INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

Adaptive Multimedia Synchronization and Scalable Multipoint Routing with Mobility Support

by

Changdong Liu

A dissertation submitted to the Graduate Faculty in Engineering in partial fulfillment of the requirements for the degree of Doctor of Philosophy, the City University of New York

1998

This manuscript has been read and accepted for the Graduate Faculty in Engineering in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

February 20, 1998
Date

Professor **Myung J. Lee**
Chair of Examining Committee

February 20, 1998
Date

Dean **Gerard G. Lowen**
Executive Officer

Professor **Tarek N. Saadawi**
City University of New York

Professor **Joseph Barba**
City University of New York

Professor **Sanghamitra Basu**
City University of New York

Dr. **Ning H. Lu**
ITT Defense & Electronics

Supervisory Committee

The City University of New York

Abstract

Scalable Multipoint Routing with Mobility Support

by

Changdong Liu

Advisers: Professor Myung J. Lee, Professor Tarek N. Saadawi

The paradigm shift in the Internet towards supporting multimedia, multipoint, mobility with QoS guarantee captures the emerging technologies in packet-switched internetworks. The work reported herein presents innovational research results in this new direction.

Application level QoS is always appreciated for multimedia/real-time applications over packet-switched internetworks. A efficient, yet simple to implement, adaptive multimedia synchronization algorithm has been developed. The user defined QoS specifications can be directly mapped into the design parameters of the algorithm. The essence of the algorithm is partitioning the vicinity of the arrival epochs of packets into regions, counting arrivals at each region and adjust the PlayBack Clock (PBC) accordingly. The algorithm is adaptive to network changes, eliminates the need for a global clock, and is immune to the clock frequency drift.

Existing multipoint and mobile support approaches face the same problem — scalability, which constantly challenge the masterminds behind the Internet. Early work on multipoint support uses source-based tree technique, which is tolerable when group members are densely packed and bandwidth is plentiful but not proper for wide area networks. Relatively new approaches use shared-tree technique, in which a center-specific

tree spanning all members of each group is used to deliver multicast traffic for that group. But how to efficiently select the center for a group is still an open issue.

The proposed work, Core-Manager based Multicast Routing (CMMR), takes a new approach to the issues of center-specific tree creation and management. The basic idea behind CMMR is that the Core-Manager keeps tracking cores of each multicast delivery tree in order to maintain a brief image of the tree. Consequently, new members will almost always be directed to a nearby core to join the group so that tree cost is well confined. However, not every new member has to actually consult the Core-Manager to join the tree due to a self-growing scheme, which further lowered the control overhead. Further improving on scalability is sought by using a hierarchical architecture, which requires the multicast addresses be administratively scoped in accordance with the hierarchy. Given such a hierarchical CMMR multicast autonomous region, different approaches to reconnecting a mobile multicast group member host to its group tree after each moving into a new subnetwork are proposed.

To Yi Yang

Acknowledgments

I wish to express my deepest appreciation to my advisers Professor Myung J. Lee and Professor Tarek N. Saadawi, for your constant encouragement, patience, support and guidance throughout the course of my study. You are the best professors I have ever met in my life. You are excellent professors as well as wonderful friends.

I was aided in the research by a grand from the Advanced Telecommunications/Information Distribution Research Program Consortium, an Army Research Laboratory organized Federated Laboratory. I am extremely grateful to Professor Saadawi and Professor Lee, the principal investigators of the Consortium at the City College, for your generosity in administrating the grant.

My sincere gratitude is to Professor Joseph Barba and Professor Sanghamitra Basu, Dr. Ning H. Lu, members of my Supervisory Committee, for taking time to review this paper and making valuable comments on this work. Deep appreciation also goes to Professor Gerard G. Lowen, the Executive Officer of the Ph. D. program in Engineering, for steady supporting and handling administrative details in such a warm and friendly manner.

Many other individuals helped me along the way. In particular, I would like to mention the remarkable members and my fellow students at the Computer Communications Lab: David, Hussein, Jay, Khaled, Peixiang, Simon, Wei and Yong. My thanks to the friendly and helpful administrative staffs and technicians in the Electrical Engineering Department, with your assistance my study life at the department became pleasant.

Finally, but most importantly, I would like to express my gratitude to Shufen and Minshi, my parents for your deepest love and most dependable support, especially to Dr. Yi Yang, my dearest wife, always at my side.

Contents

List of Tables

List of Figures

1. Introduction

People and their machines should be able to access information and communicate with each other easily and securely, in any medium or combination of media — voice, data, image, video, or multimedia — any time, anywhere, in a timely, cost-effective fashion. The trend of today's communication networks development towards this end is Multimedia, Multipoint[1], and Mobility with Quality of Service (QoS) guarantee (M3Q).

In the computer era, people's desire to communicate in natural, expressive and efficient fashions formed the primary driving force behind the provision of M3Q support in the global communications infrastructure. By nature, humans are able and *willing* to acquire information through the eyes, the ears (and some times the nose) simultaneously. Even with the electronic media, people like to keep their natural way of communications. This passion constituted the motivation to carry digital audio and video over packet-switched networks. Multipoint communication is one of the oldest forms of communication among humans. It has long been recognized that communicating a message to multiple recipients simultaneously is a very efficient method of getting the message across. This understanding stimulated the efforts of multipoint support in data networks. The desire for freedom from being physically connected to the underlying network grows up rapidly as computing devices have shrunk in size, weight, and power. This open up the enormous thrust towards the portable and wireless communications.

---

[1] The term "multipoint" is used to refer in general to all forms of communication with multiple participants, regardless of semantics. The term "multicast" having been used almost synonymously is more traditional, even though it is often used to denote point-to-multipoint communication. Throughout this document these two terms will be used interchangeably.

In vision of the M3Q themes, a great share of development efforts in the Internet devotes to QoS enhancement. Original design of IP, which forms the foundation of the Internet, adhered to *best effort* principle. It is good and efficient for non-real-time data messages. With the emerge of multimedia applications in the Internet, the classical networking techniques are falling short of their wishes despite their hard "efforts". While explicit QoS measurements at the network layer [1, 2] are crucial to the IP's survive in its competition with other techniques that have QoS supports like ATM, higher layer solutions are always appreciated since they may work alone before QoS supports at other layers become available. The adaptive multimedia synchronization algorithm [3] is a typical application layer solution to the QoS issue.

In the early days of electronic communication both point-to-point and multipoint communication were possible. Telephone is a typical example of point-to-point communication over various ranges, and radio was common means for multipoint communication. This multipoint communication was mostly simplex in nature with content being spread from a source to multiple receivers within certain covered range with little or no signaling being returned by the receivers.

In this context it is not surprising that early research of data networks and packet switching technology only focused on point-to-point communication. Even with the development of local area network technology that could easily provide multipoint communication, the focus was on how to provide point-to-point communication with multipoint communication supported almost as a later thought.

Multicasting in packet-switched networks conserves bandwidth by forcing the network to do packet replication only when necessary. But, with network layer [4] support

for multipoint applications lacking in the early packet-switched networks, many multipoint applications were developed assuming only point-to-point network support [5, 6]. This legacy continues today even though the inefficiencies are obvious.

For a rapidly increasing number of people, the largest packet-switched network infrastructure, the Internet, has become a basic means of communication. Meanwhile, over the last few years, there have been significant advances in personal computing and personal communications, including the introduction of powerful notebook computers and portable data/voice terminals. Stimulated by the increasing importance of portable computing applications in both business and consumer applications, migrating multimedia applications to portable devices grips more and more Internet community's interests.

Certainly, the most fundamental problem for connectionless internetowrking is routing packets between networks [7]. Because of their size, and geographic and administrative diversity, routing protocols are also subject to particular constraints in the Internet. This challenging work becomes more challenging when multipoint application *and* mobile services are demanded. Given that classic internetwork techniques [8–11] were focused on point-to-point communication with implicit assumption that hosts attach to the internetworks at long-lasting fixed points, some hard questions emerge.

Given the importance of multipoint applications would it make more sense to think of a entirely new infrastructure to support multipoint communication first and support point-to-point only as a special case? Or will we always need separate infrastructures to support the two forms efficiently? The answer to either question is no, since any attempt of completely getting rid of the existing infrastructures would not be accepted by the users. Then the next question would be that in supporting multipoint communication

how much of the existing point-to-point support can be reused? Literally, multipoint applications can be supported by using only point-to-point infrastructure in spite of its inefficiency. Some of the early work on multipoint communication used this reuse for multipoint solutions. For example, the reverse path forwarding technique, later adopted as the basis of Distance Vector Multicast Routing Protocol (DVMRP) [5] uses the same routing tables as in point-to-point routing. Multicast Extensions to Open Shortest Path First (MOSPF) [6], as its name says, is a direct extension to the point-to-point OSPF [9, 11].

What are the exact semantics of multipoint communication [12] with mobility support? The semantics of point-to-point communication are very clear: For every message there is one and only one recipient with a unique address. Reception reliability, packet ordering, and end-to-end flow control requirements are straightforward to define relative to this one receiver. The added dimensions of multiple receivers and their movement complicate the situation considerably [13–15]: How should this group of receivers be addressed at each level? How should the in-session applications be assured continuity when the link-level connections are not continuous? What is the adaptability the systems need in their architectures and algorithms?

Multipoint and mobile applications demand current communication protocols be further enhanced. Such applications cover a very wide spectrum, including software distribution, replicated database update, command and control systems, multimedia conferencing, computer supported collaborative work, and distributed interactive simulation (DIS), wireless information systems, and ubiquitous computing [16, 3, 17]. In the near future, the Internet users are expected to be increasingly mobile, and demand various

forms of multimedia communications over widely different network technologies and/or topologies. As computing devices have shrunken in size, weight, and power, untethering the communications link will make possible the next step in the evolution of computer environments, in which computing resources can be used more flexibly, because users can be freed from being physically connected to the underlying network. Therefore, the key issue here is to build the underlying infrastructure to support wireless communications.

Unfortunately, until recently, communication systems built around the Internet architecture were designed to support point-to-point services, where the end equipment was assumed attached to the network at permanent point. The majority of today's Internet applications rely on point-to-point transmission. The utilization of point-to-multipoint transmission has traditionally been limited to local area network applications. Point-to-point communication often depends on implicit knowledge that is no longer valid for group communication with mobile members. In a multipoint session, any participant can decide whether, where, and when it wishes to join or leave the session. The join and leave operations have to be simple, of little (if there is any) side effect on the other participants and minimal burden on the networks, so as to ensure feasibly scale to large group membership over the Internet region. Nevertheless, the success of the Internet in supporting point-to-point applications implies that the Internet Protocol (IP) suite will continue to play a dominating role in the future. Fortunately, over the past few years the Internet has seen a rise in the number of new applications that rely on multicast transmission. While some modifications to the current internet protocols is an acceptable approach during the transient period to supporting multipoint applications *or* mobile applications, innovatively designed protocol suite that unifies solutions to both multipoint

and mobile applications is absolutely needed for the new generation Internet.

It is the goal of this proposed research work to develop an approach to scalable multipoint services in highly mobile networks.

2. Higher layer Quality of Service Control: Adaptive Multimedia Synchronization

QoS is a broad issue and can addressed from many different points of view. While network layer efforts, such as QoS routing [1] and resource reservation protocol (RSVP) [18], are important, the adaptive multimedia synchronization algorithm to be presented below is a measure towards the same objective but at application layer. It can of course work together with other QoS measures at different layers as well as individually in an environment that lacks other means of QoS support.

Research on software architectures of multimedia synchronization are reported in [19, 2, 20]. In many studies, the software architecture issue is well contemplated, however, its implementation is rarely published. Although some synchronization schemes perform pretty well under certain conditions, they reveal limitations in other circumstances. For example, work in [21] provides an adaptive technique to achieve minimal voice delay, but it requires special measurement packets to estimate the mean network delay at the beginning of each silence interval. Work in [22] intends to give a general solution to the synchronization problem, but it is based on synchronized network clocks and assumes that messages are delivered reliably at least for initialization. Work in [23] copes with short-term network congestion, but applies only to delivering stored multimedia to a user. And work in [24] minimizes perceptible degradations in the quality of media playback, but needs feedback. In contrast, the adaptive synchronization mechanism presented here is dynamically adaptive to network changes, works with any underlying delay distribution, does not need a global clock or synchronized clocks, is immune to clock offset and/or clock frequency drift, does not need any feedback, is adequate to real-time interactive

environment, and provides optimal delay/buffering for a given QoS requirement, while it has a simple structure.

2.1 Adaptive Multimedia Synchronizations

The proposed adaptive multimedia synchronization algorithm is adaptive in many respects. It is adaptive to any network changes: improvement or degradation, long-term or short-term fluctuations, and different delay statistics. It is also adaptive to different types of traffic: periodic or non-periodic, continuous or noncontinuous (talkspurt-silence), constant rate or variable rate. In addition, it easily fulfils a variety of QoS requirements, such as delay, jitter, and loss ratio.

2.1.1 Problems and Principles

Two multimedia synchronization problems, intramedium synchronization and inter-media synchronization, are to be addressed. For real-time applications like teleconference, absolute delay and delay jitters are major concerns. Even when only a single medium is involved, the network inevitably introduces some variation in the delay of the delivered objects. At the receiver side the application attempts somehow to faithfully recover the object. *Faithfully recover* in the context of synchronization means that the temporal relationship among data at the source is preserved when they are played back at the sink. How to do it is the very question that should be answered by the synchronization mechanism. For intramedium synchronization the most intuitive solution may be to buffer the incoming data and then replay them at some fixed offset delay from the original generation time at the source so that every object experiences an equalized delay. This fixed offset delay must be larger than the maximal possible delay that any object may experience. In this sense synchronization is nothing but adding the controllable delay.

2.1.2 General Approach

By examining the synchronization problem in perspective, two arguments are made: First, the strongest temporal relationship among objects exists between adjacent ones since correlation between any two objects decreases as the time interval between them increases. Therefore, it is meaningless to retain temporal relation for two objects which are far apart in time. Second, small error in restoring the temporal relationship is tolerable. Actually, because so many random factors are involved, introducing some errors in synchronization is inevitable.

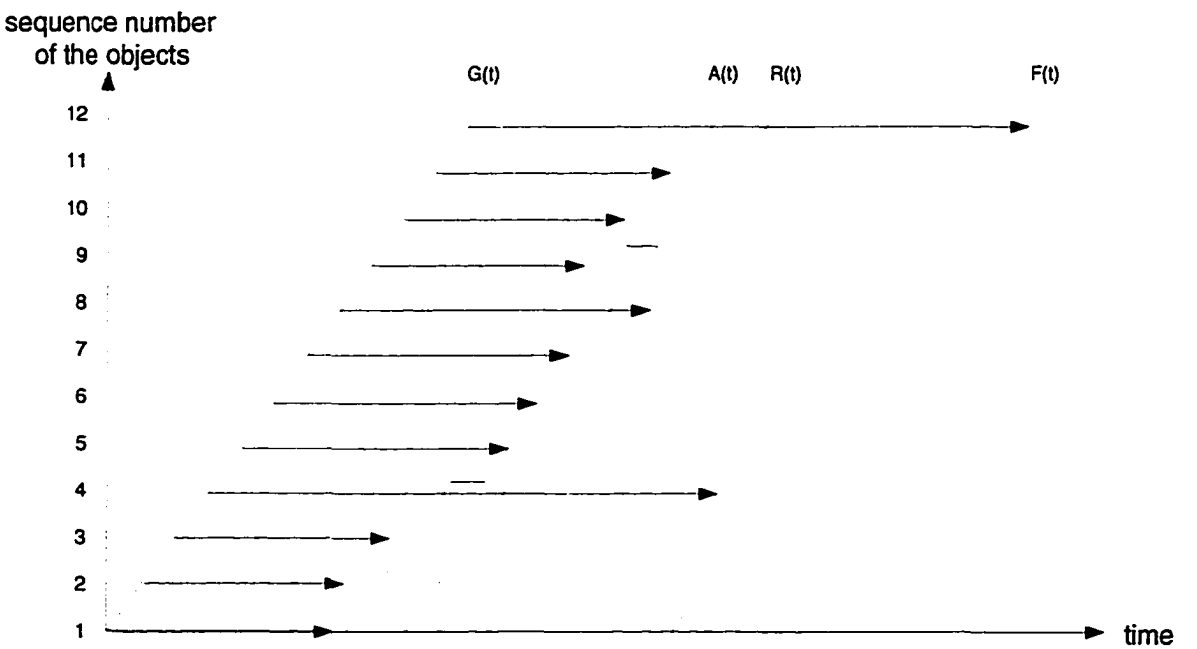


Figure 2.1 Playback times set up based on different criteria. G(t): objectis generation time; A(t): adaptive real-time playback time; R(t): real-time playback time; F(t): faithful playback time.

If the constraint of being faithfully recovered can be relaxed, it can exploit a lot from the two arguments made above. From the first: instead of using a unique reference time point, the application can adjust it from time to time. From the second: instead of

dropping all the data arriving after the playback point (which creates incomplete signal) the application may still accept those data that arrive *a little bit* after the playback point (e.g., object number 8 with respect to the playback point *R(t)* in Figure 2.1 for recovering the object even if the object is somewhat distorted. One may feel receiving something is better than nothing in some situations like a teleconference, where the users generally favor cadence altered voice over the choppy, incomprehensible one. Therefore, the goals for adaptive synchronization are set as: *find the optimal playback point based on which the original object can be* piecewisely *faithfully recovered with acceptable distortion while the loss ratio is kept under a given limit.* "Optimal" is in the sense that while the QoS is ensured, the overall average end-to-end delay, consequently the buffer requirement, is minimal. "Piecewisely" implies that instead of always using one fixed offset delay an *adaptive offset delay* is employed. Line *A(t)* in Figure 2.1 features this adaptability. Initially, the playback point could be better set with certain knowledge of the network conditions and dynamically adjusted thereafter. However, since knowing network conditions to any extent is expensive or even impossible in some situations, the adjustment mechanism should be intelligent enough to quickly converge a playback point to the optimal state even with an arbitrary initial set. This adjustment should not be based on the worst case assumptions on the network and traffic behavior, rather is computed with proper predictions in response to the actual network delay in the *recent past.* To recover the object as completely as possible, this adjustment should respond quickly to network degradation to avoid serious object loss and rather be conservative to network improvement to avoid excessive surging in playback.

2.1.3 Basic Synchronization Algorithm

The proposed synchronization algorithm uses minimal knowledge of the traffic characteristics. The algorithm assumes neither a global clock nor stationary network, does not need to register object arrival time and is immune to clock offset and/or frequency drift, while it ensures the QoS in terms of end-to-end delay, jitter and loss ratio. First, the basic algorithm is presented, then shown how it is applied to different aspects of synchronization.

The synchronization scheme requires the user to specify the maximum acceptable synchronization error, E_{max}, (the difference between the generation time intervals at sender and playback time intervals at receiver of any adjacent objects), the maximum acceptable jitter, J_{max}, (variance of the synchronization error), and the maximum acceptable object loss ratio caused by synchronization measures,[2] L_{max}. At the sender each object carries a time stamp, $t_{i,g}$, indicating its generation time. At the receiver a *PlayBack Clock* (PBC) and three event counters, namely *wait counter* C_w, *non-wait counter* C_{nw} and *discard counter* C_d, with associated thresholds T_w, T_{nw} and T_d respectively, are maintained.

The PBC is nothing more than a virtual clock at the receiver site which emulates the clock at the sender site. The motivation to have the PBC is that once the source clock can be reproduced at the sink, the synchronization problem may be readily solved.

At the receiver site the PBC is initiated according to the time stamp carried by the *first received* object, updated by the receiver clock, and adjusted based on the contents of the three counters (details will be discussed later). The vicinity of an object's arrival

[2] It is possible to detect objects lost in the network, and there are also some countermeasures against excessive loss like asking the sender to slow down.

time in reference to the PBC time is partitioned by the *wait boundary* (B_w) and *discard boundary* (B_d) into three regions: *wait* region, *non-wait* region and *discard* region (refer to Figure 2.2). These two boundaries are defined for each object. For the ith object carrying time stamp $t_{i,g}$ the two boundaries are

$$\begin{aligned} B_{i,w} &= t_{i,g} \\ B_{i,d} &= B_{i,w} + E_{max} = t_{i,g} + E_{max} \end{aligned} \tag{2.1.1}$$

The arrival time $t_{i,ar}$, in reference to the PBC, of the ith object may fall into one of the three regions with respect to its associated two boundaries as defined in Equation 2.1.1. The synchronization algorithm conforms to the following rules:

1. If the object with time stamp $t_{i,g}$ arrives before $B_{i,w}$ (within the wait region), then it will be played back at $B_{i,w}$ (waiting until the wait boundary).
2. If the object with time stamp $t_{i,g}$ arrives after $B_{i,w}$ but before $B_{i,d}$ (within the non-wait region), then it is played back immediately.
3. If the object with time stamp $t_{i,g}$ arrives after $B_{i,d}$ (within the discard region), then it is discarded.

The rules reveal that the wait boundary regarding an object turns out to be the scheduled playback time for that object.

Immediate questions that readers may ask are how to initiate the PBC and how to adjust it thereafter. The PBC is triggered by the first arrived object, say the kth object, and its initial time is set to $t_{k,g}$, the time stamp carried by this object. If there were no clock frequency drifting and the network were deterministic, the initialization by the first received object should be perfect. Unfortunately, the real situation does not allow either of the assumptions: network behavior is based on a time-varying stochastic process and

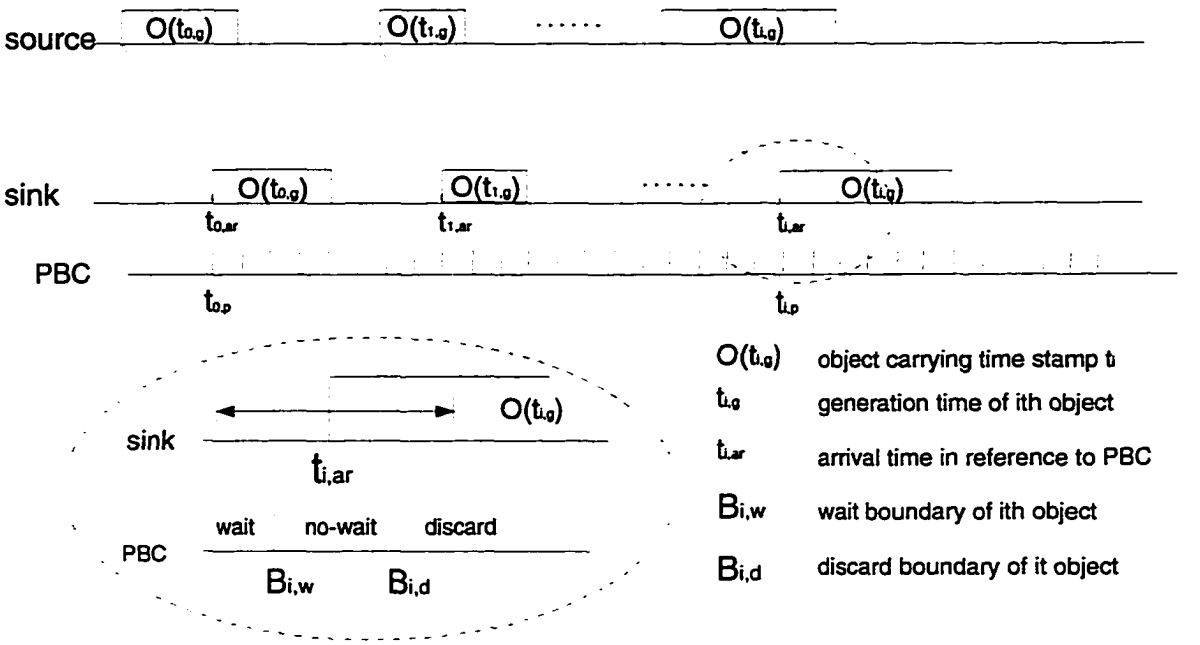


Figure 2.2 Temporal relationship among sender, receiver and PBC.

clocks are drifting from each other. Hence, the PBC has to be adjusted from time to time. Criteria of adjustment are in response to network conditions and ultimately based on the user specified QoS requirements. Here is the adaptive multimedia synchronization algorithm (B_w and B_d are defined by Equation (2.1.1), and all times are in reference to the PBC. Refer to Figure 2.3):

1. Upon first receiving an object set the initial PBC time equal to the time stamp carried by this first arrived object.
2. Upon receiving the *i*th object, compare its time stamp $t_{i,g}$ with its arrival time $t_{i,ar}$ (the current PBC time *PBC(t)*):

   if $t_{i,g} > t_{i,ar}$, increase the wait counter by one and do not playback the object until B_w;

else if $t_{i,g} \leq t_{i,ar} < t_{i,g}+E_{max}$, increase the non-wait counter by one and playback the object immediately;

otherwise (i.e., $t_{i,ar} \geq t_{i,g}+E_{max}$) increase the discard counter by one and discard the object.

3. Check the most recently increased counter:

   if $C_x < T_x$, $x \epsilon \{w, nw, d\}$, go to Step 2.

4. When the non-wait counter or the discard counter overflows:

   if the wait counter is *not* full, decrease the PBC: $PBC(t) = PBC(t) - \Delta$ and go to Step 5;

   otherwise go to Step 5.

   When the wait counter overflows:

   if the non-wait counter is *not* full, increase the PBC: $PBC(t) = PBC(t) + \Delta$ and go to Step 5;

   otherwise go to Step 5.

5. Reset all counters and go to Step 2.

In the second half of Step 4, the discard counter is not taken into consideration. A simple reason behind it is that larger number of samples always gives better statistical significance. In practice, the threshold T_{nw} of the non-wait counter is larger than the threshold T_d of discard counter. Be aware that when initiating the PBC, the first arrived object at the receiver dose not have to be the first generated object at the sender. This eliminates the need for a dedicated initialization phase. The receiver can jump in at any

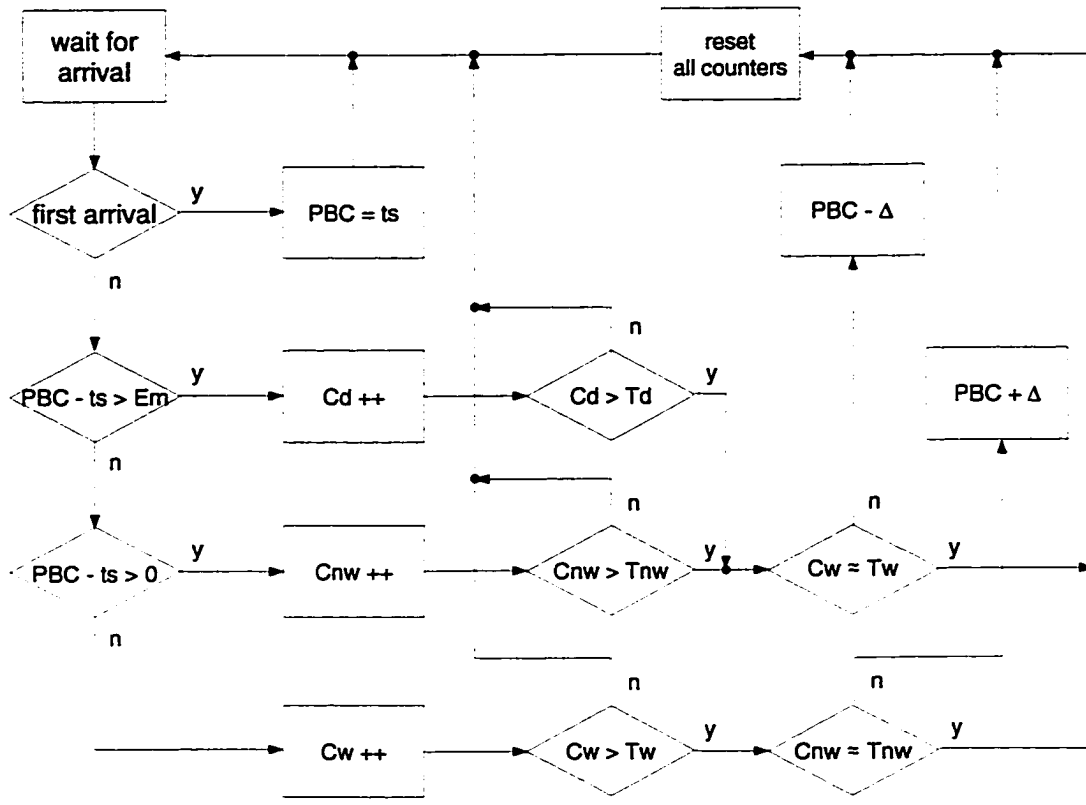


Figure 2.3 Flow diagram of PBC adjustment mechanism.

time after the sender having started transmission. Also, it is easy to restart whenever the synchronization is lost.

The thresholds of the three counters are critical to the performance of this synchronization algorithm. For example, the threshold of wait counter, T_w governs how sensitive the synchronization scheme is to the network improvement. Values of the thresholds, therefore, should be small enough to meet the needs of the user defined specifications while large enough to confine the computation overhead and achieve proper system stability.

The thresholds would be precisely determined when the underlying network delay distribution is known. Nonetheless, one can compute them with the worst case scenario. Figure 2.4 depicts a general network delay distribution and shows how the worst case

assumption has been adopted. In this figure B_w is the wait boundary and B_d is the discard boundary. Figure 2.4 (a) is a delay distribution, $f_a(x)$, with a long and thin tail. Figure 2.4 (b) shows the worst case scenario, epitomized by

$$f_b(x) = P_w\delta(x - B_w) + (1 - P_w)\delta(x - B_d) \tag{2.1.2}$$

where $P_w = \int_{D_{min}}^{B_w} f_a(x)dx$ and $\delta(x)$ is the unit impulse function. In Figure 2.4 (a) all the arrivals before B_w are buffered until B_w; all the arrivals between B_w and B_d are assumed to have arrived just before B_d. Note that the arrivals after B_d is neglected in Figure 2.4 (b), because the discarded portion is anyway bounded by the user specified L_{max}. This artificial bimodal distribution is the delay distribution after applying the synchronization algorithm with the worst case scenario. If the user specifies E_{max} and J_{max}, the variance of the delay is obtained from Equation (2.1.2)

$$\sigma^2 = J_{max} = P_w E_{max}^2 - P_w^2 E_{max}^2 \tag{2.1.3}$$

Let *counting window* be defined as $W = T_w + T_{nw} + T_d$. Due to the worst scenario assumption, $P_w = T_w/W$. The threshold of the wait counter is derived as

$$T_w = \left\lfloor \frac{1 + \sqrt{1 - 4J_{max}/E_{max}^2}}{2} W \right\rfloor \tag{2.1.4}$$

Once E_{max} is fixed, from Equation (2.1.4), J_{max} is upper bounded by $\sqrt{J_{max}} \leq E_{max}/2$.

When the worst case scenario is somewhat relaxed, the threshold for the non-wait counter can be set by assuming that all arrivals in the non-wait region are uniformly distributed as shown in Figure 2.4 (c). In this case the variance of delay is computed as

$$\sigma^2 = J_{max} = \frac{1}{3}P_w E_{max}^2 - \frac{1}{4}P_w^2 E_{max}^2 \tag{2.1.5}$$

then the T_{nw} is

$$T_w = \left\lfloor \frac{2}{3}\left(1 - \sqrt{1 - 9J_{\max}/E_{\max}^2}\right)W \right\rfloor \tag{2.1.6}$$

Determining the threshold of the discard counter, T_d is rather straight forward:

$$T_d = \lfloor L_{max} W \rfloor \tag{2.1.7}$$

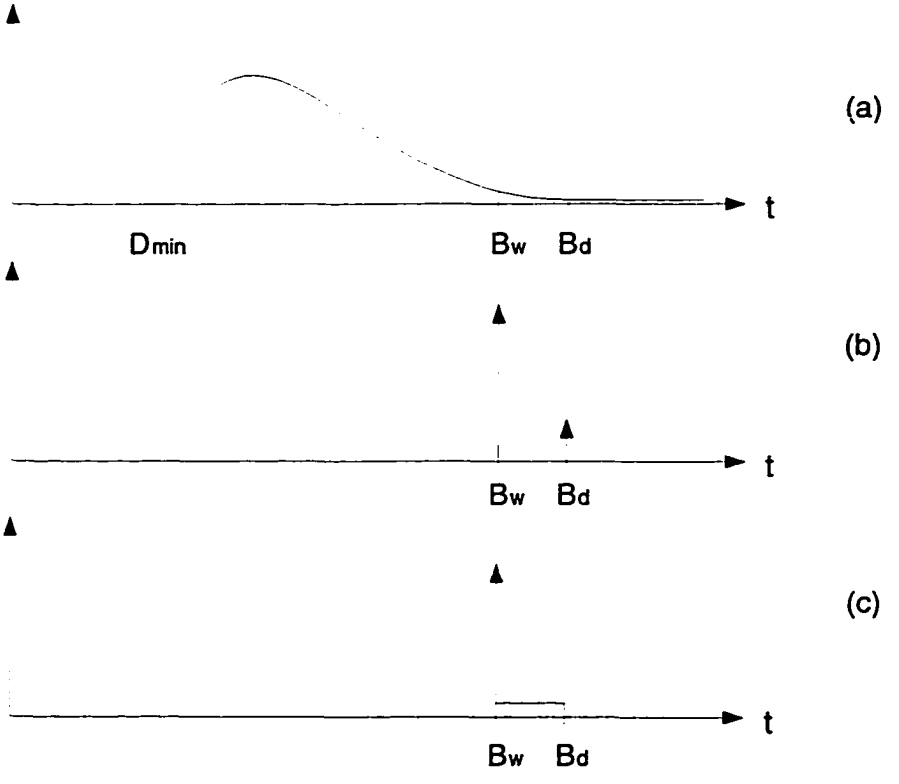


Figure 2.4 Derive T_{nw} based on worst case assumption. Where B_w is the wait boundary and B_d is the discard boundary. (a) a general network delay distribution; (b) the worst case assumption; (c) the relaxed worst case assumption.

