Université d'Ottawa · University of Ottawa

Université d'Ottawa · University of Ottawa

FACULTÉ DE ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

Mojtaba HOSSEINI

AUTEUR DE LA THÈSE - AUTHOR OF THESIS

Ph.D.(Electrical Engineering)

GRADE - DEGREE

School of Information, Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT - FACULTY, SCHOOL, DEPARTMENT

TITRE DE LA THÈSE - TITLE OF THE THESIS

An End System Multicast Protocol for Multi-party Videoconferencing Applicants

N. Georganas

DIRECTEUR DE LA THÈSE - THESIS SUPERVISOR

CO-DIRECTEUR DE LA THÈSE - THESIS CO-SUPERVISOR

EXAMINATEURS DE LA THÈSE - THESIS EXAMINERS

A. El Saddik

J. Liebeherr

D. Petriu

S. Shirmohammadi

J.-M. De Koninck, Ph.D.

LE DOYEN DE LA FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

DEAN OF THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

An End System Multicast Protocol for Multi-party Videoconferencing Applications

by

Mojtaba Hosseini, M.A.Sc.

A thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Electrical Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering

School of Information Technology and Engineering
University of Ottawa

Your file Votre référence
ISBN: 0-494-01710-4
Our file Notre référence
ISBN: 0-494-01710-4

Acknowledgement

The persons that have helped me during the course of my studies whose culmination has become this thesis are too numerous to count. I limit myself to thanking Dr. Nicolas Georganas for his supervision, mentorship, support and friendship ever since my undergraduate years and my parents whose support, encouragement and prayers have given me the strength throughout my life.

Abstract

This thesis presents the design of an end system multicast protocol appropriate for multi-party videoconferencing applications. More specifically, the said protocol attempts to address the upload bandwidth limitations of current Internet connections to the home as well as the lack of ubiquitous infrastructure support for multicasting needed for many-to-many applications. After a study of the requirements of a particular class of many-to-many applications (multi-party videoconferencing), the thesis includes the proposal, design, justification, implementation and evaluation of a novel end system multicast protocol targeting such applications that in comparison to existing systems and protocols will better deal with limitations of today's Internet connections to the home and improve their feasibility of deployment. The core of the thesis is formed by novel routing heuristics appropriate for multi-source outflow constrained overlay tree construction that significantly perform better than existing heuristics in finding solutions while taking into account priority and heterogeneous bit-rates for every source-receiver pair. Furthermore, the design and implementation of a 3D videoconferencing application with a novel visibility-based awareness mechanism provides a proof of concept application making use of our proposed end system multicast protocol.

Table of Contents

List of Abbreviations

ALM: Application Layer Multicast

CVE: Collaborative Virtual Environment

DSL: Digital Subscriber Line

DVE: Distributed Virtual Environment

ESM: End System Multicast

IP: Internet Protocol

ISP: Internet Service Provider

JMF: Java Media Framework

MBone: Multicast Backbone

MD-MSOC: Minimum Diameter Multi-Source Outflow-Constrained

MOTOR: Multi-source Outflow-constrained Tree Overlay Routing

MPEG: Moving Picture Experts Group

MSN: Microsoft Network

MVEMP: Multi-party Videoconferencing End-system Multicast Protocol

RP: Rendezvous Point

RTP: Real Time Protocol

RTCP: Real Time Control Protocol

TCP: Transmission Control Protocol

UDP: User Datagram Protocol

VR: Virtual Reality

VRML: Virtual Reality Modeling Language

WAN: Wide Area Network

Chapter 1 Introduction

1.1 Multi-party Videoconferencing

Videoconferencing refers to two or more persons communicating over a distance through audio and video media. The idea is to capture, in the form of video and audio media, the sights and sounds of a set of participants wishing to communicate, transmit these media over the distance separating the participants and display the captured media in order to allow the participants to communicate with one another.

Videoconferencing has a long history, the first manifestations of which involved analog televisions receiving audio and video signals from remote sites via satellite links. Within the last decade, videoconferencing has expanded to include computers exchanging digital audio and video data over the global inter-network (Figure 1).

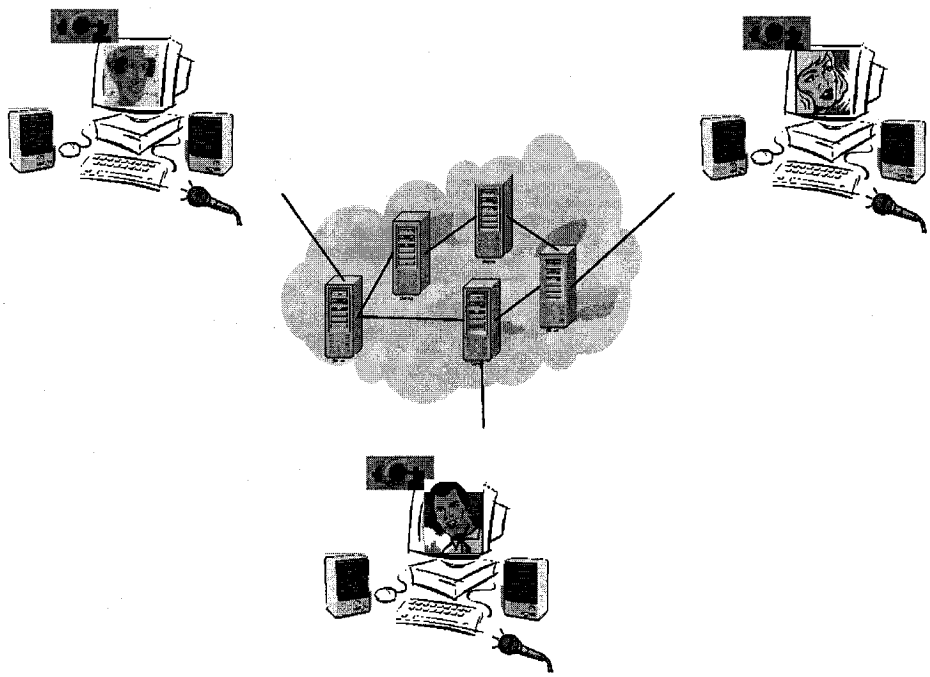


Figure 1. Video and audio conferencing over the Internet

With the goal of allowing remote persons to more easily and naturally communicate with one another, videoconferencing holds the potential of being an invaluable communication tool and in fact has already proved its utility for many businesses, educational institutions and individuals. The ubiquitous existence of computers and the growing availability of high-speed connections to the global Internet give rise to the potential of making videoconferencing tools as commonplace as the telephone.

Development of videoconferencing systems and related technologies can delve into almost every area of computing from Human-Computer Interaction (e.g. user interface design), hardware and software implementation of sensors and actuators (e.g. cameras, microphones, speakers and displays) to data representation, coding (e.g. audio/video format, coding, compression) and transmission (e.g. video streaming, congestion control, network utility and protocols). Of course most developers take advantage of available solutions in each area by, for example acquiring off the shelf equipment (cameras, microphones) and established standards (MPEG standards for audio and video coding) and would concentrate their development effort towards addressing a particular set of problems.

1.2 Motivation

Though a subject of research for many years, bi-party or point-to-point videoconferencing carried over the Internet is still faced with many issues that need to be more adequately addressed. Dealing with network congestion, providing objective and subjective Quality of Service to different users both at the application and the network layer as well as non-technical issues such as privacy and the business model and profitability of such applications are still topics of investigation and study. Despite these

issues however, bi-party videoconferencing has more recently achieved a high degree of ubiquity due to the increasing availability of high-speed Internet connections to the home and the proliferating Instant Messaging services provided by Yahoo [110] and MSN [77] that allow for audio and video conferencing in addition to their traditional text messaging service. Supporting multiple participants in a videoconference has been more difficult to achieve and the current solutions for multi-party videoconferencing applications either require special support from the network infrastructure that is not available to the typical Internet user or do not scale well (see section 2.1). As will be highlighted in the coming chapters, one of the fundamental and crucial obstacles in the way of the deployment of multi-party videoconferencing applications over the Internet is the limited upload bandwidth available, especially to the typical home user. In this regard, we propose and evaluate novel solutions at the application layer that help to alleviate the bandwidth limitations of multi-party videoconferencing applications.

1.3 Objectives

The efforts outlined in this thesis are distinguished from previous work (see section 2.1 for a survey of existing videoconferencing solutions and section 2.3 for existing application layer multicast efforts) as a result of the difference in the objectives they set out to fulfill. These are:

- The bandwidth available to most Internet users being incongruent to the bandwidth required to support multi-party videoconferencing applications, many existing approaches employ specialized or proprietary networks (e.g. satellite, ATM intranets) or special subsections of the Internet (e.g. the MBone) that are not made available to a typical home user through their Internet Service Providers. In

this context, we aim to utilize the current best-effort unicast-only Internet without requiring special services from the Internet infrastructure in order to facilitate the deployment of multi-party videoconferencing applications. We therefore assume that each conferencing participant has at most a Digital Subscriber Line connection to the Internet.

- Multi-party conferencing applications can be further categorized as those that involve small groups of participants or those with a larger scale of users. Small-group conferences with 4-10 participants typically require a high degree of awareness between all the participants (e.g. a business meeting or a small group discussion). Large-scale conferences require the support of tens or hundreds of users but do not require the same degree of mutual awareness between every participant. Within our work, we aim to support conferencing between a small group of users (4-10) but with a higher degree of mutual awareness (e.g. every participant can potentially see the video of any other participant).
- Some existing approaches to multi-party conferencing allow only one user (typically the speaker) to be the media source at any one point. In order to better foster the experience of engaging in a true multi-party session, we strive to support multiple simultaneous media sources and in consequence allow different participants the flexibility to choose whom they see.
- A common remedy to the inadequate availability of bandwidth for multi-party videoconferencing is to reduce the quality of the media corresponding to each participant (e.g. reducing resolution and/or frame rate of the video) at the cost of hampering the quality of the communication session. Our objective is to leave

unchanged the quality of the media as much as possible and instead find ways of dealing with the bandwidth limitation that are less noticeable by users of the application as compared to simply reducing the quality of the transmitted media.

Though there are efforts that encompass some of these objectives (see section 2.1), our work is, to the best of our knowledge, the first to try to address them all.

1.4 Contributions

The contributions of this thesis come about from our attempts at satisfying the objective stated in the previous section:

- Design, proof of concept implementation and evaluation of an End System Multicast protocol geared towards the unique requirements of multi-party videoconferencing applications
- New routing heuristics for constructing overlay trees in a multi-source outflow constrained environment that find a significantly greater percentage of solutions compared to similar heuristics
- In addition to finding a significantly greater percentage of solutions, one of our proposed routing heuristic creates trees with source-receiver delays comparable to the best heuristic in the literature
- Consideration of heterogeneous bit-rates and different priority of streams in the routing heuristic unlike other heuristics
- A novel visibility-based awareness management mechanism appropriate for multi-party videoconferencing and other media intensive Collaborative Virtual Environments and one that can be used in conjunction with an end system multicast protocol as well as the proof of concept implementation and evaluation

of the visibility-based awareness management in the context of a 3D virtual videoconferencing application

- Proof of concept implementation and evaluation of a multi-party videoconferencing employing the said awareness management scheme and deployable over our End System Multicast protocol

This work has thus far been published in one journal, one magazine and 5 conference publications while one more journal publication is in review:

- Hosseini, M., Georganas, N.D., "Application Layer Multicast Protocol for Collaborative Virtual Environments", *Presence: Teleoperators and Virtual Environments, Volume 13, Issue 3, June 2004*, pp. 263-278.
- Hosseini, M., Georganas, N.D., "End System Multicast Routing for Small-Group Multi-Party Videoconferencing Applications", *under submission.*
- Hosseini, M., Georganas, N.D., "Design of a Multi-sender 3D Videoconferencing Application over an End System Multicast Protocol", *Proceedings of 2003 ACM conference on Multimedia (MM'03)*, pp. 480-489.
- de Oliveira, J.C., Hosseini, M., Shirmohammadi, S., Malric, F., Nourian, S., El Saddik, A., Georganas, N.D., "Java Multimedia Telecollaboration", *IEEE Multimedia, volume 10, number 3, July-September 2003*, pp. 18-29
- Hosseini, M., Georganas, N.D., "User Behavior in 3D Videoconferencing Applications", *Proceedings of 2003 Summer Computer Simulation Conference (SCSC'03)*, pp. 755-760.
- Hosseini, M., Pettifer, S., N.D. Georganas, "Development of Awareness Driven 3D Videoconferencing Applications", 7th WSEAS International conference on

Circuits, Systems Communications and Computers (CSCC 2003).

- Hosseini, M., Pettifer, S., Georganas, N.D., "Visibility-based Interest Management in Collaborative Virtual Environments", *Proceedings of 2002 ACM Conference on Collaborative Virtual Environments (CVE'02),* Bonn, Germany, October 2002, pp. 143-144
- Hosseini, M., Malric, F., Georganas, N.D. "A Haptic Virtual Environment for Industrial Training", *Proceedings of IEEE International Workshop on Haptic Audio Visual Environments and their Applications*, Ottawa, Canada, November 2002, pp. 25-30.

Chapter 2 Background and Related Work

The background of our work consists of previous and existing approaches to multi-party videoconferencing as well as the relatively new concept of Application Layer Multicasting. We begin this chapter by a brief survey of well-established videoconferencing practices and existing systems and highlight the advantages and shortcomings of each. The chapter then continues with a study of the main characteristics of multi-party videoconferencing applications and how these translate into network and system requirements with a focus on multicasting capability. A detailed survey of current Application Layer Multicast protocols then serves to highlight a trend of approaches for providing multicast services to one-to-many and many-to-many applications. The chapter concludes by demonstrating that many of the current Application Layer Multicast protocols are not designed for and therefore do not satisfy the requirements of multi-party videoconferencing applications and thus serves as a justification of the proposals prescribed in the chapters that follow in the rest of the thesis.

2.1 Brief Survey of Multi-party Videoconferencing Systems

Being bandwidth intensive applications, advances in multi-party videoconferencing largely depend on 1) video coding and compression technologies, 2) throughput bandwidth available from the network and 3) network support for multi-point communication and possibly Quality of Service guarantees.

While advanced video compression techniques reduce the bandwidth requirement of each video stream, advanced network protocols and topologies make more efficient use of the available network resources. Advances in each of these three broad areas increase the

feasibility of multi-party videoconferencing applications.

Chen [18] gives an overview of current video compression algorithms appropriate for videoconferencing and the minimum required frame rate needed to at least detect some gestures relevant to floor control in such applications. For his virtual auditorium application, 320×240 at 15fps video is found to be sufficient to detect the facial expressions (a smile) of the captured user [18] while the lower bound on frame rate was found by Tang and Isaacs [98] to be 5fps in order for the video to be tolerable. Though even a lower frame rate can be used for the non-speaking participants in order to reduce bandwidth requirements, Chen observed that full-motion video (i.e. at least 15 fps) is necessary to convey gaze, facial expressions and body movements. In addition, a common complaint about lower frame rates was due to video of people being captured at moments that made them look 'silly' (e.g. in the middle of a movement) [19]. Throughout this thesis, we aim at supporting a minimum of 320×240 video at 15fps in order to effectively convey body and facial expressions and possibly gaze. It should be mentioned that an MPEG-4 encoder can be expected to produce an average 100kbps bit rate for a 320×240 video at 15fps [18].

Despite significant advances and years of research in the areas of video coding, as well as the relevant transport and network support required by such applications, multi-party videoconferencing on the Internet has not yet achieved the degree of ubiquity one would expect. We broadly categorize existing approaches and systems into those that do not scale well and those that require special services from the underlying network even though such services are not yet available to home Internet users through their ISPs and go on to highlight a promising trend in carrying out multi-party conferences within

computer graphics generated virtual environments.

2.1.1 Systems that do not scale well

Most commercial videoconferencing solutions such as MSN [77] and Yahoo Messengers [110] as well as Microsoft's NetMeeting [75] and other commercial products [105] fall into this category. These systems support multi-party sessions by simply transmitting from every node to every other node. Considering that a typical DSL connection to the Internet affords a maximum of a few 100kbps of upload bandwidth and that a 320×240 MPEG-4 coded video at 15 fps requires an average of 100kbps [18], one can quickly observe that such systems do not scale well to support even a small group of 4 to 10 participants. A common remedy to this problem is to reduce the resolution and/or the frame rate of the video in order to support a greater number of users, thus degrading the quality of the communication as the number of participants increases. However as noted earlier, reducing frame rate or resolution of the video comes at the cost of less effective communication as a result of the loss of conveyance of body and facial gestures. In return for their simplicity however, the great advantage of such systems lies in their ability to be immediately deployable over today's Internet.

2.1.2 Systems requiring special network services

Most multi-party videoconferencing research prototypes originating from universities and research institutions work on top of specialized networks or assume network services that are not yet widely supported on the Internet (though they may be in the future). VidZ [70] for example operates on a pure ATM test bed, Forum [52] and TELEP [53] use company intra-nets with ample bandwidth and multicasting capability available while nodes in Chen's Virtual Auditorium [18] are connected to one another through a high-speed

Ethernet network. Such research endeavors are important with regards to the study of applications of next generation Internet, the interfaces that may be appropriate and needed as well as some of the services such applications require from the underlying network infrastructure. Their deployment on the wide area network however depends on the future availability of the advanced networks they employ.

A significant step towards bringing multicasting capability to the global Internet has been the advent of IP Multicast first proposed by Deering et al [27]. The basic idea behind IP Multicast is for network components (i.e. routers) involved in a multicast session to duplicate data when necessary, thus allowing a multicasting source to transmit only one copy of its data and leaving the network to perform the data duplication necessary to transmit to multiple receivers (see Figure 2 where routers 0, 3 and 6 replicate data for the source).

Since its inception, there have been a myriad of proposals, protocols and standards improving the feasibility and ease of deployment of IP Multicast over the Internet [32]. Although implemented on most Local Area Networks, IP Multicast's deployment and availability on the global Internet has been slow due to various technical and business-related issues [30]. As a result, a Multicast Backbone (MBone) has been constructed that connects the various IP Multicast capable 'islands' via unicast connections called 'tunnels' [34] (see Figure 2). Scalable audio and video conferencing on top of the MBone is thus made possible [71] through the various MBone conferencing tools [102] [63].

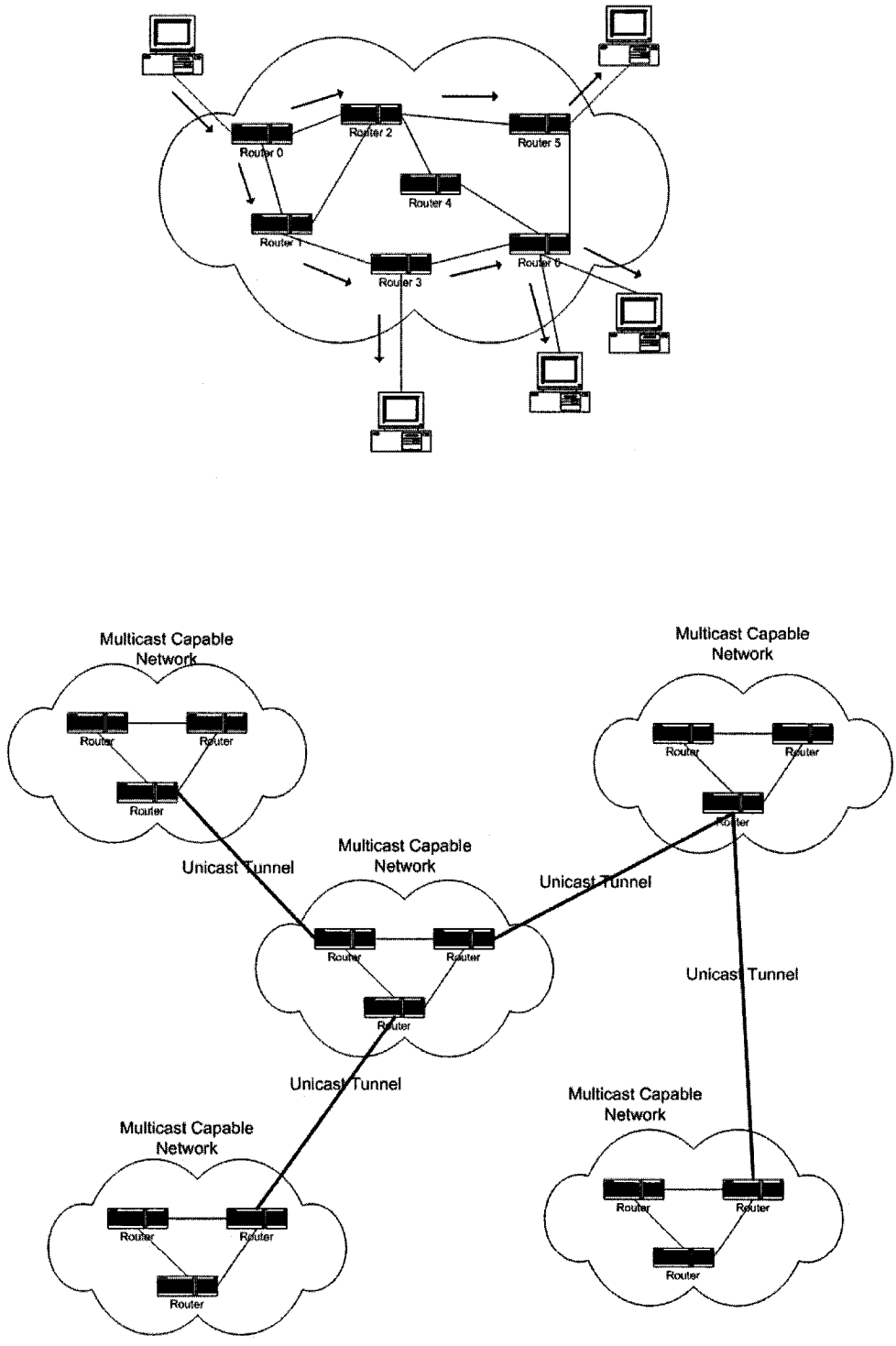


Figure 2. IP Multicast concept (top) and unicast tunnels connecting IP Multicast capable networks in the Mbone (bottom)

MBone teleconferencing, though scalable and efficient, requires all participants to be connected to this backbone. Considering that the creation of multicast 'tunnels' typically requires a network administrator to statically set up a unicast 'link' between two network components [34] and that many ISPs do not allow their users to have a connection to the MBone due to various issues [30], we can regard the MBone as a special IP Multicast capable network that is not yet available to typical home Internet users.

Although there are ISPs that provide multicast services to their customers (see [78] for a list), their number and coverage is limited, leaving commercial multi-party conferencing software vendors to seek alternatives. CUSeeMe [25] for instance employs the popular concept of 'reflector' or Multi-point Control Unit (MCU) [10] in order to support multiple participants. A reflector or MCU is a dedicated server that receives media streams from all participants in a session and multi-unicasts the streams to others as shown in Figure 3.

It is common for a reflector to also employ a bandwidth reduction scheme such as picture-in-picture or voice-activated switching [10]. A voice-activated switching scheme allows only the current (and perhaps the previous) speaker of a session to transmit their video to others. A picture-in-picture scheme (used by CUSeeMe) merges multiple video streams into one by reducing the resolution of the individual streams. The disadvantage of voice-activated switching is that users do not choose whose video they see (which is found to be undesirable [10]) while the disadvantage of picture-in-picture is inherent in its reduction of the quality of the video medium.

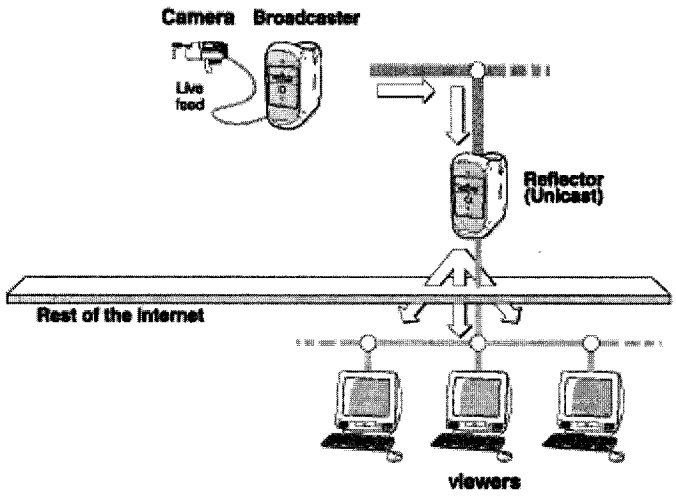


Figure 3. A reflector multi-unicasting a video stream to multiple participants

Besides the interface-related inconveniences of these approaches, they require the deployment of a dedicated reflector or MCU for each session that must have enough bandwidth available to it to receive from and transmit to multiple nodes.

2.1.3 Conferencing within virtual environments

Research into the human-computer interface of teleconferencing applications has led to proposals of carrying out multi-party conferencing sessions within computer generated virtual environments. These in turn have important implications with respect to the network requirements of such applications.

Embedding live video into a computer generated 3D space has long held the promise of carrying the notion of meeting in a physical conference room to the meeting of remote users in a virtual space [43]. It has also been argued that compared to traditional videoconferencing where multiple video streams divide and share a tiled 2D interface, video embedded inside a 3D Collaborative Virtual Environment (CVE) can better support mutual awareness and spatial consistency for users [89]. In such applications, each user is

represented by their video in a 3D computer generated virtual space, typically with the video textured on a 2D plane. Users can move and look around in the virtual space and their movements as well as their corresponding video and audio streams are distributed to other users so that the virtual space presents a consistent environment for all users. Within this 3D space users can tell where the attention of another participant is directed by way of his/her position and orientation. Recent attempts to provide gaze awareness in computer supported cooperative work have also moved towards integrating the captured video or picture of users into a computer generated 3D space and tracking their head and/or eye movements in order to convey their gaze [99] [104].

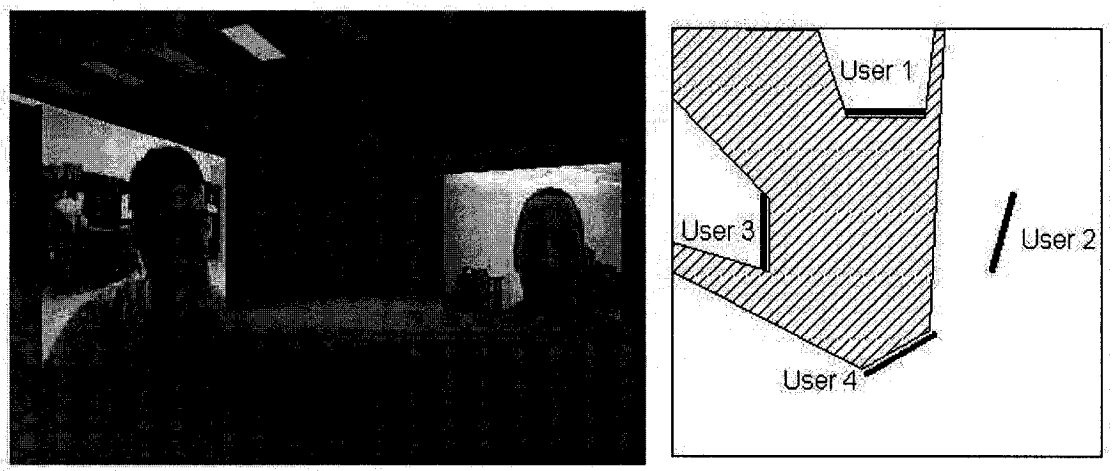


Figure 4. User 4 receives video only from users 1 and 3 since user 2 is out of view

The advantages with regards to the interface notwithstanding, teleconferencing within computer generated virtual environments has significant implications with respect to the network requirements of such applications. This is primarily due to the fact that embedding video-based objects in a computer generated 3D world associates a spatial parameter or location with each object, based on which mutual awareness between any two objects can be discerned. Distributed Virtual Environments (DVE) and CVEs have long utilized this characteristic in order to increase their scalability through various Area

of Interest Management mechanisms (see chapter 7 of [96] for an overview). More specifically adapted to teleconferencing applications, an Area of Interest Management mechanism can serve to more naturally limit the number of media streams exchanged between conferencing peers. In the MASSIVE system for instance, spatial distance between users is used to measure the degree of awareness between them in order to determine the data sources most relevant to every user [39]. Figure 4 illustrates. This concept can be further extended to provide a Quality of Service model for the teleconferencing application where different levels of awareness signify different levels of media quality each user requests to receive [40].

By making use of the spatial parameter in virtual environments, awareness driven videoconferencing (a term introduced in [89]) can avoid the transmission of media streams that are outside the focus or awareness of a participant and thereby promise better scalability and support for multi-party conferencing sessions [47]. We therefore focus our work on multi-party videoconferencing applications carried out within the context of a 3D virtual environment, whose examples can be found in [40] [39] [89] [4] [47] [104] [80] [88]. This work can however be extended to other multi-party conferencing applications since the protocol and the routing heuristic proposed in this thesis are not specific to 3D conferencing and can operate as long as some form of application filtering limits the total number of streams to the total available bandwidth. Note that while [40] [39] [89] and [80] require an IP Multicast capable network, [4] and [104] require 100Mbps Local Area Networks for their prototypes. Our work distinguishes itself from the mentioned previous efforts by employing an End System Multicast protocol (see Chapter 3), therefore not requiring special network services and being immediately

deployable over today's Internet.

2.2 *Requirements of Awareness-Driven Multi-party Videoconferencing Systems*

The previous section argued for the use of awareness-driven videoconferencing systems due to their potential for providing better scalability, in addition to exhibiting a better human-computer interface. This section outlines the main characteristics of this group of applications and how these affect the transport and network layer protocols that support them.

2.2.1 Bandwidth constrained

Transmission of audio and video media is traditionally considered to be a bandwidth-intensive undertaking. For example, a typical home user with a DSL connection to the Internet can expect an available upload bandwidth of a few 100kbps. Considering a 100 kbps 320×240 MPEG-4 coded video at 15fps [18], a DSL connection potentially allows a user to transmit its video to only 2 to 3 other users, insufficient even for small-group conferences of 4 to 10 participants. Note also that the asymmetric nature of most DSL lines implies a more stringent bottleneck in uploading than downloading. It is therefore important for any supporting protocols to take into account the upload bandwidth constraints of each participating node.

2.2.2 Multiple simultaneous sources

The limited bandwidth is further exasperated by the possibility of multiple simultaneous sources transmitting to their receivers. For instance many multi-party videoconferencing prototypes (such as [40] [39] [89] [4] [47] [80] [104]) represent every conferencing participant as a video source that can be seen by others and may therefore be transmitting simultaneously with other sources. As will be discussed in section 2.3, this particular

characteristic invalidates the use of many existing protocols that are designed for a single source scenario.

2.2.3 Application layer filtering

As was mentioned in section 2.1.3, one significant advantage of virtual environment-based multi-party videoconferences is the possibility of employing application-specific filtering mechanisms to limit the number of media streams a participant receives to only those he/she is aware of or is most interested in. This implies that at different times during the session, the set of receivers a source must transmit to changes depending on changes in awareness/focus/interest. In fact, not every user is expected to transmit its corresponding video to every other user due to the awareness-driven filtering mechanism [47]. As will become evident in the following chapters, we exploit this characteristic in order to use an application layer multicast protocol for awareness-driven multi-party conferencing applications. The application layer filtering mechanism also aids in limiting the number of received streams by each user in a non-intrusive and flexible manner. Note that awareness management is one form of application layer filtering and our protocol is still relevant for any other form of filtering employed.

2.2.4 Different levels of stream priority

Since a user can receive multiple streams, different priority levels can be assigned to each stream. In [104] for example, an eye-tracker discerns for each user the participant that he/she is visually focusing on. Higher priority can be given to the corresponding video stream, implying that the reception of that particular stream is of more value to the receiver than receiving from others that are not in focus. It would thus be beneficial for the supporting protocol to prefer the delivery of higher priority streams to lower priority ones.

2.2.5 Heterogeneous bit-rates

For systems employing some form of awareness management mechanism, different bit-rates can be assigned to users attracting different levels of awareness. It is beneficial for example for users that are further away or in the periphery [40], as well as those partially occluded [47] in the virtual meeting room to decrease their bit-rates, compared to other users attracting a higher degree of focus/awareness. Although adaptive video delivery is in itself an arduous task, ideally a supporting protocol of such applications should strive to allow for heterogeneous bit-rate delivery of video from a source to its receivers.

2.2.6 Small group communication

Multi-party conferencing usually implies a small group of 3 to 10 participants. Although there are applications that involve medium sized groups with several tens of participants (such as a virtual classroom [18]), our work focuses on the small-scale scenario in order to serve as a stepping-stone towards the larger scale applications. Furthermore, we conjecture that as the number of participants increases the need to support interaction between every pair of participants decreases. In other words, in a small group meeting it is more likely that all participants would desire to be aware of one another whereas it is more difficult to conceive of a scenario where hundreds of users are all aware of one another. Consequently, for large scale applications spatial partitioning techniques can divide the group into smaller groups (see chapter 7 of [96] for examples) and for medium scale applications it is possible to allocate more awareness and therefore resources to a subset of participants such as the speaker and reduce the media quality of other participants (see [19]). Throughout our work we envision the applicability of the concepts proposed therein to small group conferences of size 3 to 10 participants, yet do not make decisions that would explicitly limit the scalability of our approach to medium-sized

groups.

From the first two characteristics outlined in this section one can deduce that a multicasting service is a crucial requirement of multi-party videoconferencing applications in order to alleviate a source's bandwidth constraints when transmitting to multiple receivers. Furthermore, the existence of multiple simultaneous sources necessitates the use of multiple multicast groups within the same session. The remaining four characteristics have an impact on how such a multicast service is provided.

Given the importance of multicasting, it is not surprising that many multi-party conferencing applications require an IP Multicast capable network (e.g. [40] [39] [89] [80] [71]). The slow deployment of IP Multicast on the Internet however has given way to an alternative model of providing multicast service called Application Layer Multicast which promises an immediately deployable multicast service.

Our contribution in this regard is an Application Layer Multicast protocol designed to support the class of applications we have thus far discussed. It is therefore necessary for the next section to give an introduction to Application Layer Multicasting as well as a survey of existing approaches and routing heuristics employed therein. It will conclude by demonstrating that current Application Layer Multicast protocols are inadequate for supporting multi-party videoconferencing applications with the characteristics outlined in this section and thus serves as a preamble to our own proposed protocol.

2.3 *Survey of Application Layer Multicast Protocols*

Though it forms a crucial requirement of many applications such as video-on-demand, live broadcast, teleconferencing, collaborative multimedia as well as some distributed CVE and DVEs, the widespread deployment of IP Multicast has been slow due to yet

unresolved issues regarding administration, management, addressing, reliability, quality of service and congestion control [30]. Though there exist overlay networks connecting IP Multicast 'islands' (the most widespread of which is the Mbone [34]), they involve the static setting up of unicast connections or 'tunnels' and are still plagued by the same issues related to IP Multicast. As a result, many ISPs do not provide multicast capability to home Internet users, leaving most applications to still rely on unicast only.

In light of the slow deployment of IP Multicast on the Internet and encouraged by the success of peer-to-peer file transfer applications, recent research endeavors have studied application layer multicast and investigated its cost and benefits relative to IP Multicast [22]. Application Layer Multicast (ALM) refers to implementing multicast capability at the application layer instead of the network layer, or in other words, constructing a multicast-capable overlay network over a unicast infrastructure.

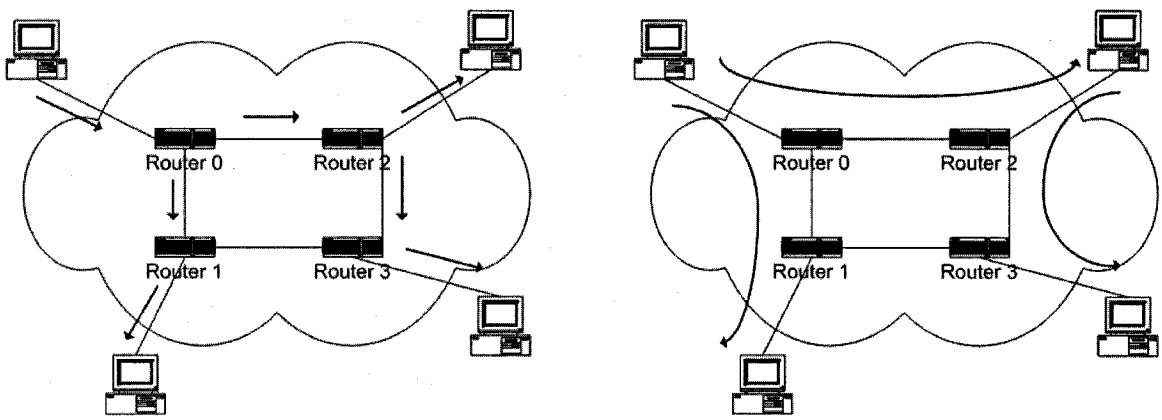


Figure 5. IP Multicast model (left) and Application Layer Multicasting (right)

The concept of ALM is simply the implementation of multicasting functionality as an application service instead of a network service. Figure 5 represents the IP Multicast model (i.e. multicast as a network service) in comparison to the ALM model.

While the former is implemented by network nodes (i.e. routers) and avoids multiple copies of the same packet on the same link as well as possibly constructing optimal trees,

the latter is implemented by application nodes (either end systems or proxies) and results in multiple copies of the same packet on the same link as well as typically constructing non-optimal trees. In exchange for its inefficiency, as compared to IP Multicast (by resulting in higher stress links and larger diameter trees), ALM remedies the key shortcoming of the IP Multicast model: easier and possibly immediate deployment over the WAN. In what can be regarded as one of the key efforts advocating ALM, Chu et. al [21] illustrated using both simulation and Internet experiments that self-organizing end systems can form overlay multicast trees that incur low performance penalties (in terms of link stress and tree stretch) compared to IP Multicast. Since then a multitude of application-layer multicast protocols have appeared with a wide range of characteristics.

A common approach to Application Layer Multicasting is for the multicast participants to establish an overlay topology of unicast links to serve as a virtual network on top of which multicast trees can be constructed. Figure 6a shows an example of 7 peers forming a topology.

