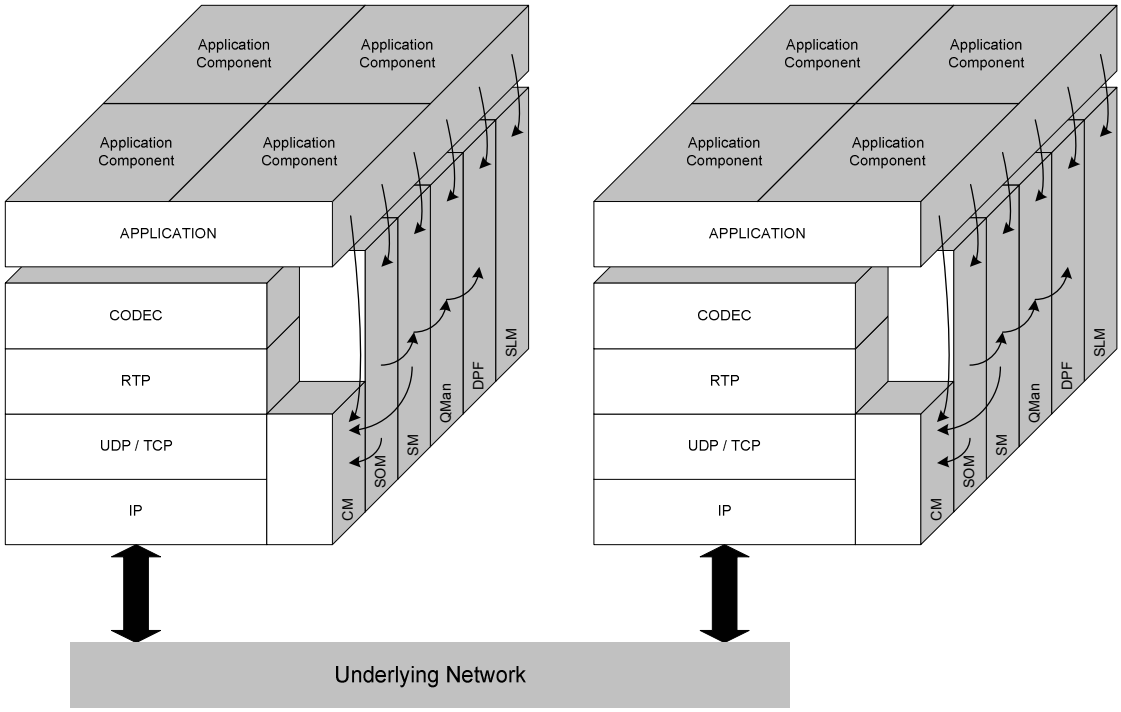

Figure 2-1: OCTOPUS architecture

OCTOPUS hides the complexity of developing multimedia applications from programmers, thus help to shorten the design and implementation cycle of multimedia applications. Key components and their functionality are presented below:

- ***Service Locating Manager (SLM)*** provides service locating and discovery service.

- ***Stream Manager (SM)*** is a light/full profile implementation of audio, video and data streams to offer distributed control and management of streams.

- ***Dynamic Protocol Framework (DPF)*** abstracts the implementation details of dynamic protocol stacks.

- ***QoS Management Framework (QMan)*** abstracts the implementation details necessary for the negotiating, monitoring, adjusting and renegotiating QoS parameters between end hosts.

- ***Connection Manager (CM)*** provides implementation of groups merging and groups disbanding to support complex group interaction.

- ***Session Orchestration Manager (SOM)*** supports a secured dynamic group membership management and access control for intra-session and inter-session applications.

2.2 RELATED WORKS ON GROUPS MERGING AND DISBANDING

A previous work done in a Master thesis [3] proposes an approach to provide groups merge and groups disband capability to the existing IP Multicast at the network level. The term Network Level Connection Manager is used throughout this thesis to refer to the previous work. It extends existing IGMP protocol by proposing new message types for merge and new message types for disband. It requires modification to the edge multicast routers to understand the extended IGMP protocol and to perform packet duplication to achieve group merge effect.

However, adding complexity to the network runs counter to some of the basic design principles of the Internet, in particular the end-to-end argument [20]. According to end-to-end argument, functionality should be pushed to higher layers as possible and routers should be kept as simple as possible. In contrast to Network Level Connection Manager, our propose solution is designed to work at application level without modifying multicast routers or changing the existing IGMP protocol. From now on, the term Application Level Connection Manager is used to refer to our proposed solution.

In the Network Level Connection Manager, every host is assumed to be connected through a multicast-supported network where every router in the network must be IP Multicast enabled. Groups merging and groups disbanding operations only work in the

network-layer multicast supported environment. However, in practise, not all routers are IP Multicast enabled. Our proposed solution is capable of supporting groups merging and groups disbanding operations without depending on multicast support from router.

In today's Internet, only routers under single administrative control are usually IP Multicast enabled. Core routers in the Internet backbone do not natively support IP Multicast. Network Level Connection Manager requires manual configuration of tunnels to connect these IP Multicast enabled networks to form the MBone [21]. It relies upon coordinated manual configuration of multicast routers on both ends of each tunnel, which makes MBone expensive to set up and maintain. Our propose solution is capable to provide groups merging and groups disbanding operations across different domains in the Internet without relying on manual configuration of tunnels.

Moreover, when group A wants to merge to group B, the previous work assumes that group A wishes to merge to all services in group B. Suppose, group A wishes to merge to group B and there are two different applications using group B's multicast address with different port number. $Application_1$ is sending audio while a$pplication_2$ is sending video. Group A wants to receive audio only i.e. merge to a$pplication_1$ in group B. As the Network Level Connection Manager does not allow user to specify group's port number that it intends to merge to, there is no way for user to achieve this. Our propose solution addresses this issue. It allows user to have a finer grain control in group merge operation.

Groups merging and groups disbanding concepts exhibit similarities to aggregated multicast [22][23][24]. In aggregated multicast, multiple multicast sessions are forced to share a single aggregated multicast tree to reduce multicast state at core routers. The

total number of multicast trees maintained by core routers is reduced as core routers only need to maintain state per aggregated tree instead of per group.

Our proposed solution does not focus on how to reduce the forwarding state of the core routers. In contrast, our approach is intended to provide a way for a group to merge to another group as a unit to share information and to disband from another group subsequently.

Another related work is the application level multicast or overlay multicast. In overlay multicast, all the multicast functions including membership management, packet replication and distribution are implemented at application layer. Overlay multicast can be used as a way to bypass IP Multicast deployment and scalability issues.

There are some research works in overlay multicast over the past few years, including ALMI [25], Narada [26][27], HMTP [28], NICE [29] and Hypercast [30].

Because overlay multicast does not require modification to the current Internet infrastructure, it can be deployed easily. However, overlay multicast is not as efficient as IP Multicast. It will incur some delay and bandwidth penalty.

Overlay multicast, which allows replication and forwarding of data packets to be done in a virtual layer build above the infrastructure network, is intended to implement IP Multicast in application layer without support from multicast routers. However, it is not designed to support groups merging and groups disbanding capabilities. In contrast, our solution is a complete solution which provides groups merging and groups disbanding capabilities, both within and beyond a multicast-island. Our solution extends groups merging and groups disbanding capabilities beyond multicast-island by leveraging on overlay multicast technology.

CHAPTER 3

CONCEPTS OF GROUPS MERGING AND GROUPS DISBANDING

This chapter first gives the definition of group. Subsequently, solutions to provide groups merging and groups disbanding in various network environments are explained using examples.

3.1 DEFINITION

3.1.1 Group Definition

A group is associated with a set of zero or more hosts identified by a single destination IP Multicast address and a port number. The destination IP Multicast address and the port number together uniquely identify a group. A group can extend beyond network's boundaries to reach end hosts in other multicast-islands.

For example, group *X* which consists of x_1, x_2 and x_3 are listening to multicast address 224.100.100.100 at port 10000. Another group, group *Y* with members y_1, y_2 and y_3 are sending to multicast address 224.100.100.100 at port 20000. Note that group X and group Y have the same multicast address. However, they are not the same group as they have a different port number.

The following provides an overview of the solution to provide groups merging and disbanding in various network environments.

3.2 GROUPS MERGING IN MULTICAST-SUPPORTED NETWORK

A multicast-supported network is a network which supports IP Multicast natively. All routers in the multicast-supported network are capable of forwarding multicast packets. Figure 3-1 illustrates a typical multicast-supported network where there are members of group *X* and group *Y*, x_i and y_i respectively. Group *X* has members residing in the following subnets: x_1 in subnet 1, x_2 in subnet 2 and x_3 in subnet 3. Group *Y* has members found in the following subnets: y_1 in subnet 3 and y_2 in subnet 4. MR_{ij} denotes a multicast router that bridges connection between subnet i and subnet j. The whole network forms a single multicast-island.

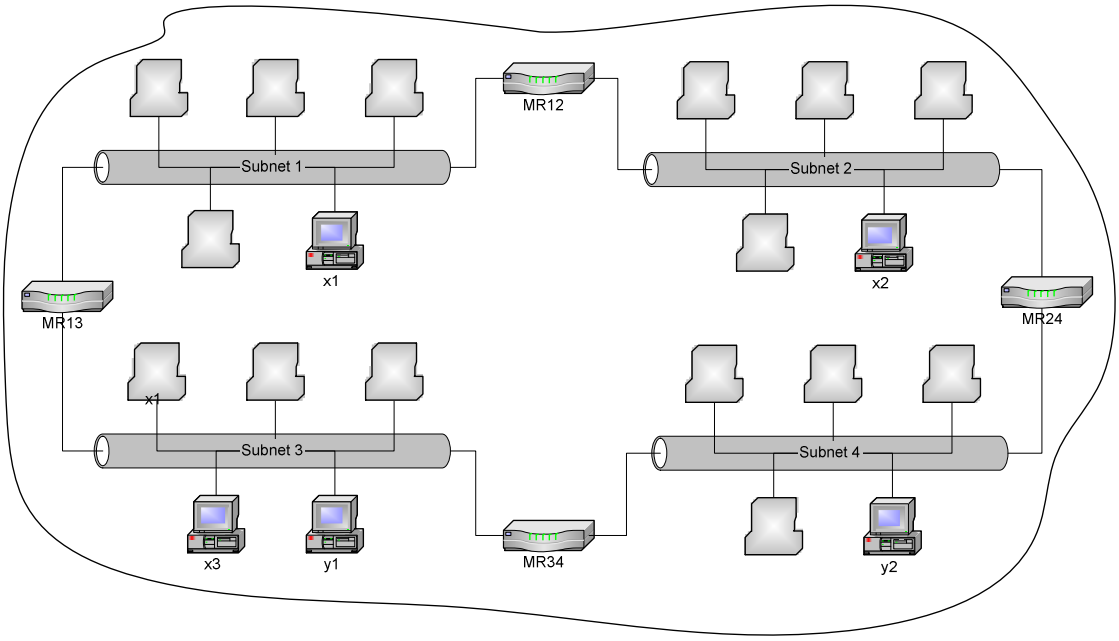

Figure 3-1: A typical multicast-supported network with members of group *X* and group *Y*

In this setup, let us suppose group *X* wishes to receive data from group *Y* in addition to its own data. The only way to achieve this in the current IP Multicast is to have each x_i send individual IGMP Membership Report to join group *Y*. This is illustrated in

Figure 3-2. Besides, each x_i is required to open another multicast socket to receive data from group *Y*.

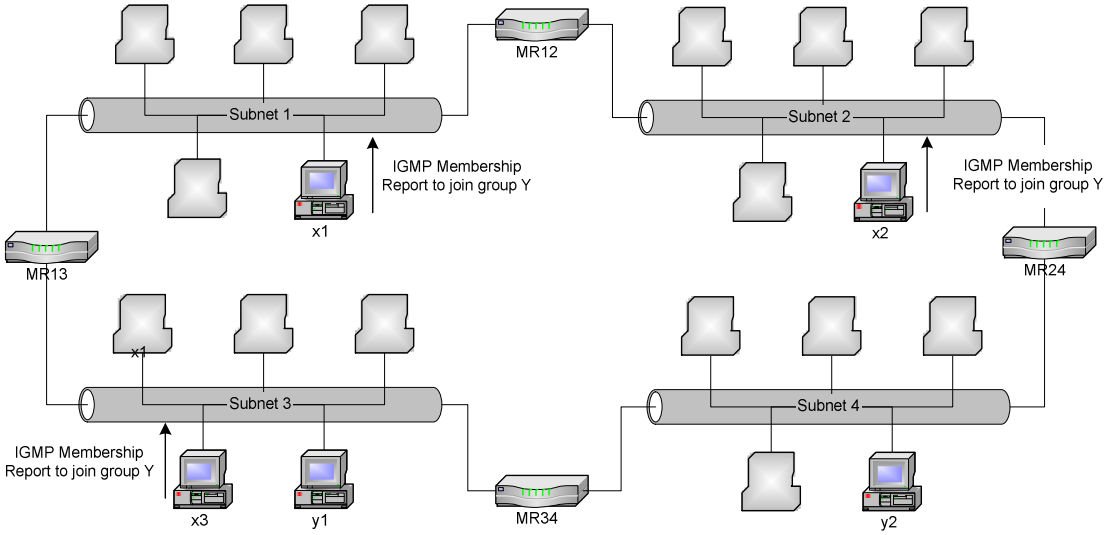

Figure 3-2: The cost of current IP Multicast in sending IGMP Membership Reports by each members of group *X* when joining group *Y*

The number of IGMP Membership Reports sent increases with the number of members. Moreover, each x_i needs more memory to create additional multicast sockets. In addition, there is no coordinated mechanism in current IP Multicast to ensure all members of group *X* has join group *Y*. Thus, leveraging current IP Multicast to achieve the effect of groups merging in multicast-supported network is neither efficient nor well-coordinated.

Three key issues raised in the previous paragraphs:

- Each member of a particular group sends an IGMP Membership Report to join to another group leading to increased network load.

- Each member of a particular group progressively requires more memory to create additional multicast sockets.

- There is no assurance that all members of a particular group has joined to another group.

These problems are addressed in our proposed solution. Figure 3-3 illustrates the proposed solution for groups merging in a multicast-supported network. Suppose that group *X* wants to merge to group *Y*. Only one member of *X* or a third party representative needs to send a MERGE message. This is indicated as step 1 in Figure 3-3. A connection manager (CM) residing in subnet 1 receives this request and sends an IGMP Membership Report packet to notify routers that it is interested in receiving group *Y*'s multicast traffic (step 2). This step is necessary for group *Y* data to be routed to CM which in turn forwards group *Y* data to group *X*. Hence all members of *X* also receive group *Y* packets in addition to those of group *X*.

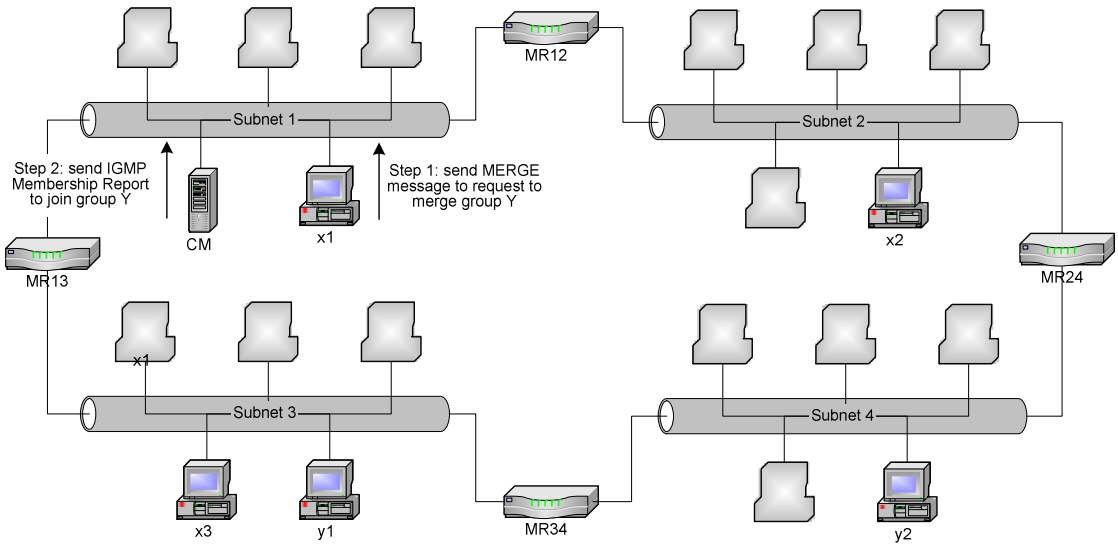

Figure 3-3: Groups merging in a multicast-supported network

In this approach, only one member of *X* or a third party representative needs to send a MERGE message. Furthermore, members of group *X* does not need to create additional multicast socket to receive data for group *Y* as packets for group *Y* are forwarded to group *X* by CM.