So far the counting window size W is yet to be found, which as a matter of fact determines the confidence level of the algorithm. One reasonable approach to finding W is to approximate the probability that certain event occurs k times in the counting

window by

$$Prob\{k\} = \binom{W}{k} P^k (1-P)^{W-k} \tag{2.1.8}$$

Due to the nature of network delays, it generally is true that $T_w > T_{nw} > T_d$. Consequently, best confidence would be obtained when it is derived based on the most sensitive factor, T_d. The probability (i. e. the confidence) that the user's specifications are satisfied, $Prob\{\frac{k}{W} \leq L_{\max}\}$, is given by

$$Prob\left\{\frac{k}{W} \leq L_{\max}\right\} = Prob\{C_d \leq T_d\} = \sum_{C_d=0}^{T_d} \binom{W}{C_d} P_d^{C_d} (1-P_d)^{W-C_d} \tag{2.1.9}$$

where P_d is the probability of discard. It generally is true that $WP_d(1-P_d) \gg 1$ because $L_{\max}$ is always very small. Therefore the right-hand side of Equation 2.1.9 can be approximated by a *Gaussian* density function due to the *Large Number Theorem*, i. e.:

$$Prob(C_d \leq T_d) \approx \frac{1}{2} + erf\left[(L_{\max} - P_d)\sqrt{\frac{W}{P_d(1-P_d)}}\right] \tag{2.1.10}$$

Thus the value of W, and consequently T_{nw}, can be easily found for a given confidence. It is worthwhile to mention that P_d is just an expected value which should be less than $L_{\max}$. However, it may not be satisfied if the discard counter fulls well before the other two counters building up. In that case the PBC adjustment is performed.

One may already realized that the synchronization algorithm is inherently immune to clock frequency drift since its effect can be treated as one of the contribution factors to network delay jitter. To see this, assume network delay has no jitter. If the clock at sender were synchronized with the one at receiver, the receiver should always receive an object carrying time stamp $t_{i,g}$ exactly at $B_{i,w}$, or $t_{i,g} = t_{i,ar}$. When the clock at the sender is faster than the one at the receiver, the receiver at one time receiving an object

carrying time stamp $t_{k,g}$ exactly at $B_{k,w}$ should find the succeeding objects always fall into the wait region, i.e., $t_{i,g} > t_{i,ar}, \ \forall\, i > k$. This would soon cause the wait counter to go beyond its threshold while the non-wait counter and discard counter remain empty. Once this happens the PBC would be increased to track the clock at the sender. *Vice versa*, it would be handled conversely if the clock at the sender site were slower than the one at the receiver site. However, this time the PBC would be adjusted more frequently than the previous case because of the relative lower thresholds of the non-wait counter and the discard counter. Moreover, objects received out of sequence will be put in order automatically as they go through the synchronization process.

The conduction of adjusting amount to the PBC is based on the following argument. If the adjusted PBC had controlled the synchronization during the last *counting period* (the time between counter reset), it must be satisfied

$$L_{\max} \geq \frac{C_d + C_{nw} \cdot \Delta^+ / E_{\max}}{C} \text{ and } P_w \leq \frac{C_w - C_{nw} \cdot \Delta^+ / E_{\max}}{C} \tag{2.1.11}$$

if the PBC is advanced, and

$$L_{\max} \geq \frac{C_d - C_{nw} \cdot \Delta^- / E_{\max}}{C} \text{ and } P_w \leq \frac{C_w + C_{nw} \cdot \Delta^- / E_{\max}}{C} \tag{2.1.12}$$

if the PBC is delayed. Therefore, when the wait counter overflows, the PBC *increase* amount is computed as

$$\Delta^+ = \min\{(L_{\max}C - C_d), (C_w - CP_w)\} \cdot E_{\max}/C_{nw} \tag{2.1.13}$$

when the non-wait counter or the discard counter overflows, the *decrease* amount is computed as

$$\Delta^- = \max\{(C_d - L_{\max}C), (CP_w - C_w)\} \cdot E_{\max}/C_{nw} \tag{2.1.14}$$

where $C = C_w + C_{nw} + C_d$ is the total counting of the last counting period.

An intuitive example helps better understanding of the proposed adaptive synchronization algorithm. For a certain period of time, assume the underlying network delay distribution is represented by the curve in Figure 2.5. Figure 2.5 (a) is the well adjusted situation, in which all the three counters are always full almost simultaneously. Figure 2.5 (b) shows the under delayed situation, in which more objects than expected fall behind the wait boundary and are discarded just because the PBC is too fast. Consequently, the discard counter always becomes full before the other counters.[3] Then the PBC should be decreased. Figure 2.5 (c) shows the over delayed situation, in which most objects have to wait unnecessarily long before being played back just because the PBC is too slow. Consequently, the wait counter always becomes full before others. Then the PBC should be advanced.

2.1.4 Intramedium Synchronization

Applying the adaptive synchronization algorithm to intramedium situation is straightforward. Figure 2.6 depicts the intramedium synchronization configuration. The time stamp of each object of a single medium is first compared with the PBC. The difference of these two times is fed into the *intramedium synchronizer*. The intramedium synchronizer, composed of the three counters (C_w, C_{nw} and C_d) and necessary buffers, controls the playback time of each object, discards the ones coming later than their discard boundaries and generates feedback signals to adjust the PBC. If for a medium intramedium synchronization is needed only for sometime, it is easy to bypass the intramedium synchronizer and let the objects go as they arrive.

[3] For another delay distribution pattern, it might be true that the non-wait counter always fills first.

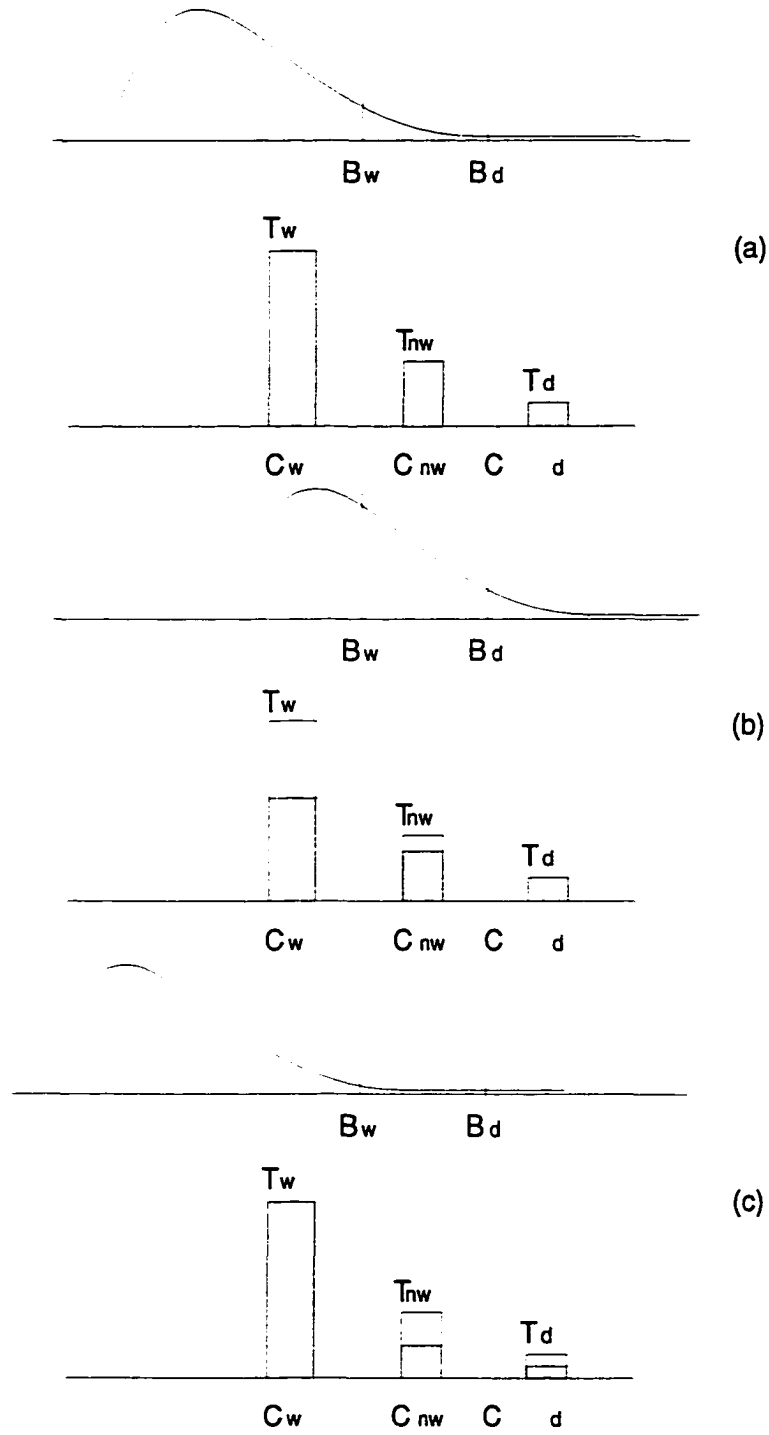


Figure 2.5 A example intuitively demonstrating the adaptive synchronization algorithm.

2.1.5 Intermedia Synchronization

The intramedium synchronization structure can be easily extended to intermedia

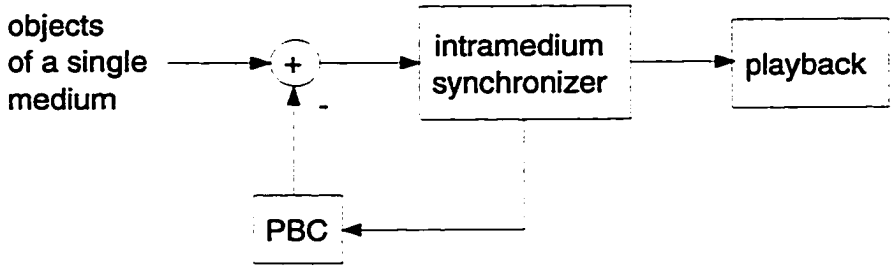


Figure 2.6 Intramedium synchronization.

situation. Assume there are N media to be synchronized. When the N media try to reach their own intramedium synchronization individually, N PBC's and N intramedium synchronizers are needed. The decision of adjusting each PBC is given by its associated intramedium synchronizer. The adjusting decisions for different PBC's are most likely different. When intermedia synchronization is required, a *Group PBC* is engaged. The reading of the Group PBC is equal to that of the slowest one among the N PBC's. This Group PBC dominates the playback of all media in the intermedia synchronization group. Meanwhile, each medium in the intermedia synchronization group does its own intramedium synchronization as if it were not in the group in order to justify its PBC, but now the discard decision is made in reference to the Group PBC.

Figure 2.7 shows the configuration of intramedium and intermedia synchronizations. This structure allows the application to flexibly pick up certain media for intermedia synchronization and conveniently switch a selected medium back and forth between intramedium synchronization and intermedia synchronization modes. These functions are useful in many scenarios. Take teleconference as an example. Suppose there are four media, namely audio, video, graphic, and text. Normally it is not necessary to synchronize all these media, rather intermedia synchronization between audio and video are commonly required. So the application simply picks up audio and video for intermedia synchronization and lets other media go as they arrive, (graphic and text may not even

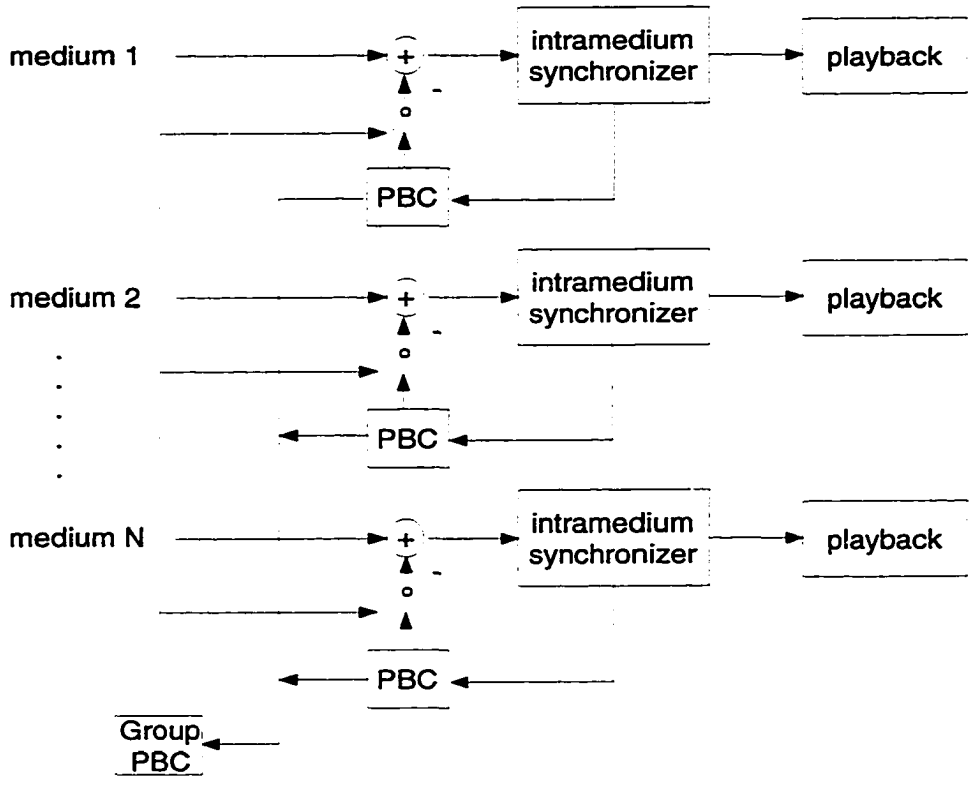


Figure 2.7 Intramedium and intermedia synchronizations.

need intramedium synchronization). However, when a conferee presents viewgraphs to the audience, the graphic has to be intermedialy synchronized with audio, maybe with video too, so that the audience can see the pointer moving synchronously with the speech. At this time, the application can switch the graphic from no synchronization mode to intermedia synchronization mode.

2.2 Implementation and Performance Studies

Figure 2.8 shows the experimental network testbed, which consists of an 100 Mbps FDDI as a backbone, 3 servers (Sun workstation 4/330) connected to FDDI, two 10 Mbps Ethernet segments, each connecting several Sun IPC's. SunOS 4.1.4 is the operating system. Each of the hosts taking part in the multimedia experiment resides on a different Ethernet, as shown in the dotted squares.

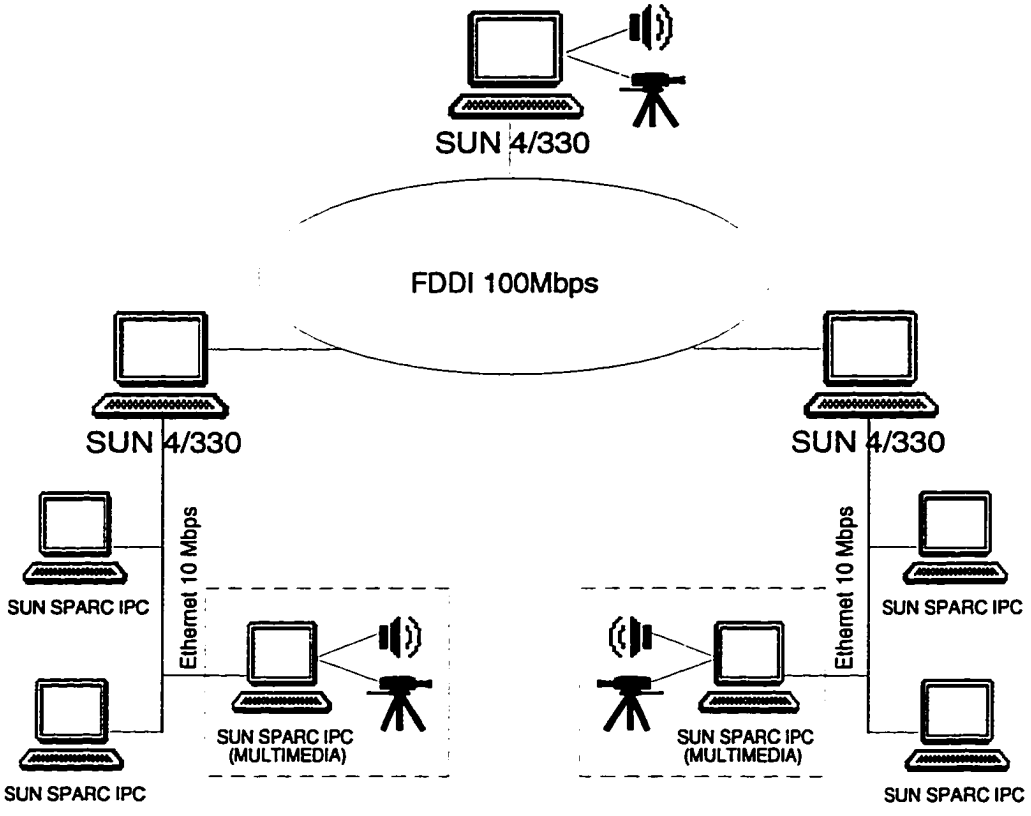


Figure 2.8 Testbed configuration

Video compression is done by JPEG (XVideo board from Parallax Graphics), which provides up to 360 by 480 pixels true color image at about 10 frames per second rate. Audio acquisition/playback is performed through the Sun workstations' audio hardware, which provides 8-bit μ-law companded audio at a rate of 8K samples per second.

2.2.1 Audio and Video Traffic

In the following experimental results regarding the performance of the proposed synchronization algorithm on the network testbed are presented. Measures were done for delays of audio and video objects for 1000 seconds between the two multimedia hosts while the microphone at the sender keeps on picking ambient audio signals in the lab and the video camera keeps on capturing a still scene and occasionally passing-by

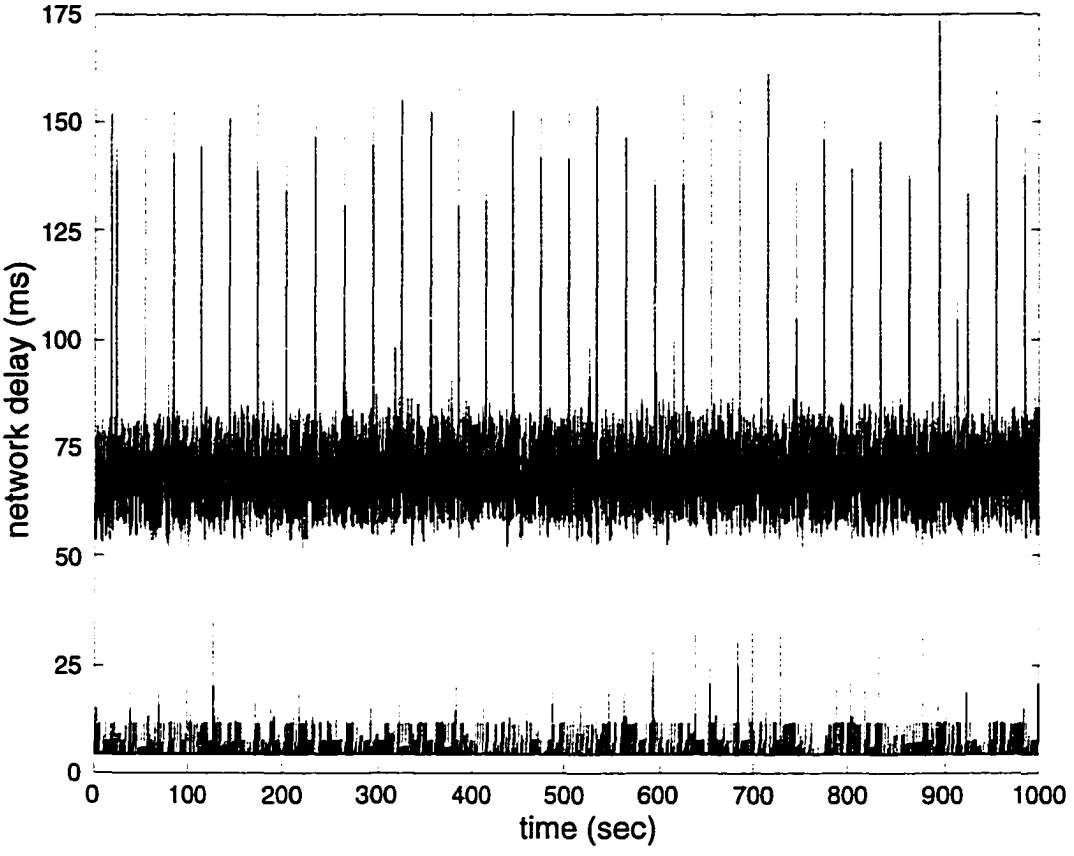


Figure 2.9 Audio and video network delays. The delays scattered into two groups: the lower one is the audio delay and the upper one is the video delay.

people. Under this condition, delays are mainly caused by processing overhead. Figure 2.9 overlaps the delays experienced by audio and video objects, from the time when an object is created to the time when the object arrives at the receiver before synchronization (hereafter referred as *network delay*). Figure 2.10 shows the corresponding network delay distribution for audio. The average delay is $\eta^a = 6.9\ ms$ and standard deviation (square root of variance) is $\sigma^a = 3.6\ ms$. Figure 2.11 shows the corresponding delay distribution for video. The average delay is $\eta^v = 69\ ms$ and standard deviation is $\sigma^v = 10.7\ ms$. It is observed that the average delay of video is 10 times longer than that of audio. It is attributed to the fact that the size of video objects varies from 8 kBytes to 12 kBytes,

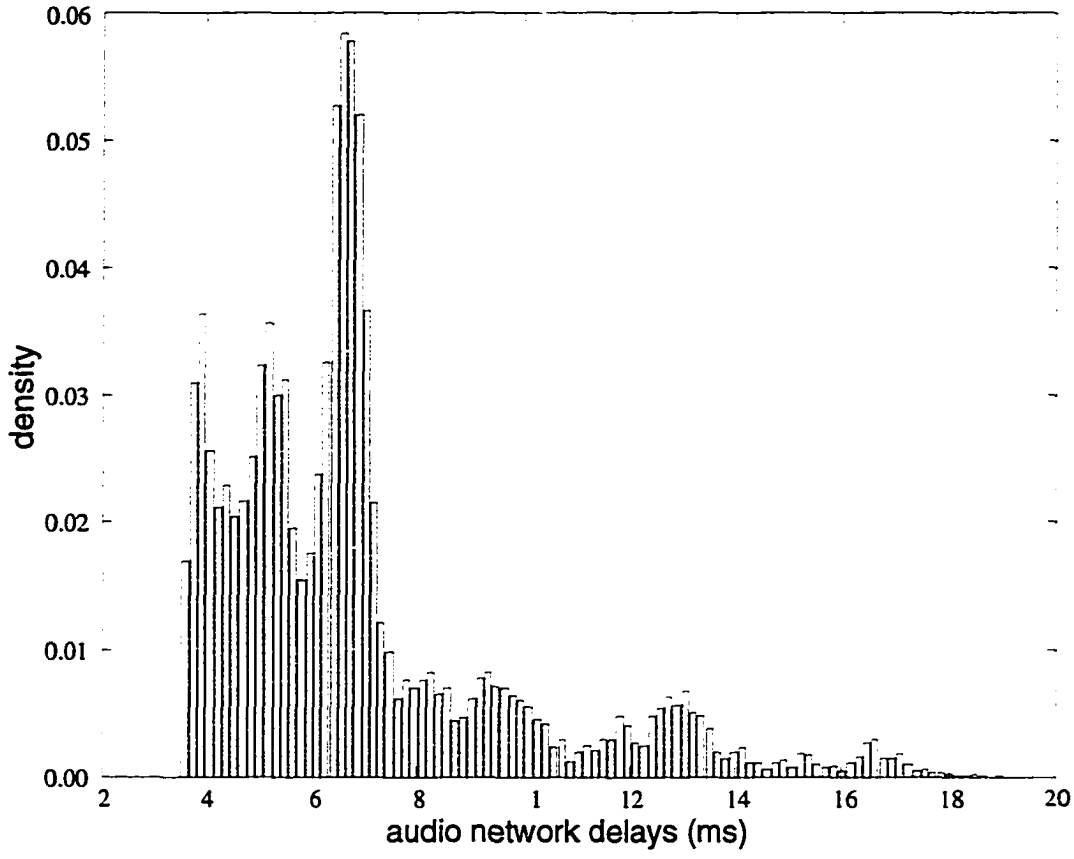


Figure 2.10 Audio network delay distribution.

about 10 times of the size of audio objects (0.8 kBytes). In the test environment the transmission delay is the major part of the network delay.

Situations being examined are the intramedium synchronization for audio with and without the background traffic, and the intermedia synchronization.

2.2.2 Intramedium Synchronization for Audio

As for the user given QoS requirements, it is assumed that L_{max} = 1%, E_{max} = 2 ms, and J_{max} = (0.8 ms)^2. It is chosen that T_w = 800, then T_d = 8 from Equation (2.1.7), and T_{nw} = 200 from Equation (2.1.6). Figure 2.12 shows the wait boundary of audio objects given by the intramedium synchronizer, which is also the scheduled playback

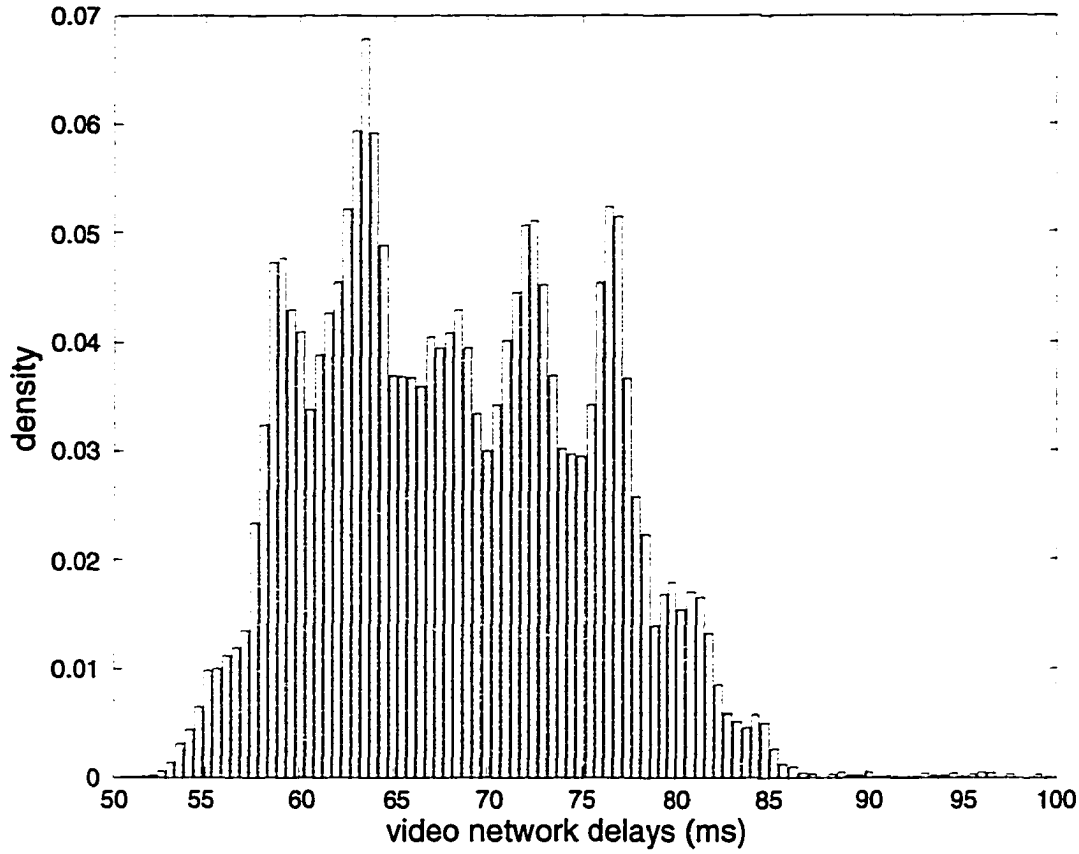


Figure 2.11 Video network delay distribution.

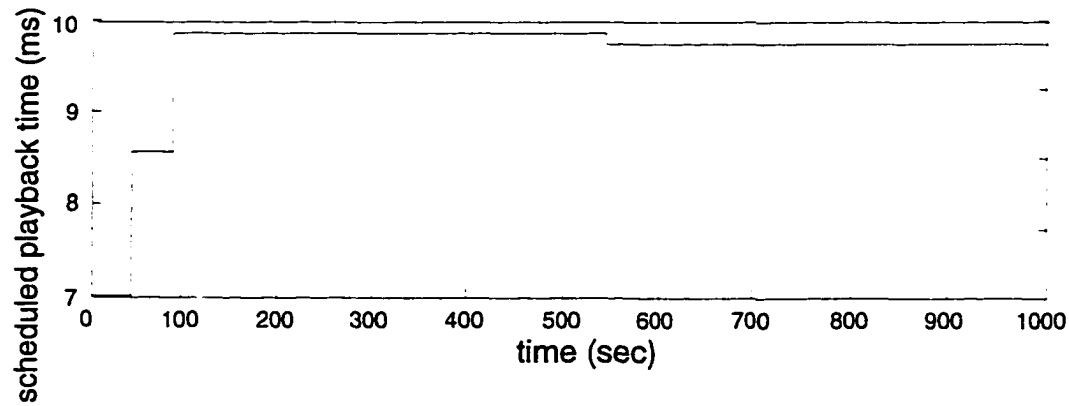


Figure 2.12 Wait boundary of audio objects.

time for audio objects. The intramedium synchronizer works well. Figure 2.13 shows the end-to-end delay (time difference between the generation time and playback time after synchronization of an object, $D_i^a = t_{i,p}^a - t_{i,g}^a$) of audio objects. The small spikes

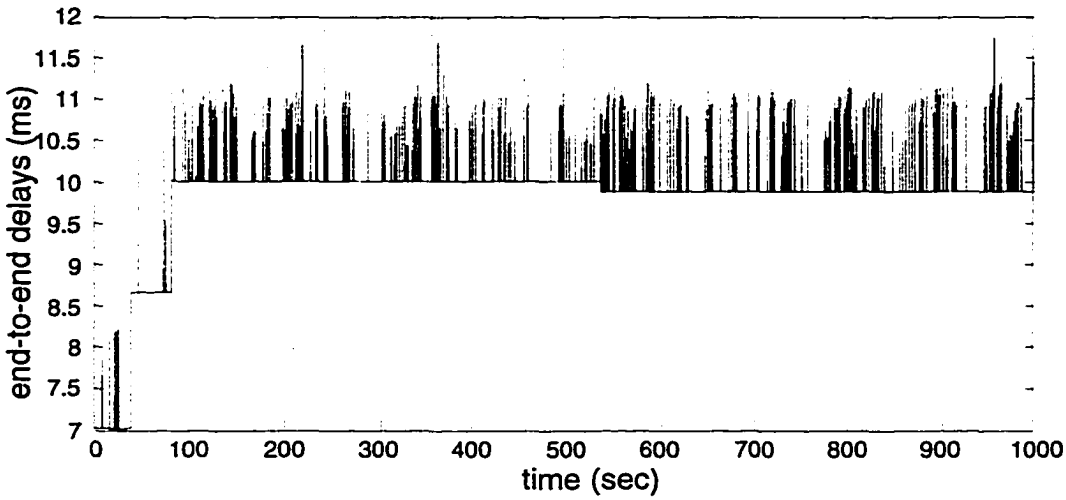


Figure 2.13 The end-to-end delay of audio objects.

above the lines are representing the actual playback times of those objects that arrive in the non-wait region. Therefore, none of their magnitudes is larger than the E_{max}, 2 milliseconds. Figure 2.14 plots the end-to-end delay distribution for Figure 2.13. The average end-to-end delay is $\eta^a_{intra} = 10\ ms$ and its variance is $(\sigma^a_{intra})^2 = (0.75\ ms)^2$. By comparing Figure 2.10 and Figure 2.14, it is observed that the variance has been significantly decreased from $(3.6\ ms)^2$ to $(0.75\ ms)^2$, while the average delay has been increased from 6.9 ms to 10 ms, obviously because of controlled synchronization delay. Since the intramedium synchronization error ϵ_{intra} is defined as the difference between the playback time ($t_{i,p}$) interval and the generation time ($t_{i,g}$) interval, i.e.,

$$\epsilon_{intra} = (t_{i,p} - t_{i-1,p}) - (t_{i,g} - t_{i-1,g}) = T_{i,p} - T_{i,g} = D_i - D_{i-1} \qquad (2.2.1)$$

where $T_{i,p}$ is the playback interval and $T_{i,g}$ is the generation interval. And its variance

$$E\left[(\epsilon^a_{intra})^2\right] = 2(\sigma^a_{intra})^2 = 2(0.75\ ms)^2 \qquad (2.2.2)$$

It is noted that without synchronization the square mean error is 2(3.6 ms)^2, while with synchronization it is significantly reduced to 2(0.75 ms)^2.