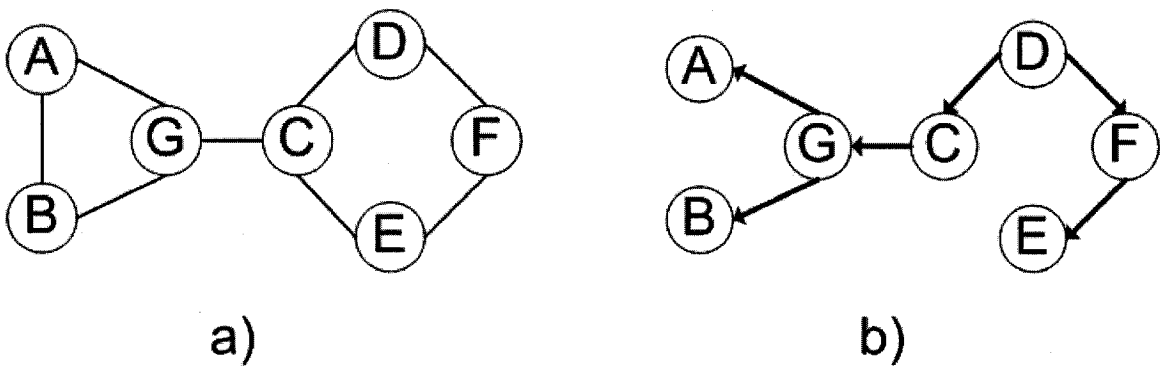


Figure 6. a) Sample overlay topology b) an overlay multicast tree

New members find out about the topology from a common bootstrap point called a Rendezvous Point (RP) and join the topology by exchanging control messages with a subset of members already part of the topology. A 'good' topology consists of a richly

connected graph (e.g. a peer is connected to other peers through multiple paths) in an efficient and cost-aware manner (e.g. distance or delay between peers is minimized while the number of connections is bounded). Other metrics such as robustness (ability to deal with members leaving the topology), scalability (ability to efficiently create topologies for very large number of peers) and low control overhead (minimizing the exchange of control messages) also determine the quality of an overlay topology. Creating and maintaining 'good' topologies thus becomes one of the core responsibilities of an ALM protocol. Once a topology is constructed and maintained, a multicast tree can be constructed on top of the graph (i.e. topology) according to a routing strategy that would commonly strive to minimize the cost of the multicast tree in terms of the delay experienced by each peer as well as the amount of data duplication each peer is required to perform. Figure 6b shows an example overlay tree over the sample topology of Figure 6a. The routing algorithm producing the tree also forms an important part of an ALM protocol. There exist a wide variety of ALM protocols each choosing a different strategy for creating and maintaining an overlay topology and creating multicast trees based on various metrics and algorithms.

Table 1 categorizes these according to the following criteria:

2.3.1 Application domain

Perhaps the most crucial feature of an ALM protocol and one that affects most of its resulting characteristics is its targeting application. The application domain determines the number of users that a protocol must support, the data type(s) a protocol's delivery tree must accommodate and the metrics that such a tree attempts to optimize. We follow the same categorization of application domains driving multicast deployment as those

according to Diot et. Al [30]:

1. Audio/video streaming: usually involves a single source distributing media to a large number of receivers. Examples include both live media streaming (e.g. a live video stream of a sporting event) and media on demand (e.g. pre-recorded news media streaming). The primary metric is bandwidth and latency to a lesser extent.
2. Audio/video conferencing: these involve small to medium size groups interacting in a multi-party conferencing session. The difference with the previous category is the smaller group size, higher degree of interactivity and the existence of multiple sources. Both bandwidth and latency are important metrics.
3. Large-scale distributed interactive simulations: potentially large number of users, each a source of its own lower (compared to audio/video) bit-rate data (e.g. massively multi-player games). The primary metric is interactivity (related to latency) and bandwidth to a lesser extent.
4. Push applications: very large number of users subscribe to a variety of information channels (such as various news topics from different sources) and receive the aggregated information at regular intervals whether the reader is actively interested or not (e.g. global event notification: news and stock prices). Bandwidth is the only metric, as the number of users can potentially be in the millions.
5. File transfer: reliable transfer and distribution of (usually large) files (e.g. distributed databases and file sharing). Bandwidth is the only metric.

As can be seen, the different classes of applications have different sets of requirements regarding reliability, latency, bandwidth and scaling (with respect to supporting large

number of users). Such requirements in turn determine the design choices of any ALM protocol regarding the group management mechanism it employs.

2.3.2 Deployment level

A key factor determining the set of assumptions an ALM protocol operates based on, is at what level the protocol is expected to be deployed: at the infrastructure level or end system level. Infrastructure-level or proxy-based ALM protocols require the deployment of dedicated servers/proxies on the Internet where they self-organize into an overlay network and provide a transparent multicast service to the end-user. End system level ALM protocols on the other hand, assume only a unicast service from the infrastructure and expect end-system hosts to participate in providing the multicasting functionality by taking on some of the forwarding responsibility.

The choice between developing an infrastructure level or an end system level ALM protocol is perhaps driven as much by business and marketing issues as purely technological ones. End systems sharing the forwarding load of a multicast session use the existing Internet infrastructure available to them and may not be expected to pay more for participating in the multicast session (as illustrated by the free nature of peer-to-peer file-transfer applications). An infrastructure of dedicated proxies deployed over the Internet that offer multicasting services however are more likely to expect a service charge. There are however technological consequences of a choice between a proxy-based or an end system level approach to ALM.

Proxy-based ALM protocols can take advantage of existing IP Multicast 'islands' by including a representative of an island as an overlay node (and therefore increase their efficiency), can assume greater bandwidth availability to the proxy nodes (compared to

the bandwidth available to end-systems), can assume longer life cycle of overlay nodes (compared to transient nature of end systems), relieve end-systems from any forwarding responsibility and therefore reduce application complexity since multicast is transparently made available to end-systems. The major disadvantage of this approach is the need for the deployment of dedicated proxies over the inter-network and so incurring the cost associated with acquiring and deploying them. Proxy-based ALM may also be less adaptable to and less optimized for specific applications since it would typically provide a generic multicast service rather than a service specific to a particular class of applications.

End system ALM protocols enjoy more flexibility, adaptability to specific application domains and immediate deployment over the Internet but may not scale well (to large number of users or large number of simultaneous sessions), must deal with limited bandwidth of end systems and require end systems to take on some of the forwarding responsibility (and therefore increase application software development complexity).

2.3.3 Peer-to-peer substrate requirement

The success of peer-to-peer file sharing applications has resulted in the development of many peer-to-peer systems that organize the peers into an application-layer overlay and perform distributed content lookup over such a network or graph with the goal of scaling to support very large number of users (e.g. millions) [97] [91] [113] [86] [38] [17] [103]. Though more appropriate for distributed database and file-sharing applications, the peer-to-peer overlay can also provide a substrate on top of which various ALM protocols can be constructed. The peer-to-peer system takes care of the management of joining and leaving nodes (and possibly performing network measurements or inferring network

topology in order to build efficient overlays) as well as the routing between any two nodes in the network. An ALM protocol then only needs to construct a tree using the lookup service of the peer-to-peer network. The performance of such an ALM protocol is thus largely dependent on the characteristics of the peer-to-peer substrate on top of which it is constructed. The reader is referred to [12] [41] [69] [107] for some comparative studies of existing peer-to-peer systems.

Being geared towards scalable content location, generic peer-to-peer systems usually treat every peer equally, use latency as the primary metric and do not constrain the fan-out bandwidth of each peer. While these assumptions may be true for a dedicated proxy infrastructure, they may not be very appropriate for the often widely heterogeneous, bandwidth constraint end systems, making peer-to-peer substrate-based ALM suitable for providing generic multicast services at the infrastructure or peer-to-peer file sharing applications.

2.3.4 Group management

The basic group management service an ALM protocol provides consists of a mechanism for new nodes to discover a multicast session (typically a Rendezvous Point), a distributed or centralized, mesh-first or tree-first mechanism for constructing a single tree or a set of trees that are either source-specific or shared trees based on some metric. Such characteristics of the group management mechanism are primarily driven by the target application domain. For instance, single-source video streaming with large number of receivers usually involves a distributed group management and construction of a source-specific tree based on bandwidth and delay metrics, whereas medium-sized conferencing applications may involve the mesh-first construction of a shared tree based on bandwidth

and delay and can afford a centralized approach to group management.

Group management can involve the direct construction of the overlay tree (tree-first) or the construction of a richer overlay graph or mesh on top of which trees can be constructed (mesh-first). A mesh-first approach is more robust and responsive to tree partitions and is more suitable for multi-source applications but typically at the cost of higher control overhead.

Large-scale applications will typically require a distributed approach to group management (although there are still incentives for a centralized approach [81]) while small-scale applications can assume a centralized management of multicast groups. The choice between a shared tree and a source-specific tree depends on whether multiple sources will utilize the same overlay tree for data distribution (shared tree) or alternatively, if for each source a tree is constructed that is rooted at that source (source-specific tree).

Table 1 summarizes the characteristics of some of the main existing ALM protocols: NICE [6] ZIGZAG [101] SpreadIt [28] ALMI [82] CoopNet [81] TAG [62] Amcast [94] HMTP [112] Yoid [37] Bayeux [115] Gossamer [15] RMX [16] OMNI [7] RITA [108] Delaunay [66] Borg [111] Scribe [11] CAN-multicast [87] OverCast [54] HBM [90] TBCP [73].

Table 1. Characteristics of existing ALM protocols

					Group Management Characteristics				
ALM Protocol	Application Domain	Deployment Level	Peer-to-peer substrate required	Exploit IP Multicast	Centralized or distributed	Shared or source-specific tree	Metric	Tree-first or Mesh-first	Refinement
NICE	1	End-system	-	No	Distributed	Source-specific	Delay Bandwidth	Mesh-first [*]	Yes
Narada	2	End-system	-	No	Distributed	Source-specific	Delay Bandwidth	Mesh-first	Yes
ZIGZAG	1	End-system	-	No	Distributed	Source-specific	Delay Bandwidth	Mesh-first [*]	Yes
SpreadIt	1	End-system	-	No	Distributed	Source-specific	Bandwidth Delay	Tree-first	No
ALMI	2	End-system	-	No	Centralized	Shared	Delay	Tree-first	Yes
CoopNet	1	End-system	-	No	Centralized	Source-specific	Delay Bandwidth	Tree-first	No
TAG	1	End-system	-	No	Distributed	Source-specific	Topology Bandwidth	Tree-first	No
Yoid	3	End-system	-	Yes	Distributed	Shared	Bandwidth Delay	Tree-first [+]	Yes
TBCP	3	End-system	-	No	Distributed	Source-specific	Delay Bandwidth	Tree-first	No
HMTP	1	Proxy-based	-	Yes	Distributed	Shared	Delay	Tree-first	Yes
AMcast	1	Proxy-based	-	No	Centralized	Shared	Delay Bandwidth	Tree-first	No
Delaunay	3	Proxy-based	-	No	Distributed	Source-specific	Geographical position	Mesh [-]	No
OverCast	4	Proxy-based	-	No	Distributed	Source-specific	Bandwidth	Tree-first	No
OMNI	1	Proxy-based	-	Yes	Distributed	Source-specific	Delay Bandwidth	Tree-first	Yes
RITA	1	Proxy-based	-	No	Distributed	Source-specific	Delay Bandwidth	Mesh-first	Yes

Gossamer	3	Proxy-based	Scatter Cast	Yes	Distributed	Source-specific	Delay Bandwidth	Mesh-first	Yes
RMX	4	Proxy-based	Scatter Cast	Yes	Distributed	Shared	Delay Bandwidth	Mesh-first	Yes
Scribe	3	Proxy-based	Pastry	No	Distributed	Source-specific	Delay Bandwidth[&]	Mesh-first	Yes
CAN Multicast	3	Proxy-based	CAN	No	Distributed	Source-specific	Delay	Mesh*	No
Bayeux	1	Proxy-based	Tapestry	Yes	Distributed	Source-specific	Delay	Mesh-first	Yes
Borg	3	Proxy-based	Pastry	No	Distributed	Source-specific	Delay Bandwidth	Mesh-first	No
HBM	2	Combined	-	No	Centralized	Shared	Delay	Tree-first	Yes

1: media on demand streaming with large number of receivers

2: Conferencing with small to medium number of participants

3: generic multicast service

4: reliable data broadcast and file-transfer

[*] A hierarchical cluster of nodes is constructed and a tree is built on top of this hierarchy

[+] A mesh also exists but is constructed after the tree

[-] Using Delaunay triangulation, an explicit tree constructing algorithm is not actually needed

[~] There is really no tree but just a mesh since nodes can receive from multiple parents

[=] Delay, based on topology

[&] since the outdegree of a node is potentially unbounded, they see if a node is overwhelmed and try to offload

* CAN-Multicast doesn't make a tree (duplicate copies)

The list of ALM protocols in Table 1 is not an exhaustive one, since it focuses on earlier efforts at the exclusion of the plethora of protocols that have since emerged, yet serves to illustrate the diversity of approaches to ALM. More recent endeavors examine the issue of trust in overlay networks [64] [74], handling heterogeneity of users [109], providing resilience [2] [69], taking into account node priority [106] or node availability [42] as well as high-bandwidth file transfer and downloading [60] [93] [13].

In addition to providing an introduction to ALM as background to the work presented in

this thesis, surveying the existing approaches to ALM helps in making two observations. Implementing multicasting at the application layer allows for the design of a myriad of protocols, each specific to a particular class of applications. These protocols are not necessarily in competition with one another since they address different application domains. Furthermore, protocols targeting the same type of application can co-exist as long as they provide different services. For example, a proxy-based ALM protocol can provide multicasting service to those developers of single-source streaming applications that are willing to incur the financial cost of using a network of proxies while an end-system ALM protocol for the same application can be available to those that wish to receive no support from dedicated proxies in return for not paying for it. This is in contrast to the network layer provision of multicasting service, where it is necessary for router manufacturers to agree on a set of multicast protocols in order to provide the service across different networks.

The second observation pertinent to our work is that most existing ALM protocols target single-source streaming applications with large number of receivers with the exception of Narada [22], ALMI [82] and HBM [90]. Although targeting small group conferences, Narada assumes that a single source will transmit at any given point (and hence constructs a single tree). HBM does not constrain the out-degree of each node while ALMI constructs a single shared tree for the conference. If multiple sources simultaneously use the same shared tree, then each node's out-degree will not be respected.

Recalling the characteristics and requirements of multi-party videoconferencing applications (section 2.2) to include the need to support multiple simultaneous sources

while respecting the outflow limitations of each node, we observe the inadequacy of existing protocols with respect to multi-party videoconferencing applications such as those presented in section 2.1.3.

This point becomes more evident when examining common routing algorithms employed by existing approaches, as presented in the next section.

2.4 *Overlay Routing Heuristics*

An important part of any ALM protocol is the routing mechanism that determines the multicast tree through which data is distributed. The routing mechanism must construct a structure of nodes and determine the path that should be taken by the data from the source to its intended receivers. As shown in Table 1, an ALM protocol's group management and hence its routing can be distributed or centralized and can use different metrics to construct the data dissemination structures. The choice of the routing mechanism is dependent on the application domain and the deployment level of the protocol. Since the ALM protocol presented in this thesis includes a novel routing heuristic for overlay networks, we devote this section to an overview of routing algorithms used by existing ALM protocols. We begin the overview by providing a basic introduction to the relevant terminology.

2.4.1 Terminology

The terminology and definitions given here are all taken from pages 86 to 94 of "Introduction to Algorithms" by T.H Cormen, C.E. Leiserson and R.L. Rivest [24].

An **undirected graph** G is a pair (V,E), where V is a finite set and E is an unordered binary relation on V. A **directed graph** is one with ordered binary relations on V: that is (a,b) is not equivalent to (b,a). The set V is called the **vertex** set of G, and its elements are

called **vertices**. The set E is called the edge set of G, and its elements are called **edges**.

If (u,v) is an edge in an undirected graph G=(V,E), we say that vertex u is **adjacent** to u and vice versa. A **complete graph** is an undirected graph in which every pair of vertices is adjacent. The **degree** of a vertex in an undirected graph is the number of edges incident on it. In a directed graph, the **out-degree** of a vertex is the number of edges leaving it and the **in-degree** of a vertex is the number of edges entering it.

A **path** from a vertex u_1 to a vertex u_2 in a graph G=(V,E) is a sequence $<v_0,v_1,v_2,\ldots.v_k>$ of vertices such that $u_1=v_0$, $u_2=v_k$, and $(v_{i-1},v_i)\in E$ for i=1,2,...k. An undirected graph is **connected** if every pair of vertices is connected by a path. In an undirected graph, a path $<v_0,v_1,v_2,\ldots.,v_k>$ forms a **cycle** if $v_0=v_k$ and $v_1,v_2,\ldots,v_k$ are distinct. A graph with no cycles is **acyclic**. A connected acyclic undirected graph is a **tree**.

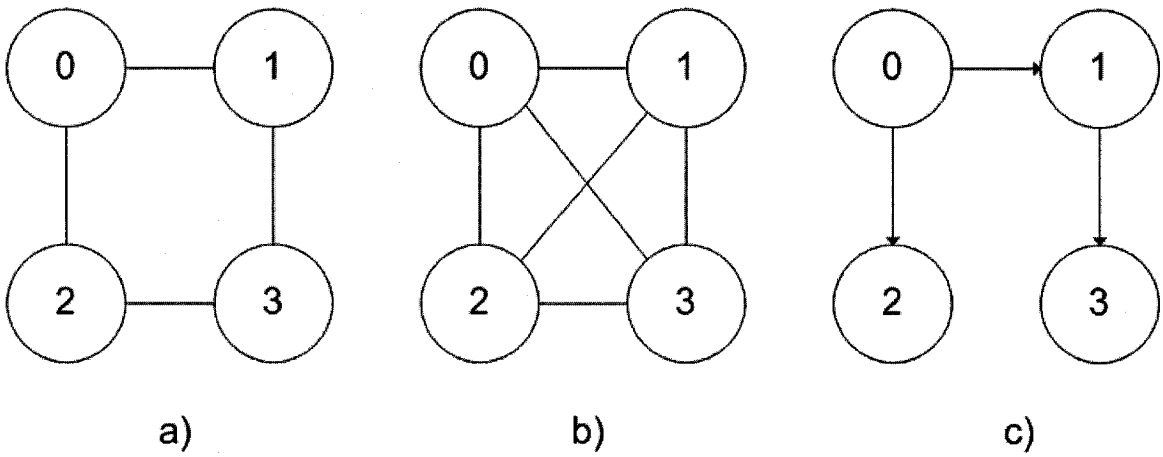


Figure 7. Undirected and complete graphs and a directed tree

A **rooted tree** is a tree in which one of the vertices is distinguished from the others. The distinguished vertex is called the **root** of the tree. A vertex of a rooted tree is a **node** of the tree. Consider a node x in a rooted tree T with root r. Any node y on the unique path from r to x is called an **ancestor** of x. If y is an ancestor of x, then x is a **descendant** of y. If the last edge on the path from the root r of a tree T to a node x is (y,x), then y is the

parent of x and x is a **child** of y. A node with no children is a **leaf** node.

Figure 7 illustrates some of the terminology with examples: Figure 7a is an undirected graph with vertices (0,1,2,3) and edges {(0,1),(1,3),(2,3),(0,2)}. Vertices 0 and 1 are adjacent. Figure 7b shows a complete graph and Figure 7c shows a directed tree rooted at node 0. Node 1 is parent of node 3 and node 2 is child of node 0. Out-degree of node0 is two while in-degree of node 3 is one.

Coming back to the media-streaming scenario, it can be seen that the routing mechanism of an ALM protocol involves the construction of a tree rooted at a source that has a path to every one of its intended receivers. If a cost metric such as delay is associated with each edge of the graph, then the protocol may desire to construct a *minimum-cost* rooted tree. Additionally, if every node in the tree has a constraint on the number of children it can support, then an *out-degree constraint* tree is appropriate. In other words, the delay parameter can be represented as a cost of each edge and the bandwidth parameter can be represented as constraints on the in-degree and out-degree of each node.

Having presented a basic introduction to graph theoretic terms and concepts, we can now proceed with giving an overview of some of the most common routing algorithms used by existing ALM protocols.

2.4.2 Overlay routing heuristic overview

Before giving an overview of common overlay multicast (i.e. ALM) routing heuristics, it is important to examine the fundamental differences between overlay multicast routing and IP Multicast routing. The differences stem from the fact that routing in an IP Multicast session involves network entities (i.e. routers) communicating with one another to set up a multicast tree, while multicast routing in overlay network involves application

entities organizing themselves into a multicast tree on top of a network of routers (Figure 8).

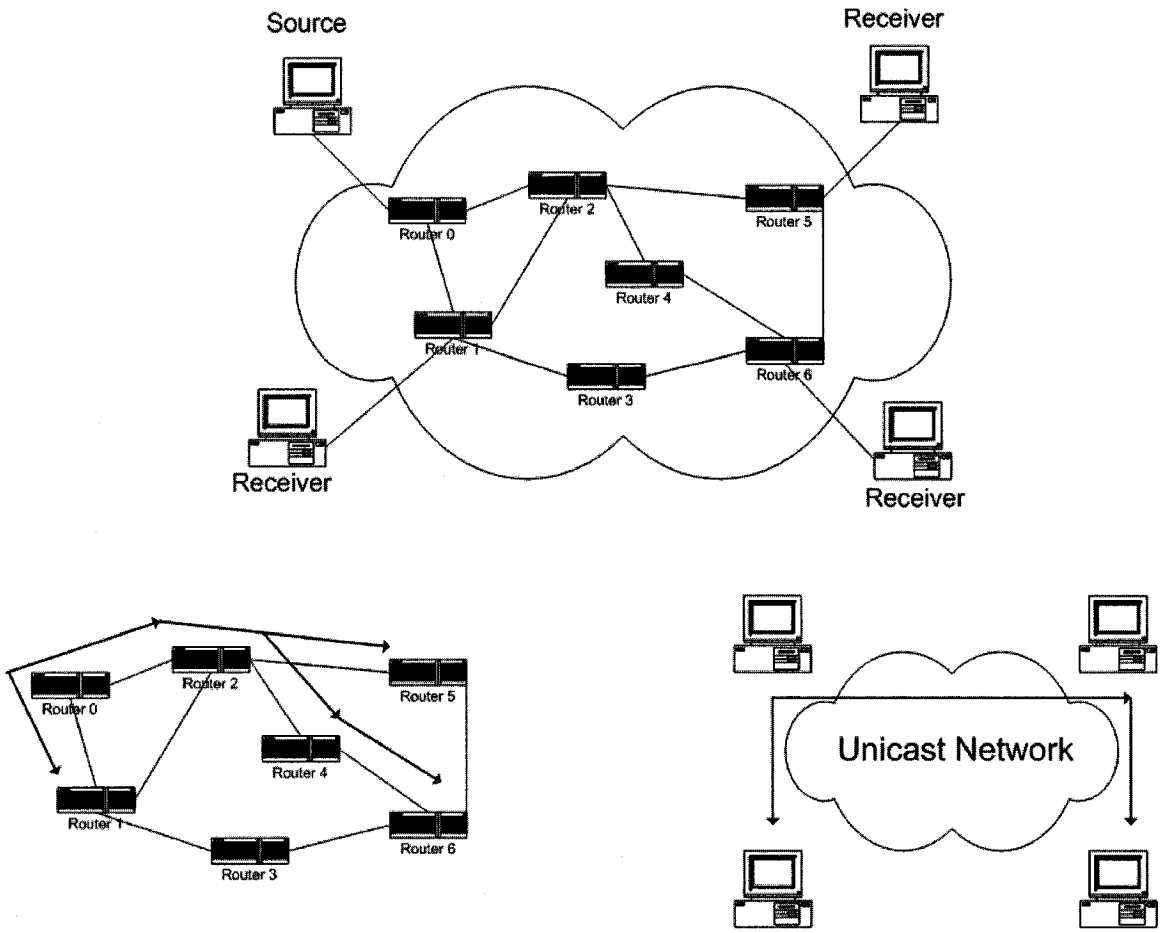


Figure 8. Network-level multicasting (left) compared to application-level multicasting (right)

Whereas a router in the network is connected to only a subset of other routers in the network, an application entity can potentially have a virtual connection to all other application entities on top of the underlying network. In graph theoretic terms, overlay multicast routing can assume to be operating on top of a *complete graph* (i.e. a graph where there is an edge between any two nodes) whereas such an assumption is invalid for IP Multicast routing. Another important distinction is that by virtue of dealing with a very large number of routers, an IP Multicast protocol must be scalable and distributed,

whereas an overlay network can consist of a small number of nodes whose organization can be determined in a central manner.

Since network components such as routers in the Internet backbone perform the data duplication necessary for multicasting, in IP Multicast it is reasonably assumed that such backbone routers have ample bandwidth available to them. As a result, an IP Multicast routing protocol would rarely consider out-degree constraints on the data duplicating nodes and instead would focus on creating efficient and low delay trees using bandwidth-intensive operations such as link state flooding. In contrast, overlay multicasting involves application entities duplicating data and since the bandwidth available to these (be they end systems or proxies) is more limited, it is important for the corresponding routing mechanism to consider degree limitations of each node, which in turn has an effect on the routing algorithms that are employed.

The aforementioned differences restrict the applicability of established IP Multicast routing protocols such as DVMRP, MOSPF, PIM-DM, CBT [59] among others and necessitate the use and design of other routing protocols for the overlay multicast setting.

As mentioned already, existing ALM protocols widely vary in their approach, the application domain they try to address and the constraints and parameters they consider. Such is the case also with the routing mechanism of each ALM protocol varying from one implementation to another. Here we will present some of the more common routing mechanisms used in ALM protocols.

2.4.3 Minimum Spanning Tree

Given a graph with a cost associated with each edge (usually delay), a Minimum Spanning Tree (MST) is a tree with minimum total cost (see Chapter 24 of [24]] for

Kruskal and Prim's algorithms for building MSTs). Given the graph with edge costs shown in Figure 9a, an MST is constructed to have the minimum total cost as shown in Figure 9b (total cost is 11 in this example).

An MST is commonly used by a centralized ALM protocol such as ALMI and HBM in order to construct a low cost shared tree that is not rooted at any particular source (a shared tree implies that all nodes use the same tree to distribute their data).

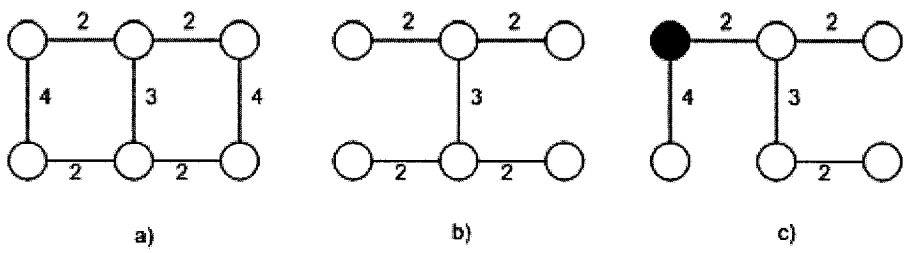


Figure 9. a) a graph with link costs b) Minimum Spanning Tree c) Shortest Path Tree

2.4.4 Shortest Path Tree

A Shortest Path Tree (SPT) on the other hand constructs a minimum cost path from a source node to all its receivers (see chapter 25 of [24] for Dijkstra's algorithm for building SPTs). An SPT or one of its variants is commonly used by ALM protocols (such as Yoid, SpreadIt, TAG, RITA) in order to construct a source-specific multicast tree or in graph theoretic terms a rooted tree. Figure 9c shows the SPT rooted at the filled-in node. It is important to note that both MST and SPT can be modified to respect degree constraints of each node [9] [94].

2.4.5 Hierarchical Clusters

In order to better organize the overlay tree and reduce control message overhead, some ALM protocols such as ZIGZAG and NICE construct a hierarchical cluster of nodes with each cluster having a 'head' representing it in the higher layer.

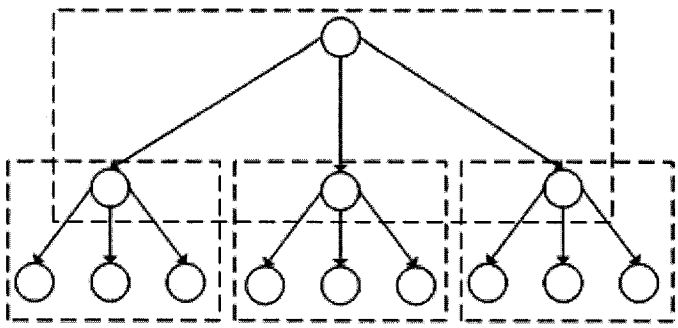


Figure 10. A hierarchical cluster of nodes with cluster size 4

The advantage of a hierarchical cluster approach to multicast tree routing is the reduction in control overhead (nodes keep state only about a subset of other nodes) and faster joining and management of the tree at the cost of a sub-optimal tree and a lack of hard guarantees on the degree limitation of each node.

2.4.6 Peer-to-peer substrate based routing

Recalling Table 1 of section 2.3.4, we observe that many ALM protocols (such as RMX, Gossamer, Bayeux, Borg, Scribe) operate based on an existing peer-to-peer substrate that serves as a mesh on top of which an overlay multicast tree can be constructed using either a reverse-path forwarding scheme (Gossamer, RMX, Scribe), a forward-path forwarding scheme (Bayeux) or both (Borg). The advantage of these approaches includes low control overhead and distributed management of the multicast tree but they do not restrict the degree of each node and are sub-optimal.

2.4.7 Degree-constrained routing

As noted earlier, MST and SPT routing algorithms can be modified to respect degree constraints of each node [9]. Additionally, Shi et al [94] examine several degree constrained routing heuristics and introduce their own.

The problem of finding minimum-cost degree-constrained multicast trees or degree-constrained Steiner trees is NP-complete due to containing the NP-complete problem of

finding a degree-constrained spanning tree [31]. There exist several heuristic approximation algorithms addressing this problem [9] [72] [57] [58] [85]. Some of these algorithms (such as [57] [58]) do not provide exact guarantees on the degree of each node in the tree and instead provide a bound on the worst-case degree. Others focus on constructing a *single* tree and do not consider multiple trees over the same graph ([9] [85] [95]). Recalling that a multi-party videoconferencing application involves multiple simultaneous sources and therefore multiple overlay trees, we will demonstrate through examples (see section 4.4) and simulations (see Chapter 5) that existing heuristics suffer as a result of constructing each tree independently of the others.

Though there has been some research with regards to constructing multiple trees on a shared graph [20], they still only provide a bound on the worst case (maximum) degree of *any* node as opposed to guarantees on the individual maximum degree for *every* node as is required for a protocol supporting multi-party videoconferencing applications.

It is towards the aim of producing heuristics that are appropriate for multi-source applications that we introduce our own multi-source outflow-constrained overlay tree heuristic and compare its performance with relevant heuristics as well as with the only other heuristic that considers the construction of multiple overlay trees [68] (see Chapter 5 for details of the comparison) and find that our heuristic performs better in finding a significantly larger percentage of solutions, while providing for heterogeneous bit-rates, priority as well as incurring a low delay penalty.

2.5 *Summary*

This section presented the basic concept behind Application Layer Multicasting and gave a survey of existing protocols through which we observed that most existing approaches

target single-source streaming applications with large number of receivers and very few target small-scale multi-party videoconferencing applications. As a result, the services they provide as well as the constraints and parameters they consider are not adequate for the class of applications that we are interested in and presented in section 2.1.3. This is especially evident from the routing heuristics employed by existing approaches that commonly construct a single tree and do not necessarily consider out-degree constraints of each node. We have therefore devised our own protocol called MVEMP (Multi-party Videoconferencing End-system Multicast Protocol) as well as heuristic routing algorithms that will be presented in Chapter 3 and Chapter 4 respectively.

Chapter 3 MVEMP: Multi-party Videoconferencing End-system Multicast Protocol

The previous chapters helped highlight the characteristics and requirements of multi-party videoconferencing applications especially with regards to multicasting and went on to illustrate that existing ALM protocols do not adequately address such applications. This chapter presents MVEMP: a Multi-party Videoconferencing End-system Multicast Protocol. The goal of MVEMP is to assist the deployment of multi-party videoconferencing applications on top of today's Internet through constructing overlay trees rooted at each video source in the session.

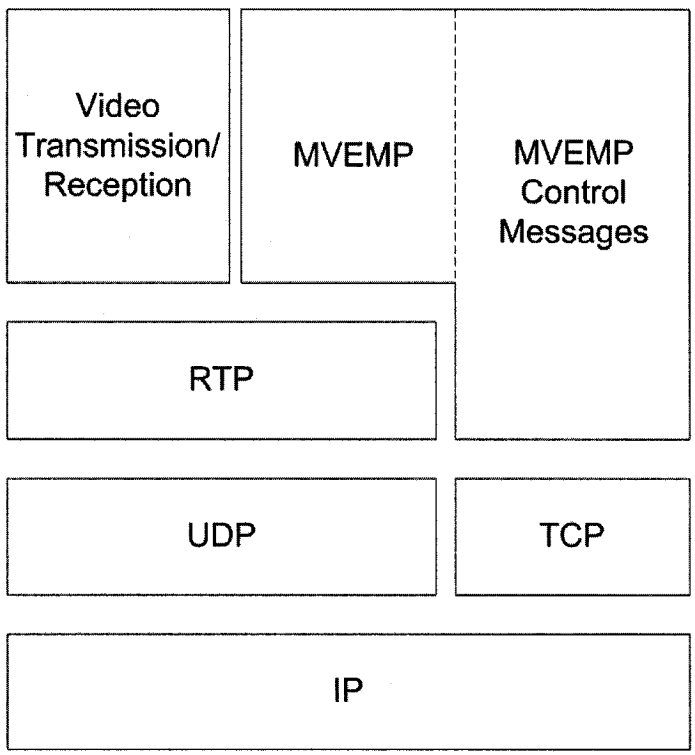


Figure 11. MVEMP in the protocol stack

It therefore directs the transmission and forwarding of video streams from one participant's computer to another. As shown in Figure 11, MVEMP works on top of an

RTP/UDP/IP protocol stack and provides a multicasting service to the application by directing the transmission and reception of video streams. Since it has been shown that measuring throughput bandwidth and delay can be significantly beneficial for ALM routing [22], future versions of MVEMP would have to incorporate the control messages received via RTCP in order to get an idea of the quality of the media received by each user and adapting the overlay tree accordingly. The control messages exchanged between nodes executing the MVEMP protocol require reliable delivery and therefore traverse a TCP/IP protocol stack.

The core design of any ESM protocol consists of 1) the choice between a distributed or centralized scheme for constructing data distribution trees or hierarchies 2) the choice between a mesh-first approach or a tree-first approach to constructing such trees 3) definition of messages the various entities must exchange in order to provide the necessary services and 4) the algorithm and mechanism for constructing, improving and maintaining trees when users join/leave trees rooted at various sources. What follows is our design of an ESM protocol and the choices therein as they relate to the characteristics and requirements of multi-party videoconferences.

3.1 Centralized vs. Distributed

An ESM protocol must choose between a centralized approach and a distributed approach to constructing, improving and maintaining trees or hierarchies representing the multicast group. While a distributed approach provides better scalability and fault tolerance by distributing the decision making process between many nodes, it is slower than a centralized approach in reacting to changes (e.g. nodes joining or leaving the hierarchy) since a joining node must iteratively search for a node to connect to. It is this iterative

search that causes continuous packet loss for seconds and sometimes tens of seconds when, for instance, an intermediate node leaves [6]. At the cost of creating a single point of failure and its limited scalability, a centralized approach can find more solutions to the routing problem (as a result of having more information) and arrive at those solutions faster than a distributed approach. As mentioned in section 2.2.3, virtual environment based multi-party videoconferencing applications can employ some form of interest/awareness management to limit the number of users wishing to receive from a source. Furthermore, given their lesser scalability requirement and the frequently changing characteristic of multi-sender teleconferences (as compared to video-on-demand and live broadcast), we therefore opt for a centralized approach similar to ALMI [82] for constructing and maintaining overlay structures. We assume the existence of a Rendezvous Point (RP) whose identity is known to all participants prior to the start of the session.

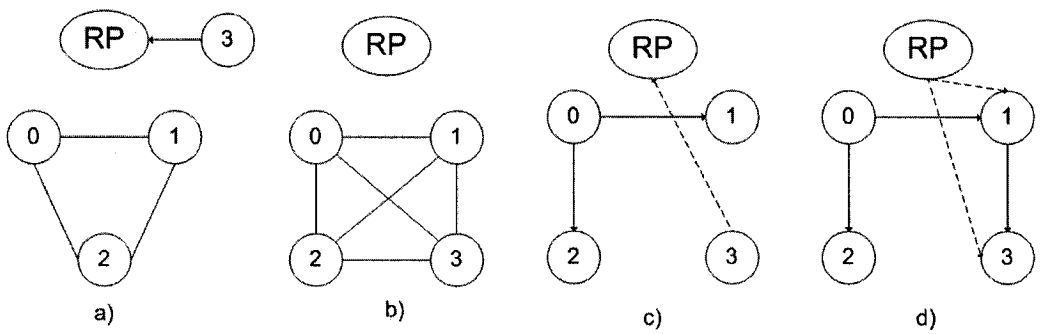


Figure 12. RP as a point of connection (a-b) and holding the centralized routing mechanism (c-d)

Similar to other protocols the RP serves as a bootstrap mechanism whereby participants contact in order to join the session but, additionally, in our protocol the RP also contains

the centralized decision making component that keeps track of the overlay tree for all sources. Figure 12 illustrates the role of the RP.

In Figure 12a, a new node joining the session finds the IP of nodes already in the session from the RP and connects to them (Figure 12b). In Figure 12c, node3 declares its wish to be attached to the tree rooted at source0. RP determines that node3 should be attached as a child of node1 and informs both about the new tree branch to be constructed (Figure 12d). Note that the RP would physically reside on one of the client machines.

Clients inform the RP of their wish to join/leave a particular source and based on the algorithms that will be discussed in the next chapter, the RP decides how to modify the overlay trees in order to accommodate every source sending to every one of its receivers (Note that it is the decision making process that is centralized and not the stream distribution for each source which is done in a peer-to-peer manner). The RP then informs the affected nodes as to how the overlay tree is to be constructed.

3.2 Mesh-first vs. Tree-first

Existing ESM protocols either construct trees rooted at a source directly from the current list of nodes or construct a richer connected graph called a mesh and then construct overlay trees based on the graph. In a tree-first approach, new nodes wishing to join the overlay tree are directly attached to the tree. In contrast, a mesh-first approach requires node to first establish connections with several other nodes and then to get connected to the overlay tree through a subset of these. The extra connections in a mesh help a disconnected node to reconnect quicker. A tree-first approach is more appropriate for single source applications and requires less control overhead since each node only exchanges information with its parent and children. A mesh-first approach typically

incurs a higher control overhead (the exception being [66] where a mesh implicitly contains the trees and therefore a tree building algorithm is not necessary) by requiring each node to have a virtual connection to nodes other than its parent and children but consequently can more quickly attach a node to another overlay tree. As a result, a mesh-first approach fits well within the context of multi-party videoconferencing applications. In our protocol the mesh serves as the graph on top of which multiple overlay trees are constructed.

3.3 *Protocol Description*

Having decided on a mesh-first centralized approach to constructing, improving and maintaining overlay trees for the multi-party videoconferences, we can now present how such a protocol provides the basic services required. They are:

- Managing a session by allowing new participants to join and leave the conference session
- Managing the construction of overlay trees to support the multicast data distribution, as users change the set of media sources they wish to receive from

Although there are other events that the protocol must handle (and these will be discussed towards the end of this section), the core operation of the protocol consists of dealing with four events:

1. New Node Join event: a new participant joins the conferencing session
2. Node Leave event: a conference session participant either leaves the session or experiences failure
3. Soft Join event: a participant already part of the conferencing session expresses its interest in receiving video from a particular source

4. Soft Leave event: a participant already part of the conferencing session and receiving video from a source wishes to discontinue receiving from that source and yet still remain part of the session

It is important to make the distinction between 'New Join Node' and 'Node Leave' events and what we have termed *'Soft Join'* and *'Soft Leave'* events [47]. Commonly referred to as simply join and leave events in existing ALM protocols, 'New Join Node' and 'Node Leave' imply construction and tearing down of connections between nodes. This is in contrast to Soft Join and Soft Leave events that refer to an existing node in the session changing the set of media sources it wishes to receive from and does not necessarily imply tearing down or construction of links but the resumption or discontinuation of stream transmission between already connected peers. It is obvious that Soft Join and Soft Leave events only apply to multi-source applications.