This approach does not require any modification to the routers. In addition, existing IGMP protocol used between hosts and multicast routers on a single physical network to establish hosts' membership in particular multicast groups can be used without any amendment. Moreover, this merge operation does not need any changes to the existing multicast routing protocol used for construction and maintenance of the multicast tree.

3.3 GROUPS MERGING IN THE INTERNET

Today's Internet is neither a single multicast-supported network nor a total non-multicast-supported network. The Internet landscape is likely to be composed of multicast-islands with no IP Multicast interoperation across these islands. CMs in various multicast-islands have to be interconnected to form Connection Management System (CMS) which is a virtual layer build above the infrastructure network for exchanging control information and forwarding data. CMS is leveraging on overlay multicast, which implements IP Multicast in application layer without support from multicast router, for communication across multicast-islands.

3.3.1 Individual end host join a group

End hosts across multiple multicast-islands can subscribe to multicast sessions published in any multicast-island. Thus, a multicast session extends group beyond network's boundaries to reach end hosts in other multicast-islands.

For example, end host a_1 provides online music that is multicast via IP Multicast session 224.1.2.3.4:5678. Let's call this IP Multicast session as *A*. End host within the same multicast-island, a_2, can tune into the music simply by subscribing to the IP Multicast group for the concert i.e. group *A*. End host beyond a_1's multicast-island, a_3, have to rely on CM to access the session.

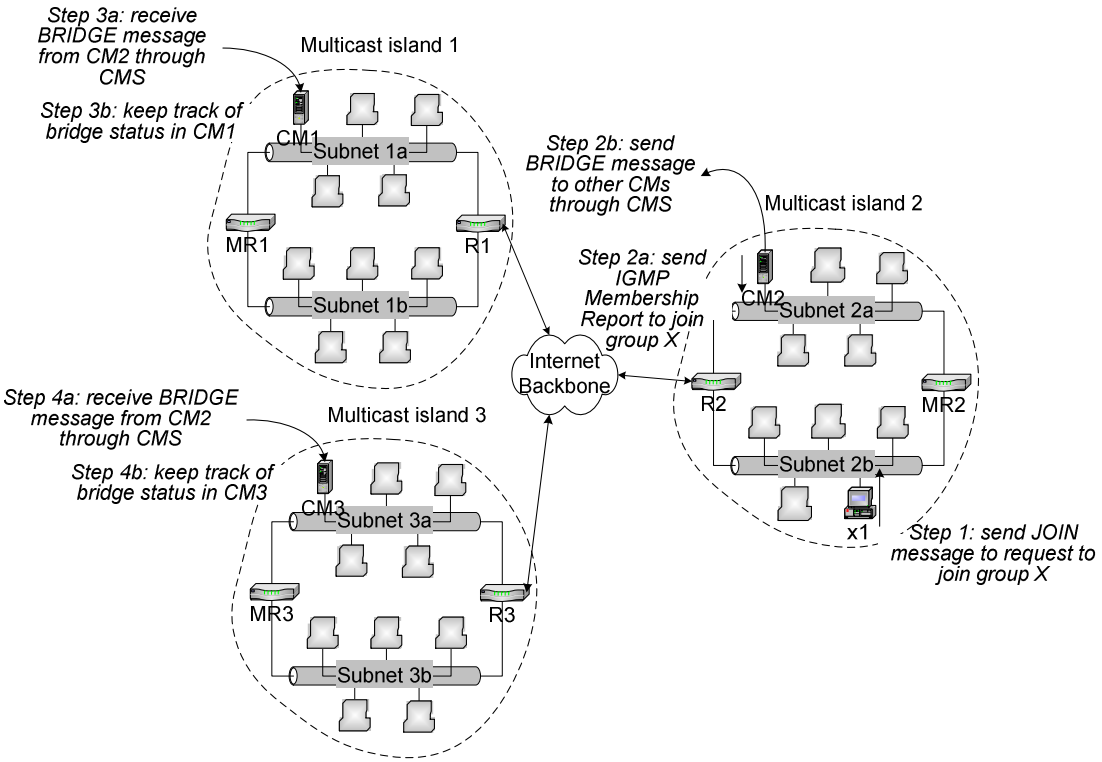

Figure 3-4: The procedure of an end host joining a new group

Figure 3-4 shows the process of an end host joining a new group. x_1 needs to send a JOIN message to a CM for joining group *X* (step 1). A connection manager (CM_2) residing in multicast-island 2 receives this request. It establishes a multicast data channel, X_{om}, and attaches itself to the multicast data channel for the delivery of group *X*'s packets across multicast-islands.

CM_2 sends an IGMP Membership Report packet to indicate its interest in receiving group *X*'s multicast traffic (step 2a). This step is necessary for group *X* data to be routed to CM_2 which in turn forwards group *X* data to other CMs that have member of group *X* in their immediate multicast-island through CMS. At this point of time in this example, no CM other than CM_2 has member of group *X* in its immediate multicast-island, therefore CM_2 does not forward group *X* data to other CMs.

CM_2 also sends a BRIDGE message to other CMs through CMS (step 2b). The BRIDGE message includes the IP Multicast group identifier specified in the JOIN message, X, and the multicast data channel identifier used to transfer IP Multicast packets across multicast-islands, X_{om}. When CM_1 and CM_3 receive this BRIDGE message through the CMS they maintain this bridge status.

Subsequently, when an end host in multicast-island 3, x_2, joins group X (step 1 in Figure 3-5), CM_3 joins to multicast data channel X_{om} to receive group X's packets from other multicast-islands. When CM_3 receives group X's packets from other multicast island through CMS using multicast data channel X_{om}, it multicasts them to group X. It also forwards group X's packets in multicast-island 3 to other multicast-island through CMS using multicast data channel X_{om}.

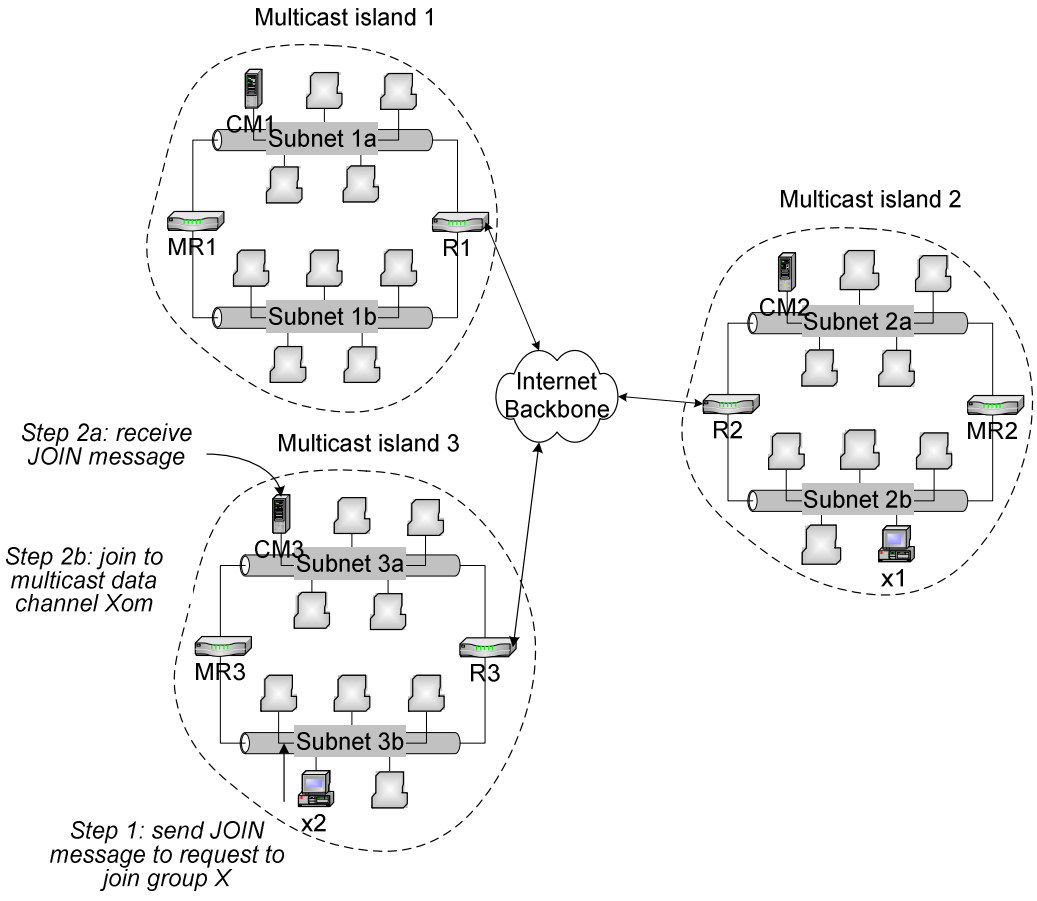

Figure 3-5: The procedure of an end host joining to an existing group

As a result of the individual JOIN operations, members of *X* can receive group *X* packets from other multicast-islands in addition to those from their immediate multicast-island.

3.3.2 Group merge to another group

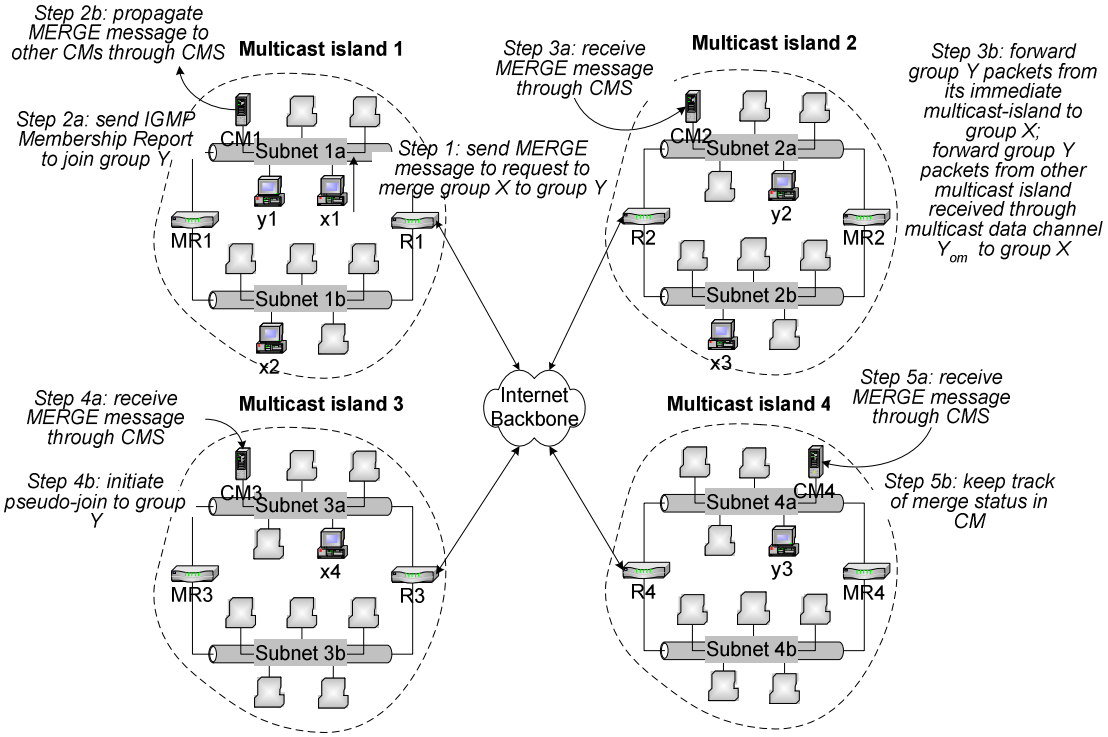

Figure 3-6: The procedure for groups merging in the Internet

Suppose that group *X* wants to merge to group *Y*. X_{om} and Y_{om} are multicast data channels used to deliver group *X*'s packets and group *Y*'s packets respectively across multicast-islands.

Only one member of *X* (x_1 in this case) needs to send a MERGE message as in step 1 in Figure 3-6. A connection manager (CM_1) residing in multicast-island 1 receives this request and sends an IGMP Membership Report packet to indicate its interest in receiving group *Y*'s multicast traffic. This step is necessary for group *Y* data to be routed to CM_1 which in turn forwards group *Y* data to members of group *X* in this

multicast-island only. At this point of time, groups merge effect has been achieved in CM_1's multicast-island. In order to achieve the group merge effect in other multicast-islands, CM_1 sends a MERGE message to other CMs through CMS. When another CM receives this MERGE message through the CMS, it acts in one of the four ways:

a. ***Both member of group X and member of group Y exist in the corresponding multicast-island (as in multicast-island 2 in Figure 3-6):*** When CM_2 receives group *Y* packets from other multicast-islands through CMS using multicast data channel Y_{om}, it forwards them to group *X* by translating group *Y* address to group *X* address. Subsequent packets for group *Y* in its immediate multicast-island are also forwarded by CM_2 to group *X*.

b. ***Only member of group X exists in the corresponding multicast-island (as in multicast-island 3 in Figure 3-6):*** CM_3 initiates a pseudo-join to group *Y* as if an end host in the multicast-island joining group *Y*. As a result, group *Y* packets from other multicast-islands can be received by CM_3 through CMS using multicast data channel Y_{om}, and are subsequently forwarded to group *X*.

c. ***Only member of group Y exists in the corresponding multicast-island (as in multicast-island 4 in Figure 3-6):*** CM_4 maintains this merge status. Subsequently, when an end host in this multicast-island joins group *X*, the CM forwards group *Y* packets from other multicast-islands and group *Y* packets from its immediate multicast-island to group *X*.

d. ***No member of group X and group Y is found in the corresponding multicast-island:*** CM maintains this merge status. Subsequently, when both members of group *X* and group *Y*, only member of group *X*, or only member of group *Y* appears

in its immediate multicast-island, CM acts as in alternative a, b or c respectively as mentioned above.

As a result of the group *X* MERGE group *Y* operation, all members of *X* also receive group *Y* packets in addition to those of group *X*.

Supporting groups merging in non-multicast-supported network is similar to supporting groups merging in the Internet. The only difference is that the end hosts do not need to send an explicit JOIN message to join a group. Instead, CMs listen for IGMP Membership Reports to discover group's members on their attached local networks.

Our proposed solution provides a way to achieve merge effect without relying on multicast router. Hence, it can operate even in a network with no multicast router.

CHAPTER 4

ARCHITECTURE OF APPLICATION LEVEL CONNECTION MANAGEMENT FRAMEWORK

This chapter provides an overview design of application level connection management framework architecture. Architecture components together with their functionalities are described.

4.1 ARCHITECTURE COMPONENTS

Application Level Connection Management Framework is composed of a collection of Connection Managers (CMs) that are interconnected via an overlay network called Connection Management System (CMS) which is used to exchange control information and deliver data.

Figure 4-1 illustrates the framework's architecture. Clouds *A, B, C, D* and *E* denote multicast-islands. These multicast-islands are connected to the Internet backbone. While IP Multicast is supported within multicast-islands, this is often not the case in the Internet backbone. Today's Internet backbone only supports unicast traffics. In every multicast-island there must be at least one CM. CMs in every multicast-islands are interconnected to an overlay multicast network named CMS, where each edge in the CMS corresponds to a unicast path between two CMs in the underlying network.