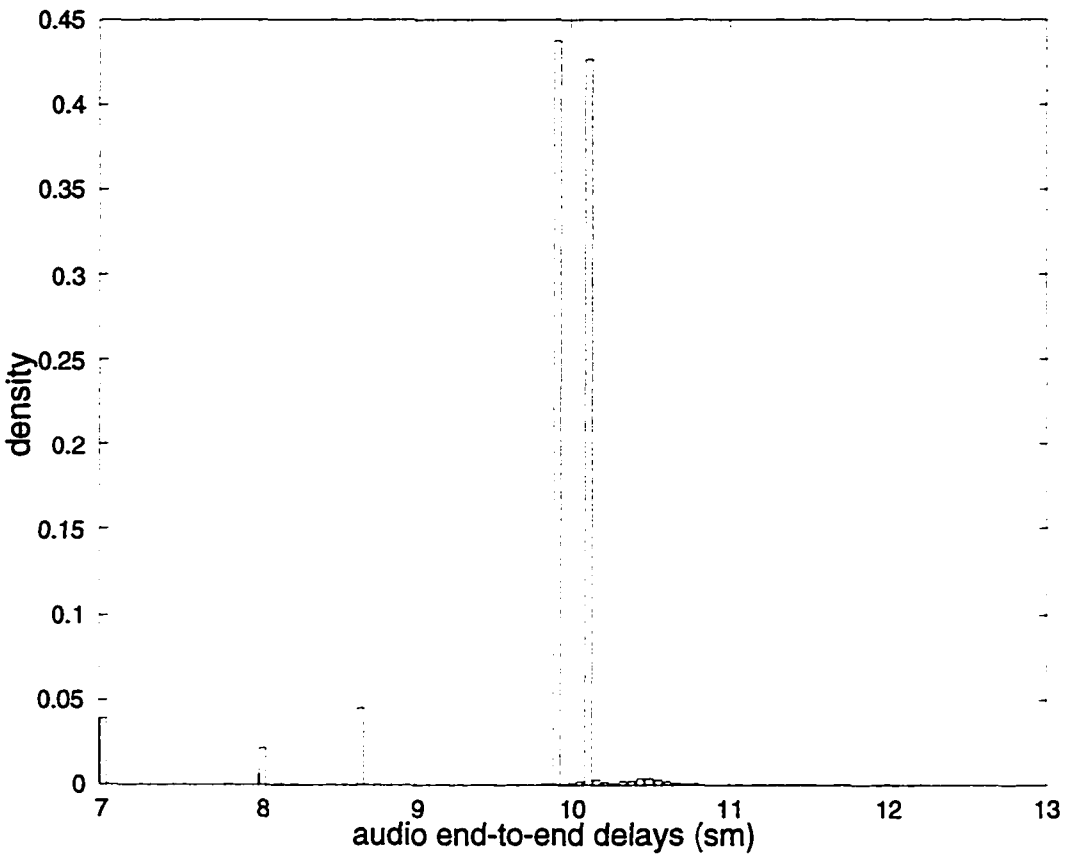


Figure 2.14 Audio end-to-end delay density distribution.

Now, let us add some background traffic in order to see how well the synchronization algorithm works when the network condition changes. Background traffic was added by having another set of hosts keep sending a text file to the communicating multimedia hosts for 5 minutes. Figure 2.15 shows the network delay experienced by the audio objects under this condition with wait/no-wait boundaries indicated. This time, network delays caused by queuing at different layers along the transmission path became significant. During the period with the added background traffic both the minimal delay and delay jitter are increased. All the parameters retain at the same values as in the static case (i.e., L_{max} = 1%, E_{max} = 2 ms, J_{max} = (0.8 ms)^2, T_w = 800, T_d = 8, and T_{nw} = 200). The intramedium synchronizer nicely tracks these changes as expected: when the network

delay increases the synchronizer follows up the jump very quickly (catches up with one jump), when the network condition improves the synchronizer responds in a conservative manner (reduces in two steps). Figure 2.16 shows the delays after the synchronization (i.e., the end-to-end delay). As evidenced by comparing Figure 2.15 and Figure 2.16, the variance of the delay is significantly reduced.

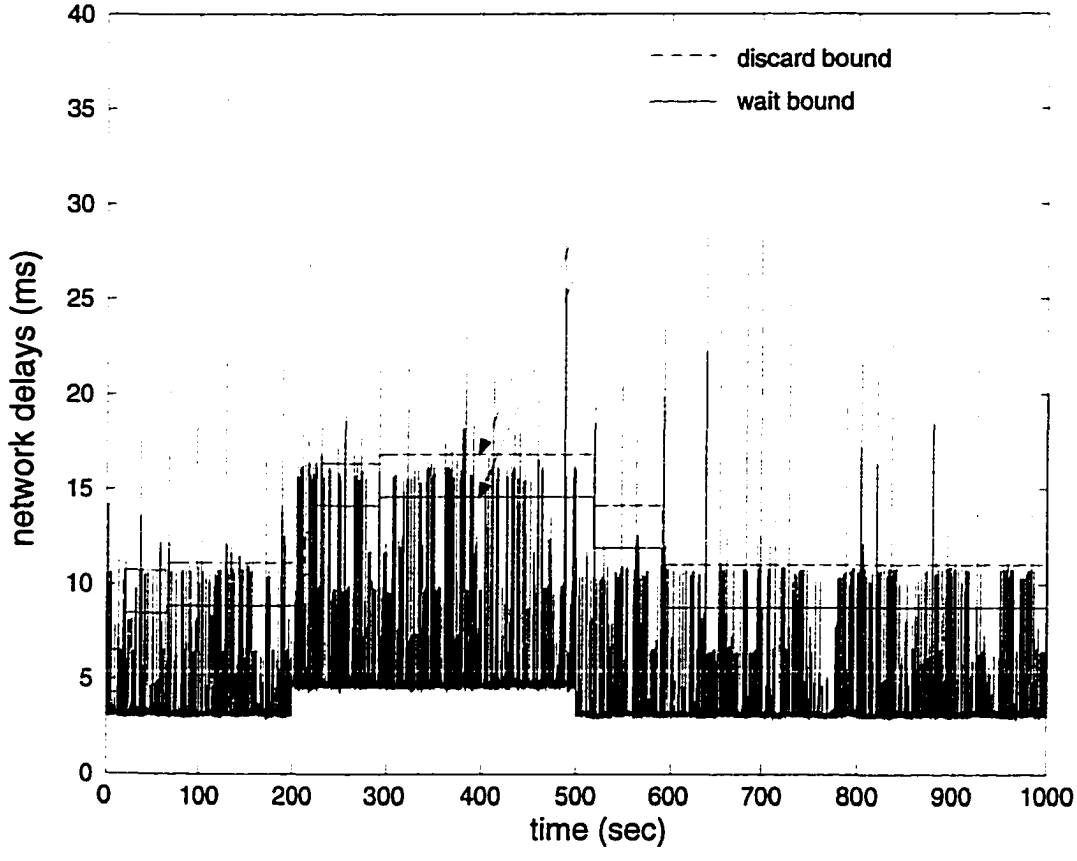


Figure 2.15 The delay behavior of audio objects when the network condition changes. The solid line is the wait boundary and the doted line is the discard boundary.

2.2.3 Intermedia Synchronization between Audio and Video

Now let us see how well the intermedia synchronizer works. Audio and video objects are not generated exactly at the same time instant. The intermedia temporal relationship with respect to a video object and the adjacent audio object immediately ahead of it in

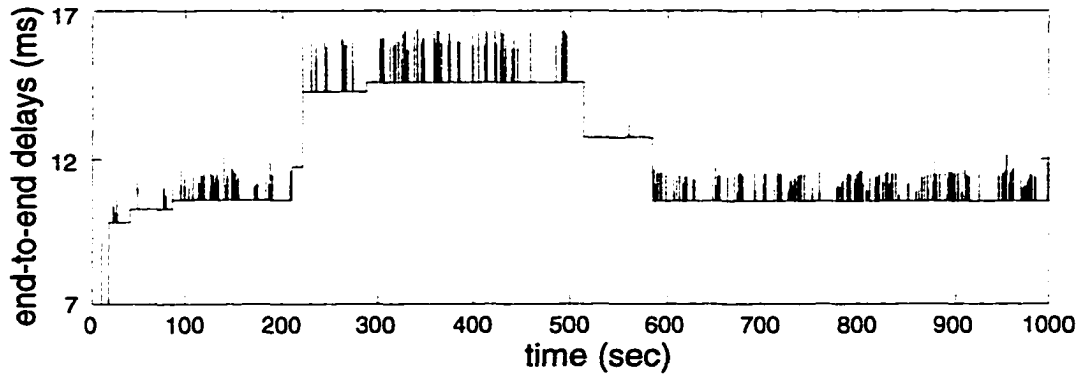


Figure 2.16 The end-to-end delay of audio objects when the network condition changes.

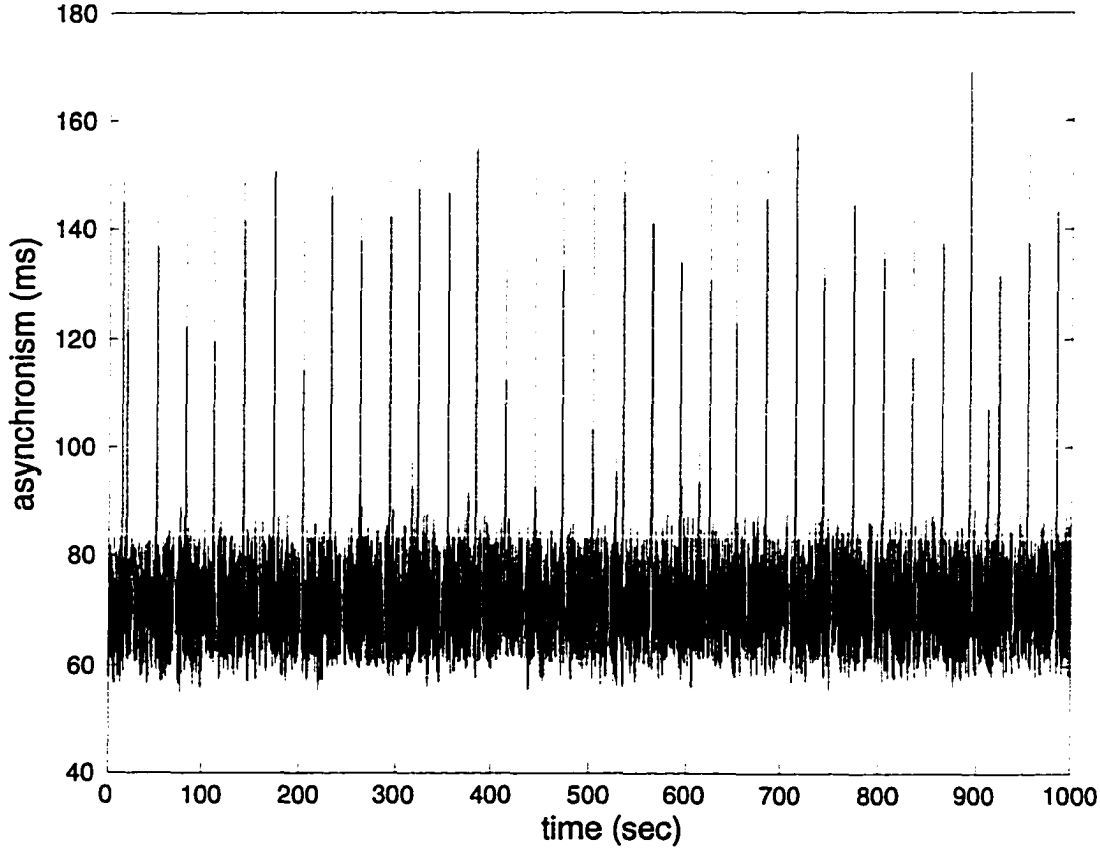


Figure 2.17 Asynchronousness between audio and video objects at arrival.

time were investigated. Denote the generation times of this pair of objects between which a temporal relationship is to be kept as $t^a_{m,g}$ and $t^v_{n,g}$, and the arrival times of them as

$t^a_{m,ar}$ and $t^v_{n,ar}$. Then the asynchronousness introduced by the network is

$$t_{i,asynch} = \left(t^v_{n,av} - t^a_{m,ar}\right) - \left(t^v_{n,g} - t^a_{m,g}\right) \tag{2.2.3}$$

Thus the intermedia synchronization error is defined as

$$\begin{aligned} \epsilon_{inter} &= \left(t^v_{n,p} - t^a_{m,p}\right) - \left(t^v_{n,q} - t^a_{m,q}\right) \\ \epsilon_{inter} &= \left(t^v_{n,p} - t^a_{m,p}\right) - \left(t^v_{n,g} - t^a_{m,g}\right) \end{aligned} \tag{2.2.4}$$

Figure 2.17 shows the asynchronousness, as defined in Equation (2.2.3), between the audio and video objects at arrival. Figure 2.18 shows the intermedia synchronization error, as defined in Equation (2.2.4), between audio and video objects when they have passed through their individual intramedium synchronizers. The magnitude and variance of the difference in arrival times have been reduced because the individual intramedium synchronizers discarded objects with abnormally long delay and smoothed each stream. Once the intermedia synchronizer is engaged, the performance becomes remarkable, as noticed from Figure 2.19. All synchronization errors are less than 10 ms and the variance is within $(2 \text{ ms})^2$.

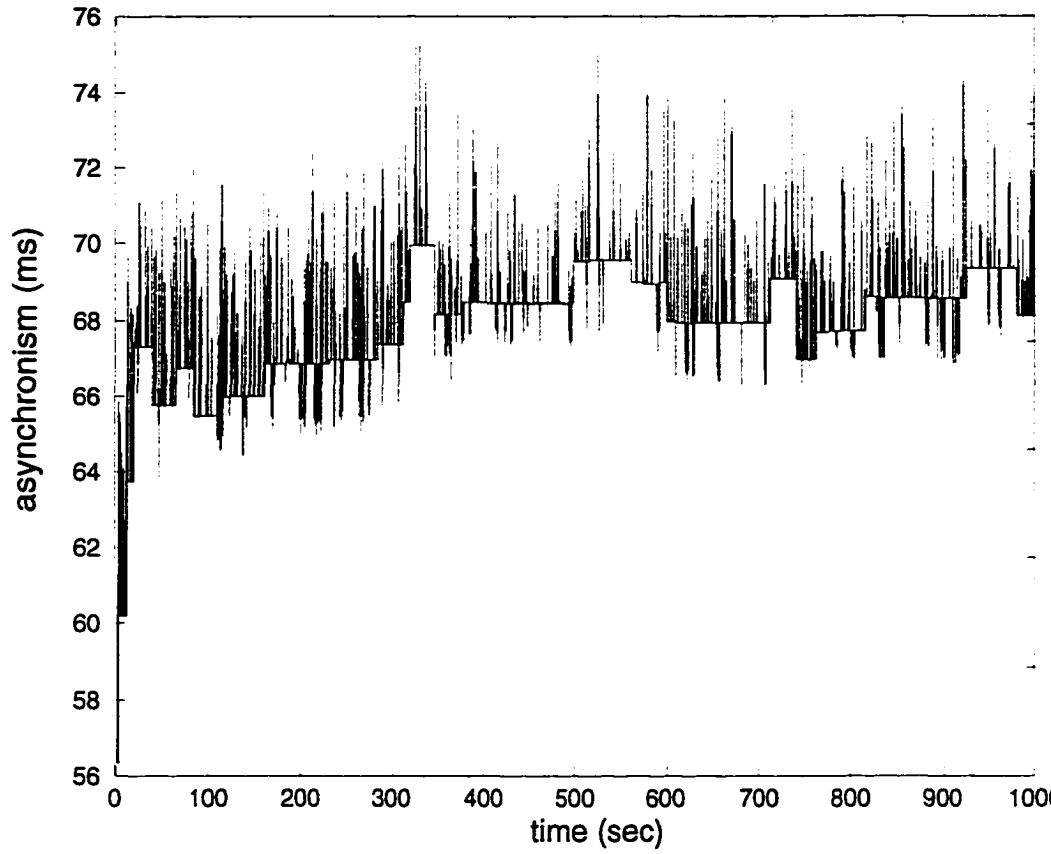


Figure 2.18 Asynchronousness between audio and video objects when only the intramedium synchronizers are engaged.

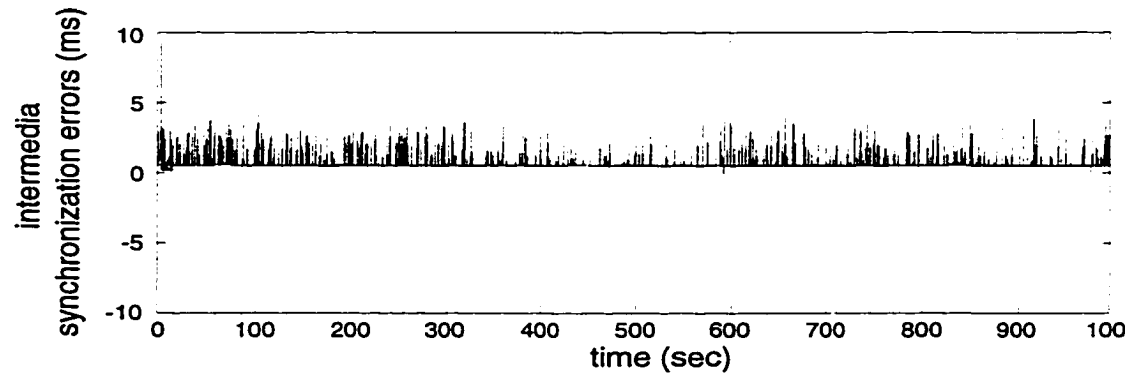


Figure 2.19 Intermedia synchronization errors when the intermedia synchronizer is engaged.

3. Multipoint Support Techniques

The best solution for building networks with large scale multicasting capability is still uncertain. Current IP multicasting (using DVMRP or MOSPF) has poor scaling properties. More scalable solutions, such as Protocol Independent Multicast (PIM) [25–27] and Core Based Tree (CBT) [28, 29] have better yet still limited scaling, but no general agreement has been reached at the Internet Engineering Task Force (IETF). Adding in the mobility issue increases the dynamics of the routing problem and causes some solutions to become impractical, especially with asymmetric links that may offer a slow (or no) return path.

While the IETF's solution to IP mobility support has recently been standardized as RFC [30], there are a lot of efforts going on to improve the inefficiencies in the basic version. As the emerge of the next generation of Internet protocols, revolutionary changes in the near future is not impossible.

Current engineering solution to multicast support is the Internet Multicast Backbone (MBone) [31]. The MBone is an interconnected set of subnetworks and routers that support the delivery of IP multicast traffic. The MBone has grown from 40 subnetworks in four different countries in 1992, to more than 3400 subnetworks in over 25 countries by March 1997. With new multicast applications and multicast-based services appearing, it seems likely that the use of multicast technology in the Internet will keep growing at an ever-increasing rate.

The MBone is a virtual network that is layered on top of sections of the physical Internet. It is composed of islands with multicast routing capability connected to other islands by virtual point-to-point links called *tunnels*. The tunnels allow multicast traffic

to pass through the non-multicast-capable parts of the Internet. Tunneled IP multicast packets are encapsulated as IP-in-IP [32, 33] so they look like normal unicast packets to intervening routers along the tunnel. The encapsulation is added on entry to a tunnel and stripped off on exit from a tunnel.

Since the MBone and the Internet have different topologies, multicast routers execute a separate routing protocol to decide how to forward multicast packets. The majority of the MBone routers currently use the DVMRP, although some portions of it execute either MOSPF or recently the PIM routing protocols.

3.1 Internet Group Management Protocol

The Internet Group Management Protocol (IGMP) [12] runs between hosts and their immediate neighboring multicast routers. The mechanisms of the protocol allow a host to inform its local multicast router that it wishes to receive transmissions addressed to a specific multicast group. Also, routers periodically query the LAN to determine if any group members are still active. If there is more than one IP multicast router on the LAN, one of the routers is elected as *querier* (also called *designated router (DR)*) and assumes the responsibility of querying the LAN for the presence of any group members (see Figure 3.1).

Based on the group membership information learned from the IGMP, a router is able to determine which (if any) multicast traffic needs to be forwarded to each of its *leaf* subnetworks (i.e., a subnetwork with group members but no multicast routers connecting to additional group members further downstream). Multicast routers use this information, in conjunction with a multicast routing protocol, to support IP multicasting across the Internet.

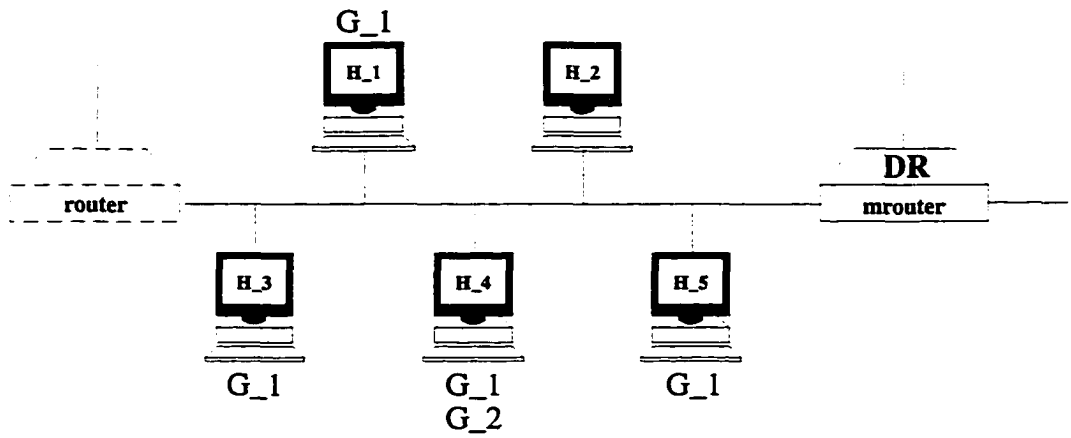


Figure 3.1 IGMP runs on a LAN with multiple routers.

There are so far 3 versions of the IGMP. In Version 1, the *Query* messages query all groups, and the leaf hosts report the groups they are actively in with *Report* messages. Version 2 allows group-specific *Query* and quicker group membership termination. Version 3 supports for (source, group) pair *Reports*, and the quick leave mechanism first introduced in Version 2 has been enhanced to support (source, group) pair leave.

3.2 Multipoint Routing Techniques

To provide an internetwork multipoint delivery service, it is necessary to define multicast routing protocols. A multicast routing protocol is responsible for the construction of multicast delivery trees across internetworks and enabling multicast packet forwarding. The following explores a number of different multipoint distribution techniques that may potentially be employed by multicast routing protocols [34].

3.2.1 Simple-Minded Techniques

Flooding: When a router receives a packet that is addressed to a multicast group, the router employs a protocol mechanism to determine whether or not it has seen this particular packet before. If it is the first reception of the packet, the packet is forwarded

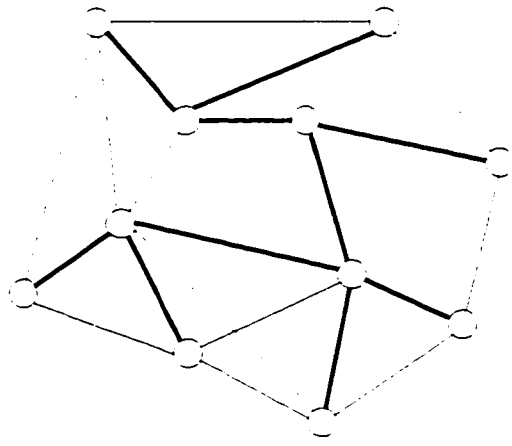


Figure 3.2 Spanning tree in a network. Bold lines represent a tree spanning all nodes.

on all interfaces except the one on which it arrived guaranteeing that the multicast packet reaches all routers in the internetwork. If the router has seen the packet before, then the packet is discarded.

Spanning Trees: A subset of the internetwork topology is selected to form a spanning tree, which defines a structure in which only one active path connects any two routers of the internetwork. Figure 3.2 shows an example. Once the spanning tree has been built, a multicast router simply forwards each multicast packet to all interfaces that are part of the spanning tree except the one on which the packet originally arrived. Test for replicated packets is not necessary.

These techniques are primitive due to the fact that they tend to waste bandwidth or require a large amount of computational resources within the multicast routers involved. They may, if ever, work for small network with few senders, receivers, and routers, but do not scale at all.

3.2.2 Source-Based Tree Techniques

An immediate step towards improving the spanning trees would be to build a spanning

tree for each potential source. These spanning trees would result in source-based delivery trees rooted at the source. There are a number of algorithms.

Reverse Path Broadcasting (RPB): For each source, if a packet arrives on a link that the local router believes to be on the shortest path back toward the packet's source, then the router forwards the packet on all interfaces except the incoming interface. If the packet does not arrive on the interface that is on the shortest path back toward the source, then the packet is discarded. One of the major limitations of the RPB algorithm is that it does not take into account multicast group membership when building the delivery tree for a source. As a result, datagrams may be unnecessarily forwarded onto subnetworks that have no members in a destination group (see Figure 3.3).

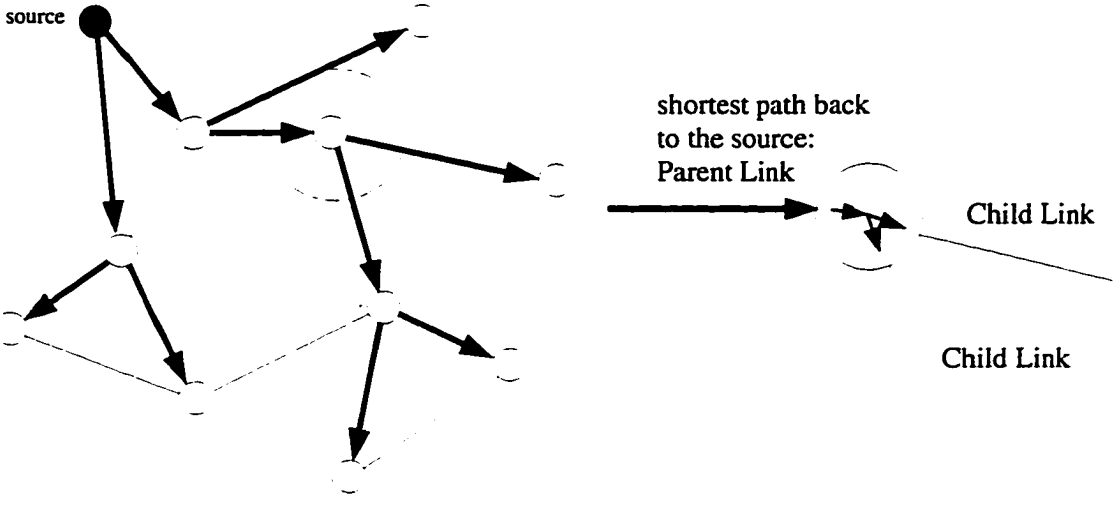


Figure 3.3 Reverse Path Broadcasting.

Truncated Reverse Path Broadcasting (TRPB): In order to overcome the limitations of RPB, with information provided by IGMP, multicast routers determine the group memberships on each leaf subnetwork and avoid forwarding datagrams onto a leaf subnetwork if it does not contain at least one member of a given destination group. Thus,

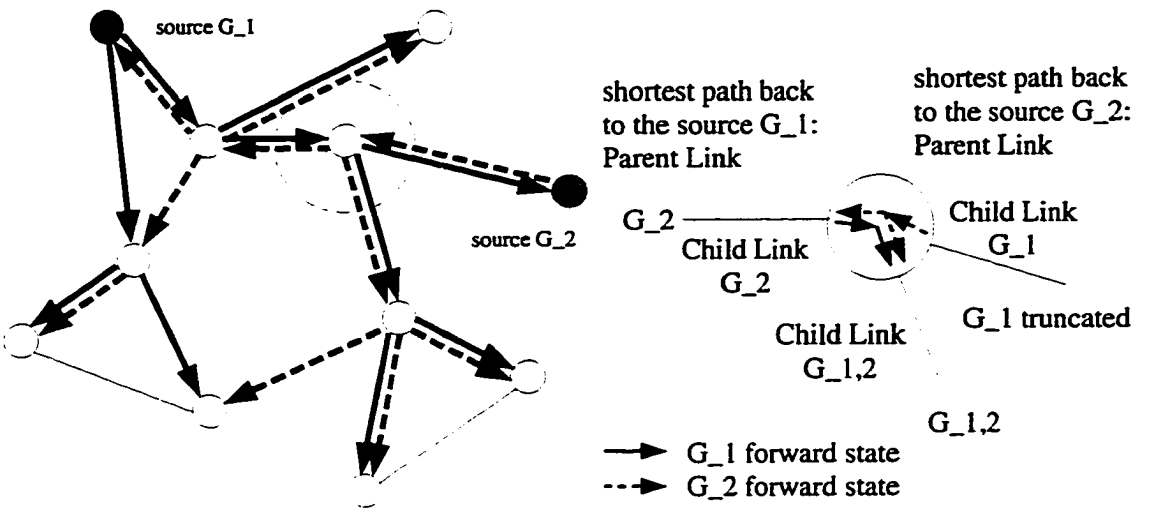


Figure 3.4 Truncated Reverse Path Broadcasting.

the delivery tree is *truncated* by the router if a leaf subnetwork has no group members. TRPB eliminates unnecessary traffic on leaf subnetworks but it does not consider group memberships when building the branches of the delivery tree (see Figure 3.4).

Reverse Path Multicasting (RPM): A further step to better efficiency is to build up a delivery tree that spans only subnetworks with group members, and routers along the shortest path to those subnetworks. RPM [35] allows the source-based *shortest-path tree* to be pruned so that datagrams are only forwarded along branches that lead to active members of the destination group. Despite the improvements offered by the RPM algorithm, there are still several scaling issues. The so used *flood and prune* paradigm is very powerful, but it wastes bandwidth. Also, every router participating in the RPM algorithm must have either a forwarding table entry for a (source, group) pair, or prune state information for that (source, group) pair for each and every (source, group) pair (see Figure 3.5).

It is clearly wasteful, especially as the number of active sources and groups increase,

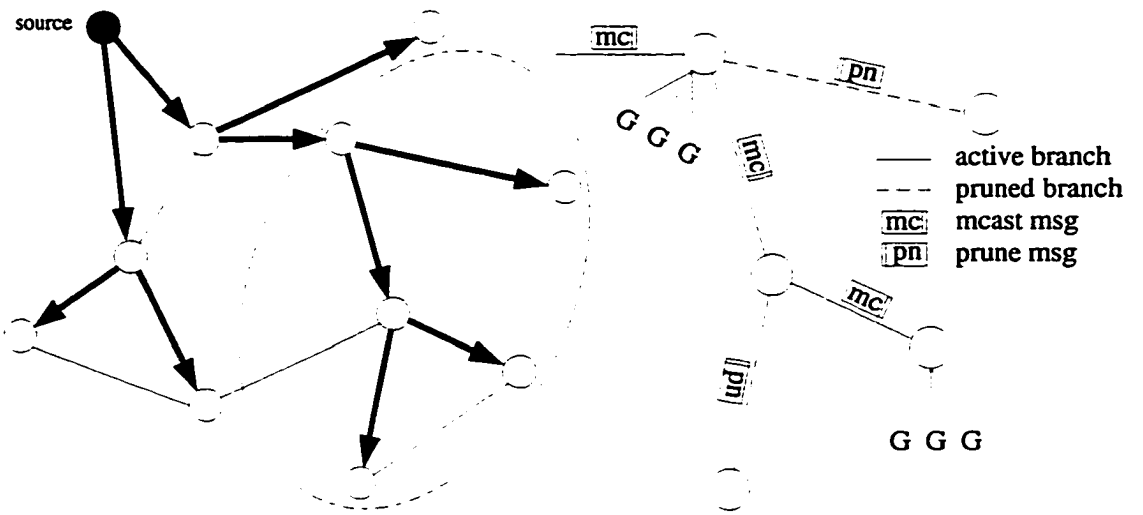


Figure 3.5 Reverse Path Multicasting.

to place such a burden on routers that are not on every (or perhaps any) active delivery tree. Unfortunately, this is still the fundamental technique of dominating multicast routing protocols used in the MBone.

3.2.3 Shared-Tree Techniques

Shared-tree techniques are an attempt to address the scaling issues, which become quite acute when most groups' senders and receivers are sparsely distributed across the internetworks.

Unlike shortest-path tree algorithms which build a source-based tree for each source, or each (source, group) pair, shared-tree algorithms construct a single delivery tree that is shared by all members of a group. The shared-tree approach is quite similar to the spanning tree algorithm except it allows the definition of a different shared-tree for each group. A shared-tree may involve a single router, or a set of routers, which comprise(s) the *core* of a multicast delivery tree. Hosts wishing to receive traffic for a multicast group

must explicitly join the shared delivery tree. Multicast traffic for each group is sent and received over the same delivery tree, regardless of the source.

Shared-tree algorithms release the burden on routers since they only require a router to maintain state information for each group. This improves the scalability of applications with many active senders since the number of source stations is no longer a scaling issue. Also, shared-tree algorithms conserve network bandwidth since they do not require the periodical flooding across all multicast routers. In addition, since receivers are required to explicitly join the shared-tree, data only flow over those links that lead to active receivers.