3.3.1 New Node Join event

Participants join a teleconferencing session by contacting the Rendezvous Point (RP) whose address is known through a mechanism outside of this protocol (e.g. URL is obtained from a website). A TCP connection is established between each joining node and the RP to allow for reliable bi-directional control message passing. Each joining node provides the RP with its maximum outflow bandwidth (obtained either through measurement or explicitly defined by each participant) as well as the maximum video bit-rate of the video it will be transmitting.

The RP holds the IP address of every participant currently part of the session as well as their maximum outflow bandwidth. The RP informs the joining participant of the address of those already in the session, as well as the ports they will be using for transmission of

their video and assigns the joining node the ports it should use for transmission of its own video.

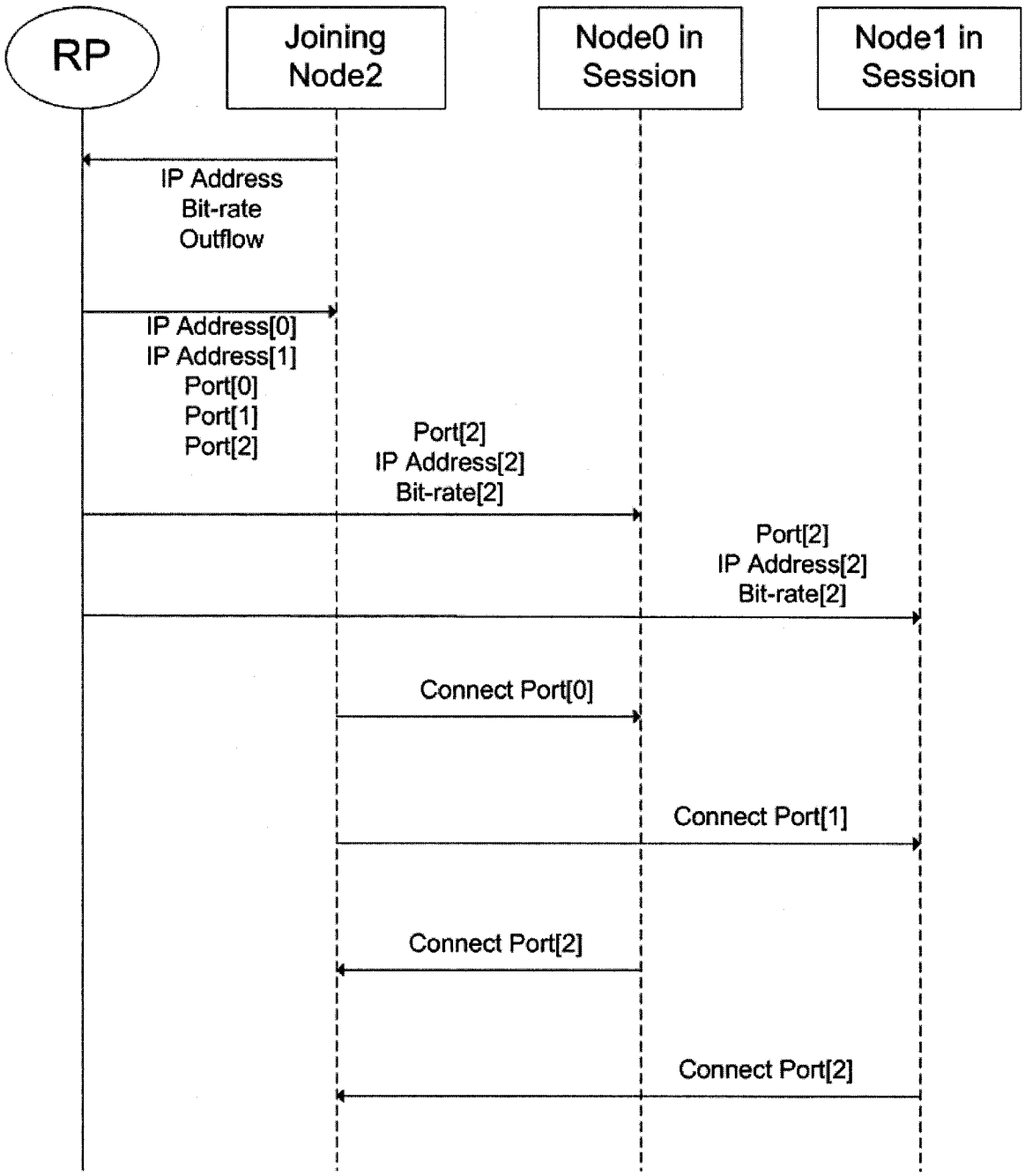


Figure 13. New node 2 joining nodes 0 and 1 already in the session

The joining participant then makes a RTP/UDP connection to all existing participants and vice versa, thus making a bi-directional Complete Virtual Graph (CVG) between the

nodes. It is a virtual graph since each node is physically connected through a series of other nodes (i.e. routers).

Figure 13 illustrates the New Node Join event process with an example: nodes 0 and 1 are already part of the session with the RP aware of their IP addresses and the ports they use for data distribution. Node 2 joining the session contacts the RP and provides its own IP address and possibly some additional information such as Internet connection (e.g. DSL, dial-up). The RP informs the new node of the existing nodes and provides the IP address and ports necessary for the new node to establish connections with the existing nodes. The RP also informs the existing nodes of the new node so two-way connections can be established.

3.3.2 Node Leave

A node gracefully leaving a session informs the RP of doing so. The RP in turn informs every other node about the leaving node. Furthermore, the RP repairs any disconnected trees resulting from the leaving node by generating a Soft Join event for each node whose parent is the leaving node (Figure 14).

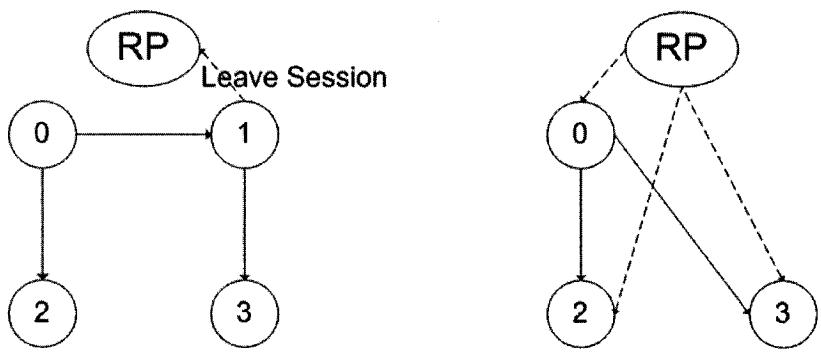


Figure 14. Node Leave

Similar to other ESM protocols, periodic 'keep alive' messages exchanged between a node and the RP are necessary to detect abrupt failure of a node. Unlike other protocols however, MVEMP can make use of the fact that each node will frequently send 'Soft

Join' and 'Soft Leave' messages to the RP and will receive forwarding instructions as a result of other node's 'Soft Join' and 'Soft Leave' events. Therefore 'keep alive' messages are exchanged only when a node has not transmitted a message to the RP after a given threshold. In fact, given the frequency at which participants exchange information with the RP and other messages that are exchanged between peers such as their movements (see section 6.4), it may be reasonable to forego the exchange of periodic 'keep alive' messages altogether and instead rely on other types of messages to serve as a form of 'keep alive' message.

Alternatively, quality-monitoring information, obtained from the RTCP back channel, can be periodically sent along side the 'keep-alive' messages to the RP in order for it to keep an updated status of each node's received media quality.

Once a node failure is detected, the procedure followed by the RP is the same as a node gracefully leaving the session: RP informs other nodes of the leaving node and repair disconnected trees, if any, by generating Soft Join events for each node whose parent is the leaving node.

It is important to note that if the failing or leaving node happens to be the client hosting the RP, then the entire session fails. The remaining participants must then restart the session by either contacting the same RP (if it becomes available) or executing another instance of the RP between themselves and joining the new conference session. In this regard, the RP presents a single point of failure in our protocol.

3.3.3 Soft Join

Once a node has successfully joined the conference session and is part of the mesh, it can request to receive data from a set of sources in the session (this would normally be a

subset of all sources if interest/awareness management is used). In other words, a node in the session can at any time specify the set of video sources it wishes to receive from based on its interest/awareness of each source.

The job of the RP is to attach the requesting node to the overlay tree of the sources it requests to join. The idea then is to find a set of trees rooted at each source and spanning the subset of users it needs to send to, given a maximum outflow bound for each node. It is also desirable to reduce the delay between the source and its receivers. Furthermore, each node may assign a priority to the sources it wishes to receive from to signify their relative importance as well as specify a particular bit-rate of the video it would wish to receive. This is the key functionality and contribution of the MVEMP protocol and is represented by novel heuristic routing algorithms to be presented in the next chapter. The goal of the heuristic is to find low delay outflow-constrained overlay trees for each source rooted at that source in order to satisfy the Soft Join request of each node. The routing heuristic generates the set of changes in the overlay tree structure necessary to accommodate each Soft Join requests and possibly a set of requests that cannot be accommodated and are therefore rejected (section 3.3.5 deals with rejected requests). The RP then communicates these changes to the affected nodes, which must then transmit or forward video to other nodes as requested by the RP.

3.3.4 Soft Leave

Just as a conference participant can request to receive a video from a particular source (i.e. Soft Join), he/she can request to discontinue the transmission of a video he/she is receiving. The Soft Leave event is thus generated when a node loses interest or awareness in another node and wishes to discontinue its reception of video from that node, in which

case the RP is informed of the change.

The job of the RP is then to repair any overlay tree that may become disconnected as a result of the requesting node leaving the tree. We take a simplistic approach to a Soft Leave event. If the requesting node has more than one child in the tree of the source, we ignore the Soft Leave request and use the requesting node as a reflector (a node that does not itself wish to receive a stream and yet is used to forward to more than one node, see section 4.5).

If, however, the node making the Soft Leave request has one or no children in the tree of the source, it is simply removed from the tree and the disconnected child reconnected to the parent of the leaving node.

3.3.5 Reject Request

There are instances when the protocol is unable to satisfy a 'Soft Join' request without violating the constraints specified by each peer. For example, there may not be enough bandwidth to accommodate a new user on an overlay tree. Under such circumstances, the heuristic rejects a set of requests according to priority of each request if priorities have been assigned. It is then the job of the RP to inform any affected node of its request having been rejected.

3.3.6 Port Management

Due to the existence of multiple streams in the session, it is important to properly manage the ports each node uses to transmit and receive streams. As mentioned in section 3.3.1, the RP assigns each joining node the ports it should use for transmission of its video. There are instances when a node may be required to transmit more than one stream to another node. For example, a node may transmit its own video to another node, as well as forward another source's video to that node. Such occurrences can be accommodated by

either multiplexing the streams or opening an additional socket. In order to reduce the complexity of implementation involved in multiplexing, we have opted for the latter approach and therefore require the RP to inform both the transmitting and receiving nodes of the additional ports they should use. Note that the alternative to use multiplexing can still implemented as an extension of the protocol.

In order to avoid a large number of ports from being open on every participant's machine, the RP sends a message to close a port that has remained idle for a specified period of time (in order of minutes).

3.4 Summary

This chapter presented a protocol designed for multi-party videoconferencing, which involves the mechanisms through which nodes join and leave a conferencing session, specify their requirements and capabilities as well as, during that session, request to receive video streams from a set of sources. We did not however present how our protocol constructs overlay trees for each source, given the outflow constraints of each node. The overlay multicast routing algorithm employed in our protocol presents one of the key contributions of this thesis and will be presented in the next chapter.

Chapter 4 Multi-source Outflow-constrained Overlay Routing

One of the key functionalities of any ESM protocol is the overlay multicast routing algorithm based on which nodes are organized into a distribution tree rooted at the source. The routing algorithm determines the quality of the distribution tree and therefore the quality of experience in the media sharing session as well as the scalability, ease of deployment and overall feasibility of the ESM protocol. Section 2.4.7 presented some of the most common routing approaches employed by existing ALM protocols, some of which did not respect individual out-degree constraints of each node and all of which constructed a single tree and did not consider multiple trees on the same graph. In this regard, we first demonstrate the shortcoming of existing approaches to overlay routing and go on to present our own routing heuristic called MOTOR: Multi-source Outflow-constrained Tree Overlay Routing.

Given the multi-party videoconferencing requirements outlined in section 2.2 and the centralized group management presented in the previous chapter, we can define the routing problem as: given a set of users connected to one another via a unicast infrastructure, each with a limited upload bandwidth, we must construct for every source a tree rooted at that source and spanning its intended receivers without violating the bandwidth limitation of any user. Furthermore, each stream may have, according to the receiver's preference or capability, a different bit-rate as well as a priority assigned to it. The overlay multicast tree should be constructed such that every node forwarding a

stream of a specific bit-rate should be receiving that stream with an equal or greater bit-rate. If it is not possible to construct such a tree, lower priority streams should be dropped first. Finally, it is desirable for the constructed trees to be low cost where the cost of an edge connecting two nodes represents the delay between them.

4.1 Problem Definition

More formally, we define the following:

- G=(V,E): An undirected complete graph with nodes V and edges E
- c(i,j): cost associated with edge $\{i,j\} \in E$, representing delay on that edge
- outflow_max(v): the maximum allowable out-flow of node $v \in V$
- outflow(v): current out-flow of node $v \in V$
- $Rs=\{t_1,\ldots t_M\}$: a possibly empty set of nodes $t_i \in V$ for every node $s \in V$ (Rs represent the receiver set for source s)
- p(i,j): priority level of a request from source i to receiver j
- fw(i,j): requested flow weight from node i to j (represents bit-rate requested)
- Ts: a tree rooted at node s and spanning Rs
- $\succ$ operator in Ts: node c is the child of node d in tree Ts if $c \succ d$

The Minimum Diameter Multi-Source Outflow-Constrained (MD-MSOC) routing problem is then defined as:

Given graph G, outflow_max(v) for every node $v \in V$, Rs for every node $s \in V$, p(i,s) and fw(i,s) for every pair $i,s \in V$, find a tree Ts rooted at every node $s \in V$, spanning its receiver set Rs such that:

$outflow(v) \leq outflow\_max(v)$ *for all* $v \in V$

if $i \succ j$ *then* $fw(i,s) \leq fw(j,s)$ *for* $i,j \in Ts$ *for all* $s \in V$

$max(c(e_i)) < \varepsilon$ *for all* $e_i \in Ts$

In graph theoretic terms, this is an extension of the minimum-cost degree constraint problem: The MD-MSOC problem requires the building of *multiple* minimum-cost degree-constraint trees over the same graph.

4.1.1 Outflow versus out-degree

The distinction we have made here between outflow and out-degree is worthy of note. As defined in section 2.4.1, the **degree** of a vertex in an undirected graph is the number of edges incident on it. In a directed graph, the **out-degree** of a vertex is the number of edges leaving it and the **in-degree** of a vertex is the number of edges entering it. The definition of degree is sufficient to represent the bandwidth constraint of nodes when dealing with a tree with all nodes receiving the same flow.

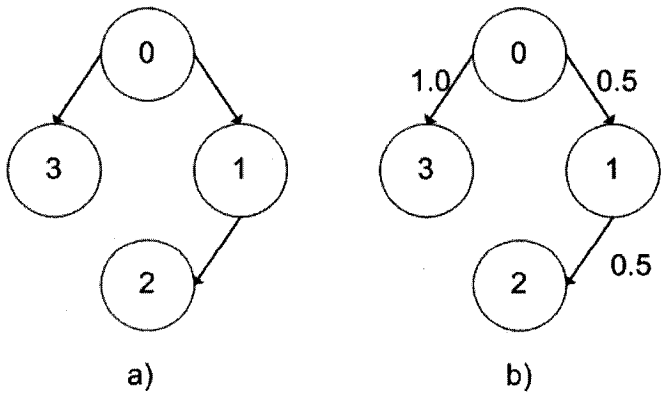


Figure 15. Out-degree versus outflow

When constructing trees for sources where the bit-rate associated with each receiver can

be different, this definition does not adequately represent the problem domain. Figure 15 illustrates with an example. The out-degree of node0 is 2 in Figure 15a since there are a total of 2 edges leaving it with both receivers 1 and 3 receiving the same normalized flow of 1. In Figure 15b different receivers can receive different flow weights (i.e. bit-rates) and although there are 2 edges leaving node 1, the outflow of that node is 1.5 since receiver 1 receives a flow weight of 0.5 (i.e. the bit-rate received by node 1 is half the bit-rate received by node 3). In order to be able to deal with cases where the bit-rate of each receiver can be different, we consider the outgoing bandwidth constraints of each node as an **outflow** constraint rather than an out-degree constraint. In the special case where all receivers have the same bit-rate, the outflow-constraint problem reduces to an out-degree-constraint problem. Note that other heuristics cannot deal with the most generic formulation of the problem (with heterogeneous flow weights for receivers) by virtue of providing out-degree constrained trees rather than outflow constrained trees.

It is worthy of note that unlike the end-to-end delay between two nodes in a network, measuring throughput bandwidth between two nodes is non-trivial. Section 7.3 further discusses the issue of measuring available bandwidth on the Internet.

4.1.2 Restricting inflow

It is important to note that the MD-MSOC problem does not constrain the inflow of each node. Similar to the **outflow** constraints representing the **upload** bandwidth limitations of each node, the **inflow** constraints represent the **download** bandwidth limitations of each node. The rationale for not considering the inflow constraint of each node is derived from the ESM protocol that the routing heuristic is a part of (i.e. MVEMP as presented in the previous chapter) and the application domain targeted by this protocol (i.e. multi-party

videoconferencing). Recall from section 2.1.3 that a multi-party videoconferencing application *must* employ an application-layer filtering or awareness management mechanism in order to limit the number of media streams it deals with. In fact, such a mechanism is assumed to exist and is a pre-condition for the use of an ESM protocol as discussed in section 2.2.3 and demonstrated in section 6.4. The application-layer restricts the total number of media streams by restricting the number of streams each user can request to receive. In other words, the fact that an ESM protocol (and hence an overlay routing algorithm) is being used precipitates from the existence of an application-layer filter that restricts the inflow of each node. As shown in [47] and demonstrated in section 6.4, restricting the inflow of each node at the application layer does not necessarily imply restricting the outflow. Therefore, the routing heuristics presented in this thesis focus on the outflow constraints of each node and assume that the inflow is constraint by the application itself.

As a final note, it is important to highlight the fact that today's Asymmetric Digital Subscriber Lines (ADSL) that provide Internet access to home users have asymmetric download and upload bandwidth constraints. A typical ADSL line for instance has a maximum download bandwidth of 1Mbps while the upload bandwidth is only a few 100kbps. Considering a 320×240 MPEG4 coded video at 15fps which produces an average of 100kbps as an example, we can observe that a typical home Internet user is able to receive several video streams but may not be able to transmit more than 1 or two streams. The more restricted constraint of the upload bandwidth as compared to download bandwidth is another reason why outflow forms the major constraint in the MD-MSOC problem presented. Others have used the asymmetric nature of ADSL lines

as a basis of their assumption that download bandwidth or inflow of nodes is essentially unbounded compared to the outflow [68].

4.2 NP-Completeness of MD-MSOC and Complexity of Exact Solutions

We have already alluded to previous work that prove the NP-Completeness of minimum cost degree-constraint spanning trees [9] [31] [94]. The MD-MSOC problem is also NP-complete by containing the NP-complete problem of finding minimum cost degree-constraint spanning trees. That is, we do not expect to find exact solutions to the MD-MSOC problem that are solvable in polynomial time. Consequently, heuristic approximation algorithms must be devised.

The first question that now arises out of the problem defined in the previous section is: given that we intend to apply the routing mechanism to small-group conferencing application, is it possible to use a 'brute force' or exhaustive algorithm to find the exact solution to the minimum-cost degree-constrained trees for each source? In other words, instead of a heuristic approximation, can we employ a 'brute force' approach to get to the exact solution? As noted in [9], finding an exact solution for the minimum-cost degree-constrained tree is prohibitively expensive computationally. The question remains however as to the possibility of using an exact solution for small number of nodes. For this purpose, we examine the algorithmic complexity of a 'brute force' approach which examines all possible combinations of trees in order to find the exact solution to the minimum-cost degree-constrained routing problem.

An exhaustive approach to MD-MSOC problem has algorithmic complexity of $O((N!)^{2N})$.

Consider a graph with N=|V| vertices. An exhaustive approach to building a single rooted tree on this graph starts with the root of the tree as the only node in the tree and at each

iteration considers every combination of adding a new node to the tree. At the first iteration, there is only one node that can serve as a parent (i.e. the tree root) and N-1 nodes that can be chosen as the child to be added to the tree (Figure 16a). At the second iteration, there are 2 nodes in the tree (i.e. two possible parents) and (N-2) node that can be chosen as the next child to be added to the tree (Figure 16b). At the Nth iteration, there are (N-1) nodes in the tree and only 1 node to be added as the child (Figure 16c). There are therefore (N-1)!×(N-1)! possible trees to be considered.

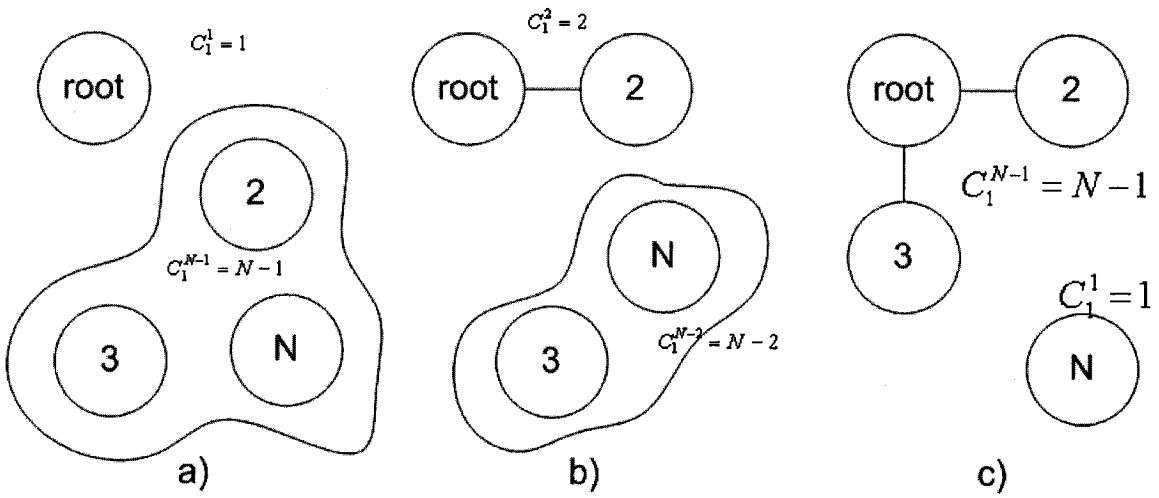


Figure 16. An exhaustive approach to MD-MSOC problem: a) first iteration with 1×(N-1) possibilities b) second iteration with 2×(N-2) possibilities and c) N^{th} iteration with (N-1)×1 possibilities

The exhaustive approach considers each of these trees to see whether it satisfies the degree constraints of each node and finds the minimum cost tree between all the trees that satisfy the constraint. The algorithmic complexity of such an exhaustive approach is thus $O((N!)^2)$. In the MD-MSOC problem however, a tree must be constructed for each source and with N nodes in the graph, in the worst-case scenario there are N trees to be constructed. Therefore an exhaustive approach would have to consider every possible combination of these trees, thus having an algorithmic complexity of $O((N!)^{2N})$. This is

indeed a prohibitively expensive undertaking even for small number of nodes. Consider for instance that when N=6 then the worst-case complexity rises to 10^{34} which for a machine executing 10^9 instructions per second (i.e. a GHz computer) could take a very long time to solve.

The exhaustive approach presented is simplistic (it produces duplicate copies of the same tree) but serves to illustrate the general complexity of the problem. A more efficient approach to finding exact solutions to the MD-MSOC comes about by noting Cayley's formula: given N nodes, there are $N^{(N-2)}$ unique spanning trees or labeled spanning trees [14]. Each of these trees can be uniquely represented by a Prüfer sequence [35]. There are therefore $N^{(N-2)}$ Prüfer sequences that can be generated for a tree connecting N nodes. Given that there exist algorithms for generating all possible trees that do so in constant time for each tree [65], we can therefore have algorithms that list all possible trees with N nodes in $O(K \times N^{(N-2)})$ where K is a constant. Once again, since there are a maximum of N trees to be constructed and therefore a similar approach to the MD-MSOC problem would yield a running time of $O(N^{N \times (N-2)})$, which is still prohibitively expensive even for small N.

A further improvement to the running time of an algorithm that finds a minimum-cost degree-constraint tree in a graph can be achieved by employing the Branch and Bound approach such as the one presented in [61]. This approach however works only in the case of a single tree on a graph. If a single minimum-cost degree-constraint tree is found by a Branch and Bound tree, there is no guarantee that a feasible solution can be found for the other trees on the same graph.

There is another shortcoming with the existing exact solutions: they all assume the

receiver set of the tree is given at the outset but as will be demonstrated in section 4.5, there are cases when it is impossible to construct a tree for a source spanning its receivers unless a node (called a 'reflector') not originally part of the receiver set is added to the receiver set of the tree *during the execution of the algorithm.* Consequently, the exact solutions to the minimum-cost degree-constraint tree problem presented in the literature do not provide an exact solution to the MD-MSOC problem presented here.

Given the prohibitively expensive nature of exact solutions to the MD-MSOC problem as well as the fact that they do not cover all the possible cases, we therefore investigate approximation algorithms that may not find the best solution but will find an approximation in polynomial-time.

4.3 Dynamic vs. Static Approach to Multicast Routing

Before going further, it is necessary to outline the two approaches to multicast routing. The *dynamic* approach considers the creation of multicast trees as nodes dynamically join and leave various sessions [1]. The goal of the dynamic approach is to attach a joining node to the already existing multicast tree, preferably causing few topology changes to that tree. The *static* approach on the other hand considers the global formulation of the problem by constructing a multicast tree from scratch every time. The static approach is applicable to scenarios where there are no late join or early leave multicast nodes and is able to find more and better solutions by constructing a tree from scratch. The dynamic approach is applicable to scenarios where nodes join and leave the session but suffers in terms of its ability to find solutions as well as the quality of the trees it produces due to the restriction on the number of changes in the multicast tree that are allowed.

In large-scale multicast scenarios, rearrangement of all trees for every request causes

instability and is therefore not feasible. For small groups however, tree rearrangement constitutes a smaller number of changes and may therefore incur a tolerable penalty.

In this thesis, we present both a dynamic and a static heuristic algorithm for the MD-MSOC problem given above. We intend to use the dynamic approach in order to quickly construct trees without causing too many topological changes and intend to use the static approach in order to improve the performance of the solution over a period.

4.4 Approximation Algorithms for the MD-MSOC problem

Section 2.4.7 presented other degree-constrained routing problems in graph theory and the heuristics that provide an approximation. Section 2.4 also presented common Application Layer Multicast routing heuristics. Of these heuristics, the ones that do not put exact limitations on individual out-degree of each node are not applicable to the MD-MSOC problem since each node has an exact maximum of outgoing flow (i.e. streams) it can support that must be respected. Among the heuristics that respect the exact constraints of each node, the most relevant and recently developed are those by Shi and Turner [94] [95] and Liu et al [68].

In [94] and [95] the authors provide several centralized heuristic routing algorithms for finding a minimum diameter degree constrained spanning tree in a complete graph. The Compact Tree (CT) algorithm [94] is a greedy algorithm based on Prim's Minimum Spanning Tree [24] that attaches nodes to the tree in the order that produces the smallest increase in diameter. As a result, CT produces low diameter trees with high out-degree nodes close to the root. Balanced Degree Allocation (BDA) [95] on the other hand distributes the out-degree of nodes throughout the tree but increases the diameter as a result. In the performance evaluation presented in [95], BDA has the best (lowest)

rejection rate while CT has the best (lowest) diameter. CT is similar to shortest path tree algorithms presented in section 2.4.3 and employed by ALM protocols such as Yoid [37], TBCP [73] and OMNI [6] while BDA distributing the out-degree throughout the tree resembles routing approaches of CAN-multicast [87] and ZIGZAG [101] but unlike CAN-multicast and ZIGZAG, BDA respects the heterogeneous out-degree of all nodes (instead of only providing a bound). BDA and CT can serve as centralized versions of the myriad of routing algorithms employed by ALM protocols.

We demonstrate the general approach of CT and BDA with a simple example. Figure 17a shows a complete graph with 4 nodes and the out-degree constraints displayed beside each node and the cost of each edge also displayed beside the edge. Assume that a tree rooted at node A must be constructed that spans nodes B, D and C.

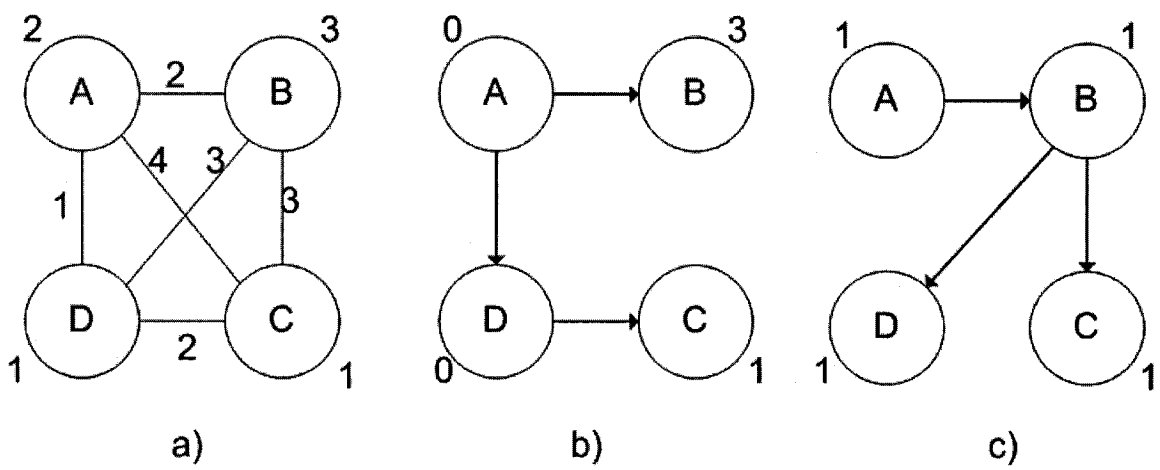


Figure 17. a) Given graph b) tree constructed by CT c) tree constructed by BDA

CT iteratively adds the node with the smallest distance to the root if spare degree is available. So for the tree rooted at node A, the first node to be added is D with A as its parent (since link A-D has the smallest distance to A). Next, node B has the smallest distance to A and so it is added to the tree with A as its parent. Finally, node C is added to the tree with D as its parent since path C-D-A has smaller distance than C-B-A. CT

constructs a tree with minimum diameter tree (here we take diameter to be the largest distance from the source to its receiver) and exhausts the out-degree of the top nodes (Figure 17b).

BDA does not consider cost of links and instead tries to maximize the available degree of nodes on the tree. For the tree rooted at node A, node B is added first since it has the largest available degree. Subsequently, nodes C and D are added as children of node B in order to maximize the available degree of each node (Figure 17c). Note how the tree constructed by CT has the shortest distance between the root (A) and its receivers but has exhausted the available degree of 2 nodes (A and D), while the tree constructed by BDA has a larger distance between the root and its receivers but has allowed each node to keep a spare degree.

We can observe from this simple example the reason behind CT's higher rejection rate as compared to BDA. Consider for instance if on the same graph, node D is to transmit to node C. Given the trees constructed by CT and BDA, we can observe that BDA would be able to accommodate a tree rooted at node D while CT would have to reject such a request since D has exhausted its maximum degree.

Parallel to the work present in this thesis that we first published in [47], Lie et al also introduced several centralized heuristics for building out-degree constrained overlay trees [68]. CLUSTER strives to minimize the number of sessions a node forwards data for in order to accommodate more multicast sessions while DISPERSE distributes the forwarding responsibility among participants similar to BDA. Both favor attaching nodes with highest available out-degree first to the tree. Figure 18 illustrates how CLUSTER builds trees with an example (remaining out-degree is written beside each node). The tree

rooted at node A is constructed by adding nodes with highest out-degree first (Figure 18a): node B is first attached to the tree (with A as its parent) since node B has the largest available degree. Next A is chosen as the parent since it is already forwarding and between C, D and E there is no preference as the next child since they all have the same available degree (for the example we chose E). Nodes C and D are added to B since B has the largest available degree. Subsequently, when building another tree rooted at node E, a node already forwarding (i.e. node B) is used as parent before others (Figure 18b).

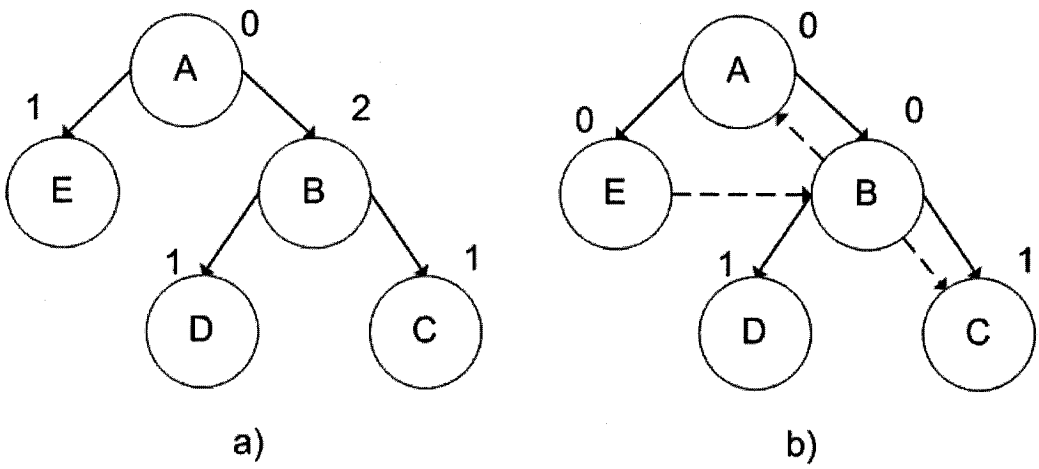


Figure 18. CLUSTER

One shortcoming of CLUSTER is that a node that may not have any spare out-degree for its own tree may be a parent in a tree rooted at another node. A variation called CLUSTER-swap shifts an edge from one tree to another in order to accommodate such a scenario. In the performance evaluation presented in [68], CLUSTER-swap and DISPERSE-swap result in similar rejection rates lower than other variations. It should be noted that the heuristics present in [68] do not consider edge cost (i.e. delay) in their algorithm and instead concentrate on building low-depth trees where depth is measured as the number of hops or links between a source and a target. A a low-depth tree is not necessarily a low-cost tree since it is possible for multiple hops to have a smaller

cumulative cost compared to a single but very costly hop.

All of the above algorithms (BDA, CT, CLUSTER-swap) are centralized and consider the *static* approach to multicast routing. As mentioned in the previous section, a dynamic approach may be more desirable when there are late join and early leave events during the multicast session. Liu et al introduce dynamic versions of their heuristics (henceforth called CLUSTER-Dynamic and DISPERSE-Dynamic). CLUSTER-Dynamic attaches a late joiner to a node with minimum available bandwidth that can support the new node. Conversely DISPERSE-Dynamic attaches a late joiner to a node with maximum available bandwidth. These two algorithms are considered when evaluating the performance of our heuristic under dynamic conditions and are compared to the dynamic heuristic introduced in this thesis. In the next section, we will demonstrate the shortcomings of the heuristics discussed in this section through examples and go on to devise our own heuristics. In Chapter 5 we compare the performance of our heuristic (MOTOR dynamic and static) with BDA, CT, CLUSTER-swap, CLUSTER-Dynamic and DISPERSE-Dynamic and demonstrate that MOTOR-dynamic is able to find a greater percentage of solutions than other dynamic approaches while MOTOR-static finds a greater percentage of solutions than all the mentioned heuristics. Furthermore, unlike the aforementioned heuristics, MOTOR considers priority of each stream as well as heterogeneous flow weights while incurring a relative delay penalty comparable to the heuristic with lowest relative delay penalty (i.e. CT).

4.5 Shortcomings and Opportunity for Improvements

The heuristics presented in this thesis take advantage of certain shortcomings in existing algorithms in order to achieve better performance. We will first demonstrate these

shortcomings through examples and then go on to describe how our heuristic (MOTOR) tries to address them.

Except CLUSTER-swap, whose introduction was parallel to our own work, all other heuristics construct overlay trees independent of one another. This is derived from the assumption that different multicast sessions belong to different applications and are independent of one another, which is the case when several independent single-source applications exist on the same network. In a multi-party videoconferencing application however, there are multiple multicast sessions that make up the same conferencing session. The routing mechanism can exploit this feature by building trees in the order that will accommodate more solutions. We illustrate with an example.

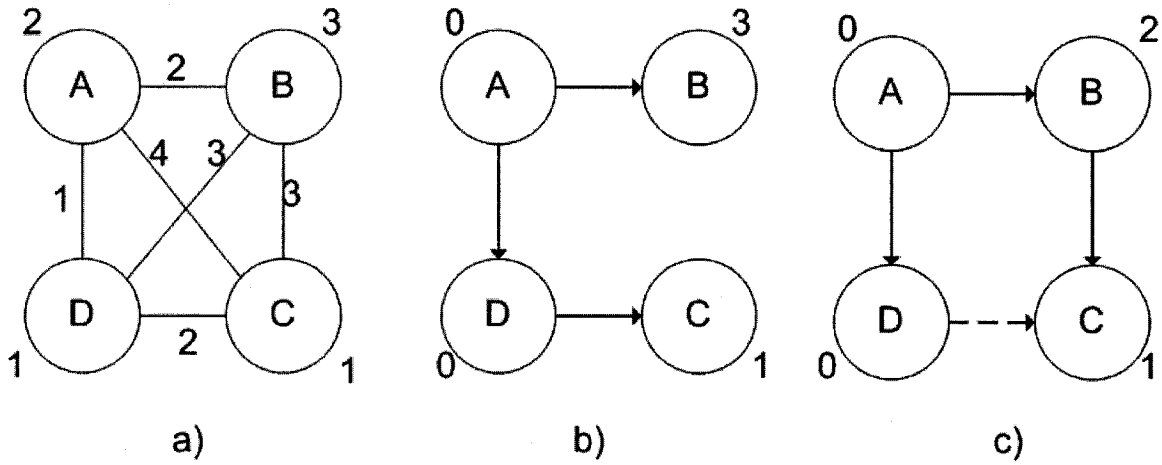


Figure 19. Considering order of overlay tree construction

Given 4 nodes A, B, C, D with out-degrees 2, 3, 1, 1 available to them respectively as shown in Figure 19a, recall that CT constructs a tree rooted at node A and spanning B, C, D as shown in Figure 19b. Now CT will reject the construction of an overlay tree rooted at node D since node D does not have any out-degree available to it. Now consider if the tree rooted at node D is constructed before the tree rooted at node A. Doing so will prevent the tree rooted at node A from using node D's only available out-degree and

therefore will accommodate both trees as shown in Figure 19c.