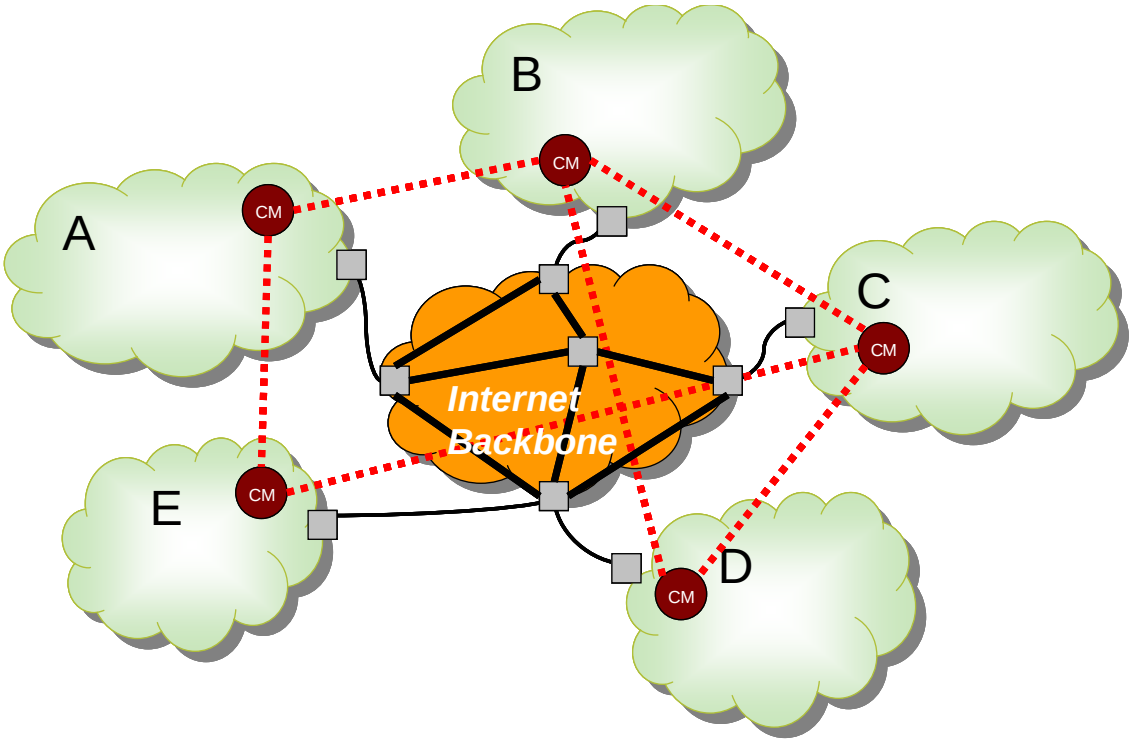

Figure 4-1: Architecture of Application Level Connection Manager Framework

4.2 CONNECTION MANAGER

Connection Manager (CM) is a distributed component for keeping track of group memberships in its immediate multicast-island, bridging each group in its immediate multicast-island to its respective group in other multicast-islands and managing requests to merge/disband a group to/from another group.

CM keeps track of group memberships in its immediate multicast-island by monitoring requests (described in section 5.1.1) to join group. In addition, CM forwards data from groups in its immediate multicast-island to their respective groups in other multicast-islands through the CMS. Moreover, when CM receives a request (described in section 5.1.3) to merge group *X* to group *Y*, it is responsible for acquiring data from group *Y* and forwarding the data to group *X* to achieve the merge effect.

CM maintains a data structure called *join_list* (Figure 4-2) to keep track of active groups in its immediate multicast-island. In addition, it stores the mapping between IP Multicast group and multicast data channel used to deliver packets across multicast-islands for that particular IP Multicast group in a data structure called *bridge_table* (Figure 4-3). Moreover, it maintains a data structure called *merge_table* (Figure 4-4) to keep track of the list of successful merge requests from CMS and from end hosts.

4.2.1 Modules of Connection Manager

Connection Manager consists of three subsystems: control module, data forwarding module and manager module.

Control module is responsible for setting up control channels, handling requests for joining group and groups merging/disbanding from end hosts and exchanging its knowledge of bridge status and merge status with other CMs through CMS.

Data forwarding module is responsible for setting up data communication channels and forwarding packets via those channels.

Manager module is responsible for coordinating interaction between modules and maintaining internal data structures: *join_list*, *bridge_table* and *merge_table*.

The structure of *join_list* is illustrated in Figure 4-2. Each node contains group information which is composed of IP Multicast address and port number.

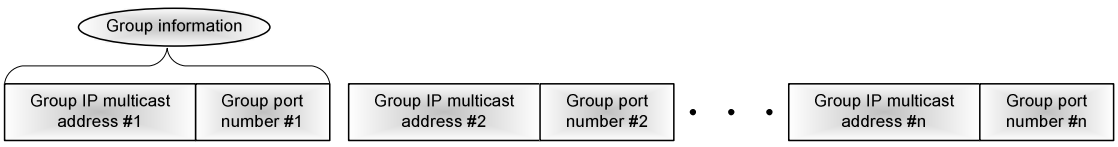

Figure 4-2: Structure of *join_list*

The structure of an entry in *bridge_table* is illustrated in Figure 4-3. Each entry contains IP Multicast group information and its corresponding multicast data channel used to deliver packets across multicast-islands for that particular IP Multicast group. Group information is composed of IP Multicast address and port number.

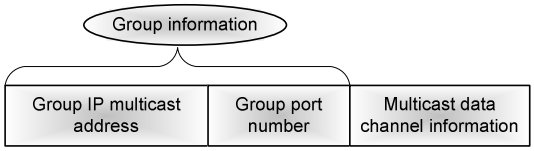

Figure 4-3: Structure of an entry in *bridge_table*

The structure of *merge_table* is illustrated in Figure 4-4. Each entry contains group information and its corresponding subgroups information. Group information and subgroup information are composed of IP Multicast address and port number.

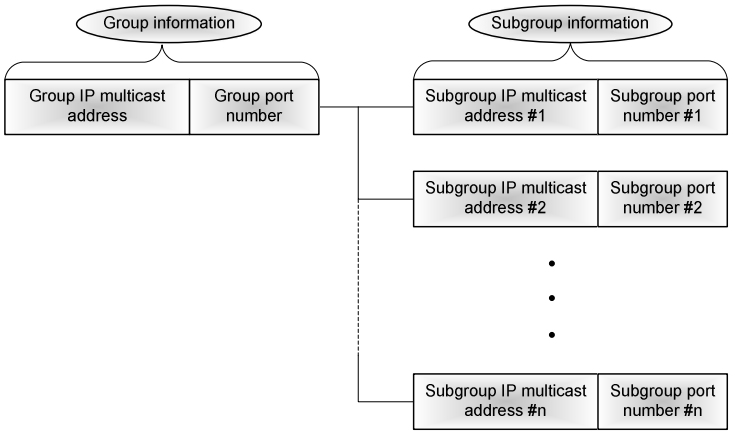

Figure 4-4: Structure of an entry in *merge_table*

4.2.2 Connection Manager Operations

The followings are operations provided by Connection Manager:

- **Join operation:** Any valid end host can initiate join operation by sending a join message (described in section 5.1.1) to Connection Manager to indicate its interest

in receiving data for a multicast group from other multicast-islands. It is different from IGMP Membership Report sent to end host's local multicast router to indicate end host's interest in receiving data for a multicast group from local multicast-island.

- **Bridge operation:** Connection Manager initiates a bridge operation (bridge message is described in section 5.1.2) to send/receive a particular multicast group data to/from other multicast-islands when there exists at least a member of the group in its immediate multicast-island.

- **Groups merging operation:** Any authorised end host or third party can initiate groups merging operation by sending merge message (described in section 5.1.3) to Connection Manager.

- **Groups disbanding operation:** Any authorised end host or third party can initiate groups disbanding operation by sending disband message (described in section 5.1.4) to Connection Manager.

4.3 CONNECTION MANAGEMENT SYSTEM

Connection Management System (CMS) is an overlay multicast network formed by interconnected CMs across multiple multicast-islands for exchanging control information and forwarding data.

CMs use overlay multicast on CMS to relay information across multicast-island. CMS leverages on overlay multicast technology for exchange of control information and forward of data packets to other multicast-islands.

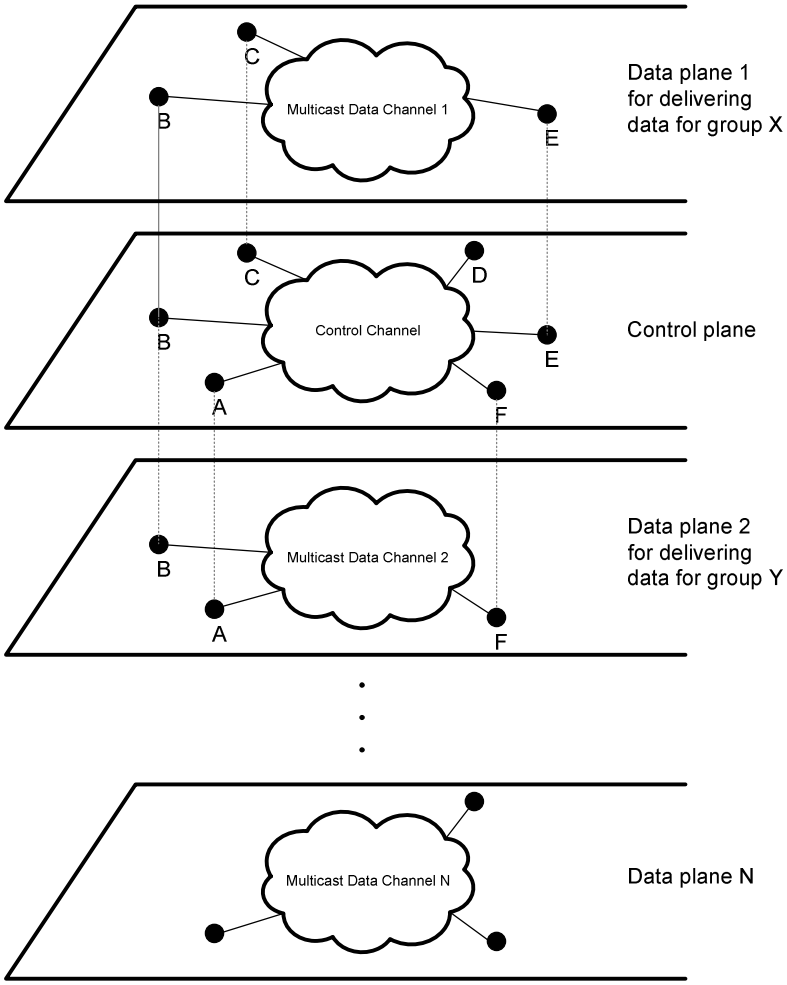

Figure 4-5: Multiple multicast channels on Connection Management System

There is one multicast control channel and multiple multicast data channels in CMS as illustrated in Figure 4-5. Node *A, B, C, D, E* and *F* denotes CMs. *B, C* and *E* have members of group *X* in their immediate multicast-island. *A, B* and *F* have members of group *Y* in their immediate multicast-island. Clouds denote CMS control channel and data channels.

All CMs connects to the control channel, forming a control plane, to exchange control information. Each data channel is dedicated to deliver data for a specific group. In Figure 4-5, *multicast data channel 1* is used to deliver data for group *X* and *multicast data channel 2* is used to deliver data for group *Y*.

A particular CM does not connect to all data channels. It only connects to a group's data channel if there exists at least a member of the group in its immediate multicast-island. In Figure 4-5, only *B, C* and *E* are connecting to *multicast data channel 1* for sending and receiving group *X* data. *A, B* and *F* connect to *multicast data channel 2* for delivering group *Y* data. *D* connects neither to *multicast data channel 1* nor *multicast data channel 2*, as there is no member of group *X* or member of group *Y* in its immediate multicast-island. It should also be noted that a CM may connects to multiple multicast data channels, such as *B*.

There are a number of overlay multicast protocols proposed in the literature [25][26][27][28][29][30], but no single protocol has been deployed globally. This is due to a variety of reasons including technical shortcomings in the protocols and their implementations.

To operate across a range of overlay multicast protocols, CMS has a customisable module called *OverlaySocket* (described in section 6.1.2) which isolates the implementation of an end point communication for a certain overlay multicast protocol.

CHAPTER 5
CONNECTION MANAGEMENT PROTOCOLS

This chapter elaborates various connection management protocol messages. Protocols for handling these messages are also explained in details using flowcharts.

5.1 CONNECTION MANAGEMENT PROTOCOL MESSAGES

Four types of connection management protocol messages are defined: *join message, bridge message, merge message* and *disband message*. The purpose for each of the message is elaborated in the following paragraphs.

5.1.1 Join Message

The format of join message is shown in Figure 5-1. The message type of 1 indicates this is a join message. Join message contains information of the group which end host wants to join to.

Figure 5-1: Join message format

Join messages is used by Connection Manager (CM) to keep track of group memberships in its immediate multicast-island. End host *x* sends a join message to CM periodically (every *j* seconds) for joining group *X* (section 5.2.1 describes logic for handling join message by the CM in details). This allows late-joining CM to build up its knowledge of group memberships present on its immediate multicast-island. In

addition, sending join message periodically covers the possibility of the initial join message being lost or damaged.

If no join message for group *X* is received by CM after *2j* seconds, CM assumed that there is no member of group *X* on its multicast-island (section 5.2.2 describes logic for handling expiration of join status by the CM in details). By using soft-state group membership, CM refreshes its knowledge of group memberships present on its immediate multicast-island periodically.

In order to avoid an implosion of concurrent join messages, when it is time to send join messages, rather than sending them immediately, the end host starts a delay timer for each group. Each timer is set to a different, randomly-chosen value between zero and *D* seconds. When a timer expires, a join message is generated for the corresponding group. Thus, join messages are spread out over a *D* seconds interval instead of all occurring at once.

To reduce the total number of join messages transmitted, when an end host hears a join message for a group, the end host stops its own timer for that group and does not generate a join message for that group. Thus, in the normal case, only one join message will be generated for each group, by the end host whose delay timer expires first.

5.1.2 Bridge Message

The format of bridge message is shown in Figure 5-2. The message type of 2 indicates this is a bridge message. Bridge message contains a list of IP Multicast group and multicast data channel pairs.

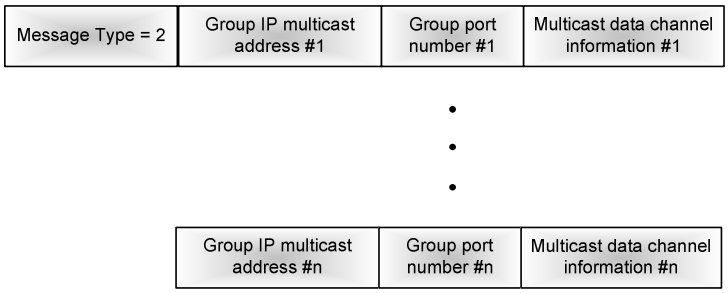

Figure 5-2: Bridge message format

Bridge message is sent by a CM to other CMs through CMS for bridging a group in its immediate multicast-island to other multicast-islands. Upon receiving join message from end host, CM immediately sends bridge message to other CMs through CMS to update other CMs' knowledge of bridge status. In addition, CM sends bridge message to other CMs periodically (every *b* seconds), reporting multicast data channel information for each group in its immediate multicast-island reflected in the *join_list* (section 5.2.3 describes logic for handling bridge message by the CM in details). Sending bridge message periodically allows late-joining CM to build up its knowledge of bridge status and covers the possibility of the initial bridge message being lost or damaged.

If no bridge message for a particular group is received by CM after *2b* seconds, CM assumed that it no longer needs to bridge the group to other multicast-islands (section 5.2.4 describes logic for handling expiration of bridge status by the CM in details). By using soft-state bridge status, CMs refreshes their knowledge of bridge status periodically.

When the CM hears a bridge message, it uses the information contains in the bridge message to validate its own *bridge_table*. Those IP Multicast group and multicast data channel pairs in the bridge message that are contained in its *bridge_table* are marked as *refreshed*. When the next time it sends bridge message, those entries in *bridge_table*

that are marked as *refreshed* are not sent. The *refreshed* status is reset after the bridge message is sent. Those IP Multicast group and multicast data channel pairs in the bridge message that are not contained in its *bridge_table* are added to its *bridge_table.*

5.1.3 Merge Message

The format of merge message is shown in Figure 5-3. The message type of 3 indicates this is a merge message.