Despite these benefits, shared-trees may result in traffic concentration and bottlenecks near core routers since traffic from all sources traverses the same set of links as it approaches the core. In principle, shared-trees are an engineering compromise between scalability and efficiency (or cost) of the multicast delivery trees. The best (or lowest cost) delivery trees are the source-group specific shortest-path trees that require great detail information about the group membership and network topology. Thus, they do not scale well for Internet-wide applications.

3.3 Multicast Protocols Used by MBone

Early deployed solutions to providing multicast services in the Internet are represented by DVMRP [5] and MOSPF [6].

DVMRP is a distance-vector routing protocol designed to support the forwarding of multicast datagrams through an internetwork. It is the first standardized multicast routing protocol and today still dominates more than half of the routers on the MBone.

DVMRP was first defined in [5]. The original specification was derived from the Routing Information Protocol (RIP) [8, 36] and employed the TRPB technique. The

major difference between RIP and DVMRP is that RIP calculates the next-hop toward a destination, while DVMRP computes the previous-hop back toward a source. Since *mrouted 3.0*, DVMRP has employed the RPM algorithm. Thus, the latest implementations of DVMRP (*mrouted 3.8*) are quite different from the original RFC specification [8] in many regards.

Open Shortest Path First (OSPF) [11] is an Interior Gateway Protocol (IGP) [4, 37] that distributes unicast topology information among routers belonging to a single OSPF *autonomous area*. OSPF is based on link-state algorithms [37] which permit rapid route calculation with a minimum of routing protocol traffic. In addition to efficient route calculation, OSPF is an open standard that supports hierarchical routing, load balancing, and the import of external routing information.

The MOSPF is defined in [6]. MOSPF routers maintain a current image of the network topology through the unicast OSPF link-state routing protocol. The multicast extensions to OSPF are built on top of OSPF Version 2 [11] so that a multicast routing capability can be incrementally introduced into an OSPF Version 2 routing domain. Routers running MOSPF will interoperate with non-MOSPF routers when forwarding unicast IP data traffic. MOSPF does not support tunnels.

3.4 Protocol Independent Multicast

The PIM routing protocols have been developed by the Inter-Domain Multicast Routing (IDMR) working group of the IETF.

PIM is actually two protocols: PIM-Dense Mode (PIM-DM) [25] and PIM-Sparse Mode (PIM-SM) [26, 27]. While PIM-DM and PIM-SM share part of their names, and

they do have related control messages, they are actually two completely independent protocols.

PIM[4] receives its name because it is not dependent on the mechanisms provided by any particular unicast routing protocol. However, any implementation supporting PIM requires the presence of a unicast routing protocol to provide routing table information and to adapt to topology changes.

PIM makes a clear distinction between a multicast routing protocol that is designed for dense environments and one that is designed for sparse environments. Dense-mode refers to a protocol that is designed to operate in an environment where group members are relatively densely packed and bandwidth is plentiful. Sparse-mode refers to a protocol that is optimized for environments where group members are distributed across many regions of the Internet and bandwidth is not necessarily widely available. It is important to note that sparse-mode does not imply that the group has a few members, just that they are widely dispersed across the Internet.

3.4.1 Protocol Independent Multicast — Dense Mode

The PIM-DM is similar to DVMRP in that it employs the RPM algorithm. The PIM-DM protocol [25] floods the network with data packets to set up a source-based tree for every sender to every group. Initially, these trees reach every potential receiver in the network. After receiving multicast data for a group, each router that has no members of the group on its LAN sends prune messages toward the senders to remove unwanted branches from the tree of the group. This broadcast-type behavior returns periodically after the pruned interfaces have timed out. For those cases where group members

[4] Any references to "PIM" apply equally well to either PIM-DM or PIM-SM.

suddenly appear on a pruned branch of the delivery tree, PIM-DM, like DVMRP, employs graft messages to reattach the previously pruned branch to the delivery tree.

3.4.2 Protocol Independent Multicast — Sparse Mode

The PIM-SM designers observed that several hosts wishing to participate in a multicast conference do not justify flooding the entire internetwork periodically with the group's multicast traffic. To eliminate these potential scaling issues, PIM-SM is designed to limit multicast traffic so that only those routers interested in receiving traffic for a particular group *see* it. This idea makes PIM-SM differ from existing dense-mode multicast protocols in two key ways:

1. Routers with adjacent or downstream members are required to explicitly join a sparse mode delivery tree by transmitting join messages. If a router does not join the predefined delivery tree, it will not receive multicast traffic addressed to the group.
2. PIM-SM employs the concept of a rendezvous point (RP) where receivers *meet* senders.

When a new multicast group is introduced to a network that uses the PIM-SM, a RP is assigned through a hash function to the group [27]. The RP then becomes the center node of a directed, shared-tree (called RP-tree) for the group.

Each multicast router that learns via IGMP that a local host has joined a group sends a join message along the shortest path to the RP for that group. The join message triggers each router on its path to the RP to set up or update a routing entry for the RP-tree for the group. The RP-tree that is built by these actions is a directed tree rooted at the RP that can be used to deliver packets to each member of the group. Each new sender to

a group registers with the RP. In response, the RP initiates construction of a path from the sender to the RP.

The RP-tree provides connectivity for group members but does not optimize the delivery path through the internetwork. PIM-SM supports a scheme that allows router with local receivers either continue to receive multicast traffic over the shared RP-tree, or subsequently create a source-based shortest-path tree on behalf of their attached receiver(s). The change-over may be triggered by the event that the data rate from the source exceeds a predefined threshold.

3.5 Core Based Tree

The CBT [28, 29], another multicast architecture that is based on a shared delivery tree, is specifically intended to address the important issue of scalability. CBT sets up and maintains a single shared-tree for every multicast group that is active in the network. When a multicast router is notified via IGMP that a local host would like to join a group, the router sends a join message for that group toward the core node via the shortest path. The core node performs a function similar to that of the RP in the PIM-SM. A tree rooted at the core is constructed as the acknowledgments to the join messages are processed. The resulting tree is a bidirectional, loop-free graph that reaches every member of the group.

Forwarding packets to the group members using CBT is straightforward. When a node on the tree receives a packet addressed to the group, it forward copies of the packet on all branches of the group's tree except for the branch on which the packet arrived.

CBT is protocol-independent. CBT employs the information contained in the unicast routing table to build its shared delivery tree. It does not care how the unicast routing

table is derived, only that a unicast routing table is present. This feature allows CBT to be deployed without requiring the presence of any specific unicast routing protocol.

In a significant departure from PIM-SM, CBT has decided to maintain it's scaling characteristics by not offering the option of shifting from a shared-tree (e.g., PIM-SM's RP-Tree) to a shortest-path tree to optimize delay. The designers of CBT believe that this is a critical decision since when multicasting becomes widely deployed, the need for routers to maintain large amounts of state information will become the overpowering scaling factor.

Finally, unlike PIM-SM's shared-tree state, CBT state is bidirectional. Data may therefore flow in either direction along a branch. Thus, data from a source which is directly attached to an existing tree branch need not be encapsulated.

3.6 IPng and Multicast Support in IPv6

The primitive motivations for the Internet Protocol Next Generation (IPng) are that the Internet was in great danger of running out of network numbers, the routing tables were getting too large, there was even a risk of running out of addresses altogether. Nevertheless, the Internet could not have been so successful in the past years if IP current version (IPv4) had contained any major flaw. IPv4 was a very good design, and IPng should indeed keep most of its characteristics [38].

The IPng should solve the Internet scaling problem, provide a flexible transition mechanism for the current Internet, and meet the needs of new markets such as nomadic personal computing devices, networked entertainment, and device control [39].

3.6.1 Internet Protocol Version 6: IPv6

The consensus that the IETF has reached upon as a solution to the needs of the next generation Internet Protocol is IP version 6, and is formally called IPv6.

Ease of transition is a key point in the design of IPv6. As an evolutionary step, IPv6 provides a platform for new Internet functionality, yet is still able to interoperate with IPv4. The major changes are:

- *Expanded Addressing Capabilities* IPv6 increases the IP address size from 32 bits to 128 bits, to support more levels of addressing hierarchy, a much greater number of addressable nodes, and simpler auto-configuration of addresses.
- *Improved Support for Options*Changes in the way IP header options are encoded allows for more efficient forwarding, less stringent limits on the length of options, and greater flexibility for introducing new options in the future.
- *Scoped Multicast Addresses* The scalability of multicast routing is improved over IPv4 multicast by adding a *scope* field to multicast addresses.
- *Anycast Addresses* Anycast addresses identify a set of interfaces such that a packet sent to a anycast address will be delivered to one member of the set. The use of anycast addresses in the IPv6 source route allows nodes to control the path which their traffic flows.
- *Header Format Simplification* Some IPv4 header fields have been dropped or made optional, to reduce the common-case processing cost of packet handling and to limit the bandwidth cost of the IPv6 header. The way encoding IP header options is more efficient and flexible.

- *QoS Capabilities* The *Flow Label* and the *Priority* fields in the IPv6 header may be used by a host to identify those packets for which it requests special handling by routers, such as non-default quality of service or real-time service. This capability is important in order to support applications which require some degree of consistent throughput, delay, and/or jitter.
- *Authentication and Privacy Capabilities* Support for authentication, data integrity, and confidentiality are provided as basic elements of IPv6.

Figure 3.6 showing headers of IPv4 and IPv6 gives flavors of the changes.

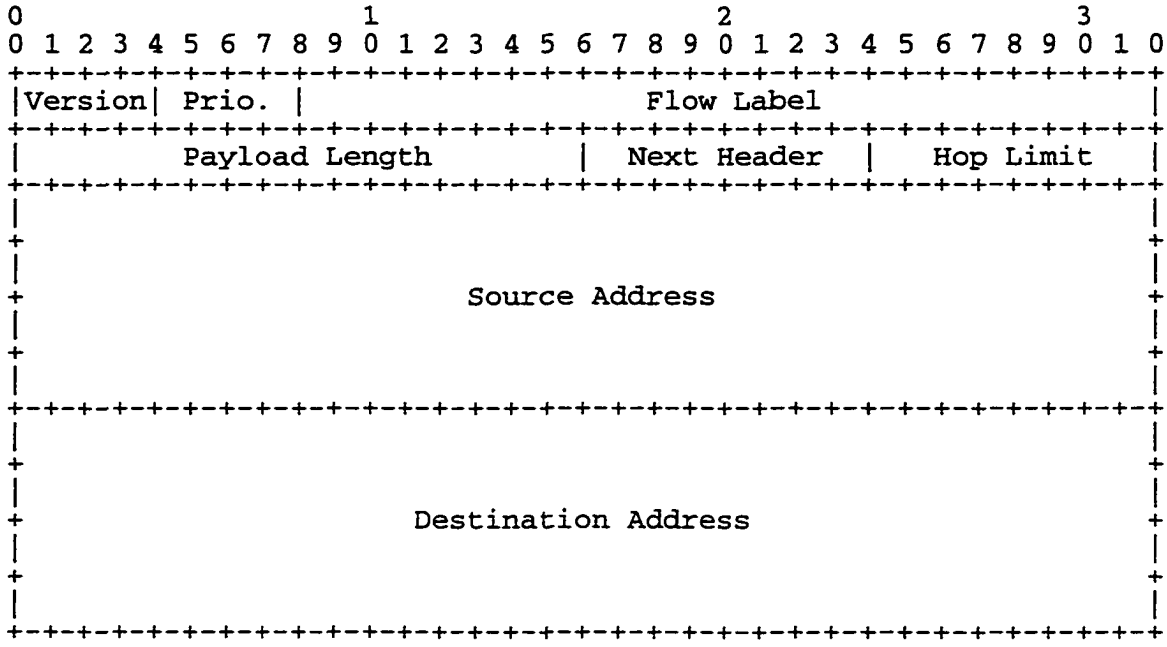
It is notable that IPv6 itself alone is not the answer to multicast or mobile IP. What the IPv6 contributes are the newly added capabilities that along with other IPv6 enhanced Internet protocols [40–42] will substantially improve multipoint and mobility support across the Internet.

3.6.2 Routing in IPv6

Routing in IPv6 is almost identical to IPv4 routing under classless inter-domain routing (CIDR) [43] except that the addresses are 128-bit IPv6 addresses instead of 32-bit IPv4 addresses. With very straightforward extensions, all of IPv4's routing algorithms (OSPF, RIP, IDRP, etc.) can be used to route IPv6. IPv6 also includes simple routing extensions which support powerful new routing functionality, such as *provider selection* (based on policy, performance, cost, etc.), *host mobility* (route to current location), *auto-readdressing* (route to new address).

The new routing functionality called *address sequence* is obtained by creating sequences of IPv6 addresses using the IPv6 *Routing* option. The routing option is used by a IPv6 source to list one or more intermediate nodes (or topological group) to be *visited*

The IPv6 Header:



The IPv4 Header:

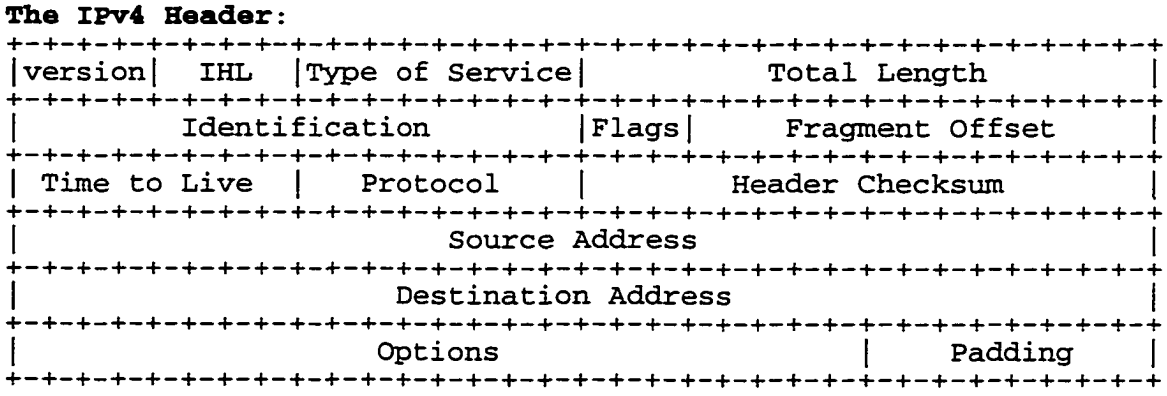


Figure 3.6 Headers of IP version 4 and version 6.

on the way to a packet's destination. This function is very similar to IPv4's *loose source and record route* option [7].

In order to make address sequences a general function, IPv6 hosts are required in most cases to reverse routes in a packet it receives (if the packet was successfully authenticated using the IPv6 Authentication Header) containing address sequences in order to return the packet to its originator. This approach is taken to make IPv6 host implementations

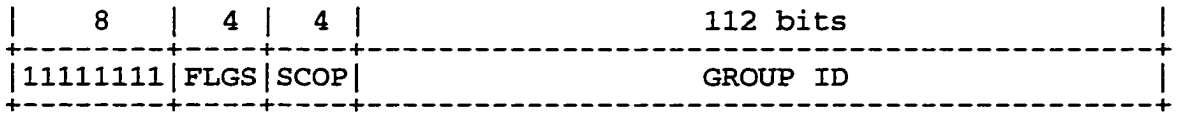


Figure 3.7 IPv6 multicast addresses format.

from the start support the handling and reversal of source routes. This is the key for allowing them to work with hosts which implement the new features such as provider selection or extended addresses.

3.6.3 Multicast Support in IPv6

Multicast capabilities were formally added to IPv4 in 1988, with the definition of class D addresses and the IGMP protocol. The deployment of these capabilities was stimulated by the arrival of the MBone in 1992. But this deployment is still very limited in comparison with the whole Internet. IPv6 takes advantage of the deployment of a new protocol to make sure that multicasting is available on all IPv6 nodes in the first place. The functions of IPv4's IGMP are incorporated in the basic Internet Control Message Protocol (ICMP) protocol of the IPv6 (ICMPv6) [40], and all routers are made capable of routing multicast packets.

Enhanced multicast capabilities over IPv4 multicast is a direct result of the scoped multicast addresses. IPv6 multicast addresses format is shown in Figure 3.7 The first octet of the address `11111111` is the multicast prefix. `FLGS` is a set of 4 flags: `000T`. The high-order 3 flags are reserved, and must be initialized to `0`. The fourth bit is abbreviated as `T` for Transient. `T=0` indicates a permanently-assigned (well-known) multicast address, assigned by the global Internet numbering authority, `T=1` indicates a non-permanently-

value	scope	value	scope
0	reserved	8	organization-local scope
1	node-local scope	9	(unassigned)
2	link-local scope	A	(unassigned)
3	(unassigned)	B	(unassigned)
4	(unassigned)	C	(unassigned)
5	site-local scope	D	(unassigned)
6	(unassigned)	E	global scope
7	(unassigned)	F	reserve

Table 3.1 Assigned numbers for the SCOP values in the IPv6 multicast addresses.

assigned (transient) multicast address. The transient multicast addresses are available for reused by different groups. At initiation of a multicast session a free address is selected, which will be returned to the address pool at session termination. SCOP is a 4-bit multicast scope value used to limit the scope of the multicast group. The values are listed in Table 3.1. While the IPv4 solely depends on careful tuning the TTL field to perform the same functionality, the IPv6 allows a much more precise control over the scope. However, it relies on routers to enforce scope boundaries. GROUP ID identifies the multicast group, either permanent or transient, within the given scope.

Shared-tree based protocols like PIM and CBT scale better than source-tree based ones like DVMRP and MOSPF. But the shared-trees created by existing shared-tree based protocols are non-optimal basically because of the randomness of the RPs/Cores due the hashing function is used. The questions then become are there any ways with limited computation cost that can generate more efficient multicast shared-trees than hashing functions, and can new feature of IPv6 help? The next chapter is devoted to discussions

of an innovative avenue to the answers.

4. Core-Manager Based Multicast Routing

Several multicast routing protocols, such as DVMRP, MOSPF, PIM-DM/SM and CBT, have been produced over the recent years. However, all of these protocols seem faced some of the scaling, stability and policy issues.

Any scalable protocol will have to minimize forwarding state requirements. In the case of dense mode protocols (DVMRP, PIM-DM), routers may carry forwarding or prune state for every (G,s) pair in the entire multicast region. This is true even for routers that may not be on any delivery tree. It seems likely that as multicast deployment scales to the size of the Internet, maintenance of (G,s) state will become intractable.

Shared-tree protocols, on the other hand, have the advantage of maintaining a single (G,*) entry for a group's receivers, thus relaxing the requirement of amount of forwarding state maintained. However, this is not without its own disadvantages, since it is well possible the case that participants of a specific multicast group belong to some service providers, say *A* and *B*, have to rely on a RP/core [44] belongs to a third service provider, say *C*, even if *C* has no receivers for the group(s) served by the RP/core. This so called *Third-Party Resource Dependencies* adds specific hardship to the provision for administrative and policy control, which is currently missing but should be required in today's Internet.

The objective of a multicast forwarding state distribution mechanism is to ensure that multicast traffic is efficiently distributed to those parts of the topology where there are receivers. Dense and sparse mode protocols accept different overhead based on design trade-offs. In the dense mode case, the data-driven nature state distribution has disadvantage that data is periodically distributed to branches of the distribution tree

which don't have receivers due to the flood and prune behavior. It seems unlikely that this mechanism will be scalable. On the other hand, sparse mode protocols use receiver-initiated, explicit joins to establish a forwarding path along a shared delivery tree. While the on-demand nature of sparse mode protocols have favorable properties with respect to distribution of forwarding state, it also has the possible disadvantage of traffic concentration problem and distant RP/core problem.

4.1 Spanning Tree Algorithms

Multicasting uses a spanning tree for a source to reach all its receivers. Depending on different design objectives, the spanning tree can be of different characteristics. Examples are shortest path tree, minimum spanning tree (Steiner tree), or a suboptimal spanning tree. Considering end-to-end delay and required bandwidth, it is ideal for multicast to distribute packets along the *source-based trees* — the collection of the shortest paths from a data source to all receivers defines a source-specific tree. Unfortunately, this approach does not scale as the number of multicast groups and sources grows because the amount of state kept at each multicast router grows explosively. One of the common ways to restrain the amount of state uses shared-tree, in which data from all sources to a group is distributed along a single shared-tree, rather than a separate tree for each source. This eliminates the need to keep per-source information for the group at each intermediate router, and consequently tends to scale better.

Ideally, a shared-tree should use a *Minimal Spanning Tree (MST)* to minimize the cost (e. g. average latency and number of hops). Finding the MST for a subset of nodes in a given network can be modeled as the NP-complete *Steiner problem* in graphs [45–48]. Consequently, its explicit solutions [47], although well established,

are prohibitively expensive for (semi-)real-time routing purpose. For example, two popular explicit algorithms, the *spanning tree enumeration* algorithm and the *dynamic programming* algorithm [48], have algorithmic complexities of $O\left(p^2 2^{(n-p)} + n^3\right)$ and $O(n3^p + n^2 2^p + n^3)$, respectively, where n is the number of nodes in the graph and $p \leq n$ is the number of nodes in the subset. Besides, these algorithms need the complete knowledge of network topologies, which is a rather tight requirement in the real-world internetworks.

A number of good, less expensive heuristics exist for near-optimal solutions to the Steiner problem [46, 47, 49]. Many of them are based on reducing the Steiner problem to the MST problem, and using a distributed MST algorithm [50]. A Steiner tree is then created by pruning the MST by removing sub-trees containing no nodes of the subset. However, distributed Steiner heuristics based on MST algorithm suffer from two drawbacks: (1) All nodes in the network must participate in the execution of the algorithm, which is undesirable, maybe even impractical in large internetworks, in order to find the MST. (2) The theoretical upper bound on competitiveness (see page 58) of a pruned MST has shown to be $s + 1$, where s is the number of Steiner nodes [49]. In comparison, the same theoretical upper bound on the shortest path heuristics for the Steiner tree problem is only $2(1 - 1/p)$, where p is the number of nodes in the subset [48].

A relaxed approach to constructing a shared-tree is to use a center-specific tree, which chooses a single node near the center of the group and takes the shortest-path tree rooted at the center of the shared-tree. Previous work [51] shows that a topologically centered tree gives a delay bound of twice that of source-specific trees. If the root is moved to a group member, the bound becomes three times that of source-based trees. Combining

the provision of cores, a more generic notion of the center-specific tree would be *multi-core based trees*. The essence of a multi-core based tree is to represent each cluster of members of a group with a core that locates at the approximate center of the cluster, then a MST is constructed with respect to the cores. The *member cluster* is a loose concept, and conceptually it means members of a multicast group are partitioned in as much a way that the *topological distances* among cluster centers are much greater than those among members of the same cluster. Obviously, finding such a partition for a given group membership is by no means a easy task, sometimes it may not be possible at all. But this does not blemish it as a positive clue on alternative approaches of constructing efficient multicast delivery trees.

In a distributed environment like the Internet, topological information is often distributed across all nodes, so that many nodes may not have complete topological information. Thus, an ideal algorithm to locate the center of a multicast group (or a member cluster) should require only a small amount of information at each node and minimal interaction among neighboring nodes. It is understood that multicast groups are dynamic, thus the optimal center will change over the time. Once the optimal center has been determined for a group with dynamic membership, the optimal center tree will gradually degrade toward a rendomly-centered tree as nodes join and leave the group. However, the larger the membership and the more uniform its distribution gets, the slower the degradation undergoes. Indeed, the optimal center is unlikely to move very much at all for groups with a relatively large number of sparsely distributed members at steady state, with members leaving and joining randomly. Consequently, there exists a trade-off between overhead of computing the optimal center and the *centrality* of the center.

From engineering point of view, it is reasonable to define a criterion which assesses a good-enough center-specific tree versus the optimal center-specific tree.

A common consensus on multicast shared-trees is that it does not have to guarantee to be the optimal trees in anytime, since as the membership changes, cost of the shared-tree may degrade to any extent if no action is taken against it. For long-term multicast sessions and those with special QoS requirement, it may be worthwhile to trade additional computation overhead with reduced tree cost. Thus, the goal is to find an on-line algorithm that can monitor the accumulated damage to an existing multicast delivery tree and modifies the tree accordingly. In the language of graph theory, the problem is defined as[5]: ***Given*** *a simple, undirected, connected, finite graph* $G = (V, E)$, *with vertices* $V = \{v_1, v_2, \ldots, v_m\}$, *edges* $E = \{e_1, e_2, \ldots, e_n\}$ *and nonnegative edge costs* $c(e_i) \geq 0$, $\forall e_i \in E$, *a subset of the vertices* M $(\subseteq V)$ *and a spanning tree* $T^{init}(M) = (V^{init}, E^{init})$, *where* $M \subseteq V^{init} \subseteq V$ *and* $E^{init} \subseteq E$, ***find*** *a sequence of trees* $T_1, T_2, \ldots, T_s$ *such that each tree* T_i *spans vertices* $M_i = M \cup R_i$ *(i.e.,* M *modified by a* request vector $R_i = (r_1, r_2, \ldots, r_i)$, $r_j \in \{join, leave\}$, $j \in \{1, 2, \ldots, i\}$*), and its cost* $C(T_i)$ *is the minimum among all possible choices for the tree* T_i, *i.e.,*

$$C\left(T_i^{C_{\min}}\right) = \min_{\forall T(M \cup R_i)} \{C(T(M \cup R_i))\} \tag{4.1.1}$$

Algorithms are available if computational cost is not concerned. In the extreme case, a multicast tree may be completely rebuilt after each change request using any static Steiner problem algorithm. Real-time multicast, however, can not afford the computation cost and does not tolerate too frequent tree-rebuilt which causes disruption in packet

[5] For definitions of various terms, notations and expressions used herein, see Appendix A.

delivery. Hence, ideal algorithm should not only be itself effective but also minimize data flow disruption. But, it is known that the Steiner problem is NP-complete so there is no such an ideal algorithm.

Algorithms with practical value ought to balance run-time against *competitiveness*

$$Comp(T) = \frac{C(T(M))}{C(T_{opt}(M))} \tag{4.1.2}$$

ratio between the cost of a solution tree and that of the optimal tree that spans the same member set. Hence, a relaxed version of the problem can be stated as: Given the same conditions, ***find*** *a sequence of trees* $T_1, T_2, \ldots, T_s$ *such that each tree* T_i *spans vertices* $M_i = M \cup R_i$, *and its competitiveness does not exceed a given bound:*

$$Comp(T_i) \leq K \tag{4.1.3}$$

Cost of a tree normally can be well estimated distributively in various ways by using local distance information at each node. Let M be the set of members of a group, the appropriate approximations of the tree cost may be the maximum distance, the average distance or maximum diameter:

$$D_{\max} = \max_{u \in M} \{sp(center, u)\} \tag{4.1.4}$$

$$\overline{D} = \frac{1}{|M|} \sum_{u \in M} sp(center, u) \tag{4.1.5}$$

$$D^{\phi}_{\max} = \max_{u \in M} \{sp(center, u)\} + \max_{v \in M, v \neq u} \{sp(center, v)\} \tag{4.1.6}$$

Another approach to approximation might be computing the upper and lower bounds on the estimated tree cost. To get a lower bound on the cost of the tree, it is observed that a best possible case tree could be *linear*, meaning that all members lie on the path from the center to the farthest member, so that the cost of the tree is simply the maximum distance from the center to any group member. When the distances are given as hop counts, the bound will be slightly tighter. Specifically, if two members are at an equal distance, the tree can not be completely linear, but must have at least one additional branch. In this case, the lower bound is

$$C_{est}^{-} = \max_{u \in M} \{sp(center, u)\} + I \tag{4.1.7}$$

where I is the number of members whose distances to the center are equal, i.e.

$$sp(center, v_i) = const,\ v_i \in \{v_1, v_2, \ldots, v_I\} \subseteq M \tag{4.1.8}$$

To get an upper bound on the cost of the tree, it is noted that a worst-case could be no links of the tree are shared among the paths from the center to each member. Thus, the maximum tree cost is $\sum_{u \in M} sp(center, u)$. Since the number of group members may be greater than the degree of the center, $|M| > d(center)$, a tightened upper bound is given as

$$C_{est}^{+} = \begin{cases} \sum_{u \in M} sp(center, u) & if\ |M| \leq d(center) \\ \sum_{u \in M} sp(center, u) - (|M| - d(center)) & otherwise \end{cases} \tag{4.1.9}$$

The estimated cost of the tree may be defined as

$$C_{est} = \frac{C_{est}^{-} + C_{est}^{+}}{2} \tag{4.1.10}$$

Certain properties of real networks can make the actual problem diverge from the traditional Steiner problem, which virtually relaxes the involvement. For instance, the shared-tree multicasting confines the center of any group to a RP/core. Because every group has to have at least one RP/core, it is natural to start the tree from the (primary) RP/core.

Another varied version of the center-specific tree problem represents a situation where only a list of sources maintained at the center, but the list of member is unknown. In this case, it is desirable to have the tree's center at the center of all sources. This is especially substantial when the number of sources is large.

The bottom line is that the algorithm will restrict to sparse networks with low average degree (2 to 3) at each node. While no consensus on definition of the sparse network is reached, it is assumed, for the purpose of this research, to be defined by $E \leq V \log V$ [52]. This is because so defined sparse networks are more representative of real-life networks, and they are inherently more difficult to solve because, in general, fewer solutions exist in a sparse network than in a dense one [47, 48]. Similarly, sparse-mode multicast satisfies $M \leq V/10$. Competitiveness based on different cost of the tree will be used to justify a tree's quality.

4.2 Core-Manager Based Multicast Routing Structure

The Core-Manager based Multicast Routing (CMMR) structure is based on a hybrid of PIM-SM and CBT with improved solutions to the issues of core selection, placement, and management. The CMMR architecture introduces a semi-centralized core management server called the Core-Manager (CM). All routers in a multicast region need to know only the address of the CM in order to initiate or join a group (on behalf of their attached

hosts), and each *Candidate Core* (CC) in the multicast region needs to make itself known only to the CM. The CM intelligently selects cores from CCs known to it for multicasting groups. If location information is available by other means, CM can find a core close to a desired point, usually close to the majority of group members. If a session has special service requirements (e.g., high guaranteed throughput), CM can select a core in such a way that the constructed multicast delivery tree conforms with the requirements. Consequently, more efficient multicast delivery trees are created with considerably lower overhead. To further improve the scalability, a hierarchical architecture using the IPv6's multicast address scope feature is proposed. The capability to intelligently select cores for multicast groups and build up more efficient multicast delivery trees with considerably lower overhead is the fundamental difference between CMMR and other shared-tree type multicast protocols.

4.2.1 Architecture Overview

A general process of joining a group in a CMMR multicast region is accomplished in the following steps: When the DR of a LAN receives a joining request (by running IGMP, for example) for a group, *G*, it sends a `Core_Inquiry` message hop-by-hop to the CM. According to where the message came from and how the message arrived the CM selects a core for the requested group, creates/updates the (group, cores) association list, and sends the identity of the selected core in a `Core_Response` message to the Core-Inquirer that is identified by the source address of the `Core_Inquiry` message. Then the Core-Inquirer sends a `Join` message hop-by-hop to the selected core. Intermediate routers that forward the `Join` message set up *G* state such that a new branch of the multicast delivery tree is created. Branches (consequently the entire tree) produced in this way is

symmetric in both directions, meaning that a branch is used to deliver multicast packets to its attached leaves (group members), as well as to distribute multicast datagrams sent by those leaves to other parts of the tree.

Process of leaving a group is relatively simpler: After all attached members of a group having left the group, the DR prunes itself off the multicast delivery tree if it finds itself a leaf-node of the tree and itself is not a core; if the DR is a core, it has to get permission from the CM before pruning itself off the tree. Because if this DR is the only core of the group, it has to stay on tree even as a leaf-node. This process is chained in such a way that any leaf-nodes resulted from the previous pruning will recursively run it until one of above mentioned checkups fails.

4.2.2 Functions and Operations

4.2.2.1 Bringing Up the CM and Maintaining a Core-Set

A principle for the selection of the CM is that it should be a node (router) having relative high degree of network interfaces located at about the center of an internetwork. Later it will be clear why the CM placed this way could select better cores for the multicast group members. The CM is responsible for advertising itself to all routers in the multicast region, maintaining an up-to-date core-set and a (group, cores) association list for each active group, selecting cores for multicast groups specified by `Core_Inquiry` messages, and replying the Core-Inquirers with the selected cores. Since all activities of group joining in a multicast region rely on the proper operation of the CM, certain redundancy in the CM bringing up scheme helps preventing it become a potential single point of failure.

In a multicast region, a small number of routers are configured as candidate CMs (CCMs). Each CCM maintains a core-set (initially empty) and advertises to the multicast region. An election routine runs periodically such that one CCM is elected as the CM, and the others become backups. Then the CM starts advertising itself to all routers in the region. Routers configured as CCs periodically register with the CM to show their availability as cores. Upon reception of a CC's register message, the CM creates a corresponding entity in the core-set if none exists yet or refresh the existing entity. Aged entities will be deleted from the set. Incremental of the core-set is sent to backup CMs to keep their copies synchronized.

4.2.2.2 Selecting Cores for a New Group

The fundamental principle in selecting cores is that a core should be as close to (at least some of) the group members as possible. In responding to a `Core_Inquiry` message, the CM examines the type of the message sender (a regular router or a CC) and from which network interface the message is received so as to deduce a core for the requested group by one of the following three basic core selection rules.