As mentioned before, CLUSTER can avoid having a node unable to transmit its own stream due to its forwarding of another stream by shifting forwarding responsibility away from the tree root to another node on the same tree (called the CLUSTER-swap variation in [68]). CLUSTER-swap however does not consider the more general approach to shifting branches in another tree in order to accommodate more requests. Consider the same node out-degree constraints of Figure 19a and the trees rooted at nodes A, C and D as shown in Figure 20a. At this point a late joiner requesting to be attached to tree rooted at node D must be rejected since no node on the tree has any available out-degree to become the parent of the new node. Relocating an edge in the tree rooted at node A however can result in a spare out-degree in tree rooted at node D and thus the attachment of the new node as shown in Figure 20b.

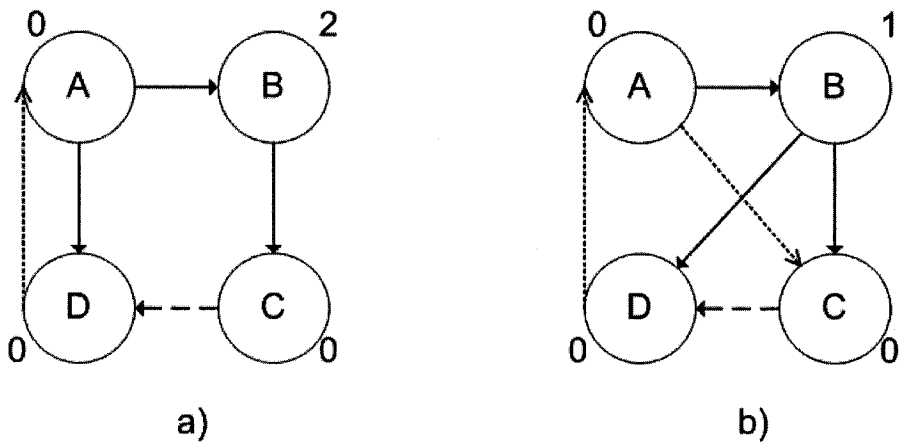


Figure 20. Relocating out-degree in order to accommodate more nodes

Another way overlay routing heuristics can be improved is by considering a node not on the tree but with available out-degree as a 'reflector' in a tree that cannot otherwise accommodate all its receivers. Consider the same node out-degree constraints as those in Figure 19a and the trees rooted at nodes A and D as shown in Figure 21a.

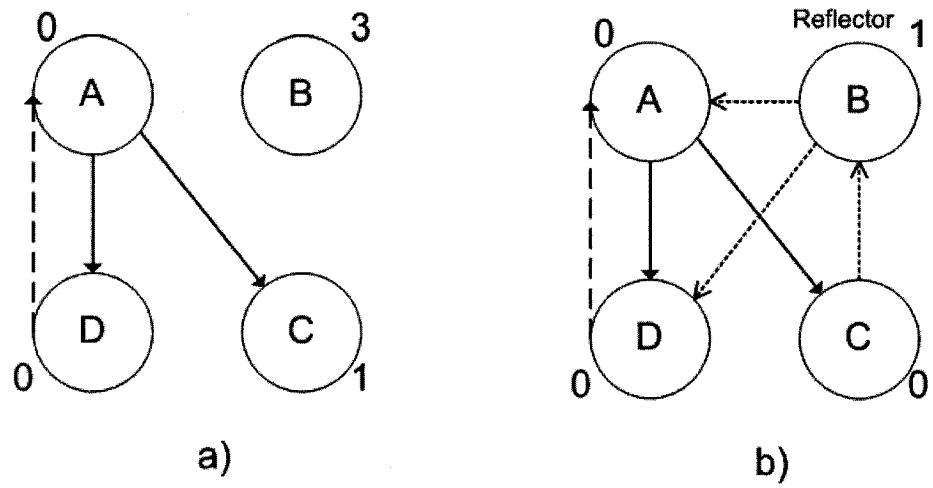


Figure 21. Using a 'reflector' in overlay trees

A tree rooted at node C and containing nodes A and B cannot be built by the heuristics presented so far since the node with available out-degree (node B) is not in the receiver set of node C. If node B is used as a 'reflector' (i.e. a node that despite not requesting to be in a tree is included in the tree in order to make use of its available outflow), then the tree rooted at node C can be constructed as shown in Figure 21b.

An important shortcoming of all the heuristics presented in [68] is that instead of minimizing the diameter or cost of the tree, CLUSTER, DISPERSE and their variations all consider tree depth. Tree depth however is not an accurate measure of tree depth or in networking terms, the number of hops (i.e. tree depth) does not necessarily reflect the delay between two nodes in the network. That is why these algorithms perform poorly in terms of the delay penalty they incur (see sections 5.3.1 and 5.3.2) compared to heuristics that consider edge costs as a metric instead of tree depth.

Finally, we should reiterate the point made earlier that none of the heuristics discussed so far consider different priority of nodes in the tree when rejecting a request that cannot be accommodated. Furthermore, all the said heuristics consider out-degree as a measure of outgoing bandwidth but as mentioned in section 4.1.1, when there are heterogeneous bit-

rates out-degree is not adequate for representing outgoing bandwidth of each node. It is for this reason that none of the mentioned heuristics can be used to solve the MD-MSOC problem in its most generic form (when each receiver may request a different bit-rate). MOTOR however considers the outflow of each node and constructs trees in such a way as to accommodate heterogeneous flow weights (i.e. bit-rate) for each receiver.

Having outlined some of the major limitations of existing minimum-cost degree-constraint overlay tree-building heuristics, we now present our novel heuristics that attempt to address these limitations and shortcomings.

4.6 *MOTOR: Multi-source Outflow-constrained Tree Overlay Routing*

In the previous section, we highlighted how existing heuristics do not find certain groups of solutions when building multiple degree-constrained trees on the same graph and cannot solve the MD-MSOC problem in its most generic formulation. This section presents two heuristics. MOTOR (Multi-source Outflow-constrained Tree Overlay Routing) improves on the existing approaches by finding more solutions in addition to taking into account priority and heterogeneous flow weights on the tree while constructing low cost trees. A dynamic version of this heuristic, called MOTOR-Dynamic, builds on the ideas of MOTOR in order to better accommodate dynamic joining and leaving of nodes in a multicast session by restricting the number of changes that each request can cause in the overlay topology.

Here we will only give the main principles of the two heuristics through examples and general outlines, while the appendix provides the detailed account of each heuristic. We first present the overall static approach to the problem addressed by MOTOR and then go on to illustrate how MOTOR-Dynamic incorporates the main ideas of MOTOR under

dynamic conditions.

4.6.1 The Static Approach: MOTOR

MOTOR takes a two-step approach to the minimum-cost outflow-constrained problem. The first step involves assigning 'per tree outflows' to each node such that the total outflow assigned to each node does not exceed its outflow constraint. The 'per tree outflow' assigned to each node represents how much forwarding responsibility is allotted to that node for a particular tree. Other than respecting the maximum outflow constraint of each node, the per tree outflow assignment should be done in such a way so as to accommodate the total outflow required by each tree. For example, if a tree is to span a set of M receivers that all wish to receive a flow weight of 1, then the total outflow assignment for that tree should be at least M. Once a feasible solution to the first problem has been found, the second step involves finding a minimum cost tree for each source while respecting the per tree outflow assigned to each node in the first step. As we shall see in Chapter 5, this approach helps to construct low cost trees but avoids the large rejection rate of heuristics that greedily create low cost trees from the beginning.

If a feasible solution is not found for the first step, then a request or a set of requests with lower priorities are rejected in order to accommodate higher priority requests. The general flow and logic of the algorithm is demonstrated in Figure 22.

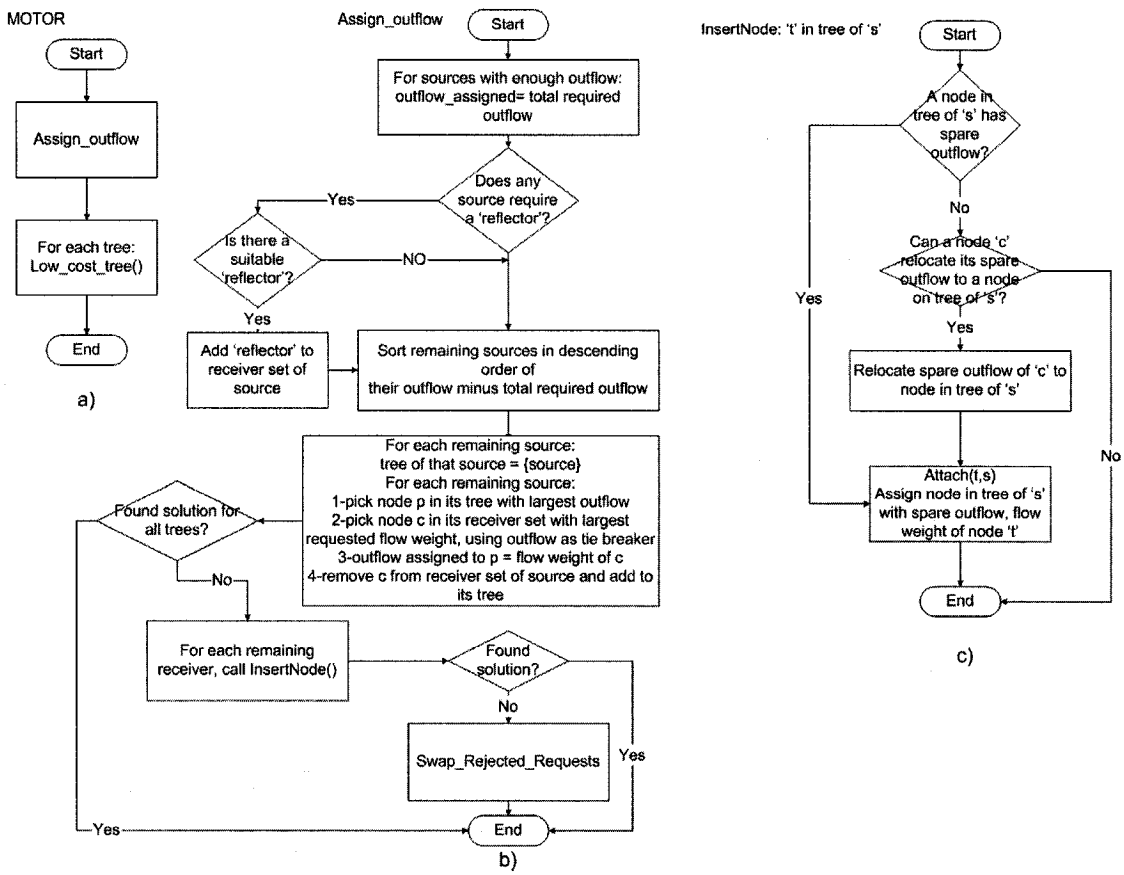


Figure 22. General flow of MOTOR

The *Assign_Outflow* routine whose general flow is depicted in Figure 22b and is outlined in more detail in Figure 23 represents the first stage of the process. *Assign_Outflow* first distinguishes between sources that have enough spare outflows to support all their receivers and those that require other nodes to forward for them. For those nodes with maximum outflows greater than the total required by their receiver set, the overlay tree will start with a star topology (every receiver is directly connected to the source node). For other nodes, overlay trees are constructed in descending order of the difference between maximum outflow and total outflow required (i.e. trees are constructed first for sources requiring less forwarding from other nodes).

In this way the algorithm tries to prevent a node from forwarding for other nodes when that node itself may not have enough spare outflow for its own tree.

Assign_outflow() $O(N^7)$

Input: Given a complete graph G=(V,E), a maximum outflow outflow_max(v) for all nodes $v \in V$, a target set Rs for each source and a request flow weight fw(s,t) and priority p(s,t) for each source-target pair (s,t). fw(s,t) for s=t is infinite.

Output: a set of trees Tk rooted at each source $k \in V$ and spanning its targets Rk such that outflow_max(v) is respected for all nodes $v \in V$ and if i>j then $fw(i,s) \leq fw(j,s)$ for $i,j \in Ts$ for all $s \in V$. Returns TRUE if successful.

$A=\{s \mid s \in V, outflow\_max(s) > \sum fw(s,i) \ \forall \ i \in Rs\}$
$D=\{s \mid s \in V, s \notin A\}$

$\forall \ s \in A, v \in Rs$:
    add edge={s,v} to Ts with flow weight fw(s,v)
$outflow(s)=0 \ \forall \ s \in D$
for all $s \in D$ in descending order of $(outflow\_max(s) - \sum fw(s,i) \ i \in Rs)$:
    outflow(s)=outflow_max(s)
    avail_outflow=outflow_max(s)+∑outflow(k), $k \in Rs$
    if(avail_outflow<$\sum fw(s,m) \ \forall \ m \in Rs$):
      if $\exists \ r \in V \mid r \notin Rs, r \neq s, outflow(r) \geq (fw(s,n)+fw(s,m))$ $m,n \in Rs$:
        avail_outflow+=outflow(r)
        Rs+={r}
        fw(s,r)=Max(fw(s,n),fw(s,m))
    Total_outflow=outflow_max(s)
    Ts={s}
    min_flow = Min(fw(s,i) $i \in Rs$)
    while(Total_outflow>= min_flow & |Rs|>|Ts|) do:
        find u | $u \in Ts$ with Max(outflow(j)) $\forall j \in Ts$)
        $P=\{p \mid p \in Rs, fw(s,p) \leq outflow(u), fw(s,p) \leq fw(s,u)\}$
        Choose $q \in P$ with Max(fw(s,q)) and use greater outflow(q) for tie-break
        add edge={u,q} to Ts with flow weight fw(s,q)
        Rs - = {q}
        Total_outflow+=[outflow(q)- fw(s,q)]
        Min_flow = Min(fw(s,i) $\forall \ i \in Rs$)

$\forall \ s \in D$, |Rs|>|Ts|:
    Insertnode(s,t) $\forall \ t \in Rs, t \notin Ts$
$\forall \ s \in D$, |Rs|>|Ts|:
    Swap_rejected_requests(s,Rs)
else Return TRUE

Figure 23. First step of MOTOR: assigning per tree outflows

As demonstrated in section 4.5, it is possible that the total outflow required to construct a tree exceeds the sum of the available outflow of all nodes intended to be included in that

tree and therefore it may be necessary for a node that is not a receiver to act as a 'reflector'.

The concept of a 'reflector' is commonly used by multi-party videoconferencing applications and involves a dedicated entity forwarding the stream of one source to multiple destinations even though that entity itself is not a conferencing participant but typically a server or Multi-point Control Unit [10]. In our approach, we generalize the concept of a reflector to any node that, though not part of the receiver set of a source, can temporarily assist in forwarding its flows. Therefore *Assign_Outflow* first investigates whether a 'reflector' is required to construct the multicast tree and if so, searches to find such a node and includes it in the receiver set of the source if it is found. A node is eligible to be considered as a 'reflector' for a source if it has enough spare outflow to be the parent of at least two nodes on the tree of that source.

Attention should be drawn to the fact that using a node as a 'reflector' implies the use of that node's bandwidth and processing power for the benefit of other nodes, leading to the issue of trust and cooperation between peers in a peer-to-peer system which is the subject of a brief discussion in section 7.4.

After determining whether a 'reflector' is needed for a source or not, nodes in the receiver set of a source are attached to the tree in descending order of their flow weight and, if they have the same flow weight, in descending order of their available outflows, with the node already in the tree with the largest available outflow parenting the new node. By choosing the node with largest available outflow in the tree as the parent, the algorithm tries to increase the chance of accommodating future branches in the tree. Likewise (when requested flow weights are the same), the next node to be added to the

tree is the one with the largest available outflow in order to increase the total available outflow on the tree.

As an example of how the *Assign_Outflow* routine works, Figure 24 illustrates the steps taken to construct a tree rooted at source A and spanning nodes (B,C,D,E) with requested flow weights of (1.0, 0.5, 0.25, 0.25) respectively. First, if possible, nodes with highest requested flow weight are attached and if more than one request have the same flow weight, the receiver with the highest outflow available is attached first.

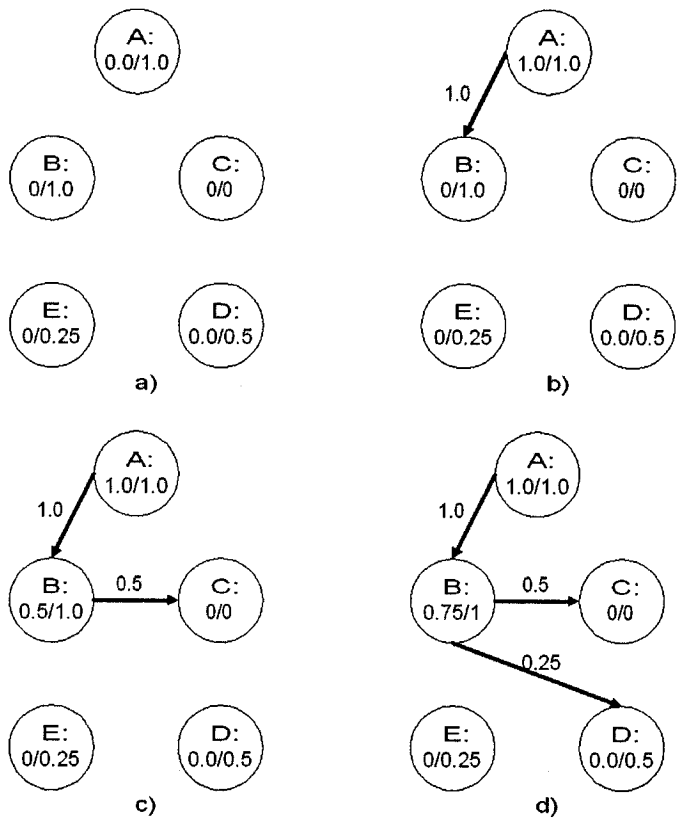


Figure 24. Assigning outflows

At the final step, the *Assign_Outflow* routine searches for any node that has not yet been attached and attempts to attach it through the *InsertNode* routine. We will return to the issue of complexity of *Assign_Outflow* at the end of this section.

Figure 22c shows the general outline of *InsertNode* and Figure 26 the details. *InsertNode*

attempts to attach a node to a tree that does not have spare outflow to accommodate it. In order to do so, it searches for possibilities of first relocating the spare outflow of some node on *another* tree to the targeted tree in order to allow the attachment of the new joining node. The *Relocate_spare_outflow* routine (Figure 25) investigates whether such a possibility exists or not and then the *Attach* routine (Figure 29) attaches a new node to the requested tree once enough outflow has been relocated to that tree.

```
Relocate_spare_outflow(c,s,t) O(N^4)

Input: given a source s∈V, a node c∈V with outflow_max(c)- outflow(c)>=fw(s,t)

Output: relocate spare outflow of c to a node d∈Rs, such that outflow_max(v) is respected for all
nodes v∈V. Returns a node d with outflow(d)≥fw(s,t) if successful, NULL otherwise.
d=NULL
∀ d | d∈Rs OR d=s, d≠c:
        if can_relocate(c,d,s,t)=TRUE: Return d
if ∃ a,d | a∉Rs, a≠c, outflow_max(a)-outflow(a)<fw(s,t), can_relocate(c,a,s,t)=TRUE,
          d∈Rs OR d=s, d≠c,d≠a, can_relocate(a,d,s,t)=TRUE:
                Return d

can_relocate(c,d,s,t) O(N^2)

Input: given a node c∈V with outflow(c)≥fw(s,t) and a node d∈V, c≠d

Output: see if it is possible to transfer the spare outflow of node c to node d such that
outflow(d)≥fw(s,t). Returns TRUE if successful.

if ∃ e | e∈V, d∈Te, c∈Te, |Te|>2:
        G={g | g>d,  g,d∈Te}
        if ∃ H⊆G | H≠∅, outflow(d)+removed_flow)≥ fw(s,t), outflow(c)≥removed_flow,
                        fw(e,c) ≥fw(e,h) ∀ h∈H OR c ⊢ d
                        (where removed_flow = ∑fw(e,h), h∈H):
                remove edge={d,h} from Te, h∈H
                if c ⊢ h in Te & outflow(c)+removed_flow>=fw(e,d):
                        remove edge={f,d} from Te d>f
                        add edge={f,h} to Te with flow weight fw(e,d)
                        add edge={c,d} to Te with flow weight fw(e,d)
                        Return TRUE
                else if c ⊬ h:
                        add edge={c,h} to Te, h∈H with flow weight fw(e,h)
                        Return TRUE
```

Figure 25. Trying to relocate spare outflows from one tree to another

The *Relocate_spare_outflow* routine is called for every node that is NOT part of the receiver set of the source and that has spare outflow (Figure 26). As a result *Relocate_spare_outflow* first investigates whether the node with spare outflow can take on forwarding responsibility for another node on the intended tree (through the *can_relocate* routine). If that is not successful, *Relocate_spare_outflow* attempts to relocate the spare outflow to an intermediary node, which can then relocate to a node on the intended tree (that is, relocating spare outflow from node C to node A and then from node A to a node on the intended tree). Of course the *Relocate_spare_outflow* routine can be made recursive in order to search more possibilities for relocation (i.e. N levels of relocation). Instead we opted for maximum of only 2 levels of relocation in order to keep the complexity lower.

InsertNode(s,t) $O(N^5)$

Input: given source $s \in V$ and target $t \in V$ ($t \notin Ts$) and flow weight fw(s,t)

Output: insert new target t to the tree Ts of source s with flow weight B, such that outflow_max(v) is respected for all nodes $v \in V$.

$\forall c \mid c \notin Rs$, outflow_max(c)- outflow(c)$\geq$ fw(s,t):
    d=Relocate_spare_outflow(c,s,t)
    if d$\neq$NULL:
    attach(s,t,d)

Figure 26. InsertNode routine

Figure 27 illustrates a simple example: given the two trees one rooted at node A and the other at node C (Figure 27a), relocating the spare outflow of node B to node A is necessary in order to support the attachment of node D to the tree rooted at node A (Figure 27b).

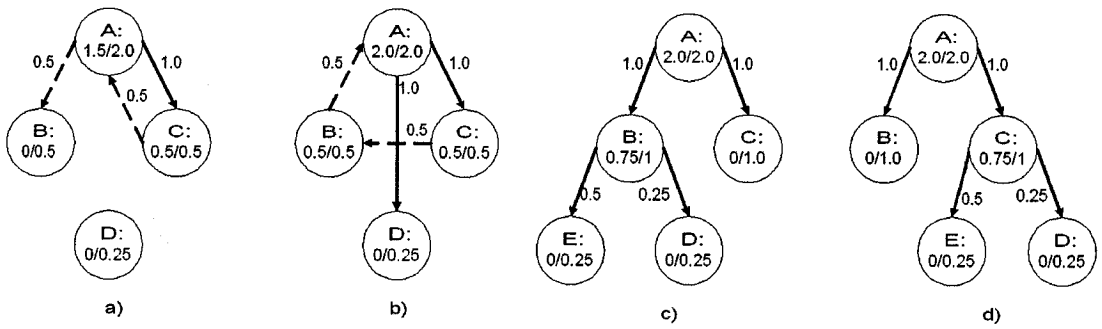


Figure 27. Relocating spare outflow from one node to another

The relocation of spare outflow involves rerouting flows in such a way that a node with spare outflow takes on the forwarding responsibility of another node. Figure 27c,d show another example of how spare outflow is transferred from one node to another: spare outflow of node C is relocated to node B through node C forwarding to nodes D and E.

Despite limiting it to a 2-level search, the complexity of *Relocate_spare_outflow* is still relatively high. Given a node 'c', the *can_relocate* routine at the worst case examines all other nodes (N-1 nodes) that 'c' can exchange spare outflows with. When examining a node, the routine at the worst case needs to look at every children of that node (possibly N-2 nodes), thereby bringing the complexity of *can_relocate* to $O(N^2)$. Note that if there are homogeneous flow weights (i.e. every request has the same flow or bit-rate), then there is no longer a need to examine every children of a node but it suffices to choose one. This is because as long as two nodes are on the same tree and one node has spare outflow, there is a way for them to exchange outflows (with heterogeneous bit-rates this is not necessarily the case). Therefore, removing heterogeneous bit-rates from the requirements would reduce the complexity of *can_relocate* to $O(N)$. The *Relocate_spare_outflow* routine at the worst case considers every pair of nodes for the exchange of outflows. Since each consideration is $O(N^2)$ from *can_relocate*, the

complexity of *Relocate_spare_outflow* becomes $O(N^4)$. *InsertNode* calls *Relocate_spare_outflow* for every node with spare outflow and has therefore a complexity of $O(N^5)$.

attach(s,t,c) O(N)

Input: given source s∈V, a target t∈V and a node c∈Rs such that outflow (c) ≥fw(s,t)

Output: attach target t to the tree Ts of source s, taking advantage of spare outflow of c

```
if c∈Ts:
        if fw(s,c) ≥fw(s,t)
                add edge={c,t} to Ts with flow weight fw(s,t)
                Return TRUE
        else if fw(s,p) ≥fw(s,t) & outflow(p) ≥(fw(s,t)-fw(s,c)), c≻p in Ts:
                add edge={c,t} to Ts with flow weight fw(s,t)
                fw(s,c)=fw(s,t)
                outflow(p) -=fw(s,t)-fw(s,c)
                Return TRUE
else if c=t:
  ∀ h∈Ts | fw(s,h) ≥fw(s,t):
        G={g | g≻h}
        if ∃ K⊆G | K≠∅, [outflow(h)+∑fw (s,k)] ≥fw(s,t) &∑fw (s,k)≤outflow(c)∀k∈K:
                remove edge={h,k} in Ts ∀k∈K
                add edge ={h,c}to Ts with flow weight fw(s,c)
                add edge={c,k} to Ts with flow weight fw(s,k) ∀k∈K
                Return TRUE
Return FALSE
```

Figure 28. *Attaching* a node to a tree

If *Relocate_spare_outflow* succeeds, the *Attach* routine joins a new node to the requested tree with the parent who received the spare outflow as a result of the outflow relocation. The *Attach* routine does so by either attaching the new node as a child of the node with spare outflow or (if the node with spare outflow happens to be the node requesting to be attached to the tree) by rerouting the flows through the joining node. That is, the joining node, instead of becoming a leaf node (i.e. one with no children in the tree), becomes a parent of other nodes in order to free up spare outflow for nodes that will become its parent (Figure 28).

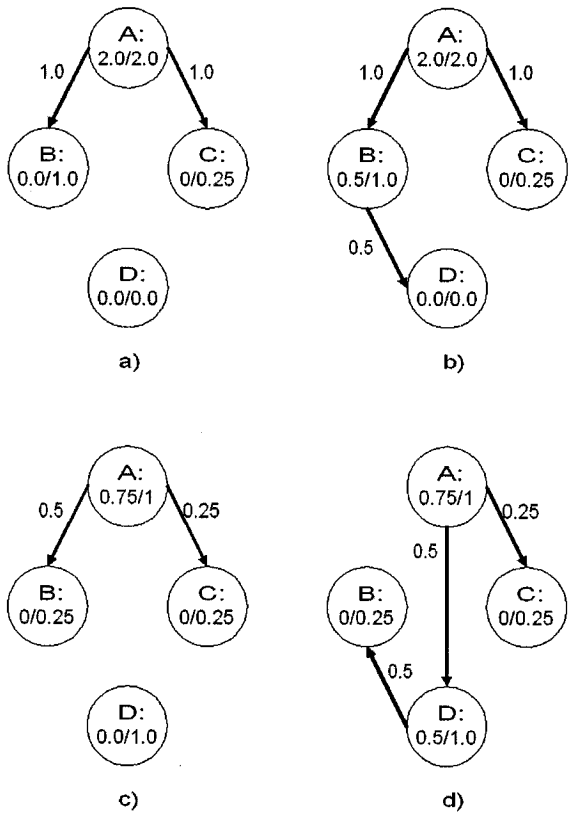


Figure 29. *Attach* routine example

Figure 29 illustrates this with two examples. Consider four nodes (A,B,C,D) having outflow maximums of (2.0, 1.0, 0.25, 0.0) respectively, with an overlay tree rooted at node A and connected to nodes B and C with flow weight of 1.0 (Figure 29a). Node D wishing to join this tree is simply attached to node B which has spare out flow (Figure 29b). In contrast, in Figure 29c, none of the nodes on the tree rooted at node A have spare out flow to support a new node D. The flow is therefore rerouted through D (that has spare outflow) as shown in Figure 29d.

For the *Attach* routine, the first case (if a node on the tree has spare outflow), the complexity is constant K. However, when the attached node happens to be the node with spare outflow, the routine searches the children of every node on the tree in order to find a set of children that it can attach to the joining node. Since a node cannot appear

multiple times in the *same* tree, the routine will execute a maximum of N-1 times or has complexity O(N).

If a solution is still not found after the *InsertNode* routine (i.e. a number of requests are to be rejected) the *swap_rejected_requests* attempts to substitute higher priority requests into the tree by dropping lower priority requests. This routine first finds the rejected request with highest priority and removes all requests with lower priority from the receiving set of the source (while retaining requests with equal or higher priority). It then calls the *Assign_outflow* routine again in order to find a tree for the new receiving set. At this point, a last attempt is made to attach the omitted requests into the tree through the *InsertNode* routine. Note that the *InsertNode* routine only attempts to add a new node to an existing tree and does not drop any node from the existing tree (therefore there is no chance of lower priority requests that were omitted from causing the detachment of higher priority requests from the tree).

Swap_rejected_requests

Input: a source $s \in V$, priority for each request p(s,t) from source s to receiver t and Rs the receiver set of source s

Output: a tree rooted at source s and spanning a subset of Rs with lower priority requests rejected

```
∀ s∈V such that |Rs|>|Ts|:
        find u such that p(s,u)>=p(s,t) for all t,u∈Rs AND t,u∉Ts,
        Q={q∈Rs such that p(s,q)<p(s,u)}
        ∀ q∈Q:
                Rs=Rs-{q}
        Assign_outflow(Rs)
        ∀ t∈Q:
                Insertnode(s,t)
```

Figure 30. Rejecting lower priority requests

Recall that complexity of *InsertNode* is $O(N^5)$ with heterogeneous bit-rates and $O(N^4)$ without. *Assign_outflow* calls *InsertNode* for every request that is not satisfied by the end

of its first stage. At worst case therefore, *Assign_outflow* will call *InsertNode* in the order of N^2 times (there are a maximum of N(N-1) possible requests), thereby having a complexity of $O(N^7)$. If lower priority requests are rejected through the *Swap_rejected_requests* routine, the complexity rises to $O(N^8)$ since *Assign_outflow* is called for every source whose tree does not span all its targets.

4.6.2 Constructing low cost trees

The algorithm presented so far does not consider the cost of each edge when constructing trees and instead tries to satisfy the outflow constraint of each node. The result of the *Assign_outflow* routine is therefore a set of trees that respect the outflow constraint of each node but may not be low cost. The trees constructed by *Assign_outflow* however can be regarded as per-tree outflow assignments for each node that can then be used to construct low cost trees using a minimum-cost tree-building algorithm. We use the Compact Tree (CT) algorithm of [94] with the variation the authors of the paper mention, which defers the addition of a node that reduces the available outflow of the tree to zero when there are more nodes to be added to the tree. Figure 31 illustrates with an example. Given the edge costs and node outflow constraints shown in Figure 31a, consider the trees rooted at nodes A and D constructed by *Assign_outflow* as those shown in Figure 31b. For the tree rooted at node A, node B has two children and therefore has an outflow assignment of 2 while node A has one child and one spare outflow not used by any other tree and therefore has an outflow assignment of 2. Node D has outflow assignment of 0 since its only outflow is used for its own tree. The resulting outflow assignment is shown in Figure 31c. Using the variation mentioned of the CT algorithm of [94], the resulting low cost tree rooted at A is shown in Figure 31d. This leaves for the tree rooted at node D

the outflow assignments shown in Figure 31e, used by the variation of CT to get a low cost tree rooted at D as shown in Figure 31f.

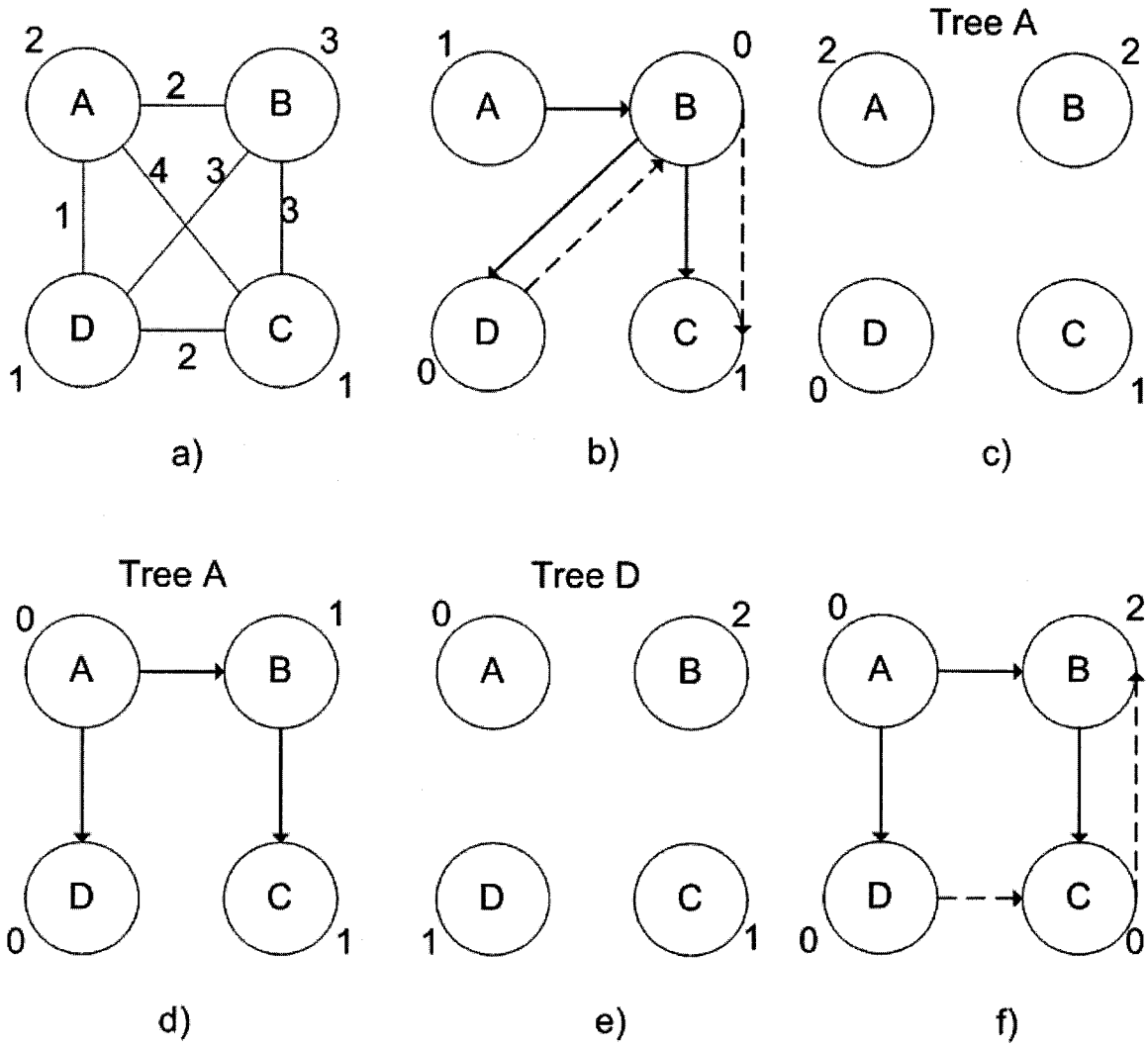


Figure 31. Trees of *Assign_outflow* regarded as per-tree outflow assignments in order to build low cost trees

As will be demonstrated in the next chapter, this two-step approach to making low cost outflow-constraint trees remedies the high rejection rate of greedy low cost out-degree constraint algorithm such as CT and yet constructs similarly low cost trees.

4.6.3 The Dynamic Approach: MOTOR-Dynamic

MOTOR-Dynamic borrows its main concepts from MOTOR and applies them in a

dynamic setting by restricting the number of changes that a request is allowed to cause by joining the multicast session. The application must specify the maximum number of changes that the application tolerates when a new node 'soft joins' a multicast session (recall definition of 'soft join' from section 3.3).

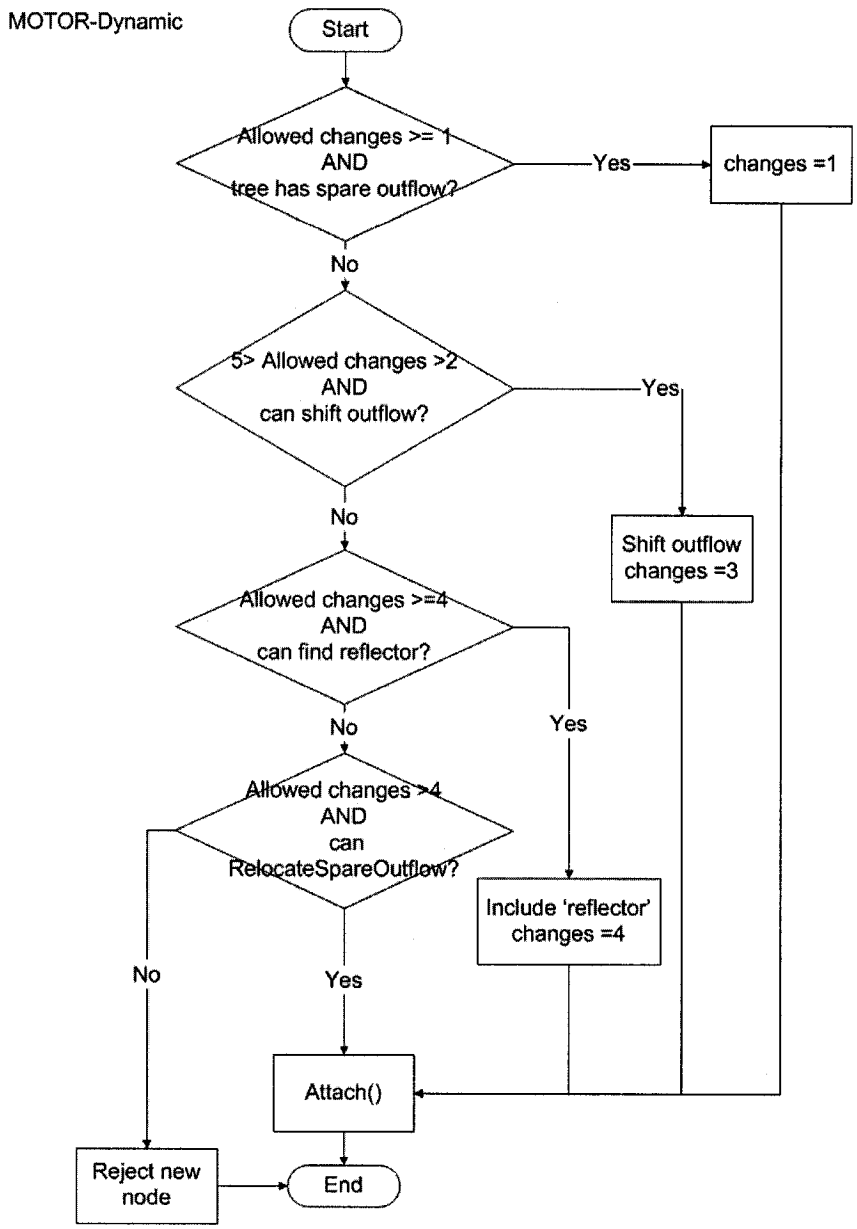


Figure 32. MOTOR-Dynamic overall flow

MOTOR-Dynamic uses this value as a parameter when trying to join a node to an existing overlay multicast tree. It iteratively considers the various ways of connecting a

node to the request tree from the ones causing the least number of changes to the ones causing the maximum allowable number of changes. Figure 32 shows the general flow of MOTOR-Dynamic. Note that MOTOR-Dynamic only works with homogeneous flows (all requests have the same flow).