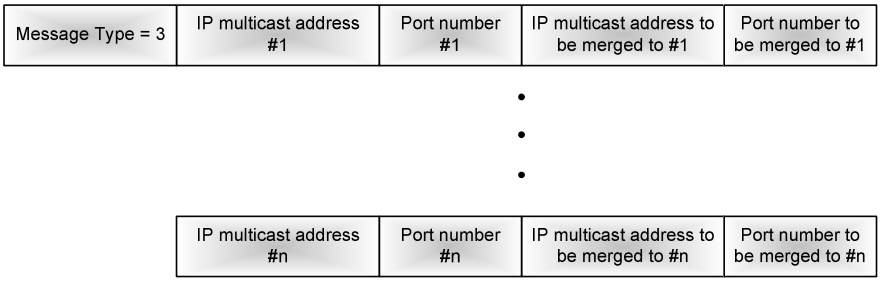

Figure 5-3: Merge message format

Merge message is sent by an end host or a third party control to CM for merging a group to another group (section 5.2.5 describes logic for handling merge message by the CM in details). Upon receiving merge message from the end host, CM immediately propagates merge message to other CMs through CMS to update other CMs' knowledge of merge status. In addition, CMs sends merge message to other CMs periodically (every *m* seconds) to refresh their knowledge of merge status. Sending merge message periodically allows late-joining CM to build up its knowledge of merge status and covers the possibility of the initial merge message being lost or damaged.

In order to reduce the total number of merge message transmitted, when a CM hears a merge message, the CM resets its own timer and does not generate a merge message for that cycle.

5.1.4 Disband Message

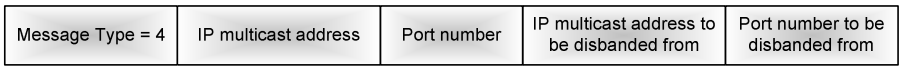

Figure 5-4: Disband message format

Disband message is sent by an end host or a third party control to CM for disbanding a group from another group (section 5.2.6 describes logic for handling disband message by the CM in details). Upon receiving disband message from end host, CM immediately propagates disband message to other CMs through CMS to update other CMs' knowledge of merge status.

To cover the possibility of the initial disband message being lost or damaged, it is repeated after short delays.

5.2 CONNECTION MANAGEMENT PROTOCOL

The following sections describe in details how a CM behaves when it receives a join message, bridge message, merge message and merge message. The way CM handles expiration of join status and bridge status are also explained in details.

5.2.1 Handling of Join Message

End host sends join request to join to a specific group through a multicast channel. CM listens to the multicast channel and upon receiving join request; it processes the request as shown in the flowchart in Figure 5-5.

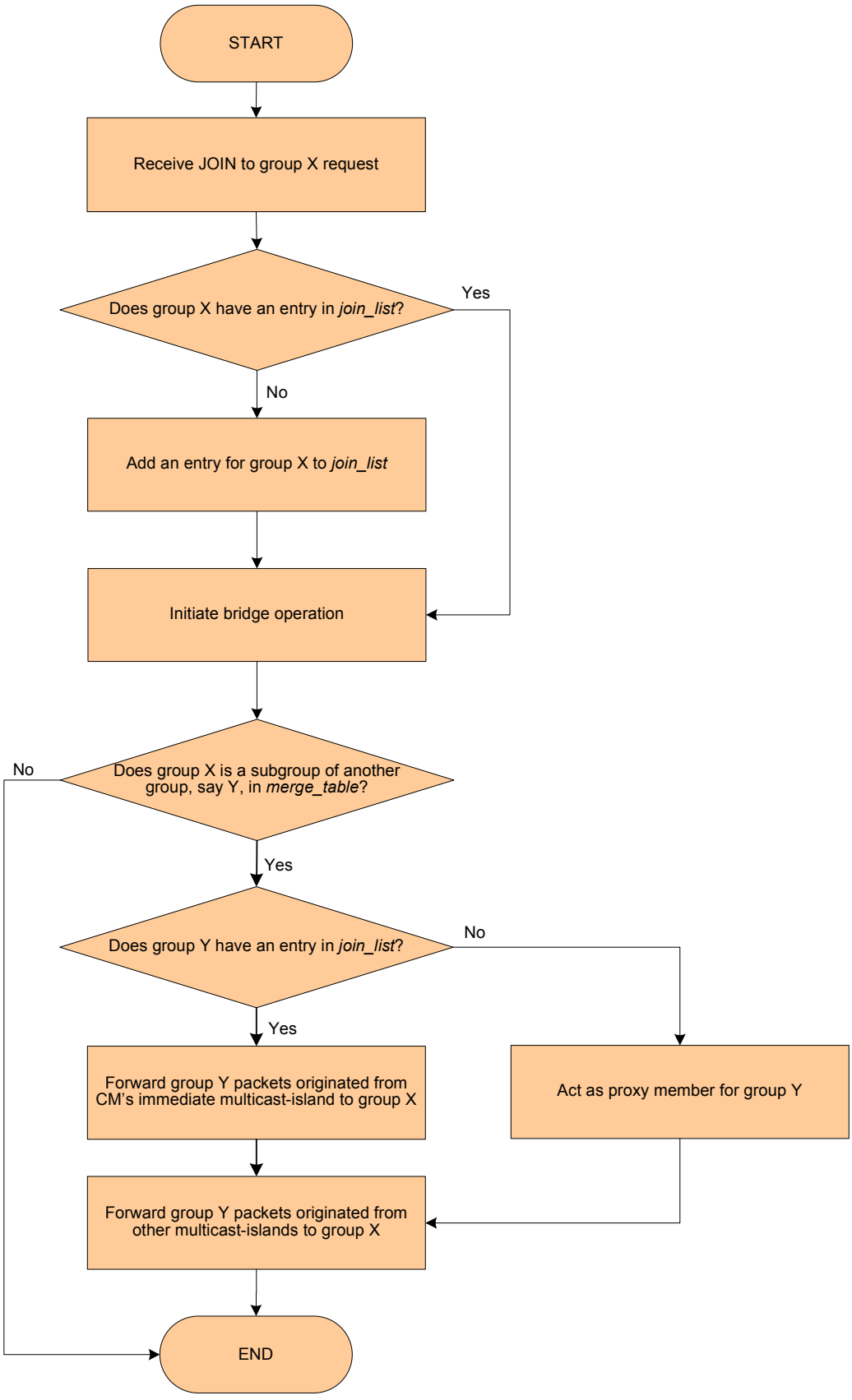

Figure 5-5: Logic for handling join request by CM

Figure 5-5 illustrates the logic of CM handling an end host request to join group *X*. CM first checks if group *X* already has an entry in *join_list*. CM adds an entry for group *X* to *join_list* if group *X* does not have an entry in *join_list.* It then proceeds to bridge group *X* in its immediate multicast-island to group *X* in other multicast-island through the CMS. The bridge operation logic flow is explained in detail in section 5.2.3. Up to this step, members of group *X* will be able to receive group *X* data from their immediate multicast-island as well as those from other multicast-islands.

If group *X* has previously merged to another group, say group *Y*, then by joining group *X*, members of group *X* should be able to receive data from group *Y* too. To cater for this late-joining scenario, CM needs to verify if group *X* has previously merged to group *Y* by checking the *merge_table*. If group *X* has never merged to other groups, nothing else needs to be done. Otherwise, CM has to perform additional actions to ensure that group *X* members in its immediate multicast-island receive group *Y* data both from its immediate multicast-island and other multicast-islands.

First, to ensure that group *X* members in CM's immediate multicast-island receive group *Y* data from its immediate multicast-island, it forwards group *Y* packets originated from its immediate multicast-island to group *X* if there exists group *Y* member in its immediate multicast-island. Otherwise, it acts as proxy member for group *Y*. This is to ensure that subsequent packets for group *Y* will be received by the CM which in turn forwarded to group *X*.

Second, to ensure that group *X* members in CM's immediate multicast-island receive group *Y* data from other multicast-islands, subsequent group *Y* packets originated from other multicast-islands received through CMS are forwarded by CM to group *X*.

5.2.2 Handling of Join Status Expiration

If no join message for a specific group is received by CM after *2j* seconds, CM assumed that there is no member of this group on its immediate multicast-island. This is why an end host need to send a join request periodically (every *j* seconds) to refresh the join status. When a join status for a specific group expires, CM handles it as shown in the flowchart in Figure 5-6.

Figure 5-6 illustrates the logic of CM handling an expired join status for group *X*. CM removes an entry for group *X* from *join_list.* It then stops receiving group *X* data originated from other multicast-islands. Up to this step, group *X* data originated from other multicast-islands are no longer delivered to this CM as there is no member of group *X* in its immediate multicast-island.

If group *X* has previously merged to another group, say group *Y*, then by leaving group *X*, CM no longer needs to forward data from group *Y* to group *X*. To cater for this scenario, CM needs to verify if group *X* has previously merged to group *Y* by checking the *merge_table*. If group *X* has never merged to other groups, nothing else needs to be done. Otherwise, if there exists member of group *Y* in its immediate multicast-island, CM stops forwarding packets originated from group *Y* in its immediate multicast-island to group *X*. In addition, CM stops forwarding packets originated from group *Y* in other multicast-islands to group *X*.

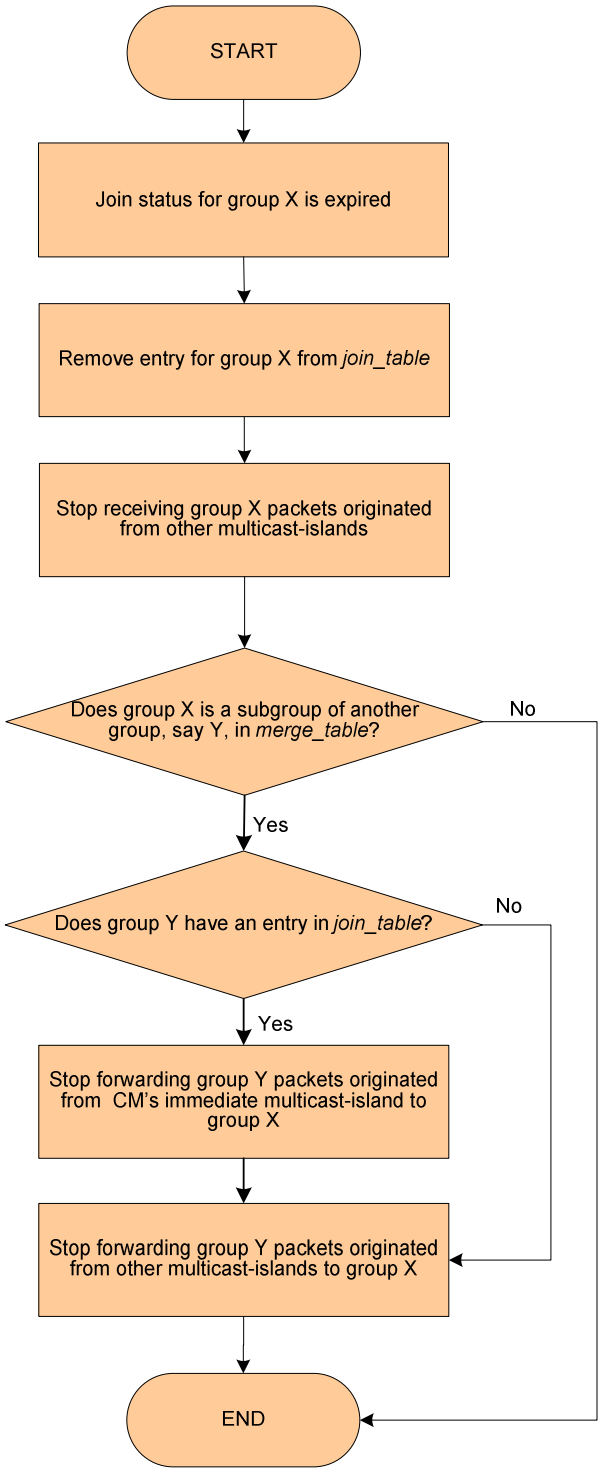

Figure 5-6: Logic for handling expired join status by CM

5.2.3 Handling of Bridge Message

CM sends bridge requests to other CMs through CMS. When another CM receives a bridge request; it processes the request as shown in the flowchart in Figure 5-7.

Figure 5-7 illustrates the logic of CM handling a request to bridge group *X*. A bridge message may contain a list of IP Multicast group and multicast data channel pairs. For a pair of IP Multicast group and multicast data channel, CM first checks if multicast data channel for delivering group *X* data across multicast-islands has already been established by checking *bridge_table*. If group *X* does not have an entry in *bridge_table* and the bridge request is not coming from another CM through CMS, CM establishes an multicast data channel for delivering group *X* data (in this case X_{om}) and propagates bridge request to other CMs through CMS. An entry for group *X* and multicast data channel X_{om} is then added to *bridge_table*.

CM proceeds to check if there is any member of group *X* in its immediate multicast-island by checking *join_list*. If group *X* does not have an entry in *join_list*, nothing else need to be done. Otherwise, CM forwards packets from group *X* in its immediate multicast-island to group *X* members in other multicast-islands through multicast data channel X_{om}. Subsequent packets from group *X* in other multicast-islands received through multicast data channel X_{om} is also forwarded by CM to group *X* in its immediate multicast-island.

This process is repeated until all pairs of IP Multicast group and multicast data channel in the bridge message have been processed.

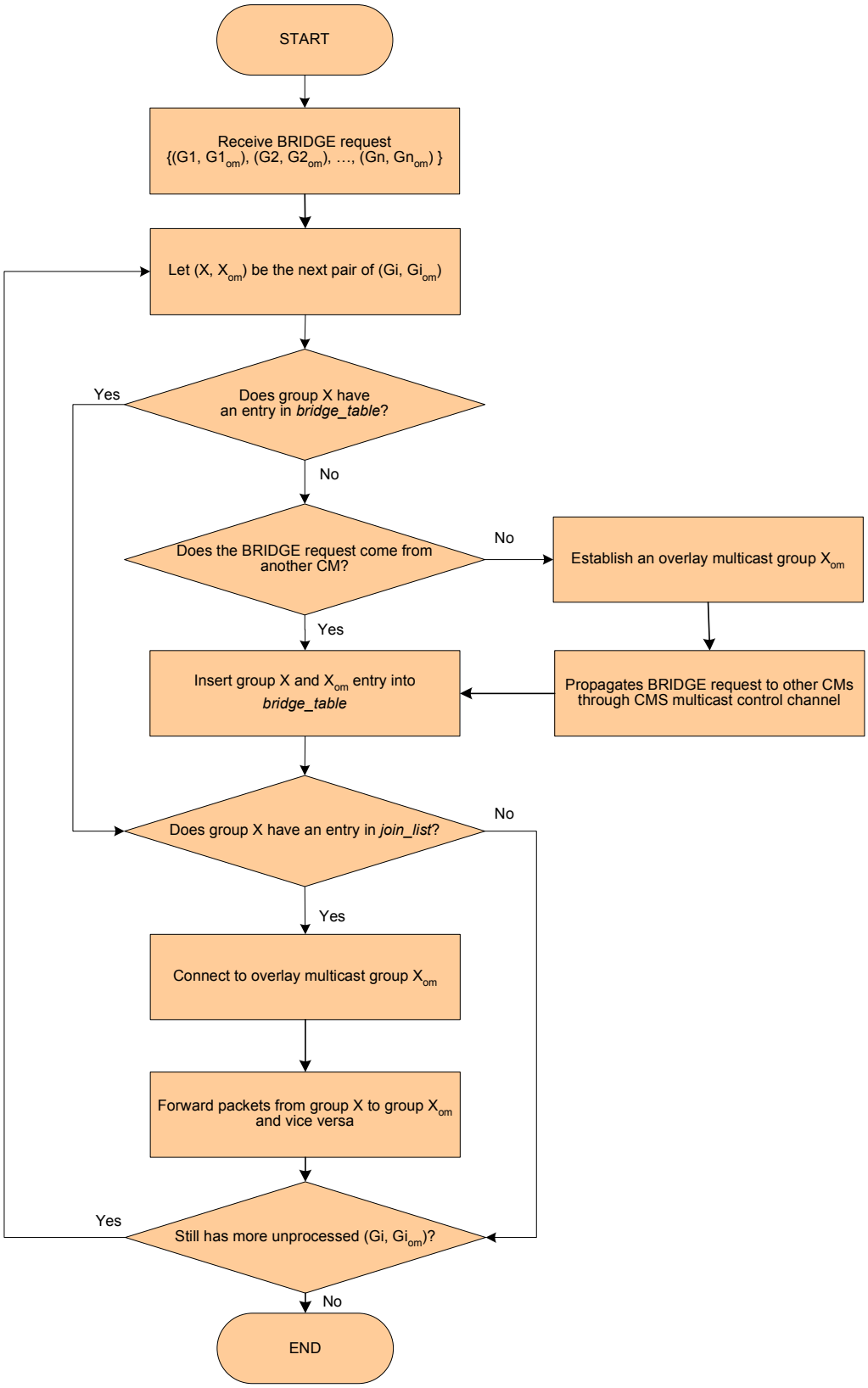

Figure 5-7: Logic for handling bridge request by CM

5.2.4 Handling of Bridge Status Expiration

If no bridge message for a specific group is received by CM after *2b* seconds, CM assumed that it no longer needs to bridge the group to other multicast-islands. This is why a CM needs to send a bridge group request periodically (every *b* seconds) to refresh the bridge status. When a bridge status for a specific group expires, CM handles it as shown in the flowchart in Figure 5-8.