En-Route-Core rule: If a `Core_Inquiry` message for a group is from a CC, this CC is selected as a core for the requested group. Clearly, this rule applies when the DR initiated the`Core_Inquiry` message happened to be a CC. But this case is relatively rare because only a small percent of the nodes in a multicast region are configured as CCs. A more general case where the En-Route-Core rule is applicable occurs when a `Core_Inquiry` message sent by a non-CC encounters a CC on its way to the CM. Since `Core_Inquiry` messages proceeded hop-by-hop towards the CM, the chances that one of the intermediate routers is a CC is much higher than the `Core_Inquiry`

initiator itself is a CC. Let the first encountered CC take over the core inquiry process and become a proxy for the Core-Inquirer. Then this en-route CC continues the process by sending a new `Core_Inquiry` message to the CM for the same group as the one specified by the intercepted `Core_Inquiry` message. `Core_Inquiry` messages sent by a CC will not induce any proxy. Upon reception of the `Core_Inquiry` message, the CM can tell by looking at the source address that it is from a CC. So this CC is selected as a core for the requested group, and a corresponding entry in the (Group, Cores) association list is created. The CM informs the source of the `Core_Inquiry` message (i. e. the proxy CC) this result by sending it `Core_Response` message. Then the proxy CC forwards the `Core_Response` message to the original Core-Inquirer so that the latter can join the core (the proxy CC). This case is exemplified by a network shown in Figure 4.1: Router-21 initiates a group, Router-4 is the en-route CC that becomes a core for the group. Apparently, the core selected in this way is, in most cases, closer to the requesting member than a one derived by a hashing function.

Co-Interface-Core rule: If a `Core_Inquiry` message for a group is from a non-CC, the selected core is a CC that can be reached via the shortest path through the same interface on which the `Core_Inquiry` message arrived. When multiple co-interface CCs exist, a simple rule like the round-robin may be used to balance loads among the available ones. Refer to Figure 4.1. An example of this case is a `Core_Inquiry` message from Router-16 arrived on the link between the CM and Router-6, then Router-10 is selected as a core for the group. The core selected this way and the Core-Inquirer most likely locate on the same side of the CM. Therefore, this selection is again better on the average than a hashing function when the CM is placed at approximate center of the entire region.

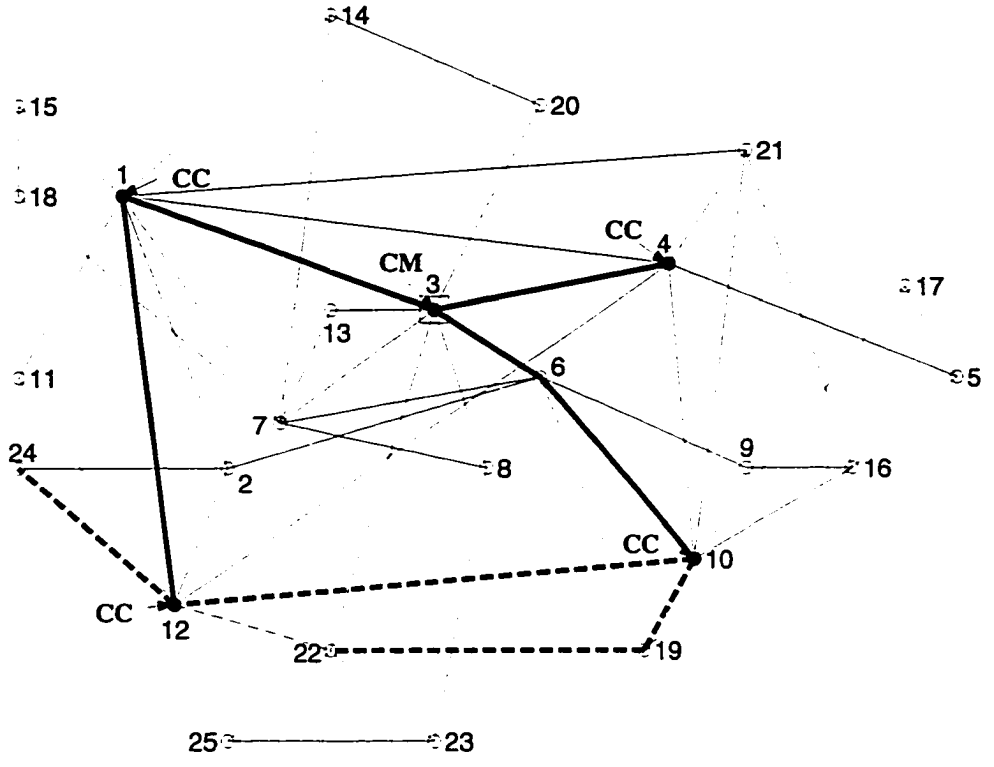


Figure 4.1 A 25-router (DR) multicast region, where Router-3 is the CM, Router-1, 4, 10 and 12 are CCs. Bold lines are the shortest path for the CM to reach the CCs. Thick dashed lines are the multicast delivery tree of a group. Thin dashed lines are the intended path of the`Join` message sent by Router-25.

Random-Core rule: If none of the above two rules is applicable and there is no other information available, a core is randomly selected from the core-set. Although, on the average, core selected in this way would not be better than the one selected by a hashing function in terms of costs of multicast delivery trees, the centralized core selection scheme leaves room to apply other policies and security.

4.2.2.3 Extending Multicast Delivery Trees

Even though the above core selection rules are discussed with respect to setting up new trees, the same principle is valid for extending an existing tree. While all three rules can be used as they are, the core pool has to be changed from the core-set to the *registered-on-tree-cores* (those cores that are currently associated with the requested

group as indicated by the CM's (group, cores) association list). Besides, the En-Route-Core rule will never work alone since even if a `Core_Inquiry` message is from a CC (either an originator or a proxy), this CC must be further directed to an on-tree-core where either the Co-Interface-Core rule or the Random-Core rule will be used so that the former can become part of the delivery tree.

Another important observation is that there are great chances for the control messages (i. e. `Core_Response` or `Join` messages) being exchanged to encounter an on-tree router when a delivery tree of the concerned multicast group already exists. The multicast delivery tree's self-growing scheme is designed to take advantage of these encounters. By running self-growing scheme, any router on the tree of a group can complete a joining process when receiving an `Core_Inquiry` or a `Join` message for the group. This clarifies why these messages are always forwarded hop-by-hop. Look at `Core_Inquiry` messages first. When using the self-growing scheme, if a `Core_Inquiry` message encounters an on-tree router on the way to the CM, the on-tree router will not forward the message further, rather it will send a `Core_Response` with its own identity as the selected core to the Core-Inquirer so that the latter can join the tree at the encountered on-tree router by sending it a `Join` message. If a `Core_Inquiry` message is not lucky enough to meet any on-tree router, the `Join` message from the same source still has a good chance to encounter one before reaching the selected core, since these two types of messages normally take different paths.

A more valuable feature of the self-growing scheme is that it eliminates potential loops. This is evident in handling `Join` messages. Since the path traversed by a `Join` message becomes an extended branch of the tree, a loop might occur if the `Join` message

passed through an encountered on-tree router and finally reached the selected core that obviously is also on the same tree. Another way of looking at this: there is already a path between the encountered on-tree router, if there is any, and the selected core, however, this path is not necessarily the route taken by the `Join` message. An illustrative example of this case is shown in Figure 4.1 where an existing tree for a multicast group is represented by the thick dashed lines. Assume Router-25 now is joining the group and Router-12 is its selected core. The intended path of the Router-25's `Join` message is shown as the thin dashed lines. If Router-22 as an encountered on-tree router did not stop the `Join` message, a loop consisting of Router-10, 12, 22 and 19 would be created.

The last thing to clarify is why an on-tree router encountered by a `Core_Inquiry` message needs to send a `Core_Response` message to trigger a `Join` message from the Core-Inquirer rather than joining it directly. The rationale behind this is: there is no guarantee that the `Core_Response` message's route is the reverse of that taken by the `Core_Inquiry` message. If the encountered on-tree router simply sent a `Join` message back to the Core-Inquirer and this message encountered another on-tree router on its way to the Core-Inquirer, not only was a loop created but also the Core-Inquirer failed to join the tree. Thus, basic rule is to always let an isolated node join the tree, not the other way around (i. e. an on-tree router joins an isolated node).

4.2.3 Nonmember Hosts Sending to a Multicast Group

When a nonmember host first transmits a multicast packet to a group, it has to find out whether its DR has been on the group's delivery tree. If it is the case, multicast packets can be sent in bare form, otherwise they must be encapsulated in a unicast `Core_Inquiry` messages and addressed to the CM that will redirect it to a core of

the group for distribution across the delivery tree after decapsulating. Meanwhile, the envelop packet will be processed as usual. All previously discussed rules in processing the `Core_Inquiry` messages are applicable here. In so doing, the source node will eventually realize a nearby on-tree node such that the DR can switch the destination of encapsulated multicast packets from the CM to this on-tree node. Alternatively, the DR may decide join the delivery tree in order to send plain messages, should the amount of data to be sent be large.

4.2.4 Algorithm of Building Up Multicast Delivery Trees

The complete multicast delivery tree building up algorithm is a combination of the basic core selection rules and the self-growing scheme. Here are the routines for different protocol entities. A router running this routine is referred to as *this router* in the following Routine description.

Core-Inquirer Routine: run by a DR that receives a join request from a local host for a group, *G*, that the DR is not currently in, or run by a Core-Inquirer's proxy.

- Prepare a `Core_Inquiry` message indicating *this router* is a CC or not.
- Send the `Core_Inquiry` message hop-by-hop to the CM.
- Wait for a `Core_Response` message.
- Send a `Join` message hop-by-hop to the selected Core as indicated by the received `Core_Response` message.

Intermediate Router Routine: run by each intermediate router.

- **If** the message being forwarded is `Core_Inquiry`,

  if *G* state exists at *this router*,

intercept the message,

send a `Core_Response` message with *this router*'s identity as the selected Core back to the Core-Inquirer.

else if the message is sent by a non-CC *and this router* is a CC,

intercept the message,

call the Core-Inquirer Routine,

send a `Core_Response` message with *this router*'s identity as the selected core back to the Core-Inquirer.

else

forward the message.

- **If** the message being forwarded is `Join`,

if no *G* state exists at *this router*,

create *G* state,

forward the message.

else

stop the message,

update *G* state.

Core Manager Routine: run by the CM upon reception of a `Core_Inquiry` message.

- **If** *G* exists in the (group, cores) association list,

apply the Co-Interface-Core rule to the on-tree-cores.

if no core is selected,

apply the Random-Core rule to the on-tree-cores.

- **Else**

  apply the Co-Interface-Core rule to the core-set.

  if no core is selected,

  apply the Random-Core rule to the core-set.

- Send the selected core in a `Core_Response` message to the Core-Inquirer.
- **If** the Core-Inquirer is in the core-set,

  register it to the (group, cores) association list.

4.3 Performance Evaluation

In order to evaluate the performances of proposed multicast routing, various properties of multicast delivery trees created by CMMR and PIM-SM are compared. The PIM-SM is selected for its maturity among the existing shared-tree based multicast routing protocols, and its similarity with CMMR in the sense that both of them do not rely on any particular unicast routing protocols. The essence of shared-tree multicasting technique is to confine both the amount of multicast forwarding state at each intermediate router and the cost of the delivery trees for given multicasting internetworks. Sometimes it may be of interest to consider, in addition to delivery cost, the delivery latency, which is crucial for real-time multicasting services. Since, in general, for a given internetwork the minimum cost tree differs from the minimum latency tree (also known as shortest path tree) for the same membership, it is important to evaluate tree performances using both criteria.

Complexity of the protocol itself represented by control traffic overhead in managing the cores, number of steps of building up and extending a delivery tree are also of interest.

4.3.1 Protocol Overhead

The protocol overhead is composed of three parts: communication overhead, storage overhead, and computation overhead.

Assume a multicast region having N routers, n of them are configured as CCs[6]. In CMMR, the periodic control traffic of maintaining the core-set would need to transmit n unicast register messages of size R from each CC to the CM. Advertising the CM to all router requires one advertisement message of size A. The traffic overhead in managing the core-set and advertising CM is estimated by $O_{CM} = nR + A$. The PIM-SM has a bootstrap router do the same job, except that it needs to distribute the core-set, or incremental of it, to all routers in the multicast region. So its overhead can be as large as $O_{PIM-SM} = nR + nA$, where the size of advertisement message is conservatively assumed n times of CM's. Since the advertisement is broadcasted to the entire region, the difference in the overhead is significant as n becomes large. In terms of the storage overhead, the PIM-SM requires that every DR stores a copy of the core-set while CMMR needs only a single core-set to be stored at the CM though its contents is a bit more complex than that of PIM-SM due to the structure of the (*Group, Cores*) association list. Besides the control traffic in maintaining the core-set, PIM-SM also needs additional mechanism to handle the asynchronism among the core-set distributed at each and every node all over the multicast region. In other words, all node running PIM-SM have to

[6] Although PIM-SM uses *Rendezvous Point*(RP) as the official notion of core, the functionality of a RP plays in PIM is very similar to that of a core does in CMMR. In most of the times, it would be not ambiguous to use either term but not both. The notion of core, therefore, will be used solely whenever the contexts make it clear.

synchronized about CCs, but in CMMR only the CM has to be updated about any change of CCs' availability. Regarding computation overhead, the CMMR, in most time, needs to perform a searching among the co-interface Cores, which is only a subset of the core-set. Given that the computational costs of searching and hashing are comparable, the CMMR out performs the PIM-SM since the smaller the size of the searching space, the lower the computation overhead. Table 4.1 summarizes the control overhead in CMMR and PIM-SM.

		CMMR	PIM-SM
communication	advertising	CM	all CCs
	Core-set maintn.	CC unicast	CC braodcast
	Core-set synch.	no	yes
storage	location	CM	all DRs
	content	{CC,interface}	{CC}
computation		mapping	hashing

Table 4.1 Comparison of control overhead in CMMR and PIM-SM.

In CMMR, joining (creating) a multicast group always costs three message exchanges (`Core_Inquiry`, `Core_Response` and `Join`) but its latency can be very short due to the on-tree router's self-growing capability. Depending on at which stage an on-tree router is encountered the join latency varies from the best case where a `Core_Inquiry` message encounters an on-tree router to the worst case where a `Join` message does so. Advantageously, the join latency tends to decreasing as the delivery tree expands since the self-growing capability will be used more frequently. In PIM-SM, joining a multicast group claims two steps regardless of the scale of the existing tree: The DR computes a hash function to locate a core, then the DR attaches itself to the delivery tree by sending

a `Join/Prune` message to the RP/core. Though the member joining process in the PIM-SM looks simpler than in the CMMR, but it does not necessarily mean faster since the number of hops that the exchanged messages traverse and the relative distance between the DR and core also matter. Moreover, the quality of the delivery trees is an equally, if not more, important issue. This one time extra overhead in joining a group can be easily paid off by the superior tree quality of CMMR as shown next.

4.3.2 Cost of Multicast Delivery Trees: Examples

Since the delivery tree of a multicast group is heavily topology-dependent, it is very helpful to intuitively compare performances of the delivery trees produced by different protocol with a concrete example network as shown in Figure 4.1. Extensive investigation with sophisticated simulations is to be presented in the next section. This example topology represents a multicast region, where the 25 nodes are DRs for their directly attached hosts. It is assumed that efforts were made such that the CM locates at the approximate center of the whole region and each quadrant has one CC at its approximate center: Router-3 is the CM, and Router-1, -4, -10 and -12 are CCs, denoted as:

$$\mathfrak{C} = \{R_1, R_4, R_{10}, R_{12}\} \tag{4.3.1}$$

The same CCs will be used in both cases running CMMR and PIM-SM. Quality of a delivery tree is estimated by the count of the links composing the tree.

A couple of typical cases are to be examined. In an unevenly distributed membership scenario, a group, G_1, has members Router-5, -19 and -25, denoted as:

$$\mathfrak{M} = \{R_5, R_{10}, R_{12}\} \tag{4.3.2}$$

Carefully reviewing these routers, one may find that if Router-5 initiates the group, the En-Route-Core rule applies; if Router-19 initiates the group, the Co-Interface-Core rule

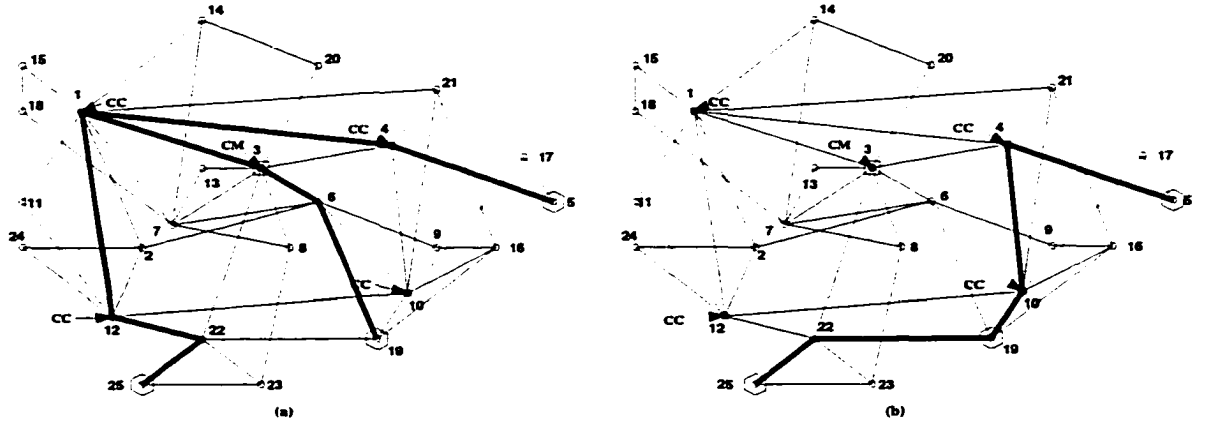


Figure 4.2 The CM made delivery trees for G_I with uneven distribution members. (a) The worst case delivery tree when Router-25 initiates. (b) The best case delivery tree when anyone other than Router-25 initiates.

applies; and if Router-25 initiates the group, the Random-Core rule applies. The CMMR will create different delivery trees depending upon the sequence that the members join the group. Three member yields $\mathfrak{S} = |\mathfrak{M}|! = 3! = 6$ possible join sequences, and the probability for each sequence is $P_{seq} = 1/\mathfrak{S}$. Assume the probability that all member at any position of the join sequence is equal. Although it is over elaborative to exhaustively walk through all possible trees for each join sequence, it is worthwhile to closely look at how the worst tree and the best tree are created. First let us look at the worst tree. If Router-25 initiates the group (with the probability of $P_f = P(R_i\ joins\ first) = 1/3,\ R_i \in \mathfrak{M}$), a `Core_Inquiry` message addressed to the CM via Router-22 will not encounter any CC. Since this a new group, the Co-Interface-Core rule is applied to the core-set, but the result is null. Consequently, the Random-Core rule is applied to the core-set, of which the worst case will be that Router-1 is selected (with the probability of $P_c = P(R_i\ selected) = 1/4,\ R_i \in \mathfrak{C}$) as the core. If Router-19 is the next joiner (with the probability of $P_s = P(R_{19}\ joins\ second \mid R_{25}\ joins\ first) = 1/2$) thus sends a `Core_Inquiry` message to the CM. The CM will apply the Co-Interface-Core rule and

maybe the Random-Core rule to the on-tree-cores. In this example, the Random-Core rule has to be applied and Router-1 will be selected again since it is the only on-tree-core. After the last member, Rotuer-5, having joined, the worst delivery tree with quality metric equals 8 as shown in Figure 4.2 (a) is created. Note that the probability of getting such a worst case delivery tree is only

$$P_{w1} = P_f \times P_c \times P_s = P_{seq} \times P_c = 1/24 \tag{4.3.3}$$

If the join sequence is Router-(25, 5, 19), and it happens that Router-1 is selected from the on-tree-cores when Router-19 joins, the worst tree is also created. The probability of this case is

$$P_{w2} = P_{seq} \times P_c \times P(R_1 \; selected \; from \; \{R_1, R_4\}) = 1/48 \tag{4.3.4}$$

Thus the total probability to obtain the worst tree is

$$P_w = P_{w1} + P_{w2} = \frac{1}{16} \tag{4.3.5}$$

Applying the above techniques again, one can easily figure out that the best tree with quality metric equals 5 shown in Figure 4.2 (b) is created with much larger probability. As a matter of fact, the join sequences that results in the best tree are not unique. The join sequence Router-(19, 25, 5) always results in the best tree no matter what. Its probability is

$$P_{b1} = 1/6 = 1/6 \tag{4.3.6}$$

The join sequence Router-(19, 5, 25) will result in the best tree if Router-10 is selected form the on-tree-cores when Router-25 joins. The probability of this is

$$P_{b2} = P_{seq} \times P(R_{10} \; selected \; from \; \{R_4, R_{10}\}) = 1/12 \tag{4.3.7}$$

The best tree is also possible out of the join sequences Router-(25, 5, 19) and Router-(25, 19, 5) under certain conditions. These conditions for the first join sequence are Router-10 is selected when Router-25 joins and Router-10 is selected again when Router-19 joins. These conditions for the second join sequence are Router-10 is selected when Router-25 joins and Router-10 is selected again when Router-5 joins. The probability for either of these two sets of conditions is

$$P_{b3} = P_{sep} \times P_c \times P(R_{10}\ selected\ from\ \{R_4, R_{10}\}) = 1/48 \tag{4.3.8}$$

Thus the total probability to obtain the best tree is

$$P_b = P_{b1} + P_{b2} + P_{b3} \times 2 = \frac{1}{3} \tag{4.3.9}$$

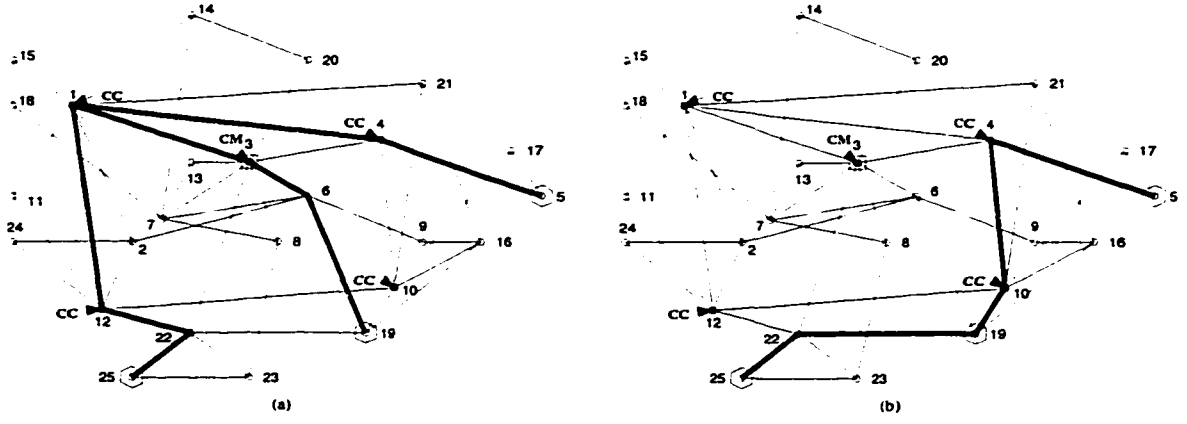


Figure 4.3 The PIM-SM made delivery trees for G_1 with uneven distribution members. (a) The worst case delivery tree regardless who initiates. (b) The best case delivery tree regardless who initiates.

Now let us look at the PIM-SM. The PIM-SM made worst tree with quality metric equals 8 and best tree with quality metric equals 5, independent of the join sequence, will look like those shown in Figure 4.3. Table 4.2 exhausts all possible trees and their

quality metric alone with their probabilities made by both CMMR and PIM-SM for easy comparison. Symbols under the "selected cores per each member" column take a format $C_i, C_j(C_k), C_l$ or similar, which represents the selected cores at each stage the members in the corresponding join sequence join the group. It means C_i is the selected core for the first member in the corresponding join sequence, C_j is the selected core for the second member in the corresponding join sequence with C_k as its proxy (refer to the En-Route-Core rule), and C_l is the selected core for the third member in the corresponding join sequence. At the bottom of the table, weighted averages of metric of the trees created by both CMMR and PIM-SM are shown.

It is clear from Table 4.1 that the CMMR trees are more efficient on the average than PIM-SM trees when the group members are unevenly distributed across the network. It is also interesting to see an evenly distributed membership scenario. Assume a group, G_2, has members

$$\mathfrak{M} = \{R_6, R_7, R_{15}, R_{17}, R_{19}, R_{20}, R_{24}, R_{25}\} \tag{4.3.10}$$

Since the number of possible trees is scaled by $|\mathfrak{M}|!$, which is too much to every one, only two typical delivery trees each made by CMMR and PIM-SM are shown in Figure 4.4. More solid evaluations will be done with simulations. The CMMR tree is better because of fewer branches, and would be more efficient due to its multi-core structure when a nonmember, like Router-16, sends to the group (see the dotted lines).

4.3.3 Cost of Multicast Delivery Trees: Simulations

In order to evaluate the performances of proposed multicast routing, various properties of multicast delivery trees created by both CMMR and PIM-SM[7] are compared.

[7] Only the shared-tree created by PIM-SM is concerned, and the source-based tree is not.

	CMMR			PIM-SM		
join sequence	selected cores per each member	metric	prob.	core	metric	prob.
5, 19, 25	4, 4, 4	6	1/6	1	8	1/4
5, 25, 19	4, 4, 4	6	1/6			
19, 5, 25	10, 10(4), 10	5	1/12	4	6	1/4
	10, 10(4), 4	6	1/12			
19, 25, 5	10, 10, 10(4)	5	1/6			
25, 5, 19	1, 1(4), 4	7	1/48	10	5	1/4
	1, 1(4), 1	8	1/48			
	4, 4, 4	6	1/24			
	10, 10(4), 10	5	1/24			
	12, 12(4), 12	6	1/48			
	12, 12(4), 4	6	1/48			
25, 19, 5	1, 1, 1(4)	8	1/24	12	6	1/4
	4, 4, 4	6	1/24			
	10, 10, 10(4)	5	1/24			
	12, 12, 12(4)	6	1/24			
average		5.81			6.25	

Table 4.2 Quality of multicast delivery trees produced by CMMR and PIM-SM.

The intuitive comparison with a simple example network and some typical memberships done in the last section is illustrative but not comprehensive. In order to avoid the comparison from any specific networks, the random graph model presented in [53] is used. In this model, nodes are randomly distributed over the unit square in a Cartesian coordinate system. The probability that an edge exists between any two nodes u and v is given by the probability function

$$P(\{u,v\}) = \beta \exp\left(-\frac{d(u,v)}{L\alpha}\right) \tag{4.3.11}$$

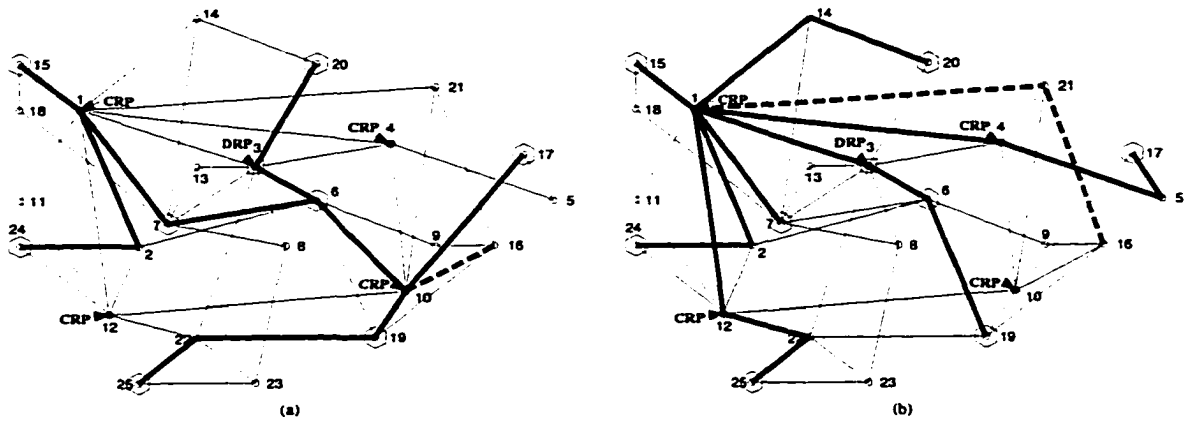


Figure 4.4 Efficiency comparison of delivery trees made by the CMMR and PIM-SM for G_2 with even distribution of members. (a) CMMR's average delivery tree for G_2. (b) PIM-SM's average delivery tree for G_2.

where $d(u, v)$ is the distance between the two nodes, L is the maximum possible distance, and α and β are parameters in the range $0 < \alpha, \beta \leq 1$. Larger values of α increase the proportion of longer edges to shorter edges, and larger values of β increase the average node degree.

Due to the limitation of the computational power of the available facilities, networks used in the simulation is decided to consist of 250 nodes. The simulation is designed as follow. A series of 250-node random networks with average node degree 3-3.5 is generated. Figure 4.5 shows a typical sample of such networks. Each of the networks is represented by $N = \{R, L\}$ with routers (i. e. nodes) $R = \{r_1, r_2, \ldots, r_m\}$ and links $L = \{l_1, l_2 \ldots, l_n\}$. There is a cost associated with each link in the network, which is defined in the simulations as the distance between its two end-routers: $c(l_{i,j}) = d(r_i, r_j)$. However, the usage of other cost metric (e. g. expected delay, link capacity, etc) would not alter the conclusions derived from the simulations and presented below, because other cost metric could be easily incorporated by modifying the weighting coefficients in the cost function. The CM is placed at the center of a network, which ensures a solid ground

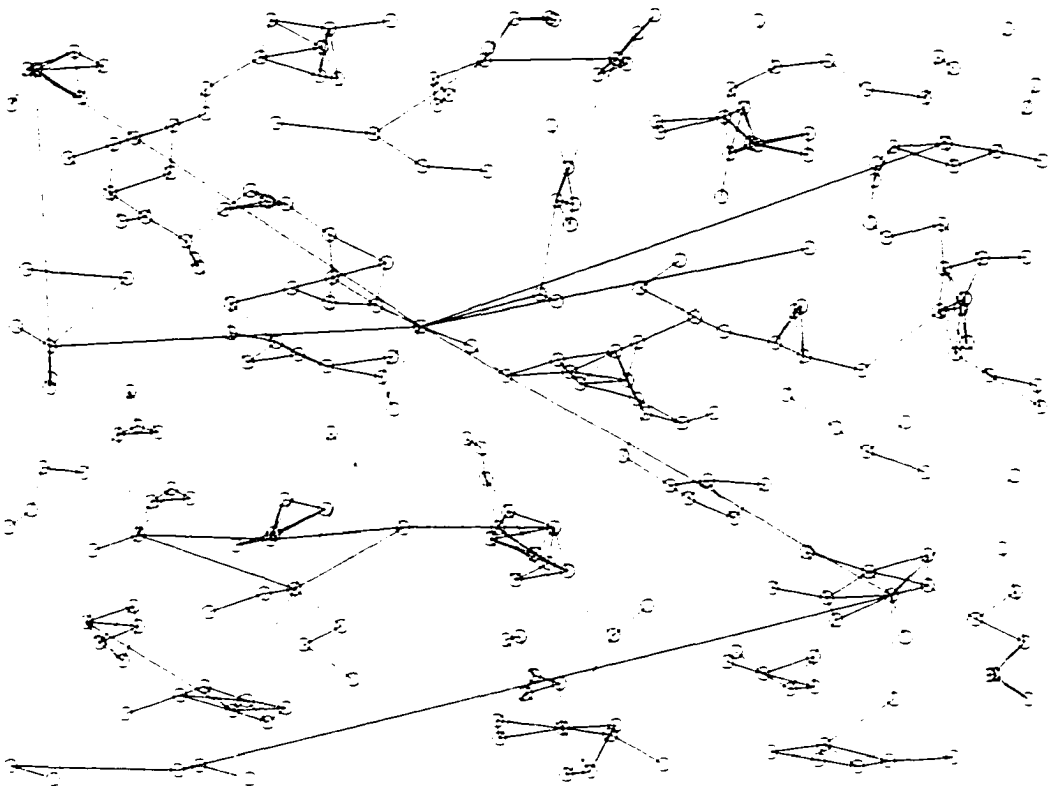


Figure 4.5 A 250-node random network with average node degree equal to 3.256.

for the Co-Interface-Core rule to play. The center of the network is the router whose average distance to all other routers in the network is minimum: If the average distance from a router r_k to all other routers in the network is $\overline{D}_{r_k} = \frac{1}{|R|} \sum_{r \in R} sp(r_k, r)$, where $|R|$ is the total number of routers, and $sp(r_k, r)$ is the shortest path along the network between r_k and r[8], then the center of the network is determined by $\overline{D}_{r_{center}} = \min_{r_k \in R} \{\overline{D}_{r_k}\}$. The density of cores and their location are factors effecting the average performance. In general, small number of cores causes severe traffic concentration at the cores; large number of cores introduces a lot protocol overhead. In the simulations, about 20% of the total routers randomly distributed throughout a network are designated as CCs with the restriction that a leaf-node will never be a CC. Performance of the CMMR trees and

[8] The shortest path between two node in a network is found by the Dijkstra's algorithm [52].