The first and trivial solution is simply adding the new node at the closest distance to the source that has spare outflow to accommodate the new node, thus causing only 1 change. If the new node itself has enough spare outflow to parent a node on the tree, then 3 changes can result in the new node being included in the tree. There are also 3 changes involved when shifting spare outflow from a node on another tree to the one requiring outflow to accommodate the new node.

The routine *RelocateSpareOutflow* presented in the appendix can be used for shifting operations that cause more than 3 changes. If there is a node that can be used as a 'reflector', there are 4 changes that would accommodate the attachment of the 'reflector' as well as the new node to the tree.

If the maximum allowable changes is set to only 1, then MOTOR-Dynamic degenerates into algorithms used by protocols such as Yoid [37] and SpreadIt [28] that simply add a new node to the closest parent in tree that has enough outflow to accommodate it. In this case the complexity of MOTOR-Dynamic is $O(N)$ since the algorithm simply examines all the nodes on the tree of a given source.

When the maximum number of changes is set to infinity, then MOTOR-Dynamic essentially degenerates into the *Relocate_spare_outflow* routine discussed in the previous section that has complexity of $O(N^3)$ when considering homogeneous flows. MOTOR-Dynamic therefore has worst case complexity of $O(N^3)$. The details of MOTOR-Dynamic

(such as how edge costs are factored in the tree building in order to construct low cost trees) are given in the appendix.

In addition to accommodating a new joining node, a dynamic approach should also deal with nodes leaving a session. Our approach to a node leaving a tree is simple. If the leaving node is a leaf node (i.e. has no children) it is simply removed from the tree. If the leaving node has one child, the child is reattached to the parent of the leaving node. If the leaving node has more than one child, the leaving node is kept as a 'reflector' in order to minimize the disruption of flow to its children. Note that it is within the context of a multi-party videoconferencing application that a leaving node can be kept as a reflector. This is because a node leaving the tree of one node is still part of the conferencing session as a whole and can therefore continue to assist in the forwarding of a particular session.

4.6.4 A Note on Rejecting Requests

It is important to note here that any heuristic would fail to satisfy every request if the total weight of all requests exceeds the total outflow available. In a single-source scenario this may not be a great limitation since every receiver can be assumed to add forwarding capability to the distribution tree it joins. In a multi-source scenario however, each participant is the source of its own data and may not be able to contribute to the forwarding capability of the distribution tree it joins. As a result it is important in such cases for the application to impose limits on the total weight or number of requests made during the session in proportion to the total outflow available from all participants. As will be shown in section 6.2.5, this is possible through careful design of the application interface in conjunction with the employment of an awareness management mechanism.

4.7 Summary

This chapter began with the statement of the routing problem applicable to the class of application presented in section 2.1.3 whose requirements were outlined in section 2.2. We then presented existing approaches that are applicable to this routing problem and highlighted their shortcomings and how they can be improved. The rest of the chapter gave an overview of how our heuristics called MOTOR and MOTOR-Dynamic attempt to address these shortcomings as well as add new features such as priority and heterogeneous bit-rate. The next chapter is an evaluation of the performance of MOTOR and MOTOR-Dynamic as compared to existing solutions.

Chapter 5 Performance Evaluation

In this section, we aim to investigate how the heuristics discussed in the previous section compare to other overlay multicast tree building heuristics under various conditions. We evaluate existing relevant heuristics (presented in section 4.4) CT, BDA, CLUSTER-swap, CLUSTER-Dynamic and DISPERSE-Dynamic and compare their performance to our own heuristics MOTOR and MOTOR-Dynamic. Note that all of these heuristics are centralized. Note also that for MOTOR-Dynamic the maximum number of changes to topology allowed is set to 4 for all the tests in this chapter.

5.1 Metrics

There are three primary metrics for gauging the performance of an overlay routing heuristic: rejection rate, relative delay penalty and link stress. When there exists a constraint on the degree of each node (be it in-degree or out-degree or both), it may become impossible to find a tree spanning all its nodes without violating the degree constraint of each node if there is not enough available degree. In such a case a node or a set of nodes on the tree are dropped from the tree. The frequency of dropping a node is termed the rejection rate in [94] and blocking probability in [68] and is a measure of how often the routing algorithm is able to find a solution to the routing problem. A low rejection rate is a desirable characteristic of a routing algorithm.

In addition to finding *how often* a solution is found, it is also of importance to have a measure of *how good* the found solution is. In the context of ALM protocols, *stretch* and *stress* of an overlay tree are the primary metrics for measuring the quality of the overlay tree [22]. Tree stretch is a statistical measure of the length of the branches of the overlay

tree, or in the terminology of networks, the delay from the source to its receivers. Tree stress on the other hand is the statistical measure of the number of duplicate packets on the same physical link. Low stretch trees are desired in order to reduce the delay a receiver experiences in receiving a stream from a source, while low stress tress are desired in order to reduce the load on the underlying network. A heuristic can aim to minimize the worst-case stretch of a tree or alternatively minimize the total/average stretch of a tree.

In the context of the work presented in this thesis, the routing heuristic constrains the outflow of each node to a given maximum. Assuming that limitation of bandwidth primarily exists on the 'last-mile link' (between the end user and the router connecting that user to the Internet) and not in the Internet backbone, we observe that the routing heuristic presented in our work limits the worst-case link stress to that specified by the outflow constraint of each node. In other words, the worst-case link stress is a function of the maximum outflow assigned to the end-system nodes which is respected by the routing heuristic when building overlay trees. Maximum stress is therefore regarded as an input parameter to degree-constraint routing algorithms. We therefore focus on measuring the rejection rate and the stretch of overlay trees constructed by each of the heuristics.

Tree stretch is commonly measured in terms of the Relative Delay Penalty (RDP) [21]. RDP is defined as the ratio of the delay between a source and its receiver in an overlay tree to the delay if the source directly transmitted to that receiver. Figure 33 shows a source transmitting to 3 receivers in an IP Multicast capable network (left) and alternatively if ALM is used in a unicast-only network (right) with link delays shown beside each link. In the IP Multicast case, the delay between source0 and receiver2 is

25=5+10+3+7 while the ALM scenario results in a delay of 35=5+10+5+5+3+7 between source and receiver2, due to receiver1 forwarding its received stream to receiver2. The RDP for this source-receiver pair is thus 35/25=1.4. ALM protocols commonly measure the cumulative distribution of RDPs for all source-receiver pairs as a measure of the overlay tree stretch.

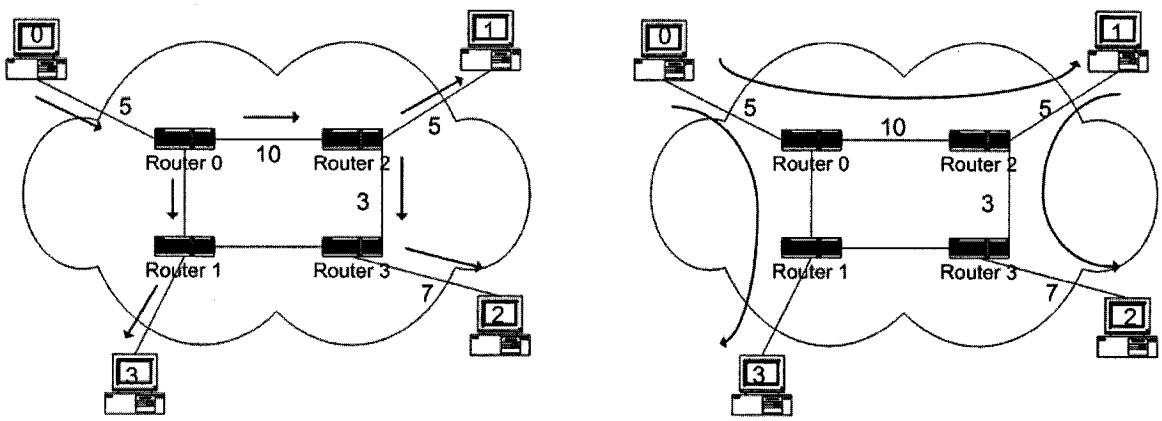


Figure 33. RDP and link stress for IP Multicast (left) and ALM (right)

The stress of all links is always 1 in an IP Multicast-capable network since routers only duplicate packets when needed, whereas stress can be more than 1 in an overlay network (in the example above the link between the source and router0 experiences a stress of 2)

In this chapter we will compare the rejection rate and RDP of MOTOR and MOTOR-Dynamic with that of CT, BDA, CLUSTER-swap, CLUSTER-Dynamic and DISPERSE-Dynamic. Since MOTOR allows for heterogeneous flow weights within the tree, there are three overall conditions under which we examine the efficacy of the said heuristics:

1. *Globally Homogeneous bit-rate*: represents the case where all receivers of all multicast sessions receive the same bit-rate (Figure 34a). This is the most commonly simulated condition (e.g. in [94] and [68]) but does not include the situation when different sessions have different bit-rates.
2. *Locally Homogeneous bit-rate*: represents the case where all receivers of a

particular multicast session receive the same bit-rate but different multicast sessions can have different bit-rates (Figure 34b).

3. *Heterogeneous bit-rate*: represents the most generic case where every receiver in a multicast session can request to receive a different bit-rate (Figure 34c).

The results of the comparison between the different heuristics are thus categorized according to the three conditions above.

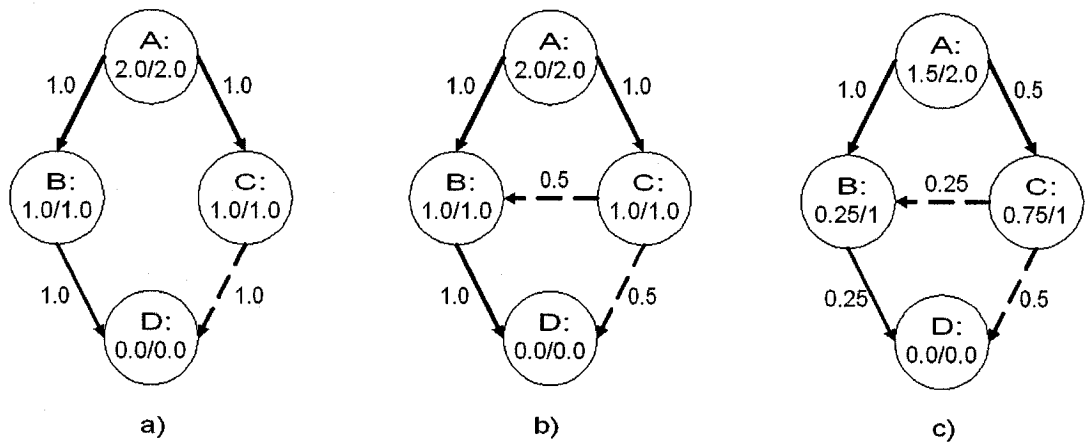


Figure 34. Three simulation scenarios: a) Globally homogeneous b) locally homogeneous c) heterogeneous

Also recall from section 4.3 that a multicast session can be assumed to be either static (no subsequent changes once it is established) or dynamic (nodes join and leave randomly). We try to present the performance of the mentioned heuristics including our own under the various conditions and simulation scenarios. Before doing so however, an important point is worthy of mention.

5.2 Comparing Metrics in Isolation

In [21] the authors point out the relationship between the out-degree assigned to each node and the effect on the RDP of the resulting overlay tree. When available out-degree is increased, RDP decreases. In general, it is important to keep in mind the effect of each

metric on the other metric especially when examining each metric in isolation as is commonly done. Consider for example, the trivial solution to the overlay routing problem that simply adds a node directly to the source if there is available degree at the source and rejects the node otherwise. In other words, the source unicasts to all its receivers until there is no more out-degree available. For this trivial algorithm to overlay multicasting, that we call 'multi-unicast' and called 'unicast' in [21], we expect a high rejection rate because every source can only support nodes that it can transmit itself. On the other hand, the RDP of the trivial solution will always be 1 or equivalent to the IP Multicast. It is important to note this tradeoff between rejection rate and RDP when comparing the RDP of routing algorithms whose rejection rates are significantly different. An algorithm with lower rejection rate may have a higher RDP but this does not necessarily imply that the algorithm builds higher stretch trees. By including more solutions, it may include higher stretch trees that were simply rejected by another algorithm with higher rejection rates. This is important to keep in mind in the sections that follow since the heuristics compared therein have widely different rejection rates.

Consider also the fact that when a heuristic does not find a solution to the routing problem, it can reject multiple requests (i.e. source-receiver pairs). In the dynamic simulation of multicast, this has a direct effect on the rejection rate of the heuristic. In the worst-case, if a heuristic rejects *all* the source-receiver pairs, then it can more easily accommodate future requests (since after rejecting all requests, it has plenty of spare out-degree to accommodate requests), thus lowering the percentage of time requests are rejection or in other words the rejection rate. On the other hand, when a heuristic cannot accommodate every source-receiver pair, it may try to accommodate as many as possible,

reducing the probability of accommodating future requests. As we shall see in section 5.3.2, some of the heuristic we study reject more than one request on average, when they fail to find a solution for all requests. The number of rejected requests per rejection is yet another hidden parameter when comparing rejection rates across multiple routing heuristics.

5.3 *Globally Homogeneous Scenario*

In this simulated scenario, used by Shi and Turner [94] [95] as well as Liu et al [68] to measure the rejection rate of their heuristics, every multicast session has the same bit-rate B and every multicast participant receives the same bit-rate. Normalizing the bit-rate, every edge of every multicast tree has a weight of 1 and nodes are assigned non-negative integers for maximum out-flow (i.e. the number of streams they can forward). Taking the same approach, we test the 4 static heuristics (MOTOR, CLUSTER-swap, CT, BDA) under both static and dynamic conditions and the 3 dynamic heuristics (MOTOR-Dynamic, CLUSTER-Dynamic, DISPERSE-Dynamic) under dynamic conditions only. The dynamic case involves nodes randomly joining and leaving multicast sessions (this is equivalent to teleconference participants randomly changing the sources they wish to receive from) whereas the static case involves creating a session out of a group of participants (i.e. all participants join at the same time and there are no subsequent changes to the session).

For both the dynamic and static scenarios, simulations were carried out with 4 to 9 nodes each with an out-degree maximum of 1 to 5. The reason for the small number of nodes is derived from the application domain of multi-party videoconferencing (see section 2.2.6) while the out-flow constraints are derived from upload bandwidth limitations of today's

DSL connection and the bit-rate of a 320×240 MPEG4 coded video at 15fps (see section 2.2.1).

Instead of randomly assigning maximum fan-outs to nodes as is commonly done, the smaller number of nodes used in our simulation affords the investigation of all possible combination of out-degrees. Given N nodes and maximum out-degree range of 1 to M, there are a total of C_{M-1}^{N+M-1} unique combinations of fan-out maximums that can be assigned to each node. Testing all possible configurations of fan-out maximums, as opposed to randomly assigning them as is commonly done, gives more confidence as to the generality of the results (but it is only feasible to do so for small number of nodes).

5.3.1 Dynamic Simulation of Multicast Sessions

For each configuration of maximum fan-outs assigned to each node, random join and leave events are generated to simulate the lifetime of multiple multicast sessions on the overlay structure. The generation of random join/leave events is restricted such that a node does not request to join a tree it is already attached to (unless it is a 'reflector' on that tree) and does not request to leave a tree it is not attached to. Also, the total number of withstanding join events is limited to be no more than the total of maximum out-degrees of all nodes (Note that in the globally homogeneous scenario out-degree and outflow are interchangeable). For example, for the maximum out-degree configuration of (1,1,2,3), the total number of withstanding join requests can be no larger than 7, thus avoiding the cases where accommodating all requests is obviously impossible, since the number of join requests is greater than the total number of available out-degrees. For each possible configuration, 10000 random join/leave events are generated and the number of requests that are rejected by all 4 algorithms recorded. Each request is also randomly

assigned a priority of either 0 (low priority) or 1 (high priority).

We repeated the simulation test 10 times and averaged the rejection rate across the 10 tests and the various out-degree maximum combinations.

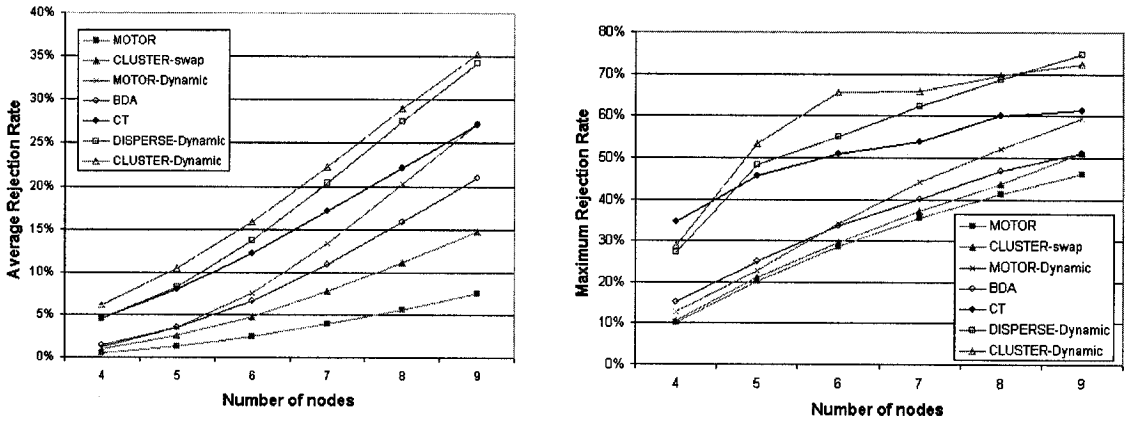


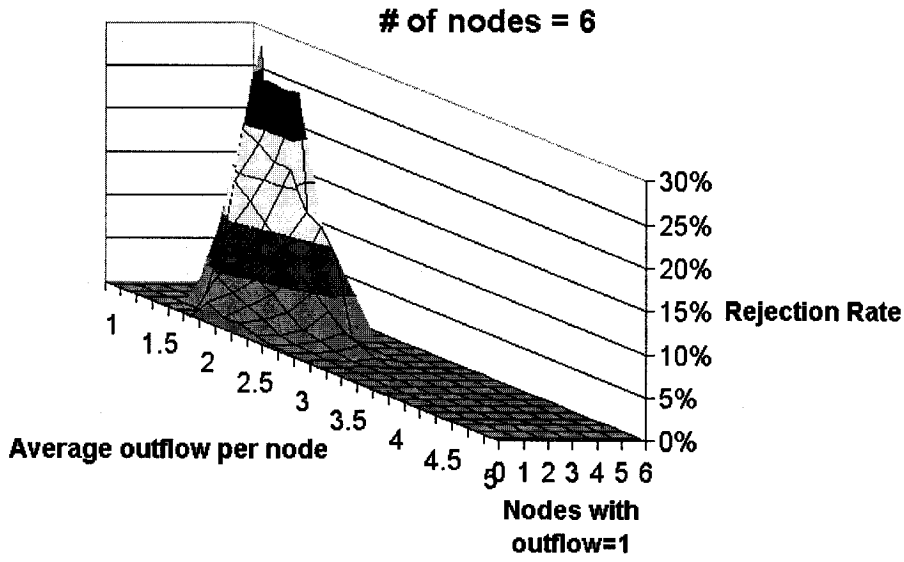
Figure 35. Results for Globally Homogeneous dynamic simulation scenario

Figure 35 illustrates the average and maximum rejection rate of the 7 heuristics for different number of nodes, while Table 2 gives the 95% confidence intervals.

Table 2. Average Rejection Rate and 95% Confidence Interval

Nodes	MOTOR	CLUSTER swap	BDA	MOTOR Dynamic	CT	DISPERSE Dynamic	CLUSTER Dynamic
4	0.5%±0.13	1.0%±0.17	1.4%±0.23	1.2%±0.20	4.5%±0.54	4.5%±0.54	6.1%±0.56
5	1.3%±0.20	2.5%±0.24	3.5%±0.30	3.5%±0.31	8.0%±0.53	8.3%±0.57	10%±0.56
6	2.5%±0.23	4.8%±0.26	6.6%±0.32	7.5%±0.37	12%±0.48	14%±0.55	16%±0.53
7	3.9%±0.24	7.7%±0.25	11%±0.30	13%±0.37	17%±0.41	20%±0.49	22%±0.46
8	5.6%±0.24	11%±0.23	16%±0.27	20%±0.33	22%±0.34	28%±0.40	29%±0.38
9	7.6%±0.23	15%±0.20	21%±0.22	27%±0.28	27%±0.28	34%±0.32	35%±0.30

Our heuristic (*MOTOR*) performs significantly better by finding a greater percentage of solutions as compared to the other heuristics: MOTOR has half the rejection rate of CLUSTER-swap, which has the lowest amongst the other heuristics. MOTOR-Dynamic performs significantly better than all other dynamic heuristics and for smaller number of nodes, performs better than CT.



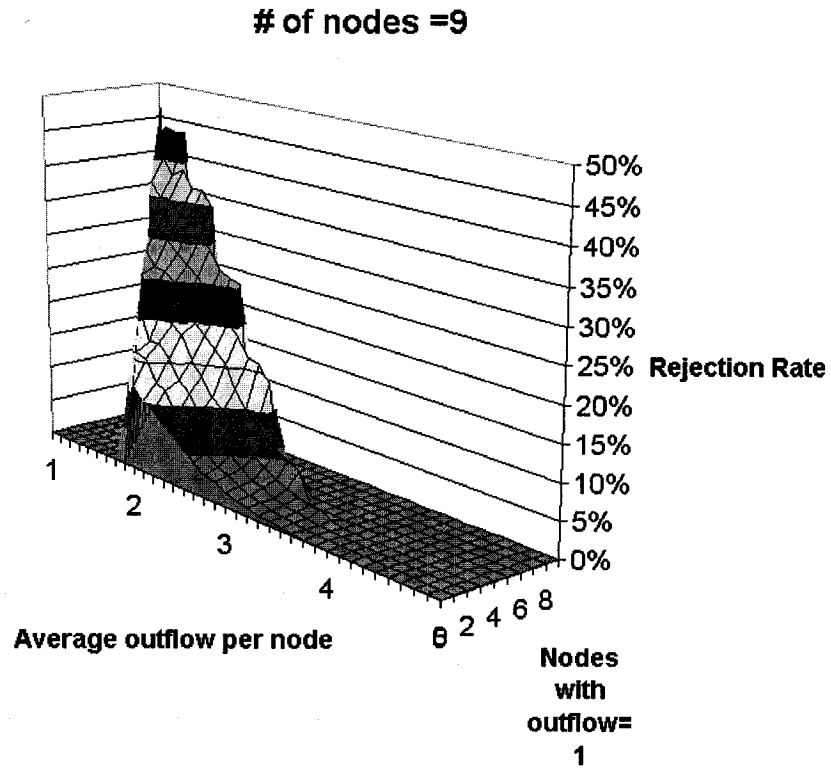


Figure 36. Rejection rate as a function of assigned outflow

The maximum (worst-case) rejection rates of all 4 heuristics are similar and occur for severely restricted configurations of fan-out maximum. Rejection rate is highly dependent on the total available out-degree assigned to the nodes. Figure 36 is a plot of the rejection rate for our heuristic for various configurations of fan-out maximum assigned to 6 and 9 nodes. As expected rejection rate decreases as more outflow is assigned to each node. The worst-case (maximum) rejection rate occurs when all nodes have fan-out maximum of 1 (i.e. each node is only able to transmit a maximum of one stream), making it less feasible to find overlay multicast trees.

Note that in Figure 36, the left part of graph with rejection rate of 0 corresponds to unfeasible outflow assignments. For example, if there are no nodes with outflow of 1 (i.e. number of nodes with outflow of 1 is zero), then the minimum outflow per node is 2 and a lower value of average outflow per node is not feasible.

In the context of teleconferences, these results emphasize that the existence of participants whose out-going bandwidth can only accommodate sending one stream (e.g. a user with a dial-up modem connection in an audio/video teleconference) reduces the feasibility of constructing overlay trees. This is more so as the number of such nodes/users increases.

When a heuristic fails to find spanning trees for every source, it rejects requests that cannot be accommodated. Since priority of either 0 or 1 were assigned, heuristics other than MOTOR (including MOTOR-Dynamic) that do not consider priority of requests, on average rejected a request with priority 0.5 (i.e. it is just as likely that a high priority request will be rejected than a low priority one) whereas MOTOR had an average priority of 0.1 for rejected requests (i.e. a high priority request has only a 10% chance of being

rejected). This is beneficial since, as mentioned in section 2.2.4, many multi-party videoconferencing applications assign different priorities to each stream and it is preferable to deny lower priority requests when it is not possible to accommodate every request.

Figure 37 shows the role assigned outflow plays in determining average rejection rate. As expected, rejection rate of all heuristics falls as more outflow is assigned to each node while MOTOR's decrease is at a faster rate. Although Figure 37 shows only the results for 6 and 9 nodes, results for 4, 5, 7 and 8 nodes follow the same pattern.

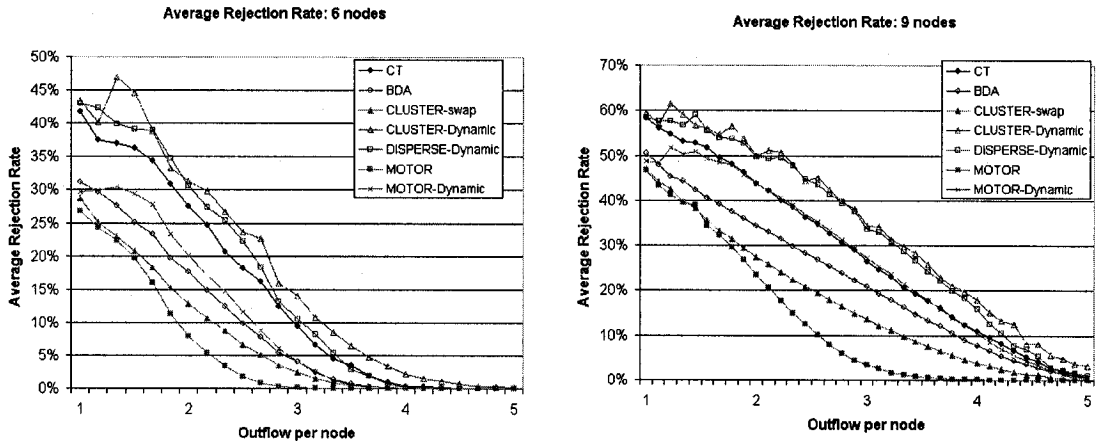


Figure 37. Average rejection rate versus outflow per node for 6 nodes (left) and 9 nodes (right)

A few more observations are due. Exclusively dynamic approaches to overlay multicast routing such as CLUSTER-Dynamic and DISPERSE-Dynamic have the worst (highest) rejection rates due to the fact that they try to add a late joining node by finding available out-degree on the existing tree as opposed to constructing the entire tree from scratch (as is done in exclusively static approaches). If there is no available out-degree available, the late joining node is rejected without considering the rearrangement of branches in the existing trees in order to accommodate the new node. The advantage of a dynamic approach is that it does not greatly disturb existing flows in order to accommodate a new

flow. When outflow is severely restricted, dynamic approaches will reject a high percentage of late join requests but as more outflow is made available, rejection rate decreases. The tradeoff between stability and rejection rate is yet another factor to consider for overlay routing algorithms.

In order to compare the stretch of the trees constructed by each heuristic, we conducted simulations with 4 to 9 nodes and assigned each node a random out-degree between 1 and 5 inclusive. Similar to the disk configuration used in [94], each node is randomly placed in a 2D space with randomly assigned coordinates so as to simulate random distance between nodes. For each node, the x and y coordinates are between 50 and 1400 milliseconds. The delay between any pair of nodes is simply calculated as the distance between their coordinates, resulting in a minimum delay of 50ms and a maximum of 2s $\approx$1980ms=$\sqrt{(1400)^2+(1400)^2}$. Given the randomly generated edge costs (i.e. node to node delays) and each node's randomly assigned out-degree, we generated 1000 random join and leave events for each of the 20 randomly generated out-degree assignments. For each event, if a heuristic is able to find a solution, the RDP of all source-receiver pairs on trees constructed by each algorithm is recorded (if a heuristic does not find a solution and rejects the request, there is no change in the RDP of all requests). In order to get confidence intervals for the 95 percentile RDP values, we carried out each test 10 times. Table 3 shows the results and Figure 38 plots them on a graph.

As mentioned in section 4.5, all heuristics except CT, MOTOR and MOTOR-Dynamic do not consider edge costs when building overlay trees. As a result, we expect CT, MOTOR and MOTOR-Dynamic to have better RDP than other heuristics as shown in Figure 38. BDA has the worst RDP since it does not consider tree cost at all while

CLUSTER-swap, CLUSTER-Dynamic and DISPERSE-Dynamic consider tree depth or hop length as a measure, which does not accurately represent source-receiver delay.

Table 3. 95 Percentile RDP and 95% Confidence Intervals (N=10)

Number of nodes	BDA	CLUSTER swap	DISPERSE Dynamic	CLUSTER Dynamic	MOTOR Dynamic	MOTOR	CT
4	3.8±0.18	2.8±0.09	2.7±0.12	2.2±0.29	1.9±0.26	1.5±0.15	1.2±0.06
5	5.1±0.34	3.6±0.17	3.6±0.17	3.3±0.22	2.8±0.18	1.9±0.12	1.3±0.05
6	5.2±0.23	3.9±0.15	3.7±0.16	4.0±0.17	3.7±0.18	2.4±0.11	1.4±0.04
7	5.5±0.25	4.2±0.16	4.1±0.19	4.1±0.15	4.1±0.21	2.6±0.16	1.5±0.03
8	5.9±0.17	4.36±0.11	4.3±0.13	4.3±0.09	4.3±0.14	2.9±0.12	1.5±0.02
9	6.0±0.16	4.53±0.13	4.5±0.12	4.3±0.11	4.4±0.12	3.0±0.1	1.5±0.03

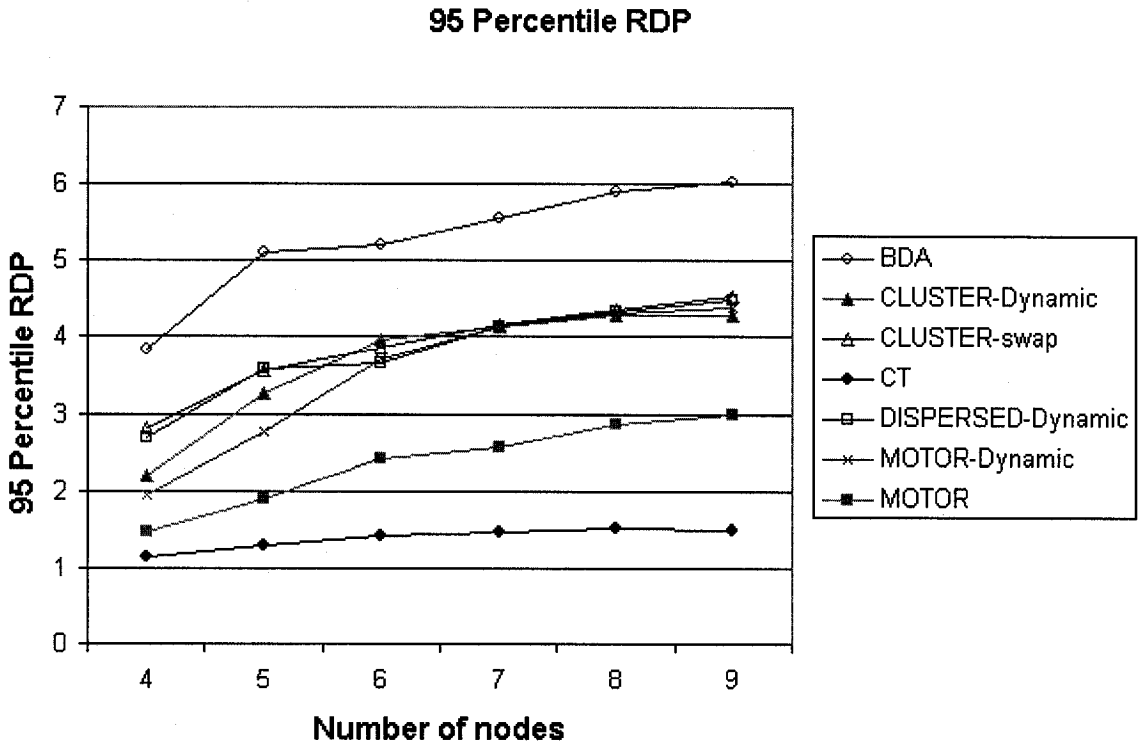


Figure 38. 95 percentile RDP

At this point we recall the point made in section 5.2 regarding the tradeoff between rejection rate and RDP values. Figure 38 does not fairly compare RDP across multiple heuristics since the heuristics widely differ in their rejection rate. We included the RDP

values shown in Figure 38 for completeness as well as to represent the RDP values that can be expected from each heuristic but a more fair comparison of RDP will be presented in the next section when we examine the static simulation of multicast sessions and are able to provide all heuristics the same input.

5.3.2 Static Simulation of Multicast Sessions

For a static simulation, multicast sessions are created on a group basis (all nodes join at the same time and no subsequent changes occur). Though not applicable to scenarios where nodes join and leave during the multicast session, the static simulation can better reflect the efficacy of a heuristic in finding solutions as well as allowing a case-by-case comparison of solutions across multiple heuristics.

For each configuration of out-degree maximums assigned to each node (generated the same as section 5.3), we can generate a (possibly empty) receiver set for each source and observe whether the various heuristics can find overlay trees for each source spanning its receiver set. Furthermore, for a small number of nodes (4 to 6) we are able to generate all possible combinations of source-receiver pairs under particular conditions. Given N nodes, there are a total of $C_M^{N(N-1)}$ unique combinations of receiver sets such that there are a total number of M receivers (with N nodes there are N×(N-1) possible source-receiver pairs and we must choose a set of size M). Considering 5 nodes for example, there are 5×4=20 source-receiver pairs as shown in Table 4. The fully loaded scenario occurs when the number of requests is equal to the total out-degree available. If the out-degree maximum configuration of the 5 nodes is (2,2,2,3,3) for instance, then the worst-case scenario is when there are 12 requests. In such a case every out-degree of every node will

be exhausted. Thus there are a total of $\binom{20}{12} = 125970$ combinations of 12 source-receiver pairs.

Table 4. Generating source-receiver pairs

		Receivers				
		1	2	3	4	5
Sources	1					X
	2					X
	3	X	X			X
	4	X				
	5		X	X	X	

Table 4 shows an example of 12 source-receivers chosen (marked by 'X') among the 20. The receiver set for source 1 is {5} while it is {1,2,5} for source 3. Compared to randomly generating multicast groups as is commonly done, generating all possible combinations of source-receiver pairs for a given out-degree configuration and simulating all possible configurations of out-degree can give more confidence about to the obtained result. It is only possible to do so however for small number of nodes since the number of test cases grow factorial-wise as the number of nodes increases. As a result we were only able to test all possible cases for 4, 5 and 6 nodes.

For each configuration of out-degree maximum for 4, 5 and 6 nodes, we can thus generate all possible combinations of fully loaded scenarios and observe the rejection rate for MOTOR, CT, BDA and CLUSTER-swap. Note that CLUSTER-Dynamic, DISPERSE-Dynamic and MOTOR-Dynamic are not applicable to the static case. Each

request (i.e. each receiver-source pair) is also randomly assigned a priority of 0 (low priority) or 1 (high priority).

Figure 39 illustrates the average rejection rate (the ratio of the number of source-receiver set combinations that a heuristic failed to find a solution for, over the total number of combinations possible) for 4, 5 and 6 nodes. Figure 39 also illustrates the rejection rate of

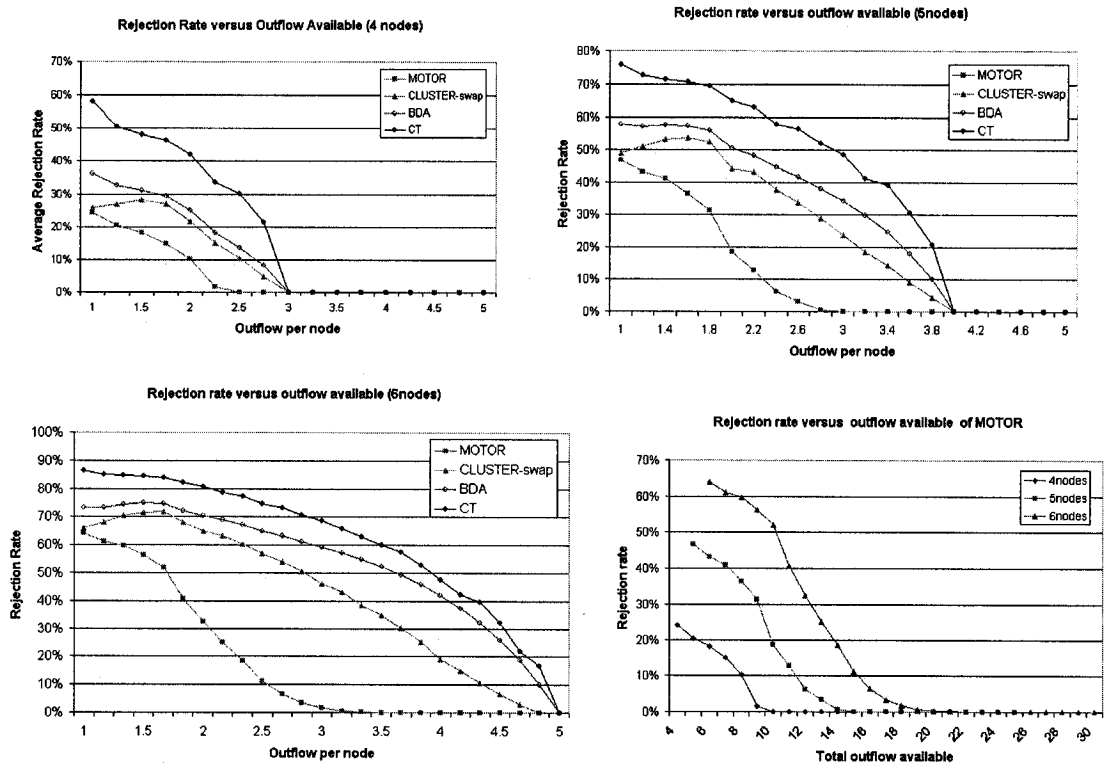


Figure 39. Static simulation results in Globally Homogeneous scenario

MOTOR as the number of nodes is increased for the same total outflow available.