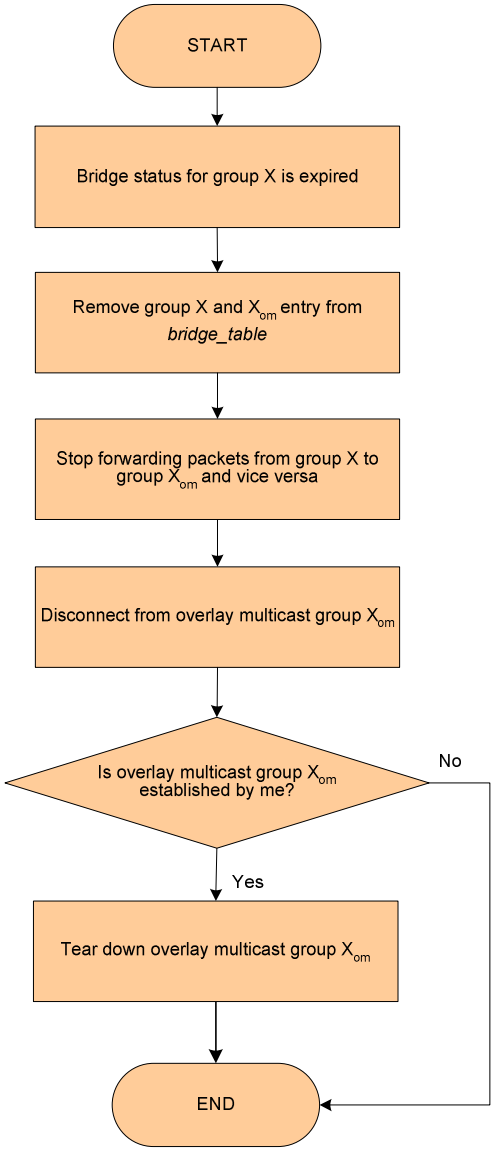

Figure 5-8: Logic for handling expired bridge status by CM

Figure 5-8 illustrates the logic of CM handling an expired bridge status for group *X*. CM removes an entry for group *X* from *bridge_table*. It stops forwarding packets from group *X* in its immediate multicast-island to group *X* members in other multicast-islands.

It then proceeds to disconnect itself from multicast data channel X_{om}, as it no longer needs to send or receive data to or from the multicast data channel. Also, it checks if the multicast data channel X_{om} was established by itself. If it was, the multicast data channel X_{om} is tore down. Otherwise, nothing else needs to be done.

5.2.5 Handling of Merge Message

Authorised host sends group merge request to merge group *X* to group *Y* to CM through a multicast channel. CM listens to the multicast channel and upon receiving group merge request, it processes the request by checking its internal data structure to decide on appropriate action to achieve the group merge effect.

Figure 5-9 shows the logic for handling group merge request by CM. A merge message may contain a list of groups and subgroups. For a pair of group and subgroup, CM checks whether the request comes from another CM when it receives a group merge request. If the request does not come from another CM, it propagates the merge request to other CMs through CMS. CM proceeds to check if (group *Y*, subgroup *X*) already has an entry in *merge_table*. CM adds an entry for (group *Y*, subgroup *X*) in *merge_table* if (group *Y*, subgroup *X*) does not already have an entry in *merge_table.* If group *X* has subgroups merged to it previously, CM initiates a merge request to merge group *Y* to all subgroup that merged to group *X* (except those which subgroup entry is group *Y*).

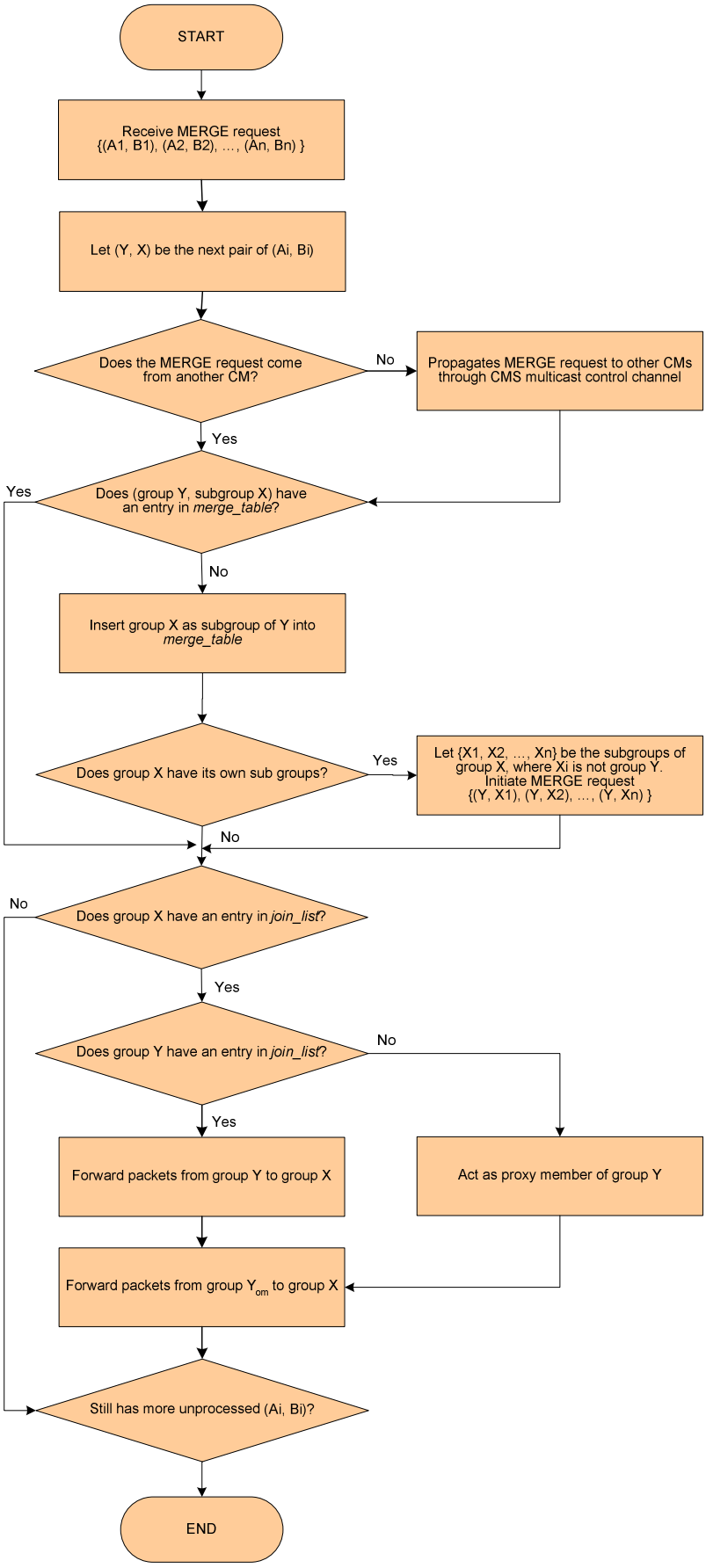

Figure 5-9: Logic for handling group merge request by CM

CM proceeds to check if there is any member of group *X* in its immediate multicast-island. If there is no member of group *X* in its immediate multicast-island, nothing else needs to be done. Otherwise, CM has to perform additional actions to ensure that the members of group *X* in its immediate multicast-island receive group *Y* data both from its immediate multicast-island and other multicast-islands.

First, to ensure that group *X* members in CM's immediate multicast-island receive group *Y* data from its immediate multicast-island, it forwards packets from group *Y* to group *X* if there exists group *Y* member in its immediate multicast-island. Otherwise, it acts as proxy member for group *Y*. This is to ensure that subsequent packets for group *Y* will be received by the CM which in turn forwarded to group *X*.

Second, to ensure that group *X* members in CM's immediate multicast-island receive group *Y* data from other multicast-islands, subsequent group *Y* packets from other multicast-islands received through CMS using multicast data channel Y_{om} are forwarded by CM to group *X*.

This process is repeated until all pairs of groups and subgroups in the merge message have been processed.

5.2.6 Handling of Disband Message

Authorised host sends a group disband request to disband group X from group Y to CM through a multicast channel. CM listens to the multicast channel and upon receiving a group disband request, it processes the request by checking its internal data structure to decide on appropriate action to cancel the group merge effect.

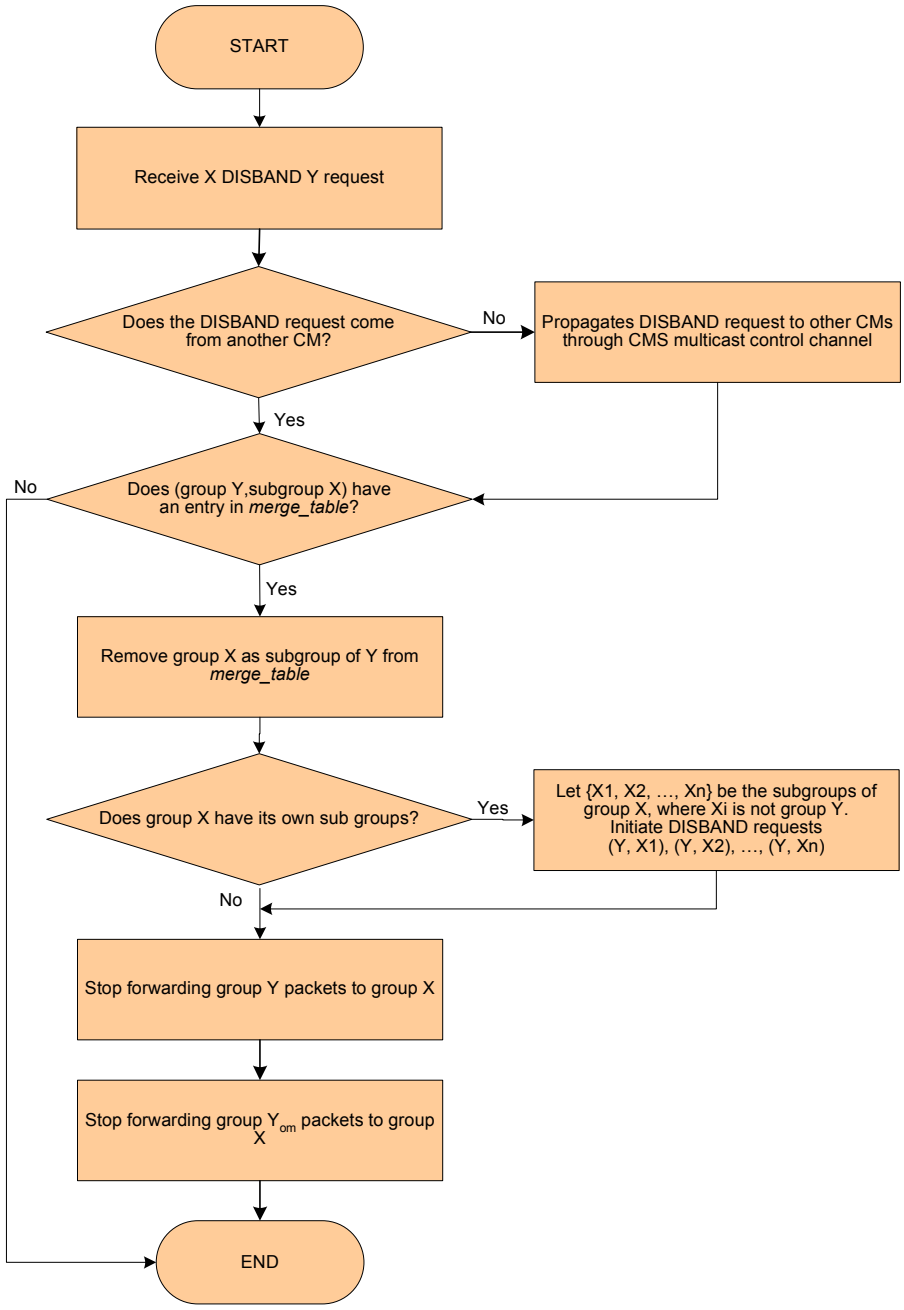

Figure 5-10: Logic for handling group disband request by CM

Figure 5-10 shows the logic for handling group disband request by CM. CM checks whether the request comes from another CM through CMS when it receives a group disband request. If the request does not come from another CM, it propagates the disband request to other CMs through CMS. CM proceeds to check if (group *Y*, subgroup *X*) already has an entry in *merge_table*. If (group *Y*, subgroup *X*) does not have an entry in *merge_table*, nothing else needs to be done. Otherwise, CM has to

remove (group *Y*, subgroup *X*) from *merge_table*. If group *X* has subgroups merged to it previously, CM initiates disband requests to disband group *Y* from all subgroup that merged to group *X*.

CM performs additional actions to ensure that the members of group *X* in its immediate multicast-island no longer receive group *Y* data both from its immediate multicast-island and other multicast-islands.

First, to ensure that group *X* members in CM's immediate multicast-island no longer receive group *Y* data from the its immediate multicast-island, it stops forwarding group *Y* packets originated from its immediate multicast-island packets to group *X*.

Second, to ensure that group *X* members in CM's immediate multicast-island no longer receive group *Y* data from other multicast-islands, it stops forwarding group *Y* packets originated from other multicast-islands to group *X*.

CHAPTER 6
IMPLEMENTATION AND EVALUATION

This chapter describes the implementation of our prototype. Functional evaluation and performance evaluation for several scenarios are also explained.

6.1 IMPLEMENTATION

Java programming language [31] was chosen to implement the prototype mainly because of its platform independent. This is important to provide groups merging and groups disbanding capabilities to users running on different platforms.

In the following sub-sections, implementation of Connection Manager and Connection Management System are discussed.

6.1.1 Implementation of Connection Manager

6.1.1.1 Control module

A socket based multi-threaded concurrent server was created using Java Network Application Programming Interface (API). The concurrent server is capable of handling several clients at the same time. End hosts accessible operations are encapsulated within a set of simple and explicit API shown below.

These API are described briefly as follows:

- ***join:*** invoked by end host to send join request to Connection Manager.
- ***merge:*** invoked by end host to send groups merge request to Connection Manager.

- ***disband:*** invoked by end host to send groups disband request to Connection Manager.

```
                                int port)

public static void merge (      String mcastAddr,
                                int port,
                                String mergeToMcastAddr,
                                int mergeToPort)


                                int port,
                                String disbandFromMcastAddr,
                                int disbandFromPort)
```

Figure 6-1: Connection Manager API accessible to end host

Control module is also responsible to exchange bridge status and merge status periodically with other Connection Managers through Connection Management System.

6.1.1.2 Data forwarding module

Main responsibilities of this module are establishing data communication channels and forwarding packets via those established communication channels.

The main component of data forwarding module is *ForwardingEngine*, which is responsible for forwarding data from one group to another group. The group can be an IP Multicast group within multicast-island or an overlay multicast group (CMS' multicast data channel) delivering data between multicast-islands.