PIM-SM trees are normalized by that of the near optimal Steiner trees computed by the heuristics presented in [54] (also see Appendix B).

Let a tree in a network $N = \{R, L\}$ be represented by $T = \{R_T, L_T\}$, $R_T \in R$, $L_T \in L$. Four metric describing different aspects of the performance of the trees are to be examined:

1. The hop count, $H = |L_T|$;
2. the link cost, $C_T = \sum_{l \in L_T} c(l)$;
3. the average distance along the tree among members,

$$\overline{D}_T = \frac{1}{|M|} \sum_{r_k \in M} \left[\frac{1}{|M|} \sum_{r \in M} sp_T(r_k, r) \right] \tag{4.3.12}$$

where $sp_T(r_k, r)$ is the shortest path along the tree between r_k and r, and $M \in R_T$ is the set of all routers in a multicast group;

4. and the tree diameter which is the maximal distance along the tree between the two farthest separated members,

$$T_{diam} = \max_{r_k \in M} \left\{ \max_{r \in M} [sp_T(r_k, r)] \right\} \tag{4.3.13}$$

Since the concern is sparse mode multicasting over a large internetwork, costs of trees are observed for group sizes from 3 members up to 10% of the total routers. It is found that when the group sizes grows beyond 10% of the total routers, the difference between performances of multicast delivery trees created by CMMR and PIM-SM tends to vanish. This behavior is actually expected. Take the extreme case where all routers are members of a group, then the PIM-SM tree becomes a shortest path tree rooted at its RP/core,

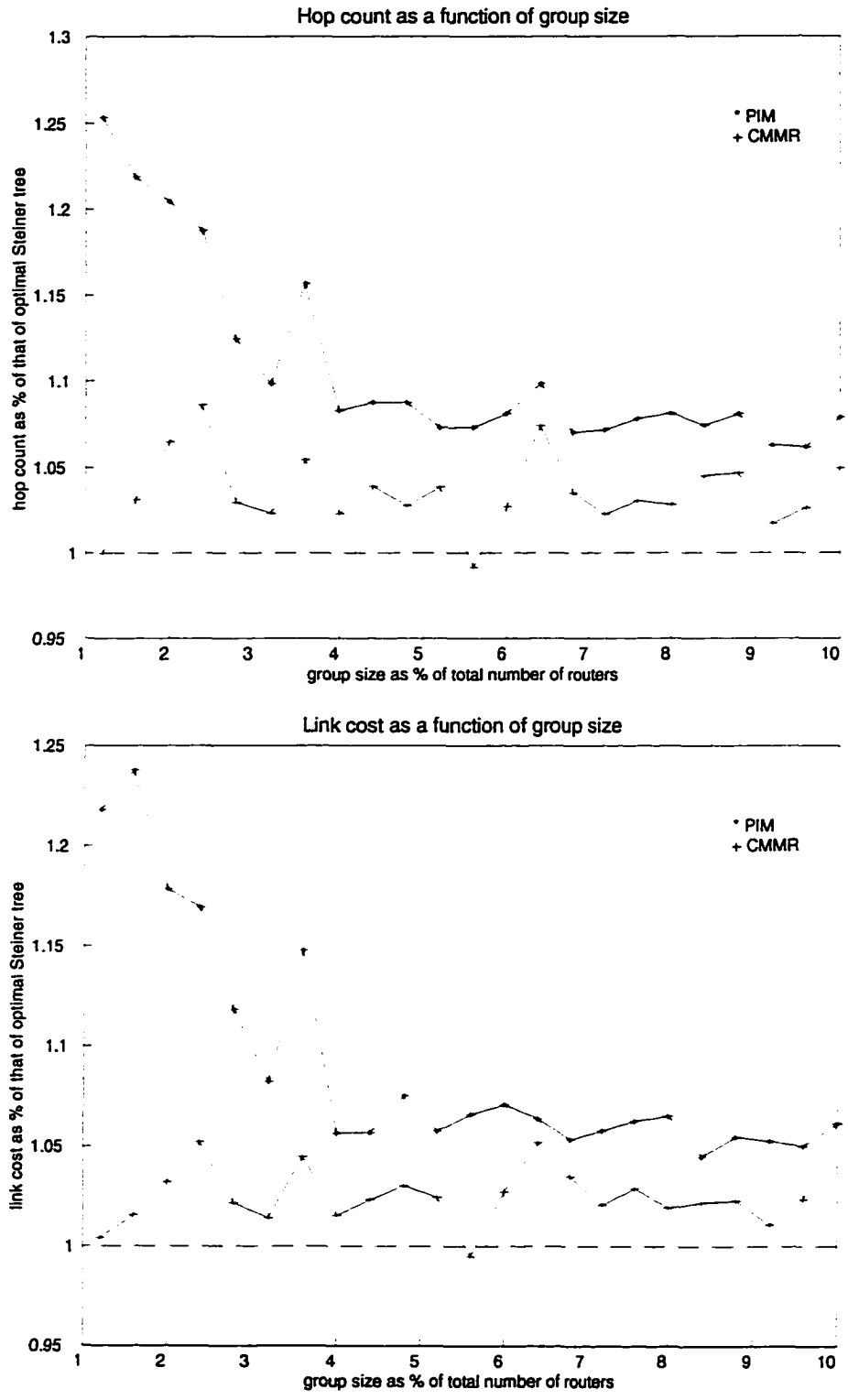


Figure 4.6 Tree costs of different group sizes.

the optimal Steiner tree becomes a minimum spanning tree, and the CMMR tree will be anywhere in between.

Simulations are run over 5 different random networks, and the averages are obtained. For each network, 10 iterations each with different membership sets are performed for every group size. The join sequence for each group size is arbitrary. The RP/core of the PIM-SM is randomly selected from the predefined CCs. Curves shown in Figure 4.6 and 4.7 are rather promising: While the trends for the hop counts and the link costs are well resembled, the costs of the CMMR trees are very close to the reference values (near optimal Steiner heuristics) regardless of the group size, and the CMMR occasionally performs even better than the reference algorithm (e. g. when the group size is 5.6% of the total number of routers). But the costs of PIM-SM trees are relatively high especially when the group sizes are small. The reason behind these phenomenons is obvious: For a small size group, with PIM-SM there are good chances that the core is far away from any members; with CMMR, due to the En-Route-Core rule and/or Co-Interface-Core rule there are great chances to locate a core that at least close to one member. When delivery latency is the concern (refer to Figure 4.7), the results are more encouraging: the CMMR's results for the average distance and the diameter of the tree are not only always better than that of the PIM-SM, but in most of the times better than that of the reference algorithm. This is not surprising because reference trees are optimized with respect to the link cost. Nonetheless, the good performance for these latency related parameters is desirable for delay-sensitive multimedia services.

So far, only static group membership is examined. It is also interesting to find out how performances of the multicasting delivery trees change with a dynamic mem-

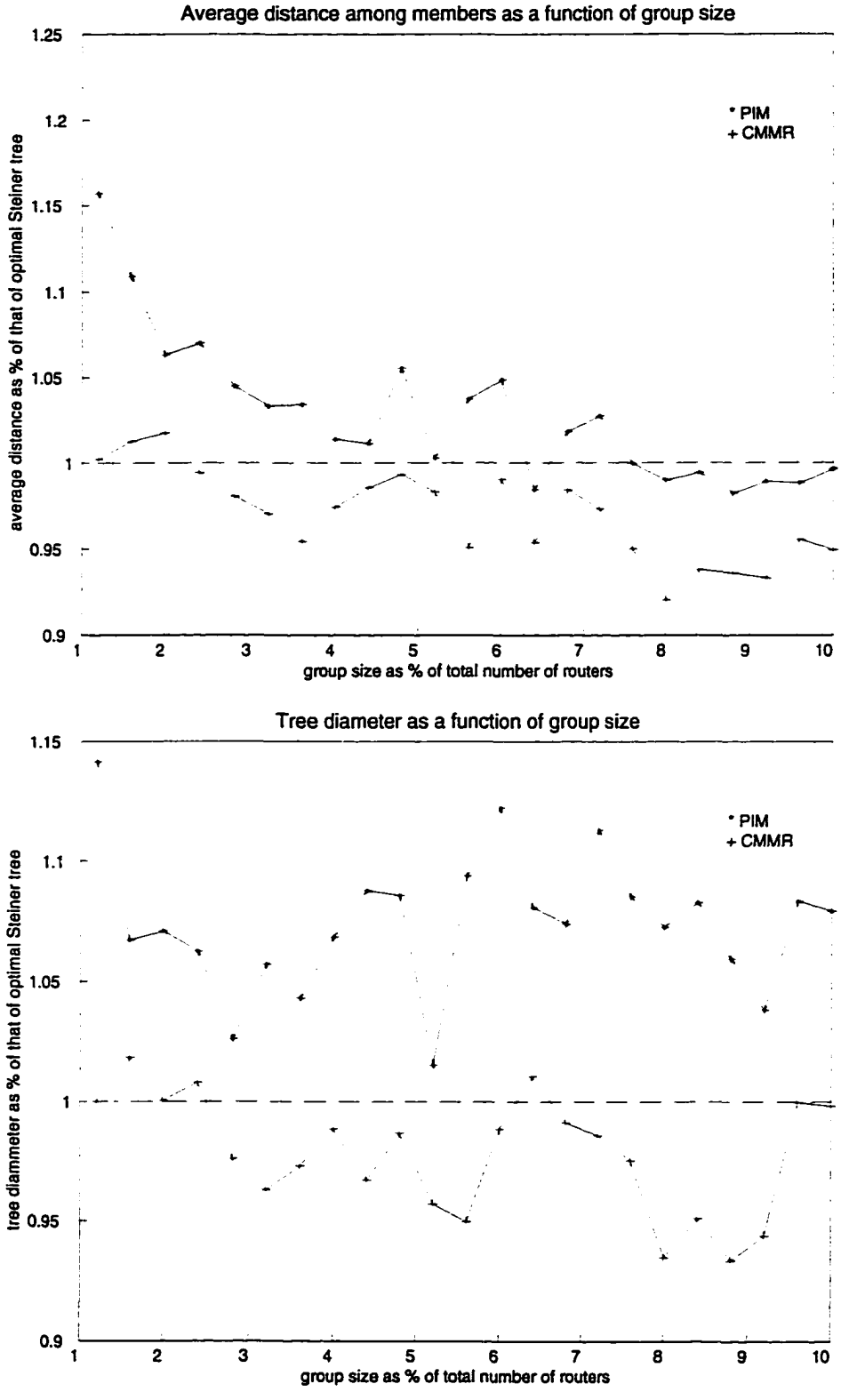


Figure 4.7 Tree distances of different group sizes.

bership. This is a real challenge with practical significance because even if an optimal Steiner tree obtained for initial members of a group, performances of the tree will inevitably degrade sooner or later as membership changes. It is to be found how CMMR behaves with dynamic membership. To this end, a sequence of join/leave events, $E = \{e_1, e_2, \ldots, e_p\}$, $e_i \in \{join, leave\}$, is used to represent the membership dynamics. While generating this event sequence, the group size is controlled at around 10% of the total number of routers in the network so as to keep the sparse nature of the multicasting. The length of the event sequence is determined in such a way that about 75% of all routers in the network are involved in the multicasting group in question. This is believed more than enough for potential participants of any group.

Figure 4.8 shows how link cost and average distance among group members change as the event sequence applied. Notice that the instantaneous group size (refer to the vertical axis at the right hand side) after each join/leave event is also plotted in the same figure to depict performance/group size relationship. As it is seen, the CMMR performs consistently well with the dynamic membership because it always tries to connect a new member to the existing tree at a nearby point. As with the case of the static membership, the PIM-SM performs poorly when the group size is small. In addition, the PIM-SM's performance experiences occasional sharp degradation with dynamic membership (see the sudden jump in PIM-SM's tree cost at event number 170). This phenomenon occurs when the core determined by the PIM-SM's hashing function is located far from the majority of the members.

4.4 Scalability Enhancement

The CMMR scheme, while inheriting advantages from PIM-SM and CBT, constructs

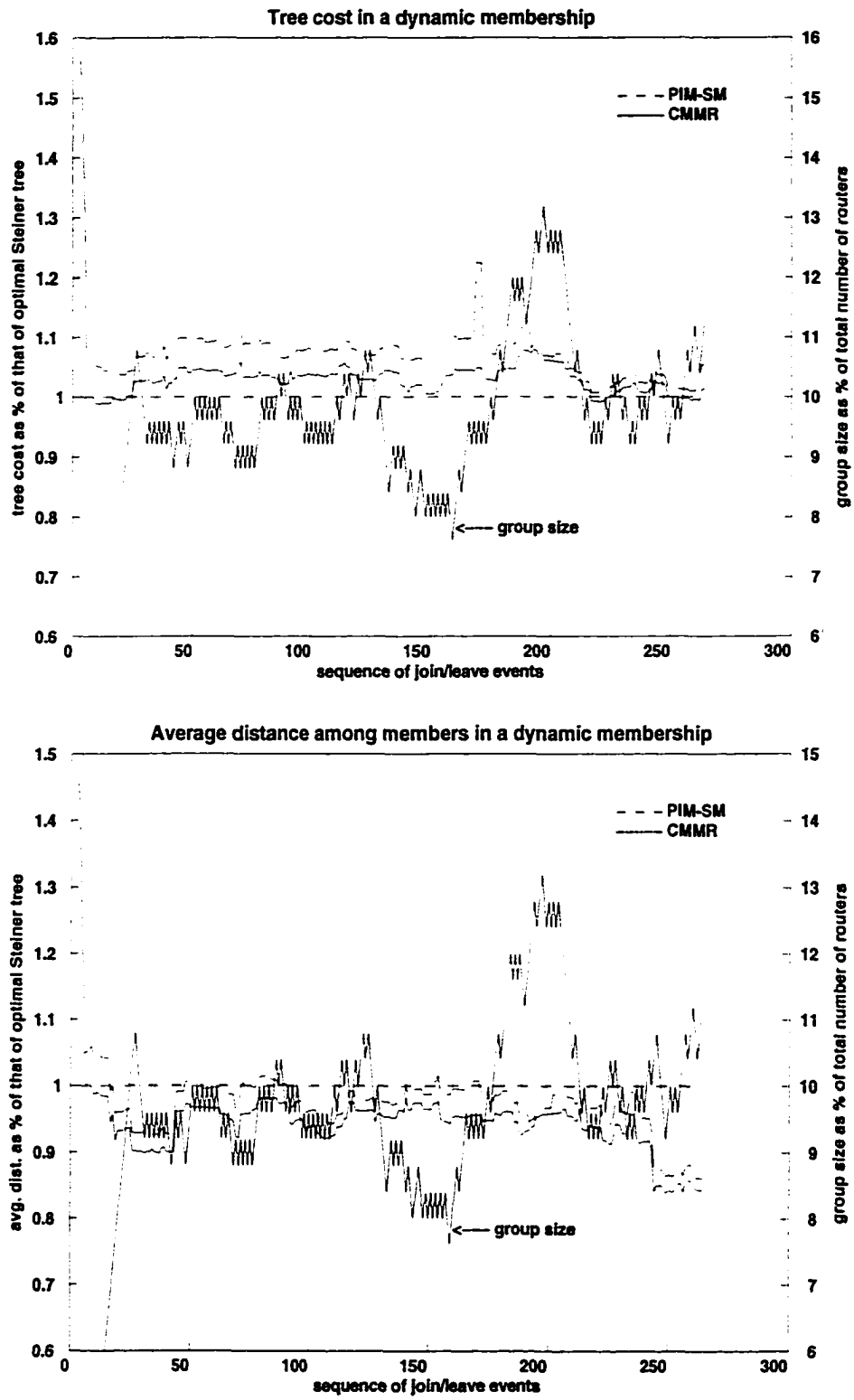


Figure 4.8 Tree properties of dynamic group membership.

delivery trees through a completely different approach. Not only does it reduce the control overhead, but it creates multicast delivery trees of lower cost, on the average, than those a hashing function creates. To enhance the scalability up to the size of global internetworks (e.g. the Internet), the hierarchical CMMR architecture, which in nature possess scalability at the expense of control overhead [55], is proposed. The basic concept of hierarchical CMMR is to configure the entire multicast region into subregions of different hierarchical levels, and administratively scope the IP multicast addresses in accordance with the hierarchy. The idea of administratively scoping multicast address space, used to be negatively discriminated by some ones, becomes increasingly favorable as the Internet community got more working knowledge with multicast support over wide areas [56]. Indeed, IPv6 has adopted the idea of multicast addresses scope in its design.

The hierarchy is constructed like this: Given a entire multicast region S^1, it can be divided into a number of subregions $S^2 = \{S_1^2, S_2^2, \ldots, S_m^2\}$ Each subregion $S_i^2 \in S^2$ can be further divided into subsubregions $S_i^3 = \{S_{i1}^3, S_{i2}^3, \ldots, S_{in}^3\}$ ... The subdivision can be carried out recursively as many as necessary so long as the divided components satisfy that

$$S_i^k = \bigcup_{\forall S_{ij}^{k+1} \in S_i^k} S_{ij}^{k+1} \tag{4.4.1}$$

Moreover, leveling along different branches of the hierarchy tree does not have to be the same. It is well allowed, take the subregions at the second hierarchical level as an example, that S_i^2 is further subdivided but S_j^2 is not. As a matter of fact, subdivisions internal to a subregion of are hidden from other subregions.

To better understand the CMMR hierarchy architecture let us look at a 2-level hierarchy example. Each subregion at the basic hierarchic level runs its own instance

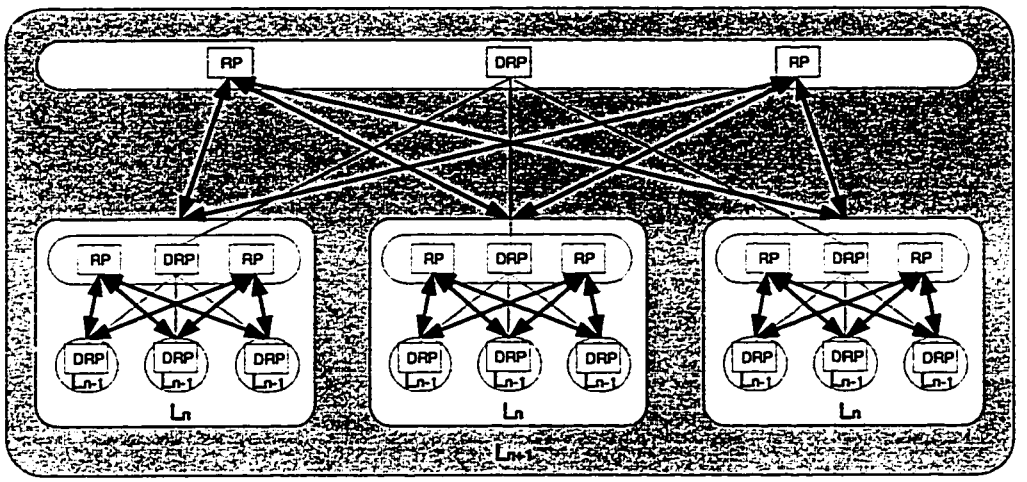


Figure 4.9 Hierarchical structure of the CMMR. Three hierarchic levels are depicted. Data flows shown as thick lines. Control flows shown as thin lines.

of CMMR with its own CM and CCs within the subregion. A CM at the basic level communicates to the CM at the higher level in order to make a group span the entire region. At the higher level, a separate set of CCs is configured. These higher level CCs register with the higher level CM, and only serve inter-subregions multicasting at the higher level CM's disposal. When implemented in real networks, the number of hierarchical levels, solely an administrative decision, can be expanded through recursive applying the hierarchical structure. Figure 4.9 depicts a more generic 3-level hierarchy.

While the overhead of coordination between hierarchies is inevitably introduced, the amount of protocol overhead charged to individual routers is well limited. While the CMMR does not restrict the number of levels in the hierarchy, the IPv6's scoping scheme (shown in Table 3.1) may be adopted for interoperability. Thus, size of a link-scope region conforms that defined by the link-layer technique. In generic, it is the size of a LAN. How to define region of site-scope and organization-scope is only an administrative issue.

Link-local scope multicast relies on the link layer technique used. Site-local scope

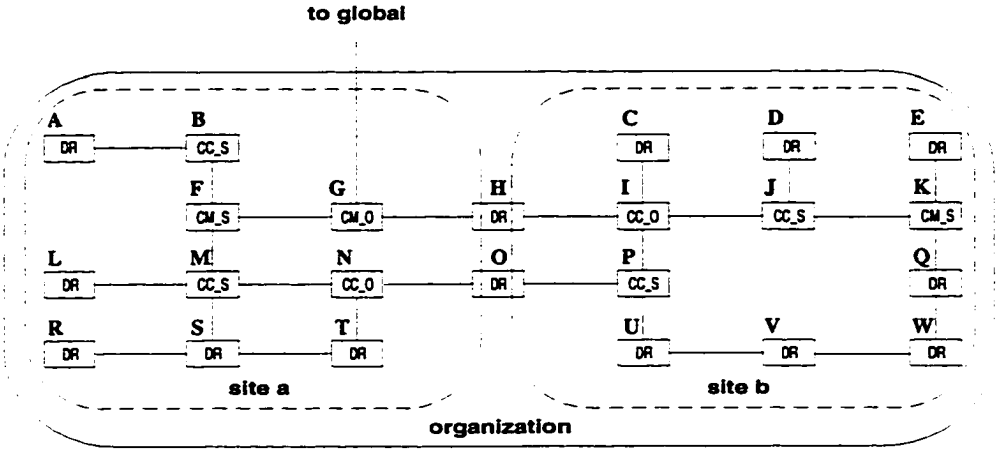


Figure 4.10 A example of a two level hierarchic CMMR region for an organization consisting of two sites.

is the basic region running the CMMR, in which one (active) CM (with one or two backups) support multicast among all embodied LANs. But, it is also allowable to have an entire organization-local scope run a single instance of CMMR. However, in this case a local video-conference, for example, between two neighboring LANs may leak on all subnetworks of the organization. Figure 4.10 shows how hierarchic CMMR might be configured in an organization with two sites, site *a* and site *b*. In this particular example, a CM_S is the CM of the site it resides in, and CC_Ss are candidate CCs for site-local groups. In site-local scope multicasting they work exactly the same way as what was described in Section 4.2. CM_O is the CM of the organization, and CC_Os are candidate CCs for organization-local groups. In the context of organization-local multicasting, the CM_O is known only to the CM_Ss, CC_Os are known only to the CM_O. When a organization-local multicast group, G_{org}, is initiated, say by a host attached to Router-R, which sends `Join` messages to Router-F (the CM_S in site *a*) regardless of the group address scope since it only knows *its own* CM_S but does not know the CM_O. Router-F will forward the `Join` messages to Router-G (the CM_O) because it finds out that the

requested group uses an organization-local address. Upon reception of the `Join` message, the CM_O will select a core from its CC_O list.

All the core selection rules as discussed in Section 4.2 are readily generalized to the hierarchical structure. For example, when a organization-local multicast group, G_{org}, is initiated by a host attached to Router-T, Router-T will send `Join` messages hop-by-hop to Router-F through Router-N. By checking the group address, Router-N recognizes that the group uses an organization-local address so it takes over the `Join` due to the En-Route-Core rule. But if the `Join` message sent by Router-T had a site-local scope address, Router-N would forward it like any other regular intermediate routers.

The significant point is that CM_O only needs to manage CC_Os, and every DR only needs to know its own CM. Therefore, multicast protocol overhead for any regular router is the same regardless of the scope.

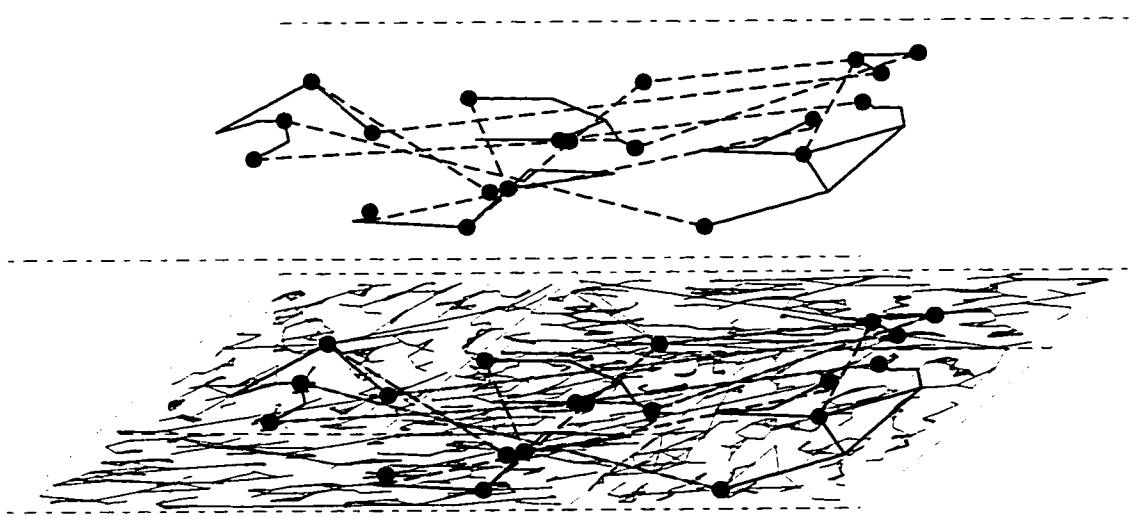


Figure 4.11 A 2-level hierarchical network. The base level is composed of 5 250-node subnetworks, and the top level consists 20 nodes, of which 4 from each subnetwork and randomly connected. The network on top represents the top level network, where the dotted lines indicate the inter-subnetwork connections.

In order to evaluate the performance in generic a series of simulations is conducted over a random 2-level hierarchical network. The base level of hierarchical network (shown in Figure 4.11) is composed of 5 250-node random networks, each of them represents a subregion of the simulated autonomous multicasting region. The higher level network interconnecting the subregions is 20-node random network shown as bold lines in the same figure. These 20 nodes, 4 from each of the 5 subregion, are literally the border routers for inter-subregion communications. So routers with relatively high degrees at each subregion are selected as the border routers.

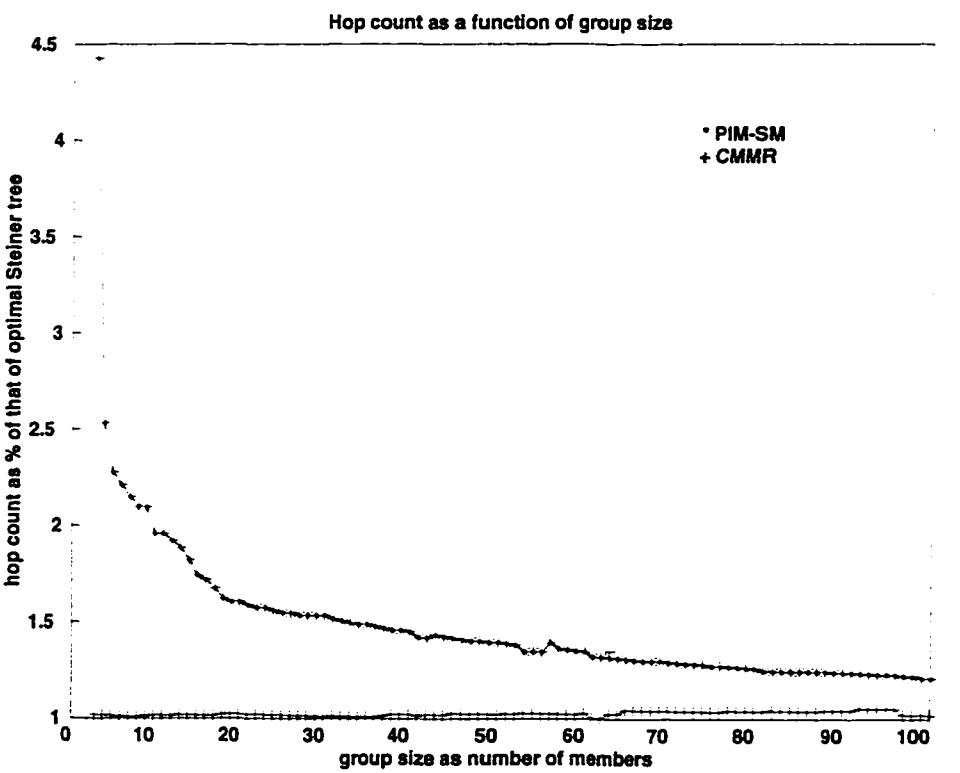


Figure 4.12 Link-count of trees in hierarchical CMMR.

Performance are again evaluated in reference to the near optimal Steiner tree heuristics presented in [54]. But the heuristics was run over the entire 1250-node region as a whole, rather than using any hierarchical structure. Besides, only inter-subregion multicast group

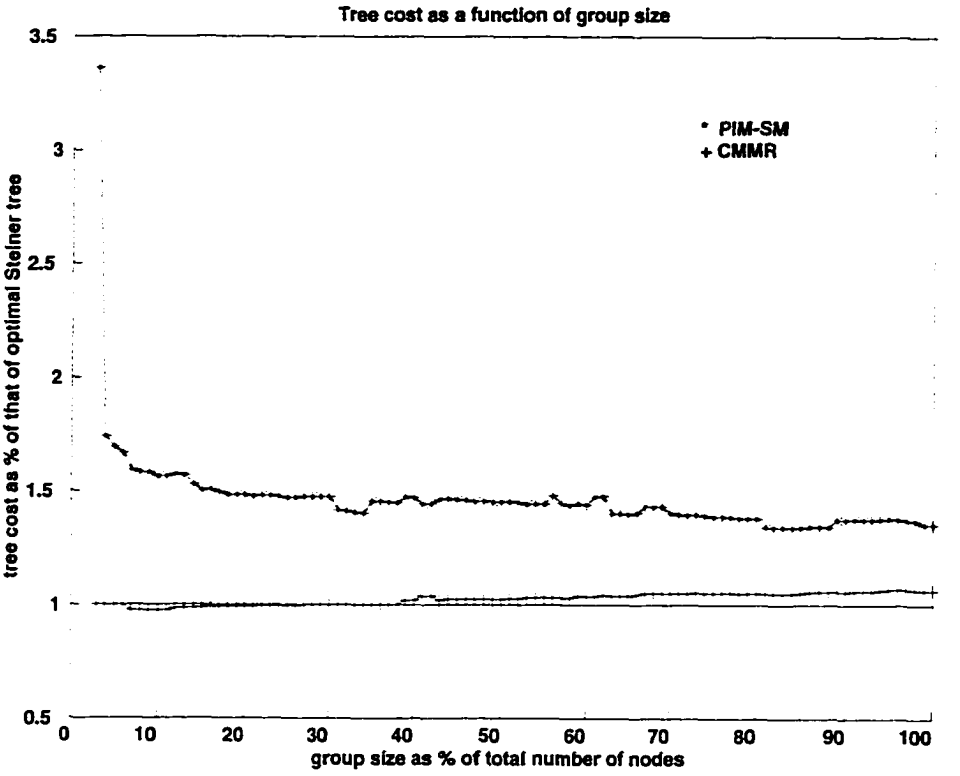


Figure 4.13 Tree costs in hierarchical CMMR.

are observed. There are 10 higher level CCs 2 in each of the 5 subregions. The higher level CCs were not selected randomly because of the relatively small number of nodes at the higher level. It is concerned that random selection in this case would easily result in that some subregions did not have any higher level CCs at all, which apparently should be avoided. Locations of the multicast group members in this simulations are purely random. Different group sizes changing from its minimal possible (3 members) to 100 members (8% of the total number of nodes in the region) are examined. The performance curves for the hierarchical CMMR shown in Figures 4.12 and 4.13 consistently showed the relationship among performance of CMMR, PIM-SM and the reference heuristics. The smoothness of the curves is due to large number of sample runs. Yet, the crucial benefit got from hierarchy but not shown in the figures is that the control overhead imposed to

individual routers is much less than that in non-hierarchy network of the same size. The curves tending smoother is due to the simulation was run over inter-subregion groups where members assumed evenly distributed across all subregions. Since the simulation was run over a larger network, and more trails were executed for each membership size, statistical relationship of performances between CMMR and PIM-SM became clearer. Once again, cost of the multicast delivery trees tends to not constantly drop as the group size grows. The reason behind it is apparent: The chances that a new member of a group is an on-tree node increase as the group size getting larger.

5. Mobile Internet Protocols

5.1 Mobility Support in IPv4

Current version of the Internet Protocol (IPv4) [7] made an implicit assumption that a node's point of attachment to the Internet remains fixed, so that a node's IP address uniquely identifies the node's point of attachment to the Internet and, consequently, is used for identifying the node. Meanwhile, datagrams are sent to a node based on the location information contained in the node's IP address, which means an IP address, as the same says, is also used for delivering datagrams.

If a node moves but still needs to retain its ability to communicate, what IPv4 can do is either change the node's IP address, or propagate node-specific routes throughout the Internet. Unfortunately, none of these alternatives is practically acceptable. The first makes it impossible for a node to maintain transport and higher-layer connections when the node changes location. The second has obvious and severe scaling problems.

The recently standardized work in Mobile IP [30] specified IPv4 enhancements that allow transparent routing of IP datagrams to mobile nodes in IP networks.