Given the same input, our heuristic performs significantly better by finding a greater percentage of solutions. As expected, as the number of nodes increases, it takes more outflow per node in order to accommodate all the requests and the rejection rate is very high when the outflow constraints are very tight (1 outflow per node for instance). In addition to recording the number of combinations that each heuristic rejected, we also

recorded how many requests were rejected every time a source-receiver combination set was rejected. Figure 40 shows the average number of requests (i.e. source-receiver pairs) each heuristic rejected once it could not find a solution satisfying all the pairs.

When it is not possible to accommodate all requests, MOTOR and CLUSTER-swap try to accommodate as many requests as possible but BDA and CT reject more requests. As presented in section 4.4, heuristics that consider each tree independent of others can ask a node to be a parent in a tree while that node does not have enough out-degree to become the root of its own tree. In such a case, the entire tree rooted at that node is rejected, resulting in a higher number of rejected requests per rejection.

It should also be mentioned that similar to the previous simulation the average priority of rejected requests for MOTOR was 0.1 and for others 0.5, signifying that MOTOR denies requests with lower priorities whereas the other heuristics do not consider priority of a request when rejecting a request.

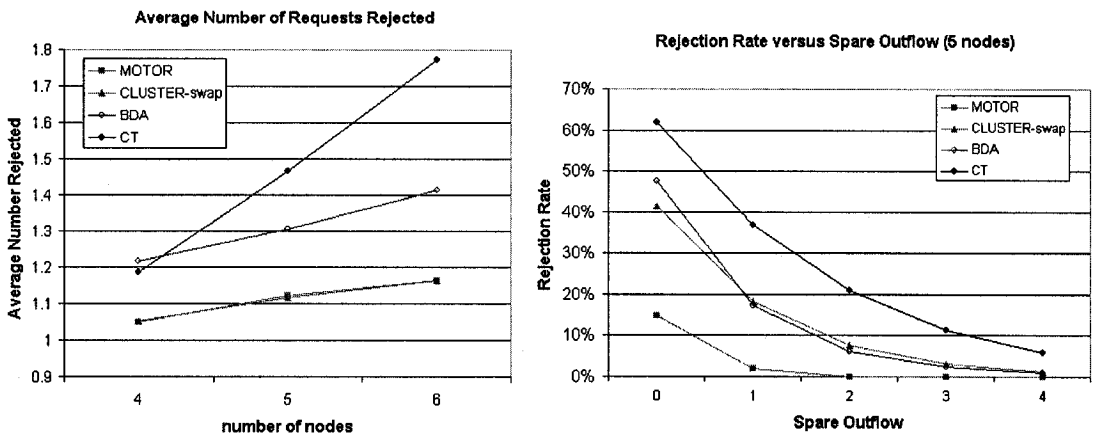


Figure 40. Average number of rejected requests (left) and rejection rate under partially loaded conditions for 5 nodes (right)

As mentioned earlier, the results shown in Figure 39 correspond to the worst-case scenario where the number of requests equals the total outflow available from all users

(i.e. the system is fully loaded). Figure 40 shows the rejection rate of the four heuristics under partially loaded conditions. 'Spare outflow' corresponds to the difference between the total outflow available from all nodes and the number of requests. In other words, the amount of remaining outflow that is not used up if a solution is found. As the load on the system is reduced, all heuristics find it easier to find solutions. As will be seen in section 6.4, in a typical awareness-driven multi-party videoconference we may expect the system to be fully loaded only for a small fraction of the entire conferencing session (3% of the time in the sample conferences conducted). An application can further limit the total number of requests in order to increase the chance of the routing heuristic being able to find a solution.

Static simulation allows for a case-by-case comparison across different heuristics. In other words, given a particular combination of receiver sets, we can discern which of the 4 heuristics can find a solution for it. The overall comparison thus yields relationships such as overlap, mutual exclusivity and inclusion between the solution spaces of the different heuristics. The rejection rate of the CT heuristic being too high, we only consider BDA, CLUSTER-swap and MOTOR. Between the three heuristics and given a particular combination of receiver sets, we examine the 2^3= 8 possible cases: when all three find a solution, when none find a solution, when only one finds a solution (3 cases) and when two find a solution at the exclusion of the third (3 cases). Table 5 summarizes the percentage of occurrence for each of the 8 possible cases. '0' signifies a heuristic not finding a solution and '1' signifies a heuristic succeeding to find a solution. The order of the three heuristics is MOTOR, CLUSTER-swap and BDA. Therefore a value of '101' in the table signifies the percentage of instances when MOTOR and BDA find a solution

and CLUSTER-swap fails to find a solution.

Table 5. Percentage of occurrence for each of the 8 possible cases for the 3 heuristics

Number of nodes	Total test cases	000	001	010	011	100	101	110	111
4	9671	12.8%	0%	0%	0%	5.2%	5.3%	8.8%	67.9%
5	6545811	14.8%	0%	0%	0%	16.7%	10%	16.2%	42.3%
6	1.3×10^9	11.8%	0.02%	0.02%	0.003%	30.861%	11.7%	21%	24.6%

From these results we can see that when MOTOR does not find a solution, the other two heuristics do not either in the great majority of cases, signifying that the solution space of our heuristic approximately includes the solution space of the other two heuristics (BDA and CLUSTER-swap) for small number of nodes. This is less true as the number of nodes increases. There is overlap of solution space between BDA and CLUSTER-swap (as evidenced by both heuristics finding solutions for some configurations) but they do not include one another (as evidenced by both heuristics finding solutions for configurations that the other does not find solutions for).

Given that the number of test cases grows factorial-wise when all possible combinations are tested, we were only able to test with only 4, 5 and 6 nodes. In order to test higher number of nodes, the static simulation is relaxed so that source-receiver pairs are generated randomly instead of examining all possible cases. Figure 41 shows the average rejection rate and the number of nodes that were rejected once a solution was not found, for each heuristic algorithm.

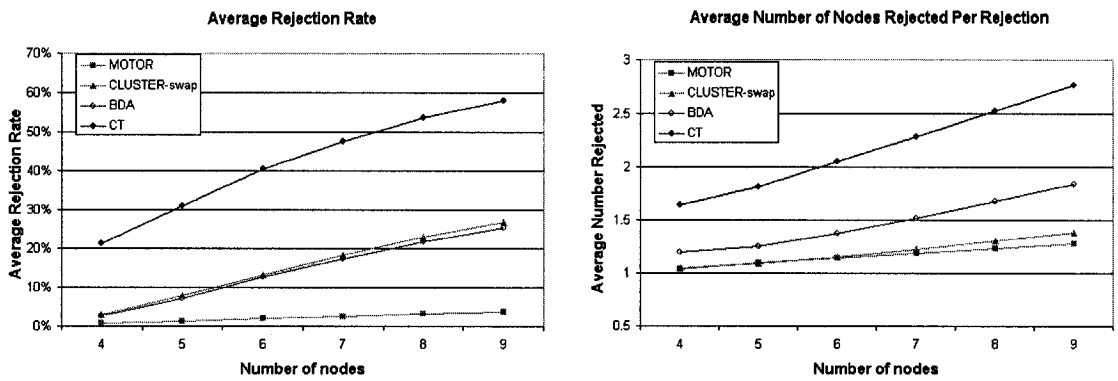


Figure 41. Average rejection rate and average number of rejected nodes in static simulation with randomly generated test cases

The results are obtained from randomly generating 10000 test cases for all possible combination of outflow maximum between 1 and 5 assigned to each user. Recall that the worst-case scenario (i.e. fully loaded) occurs when the total number of requests equals the total number of outflows available. For each test case, we randomly choose the total number of requests from a uniform distribution between the fully loaded scenario and half of that value. In other words, given a total outflow of M assigned to N users, we randomly choose the total number of requests, R, from a uniform distribution between M and 0.5×M. Having randomly chosen R, we can then randomly choose R unique source-receiver pair from the N by N-1 matrix, as shown in Table 4. The results of simulating 10000 of such test cases for all combinations of outflow maximum between 1 and 5 assigned to each node (nodes between 4 and 9) are shown in Figure 41. MOTOR has the best (lowest) rejection rate and along with CLUSTER-swap rejects fewer nodes than CT and BDA when it cannot find a solution for all the requests.

As before, we are also interested in knowing the quality of the trees produced by each of the 4 heuristics, but we recall from section 5.2 that comparing RDP values across heuristics with widely varying rejection rates can be misleading. A heuristic with a larger

rejection rate can produce trees only for the trivial test cases and consequently produce low stretch trees (e.g. star shape trees with every node connected directly to the source). Figure 42 illustrates with two sets of simulation results. For one simulation, each heuristic is tested with 10000 test cases, where each test case represents a randomly chosen out-degree maximum for each node (between 1 and 5), a randomly chosen coordinate in 2D space (as discussed in 5.3.1) to represent delay and with a randomly chosen number of source-receiver pairs as in the previous simulation. When a heuristic finds a solution for a test case, the RDP of each source-receiver pair is recorded.

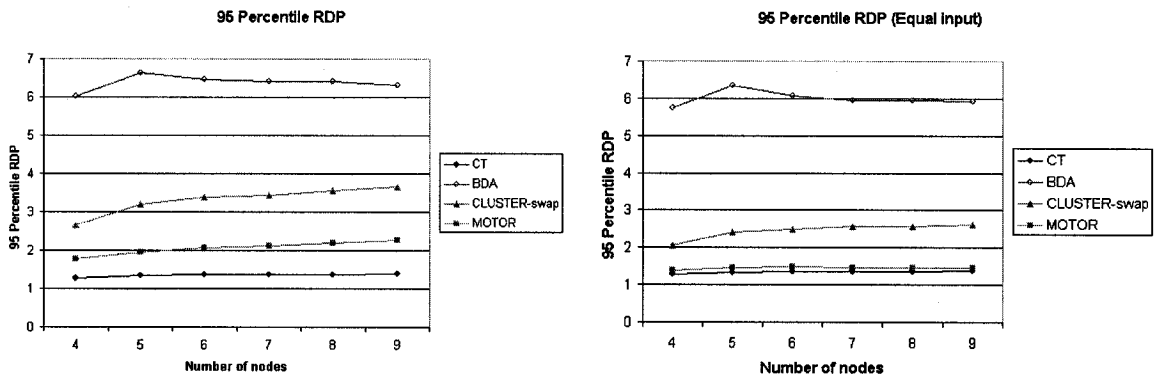


Figure 42. RDP with unequal input (left) and equal input (right)

Figure 42 (left) shows the 95 percentile RDP value of each heuristic for different number of nodes. CT has the best (lowest) RDP followed by MOTOR and CLUSTER-swap. BDA performs the worst since it does not consider delay as a parameter in constructing overlay trees. The second simulation is carried out the same way with the exception that RDP values are recorded only when all 4 heuristics find a solution, which we term 'equal input'. This is so that we can compare the quality of the trees constructed when all 4 heuristics actually construct trees. Figure 42 (right) shows the results of the second simulation. As can be seen, MOTOR, CLUSTER-swap and BDA have lower values of RDP (and hence closer to CT) compared to the previous simulations. This illustrates the

effect of the rejection rate on the RDP measurement. When a heuristic has a higher rejection rate compared to another, it is misleading to compare RPD percentile values between them since the heuristic with better (lower) rejection rate will accommodate more requests and will have a higher RDP as opposed to the heuristic that simply rejects some of the requests instead of accommodating them.

It is worth noting the close values of RDP between MOTOR and CT. Although MOTOR has a significantly better rejection rate than CT, it still has a low RDP and under the same circumstances (when both CT and MOTOR find a solution), MOTOR finds trees with very similar RDP characteristics than CT. This is a result of the two step process of assigning per-tree outflows to each tree and then constructing low cost trees from the result, as discussed in section 4.6.2.

The simulations so far assumed a globally homogeneous scenario, where all requests have the same flow weight (bit-rate). In reality, it is possible for different multicast sessions to have different bit-rates, leading us to the locally homogeneous scenario.

5.4 Locally Homogeneous Scenario

Before describing the simulation setup for this scenario it should be mentioned that CLUSTER-swap was not originally designed to handle different bit-rate multicast sessions. CLUSTER-swap swaps children between trees in order to free up outflow of a node and this is only possible if both trees have the same bit-rate. We therefore modify CLUSTER-swap so that swapping is done only between trees with the same bit-rate.

Assuming a reference bit-rate carrying a weight of 1.0, we simulated different multicast sessions with randomly assigned bit-rates of 1.0, 0.5 and 0.25. Maximum outflows are also randomly assigned to be one of 0.25, 0.5, 1.0, 2.0 or 3.0 with the restriction that the

multicast session bit-rate assigned cannot be larger than the maximum outflow of its source (once again a priority of either 0 or 1 was also assigned to each source-receiver pair). For different number of nodes from 4 to 9, we simulated 1000 scenarios (randomly assigned multicast session bit-rate and maximum outflow for each node) and for each scenario, we simulated 10000 random join and leave events similar to section 5.3.1: a node can request to join any node it is not already receiving from as long as the sum of flow weights of all withstanding requests does not exceed the total outflow of all nodes. A node can request to leave the tree of a node it is receiving from. Figure 43 illustrates the 90 percentile, average and maximum rejection rates between the 4 heuristics.

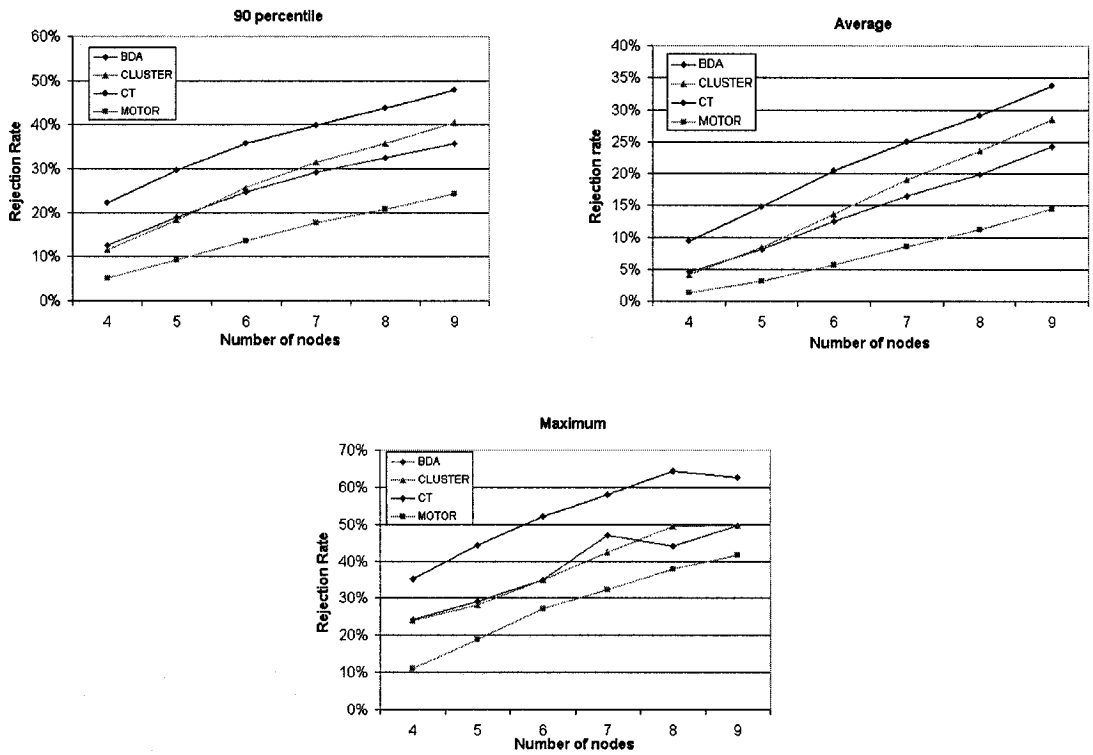


Figure 43. Locally Homogeneous simulation results

Once again our heuristic finds a greater percentage of solutions as compared to other heuristics. The increase in the value of the average rejection rate from the globally

homogeneous case to the locally homogeneous case is also worthy of note, signifying that it is more difficult to find solutions for the locally homogeneous scenario due to its more restrictive nature (it is more difficult to swap or relocating spare outflows since different trees have different bit-rates). It should be mentioned that, unlike other heuristics, when our heuristic does not find a solution, it rejects lower priority requests first.

In the simulations so far all receivers within the same multicast session received the same bit-rate. It is possible, however, for different receivers to request a different bit-rate from the same source based on their preference, capability or due to different levels of awareness/focus (see section 2.2.5). As a result, the most generic formulation of the problem given in section 4.1 allows for every node to potentially request to receive a different flow weight from a source.

5.5 *Heterogeneous Scenario*

Although the implementation of a multi-party conferencing application that incorporates video adaptation and heterogeneous bit-rates for every receiver encompasses many issues and challenges, in this paper we concentrate on the routing problem. It was with this aim that our heuristic was designed to allow for heterogeneous bit-rates within the same session (see section 4.6 and appendix for details) and this section will present our evaluation of it in the most generic scenario. Note that the three heuristics used for comparison in the two previous sections (BDA, CT, CLUSTER-swap) cannot be used in this section since they do not allow for different bit-rates within the same multicast session. Therefore, in this section, we compare our heuristic to a 'naïve' algorithm whose general approach is outlined in Figure 44.

For each tree, 'naïve' simply attaches the candidate receiver with the highest bit-rate request first and attaches it to the node in the tree with the largest available outflow. If two receivers have the same bit-rate request, then the one with the largest available outflow is chosen as the candidate receiver. If no node on the tree can support the candidate receiver, the candidate receiver is rejected.

```
Begin
Ts={s}
While Rs not empty:
   find r∈Rs with maximum w(r,s) in Rs
   if w(q,s)=w(t,s), q,t∈Rs:
      r = node with larger outflow_max-outflow
   find p∈Ts with maximum outflow_max(p)-outflow(p)
   if outflow_max(p)-outflow(p)>=w(r,s):
      add edge {p,r} to Ts
      Rs-={r}
End
```

Figure 44. A 'naïve' algorithm for heterogeneous simulation scenario

When simulating this scenario, like the previous scenario, each source is randomly assigned a bit-rate weight of 1.0, 0.5 or 0.25 and each node is randomly assigned outflow maximums of 3.0, 2.0, 1.0, 0.5 or 0.25 with the restriction that the source's bit-rate cannot be larger than its outflow maximum. Unlike the previous scenario however, each receiver can request a bit-rate less than or equal to the source's bit-rate. For example, a receiver can request to receive a flow of weight 0.25, 0.5 or 1.0 from a source with bit-rate 1.0. In this way, a multicast session can comprise of nodes receiving different bit-rates of the same flow. Figure 28 shows the 90 percentile, average and maximum rejection rate for our heuristic and the 'naïve' algorithm for 4-9 nodes.

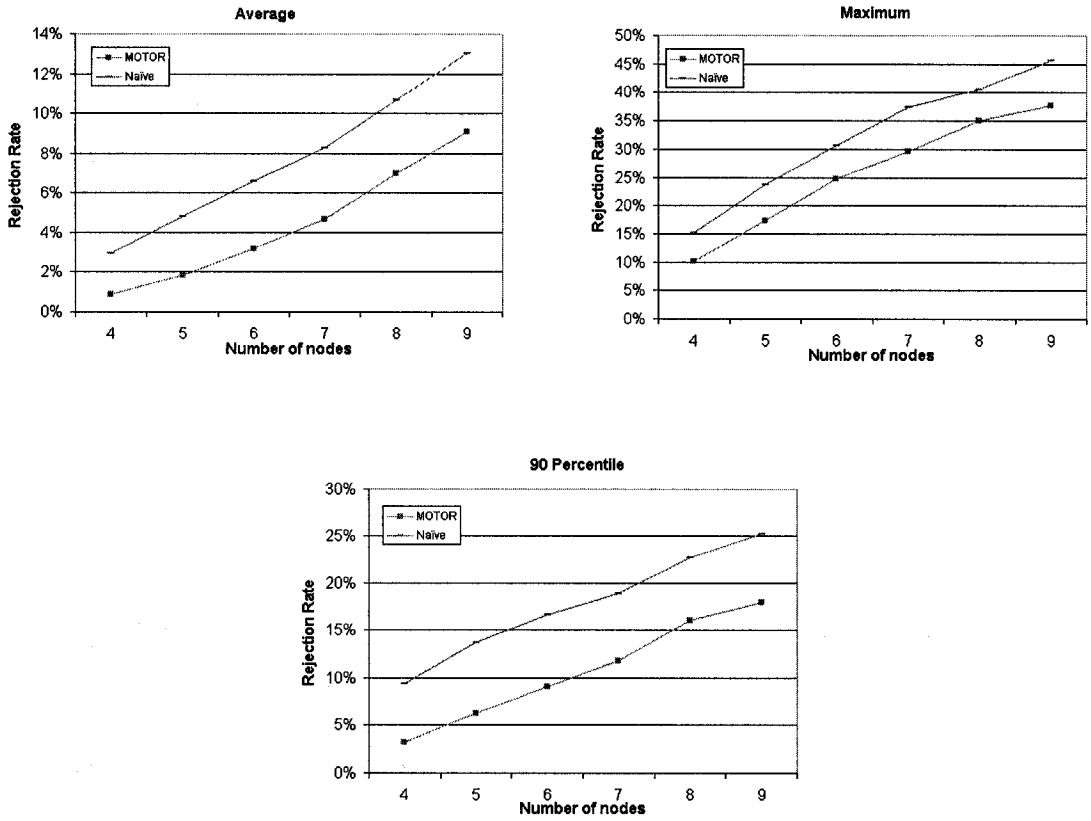


Figure 45. Results for globally heterogeneous simulation

Once again, our heuristic is able to find a greater percentage of solutions than a 'naïve' approach. It should be noted that the improvement provided by our heuristic is less compared to the previous two scenarios. We conjecture that this is due to the greater restrictions put on the construction of trees when considering different flow weights within the same tree. Not only the resultant tree must respect the outflow maximum of each node, but also there is an additional requirement of a node higher in the tree receiving a flow weight higher or equal to its children (i.e. nodes with lower flow weight should be at the bottom of the tree). However, carrying out the globally heterogeneous simulation scenario serves to illustrate the flexibility of our heuristic in being adaptable to the most generic formulation of the problem.

5.6 *Algorithm Complexity*

For an N node session, the complexity of both CT and BDA algorithms are $O(N^3)$ for constructing a single tree. Since there may be a tree for every node, the complexity of both CT and BDA is then $O(N^4)$ for the MD-MSOC problem. CLUSTER-swap has a complexity of $O(N^2 \log N)$ for building one tree and therefore has a complexity of $O(N^3 \log N)$ when every source has a tree. MOTOR on the other hand has a complexity of $O(N^8)$ due to its more rigorous search for solutions as well as incorporating request priority and heterogeneous bit-rates in the routing algorithm. As discussed in section 4.6.1, removing the consideration for priority of requests and removing heterogeneous bit-rates within the same session, the complexity of MOTOR is reduced to $O(N^6)$. Given that MOTOR is designed for small-group conferencing sessions with 4-10 participants and the significant improvement it provides compared to similar heuristics, the significantly higher complexity of MOTOR may be seen as justified. Once again an exhaustive solution has a complexity of $O(N^{N\times (N-2)})$ and still does not find all the solutions (such as ones requiring reflectors) as shown in section 4.2.

CLUSTER-Dynamic and DISPERSE-Dynamic both have time complexity of $O(N)$ while MOTOR-Dynamic has the worst case complexity of $O(N^3)$ (if there is no limit on the number of changes) in return for having a significantly better rejection rate and constructing lower cost trees. Note that if the maximum allowable changes to topology is 4 or less, then the complexity of MOTOR-Dynamic equals $O(N)$.

The large complexity of MOTOR is one of the characteristics that may limit its scalability to large number of nodes (in the long run the centralized approach of all the heuristics used in this chapter is another characteristic that limits their scalability). For small number of nodes however, MOTOR is able to find a solution in a reasonable time.

When carrying out the simulation discussed in section 5.3.1 (dynamic simulation in homogeneous bit-rate scenario), we also recorded the average time it took each heuristic to arrive at a solution for each request.

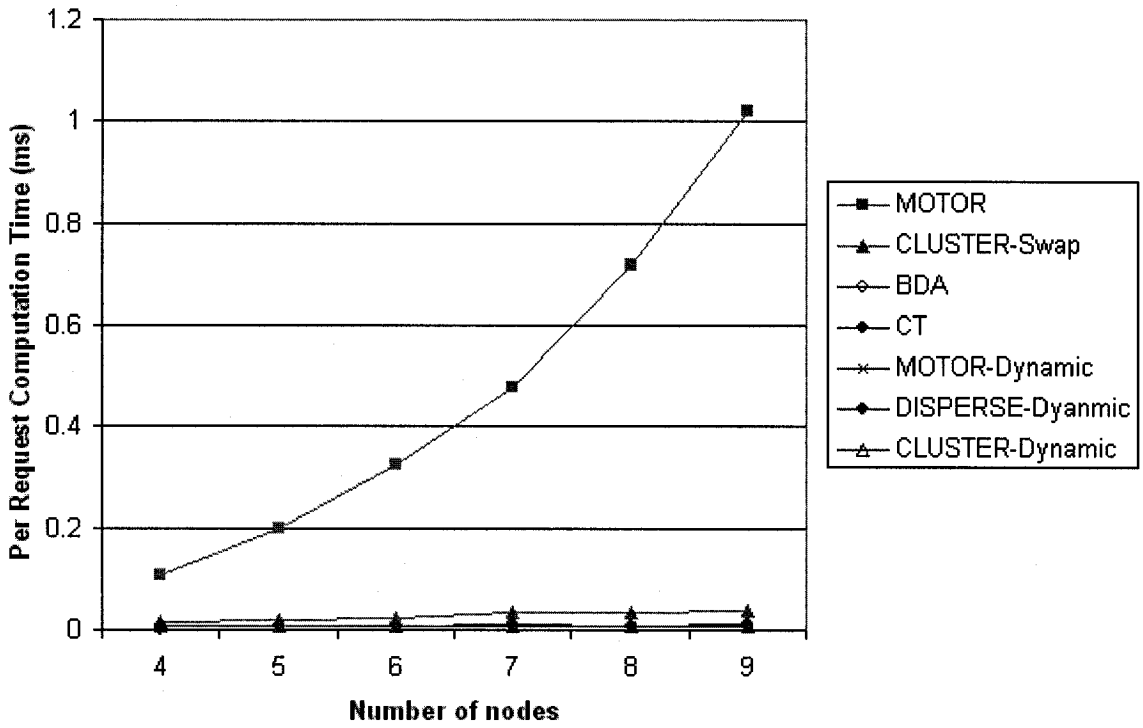


Figure 46. Average per request computation time of the heuristics

Figure 46 shows the average computation time of each heuristic for finding attaching a new node to an existing tree. The simulations were carried out on a 1GHz CPU computer. As expected from its complexity, the computation time of MOTOR dramatically increases as the number of nodes increases. Even for 9 nodes however, MOTOR is able to find a solution in about 1ms.

5.7 Summary

In this chapter we illustrated that our routing heuristics MOTOR and MOTOR-Dynamic find a greater percentage of solutions than similar overlay routing heuristics. In MOTOR we have additionally included consideration of stream priority and heterogeneous flow

weights, neither of which are considered in existing relevant routing heuristics. Furthermore in terms of delay, MOTOR achieves similar RDP values to CT which constructs low stretch trees. MOTOR-Dynamic exhibits very good rejection rate compared to other dynamic solutions and constructs low cost trees.

The advantages of MOTOR come at the cost of greater complexity of implementation as well as their applicability to only small and medium number of nodes but overall they present adequate routing heuristics for multi-source outflow-constrained scenarios such as multi-party videoconferencing applications.

Chapter 6 Implementation and Proof of Concept Application

This chapter represents the culmination of the work presented so far in bringing about a multi-party videoconferencing application that can be deployed on today's Internet. The assumption is that the participants are connected through a typical DSL link (i.e. without access to IP Multicast and with limited upload bandwidth). In this regard the chapter's contributions are a novel awareness management mechanism for 3D videoconferencing applications, the design and implementation of a multi-party videoconferencing application deployed over the ESM protocol described in previous sections (i.e. MVEMP) and the evaluation of MVEMP in the context of a real-world application as opposed to simulated scenarios.

6.1 Design Criteria and Requirements

As a class of multimedia tools, multi-party videoconferencing applications can take many forms. Earlier in section 2.1, we provided a brief overview of existing approaches to multi-party videoconferencing and argued for embedding live video into a computer generated 3D space in order to carry the notion of meeting in a physical conference room to the meeting of remote users in a virtual space. Furthermore, it has been argued that compared to traditional videoconferencing where multiple video streams divide and share a tiled 2D interface (Figure 47a), video embedded inside a 3D world can better support mutual awareness and spatial consistency for users [89] (Figure 47b). Mutual awareness refers to the idea of participants being able to infer where the focus and awareness of other participants lie and spatial consistency refers to the remote participants 'sharing' a

common 3D virtual space.

Within a 3D space users can tell where the attention of another participant is directed by way of his/her position and orientation. This has been further illustrated through attempts to provide gaze awareness in computer supported cooperative work that have also moved towards integrating the captured video or picture of users into a computer generated 3D space and tracking their head and/or eye movements in order to convey their gaze [99] [104]. More sophisticated interfaces involving videoconferencing in an augmented or mixed reality setting with the 3D video of the remote person superimposed on the real world have already emerged [83].

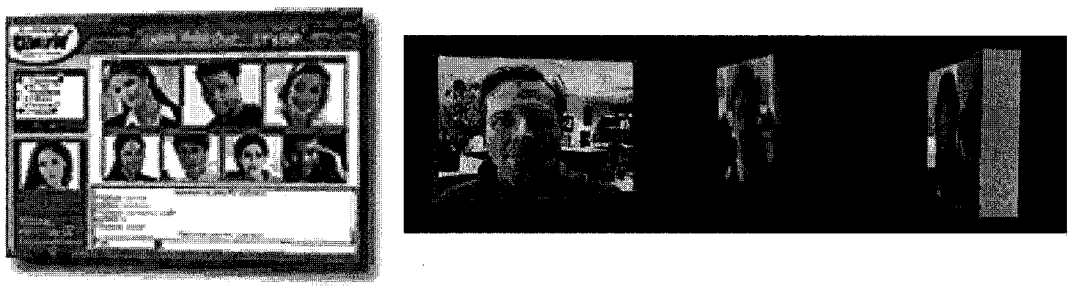


Figure 47. A 2D tiled interface (courtesy of [25]) and a 3D interface (courtesy of [104])

Videoconferences, be they in 2D or 3D interfaces, share the common limitation of scalability in their inability to support even a small group of participants (e.g. 4-10) without requiring specialized networks, compromising the fidelity of the video stream or reducing the effectiveness of communication (see section 2.1). In this context our goal is to develop a 3D videoconferencing system that can support 4-10 participants but should not rely on the availability of specialized networks and should instead try to stay within the bandwidth limitations of commercially available inter-network connections such as high-speed DSL connections provided by today's ISPs (e.g. 1Mbps maximum

downstream, 200-300Kbps maximum upstream). In addition, the systems should try to support real time video with acceptable quality (e.g. 320×240 at 5-15fps) in order to support the accurate and timely communication of body and facial gestures between participants. A further requirement of our application is to allow for multiple simultaneous sources of video as opposed to forcing only one stream to be seen by all participants. This requirement is derived from participants wishing to choose whom they see in a conference and finding the system dictating the received video as undesirable [92].

The incongruent ratio of available bandwidth and the number of video streams each user must transmit in a multi-sender conferencing application can be addressed at several levels. While advanced video compression techniques and minimizing frame rates reduce the bandwidth requirement of each video stream, advanced network protocols and topologies make more efficient use of the available network resources. Chen [19] gives an overview of current video compression algorithms appropriate for videoconferencing and the minimum required frame rate needed to at least detect some gestures relevant to floor control in such applications. For his virtual auditorium application, 320×240 at 15fps video is found to be sufficient to detect the facial expressions (e.g. a smile) of the captured user [18] while the lower bound on frame rate was found by Tang and Isaacs [98] to be 5fps in order for the video to be tolerable. Though even lower frame rates can be used for the non-speaking participants in order to reduce bandwidth requirements, Chen observed that full-motion video is necessary to convey gaze, facial expressions and body movements. In addition, a common complaint about lower frame rates was due to video of people being captured at moments that made them look "silly" (e.g. in the

middle of a movement) [19]. We therefore aim at supporting 320×240 video at 15fps in order to effectively convey body and facial expressions and possibly gaze. It should be mentioned that an MPEG-4 encoder can be expected to produce an average 100kbps bit rate for a 320×240 video at 15fps [18].

Though special high performance networks (e.g. ATM) are more apt at providing the bandwidth as well as Quality of Service requirements of multi-sender videoconferences compared to the best-effort Internet, they are expensive and generally not available to regular end-users. Instead ISPs currently provide best effort services with maximum downstream bandwidth of a few Mbps (typically 1Mbps) and maximum upstream bandwidth of less than 1Mbps (typically 200-300 Kbps) through high-speed DSL connections to the home. Our aims are to develop a videoconferencing system that can accommodate the bandwidth limitations of such connections and not rely on special networks.

Carrying out conferences on top of the Multicast Backbone (MBone) has the advantage that each user needs only to send his/her video to only one multicast address and let the network protocol replicate the stream to all subscribers (see section 2.1.2). Multicasting is a very crucial requirement for scalability of videoconferencing applications but it is not yet made available to home users by many local ISPs. We strive to provide multi-sender videoconferencing in the absence of IP Multicast since this is the case for most home Internet users.

An approach taken by commercial videoconferencing systems to support multi-sender videoconferences on the Internet has been to use "reflectors" or Multi-point Control Units (MCU) (see section 2.1.2). In its simplest form, a reflector or MCU multiplexes video

streams received from multiple clients to all other clients, giving the conference a star topology and reducing the number of independent connections from n^2 to 2n, where n is the number of users. In order to reduce the bandwidth requirements, either picture-in-picture is used (merging all videos into one) or voice-activated switching (transmitting video of only the current and previous speaker) [10]. While picture-in-picture reduces resolution of each video, voice activated switching can reduce the effectiveness of communication or be unpleasant as a result of not allowing users to choose whom they see [92]. As we already discussed in section 4.6.1, in our work we have included the concept of a reflector or MCU in our ESM protocol by allowing nodes with ample out-going bandwidth to reflect streams for sources in the session that are unable to transmit their own stream (due to their out-going bandwidth restrictions).

Finally, it should be mentioned that building a 3D videoconferencing application making use of a peer-to-peer topology and running on an ESM protocol (as opposed to using IP Multicast) sets our work apart from previous research on scalability of large-scale networked virtual environments (see Chapter 7 of [96] for an overview of these) and more specifically other 3D videoconferencing tools [4] [39] [40] [104].

We now proceed to outline the design of a multi-party videoconferencing application that makes use of the MVEMP protocol proposed in previous chapters and evaluate its efficacy in the context of an application. A critical component of the design is our novel visibility-based awareness management scheme, which will be presented in the next section.

6.2 *Visibility-based Awareness Management*

One of the underlying assumptions of current Application Layer Multicast protocols is that the upload bandwidth of receivers is available for forwarding a received stream (from the original source) to another peer. Considering the maximum upload bandwidth of today's DSL connections (200-300kbps), one can observe that each participant of a multi-party conference will not have enough upload bandwidth to transmit its own video, let alone forward the stream of another participant (assuming a 320×240 video at 15fps). It is therefore necessary to limit the number of simultaneous video streams that the system must deal with and yet provide users with the flexibility to choose whom they see. Traditional approaches such as picture-in-picture (merging N videos into one at a central point) or voice-activated switching (transmitting the video of current and previous speaker only) either reduce the quality of the video or do not allow each user to choose whom he/she sees. In addition to providing better mutual awareness and spatial consistency as argued in [89], embedding video in 3D virtual spaces can provide a flexible interface for users to choose whom they see. Furthermore, the number of users each can have in view is limited as a result of the limited field of view of a typical desktop screen. Therefore, the first step towards the design of a multi-sender videoconference is to devise a mechanism whereby the system can determine the mutual awareness of users in order to stream only those video streams that they are aware of (called awareness driven video [89]).

The concept of mutual awareness defined based on spatial distance between users in the virtual world (e.g. MASSIVE's aura/nimbus/focus [39]) serves as a flexible model of awareness and has in fact been used to build an awareness driven video quality of service architecture [40]. We extend such work by proposing and implementing a visibility-based

awareness model as computed by each user's rendering mechanism.

The novelty of our approach lies in exploiting recent development in graphics card technology and API that allows an application to query the occlusion of objects in the rendered scene to determine the mutual visibility of users. In order to accelerate graphics rendering, manufacturers such as NVIDIA [79] and ATI [3] have added the capability to query the occlusion of an object by rendering the object and reporting the number of pixels drawn as a result of the query. By querying the occlusion of simpler primitives that encompass more complex polygonal models, rendering of large polygonal sets can be accelerated [8]. We propose to use the same capability to determine the visibility/occlusion of video sources in a 3D videoconference. As a result the visibility or occlusion of a participant (represented by a video textured object in the 3D scene) leads to receiving or filtering of the corresponding video stream. If only a limited number of participants are visible at a given time, such a mechanism would limit the number of sent/received video streams and yet allow users the flexibility to choose whom they see.

Our visibility-based awareness management scheme has the advantages of efficiency, flexibility and accuracy as compared to existing approaches such as one used by MASSIVE [39] that are spatial distance based.

6.2.1 Client-side Awareness Detection

Assuming each participant is informed of the position in 3D virtual space of other participants in the session, each participant can query his/her own rendering mechanism as to the visibility of other participants and does not need a common awareness-managing server to detect collision of focus and nimbus (as done in MASSIVE and other similar systems) thus eliminating the extra delay of going through a managing server.

Note that in our approach, the application developer is also saved the effort to implement the awareness collision-detecting module of the awareness manager and the computation time since this is done by the hardware graphics rendering utility. Instead, the application makes a query of the visibility of each participant locally and subscribes to only those video streams whose geometrical representation are visible.