To minimise drop packets, a buffer is used to store received packets before they are forwarded to another destination.

6.1.1.3 Manager module

Manager module acts as the central manager of Connection Manager. It manages interaction between various modules of Connection Manager. In addition, it maintains internal data structures: *join_list*, *bridge_table* and *merge_table*.

6.1.2 Implementation of Connection Management System

Connection Management System has a module called *OverlaySocket* which provides a programming interface for communications using overlay multicast protocol. There may exists more than one implementation of the *OverlaySocket* interface, each for a particular overlay multicast protocol. The *OverlaySocket* interface declares several methods expected to be implemented by a class implementing *OverlaySocket*.

```
public interface OverlaySocket {

     public void init();
     /** Performs initialization.

     public void createGroup(String overlayID);
     /** Creates an overlay multicast group

     public void joinGroup(String overlayID);
     /** Joins an overlay multicast group

     public void leaveGroup(String overlayID);
     /** Leaves an overlay multicast group

     public void send(DatagramPacket dp);
     /** Sends a datagram packet to the overlay socket



     public void send(Message msg);
     /** Sends a message to the overlay socket. The message can be
     a join message, bridge message, merge message or disband
     message

     public DatagramPacket receive() throws IOException;
     /** Receives a datagram packet from the overlay socket

     public void close();
     /** Close this overlay socket

}
```

Figure 6-2: *OverlaySocket* interface

In the prototype implementation, *OverlaySocket* implementation based on Hypercast overlay multicast protocol has been completed.

6.2 EVALUATIONS

6.2.1 Functionality Test

JMStudio [32], a Java application that uses the Java Media Framework API [33] to play, capture, transcode and write media data, is used to demonstrate the functionality of the implemented prototype. JMStudio uses RTP [34] to transmit audio and video data.

The setup for testing the functionality of the implemented prototype is shown in Figure 6-3.

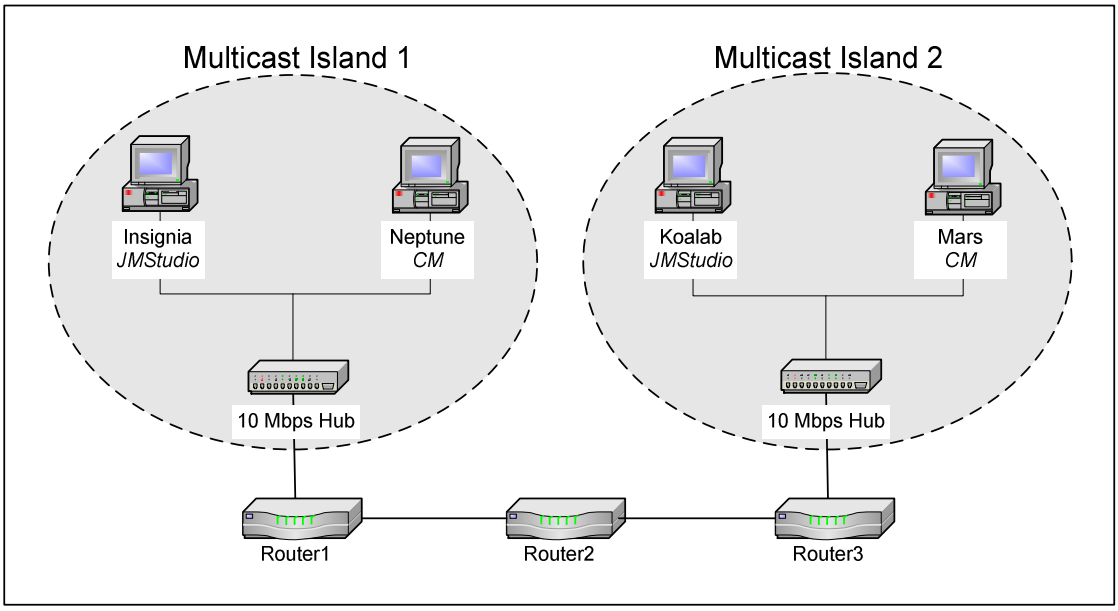

Figure 6-3: Setup for functionality testing

Insignia and *Neptune* are located at *multicast island 1*. *Insignia* runs the JMStudio program and *Neptune* is setup as Connection Manager. In *multicast island 2*, *Koalab* runs the JMStudio program and *Mars* is setup as Connection Manager. *Router1*, *Router2* and *Router3* perform unicast routing only. IP Multicast routing is not enabled in Router1, *Router2* and Router3. Multicast *island 1* and *multicast island 2* are connected to each other by *Router1*, *Router2* and *Router3*.

By using the JMStudio program, the following scenarios are tested:

- Join operation
- Groups merging operation
- Groups disbanding operation

6.2.1.1 Test on Join operation

A JMStudio session is started at *Insignia* which is located at *multicast island 1* to transmit video to group *X* (224.100.100.100 port 10000). Another JMStudio session is started at *Koalab* which is located at *multicast island 2* to receive video transmission on group *X*. If no join request is sent by *Koalab*, it is unable to receive the video transmission from Insignia. This is because *Insignia* and *Koalab* are located in different multicast islands. By invoking a join request to join group *X*, *Koalab* is able to receive the video transmission sent by *Insignia.*

6.2.1.2 Test on Groups merging Operation

A JMStudio session is started at *Insignia* which is located at *multicast island 1* to transmit video to group *X* (224.100.100.100 port 10000). Another JMStudio session is started at *Koalab* which is located at *multicast island 2* to receive video transmission on group *Y* (224.200.200.200 port 20000). If no groups merge request is sent, *Koalab* is unable to receive the video transmission from Insignia. By invoking a groups merging operation to merge group *Y* to group *X*, *Koalab* is able to receive the video transmission sent by *Insignia.*

6.2.1.3 Test on Groups disbanding operation

Continuing from test on groups merging operation, after a groups merging operation is invoked to merge group *Y* to group *X*, *Koalab* is able to receive the video transmission sent by *Insignia.* Now, *Koalab* sent a groups disband request to disband group *Y* from group *X*. After the groups disbanding operation is invoked, *Koalab* stops receiving group *X* packets originated from *Insignia*.

6.2.2 Performance Test

6.2.2.1 Merge Latency and Disband Latency

Merge latency is defined as how much time it takes for a merge request to take effect. It is measured by taking the time difference between the time when a merge request is invoked ($time_{\ merge\ request\ invoked}$) and the time when first merge packet is received by receiver ($time_{\ first\ merge\ packet\ arrived}$).

$$Merge\ Latency = time_{\ first\ merge\ packet\ arrived} - time_{\ merge\ request\ invoked}$$

Disband latency is defined as how much time it takes for a disband request to take effect. It is measured by taking the time difference between the time when a disband request is invoked ($time_{\ disband\ request\ invoked}$) and the time when last merge packet is received by receiver ($time_{\ last\ merge\ packet\ arrived}$).

$$Disband\ Latency = time_{\ last\ merge\ packet\ arrived} - time_{\ disband\ request\ invoked}$$

The setup used in measuring the *Merge Latency* and *Disband Latency* of Application Level Connection Manager is shown in Figure 6-4.

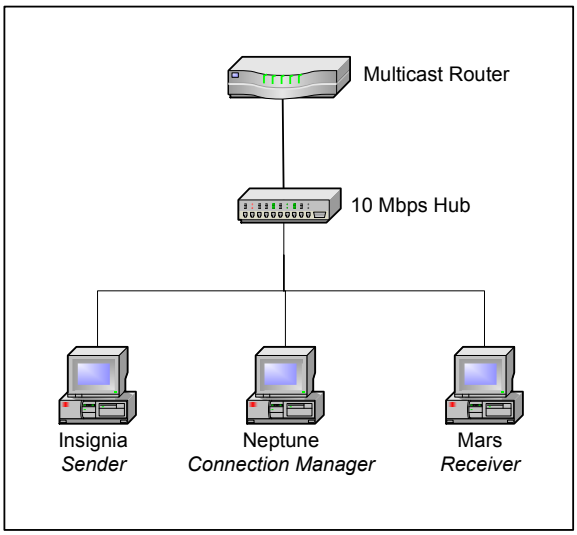

Figure 6-4: Setup for testing on merge latency and disband latency of Application Level Connection Manager

Three machines, *Insignia, Neptune* and *Mars*, are connected to a 10 Mbps hub which connects to a multicast router. *Insignia* sends multicast packets to group *X* in intervals of 60 milliseconds. *Neptune* is setup as Connection Manager to handle merge and disband requests. *Mars* listens to group *Y* and send a merge request to merge groups *Y* to group *X*. Subsequently *Mars* sends a disband request to disband group *Y* from group *X*.

The time when *Mars* sends a merge request is recorded as $time_{merge\ request\ invoked}$ and the time when *Mars* sends a disband request is recorded as $time_{disband\ request\ invoked}$. The time when *Mars* first receives a merge packet while listening on group *Y* is recorded as $time_{first\ merge\ packet\ arrived}$. The time when *Mars* last receives a merge packet while listening on group *Y* is recorded as $time_{last\ merge\ packet\ arrived}$.

We also measure *Merge Latency* and *Disband Latency* of previous work done in a Master thesis (Network Level Connection Manager). In Network Level Connection Manager, groups merging and groups disbanding capabilities are implemented at network layer. Two new IGMP messages are introduced: *IGMP_Merge_Request* and

IGMP_Disband_Request. Multicast router is modified to understand these new messages. In addition, a 0-CMM is needed to receive merge and disband request from end host in the form of UDP packets. 0-CMM then sends *IGMP_Merge_Request* and *IGMP_Disband_Request* to the modified multicast router for merge request and disband request respectively. The setup used in measuring the *Merge Latency* and *Disband Latency* of previous implementation is shown in Figure 6-5.

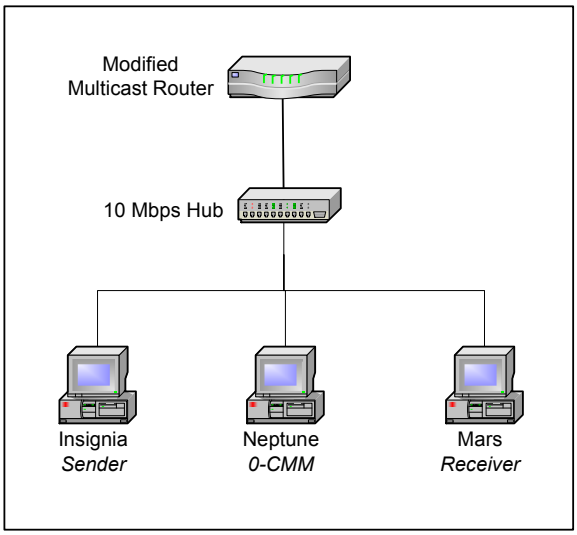

Figure 6-5: Setup for testing on merge latency and disband latency of Network Level Connection Manager

The experiment to measure *Merge Latency* and *Disband Latency* is repeated for ten times. The experiment results are shown in Table 6-1.

No	Application Level Connection Manager		Network Level Connection Manager	
	Merge Latency (ms)	Disband Latency (ms)	Merge Latency (ms)	Disband Latency (ms)
1	24	33	57	31
2	24	48	36	32
3	33	28	68	36
4	59	37	70	35
5	46	32	61	28
6	24	44	47	57
7	52	38	68	27
8	31	36	33	28
9	37	43	44	48
10	45	27	32	50
Average	***37.5***	***36.6***	***51.6***	***37.2***

Table 6-1: Merge Latency and Disband Latency

As shown in Table 6-1, *Merge Latency* for the Application Level Connection Manager is 37.5 ms on average. The merge latency remains less than 60 ms, which means the CM manages to process the merge request in time for the receiver to receive the next multicast packet as a merge packet.

Disband Latency for the Application Level Connection Manager is 36.6 ms on average. The disband latency remains less than 60 ms, which means the CM manages to process the disband request in time so that the receiver stop receiving the next multicast packet as merge packet.

With reference to Table 6-1, *Merge Latency* and *Disband Latency* for the Application Level Connection Manager have shown improvement over *Merge Latency* and *Disband Latency* for the Network Level Connection Manager (previous implementation). To invoke groups merge or groups disbanding operation in the Network Level Connection Manager, end host has to send merge or disband request to 0-CMM in the form of UDP packet. 0-CMM then transforms this merge or disband request to *IGMP_Merge_Request* or *IGMP_Disband_Request* and send it to modified multicast router. This causes longer *Merge Latency* and *Disband Latency* for the previous work.

6.2.2.2 Groups Merging Overhead from End User Perspective

Latency is defined as the time taken for packets to get from sender to receiver. There are two main factors that contribute to latency:

- *Transmission delay*: the time sender takes to transmit the packets.

- *Propagation delay*: the time packets take to travel from one place to another at near the speed of light.

$$Latency_{\ without\ groups\ merging} = Transmission\ Delay + Propagation\ Delay$$

However, there is an overhead of groups merging operation that contributes to the latency. When Connection Manager receives a packet, it takes time to examine the packet and forward the packet if necessary. In the case of inter multicast-islands groups merging, overlay network delay also contributes to the overhead. This overhead is defined as *CM Processing Delay*.

$$Latency_{\ with\ groups\ merging} = Transmission\ Delay + Propagation\ Delay + CM\ Processing\ Delay$$

To measure the *CM Processing Delay*, first we need to measure $Latency_{\ without\ groups\ merging}$ and $Latency_{\ with\ groups\ merging}$. Assuming *Transmission Delay* and *Propagation Delay* remain constant, we can get *CM Processing Delay* by comparing the latency with and without groups merging,

In the next two sections, *CM Processing Delay* for intra multicast-island and inter multicast-islands are measured and analysed.

- ***CM Processing Delay for Intra Multicast-island Groups Merging***

To measure $Latency_{\ without\ groups\ merging}$, *Insignia* and *Mars* are connected to a 10 Mbps hub which connects to a multicast router as shown in Figure 6-4. *Insignia* sends multicast packets to group *X* in intervals of 60 milliseconds. *Mars* listens to group *X*. Subsequently *Mars* stops listening to group *X* after *n* packets have been received. The experiments is done for the case of n = 500 and 1000.

The time when *Mars* starts listening to group *X* is recorded and the time when all *n* packets have been received is also recorded. The difference between the time when all *n* packets have been received and the time when *Mars* starts listening to group *X* is the $Latency_{without\ groups\ merging}$.

To measure $Latency_{with\ groups\ merging}$, *Insignia* and *Mars* are connected to a 10 Mbps hub which connects to a multicast router as shown in Figure 6-4. *Neptune* is setup as Connection Manager to handle merge and disband requests. *Insignia* sends multicast packets to group *X* in intervals of 60 milliseconds. *Mars* listens to group *Y* and send a merge request to merge group *Y* to group *X*. Subsequently *Mars* stops listening to group *Y* after *n* packets have been received. The experiments is done for the case of *n* = 500 and 1000.

The time when *Mars* sends a merge request is recorded and the time when all *n* packets have been received is also recorded. The difference between the time when all *n* packets have been received and the time when a merge request is sent is the $Latency_{with\ groups\ merging}$.

We also measure $Latency_{with\ groups\ merging}$ of Network Level Connection Manager. The setup used is shown in Figure 6-5. Experiment for each case is repeated for ten times. The experiment results are shown in Table 6-2.

By computing the difference between $Latency_{with\ groups\ merging}$ and $Latency_{without\ groups\ merging}$, we can get *CM Processing Delay*. *CM Processing Delay* for Application Level and Network Level Connection Manager for the case of intra multicast-island groups merging are shown in Table 6-2 and plotted in a chart for easy comparison in Figure 6-6.