Mobile IP is intended to enable nodes to move from one IP subnetwork to another while maintain continuous transport or higher-layer connections. It is just as suitable for mobility across homogeneous media as it is for mobility across heterogeneous media. Mobile IP can be thought of as solving the *macro* mobility management problem. It is less well suited for more *micro* mobility management applications, such as handoff among wireless transceivers, each of which covers only a very small geographic area. For situations where node movement does not occur across different IP subnetworks,

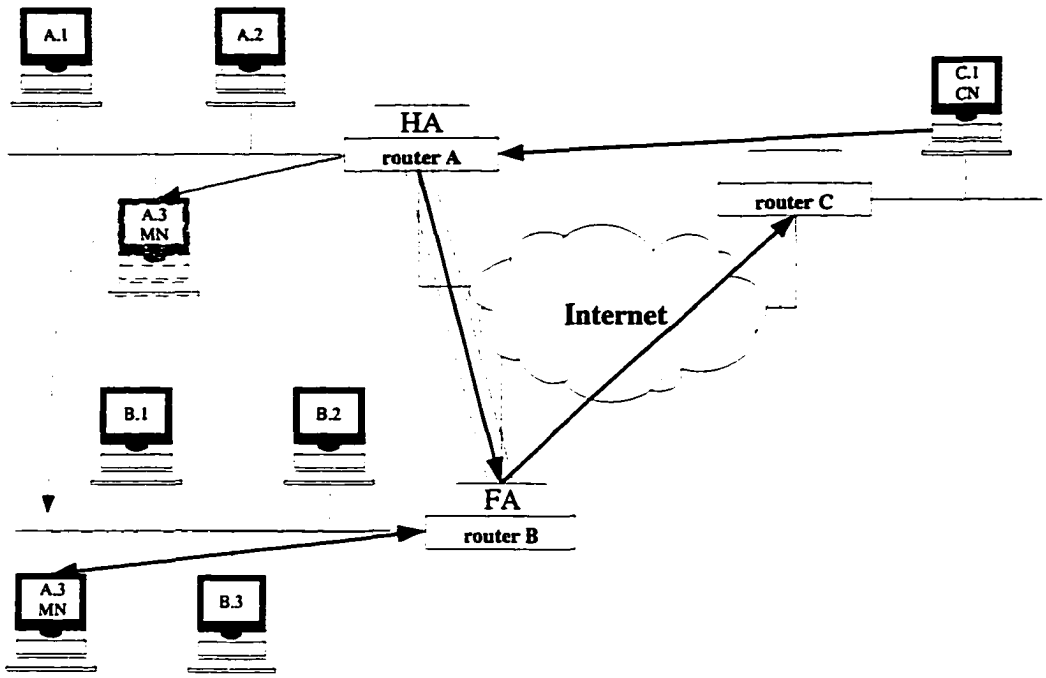


Figure 5.1 Mobile IP using tunnelling mechanism.

link-layer mechanisms for mobility (i.e., link-layer handoff) may be faster convergence and less overhead than Mobile IP.

5.1.1 Basic Version of Mobile IP

The basic version of mobile IP routing, called Mobile IPv4, works as follows (refer to Figure 5.1). While visiting a foreign network, a *mobile node* (MN) is served by the *home agent* (HA) on its home network via a care-of address indicating its current location. The association between a mobile node's home address and its care-of address is known as a *mobility binding*. The care-of address is, in many cases, the address of a *foreign agent* (FA) on the network being visited by the mobile node, which forwards arriving datagrams locally to the mobile node. Alternatively, the care-of address maybe temporarily assigned to the mobile node using Dynamic Host Configuration Protocol (DHCP) [57] or other means. All IP datagrams addressed to the mobile node are routed

by the normal IP routing mechanisms to the mobile node's home network, where they are intercepted by the mobile node's home agent, which then tunnels each datagram to the mobile node's current care-of address. Datagrams sent by a mobile node use the foreign agent as a default router but require no other special handling or routing.

Thus, Mobile IPv4 relies on protocol tunneling to deliver packets to mobile nodes that are away from their home network. The mobile node's home address is hidden from routers along the path from the home agent to the mobile node due to the encapsulation. Nodes that do not implement mobility functions, like the *correspondent node* (CN) in Figure 5.1, can communicate with mobile nodes the same way as with any other fixed nodes due to the presence of home agents.

The mobile computing environment is potentially very different from the ordinary computing environment. The tunneling feature is widely understood to be a security problem in the current Internet if not authenticated [58]. Hence, all mobile nodes and mobile agents need to perform a strong authentication mechanism known as keyed MD5 [59].

5.1.2 Mobile IP with Route Optimization

The Mobile IPv4 scales to handle a large number of mobile nodes in the Internet. While supporting transparent interoperation with mobile nodes, the Mobile IPv4 forces all datagrams for a mobile node to be routed through its home agent. Thus, datagrams to the mobile node are often routed along paths that are significantly longer than optimal. For the example shown in Figure 5.2, if a mobile node, say *A.3*, is visiting network *B*, datagrams from a correspondent node *C.1* must first be routed through the Internet to *A.3*'s home agent on *A.3*'s home network *A*, only to then can they be further tunneled to

A.3's foreign agent on the visited network *B* for delivery to *A.3*. Even if the correspondent node is on network *B*, similar procedures would have to happen. This indirect routing can significantly delay the delivery of datagrams to *A.3* and places an unnecessary burden on the networks and routers along this path through the Internet. If the correspondent node in this example is actually another mobile node, say *X.Y*, then datagrams from *A.3* to *X.Y* must likewise be routed through *X.Y*'s home agent on *X.Y*'s home network (not shown in the Figure) and back to the original subnetwork for delivery to *A.3*. Besides, without route optimization [60], the home agent is a potential bottleneck when serving many mobile nodes. Therefore, although not required by the Mobile IPv4 standard, optimization of the path from a correspondent node to a mobile node is desirable.

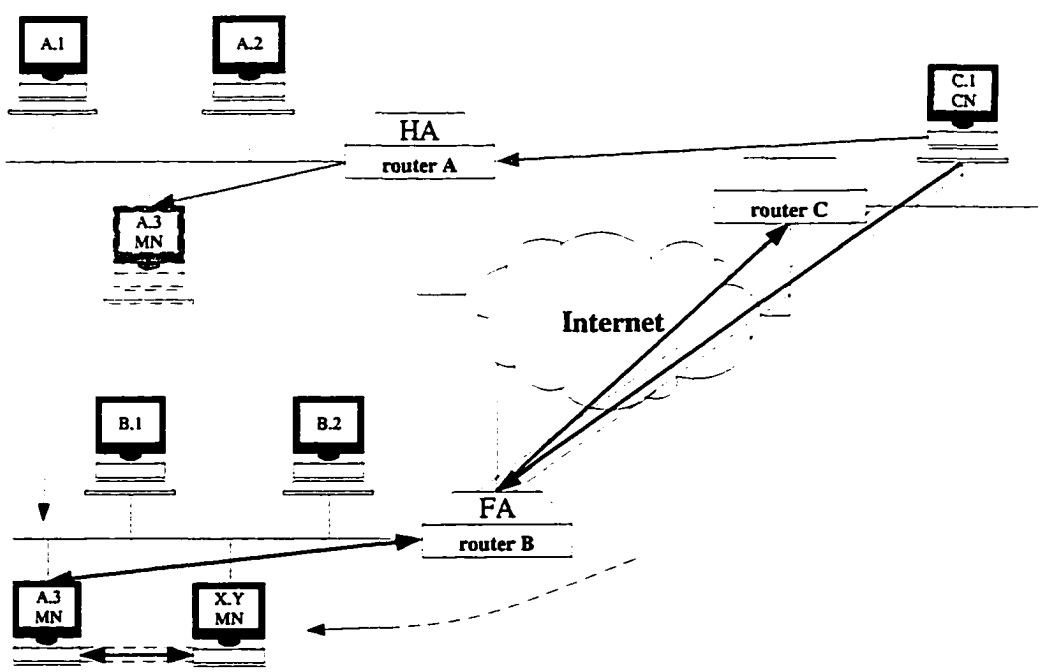


Figure 5.2 Mobile IP with routing optimization.

Extensions to the Mobile IPv4 protocol allow for the optimization of datagram routing from a correspondent node to a mobile node. These extensions provide a means for nodes

that implement them to cache the binding of a mobile node and then tunnel their own datagrams directly to the care-of address indicated in that binding, bypassing the possibly lengthy route to and from that mobile node's home agent. Extensions are also provided to allow datagrams in flight when a mobile node moves, and datagrams sent based on an out-of-date cached binding, to be forwarded directly to the mobile node's new care-of address. These extensions are collectively referred to as *route optimization.* All operation of route optimization that changes the routing of IP datagrams to the mobile node is authenticated using the same type of authentication mechanism used in the Mobile IPv4 protocol. This authentication generally relies on a mobility security association established in advance between the node sending a message and the node receiving the message that must authenticate it. When the required mobility security association has not been established, a Mobile IP implementation using route optimization operates in the same way as the Mobile IPv4 protocol.

5.2 Mobility Support in IPv6

Mobility support in IPv6 has moved in the direction of end-to-end location updates using the facilities of IPv6 to send binding updates. Communicating via the home agent is only necessary if a correspondent node does not know the current location of the mobile node or if the mobile node wants to hide its location. Although there are important details to be worked out and tested, the reduction in network load and reduction in end-to-end latency over the IPv4 solution makes Mobile IPv6 attractive for unicast applications. An issue that needs more fundamental research, however, is solutions for mobile multicast. Efficient multicast cannot be done end-to-end, since it requires network support in the form of multicast routing algorithms.

A mobile node is always addressable by its home address, whether it is currently attached to its home network or is away from home. While a mobile node is at home, packets addressed to the mobile node's home address are routed to it using conventional internet routing mechanisms in the same way as if the node were never mobile. Since the network prefix of a mobile node's home address is equal to the network prefix of its home network, packets addressed to it will be routed to its home network.

While a mobile node away from home is attached to some foreign network, it is also addressable by one or more care-of addresses, in addition to its home address. A care-of address is an IP address associated with a mobile node only while visiting a particular foreign network. The network prefix of a care-of address being used by a mobile node is equal to the network prefix of the foreign network to which the mobile node is link-level connected, and thus packets addressed to this care-of address will be routed to the mobile node's location at the visited foreign network. The association between a mobile node's home address and care-of address is known as a *binding* for the mobile node. A mobile node typically acquires its care-of address through stateless [61] or stateful [62] address auto-configuration, according to the methods of IPv6 Neighbor Discovery [41], although other methods of acquiring a care-of address are also possible.

While away from home, the mobile node registers one of its binding with a router in its home network, requesting this router to function as the home agent for the mobile node. The care-of address in this binding registered with the home agent is known as the mobile node's *primary care-of address*. The mobile node's home agent thereafter uses proxy neighbor discovery to intercept any IPv6 packets addressed to the mobile node's home address on the home network, and tunnels each intercepted packet to the

mobile node's primary care-of address. To tunnel each intercepted packet, the home agent encapsulates the packet using IPv6 encapsulation [63], addressing to the mobile node's primary care-of address.

Mobile IPv6 provides a mechanism for IPv6 nodes communicating with a mobile node to dynamically learn and cache the mobile node's binding. When sending a packet to any IPv6 destination, a node checks its cached bindings for an entry for the packet's destination address. If a cached binding for this destination address is found, the node uses an IPv6 Routing header [64] (instead of IPv6 encapsulation) to route the packet to the mobile node through the care-of address indicated in this binding. If, instead, the sending node has no cached binding for this destination address, the node sends the packet normally (with no Routing header), and the packet is subsequently intercepted and tunneled by the mobile node's home agent as described above.

6. Core-Manager based Multicast Routing Mobility Enhancement

Generally speaking, multicast routing protocols do not discriminate mobile group members at home network and those away from home. However, continuous delivery of group traffic to mobile members is not inherent to most multicast routing protocols. Given that the current mobile IP solutions (both IPv4 and IPv6) require mobile nodes not change subnetworks more frequent than once per second, the question is how to minimize sensible disruptions, if there are any, in data flows to mobile members.

Except a few [65–69] most published work, including that posted over the Internet, attacks either the multipoint routing *or* the mobility support individually. The fact is, however, special considerations are needed in order to support multipoint and mobility at the same time. Varies approaches to mobility support in a CMMR multipoint supporting infrastructure are to be observed.

Given a multicast group, its coverage is assumed to be confined to certain scope as specified by the group address. There are two types of significance that describe the mobility of hosts with respect to this scope concerning the reconnection support.

1. *No-transition*: Intra-scope movement — a member host moves from one subnetwork to another subnetwork, and both subnetworks are within the coverage of the group's scope. Disruption of inter-reachability with other member may or may not occur depending on the membership status local to the new subnetwork.
2. *Scope-transition*: Inter-scope movement — a member host moves from one subnetwork within the group's scope covered region to another subnetwork outside the

region. Disruption of service is most likely inevitable regardless of the maintenance of higher layer connections [70].

6.1 Sensing Boundary Crossings

As being seen in the next two Sections, efficient approaches to support of intra-scope mobility and inter-scope mobility are different, yet they have to interoperate with each other. Therefore, it is very important for the mobile nodes to be aware when they are crossing the boundary of the scope that is associated with a currently participated group.

For the sake of clarity of the discussion, a relative simple scenario is assumed: A multicast group, G, covers a scope, S. In addition to some fixed members, a mobile node originally joined G at its home network that is within S. While continued participation in G is required, the mobile node may be roaming within or beyong S. However, the mobile suppose to be aware when it moves out of S so that different schemes could be applied for efficiently re-attaching it back onto the multicast delivery tree under different conditions.

It is believed a good approach should take advantage of other unavoidable message exchanges like those needed by the location registration process taking place between a mobile node and its Home Agent [30]. The overhead incurred by the multicast enhancement upon the standard mobility support registration process is moderate. When a mobile node registers with its home agent after moving into a new subnetwork, it includes in the `Registration Request` message the address of multicast group, A_G, which it was participating when at the last visited subnetwork and wants to continue its participation. Thus the `Registration Request` message will register the interested multicast group(s) as well as the mobile node's current location. Each intermediate router forwarding the `Registration Request` message will be asked whether a

packet with A_G as the destination address would be forwarded between the incoming and outing interfaces of the `Registration Request` message. If every intermediate router says "yes", the mobile member is roaming within the scope of the concerned group. If a router says "no", then it must be a router on the border of S and so is called *en-route border router*. Consequently, not only is boundary crossing sensed, but also a border router can be pinpointed.

6.2 Support for No-transition Mobility

Intra-scope mobility support is relatively simple. When a member of a multicast group, G, is a mobile multicast group member host (simply referred as a *mobile member host* whenever the context is clear) and moves from one subnetwork, N_i, to another, N_j, it may or may not have to rejoin the same group depending on the membership status local to N_j. If the members of G already exist in N_j, the mobility issue becomes trivial (and hence precluded from the following discussions) since the mobile member host can receive the multicast datagrams as soon as its lower layer connection with the new subnetwork having been established. Otherwise, the mobile member host has to join G again. When rejoining is required, it is always possible for the mobile member host just moved into a new subnetwork to join G as a complete new member. In this way mobility does not bring in any specific issue to the multicasting structure. However, unlike the original joining to a group, a mobile member host (rejoining the group in which it has been participating due to its mobility) is more sensitive to the join latency because noticeable intermittent in the multicast traffic is undesirable. Hence, if the link layer migration from N_i to N_j can be done in a reasonably short time, it is meaningful to spend efforts on providing virtually continual data feed at the network layer. Otherwise,

if the link layer migration takes long, the data flow interruption would be noticeable anyway. Notice that a quick migration is not meant it has to be taken place between adjacent nodes. In order to recover from service disruption caused by roaming, a mobile member host can do better than rejoining. If the mobile member host can send out an `Unsolicited Report` [71] immediately after being connected to N_j, the N_j's DR will start joining G right away. But still that the N_j's DR joins the delivery tree through the regular join procedure does not sound like the best way to reconnect the mobile member host back to G. Thus, it is worthwhile to look for some possible shortcuts.

The essence of any shortcuts is to have the mobile member host retain some knowledge about G's delivery tree, and pass it on to its new DR after each migration. This knowledge should be sufficient enough for the new DR to skip the core inquiry steps and join the tree at a known point directly while be simple enough so as to minimize the introduced overhead. Now the question becomes what information is most efficient to pass. An intuitive thought is to run the mobile binding between the mobile member host and a core. One might let the DR of a subnetwork where the mobile member host first joined the group memorize the core to whom it has intended to join and convey this piece of information to the mobile member host. All of these exchanges can be fulfilled as additional payload to the IGMP messages. If a mobile member host, having joined G in N_i whose selected core were C_i, moved into N_j and demanded retaining its participation in G, it might furnish the N_j's DR with the identity of C_i. As a consequence, N_j's DR could join G by sending a `Join` message directly to C_i.

Unfortunately, this simple and easy approach is not always available. Since the association between a DR and its selected core is significant only when the DR is joining

the group and vanishes after that. It is very well possible that a DR stays in the group longer than its selected core but is not aware when the core left. In other words, the information carried by the mobile member host regarding the selected core could be out-of-date anytime. Attempt of joining such a out-of-date core would fail the mobile member host from being reconnected to the group's delivery tree.

In light of the above observation, a more reliable way to expedite the rejoining process would be taking advantage of the so called *residual membership*, which refers to the fact that the DR of a subnetwork with local members to a multicast group remains on the multicast delivery tree for certain time after the last member leaves the group. Since it is always true that the N_i's DR is on G's delivery tree when the mobile member host is leaving, the knowledge about this DR becomes the very appropriate information to be passed to N_j's DR. The mobile member host can include the identity of the N_i's DR in its `Unsolicited Report` so that the N_j's DR can send a `Join` directly to the N_i's DR. One may be worried about that the residual membership at N_i would not last long enough when the mobile member host itself happens to be the last member of G in N_i such that the rejoining could fail if the N_i's DR had pruned itself off G's delivery tree before the `Join` message from N_j arrived. This is an understandable but over prudent concern. The chance for a event like this to happen is extremely rare and can be easily precluded by carefully engineering the protocol. As a matter of fact, the IGMP standard requires the amount of time that must pass before a DR decides there are no more members of a group on its affiliated subnetwork — the *pruning delay* — be the `Group Membership Interval` , whose default value is as long as 260 seconds (more than 4 minutes!) [71]. This is much much more than sufficient for most types of mobile

member host migration that may require continued feed of multicast data flows, otherwise it is not rational for the mobile member host to ask for continual data feed in the first place. Even if for any reason, the DR in question sets its `Group Membership Interval` less than the default, the actual duration of the residual membership will be longer than the `Group Membership Interval` since a mobile member host is unaware of in advance when it is going to migrate, so it will normally not send a `Leave Group` message before it leaves a subnetwork. Furthermore, if the DR is a non-leaf node or the only core on G's delivery tree, it will not be pruning itself at all. Nonetheless, for the sake of robustness any implementation of the protocol should have a way to properly handle a short-lived residual membership. For example, set a timer to start the regular join routine should the `Join` message missed the residual membership.

6.3 Support for Scope-Transition Mobility

A more challenging than missing the residual membership issue is the potential restrictions on the mobility support coverage due to the hierarchical CMMR structure. Assume a multicast group G uses a site scope address which covers a entire subregion S_i. A mobile member host joined G at a subnetwork within S_i. Later on, the mobile member host moves out of S_i into a peer subregion S_j but still needs to continually participate in G. In this case, the residual membership method would not work because of the hierarchical CMMR structure and its multicast address scoping scheme. This depicts a scenario where scope-transition mobility support is needed.

While a solution may work equally well to serve either purpose, it is preferred to distinguish *scope augmentation* from *out-of-scope support* for better focusing while elaborating the solution. Scope augmentation refers the situation where a group is initiated

with certain scope and later for whatever reasons needs to include members beyond the original scope. On the other hand, the out-of-scope support targets the scope-transition mobility of mobile member hosts.

The most straight forward way of providing out-of-scope support is scope-upgrade by which a completely new group with a larger scope is initiated. But this approach requires every group member to leave the current group and join the new one, causing disruption in service to all members. It is obviously inefficient and also disturbing to ask all member hosts to change group just because as few as a single member moves out of the scope. A good mobility support solution should be transparent to other (fixed) members as much as possible.

Another idea to accomplish out-of-scope support relies on a concept called *virtual group*. A new group at a higher hierarchical level, G_h, whose scope covers both S_i and S_j is initiated when the mobile member host moves there. Members of G_h are (restricted to) the DR of the subnetwork being currently visited by the mobile member host in S_j and a border router (e. g. the en-route border router) in S_i. The border router hence has to be in both groups G and G_h, and realizes that these two groups are virtually a single group. At this border router where multicast delivery trees of these two groups meet, datagrams of one group must be forwarded to the other. It is clear that the virtual group method could create near optimal delivery path in supporting mobile member hosts (see Section 4.3). However, it is not flexible to support highly random roaming which may claim frequent/recursive applying of the virtual group mechanism, and also not scalable to support large number of mobile member hosts roaming across many different subregions. That is why the less efficient (as far as delivery path concerned) tunneling approach to

out-of-scope support can be more generic.

Because of the mechanism to be described below, the tunneling approach is named as *proxy member tunneling*. In this approach, a mobile member of group G away from home subnetwork always communicates with other group members via its home agent if the visited subnetwork is out of S_i. The home agent joins G that the mobile member host substantially participates in the same way as for any local (fixed) members. Datagrams addressed to G being received in their native form (not encapsulated) are encapsulated and tunneled to the mobile member host. When the mobile member host needs to send to the group, datagrams addressed to G are encapsulated and sent to its home agent, the latter in turn decapsulates the datagrams and forwards them to G's delivery tree. Since the encapsulated datagrams are treated as unicast ones by the network, their reachability to any part of the internetwork is not restricted by the group scope. Although inefficient as far as the delivery path concerned, this tunneling approach should not be overlooked due to its advantages such as transparent to other group members and minimum functionality requirements imposed upon the mobile member host. The standard mobility support has made a mobile member host capable of handling (de/en)capsulations. When the proxy member tunneling is utilized, special cares should be given to a situation in which a $DR_{current}$ is being visited by more than one mobile members of the same multicast group supported by different home agents [72]. In this case, one tunnel would be sufficient to support all the mobile members. Hence, the $DR_{current}$ is respossible to coordinate so that only one tunnel is built.

Support for dynamic-scope can be achieved by a hybrid of the residual membership mechanism and the proxy member tunneling. When a mobile member host is roaming

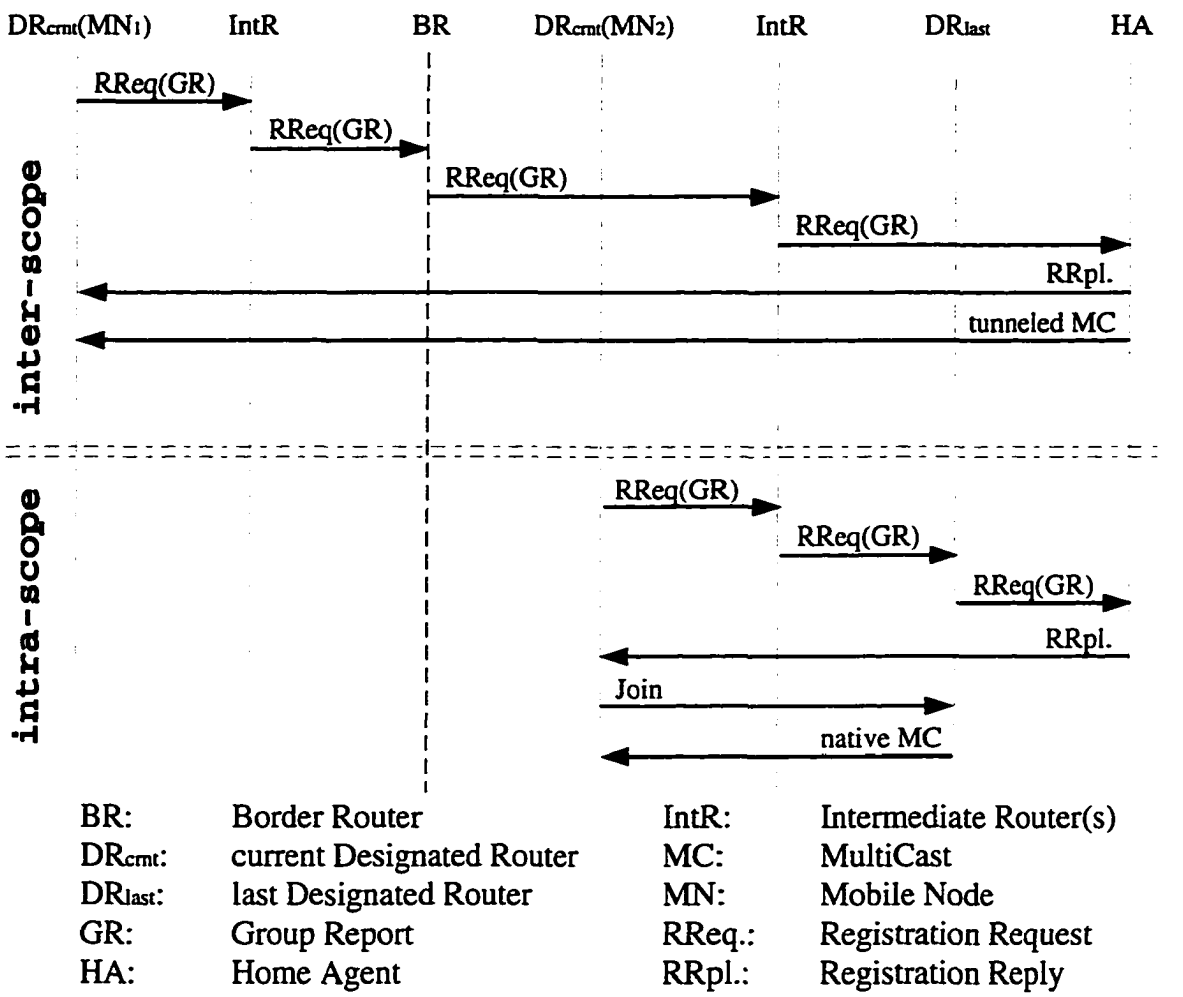


Figure 6.1 Message exchanges in supporting for mobile multicast group member in the CMMR.

around within the scope of the concerned group, residual membership procedure is followed; when the mobile goes out of the scope, the proxy member tunneling is used. Figure 6.1 depicts timing of message exchanges in a generic scenario.

Although not fitting well for out-of-scope support, the virtual group is a practical approach to scope augmentation. In generic, a scenario where scope augmentation applies can be illustrated as following. Assume N groups, $G_1, G_2, \ldots, G_N$, called *elemental groups* with scopes $S_1, S_2, \ldots, S_N$ respectively, and scopes of any two groups may or may not be at the same hierarchical level. For whatever reasons, if the N elemental

groups have to merge into one, in other words, members of the elemental groups need to communicate one another, the virtual group scheme is ideal in fulfilling it without disrupting service to most of their members. A new group G_h at a higher hierarchical lever whose scope satisfies

$$\bigcup_{i \in \{N\}} S_i \subseteq S_h \tag{6.3.1}$$

will be used to interconnect the elemental groups. Each elemental group needs to elect a *representative node*, which must be on the elemental group's delivery tree for obvious reasons, to handle the join/leave with respect to G_h. Once a representative node R_i joined G_h on behalf of its elemental group G_i, the R_i is responsible for relaying multicast traffics back and forth between G_h and G_i. Who is responsible to initiate the merge and how a representative node is elected by each elemental group are totally at the disposal of higher layer applications.

An alternative approach to the tunneling implementation of the virtual group is forward state mapping, which requires all on-tree routers map forwarding state of G_i to that of G_h. In so doing, service disruptions are expected to be minimized.

6.4 Performance Analysis

Two aspects of the performance are to be examined: control message overhead and changes in shared-tree cost.

6.4.1 Cost of No-transition Mobility Support

As far as the shared-tree cost concerned, the effect made by the mobility of member nodes is equivalent to that due to the dynamic group membership, which has been thoroughly studied in Section 4.3.3. Indeed, when a mobile member moves from a

subnetwork served by DR_{last} to one served by $DR_{current}$, the shared-tree changes as if DR_{last} prunes itself off the tree (if there is no other member left in its subnetwork) and $DR_{current}$ joins itself onto the tree (if there was not any member in its subnetwork before the mobile node moves in). Even though it maybe subtle, different re-attachment procedures affect the cost of shared-trees differently.

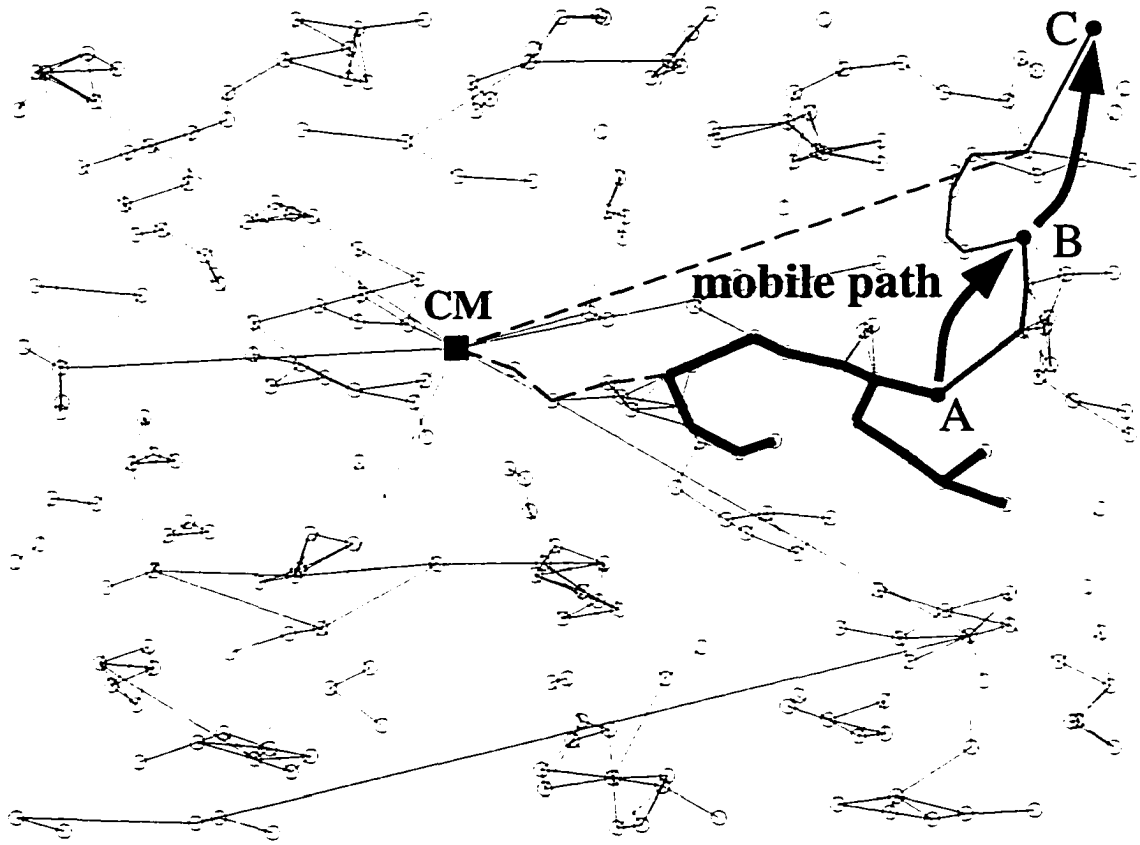


Figure 6.2 Re-attaching a mobile member to the shared-tree.

In no-transition mobility support, two procedures, namely joining as a new and residual membership, are available for re-attaching a mobile member to the shared-tree. Joining as a new ensures a best shared-tree cost that the CMMR can achieve, at a price of relatively higher control message overhead. The $DR_{current}$ has to send a

`Core_Inquiry` message to CM, then the CM (or an on-tree router encountered by the `Core_Inquiry` message) sends a `Core_Response` message back to the $DR_{current}$, and then the $DR_{current}$ sends a `Join` message to the selected Core. On the other hand, it costs only one message for the residual membership procedure to reconnet the mobile member, that is a `Join` message from the $DR_{current}$ to the DR_{last}. In so doing, the shared-tree is actually stretched along the direction pointed by the mobile member's movement. Therefore, the cost of the new shared-tree is not guaranteed optimal. Figure 6.2 shows such an example. The most-heavy bold lines represent an existing shared-tree, and a mobile member moves from the subnetwork served by Router-A to that by Router-B, then to that by Router-C. The second-heavy bold lines (connecting Router-A and Router-B) represent the stretched portion of the shared-tree after first move created by residual membership procedure. When join-as-a-new procedure is taken after the first move, it is easy to verify that the new shared-tree would look the same regardless where the selected Core is (of course it must be somewhere on the tree). The least-heavy bold lines represent the residual membership created stretched portion of the shared-tree after the second move, and the dashed lines represent the additional portion of the new shared-tree if join-as-a-new procedure is taken after the second move. The additional cost for serving the mobile member, in terms of hop-count, of the residual membership stretched shared-tree is 9, while that of the join-as-a-new created shared-tree is only 6. Hence, the residual membership approach trades off delivery cost for shorter reconnection latency. However, this trade off is quite limited, especially when the group size is getting larger. As a matter of fact, the larger a multicast group whose members randomly scattering the network, the less the number of moves it takes before the mobile member runs into a

subnetwork which is already on the shared-tree. The stretching processe would not keep going unless the mobile member is moving deep into a portion of the network where no other group members exist at all. Actually, it will restart again from the "hitting point" after each time run into the shared-tree.