6.2.2 Terrain and Occlusion Aware

The spatial trading approaches similar to that employed by MASSIVE ignore the possible impact of the terrain or environment in which a virtual conference takes place with regards to user's awareness of one another and instead only takes into consideration the spatial distance between users. Figure 48 represents a 3D virtual space where multiple users 'meet'. Each user has a representative embodiment in the virtual world (called his/her avatar) that can be seen by other. The movement of each user is transmitted to all other users so that all users have a consistent view of the 3D world. The job of the awareness management (be it spatial distance based or visibility based) is to infer the level of awareness between every pair of participants. The system can then use awareness information to filter out data that does not lie in the sphere of interest or awareness of each user. Figure 48 illustrates possible inaccuracies of spatial trading as compared to visibility-based visibility calculations

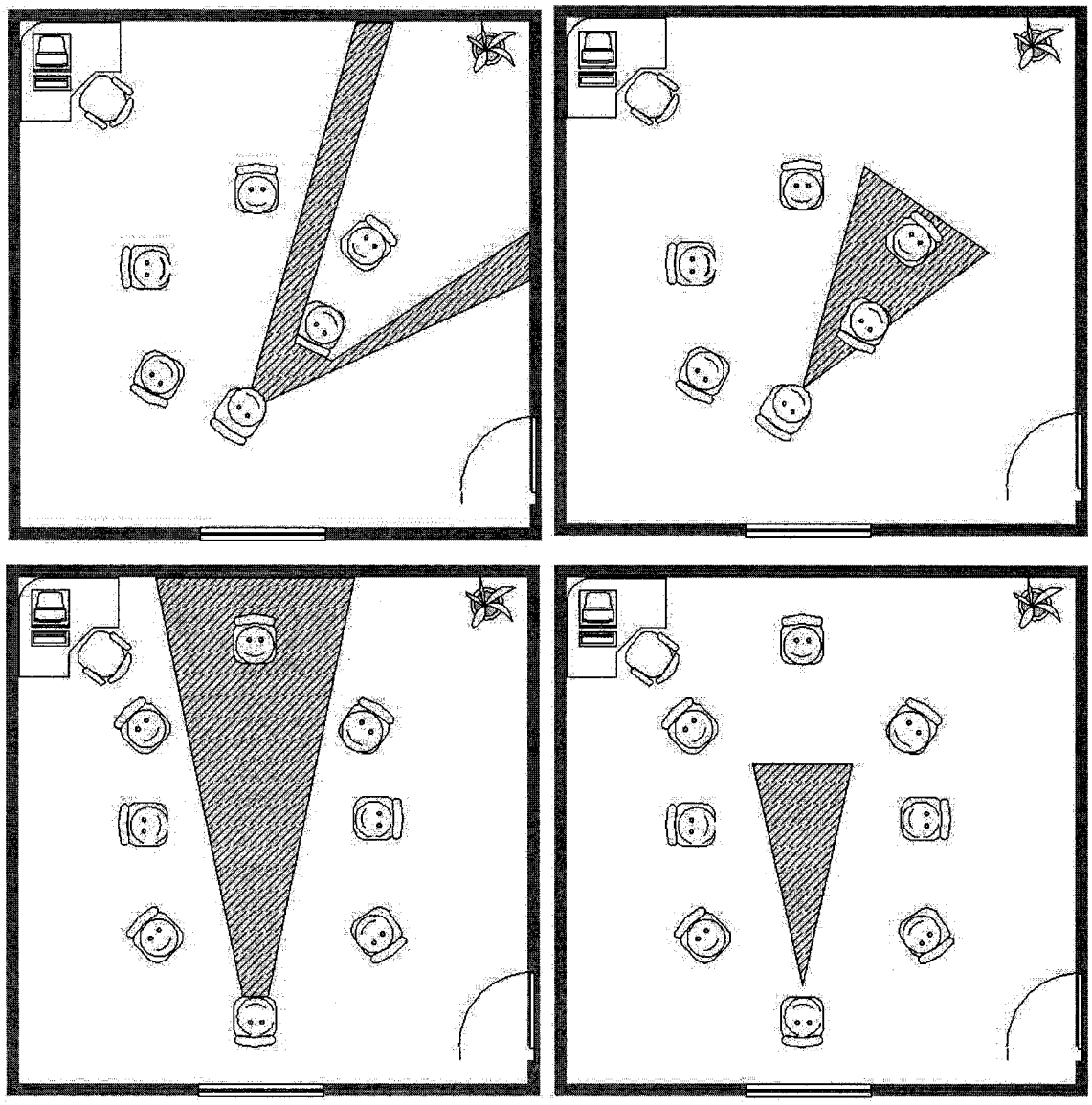


Figure 48. Spatial distance based awareness (right) compared to visibility based awareness (left)

In the setting of a 3D videoconference, participants may be occluded by other participants, terrain elements (e.g. walls) or artifacts in the world (e.g. a 3D model of a car that is the subject of discussion). Querying the graphics renderer for visibility would accurately take these into account while a spatial approach would determine a non-existing mutual awareness as a result of spatial proximity. Conversely, two participants may be relatively far apart but in clear view of one another. In both of these instances, a

visibility-based approach determines the correct awareness and a spatial distance approach does not.

In addition to excluding video whose source is out of view, this approach allows for pruning of streams when the source has its back to the viewer or when it is only partially visible. For example, by putting a 'wall' behind a video textured object, a source with its back to the viewer will result in its occlusion and therefore the omission of the corresponding stream (Figure 32)

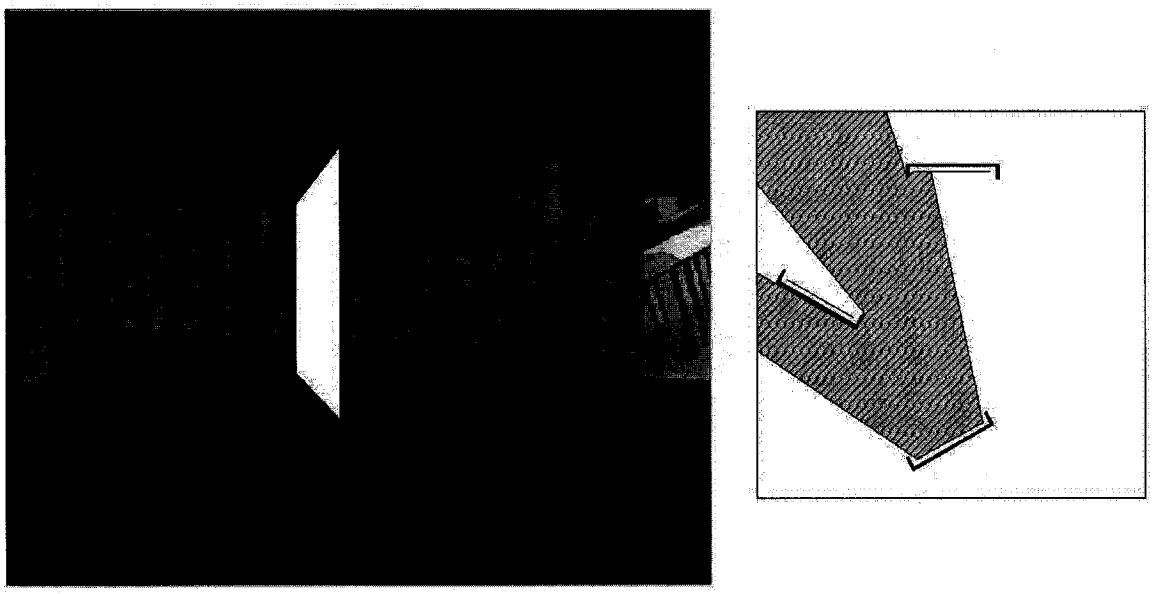


Figure 49. Detecting a user turned away in visibility-based awareness

In summary, since in our approach the visual awareness of other users is directly determined by the graphics engine rendering the 3D scene, querying the same graphics engine for visibility information gives a more accurate measure of mutual awareness.

6.2.3 Flexibility

It should be noted that the need to define the spatial area of interest explicitly for each user in the spatial-based approach to awareness could lead to inflexibilities in cases where participants have multiple views of the virtual world (called 'peripheral lenses' to assist

in object-focused interaction in CVEs [44]), subjective views of the world (each participant is presented with a different version of the 3D world appropriate to their role [23]) or when participants change their own viewing parameters (such as field of view, clip distances, etc). In such cases, there is no direct link between the spatial interpretation of awareness and what the user sees. An explicit mapping between these would be necessary by the developer in a spatial trading approach to awareness. Conversely, tying the awareness of sources of visual data to the rendering module ensures flexibility and consistency between what the user sees and what the underlying system determines as the user's awareness.

6.2.4 Correlation with Quality of Service

In addition to filtering video streams that fall outside the awareness of a participant, an awareness management mechanism can be used to discern the Quality of Service (QoS) level appropriate for those video sources that lie within the awareness of that participant [40]. In other words, various levels of awareness between participants can be correlated with the various levels of quality of the video the participants would want to exchange. For example, Greenhalgh et al [40] divide the region of awareness into three levels based on the spatial position of participants, with the highest level of awareness attributed to face-to-face and lowest level of awareness to a user being in the periphery. Once again, our visibility-based approach to interest management provides a more accurate reflection of awareness by providing the application with the exact number of pixels a video object viewed in virtual 3D space occupies on the screen. From this information an application can determine the video quality most appropriate to that video object.

Consider the example when he visibility query from the computer graphics renderer

informs the application that a participant in full view occupies about 70000 pixels when rendered on the graphics device. The most appropriate resolution and quality for this participant would then be 320×240=76800 pixels per frame. Conversely, a participant further away in the view would only occupy about 20000 pixels on the screen, therefore requiring a maximum quality of 160×120=19200 pixels per frame. If a higher resolution video were to be transmitted for the latter video object, it would be a waste of bandwidth resources since no more than 20000 pixels will be drawn on the receiving participant's screen.

The visibility-based awareness management can therefore be used to more accurately determine the preferred quality of video transmitted in a 3D videoconferencing session.

6.2.5 Visibility-based Awareness and an ESM protocol

An important parameter of an awareness driven 3D videoconferencing system making use of the visibility-based awareness model is the field of view chosen for the computer graphics renderer. Not only the field of view affects the user experience by determining the span of space he/she can have in view but it also impacts the bandwidth requirements of the system. A large field of view allows many users to be simultaneously in view and results in a larger number of streams that must be transmitted/received. As discussed in section 4.6.4 the maximum total number of streams transmitted by all users is an important parameter for an ESM protocol. For instance, if each participant in a 6-person videoconference has a maximum fan-out bandwidth capable of transmitting two video streams, then the system must ensure that the maximum total number of streams requested by users does not exceed twelve (or the rejection rate of the routing algorithm will be high). Requests that bring the total to more than twelve would be rejected due to

the lack of bandwidth available to support them.

In order to predict the maximum number of video streams that will be requested, in our application we assign static seats for each user and only allow them to rotate their view (i.e. allow only changes in orientation and not position). For example, a 4-person 3D videoconference with users positioned equal distances apart around a circle with field of view of 70-90 degrees will have a maximum of two users simultaneously in view. Therefore, every user can only receive a maximum of two streams (out of the possible three) and request to transmit a maximum of three streams (out of the possible three). Overall, the maximum total number of sent/received streams is limited to eight (occurs when simultaneously all four users have two users in view).

6.3 *Proof of Concept Implementation*

Given the requirements and criteria for multi-party videoconferencing, the merits of a visibility-based awareness management in flexibly, efficiently and intuitively limiting the total number of video streams concurrently transmitted in a session and an ESM protocol designed for multi-party videoconferencing (MVEMP), we can now present a proof of concept implementation of such an application along with its awareness management scheme deployed over MVEMP. The overall software architecture is shown in Figure 50. The user interacts with the application through the traditional Graphical User Interface (GUI) components such as buttons and menus, through computer generated 3D graphics environment with video objects embedded therein as well as hardware and software devices used for capturing audio and video streams. The awareness management component queries the visibility of objects from the graphics hardware and informs the application of changes in awareness based on a set of rules defined by the awareness

policy. As mentioned before, our implementation of the awareness management mechanism makes use of recent feature of graphic cards that allows for the visibility query of objects in the 3D scene through API extension to OpenGL.

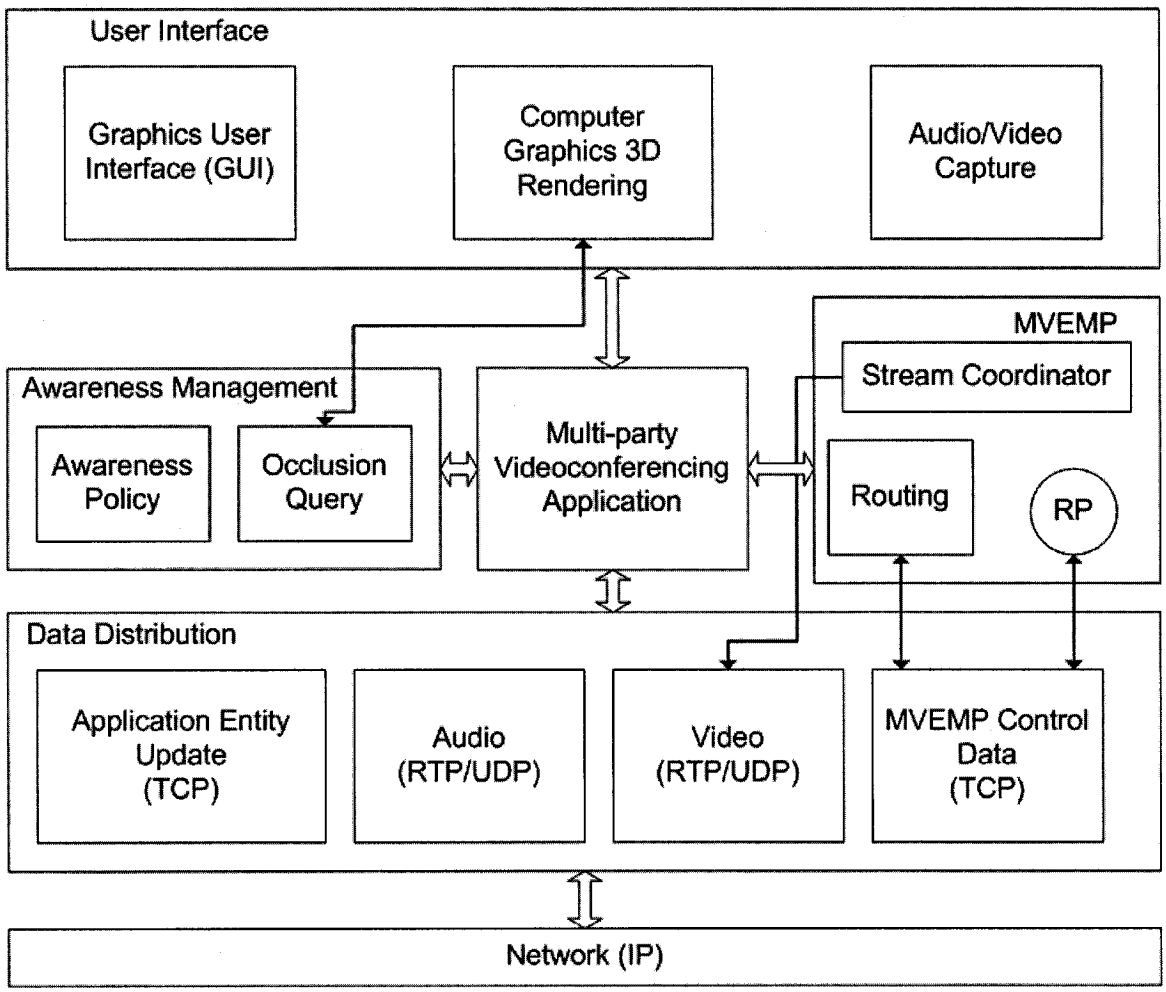


Figure 50. Overall software architecture and components

Each occlusion query returns the number of pixels the queried object occupies on the screen which is used by the application to subscribe or unsubscribe from a media source. MVEMP provides services for connecting conference participants to one another (through its RP), for participants to subscribe and unsubscribe from video streams, for determining the overlay multicast tree that must be constructed (through its routing heuristic) and coordinating the software video distribution components to perform the

transmission, reception and forwarding of video streams so as to build the overlay multicast tree determined by the routing heuristic. Note that only one participant acts as the RP and performs the routing heuristic. The others only execute the coordination function in assisting to build the overlay tree.

Entity state updates (e.g. 3D position changes of participants in the virtual conference room) and other application updates as well as the control messages of MVEMP are exchanged between participants in a peer-to-peer manner using the reliable TCP/IP protocol stack while audio and video are transmitted using RTP/UDP/IP protocol stack. Position updates, though frequent, are very small in size and produce an average bit-rate of less than 1kbps. Video is captured at 320×240 resolution and 15 frames per second and encoded using a (Java Media Framework) H.263 codec, resulting in an average bit-rate of 100kbps.

For audio we use Teamspeak [100], which sets up a server on one of the client's machine to act as a distribution point or 'reflector'. With silence suppression enabled, audio requires an average bit-rate of 5kbps. Although there is no technical limitation for using MVEMP and overlay multicast trees for the audio as well as video, we have not at this point implemented this feature.

With the exception of the visibility query portion of the awareness management component, which requires direct call to the graphic card's API (using the C-language), the rest of the implementation is in the Java language with Java3D [55] for 3D rendering and the Java Media Framework (JMF) [56] for capture, transmission and reception of video streams.

6.3.1 Visibility-based Awareness Implementation

The key part of the implementation for visibility-awareness uses the OpenGL extension *GL_NV_Occlusion_Query* [79] to retrieve the number of pixels drawn as a result of rendering an object. In our implementation, 3D objects are divided between those that represent a visual data source (planes textured with the video of remote participants) called *VisSourceObjects* and others (e.g. 3D room where the conference takes place and any additional artifacts such as table, chair etc.) called *NonVisSourceObjects*. A frame rendering cycle consists of the following steps:

1. Beginning of frame operations
2. Frustum culling of objects in the scene
3. For objects in the view frustum:
    - Draw *NonVisSourceObjects*
    - Begin query occlusion of *VisSourceObjects*
    - Draw object
    - End query occlusion of *VisSourceObjects*
4. Retrieve and store pixels drawn for each *VisSourceObject*
5. End of frame operations

The first step involves event checking and other operations before the rendering of the frame. The second step eliminates objects that are not in the view frustum (the space that will be visible to the user). For those in the view frustum, we draw *NonVisSourceObjects* first since these include the terrain, environment and artifacts in the scene that may occlude *VisSourceObjects*. We now use *glBeginOcclusionQueryNV()* method to make a query on any *VisSourceObject* in the view frustum, then draw this object by calling the

appropriate OpenGL draw methods and end the query by using *glEndOcclusionQueryNV()*. We can then retrieve and store the number of pixels that were drawn which in turn are used by an Awareness Management to determine visibility of *VisSourceObjects* according to a given policy. For example, in our implementation the policy is simple: if the number of pixels drawn is greater than a threshold (500 pixels), the object is considered visible. Otherwise, it is considered occluded or out of view.

Note that querying of occlusion need not be done for every frame. In fact, we perform occlusion query once every second since occlusion of whole objects does not change very rapidly. Once the visibility of *VisSourceObjects* is determined, the application subscribes to or leaves sources corresponding to visible or occluded *VisSourceObjects*. In other words, each user subscribes to the video of sources that are visible to him/her.

6.3.2 Coordination of Video Transmission and Reception by MVEMP

The use of MVEMP or any other ESM protocol for that matter necessitates the development of highly flexible and modular video distribution components. This is due to the fact that an application running on top of an ESM protocol must not only capture, transmit and receive video streams but must also implement a *forwarding* capability for each peer so as to allow a receiving peer to transmit it to other peers. In a multi-party setting where there are multiple sources present, the implementation must include functionality to manage the forwarding of any of its receiving streams. Figure 51 shows the approach taken in our approach.

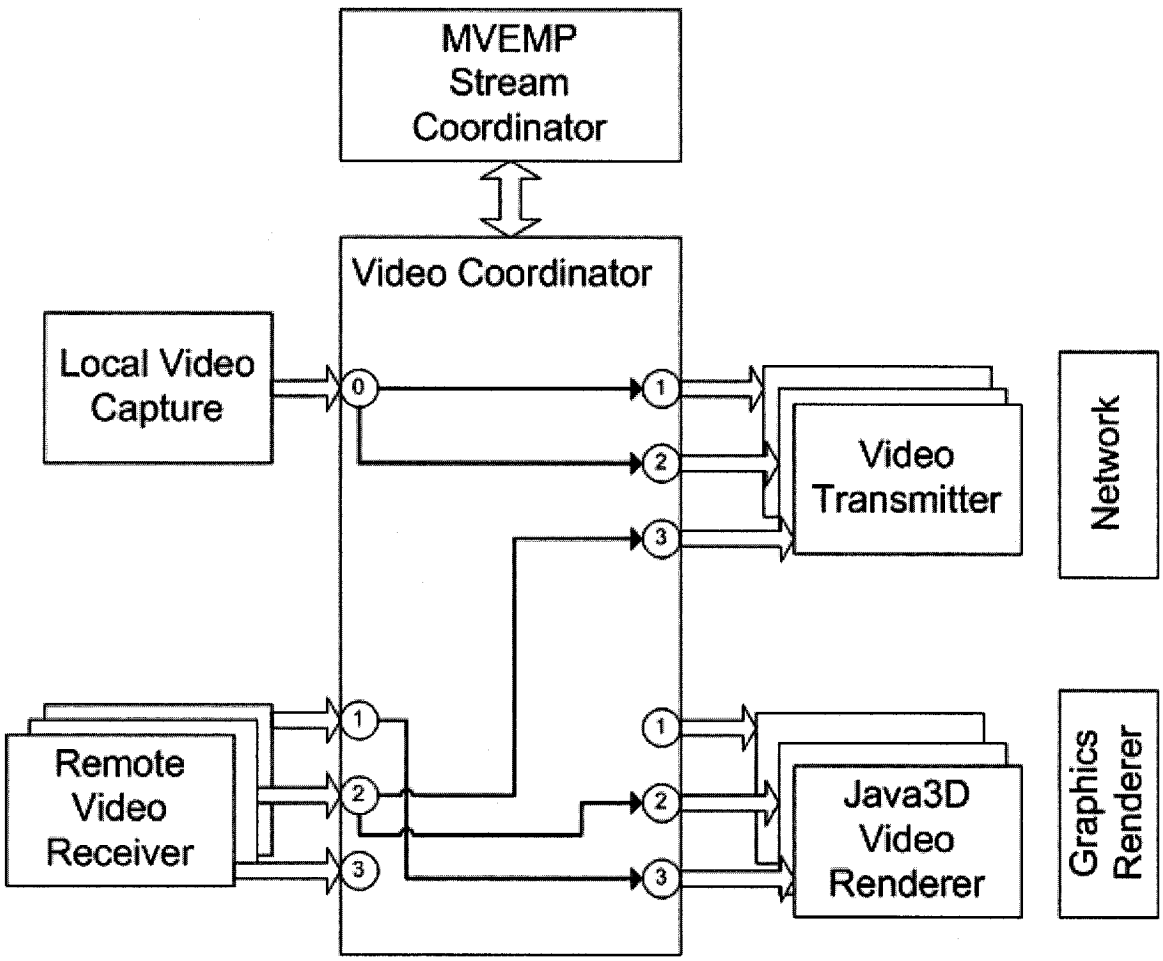


Figure 51 Video coordinator for flexible transmission, reception and forwarding of video streams

A 'Video Coordinator' running locally on each participant's machine acts as a switchboard between the local video capturing and receiving components and the local video rendering and transmission components. This implies that the 'Video Coordinator' can switch any of its inputs (from either the video capture or receiving components) to any of its outputs (to either the video rendering or transmission components). Furthermore, the switching is not necessarily a one-to-one mapping between input and outputs. The 'Video Coordinator' must also allow for one input to be connected to multiple outputs. Figure 51 illustrates one scenario. The captured video is transmitted to two outgoing transmission components (i.e. two other participants directly receive video

from this participant). The video received from input 2 is rendered by the corresponding video renderer (i.e. the video received from channel 2 corresponds to the participant number 2). Additionally, this input is also connected to outgoing transmission component 3, implying that this participant forwards the received video to participant 3. Finally, input 1 is connected to renderer component 3, implying that although the received stream comes through channel 1, the video corresponds to participant 3 (in other words, participant 1 is forwarding the video stream for participant 3). As can be seen, running a multi-party conferencing application over an application layer multicast protocol inevitably leads to higher complexity of application development since the application itself must be involved in the multicasting functionality.

As mentioned before, the video distribution part of the application is implemented in the Java Media Framework (JMF) [56]. Figure 52 illustrates the general class hierarchy of the video distribution part. Note that an instance of this hierarchy exists for each client, except the Rendezvous Point, which only exists on one machine. The 'Video Coordinator' logic illustrated above is implemented in a "*SwitchDataSource*" class. *SwitchDataSource* acts as a switchboard between the multiple sources of video and the multiple receivers or Data Sinks of video. Video can be streamed to every remote participant via the corresponding *DataSink* (an RTP transmitter in our case) and video is received from every remote participant via the corresponding *CloneableDataSource* (sources can be cloned for forwarding and multi-unicasting). An additional *CloneableDataSource* represents the locally captured video. Each *DataSink* is associated with a *SwitchDataSource* with the latter determining the source of the video transmitted by the *DataSink.*

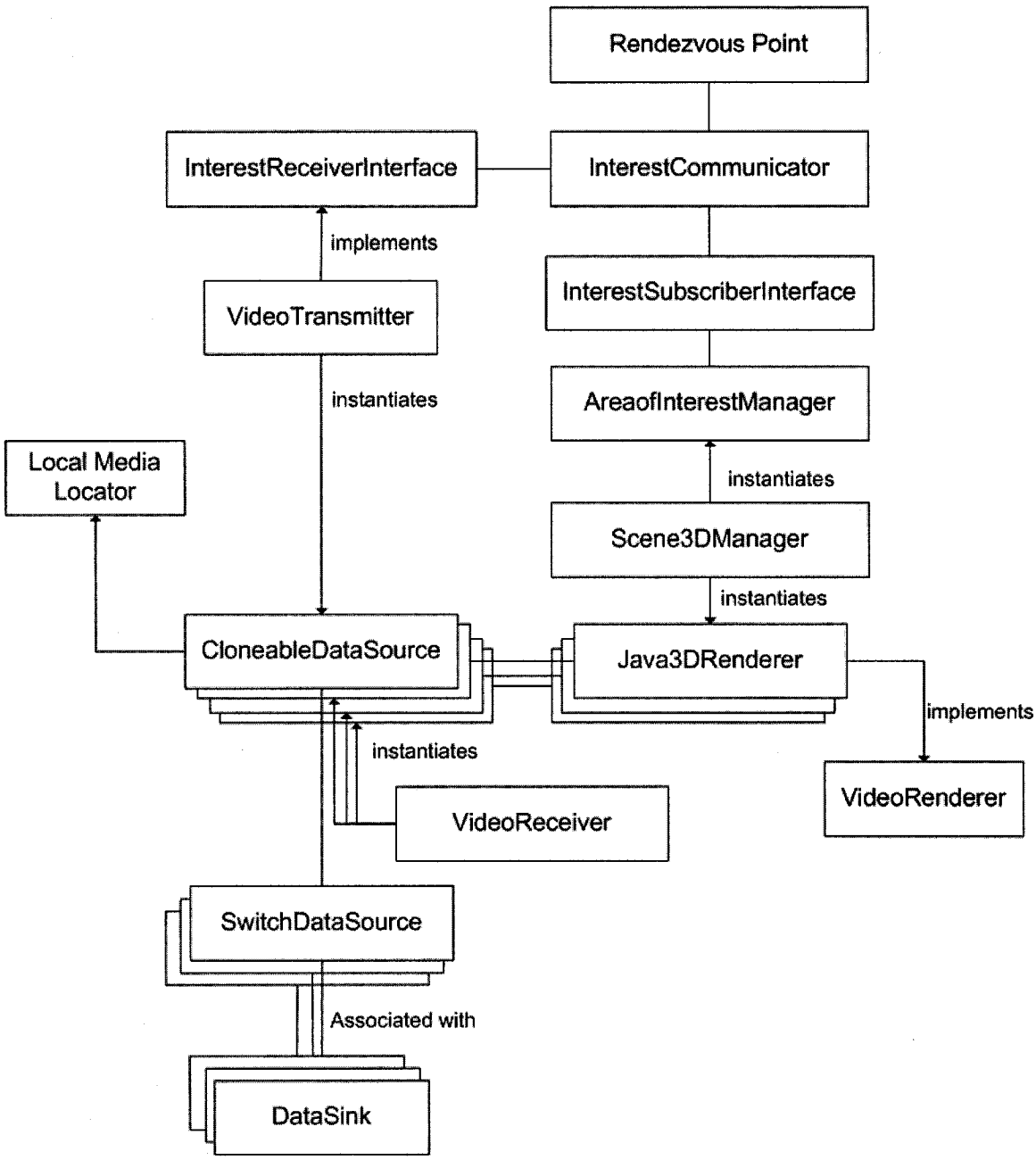


Figure 52. General class hierarchy of video distribution in proof of concept implementation

The *VideoTransmitter* controlling the transmission of all data sources implements the *InterestReceiverInterface* in order to receive directions as to the *DataSink* each source should be going to. The routing decisions are made by the Rendezvous Point based on the heuristic presented in previous chapters (currently we use MOTOR) and are

communicated to the concerned clients through an *InterestCommunicator*, which also acts a receiver of media subscription requests for the Rendezvous Point. Each client determines their interest in each video source through their *AreaofInterestManager* (based on a the visibility-base awareness management scheme described earlier) and requests subscriptions to specific sources through an *InterestSubscriberInterface*. Each data source is manifested as a textured plane in the 3D world through the *Java3DRenderer* class, whose position and orientation is controlled by the *Scene3DManager*.

The rest of the application functionality is built on COSMOS, an MPEG4 based distributed multimedia collaboration framework implemented in Java [26] [51]. It consists of loading VRML files for the 3D environment (room, table and chairs) and exchanging entity update messages (user orientation and movement in the virtual world) and rendering of the virtual world in addition to the video 'avatars' representing each user.

The received video is textured on the flat plane representing the corresponding user. Users can look around the room and at other participants by dragging the mouse over the interface. Orientation update messages are exchanged between users so every user sees the orientation of users in their view (the bit rate for orientation data is less than 1kbps). The rendering field of view of 70 degrees allows for a maximum of two users to be in view at the same time (to see the third participant, the user must drag the mouse over the interface in the direction of the third participant). Users are seated around a circle of radius 3 (virtual) meters with equal distance between each user. Depending on the number of users, the circle radius and the distance between each user is adjusted. Figure

53 shows the final user interface of the proof of concept application.

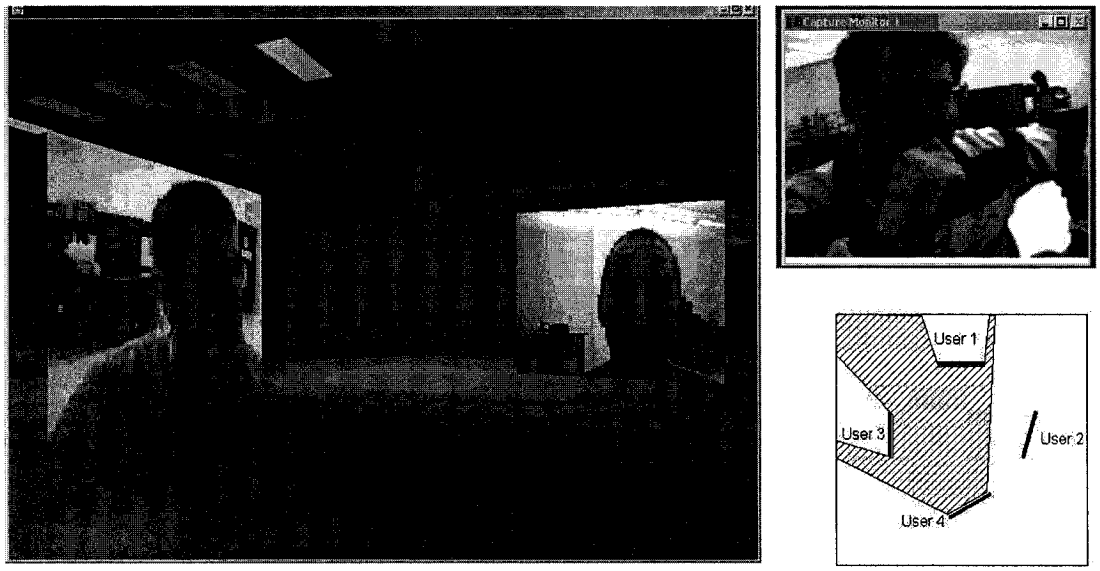


Figure 53. Application screenshot

6.4 Characteristics and Evaluation

In order get an idea of the characteristics of the system presented in the previous section, we conducted six short duration (4-6 min) teleconferences between 4 persons using our awareness-driven 3D videoconference system running on a Local Area Network (LAN) and recorded the number of video streams each user received and transmitted. Each user was represented by a flat square plane of side length 2, seated equally spaced around a virtual table of radius 3. The received video is textured on the flat plane representing the corresponding user. Figure 37 shows the average percentage of time a user received 0,1,2,3 video streams and transmitted 0,1,2,3 video streams.

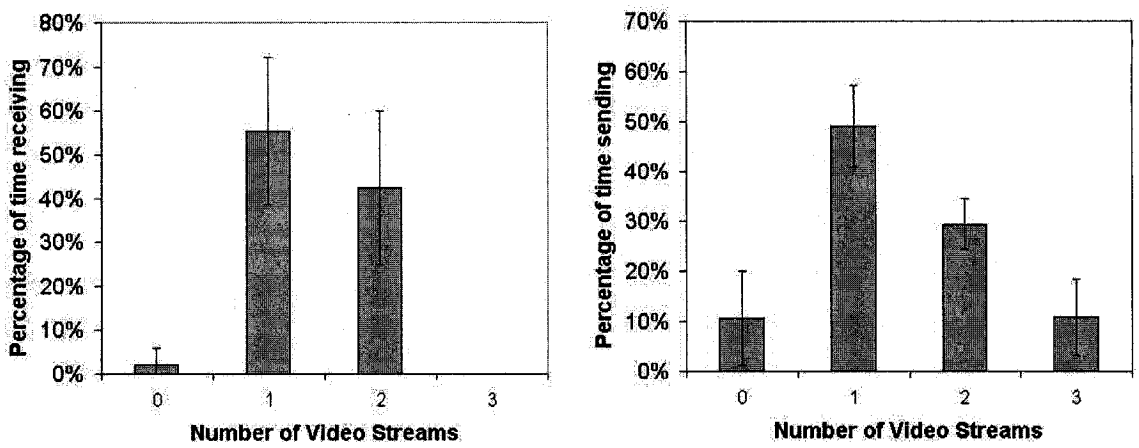


Figure 54. Average percentage of time receiving and transmitting 0,1,2,3 video streams

As expected, out of the possible three video streams, a user on average receives one or two since the field of view does not allow for three users to be simultaneously in view. However each user is still required to transmit three video streams when all three users have him/her in their view. Table 6 shows the average percentage of time N streams were transmitted (N= 0 to 12 for a 4-person teleconference) during the six teleconferences conducted. As expected, the maximum number of streams transmitted is 8 (compared to 12 if every user transmitted to every other user).

Table 6. Percentage of time system must deal with N streams for a 4-person 3D videoconference (field of view of 70 degrees)

N<3	N=3	N=4	N=5	N=6	N=7	N=8	N>8
1%	2%	11%	33%	30%	20%	3%	0%

These results merely confirm the characteristics that the particular application was designed to exhibit. What is of significance here is that we have succeeded in reducing the maximum total number of video streams transmitted while still allowing the user to choose the subset of other users whom he/she has in view. Furthermore, it is of significance that each user does not transmit the maximum possible number of streams *at*

all times. This is an important characteristic for allowing the application to run over an ESM protocol since users that are not transmitting at full capacity can be used to forward a video stream from a source that is strained to transmit to many users. Note also that the maximum load on the system (i.e. when all users have 2 other users in their view and the system must accommodate transmitting a total of 8 streams) occurs for only a small fraction of the entire duration of the conference (3% in Table 6). This is important since the routing heuristics have more difficulty finding solutions when the system is fully loaded (recall section 5.3.2). Note also that for about 10% of the time a user would be transmitting no video since no one had that user in view. This is important because the user being outside the awareness or focus of other users for that duration can assist in forwarding the video stream of another user that may be in focus of all other users and may hence require help in transmitting its stream in an ESM setting.

In order to illustrate the effect of utilizing an ESM protocol for a multi-party videoconferencing application, we replayed the six conferencing sessions already mentioned over MVEMP and assigned each of the four participants a maximum normalized outflow bandwidth of 2. This corresponds to participants having DSL connections with upload bandwidths of 200kbps and transmitting 100kbps video streams (320×240 at 15 fps). Figure 55 shows the average percentage of time a user transmits 0,1,2,3 streams in the absence of MVEMP and when using it.

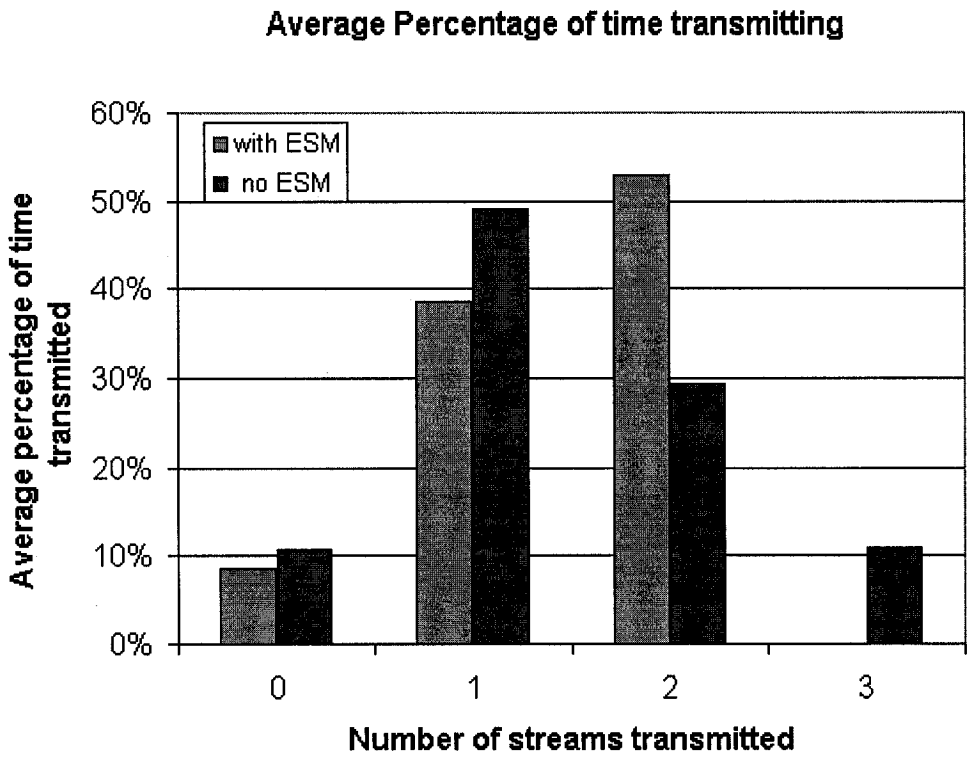


Figure 55. Number of video streams transmitted with and without MVEMP

Note that with MVEMP in place, no user exceeds its maximum fan-out but must take on the forwarding of streams for other sources at times as evident by the increase in the percentage of time transmitting 2 streams. These results illustrate the potential of MVEMP for carrying out multi-party videoconferences over the Internet without a need for IP Multicast capability since the protocol allows for different participants to define their maximum outflow bandwidth (based on their Internet connection and the video stream they will transmit) and will construct overlay trees for each source to distribute video to interested participants.