No	Latency without groups merging (ms)		Latency with groups merging of Application Level CM (ms)		Latency with groups merging of Network Level CM (ms)	
	500 packets	1000 packets	500 packets	1000 packets	500 packets	1000 packets
1	30020	60037	30054	60080	30043	60082
2	30021	60040	30051	60106	30042	60092
3	30024	60043	30052	60102	30045	60079
4	30023	60047	30047	60101	30040	60088
5	30020	60043	30050	60105	30043	60087
6	30018	60042	30049	60099	30045	60091
7	30020	60045	30048	60109	30040	60082
8	30022	60039	30050	60103	30046	60083
9	30019	60045	30053	60108	30041	60089
10	30025	60042	30051	60098	30042	60081
Average	***30021.2***	***60042.3***	***30050.5***	***60101.1***	***30042.7***	***60085.4***
CM processing delay			***29.3***	***58.8***	***21.5***	***43.1***

Table 6-2: Latency without groups merging and Latency with groups merging for Application Level and Network Level Connection Manager

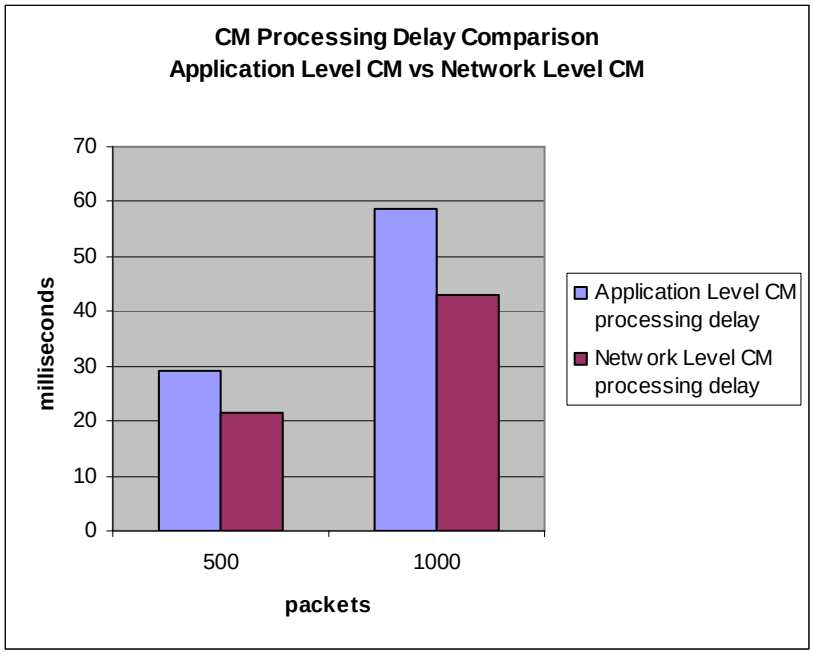

Figure 6-6: *CM Processing Delay* of Application Level CM and Network Level CM in the case of intra multicast-island groups merging

With reference to Figure 6-6, *CM Processing Delay* for the Application Level Connection Manager is longer than Network Level Connection Manager. In Application Level Connection Manager, packet forwarding to achieve merge effect is done at application level. Packets have to traverse up to the application level and forwarded to a group. In Network Level Connection Manager, packet forwarding to achieve merge effect is done at network level. Packets traverses up to the network level and forwarded to a group.

However, we believe that the potential benefits of transferring groups merging and groups disbanding capabilities from network level to application level significantly outweigh the performance penalty incurred. Application Level Connection Manager works without modifying IP Multicast routing protocols or changing the existing Internet Group Management Protocol (IGMP). It does not relying on multicast support from routers. Therefore, it can be easily deployed in the Internet.

- ***CM Processing Delay for Inter Multicast-islands Groups Merging***

The setup used in measuring the *CM Processing Delay* of Application Level Connection Manager for the case of inter multicast-islands groups merging shown in Figure 6-7.

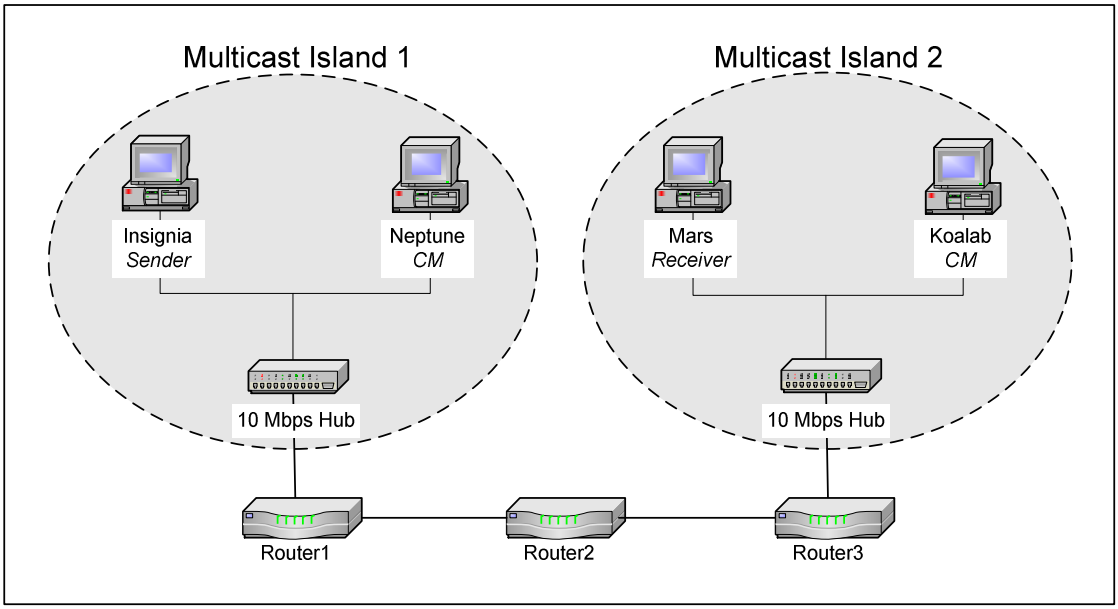

Figure 6-7: Setup used for measuring CM Processing Delay for inter multicast-islands groups merging

Insignia and *Neptune* are located at *multicast island 1*. *Insignia* acts as sender and *Neptune* is setup as Connection Manager. In *multicast island 2*, *Mars* acts as receiver and *Koalab* is setup as Connection Manager. *Router1*, *Router2* and *Router3* perform unicast routing only. IP Multicast routing is not enabled in *Router1*, *Router2* and *Router3*. Multicast *island 1* and *multicast island 2* are connected to each other by *Router1*, *Router2* and *Router3*.

To measure *Latency* $_{with\ groups\ merging}$, *Insignia* sends multicast packets to group *X* in intervals of 60 milliseconds. *Mars* listens to group *Y* and send a merge request to

merge group *Y* to group *X*. Subsequently *Mars* stops listening to group *Y* after *n* packets have been received. The experiments is done for the case of *n* = 500 and 1000.

The time when *Mars* sends a merge request is recorded and the time when all *n* packets have been received is also recorded. The difference between the time when all *n* packets have been received and the time when a merge request is sent is the *Latency with groups merging*.

To measure *Latency without groups merging*, IP Multicast routing is enabled in *Router1*, *Router2* and *Router3*. *Insignia* sends multicast packets to group *X* in intervals of 60 milliseconds. *Mars* listens to group *X*. Subsequently *Mars* stops listening to group *X* after *n* packets have been received. The experiments is done for the case of *n* = 500 and 1000.

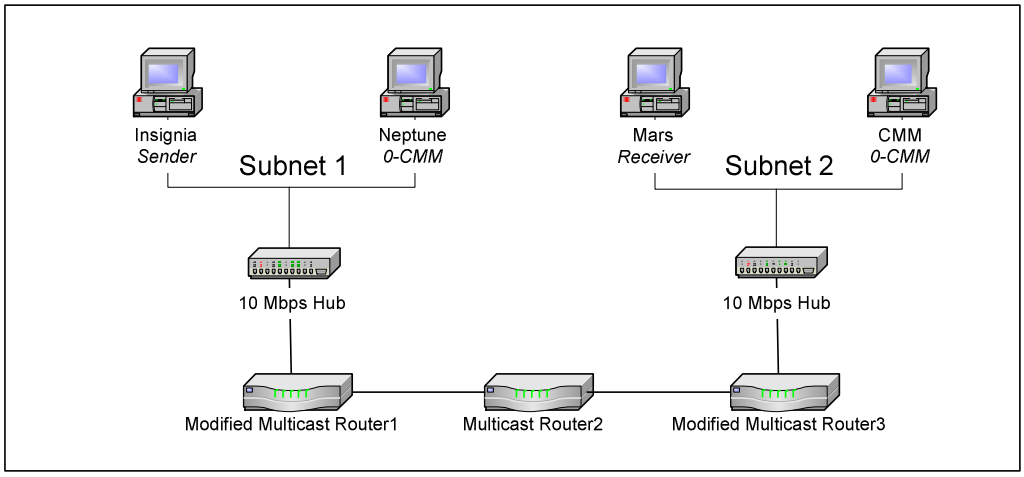

Figure 6-8: Setup for measuring *Latency with groups merging* of Network Level Connection Manager in distributed network

We also measure *Latency with groups merging* of Network Level Connection Manager in distributed network. The setup used is shown in Figure 6-8. Experiment for each case is repeated for ten times. The experiment results are shown in Table 6-3.

By computing the difference between the latency with and without groups merging, we can get *CM Processing Delay*. *CM Processing Delay* for Application Level for the case of inter multicast-island groups merging and *CM Processing Delay* for Network Level for the case of distributed network groups merging are shown in Table 6-3 and plotted in a chart for easy comparison in Figure 6-9.

No	Latency without groups merging (ms)		Latency with groups merging of Application Level CM (ms)		Latency with groups merging of Network Level CM (ms)	
	500 packets	1000 packets	500 packets	1000 packets	500 packets	1000 packets
1	30029	60060	30089	60173	30063	60125
2	30025	60059	30085	60178	30060	60128
3	30030	60054	30088	60169	30062	60119
4	30027	60061	30090	60175	30058	60127
5	30033	60058	30083	60176	30065	60126
6	30028	60062	30088	60175	30059	60130
7	30029	60059	30091	60177	30068	60126
8	30031	60055	30087	60174	30066	60125
9	30027	60057	30088	60172	30064	60131
10	30032	60058	30084	60180	30062	60121
Average	***30029.1***	***60058.3***	***30087.3***	***60174.9***	***30062.7***	***60125.8***
CM processing delay			***58.2***	***116.6***	***33.6***	***67.5***

Table 6-4: Latency without groups merging, Latency with groups merging for Application Level CM for inter multicast island groups merging and Latency with groups merging for Network Level CM for distributed network

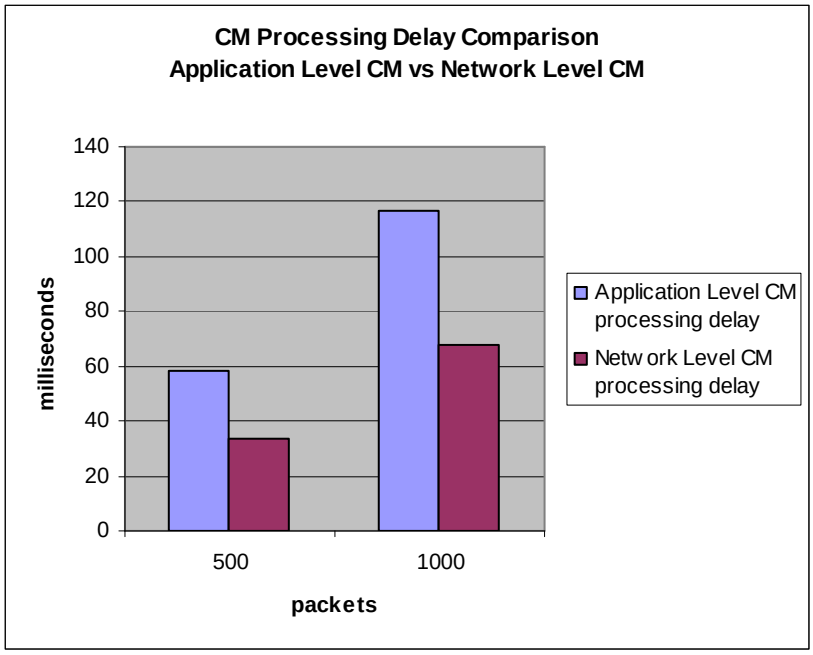

Figure 6-9: Application Level CM Processing Delay in the case of inter multicast-island groups merging and Application Level CM Processing Delay in the case of distributed network groups merging

With reference to Figure 6-9, Application Level CM Processing Delay in the case of inter multicast-island groups merging is longer than Network Level CM Processing Delay in the case of distributed network groups merging. This is because Application Level CM does packet forwarding at application level while Network Level CM does packet forwarding at network level.

In the case of Network Level CM distributed network groups merging, packets travelling from sender to receiver going through address translation once i.e. at the modified router attached to the receiver's subnet as shown in Figure 6-10.

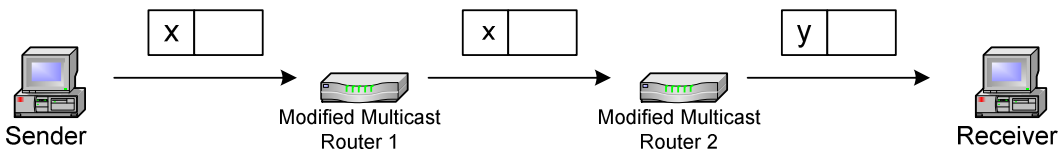

Figure 6-10: Packets travelling from sender to receiver have to go through address translation once in Network Level CM distributed network groups merging

In the case of Application Level CM inter multicast-island groups merging, packets travelling from sender to receiver have to go through address translation twice, one at the CM on the sender's immediate multicast-island and another at the CM on the receiver's immediate multicast-island as shown in Figure 6-11.

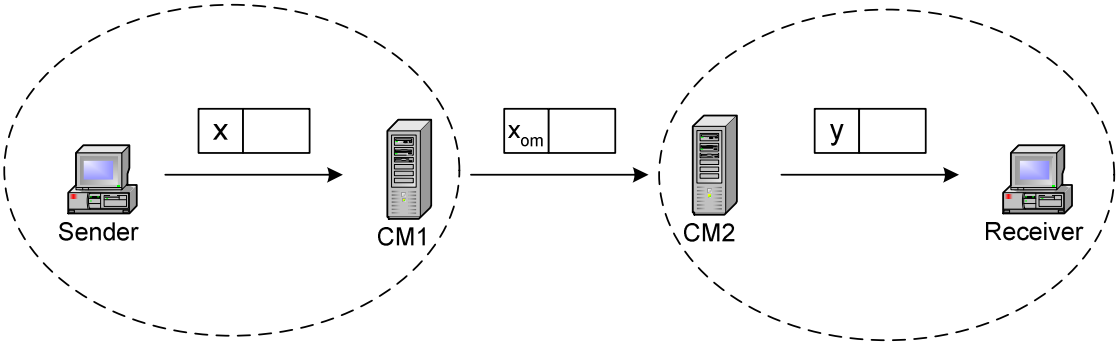

Figure 6-11: Packets travelling from sender to receiver have to go through address translation twice in inter multicast-islands groups merging

This is another factor that contributes to longer CM Processing Delay in the case of Application Level CM inter multicast-island groups merging. However, in the real world where IP Multicast is not available Network Level CM cannot be used. We believe that the potential benefits of providing groups merging and groups disbanding capabilities in non multicast-supported network significantly outweigh the performance penalty incurred.

We also observe that Application Level Connection Manager CM Processing Delay for inter multicast-island groups merging is longer than Application Level Connection Manager CM Processing Delay for intra multicast-islands groups merging. In intra multicast-island groups merging, packets travelling from sender to receiver have to go through address translation once as shown in Figure 6-12.

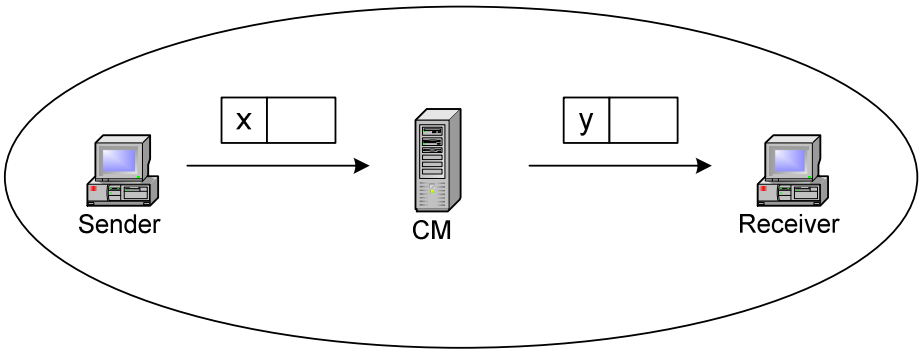

Figure 6-12: Packets travelling from sender to receiver have to go through address translation once in intra multicast-island groups merging

In inter multicast-islands groups merging, packets travelling from sender to receiver have to go through address translation twice as shown in Figure 6-11. This contributes to longer CM Processing Delay in the case of inter multicast-islands groups merging.