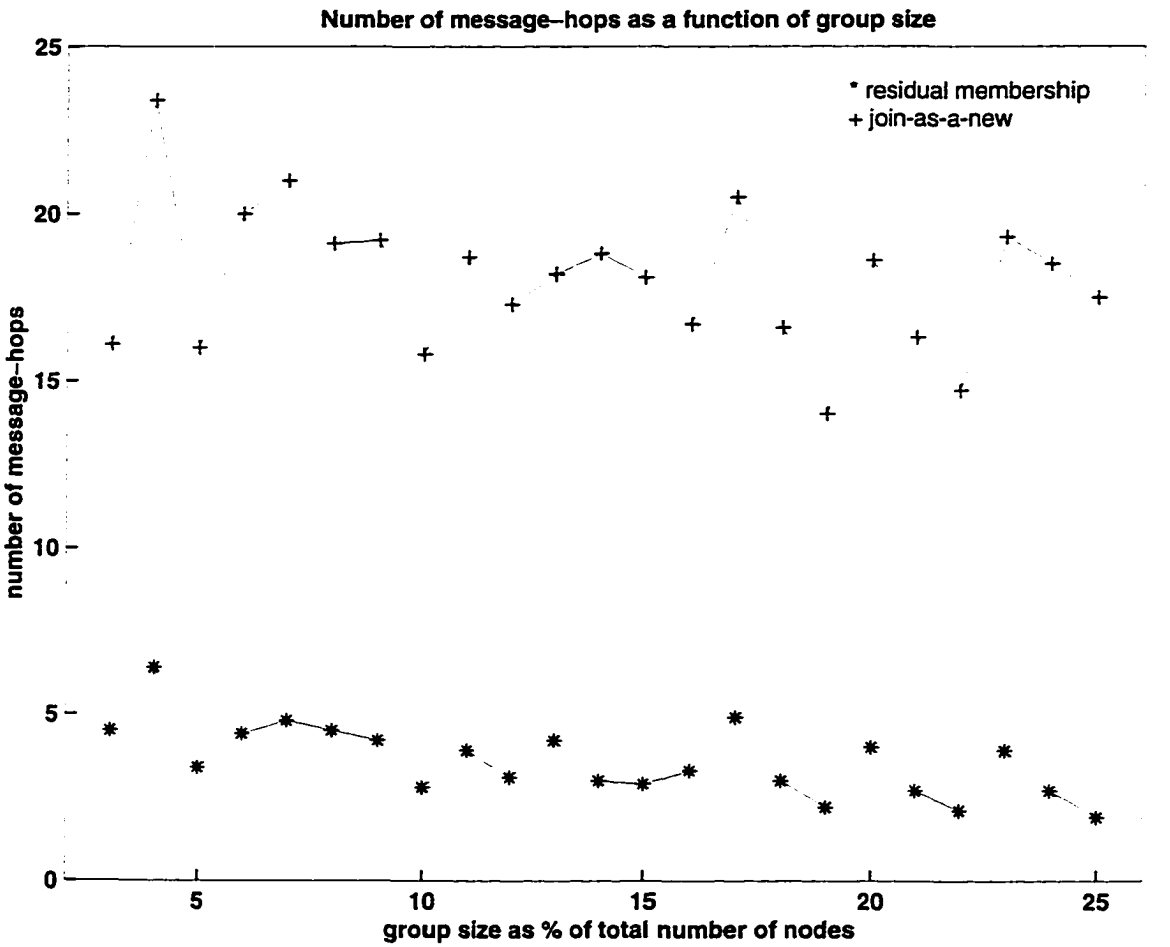


Figure 6.3 Control message overhead in intra-scope mobility support.

Figure 6.3 shows some simulation results of control message overhead in intra-scope mobility support, which is measured by

$$\sum_{i} |M_i| \times |H_i| \qquad \begin{matrix} M_i \in \{control\ messages\} \\ H_i \in \{M_i\ traversed\ hops\} \end{matrix} \tag{6.4.1}$$

a summation of the number of hops that each control message traverses in terms of *message-hop* counts. The simulation is conducted over the same series of 250-node random networks that have been used in simulations study of cost of multicast delivery trees. Control message overhead was examined with respect to multicast groups of different sizes which changes from 3 nodes up to 10% of the total number of nodes in the network. (See Section 4.3.3 for details). All trials simulates one step movement which is a result of a randomly selected member roams from its currently location (which is on the shared-tree) to a random new node (which is not a member node but may or may not be on the shared-tree). The two curves in Figure 6.3 show control message overhead of two approaches to intra-scope mobility support, namely join-as-a-new and residual membership, respectively. While both curves tend to go down slightly as the group size increases, it is obvious that residual membership procedure costs much less than join-as-a-new does. On the average, cost of residual membership is only about a quater of that of join-as-a-new.

6.4.2 Cost of Scope-Transition Mobility Support

Three procedures of scope-transition mobility support in a hierarchical CMMR autonomous multicast region have been discussed. Similar to the no-transition case, cost of the multicast delivery trees would not be affected by the fact that a group member is mobile. However, delivery costs resulted from different re-attaching procedures are normally different yet intuitive. For example, tunneling approach is least efficient. Therefore, the following performance studies will be concentrated on control message overhead only.

Overhead of the scope augmentation approach to out-of-scope support is pretty large

due to its essence that terminates the previous group with a small scope and originate a new one with a larger scope. But when this procedure being implemented, it is not necessary to entirely abandon the previous group. Rather, the existing shared-tree can be upgraded to become part of the new group's shared-tree. One of many possible easy ways of doing this is to let an authorized on-tree-router join the new group first, then distribute a forwarding state upgrade message along the existing shared-tree (of the previous group). Even in this way, the involved overhead is still considerably large, especially it grows proportionally along with the size of the previous group. Fortunately, at this price a relative efficient delivery tree can be obtained.

Figure 6.4 shows simulation results of control message overhead in scope-transition mobility support in terms of message-hops.

Virtual group procedure claims much less control message overhead, which actually is the result of trading off delivery efficiency for less control message overhea. In this approach, only the $DR_{current}(s)$ of mobile member(s) and a limited number of nodes, i. e. the border router(s) leading to those $DR_{current}(s)$ are involved in joining a new group. Consequently, the control message overhead is pretty much depending on how far the mobile members are away from the scope of the original group, and fairly independent of the group size. This is clearly shown in Figure 6.4. While the virtual group approach in most case can provide the same quality of delivery trees as the scope augmentation approach does, it does require additional functionality at the border router(s).

Further reduce in control message overhead can be achieved through the proxy member tunneling approach. In this case, there is no need to pinpoint a border router. Rather the Home Agent will provide multicast feed right after receiving the `Unsolicited`

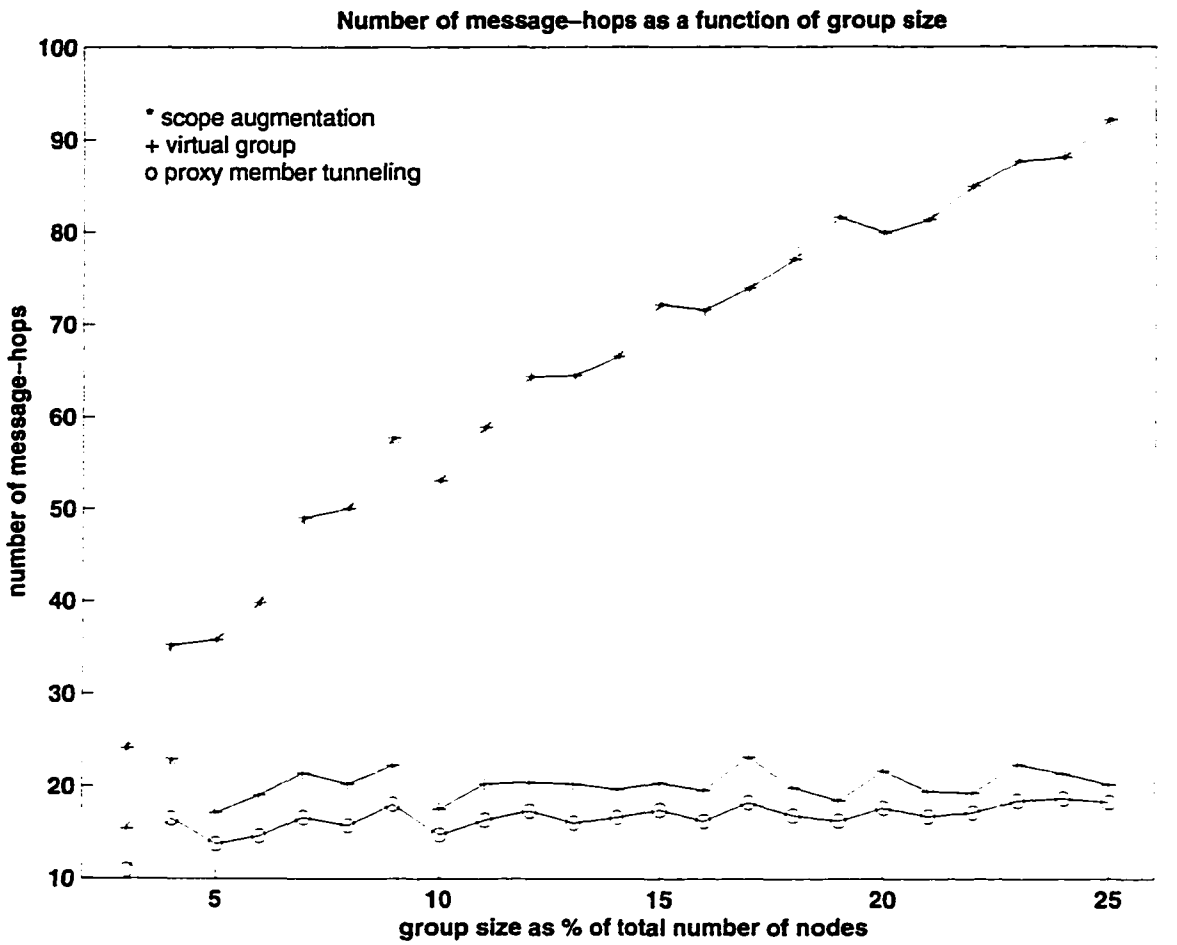


Figure 6.4 Control message overhead in scope-transition mobility support.

`Group Report` piggybacked by the `Registration Request` message, should boundary crossing has been sensed. The price for this light control message overhead is obviously the higher multicast message delivery cost.

By this point, it becomes clearer why dynamic-scope mobility support chooses a hybred of residual membership and proxy member tunneling, because they are the most efficient procedures in no-transition mobility support and scope-transition mobility support, respectively.

7. Conclusions and Future Work

The driven force of network development has been expanding from computers only to including many other areas such as entertainment and device controls. Nomadic computing devices will soon become ubiquitous, of which a key characteristic is that they will be networked. The growth of the Internet and countless other smaller internetworks which are not connected to the Internet yet constantly arise challenges to the masterminds behind the Internet. To cope with the ever explosive growth of the Internet, the next generation internet protocol — IPv6, and associated routing and controlling protocol suite are being developed. Therein a lot of efforts are spend on the M3Q issues. The reported research has the same focus with emphasis on multipoint and mobility support in large internetworks.

The adaptive multimedia synchronization algorithm is an efficient, yet simple, means of QoS support at application layer. Even with the presence of recently developed network layer QoS supporting protocols, its share of contribution to the end-to-end QoS insurance is not diminished.

The proposed CMMR outperforms, in many aspects, existing multicast routing protocols. Together with mobility support, it becomes a strong candidate for tomorrow's scalable multicast routing scheme. Continuing work on this direction may concentrate on mobility support to mobile subnetworks and combining QoS support into the CMMR structure.

The demands for mobility support to mobile subnetworks is well justified in reality. Imagine a transport platform of reasonable large size with on board networks in a large internetwork environment. A mobile node in this case is a router, which is responsible for

the mobility of one or more entire subnetworks moving together. The nodes connected to a subnetworks served by the mobile router may themselves be fixed nodes or mobile nodes or routers, and are normally called mobile subnetworks.

Typical routing to a mobile node via a mobile router is the case in which the mobile node is mobile with respect to the subnetwork, which itself is also mobile. While the residual membership method serves well for the no-transition mobility support to mobile subnetworks, special attention must be paid to out-of-scope support when the moving entities are mobile subnetworks. Implementing a naive merge of *recursive tunneling* [30] and proxy member tunneling would be quite inefficient. It may well be the case that the mobile subnetwork's home network, the mobile node's home network, and the current location of the mobile node (and its visiting subnetwork) are well separated from one another. It more eagerly needs for dynamic mobile binding with optimal routing.

The whole idea of dynamic mobile binding with optimal routing is to connect the mobile member host to the group's multicast delivery tree through a (near) shortest path. When there is only one gateway from the group covered scope to the external world, the matter is simple: let this gateway join the group on behalf of all out-of-scope mobile member hosts, and the latter perform mobile binding with the former as they roam around. But when several gateways available, a smart way of dynamic choosing the nearest gateway is appreciated.

A potential approach to improved performance in this case might be through multiple home agents so that it is possible to optimize among available home agents with respect to multiple gateways when the proxy tunneling is needed.

Appendix A. Definitions of Terms, Notations and Expressions

A *graph* $G(V, E)$ is a structure that consists of a set of *vertices* $V = \{v_1, v_2, \ldots\}$ and s set of *edges* $E = \{e_1, e_2, \ldots\}$; each edge e is *incident* to the elements of an unordered pair of vertices $\{u, v\}$. An edge e connecting two vertices $\{u, v\}$ is denoted as

$$e_{\{u,v\}} \tag{A.1}$$

A *cost* function $c(e) \geq 0$ is defined on each edge. $c(e_{\{u,v\}}) = \infty$ implies that there is no edge between vertices u and v. The cost of a graph is $C(G) = C(V, E) = \sum_{e \in E} c(e)$.

The *degree* of a vertex, $deg(v)$, is the number of times v is used as an endpoint of the edges.

A *path* is a sequence of edges $e_1, e_2, \ldots, e_p$ such that e_{i-1} and e_i have a common endpoint, and e_i and e_{i+1} have a common endpoint, $\forall i \in \{2, 3, \ldots, p-1\}$. The shortest path between a pair of vertices, $\{u, v\}$, in graph G is denoted as $sp_G(u, v) = \{E_{sp(u,v)}\}$, $E_{sp(u,v)} \subseteq E$, and may be, for simplicity, referred as $sp(u, v)$ whenever the context is clear.

If both $V = \{v_1, v_2, \ldots, v_m\}$ and $E = \{e_1, e_2, \ldots, e_n\}$ are finite, then G is finite. The number of elements in a set is denoted as $|V| = |\{v_1, v_2, \ldots, v_m\}| = m$.

A *tree* is a circuit-free and connected graph.

A tree spans a given subset M of the vertices V is denoted as

$$T(M) = (V_T, E_T), \quad M \subseteq V_T \subseteq V \tag{A.2}$$

The *cost of the tree*, $T(M) = (V_T, E_T)$, is the summation of costs of all its edges

$$C(T) = C(V_T, E_T) = \sum_{\forall e \in E_T} c(e) \tag{A.3}$$

The *distance* between two vertices u and v is the cost of the shortest path between them, $dist(u,v) = C(sp(u,v)) = \sum_{e \in E_{sp(u,v)}} c(e)$. Likewise, the distance between a vertex u and a tree $T = (V_T, E_T)$ is the cost of the shortest path between the vertex and any on-tree vertex, $dis(u,T) = \min_{\forall e \in E_T} dis(u,e)$.

When discussing networks modeled by graphs, some synonyms used in place of the terms defined above are graph — network; vertex — node, router; edge — link.

Appendix B. A Near Optimal Solution to the Steiner Tree Problem

The Steiner tree problem in graphs is a generalization of the minimum spanning tree problem. The Steiner tree problem is described as: *Given a connected graph* $G(V, E)$, *a subset of vertices* $M \subseteq V$, *and a* cost *function* $c(e) \geq 0$ *defined on edges. Find a tree* $T_s(W, F)$ *in* G, *such that* $M \subseteq W \subseteq V$, $F \subseteq E$ *and* $\sum_{e \in F} c(e)$ *is minimum.* The vertices $S = W - M$ are called *Steiner vertices.*

Previous reported work [48, 54, 49, 47] on heuristic solutions of the minimum Steiner tree problem generally took one of the two following approaches. In the first approach, likely Steiner vertices are selected and then a minimum spanning tree of these vertices, together with those in M, is constructed. In the second approach, a minimum spanning tree of the whole graph G is pruned to remove superfluous vertices and then perturbed to try to include more Steiner vertices that may reduce the overall cost of the spanning tree. The original version of the latter approach is known as KMB heuristics after names of its developers: Kou, Markowsky and Berman [73]. The reference algorithm used in Chapter 4 is an improved version of KMB, whose competitiveness bound is

$$2(1 - 1/L) \tag{B.4}$$

where L represents the number of leaves in the optimal Steiner tree.

The algorithm consists of four parts:

1. Construct the forest F of disjoint trees out of G.
2. Compute all the shortest paths between every pair of distinct Steiner vertices. Construct the connected undirected distance graph $G_1 = (V_1, E_1, d_1)$ in such a way

that $V_1 = M$ and for every edge $\{v_i, v_j\} \in E_1$, $d_1(\{v_1, v_j\})$ is the length of the shortest path in G.

3. Compute the minimum spanning tree T_{near} of G_1.
4. Construct a Steiner tree T_s, as a subgraph of G, by replacing every edge in T_{near} by its corresponding shortest path in G.

Appendix C. Abbreviations and Acronyms

CBT: core based tree

CC: candidate core

CCM: candidate core-manager

CIDR: classless inter-domain routing

CM: core-manager

CMMR: core-manager based multicast routing

CN: correspondent node

CRP: candidate rendezvous point

DHCP: dynamic host configuration protocol

DHCPv6: dynamic host configuration protocol for the IPv6

DIS: distributed interactive simulation

DR: designated router

DVMRP: distance vector multicast routing protocol

FA: foreign agent

FN: foreign network

HA: home agent

HN: home network

ICMP: internet control message protocol

ICMPv6: internet control message protocol for the IPv6

IDMR: inter-domain multicast routing

IDRP: inter-domain routing protocol

IETF: the Internet engineering task force

IGMP: internet group management protocol

IGP: interior gateway protocol

IP: internet protocol

IPng: internet protocol next generation

IPv4: internet protocol version 4

IPv6: internet protocol version 6

LANE: LAN emulation

MAC: medium access control

MBone: multicast backbone

MN: mobile node

MOSPF: multicast extensions for open shortest path first protocol

MST: minimum spanning tree

OSPF: open shortest path first

PBC: playback clock

PIM-DM: protocol independent multicast — dense mode

PIM-SM: protocol independent multicast — sparse mode

QoS: quality of service

RFC: request for comments

RIP: routing information protocol

RIPng: routing information protocol next generation

RP: rendezvous point

RPB: reverse path broadcasting

RPM: reverse path multicasting

SN: stationary node

TCP: transmission control protocol

TRPB: truncated reverse path broadcasting

TTL: time to live

UDP: user datagram protocol

References

[1] B. Rajagopalan, H. Sandick, and E. Crawley, "A Framework for QoS-based Routing in the Internet", *Work in progress*, July, 1997.

[2] L. Zhang, "Internet Support for Multimedia Applications", in *Multimedia Networking Symposium*, New York, October, 1994.

[3] C. Liu, Y. Xie, M. Lee, and T. Saadawi, "Multipoint Multimedia Teleconference System with Adaptive Synchronization", *IEEE Journal on Selected Areas Communications*, vol. 14, no. 7, pp. 1422–1435, September, 1996.

[4] T. Saadawi and M. Ammar, *Fundamentals of Telecommunication Networks*. John Wiley & Sons, 1994.

[5] S. Deering, C. Partridge, and D. Waitzman, "Distance Vector Multicast Routing Protocol", *RFC 1075*, November, 1988.

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

[7] J. Postel, "Internet Protocol", *RFC 791*, September, 1981.

[8] C. Hedrick, "Routing Information Protocol", *RFC 1058*, June, 1988.

[9] J. Moy, "OSPF Protocol Analysis", *RFC 1245*, August, 1991.

[10] J. Moy, "OSPF Version 2", *RFC 1247*, August, 1991.

[11] J. Moy, "OSPF Version 2", *RFC 1583*, March, 1994.

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

[13] J. Ioannidas, D. Duchamp, and G. Maguire, "IP-based Protocols for Mobile Internetworking", in *Proceedings ACM SIGCOMM Conference*, 1991.

[14] J. Ioannidas and G. Maguire, "The Design and Implementation of a Mobile Internetworking Architecture", *Proceedings Winter USENIX Symposium*, 1993.

[15] R. Caceres and L. Iftode, "Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments", *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 5, pp. 850–857, June, 1995.

[16] J. Postel and J. Reynolds, "File Transfer Protocol", *RFC 959*, October, 1985.

[17] J. Mello and P. Wayner, "Overview: Wireless Mobile Communications", *Byte*, vol. 18, no. 2, pp. 146–154, February, 1993.

[18] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource ReSerVation Protocol (RSVP) — Version 1 Functional Specification", *RFC 2205*, September, 1997.

[19] W. Leung, T. Baumgartner, Y. Hwang, M. Morgan, and S. Tu, "A Software Architecture for Workstations Supporting Multimedia Conferenceing in Packet Switching Networks", *IEEE Journal on Selected Areas Communications*, vol. 8, no. 3, pp. 380–390, April, 1990.

[20] C. Liu, Y. Xie, M. Lee, and T. Saadawi, "Adaptive Synchronization in Real-time Multimedia applications", in *Proceedings of Multimedia Communications and Video Coding*, New York, October, 1995.

[21] F. Alvarez-Cuevas, "Voice Synchronization in Packet Switching Networks", *IEEE Network*, vol. 7, no. 5, pp. 20–25, September, 1993.

[22] J. Escober, "Flow Synchronization Protocol", *IEEE/ACM Transactions on networking*, vol. 2, no. 2, pp. 111–121, Vol. 2, No. 2, April, 1994.

[23] H. Katseff and B. Robinson, "Predictive Prefetch in the Nemesis Multimedia Information Service", *Proceedings of ACM Multimedia 94*, October, 1994.

[24] S. Ramanathan and P. Rangan, "Adaptive Feedback Techniques for Synchronized Multimedia Retrieval over Integrated Networks", *IEEE/ACM Transactions on Networking*, vol. 1, no. 2, pp. 246–260, April, 1993.

[25] D. Estrin, D. Farinacci, A. Helmy, V. Jacobson, and L. Wei, "Protocol Independent Multicast — Dense Mode (PIM-DM): Protocol Specification", *Work in progress*, September, 1996.

[26] S. Deering, D. Estrin, D. Farinacci, M. Handley, A. Helmy, V. Jacobson, C. gung Liu, P. Sharma, D. Thaler, and L. Wei, "Protocol Independent Multicast — Sparse

Mode (PIM-SM): Motivation and Architecture", *Work in progress*, January, 1995.

[27] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, V. Jacobson, C gung Liu, P. Sharma, and L. Wei, "Protocol Independent Multicast — Sparse Mode (PIM-SM): Protocol Specification", *Work in progress*, January, 1995.

[28] A. Ballardie, "Core Based Trees (CBT) Multicast: Architectural Overview", *Work in progress*, March, 1997.

[29] A. Ballardie, "Core Based Trees (CBT) Multicast: Protocol Specification", *Work in progress*, March, 1997.

[30] C. Perkins, "IP Mobility Support", *RFC 2002*, October, 1996.

[31] V. Kumer, *MBone: Interactive Multimedia on the Internet.* Macmillan Publishing, 1995.

[32] C. Perkins, "IP Encapsulation within IP", *RFC 2003*, October, 1996.

[33] C. Perkins, "Minimal Encapsulation within IP", *RFC 2004*, October, 1996.

[34] T. Maufer and C. Semeria, "Introduction to IP Multicast Routing", *Work in progress*, March, 1997.

[35] S. Deering, "Multicast Routing in Datagram Internetworks and Extended LANs", *ACM Transactions on Computer Systems*, vol. 8, no. 2, pp. 85–110, May, 1990.

[36] G. Malkin, "RIP Version 2 Carrying Additional Information", *RFC 1723*, November, 1994.

[37] C. Huitema, *Routing in the Internet.* Prentice Hall, 1995.

[38] C. Huitema, *IPv6 The New Internet Protocol.* Prentice-Hall, 1996.

[39] F. Kastenholz and C. Partridge, "Technical Criteria for Choosing IP: The Next Generation (IPng)", *RFC 1726*, December, 1994.

[40] A. Conta and S. Deering, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6)", *RFC 1885*, January, 1996.

[41] T. Narten, E. Nordmark, and W. Simpson, "Neighbor Discovery for IP Version 6 (IPv6)", *RFC 1970*, August, 1996.

[42] G. Malkin and R. Minnear, "RIPng for IPv6", *RFC 2080*, January, 1997.

[43] V. Fuller, T. Li, J. Yu, and K. Varadhan, "Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy", *RFC 1519*, September, 1993.

[44] D. Meyer, "Some Issues for an Inter-domain Multicast Routing Protocol", *Work in progress*, March, 1997.

[45] J. Beasley, "An SST-based Algorithm for the Steiner Problem in Graphs", *Networks*, vol. 19, no. 1, pp. 1–16, January, 1989.

[46] K. Bharath-Kumer and J. Jaffe, "Routing to Multiple Destinations in Computer Networks", *IEEE Transactions on Communications*, vol. COM-31, no. 3, pp. 343–351, March, 1983.

[47] F. Hwang and D. Richards, "Steiner Tree Problems", *Networks*, vol. 22, no. 1, pp. 55–90, January, 1992.

[48] P. Winter, "Steiner Problem in Networks: A Survey", *Networks*, vol. 17, no. 2, pp. 129–168, February, 1987.

[49] S. Voss, "Steiner's Problem in Graphs: Heuristic Methods", *Discrete Applied Mathematics*, vol. 40, no. 1, pp. 45–72, 1992.

[50] R. Gallager, P. Humblet, and P. Spira, "A Distributed Algorithm for Minimum-Weight Spanning Trees", *ACM Transactions on Programming Languages Systems*, vol. 5, no. 1, pp. 66–77, January, 1983.

[51] D. Wall, "Mechanisms for broadcast and selective broadcast", *Ph. D. dissertation, Stanford University*, June, 1980.

[52] R. Sedgewick, *Algorithms*. Addison-Wesley, 1988.

[53] B. Waxmen, "Routing of Multipoint Connections", *IEEE Journal on Selected Areas Communications*, vol. SAC-6, no. 9, pp. 1617–1622, December, 1988.

[54] L. Kou and K. Makki, "An Even Faster Approximation Algorithm for the Steiner Tree Problem in Graphs", *Congressus Numerantium*, vol. 59, pp. 147–154, 1987.

[55] L. Kleinrock and F. Kamoun, "Hierarchical Routing for Large Networks: Performance Evaluation and Optimization", *Computer Networks*, vol. 1, no. 3, pp. 155–174, 1977.

[56] D. Meyer, "Administratively Scoped IP Multicast", *Work in progress*, December, 1996.

[57] R. Droms, "Dynamic Host Configuration Protocol", *RFC 1541*, October, 1993.

[58] S. Bellovin, "Security Problems in the TCP/IP Protocol Suite", *Computer Communications Review*, vol. 19, no. 2, pp. 32–48, March, 1989.

[59] R. Rivest, "The MD5 Message-Digest Algorithm", *RFC 1321*, April, 1992.

[60] D. Johnson and C. Perkins, "Route Optimization in Mobile IP", *Work in progress*, February, 1996.

[61] S. Thomson and T. Narten, "IPv6 Stateless Address Autoconfiguration", *RFC 1971*, August, 1996.

[62] J. Bound and C. Perkins, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", *Work in progress*, March, 1997.

[63] A. Conta and S. Deering, "Generic packet tunneling in IPv6 specification", *Work in progress*, June, 1996.

[64] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", *RFC 1883*, January, 1996.

[65] A. Acharya, A. Bakre, and B. Badrinath, "IP Multicast Extensions for Mobile Internetworking", in *IEEE INFOCOM'96 Proceedings*, March, 1996.

[66] A. Acharya and B. Badrinath, "Delivering Multicast Messages in Networks with Mobile Hosts", in *Proceedings of the 13th International Conference on Distributed Computing Systems*, May, 1993.

[67] D. Duchamp, *Issues in Wireless Mobile Computing*. IEEE Computer Society Press, January, 1992.

[68] C. Liu, M. Lee, and T. Saadawi, "A Scalable Multicast Routing Structure", in *IEEE MILCOM 97 Proceedings*, California, November, 1997.

[69] C. Liu, M. Lee, and T. Saadawi, "Large Scale Multicasting in Military Networks", in *ATIRP Annual Conference Proceedings*, Maryland, February, 1998.

[70] "IEEE P802.11 Draft Standard for Wireless LAN", *P802.11 D5.0*, July, 1996.

[71] W. Fenner, "Internet Group Management Protocol, Version 2", *Work in progress*, January, 1997.

[72] G. Xylomenos and G. Plyzos, "IP Multicast for Mobile Hosts", *IEEE Communications Magazine*, pp. 54–58, January, 1997.

[73] L. Kou, G. Markowsky, and L. Berman, "An Faster Algorithm for Steiner Trees", *Acta Informatica*, vol. 15, no. 2, pp. 141–145, 1981.

Index

N

O

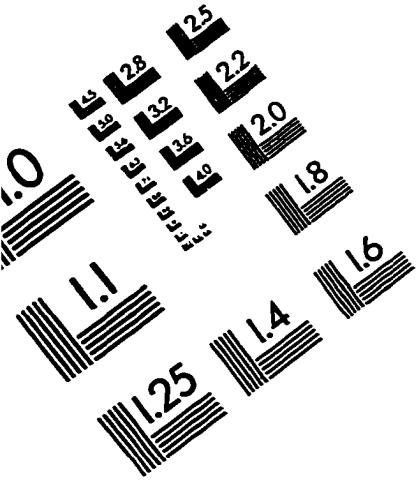
P

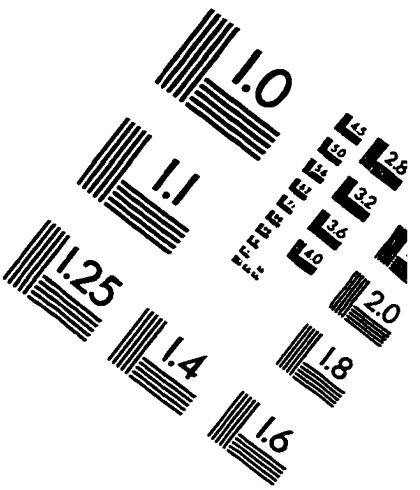
Q

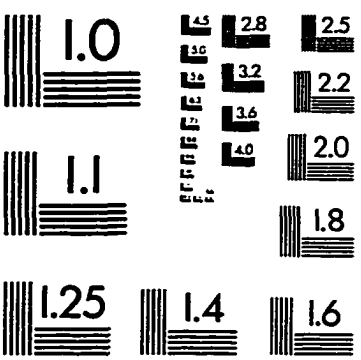
R

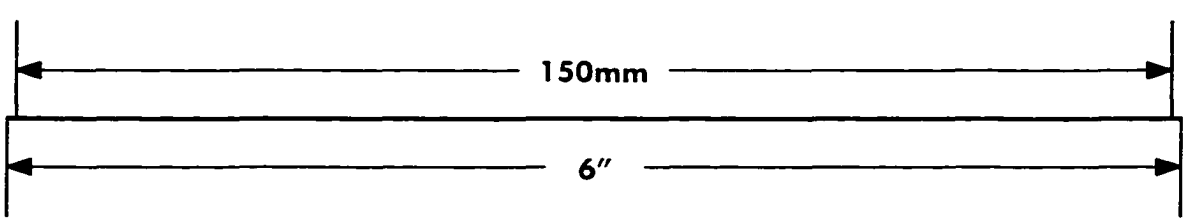
S

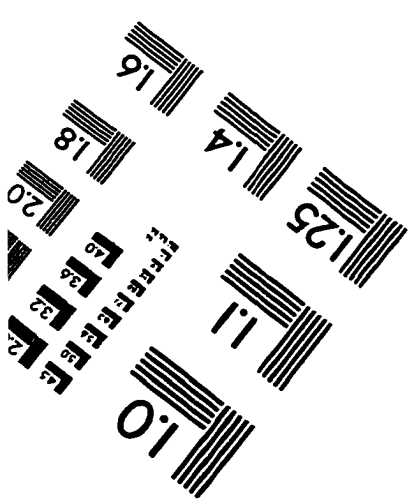
IMAGE EVALUATION TEST TARGET (QA-3)

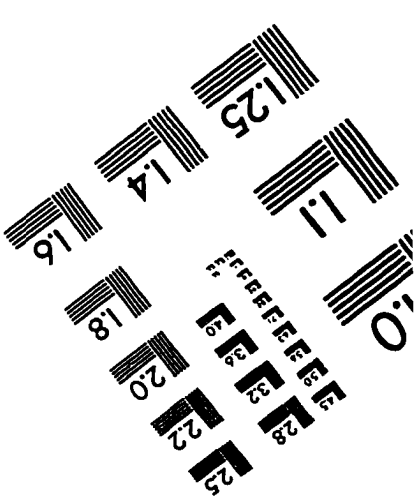












APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989