CHAPTER 7
DISCUSSION

7.1 SCALABILITY ISSUE

Having one CM to serve the whole multicast-island is fine for small multicast-island as the data traffic may not be too high. For a large multicast-island, this can lead to potential performance problem. When there is only one CM in a multicast-island, it serves as the single entry/exit point. The CM may become a bottleneck when the traffic coming into or going out of the island is high. Hence, to have one CM per multicast-island may lead to a single point of failure and may not be scalable. Two solutions to the above problem are discussed in this section.

The first solution is to allow multiple CMs to serve each multicast-island. Several CMs share the workload of handling merge/disband operation and forwarding data. Although this solves the scalability issue, it introduces another problem. Having more than one CM to do packet forwarding may lead to packet duplication problem. The following paragraphs explain this problem in greater details and propose approach to solve it.

The packet duplication problem is shown in Figure 7-1. Suppose group *X* wishes to merge to group *Y*. x_1 sends out a MERGE message to request to merge group *Y*. Both CM_1 and CM_2 receive this request, and forward group *Y*'s packets to group *X* in response to the MERGE request. As a result, group *X* members receive duplicate packets.

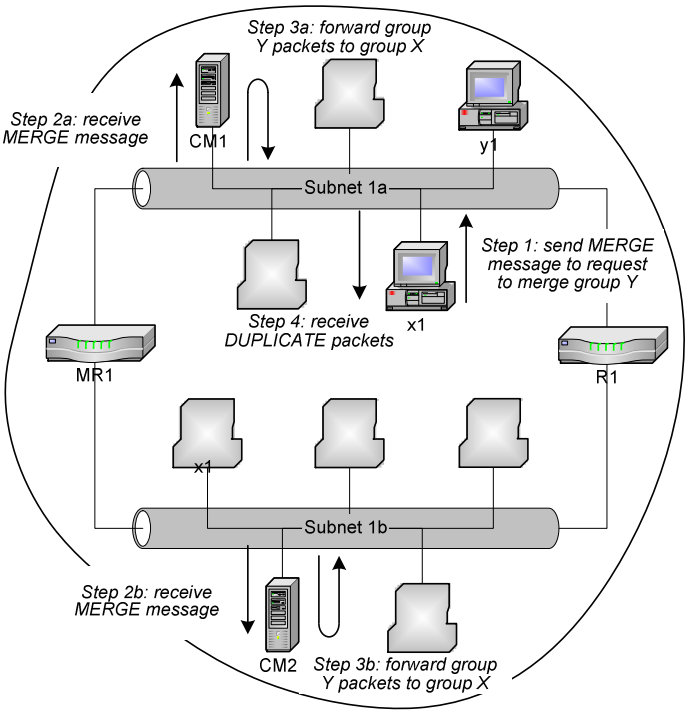

Figure 7-1: Packets duplication due to having more than one CM to do packets forwarding

To avoid packet duplication problem, it is important to ensure that there is only one CM forwarding multicast packets for a particular merge request. A single CM has to be selected among multiple CMs in the multicast-island to process a particular request.

When an authorised host wants to issue a group merge request, it first has to select a CM among multiple CMs in its multicast island to handle the request. The objective of the selection is to divide the amount of work between two or more CMs so that no single CM is overwhelmed. The strategy to achieve the objective is to select the lightest load CM to handle the group merge request.

Using IP multicast, the host sends a request for service to the CMs which receive and evaluate the request. Each CM sends a response to the client, indicating its bid to handle the request. The bid is weighted so that it is inversely proportional to CM's current load. The host collects bids from CMs for a period of time. After the timeout, it initiates the actual service connection using unicast to the CM with the highest bid.

When more than one CM responds with highest bid, the host uses the first response it received. This selection algorithm chooses the CM with lightest load to handle a request for load balancing purpose.

The second solution is to elect a member host to serve as CM for its group in each multicast island. When an end host joins a group, it first checks the local well-known group directory for an announcement for CM of the group. If no such announcement is present, it assumes the role of CM for the group in its immediate multicast-island.

In this solution, the CM is only responsible for handling join request and merge request to its group. Hence, for every active group there is a member host elected to serve as CM in each multicast-island where member of the group exists.

7.2 RELIABILITY ISSUE

7.2.1 Fault Tolerance Capability

If there is only one Connection Manager (CM) in a multicast-island and it fails, groups merging and groups disbanding services in the multicast-island will no longer be available to end hosts. As the availability of CM is crucial, it is important to have a backup CM in the multicast-island to make the services available even if the main CM fails.

- ***Master-slave replication strategy:*** To improve the availability of CM, a master-slave replication strategy is used. The strategy revolves around two types of entities: master and slave.

  The master CM is the only one that executes join, groups merge and groups disband requests. Slave CM may not execute such requests. When the slave CM

receives such requests, it updates *join_list, bridge_table* and *merge_table* based on the requests. In this way, slave CM keeps a same copies of *join_list*, *bridge_table* and *merge_table* as master CM.

- ***Master election process:*** At the time when a CM starts, it operates as slave. Then it tries to find the master CM using a pre-defined multicast address. If it can find a master CM, it continues operate in slave mode, otherwise it switches to operate in master mode.

  Slave CMs polls master CM periodically. When master CM fails to reply after certain time, the slave CM takes over the master CM functionality.

7.2.2 Reliable Control Channel

CMs use overlay multicast on CMS to exchange control information. As the overlay multicast is implemented using unicast technology, CMS control channel can be established using reliable transport protocol such as TCP.

Within multicast-island, IP Multicast is used as communication channel for transmission of control information. Although IP Multicast is not a reliable protocol, high quality network in the multicast-island makes the possibility of the control message being lost or damaged is relatively small.

The reliability of the control channel is further enhanced by sending join message and merge message periodically to cover the possibility of the initial join message being lost or damaged.

7.3 PLACEMENT OF CONNECTION MANAGER

To provide groups merging and groups disbanding services in a multicast-island, a dedicated host serving as Connection Manager (CM) has to be setup in the multicast-island. Because a dedicated host is used, the provision of groups merging and groups disbanding services is more stable. However, the dedicated host serving as CM has to be strategically placed in the multicast-island.

It is a good strategy to place CM to be as near as possible to sender. However, the location of the sender is often not known in advance. It is not an easy task to place a CM strategically in such a way that it is optimum for every user of groups merging and groups disbanding services.

Instead of using a dedicated host as CM, sender is elected to serve as CM to get around the CM placement issue. The CM is only responsible for handling join request and merge request to its group. Hence, for every active group there is a member host elected to serve as CM in each multicast-island where member of the group exists.

7.4 SECURITY ISSUE

The proposed solution is fairly open for anyone to use. The openness of the solution allows a variety of models to be built to leverage on groups merging and groups disbanding such as sender initiated groups merging/groups disbanding, third-party initiated groups merging/groups disbanding and session level access control.

Due to the proposed solution openness, there is no restriction on who has the authority to invoke groups merging and groups disbanding operations. This may lead to serious Denial of Service (DOS) attack.

To prevent unauthorised access to groups merging and groups disbanding capabilities, every groups merging/groups disbanding requests is appended with a digital signature. The request is only proceed if the digital signature is verified and authenticated.

7.5 PARTIAL GROUPS MERGING

Currently, when a request to merge group *A* and group *B* is received by Connection Manager, it assumes all members in group *A* intend to merge to group *B*. However, there are occasions when some members in group *A* do not want to merge to group *B*.

A possible approach to facilitate partial group merging operation is to cooperate with Stream Manager in OCTOPUS. Recall that when group *A* merges to group *B*, all members of *A* also receive group *B* packets in addition to those of group *A*. Stream Manager can differentiate data from different sources by the session ID in the RTP header. For each data source, Stream Manager creates an end point that delivers the data to application. For members of group *A* that do not wish to merge to group *B*, no end point for group B data are created by Stream Manager. Hence, no group *B* data is delivered to members of group *A* that does not wish to merge to group *B*.

CHAPTER 8
CONCLUSION AND FUTURE WORK

Although IP Multicast has been proposed for more than a decade, its deployment is still limited to network domains under single administrative control. In addition, IP Multicast communication model is largely limited to intra-group communication, where a host is allowed to become member of a group to send or receive from the group. IP Multicast does not provide a service for a whole group to become member of the group to allow inter-groups communication.

A novel method is proposed to provide groups merging and groups disbanding to support group-to-group communication in the Internet. The proposed solution is designed to work without modifying IP Multicast routing protocols or changing the existing Internet Group Management Protocol (IGMP). It does not relying on multicast support from routers. Therefore, it can be easily deployed in the Internet.

This solution facilitates more creative development of groupware applications. Some potential applications of the solution include: (1) managing group interactions in a dynamic Computer Supported Cooperative Work (CSCW) environment such as e-learning, (2) dynamic dissemination of notifications in publish/subscribe system, (3) supporting heterogeneous QoS management of multimedia streams encoded with a layered coding scheme.

An architecture to support groups merging and groups disbanding in the Internet called Application Layer Connection Management Framework has been proposed. This architecture consists of two main components: Connection Manager and Connection

Management System. Connection Manager is a distributed component for keeping track of group memberships in its immediate multicast-island, bridging each group in its immediate multicast-island to its respective group in other multicast-islands and managing request to merge/disband a group to/from another group. Connection Management System is formed by interconnected Connection Managers across multiple multicast-islands for exchanging control information and forwarding data.

Protocols for handling join message, join status expiration, bridge message, bridge status expiration, merge message and disband message have been designed. These protocols are explained with the help of flow diagrams.

A prototype has been implemented successfully to demonstrate the feasibility of the proposed solution. The implemented prototype has demonstrated the groups merging and groups disbanding capabilities both within and beyond a multicast island. An experiment has been conducted to measure *Merge Latency*, *Disband Latency* and *CM Processing Delay*. Performance evaluation results have shown that the implemented prototype manages to perform reasonably well.

One of the future works is to conduct a larger scale performance evaluation using simulation software. Also, the performance of the implemented prototype can be improved by reducing the CM Processing Delay. Implementing packet forwarding module at kernel level can reduce the CM Processing Delay. Another future work is to implement multiple Connection Manager per multicast-island to improve scalability.

REFERENCES

[1] S. Deering and D. Cheriton, "Multicast Routing in Datagram Internetworks and Extended LANs", *ACM Transactions on Computer Systems*, May 1990.

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

[3] K. Y. Koh, "Supporting Group Merging and Group Disbanding in IP Multicast", Masters Thesis, School of Computing, National University of Singapore, July 2002.

[4] D.Waitzmann, C. Patridge, and S.Deering "Distance Vector Multicast Routing Protocol", *RFC 1075*, 1988.

[5] J.Moy, "Multicast Extensions to OSPF", *RFC 1584*, March 1994

[6] Andrew Adams, Jonathan Nicholas and William Siadak, "Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised)", *IETF Internet draft*, February 2003.

[7] Bill Fenner, Mark Handley, Hugh Holbrook and Isidor Kouvelas, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", *IETF Internet draft*, 2 March 2003.

[8] S. Deering, "Host Extension for IP Multicasting", *RFC 1112*, August 1989.

[9] W. Fenner, "*Internet Group Management Protocol, Version 2*", *RFC 2236*, November 1997.

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

[11] Jon Postel, "Transmission Control Protocol", *RFC 793*, September 1981.

[12] L.F. Cabrera, M.B. Jones and M. Theimer, "Herald: Achiving a Global Event Notification Service." In *Proceeding of the 8th Workshop on Hot Topics in Operating Systems*, 2001.

[13] Antony I. T. Rowstron, Anne-Marie Kermarrec, Miguel Castro, and Peter Druschel, "SCRIBE: The design of a large-scale event notification infrastructure," in *Networked Group Communication*, pp. 30-43, 2001.

[14] H. Gharavi, "Multilayer subband-based video coding", IEEE Transaction on Communication, vol. 39, no 9, pp. 1288-1291, Sept. 1991.

[15] N. Shacham, "Multipoint communication by hierarchically encoded data", In *Proceeding of IEEE INFOCOM*, San Fransisco, CA, pp. 124-131, May 1990.

[16] B. J. Vickers, C. Albuquerque, T. Suda, "Adaptive Multicast of Multi-Layered Video: Rate-Based and Credit-Based Approaches", In *Proceeding of IEEE INFOCOM*, April 1998.

[17] C. H. Zhang, et al, "OCTOPUS: An adaptive group communication environment for multimedia applications", *IEEE International Conference on Networks 2002*, Singapore, August 27-30, 2002.

[18] C. H. Zhang, et al, "OCTOPUS: A middleware for multimedia communication", *6th IASTED International Conference Internet and Multimedia Systems and Applications 2002*, Kauai, Hawaii, USA, August 12-14 2002.

[19] Sun Microsystems, Inc. "JINI™ Architectural Overview" Technical white paper, January 1999.

[20] J. H. Saltzer, D. P. Reed and D. D. Clark, “End-to-end Arguments in System Design”, *ACM Transactions on Computer Systems, Volume 2, Issue 4,* November 1984.
[21] H. Erikson, “MBONE: The Multicast Backbone”, *Communication of the ACM*, pp. 54–60, August 1994.
[22] Aiguo Fei, Jun-Hong Cui, Mario Gerla and Michalis Faloutsos, “Aggregated Multicast: an Approach to Reduce Multicast State”, In *Proceedings of Sixth Global Internet Symposium (GI2001) in conjunction with Globecom 2001*, San Antonio, Texas, USA, November 25-29, 2001.
[23] Aiguo Fei, Jun-Hong Cui, Mario Gerla and Michalis Faloutsos, “Aggregated Multicast with Inter-Group Tree Sharing”, In *Proceedings of Third International Workshop on Networked Group Communications (NGC2001)*, UCL, London, UK, November 7-9, 2001.
[24] D. Thaler and M. Handley, “On the Aggregatibility of Multicast Forwarding State”, In *Proceedings of IEEE INFOCOM*, March 2000.
[25] D. Pendarakis, S. Sherlia, V. Dinesh and W. Marcel, “ALMI: An Application Level Multicast Infrastructure”, In *Proceedings of the 3rd USNIX Symposium on Internet Technologies and Systems*, San Francisco, CA, USA, March 2001.
[26] Y. Chu, S.Rao and H. Zhang, “A Case for End System Multicast”, *ACM SIGMETRICS*, June 2000.
[27] Y.H. Chu, S.G. Rao, S. Seshan and H. Zhang, “Enabling Conferencing Applications on the Internet using an Overlay Multicast Architecture”, In *Proceedings of ACM SIGCOMM*, August 2001.
[28] B. Zhang, S. Jamin and L. Xhang, “Host Multicast: A Framework for Delivering Multicast to End Users”, *IEEE INFOCOMM 2002*, New York, NY, June 2002.
[29] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, “Scalable Application Layer Multicast,” In *Proceedings of ACM SIGCOMM*, August 2002.
[30] J. Liebeherr and Tyler K. Beam, “HyperCast: A Protocol for Maintaining Multicast Group Members in a Logical Hypercube Topology”, In *Proceeding of First International Workshop on Networked Group Communication (NGC '99)*, *Lecture Notes in Computer Science*, Vol. 1736, pp. 72-89, July 1999.
[31] Sun Microsystems, “The Source for Java Technology”, http://java.sun.com
[32] Sun Microsystems, “JMStudio User’s Guide”, http://java.sun.com/products/java-media/jmf/2.1.1/jmstudio/jmstudio.html
[33] Sun Microsystems, “Java Media Framework API”, http://java.sun.com/products/java-media/jmf/
[34] H. Schulzrinne, S. Casner, R. Frederick and V. Jacobson, “RTP: A Transport Protocol for Real-Time Applications”, *RFC 1889*, January 1996.