At this point it would also be of interest to know the percentage of times MVEMP's routing heuristic (MOTOR) presented in Chapter 4 was unable to find a solution. For this purpose we replayed the six (4-6 minute) teleconferencing sessions mentioned with different configuration of maximum out-degrees assigned to the 4 participants and

observed the success rate as well as the number of changes that occurred when accommodating a request.

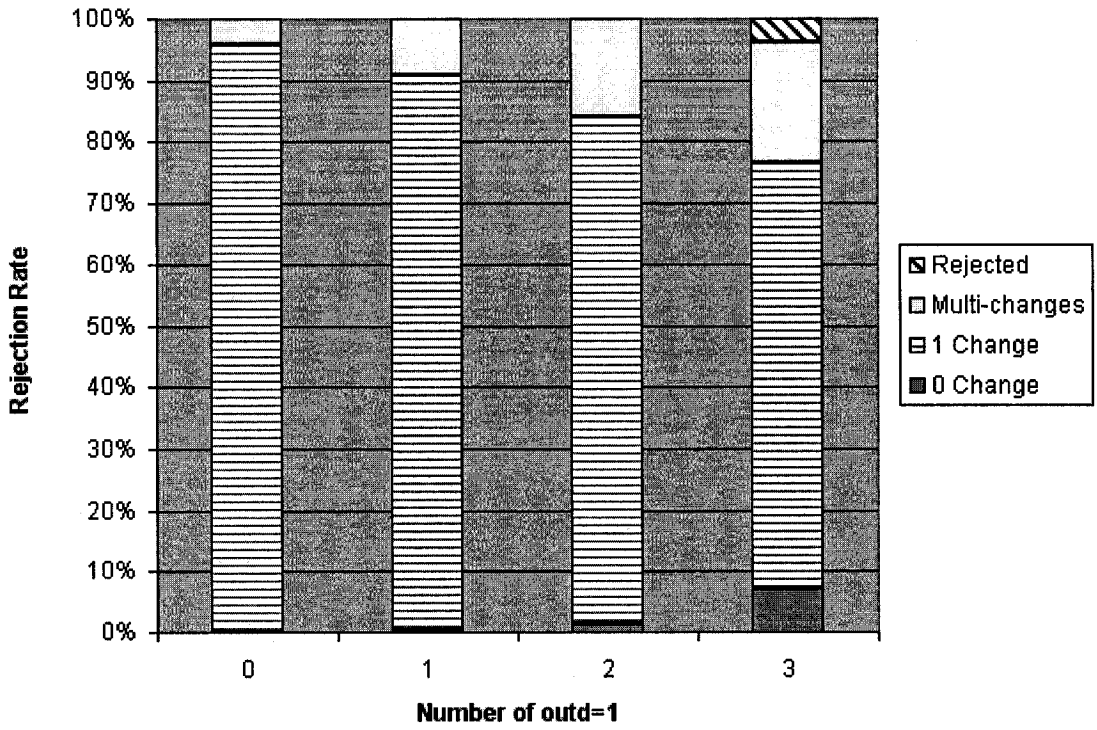


Figure 56. Percentage of requests satisfied by routing heuristics

Figure 56 shows the percentage of time a request was accommodated by a single change to the topology, by multiple changes and when no solution was found (the request was rejected) for the different configurations of out-degree maximums assigned. There were a total of 419 join requests (and 385 leave requests) during the six teleconference sessions. Note that we have grouped the different configurations of out-degrees between <1-5> according to how many users exist with out-degree of 1. For example the configurations (3,3,1,1) as well as (4,2,1,1) etc. are grouped together in Figure 56 as those having two nodes with out-degree of 2.

As the number of users with maximum out-degree of 1 increases, the probability that no solution is found increases from 0% to 4% (for all the permutations of 5,1,1,1

configurations). Furthermore, the probability that a single change in topology can accommodate a request decreases from 96% to 69% as the configuration includes more users with out-degree of 1.

Overall the high success rate of our routing algorithm, especially in the case of peers with out-degree maximums of more than one, suggest the potential of MVEMP in the context of multi-sender videoconferences and the potential of an awareness-driven 3D videoconferencing application to be able to take advantage of such a protocol in order to accommodate 4-10 users and be deployable over the Internet without a need for IP Multicast support from the infrastructure.

6.5 Summary

The purpose of this chapter was to present the implementation of MVEMP and its routing heuristic within the context of a proof of concept multi-party videoconferencing system. The system is based on a novel visibility-based awareness management mechanism that quickly, efficiently and accurately discerns awareness between any pair of users in a distributed way. The system makes use of the awareness information to filter out video streams that are outside the sphere of awareness of a user and therefore indirectly limits the total number of video streams the system must deal with. More importantly, the awareness management mechanism precipitates the situation where a user outside of the focus of the group has upload bandwidth available in order to aid another user to distribute his/her video. This in turns is used by the underlying ESM protocol designed specifically for multi-party systems to build overlay multicast trees for each source. Although there are many features and functionalities that are not implemented in the proof of concept application (e.g. priority, heterogeneous bit-rate, traffic and congestion

monitoring, etc), the chapter serves to illustrate the utility of ESM in bringing about new classes of application on today's Internet.

Chapter 7 Related Issues and Future Work

As an introduction to a brief survey of existing approaches to multi-party videoconferencing technology, we mentioned in section 2.1 that this class of multimedia application largely depends on video coding and compression technology, network data throughput available from the network and network support for multi-point communication and possibly Quality of Service. Through our experience of studying the requirements of this class of applications, designing, implementing and evaluating an ESM protocol targeting those requirements, and a proof of concept 3D multi-party videoconferencing application, we gained some insights into the broader issues relevant to their development.

7.1 Importance of Human Computer Interface

One of the most recurring themes during the design and implementation of both the MVEMP protocol and the proof of concept application was the crucial role that the human computer interface plays in determining not only the user experience but the requirements of the application from the underlying network and supporting protocols. As discussed in section 2.1, the video frame rate and resolution used in a remote conferencing session affects the ability of the participants' experience and ability to communicate and express non-verbal clues. Of course frame rate and resolution have a direct bearing on the network requirements by determining the bit-rate of the transmitted media. The acceptable inter and intra media delay and jitter are another set of parameters that affect the user's experience and have a direct relationship with how the media is transmitted through a network. More specific to multi-party videoconferencing applications, the number of remote participants that each user can see is another crucial

parameter tied to user requirements and affecting network protocol design. As mentioned in section 6.4, if every user is able to see every other user *at all times*, then it becomes infeasible to use an ESM protocol for that application. Instead, if users are constrained to see only the current speaker, then it becomes far easier to manage the multicasting of a single video stream but reduces the conferencing experience. In our multi-party videoconferencing system we tried to strike a balance between the two by allowing users to choose whom they see through the intuitive concept of orienting their view in a virtual space, but limited the number of remote participants they could have simultaneously in their view by assigning static seats in the virtual space, allowing only orientation movements and adjusting the field of view parameter. Limiting the number of received streams in this way allowed for the use of an ESM protocol in conjunction with the application and yet gave the users the flexibility to choose whom they see.

Finally, although a minor detail in the user interface, the way each user chooses the subset of remote participants to 'see' has a significant impact on the underlying ESM protocol. In our application, each user can change their view inside the virtual space by dragging the mouse in the direction they wish to view (similar to some 3D games). The speed of the movement however greatly affects the stability of the underlying overlay multicast trees. If users are given an interface whereby they can very quickly change between the remote participants they wish to see, then the underlying network protocol has difficulty in quickly responding to frequent join/leave events for each multicast session. The case is even more acute when faster methods of moving such as head or eye tracking is used instead of the mouse (such as in [104]).

In general, we can summarize that the choice of human computer interfaces for an

application often has direct implications on the network requirements and vice versa, the choice of network protocols and services greatly affects the user experience. This can be made more acute as networking functionality (such as multicasting) is pushed up towards the application layer and are tuned for specific applications.

7.2 Application Layer Multicast and IP Multicast

Application Layer Multicasting is a relatively new and exciting area of research, promising immediately deployable multi-point applications. By bringing multicasting functionality from the network layer to the application layer, ALM takes advantage of application-specific knowledge, is more easily maintainable and updateable and can rely on the larger storage space of application entities for caching and their processing ability for better traffic shaping and data transcoding [114]. ALM however suffers in its efficiency by producing duplicate packets on the same network link and increasing delay. Also by not having access to network layer information (such as routing tables), ALM can produce extra traffic in its effort to infer network topology and measurement of link costs through probing (ping) messages.

The increase in application development complexity as a result of using an ESM protocol is another factor to consider. IP Multicast abstracts the multi-point data distribution away from the application, whereas ESM requires applications to be active participants in the multicast data distribution. As was illustrated in section 6.3.2, the forwarding capability that must be supported by ESM nodes requires careful and modular design of the video reception/transmission software entities in the application. If the application is to support heterogeneous bit-rates and hence transcoding, the application is made all the more complex. This is an area where perhaps reusable software components can have a

significant impact.

Another issue related to the successful implementation of ALM is fault resilience [2] [69] and fault tolerance [115]. Although proxy-based ALM can assume longer lifetime for its nodes, both ESM and proxy-based ALM must pay special attention to the possibility of node failure within their overlay network and the ensuing disruption of data distribution. Fault tolerance and resilience are especially important to ALM since the number of nodes in the network is far smaller than in IP Multicast scenario. This is because ALM involves application nodes running a specific ALM protocol whereas in IP Multicast, every router in the network is potentially accessible so if a router fails, there are many alternative paths through other routers.

Finally it is important to dwell on the future of ALM and especially its existence within next generation networks that may ubiquitously support IP Multicast. Can ALM survive if IP Multicast receives widespread deployment? The answer is perhaps more tied to the business model of next generation networks than technical considerations. In the case where IP Multicast service translates into additional service charges for customers, ALM may still be prevalent for those that require a less efficient but free service. From a technical standpoint however, conceptually ALM is only a substitute for IP Multicast when IP Multicast is not available.

7.3 MVEMP and Network Issues

A complete protocol supporting multi-party videoconferencing goes far beyond session management and routing. It is important for the traffic generated by the protocol to behave well in relation to other protocols and processes. For example, greedily using the upload bandwidth of a user to forward RTP/UDP packets carrying video for another user

will starve TCP traffic of that user. It is therefore necessary to develop or adopta TCP fairness mechanism for the video medium. Furthermore, as discussed in section 3.3.6, the possibility of transmission of multiple streams from one node to another (e.g. a node transmitting its own video as well as forwarding another node's video to a particular user), brings about issues of congestion control and management as a result of multiple RTP/UDP streams within the same session. The incorporation of an integrated congestion management scheme such as [5] would additionally provide for efficient multiplexing of multiple flows, while ensuring proper and stable congestion behavior and fairness to TCP flows running in parallel.

As demonstrated in [22], it is also important and beneficial for an ESM protocol to monitor the received quality of media through either feedback or periodic measurements in order to adapt to the dynamic nature of Internet and fluctuations in delay and throughput bandwidth. In our setup for example, the back channel of RTP (called RTCP, Real Time Control Protocol) can provide periodic feedback on the received quality for each source (e.g. delay or received bit-rate, etc). Implementing a monitoring mechanism can greatly increase the flexibility and adaptability of a protocol.

Two of the most common metrics used by overlay routing mechanisms including the one presented in this thesis are end-to-end delay and available bandwidth. The routing algorithms operate based on the assumption that end-to-end delay and available bandwidth are provided for a given set of nodes in the overlay network. An estimation of end-to-end delay can be obtained by simply making use of 'ping' packets. Measuring available bandwidth between two nodes however is less trivial and the subject of much research: see [83] for a survey of current approach and [35] for a study of how basic

bandwidth measurement techniques may work in practice in peer-to-peer systems. Given the cost of more accurate bandwidth measurement techniques (both in terms of development and generated network traffic), many websites on the Internet opt for a coarse way of determining available bandwidth by asking the user to specify their network connection (dial-up, DSL, T1, etc) from which a very rough estimate of the available bandwidth can be inferred based on typical values for each network connection type. Subsequently simple throughput bandwidth monitoring schemes (such as providing feedback via RTCP) can aide the media source in determining the actual received bandwidth.

In the context of multi-party conferencing applications that give each user the flexibility and freedom to choose the set of sources they wish to receive from, the issue of frequent changes to the topology of overlay trees has already been alluded to. This can be addressed both at the interface level and the network protocol level. At the interface level for instance, a simple low-pass filter can eliminate transient changes in awareness or interest on the client side so as to minimize the number of changes that must be incurred by the overlay tree (at the cost of extra delay). As mentioned in section 4.3, dynamic approaches to multicasting are also invaluable in such circumstances in providing a way of connecting a new node to the tree without causing a great number of changes. The combination of the two methods is more likely to produce satisfactory results.

The six teleconference sessions mentioned in section 6.4 were all conducted on a Local Area Network with negligible delay between each user. The limited upload bandwidth of DSL connections was simulated by restricting the number of streams each user was allowed to transmit, but delay characteristics on the Wide Area Network (WAN) are sure

to have an impact on the user experience by prolonging the time it takes for a join request to materialize into packets transmitted from one node to another. In this context caching and pre-fetching techniques as well as node hand-off techniques such as those employed in [108] can help smooth the transition between the times a node requests to receive a stream to the time it starts to receive it.

7.4 MVEMP Applicability

Throughout this thesis we described MVEMP as a protocol appropriate for and targeting multi-party videoconferencing applications. The main requirements that MVEMP strives to address are those outlined in section 2.2, mainly: small group, bandwidth-constrained applications with multiple simultaneous sources such as multi-party videoconferencing. Other similar applications such as multi-player games and in general other Collaborative Virtual Environments (CVE) and even some Distributed Virtual Environments (DVE) can benefit from some of the main ideas behind MVEMP. DVEs for example have, and especially those that incorporate audio and video media, have long been challenged by their lack of scalability spurred by the slow deployment of IP Multicast. The peer-to-peer approach of data distribution that Application Layer Multicasting (and MVEMP) is based on, as well as combining the application filtering capability of DVEs with an End System Multicast protocol as we have done can help similar applications achieve the desired scalability. Direct applicability of MVEMP to such applications however remains a question, especially because different application would require different services from the ALM protocol. Consistency for instance is a crucial requirement for distributed games, as demonstrated in [33] and distributed simulations require delay bounds and much larger scalability requirements [76].

The scalability of MVEMP to support medium sized groups with tens of users primarily depends on the application interface. As mentioned already, the total number of streams that the system must deal with should be a function of the total bandwidth available to all users and controlled by the application interface by controlling the number of remote participants each user can have in his/her view. In the performance evaluation chapter of this thesis we demonstrated that as the number of nodes increases without an increase in the average outflow available per node, it become increasingly difficult to find solutions to the overlay routing problem and as a result, the rejection rate of the routing algorithm increases. To remedy this problem, an application with medium sized groups making use of MVEMP or any other ESM protocol would have to employ effective application-layer filtering in order to limit the total number of streams that the system must deal with (or alternatively reduce the quality of each stream). As the number of participants increases, it becomes less necessary for every participant to have the option of seeing every other participant. Therefore more limiting application-layer filters can reduce the number of potential sources so as to not overwhelm the available bandwidth of the overlay multicast nodes.

In section 4.6.1 the issue of trust and cooperation between peers in a peer-to-peer system was alluded to. The fundamental idea of ALM involves peers taking on the responsibility of forwarding for other peers in the overlay network. This however requires that peers cooperate with one another and be willing to share bandwidth and processing resources. For example, a node acting as a reflector in our system takes on the responsibility of receiving and distributing a stream to other nodes, yet itself is not interested in receiving that stream. The question that arises is: why should a node provide such a service to other

nodes, albeit temporarily? In the context of a small-group videoconferencing application, it may be argued that the participants coming together for the conference would inherently wish for the conference to successfully take place and would therefore be willing to share resources towards that end. In the larger context of peer-to-peer systems where peers may not be familiar with other peers or may not care to share resources for others' benefit, it may be important to provide incentives for doing so. In [64] the authors propose a mechanism by which degree of cooperation is inferred between peers, which could possibly be used to reward more cooperative peers (e.g. giving them priority of delivery, etc). The idea of cooperation and trust between peers in an application-layer multicast session however lends itself also to exploitation by users [74]. Once again, in small-group videoconferencing it may be more reasonable to assume a minimum degree of cooperation between peers, that may not exist in larger systems.

Chapter 8 Conclusions

New technologies and the applications they support typically form a 'circle of usage'. A positive usage implies that the new technology fosters new applications that are possible due to the availability of the new technology and consequently, the new technology gets utilized and further improved as applications that use it emerge and become widespread, which in turn leads to its subsequent success. A negative usage implies the improper or inadequate deployment of a new technology, leading to the improbability or infeasibility of use of its applications, which in turn eliminates the applications that produce the need for that new technology. IP Multicast seems to have partially suffered from the latter. Its lack of widespread deployment meant that multi-point applications could not be deployed easily, and with a lack of multi-point applications used by the common Internet user, both

Internet providers and network administrators saw still less reasons to deploy IP Multicast, further prolonging the emergence of multi-point applications on the Internet.

The recent popularity of peer-to-peer file transfer applications has precipitated new paradigms, challenges and solutions in many areas of technology as well as industry and business. Arguably, one of the fallouts of the emergence of peer-to-peer applications has been the idea of pushing multicasting capability from the network layer to the application layer to form Application Layer Multicast, where application peers can assist in delivery of content from one peer to multiple other peers. This in turn has spurred the development of a myriad of multicasting protocols at the application layer with some direct and indirect spin-off commercial products. It will be interesting to see the evolution of this new technology and its contribution to bringing about the next generation of Internet applications. The fact that some popular file transfer applications such as BitTorrent have already incorporated Application Layer Multicasting ideas in their implementation is promising.

In this regard, we examined the need for multicasting for multi-party videoconferencing applications and strived to bring together the various technologies that could make an Internet-deployable multi-party videoconferencing application possible. After examining some of the user requirements from such applications and their typical characteristics, we designed, implemented and evaluated an ALM protocol to address those requirements and characteristics and showed that the development of new routing heuristics is required in order to address the multi-source case. Towards that end, we examined existing approaches to overlay routing and identified areas of improvement in the multi-source scenario. The culmination of that investigation led to two new routing heuristics that

perform significantly better than similar routing heuristics with regards to the metrics commonly used to evaluate them. A proof of concept multi-party videoconferencing application was also designed, implemented and evaluated on the Local Area Network that could be deployed over our ALM protocol. As mentioned in the previous chapter, a much more complete implementation would be necessary in order for the application and the protocol to be easily used on the Wide Area Network but the proof of concept implementation served to highlight the potential of ALM and the relevant issues involved in designing and developing them.

It is unlikely that a single Application Layer Multicast protocol will satisfy all classes of multi-point applications. Instead, we conjecture that different ALM protocols will coexist, each targeting a different set of applications. Just as our ESM protocol is designed for multi-sender teleconferencing applications involving high-bandwidth flows, another protocol may target applications with low-bandwidth flows but requiring reliability and/or quality of service adaptation.

The design and implementation of an ALM protocol targeting a class of applications that hither had been neglected helped highlight some of the unique requirements of multi-party videoconferencing applications and in the larger context collaborative multimedia applications and Collaborative Virtual Environments. We hope that doing so will foster similar efforts from other researchers, whose collective endeavors will bring about new and exciting multi-point applications that have so far eluded the common Internet user.

References

[1] Aharoni, E., Cohen, R. "Restricted Dynamic Steiner Trees for Scalable Multicast in Datagram Networks", *IEEE/ACM Transactions on Networking, Volume 6, No. 3*, pp. 286-297, June 1998.

[2] Andersen, D., Balakrishnan, H., Kaashoek, F., Morris, R., "Resilient Overlay Networks", Proceedings of 2001 ACM Symposium on Operating System Principles (SOSP'01), pp. 131-145.

[3] ATI Occlusion Query: http://www.ati.com/developer/samples/dx9/OcclusionQuery.html

[4] Baker, H.H., Bhatti, N., Tanguay, D., Sobel, I., Gelb, D., Goss, M.E., MacCormick, J., Yuasa, K., Culbertson, W.B., & Malzbender, T. (2003). Computation and Performance Issues in Coliseum, An Immersive Videoconferencing System. *Proceedings of 2003 ACM conference on Multimedia (MM'03)*, 470-479

[5] Balakrishnan, H., Rahul, H.S., Seshan, S., "An Integrated Congestion Management Architecture for Internet Hosts", *Proceedings of 1999 ACM conference on applications, technologies, architectures, and protocols for computer communications (SIGCOMM'99)*, 175-187.

[6] Banerjee, S., Bhattacharjee, B., Kommareddy, C., "Scalable Application Layer Multicast", *Proceedings of 2002 ACM conference on applications, technologies, architectures, and protocols for computer communications (SIGCOMM'02)*, pp. 205-217

[7] Banerjee, S., Kommareddy, C., Kar, K., Bhattacharjee, B., Khuller, S., "Construction of an Efficient Overlay Multicast Infrastructure for Real-time Applications", *Proceedings of 2003 IEEE conference on Computer Communication (INFOCOM'03), volume 2*, pp. 1521-1531

[8] Bartz, D., Meibner, M., Huttner, T., "OpengGL assisted occlusion culling for large polygonal models", *Computer and Graphics, 23 (3)*, pp. 667-679, 1999.

[9] Bauer, F., Varma, A., "Degree-Constrained Multicasting in Point-to-Point Networks", *Proceedings of 1995 IEEE conference on Computer Communication (INFOCOM'95)*, pp. 369-376

[10] Buxton, W., Sellen, A., Sheasby, M., "Interfaces for Multiparty Videoconferences" *Video-Mediated Communication (edited by K. Finn, A. Sellen and S. Wilbur)*, Lawrence Erlbaum Associates, pages 385-400, 1997

[11] Castro, M., Druschel, P., Kermarrec, A.M., Rowstron, A., "Scribe: A large-scale and decentralized application-level multicast infrastructure", *IEEE Journal on Selected Areas in Communication (JSAC), 20(8), 2002*, pp. 1489-1499

[12] Castro, M., Jones, M., Kermarrec, A., Rowstron, A., Theimer, M., Wang, H., Wolman, A., "An Evaluation of Scalable Application-level Multicast Built Using Peer-to-peer overlays", *Proceedings of 2003 IEEE conference on Computer Communication (INFOCOM'03), volume 2*, 1510-1520

[13] Castro, M., Druschel, P., Kermarrec, A., Nandi, A., Rowstron, A., Singh, A., "SplitStream: High-Bandwidth Multicast in Cooperative Environments", *Proceedings of 2003 ACM Symposium on Operating System Principles (SOSP'03)*, pp. 298-313

[14] Cayley, A. "Theorem on Trees", *Quarterly Journal of Mathematics, Volume 23*, pp. 376-378, 1889.

[15] Chawathe, Y., "Scattercast: An Architecture for Internet Broadcast Distribution as an Infrastructure Service", *Ph.D. Thesis, University of California, Berkeley*, Dec. 2000

[16] Chawathe, Y., McCanne, S., Brewer, E.A., "RMX: Reliable Multicast for Heterogeneous Networks", *Proceedings of 2000 IEEE conference on Computer Communication (INFOCOM'00), volume 2*, pp. 795-804

[17] Chawathe, Y., Ratnasamy, S., Breslau, L., Lanham, N., Shenker, S., "Making Gnutella-like P2P Systems Scalable", *Proceedings of 2003 ACM conference on applications, technologies, architectures, and protocols for computer communications (SIGCOMM'03)*, pp.407-418

[18] Chen, M., "Design of a Virtual Auditorium", *Proceedings of 2001 ACM conference on Multimedia (MM'01)*, pp. 19-28

[19] Chen, M., "Achieving Effective Floor Control with a Low-Bandwidth Gesture-Sensitive Videoconferencing System", *Proceedings of 2002 ACM conference on Multimedia (MM'02)*, pp. 476-483

[20] Chen, S., Gunluk, O., Yener, B., "The Multicast Packing Problem,", *IEEE/ACM Transactions on Networking, volume 8, Issue 3,* pp. 311-318, 2000

[21] Chu, Y., Rao, S. G., Zhang, H., "A Case for End System Multicast", *Proceedings of 2000 ACM SIGMETRICS*, pp. 1-12

[22] Chu, Y., Rao, S.G., Seshan, S. Zhang, H., "A Case for End System Multicast", *IEEE Journal on Selected Areas in Communication, special issue on networking support for multicast*, 2002, Volume 20, Issue 8, pp. 1456-1471

[23] Churchill, E. F., Snowdon, D. N., Munro, A. J. "Collaborative Virtual Environments: digital places and spaces for interaction". *Springer-Verlog*, 2001, pg. 143-159

[24] Cormen, T.H., Leiserson, C.E., Rivest, R.L., "Introduction to Algorithms", MIT Press, McGraw-Hill Book Company, 1993

[25] CUWorld Videoconferencing Software, http://www.cuworld.com/

[26] Darlagiannis, V., Ackermann, R., El-Saddik, A., Georganas, N.D., Steinmetz, R. "Suitability of Java for Virtual Collaboration", *Proceedings of 2000 conference on Net.Object Days*, Erfurt, Germany, Oct. 2000.

[27] Deering, S., Cheriton, D., "Multicast Routing in Datagram Internetworks and Extended LANS." *ACM Transactions on Computer Systems 8(2)*, pp. 85-111, May 1990.

[28] Deshpande, H., Bawa, M., Garcia-Molina, H., "Streaming Live Media over a Peer-to-Peer Network", *Stanford CS department technical report.* 2001-31

[29] de Oliveira, J.C., Hosseini, M., Shirmohammady, S., Malric, F., Nourian, S., El Saddik, A., Georganas, N.D., "Java Multimedia Telecollaboration", *IEEE Multimedia, volume 10, number 3, July-September 2003*, pp. 18-29

[30] Diot, C., Levine, B.N., Lyles, B., Kassem, H., Balensiefen, D., "Deployment issues for the IP multicast service and architecture". *IEEE Network Magazine 2000, 14(1)*, 78-88.

[31] Douglas, R., "NP-completeness and degree restricted spanning trees", *Discrete Mathematics*, Volume 105, pp. 41-47, 1992.

[32] Edwards, B. M., Giuliano, L. A., Wright, B. R., Interdomain Multicast Routing: Practical Juniper Networks and Cisco Systems Solutions. *Pearson Education Inc.*, 2002

[33] El Saddik, A., Dufour, A., "Peer-to-Peer Suitability for Collaborative Multiplayer Games", *Proceedings of 2003 IEEE symposium on Distributed Simulation and Real-Time applications (DS-RT'03)*, pp. 101-107.

[34] Eriksson, H. "MBONE: The Multicast Backbone", *Communications of the ACM, 37(8), August 1994*, pp. 54-60

[35] Eugene Ng, T.S., Chu, Y., Rao, S.G., Sripanidkulchai, K., Zhang, H., "Measurement-Based Optimization Techniques for Bandwidth-Demanding Peer-to-Peer Systems", *Proceedings of 2003 IEEE conference on Computer Communication (INFOCOM'03)*, volume 3, pp. 2199-2209.

[36] Even, S. Algorithmic Combinatorics, *The Macmillan Company, New York*, 1973, pp. 103-106.

[37] Francis, P., "Yoid: Extending the Multicast Internet Architecture", 1999, http://www.aciri.org/yoid

[38] Ganesan, P., Sun, Q., Garcia-Molina, H., "YAPPERS: A Peer-to-Peer Lookup Service Over Arbitrary Topology", *Proceedings of 2003 IEEE conference on Computer Communication (INFOCOM'03)*, volume 2, pp. 1250-1260

[39] Greenhalgh, C., Benford, S., "Massive: a collaborative virtual environment for teleconferencing", *ACM transactions on Computer Human Interactions 2(3)*, pp. 239-261, September 1995

[40] Greenhalgh, C., Benford, S., Reynard, G., "A QoS Architecture for Collaborative Virtual Environments", *Proceedings of 1999 ACM conference on Multimedia (MM'99)*, pp 121-130

[41] Gummadi, K., Gummadi, R., Gribble, S., Ratnasamy, S., Shenker, S., Stoica, I., "The Impact of DHT Routing Geometry on Resilience and Proximity", *Proceedings of 2003 ACM conference on applications, technologies, architectures, and protocols for computer communications (SIGCOMM 2003),* pp. 381-394

[42] Hefeeda, M., Habib, A., Botev, B., Xu, D., Bhargava, B., "PROMISE: Peer-to-Peer Media Streaming Using CollectCast", *Proceedings of 2003 ACM conference on multimedia (MM'03),* pp. 45-54

[43] Han, J., Smith, B., "CU-SeeMe VR Immersive Desktop Teleconferencing", *Proceeding of 1996 ACM conference on Multimedia (MM'96)*, pp. 199-207.

[44] Hindmarsh, J., Fraser, M., Heath, C., Benford, S., Greenhalgh, C., "Object-focused Interaction in Collaborative Virtual Environments", *ACM Transactions on Computer-Human Interaction (TOCHI) Special Issue on Collaborative Virtual Environments*, Vol. 7, No. 4, December 2000, ACM Press, pp. 477-509

[45] Hosseini, M., Georganas, N.D., "Application Layer Multicast Protocol for Collaborative Virtual Environments", *Presence: Teleoperators and Virtual Environments, Volume 13, Issue 6, June 2004*, pp. 263-278.

[46] Hosseini, M., Georganas, N.D., "End System Multicast Routing for Small-Group Multi-Party Videoconferencing Applications", *submitted to for review.*

[47] Hosseini, M., Georganas, N.D., "Design of a Multi-sender 3D Videoconferencing Application over an End System Multicast Protocol", *Proceedings of 2003 ACM conference on Multimedia (MM'03)*, pp. 480-489.

[48] Hosseini, M., Georganas, N.D., "User Behavior in 3D Videoconferencing Applications", *Proceedings of 2003 Summer Computer Simulation Conference (SCSC'03)*, pp. 755-760.

[49] Hosseini, M., Malric, F., Georganas, N.D. "A Haptic Virtual Environment for Industrial Training", *Proceedings of IEEE International Workshop on Haptic Audio Visual Environments and their Applications*, Ottawa, Canada, November 2002, pp. 25-30.

[50] Hosseini, M., Pettifer, S., Georganas, N.D., "Visibility-based Interest Management in Collaborative Virtual Environments", *Proceedings of 2002 ACM Conference on Collaborative Virtual Environments (CVE'02),* Bonn, Germany, October 2002, pp. 143-144

[51] Hosseini, M., Georganas, N.D., "Suitability of MPEG4's BIFS for Development of Collaborative Virtual Environments", *Proceedings of 2001 IEEE International Workshop on Enabling Technologies for Collaborative Enterprises (WET ICE'01),* June 2001, Cambridge, MA, USA

[52] Isaacs, E.A., Morris, T., Rodriguez, K., "A Forum for Supporting Interactive Presentations to Distributed Audiences", *Proceedings of 1994 ACM conference on computer supported cooperative work (CSCW'94),* pp. 405-416

[53] Jancke, G., Grudin, J., Gupta, A., "Presenting to Local and Remote Audiences: Design and Use of the TELEP System", *Proceedings of 2000 ACM conference on computer-human interfaces (CHI'00),* pp. 384-391.

[54] Jannotti, J., Gifford, D., Johnson, K., Kaashoek, M., O'Toole, J., "Overcast: Reliable Multicasting with an Overlay Network", *Proceedings of Symposium on Operating Systems Design and Implementation, October 2000*, pp. 197-212

[55] Java3D API: http://java.sun.com/products/java-media/3D/

[56] Java Media Framework API: http://java.sun.com/products/java-media/jmf/index.jsp

[57] Konemann, J., Ravi, R., "A matter of degree: improved approximation algorithms for degree-bounded minimum spanning trees", *SIAM journal of computing, 31(6),* pp. 1783-1793, 2002

[58] Konemann, J., Ravi, R., "Primal-dual meets local search: Approximating MST's with non-uniform degree bounds", *Proceedings of 2003 ACM Symposium on Theory of Computing (STOC 2003),* pp. 389-395.

[59] Kosiur, D., "IP Multicasting: The Complete Guide to Interactive Corporate Networks", *John Wiley & Sons Inc, 1998*

[60] Kostic, D., Rodriguez, A., Albrecht, J., Vahdat, A., "Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh", *Proceedings of 2003 ACM Symposium on Operating System Principles (SOSP'03)*, pp. 282-297.

[61] Krishnamoorthy, M., Ernst, A.T., Sharaiha, Y.M., "Comparison of Algorithms for the Degree Constrained Minimum Spanning Tree", *Journal of Heuristics 7(6), Kluwer Academic Publishers*, November 2001, pp. 587-611.

[62] Kwon, M., Fahmy, S., "Topology-Aware Overlay Networks for Group Communication", *Proceedings of 2002 ACM International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'02)*, pp. 127-136

[63] Lawrence Berkeley National Laboratory Mbone tools. http://www-nrg.ee.lbl.gov/nrg.html

[64] Lee, S., Sherwood, R., Bhattacharjee, B., "Cooperative Peer Groups in NICE", *Proceedings of 2003 IEEE conference on Computer Communication (INFOCOM'03), volume 2*, pp. 1272-1282

[65] Li, G., Ruskey, F., "The advantages of forward thinking in generating rooted and free trees", *Proceedings of 1999 ACM-SIAM Symposium on Discrete Algorithms (SODA'99)*, p. 939-940.

[66] Liebeherr, J., Nahas, M., Si, W., "Application-layer Multicasting with Delaunay Triangulation Overlays", *IEEE Journal on Selected Areas in Communication (JSAC), 20(8), 2002*, pp. 1472-1488

[67] Liu, J., Bo, L., Zhang, Y.Q. (2003). Adaptive Video Multicast over the Internet. *IEEE Multimedia 10(1)*, pp. 22-33

[68] Liu, Z., Malouch, N., Misra, V., Rubenstein, D., Sahu, S., "Bandwidth-Sharing Schemes for Multiple Multi-Party Sessions", *Proceedings of 18th International Teletraffic Congress, 2003.*

[69] Loguinov, D., Kumar, A., Rai, V., Ganesh, S., "Graph-Theoretic Analysis of Structured Peer-to-Peer Systems: Routing Distances and Fault Resilience", *Proceedings of 2003 ACM conference on applications, technologies, architectures, and protocols for computer communications (SIGCOMM'03)*, pp. 395-406

[70] Loh, K.J., Chua, K.C., "Experience with Implementation of Multi-Party Video-Conferencing Application over Packet Networks", *Proceedings of IEEE GLOBECOM 1996 Communications Theory mini-conference*, pp. 53-57

[71] Macedonia, M.R., Brutzman, D.P., "Mbone Provides Audio and Video Across the Internet", *IEEE Computer, Volume 27, Issue 4, April 1994*, pp. 30-36.

[72] Malouch, N.M., Liu, Z., Rubenstein, D., Sahu, S. "A Graph Theoretic Approach to Bounding Delay in Proxy-Assisted, End-System Multicast", *Proceedings of 10th International Workshop on Quality of Service (IWQoS)*, 2002, pp.

[73] Mathy, L., Canonico, R., Hutchison, D., "An Overlay Tree Building Control Protocol," *Proceedings of 3rd International Workshop on Networked Group Communication, (NGC'01) London, U.K., Nov. 2001*, pp. 78-87

[74] Mathy, L., Blundell, N., Roca, V., El-Sayed, A., "On Cheats in Application-Level Multicast", *Proceedings of 2004 IEEE conference on Computer Communication (INFOCOM'04)*, to appear.

[75] Microsoft NetMeeting Software: http://www.microsoft.com/windows/NetMeeting/Features/Conferencing/default.ASP

[76] Moen, D.M., Pullen, J.M., "Enabling Real-Time Distributed Virtual Simulation over the Internet Using Host-based Overlay Multicast", *Proceedings of 2003 IEEE symposium on Distributed Simulation and Real-Time applications (DS-RT'03)*, pp. 30-36.

[77] MSN Messenger Service: http://messenger.msn.com/Feature/Communicate.aspx

[78] Multicast ISP List: www.multicast-isp-list.com/index2002.html

[79] NVIDIA Occlusion Query: http://www.nvidia.com/dev_content/nvopenglspecs/GL_NV_occlusion_query.txt

[80] Quax, P., Jehaes, T., Jorissen, P., Lamotte, W., "A Multi-User Framework Supporting Video-based Avatars", *Proceedings of 2003 ACM conference on networked games (NetGames'03)*, pp. 137-147.

[81] Padmanabhan, V.N., Wang, H.J., Chou, P.A., Sripanidkulchai, K., "Distributing Streaming Media Content Using Cooperative Networking", *Proceedings of 2002 ACM International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'02)*, pp. 177-186.

[82] Pendarakis, D., Shi, S., Verma, D., Waldvogel, M., "ALMI: An Application level Multicast Infrastructure", *Proceedings of 3rd Usenix Symposium on Internet Technologies and Systems, March 2001*, pp. 49-60

[83] Prasad, R., Dovrolis, C., Murray, M., Claffy, K., "Bandwidth Estimation: Metrics, Measurement Techniques, and Tools", *IEEE Network 17(6)*, December 2003, pp. 27-35.

[84] Prince, S., Cheok, A. D., Farbiz, F., Williamson, T., Johnson, N., Billinghurst, M., Kato, H., "3-D Live: Real Time Interaction for Mixed Reality", *Proceedings of 2002 ACM conference on Computer Supported Cooperative Work (CSCW'02)*, pp. 364-371.

[85] Ravi, R., Marathe, M.V., Ravi, S.S., Rosenkrantz, D.J., Hunt, H.B., "Approximation Algorithms for Degree-Constrained Minimum Cost Network Design Problems", *Algorithmica, 31:1*, pp. 58-78, 2001

[86] Ratnasamy, S., Francis, P., Handley, M., Karp, R., "A Scalable Content-Addressable Network", *Proceedings of 2001 ACM conference on applications, technologies, architectures, and protocols for computer communications (SIGCOMM'01)*, pp. 161-172

[87] Ratnasamy, S., Handley, M., Karp, R., Shenker, S., "Application-level Multicast using Content-Addressable Networks", *Proceedings of 2001 International Workshop on Networked Group Communication*, pp. 14-29

[88] Regenbrecht, H., Ott, C., Wagner, M., Lum, T., Kohler, P., Wilke, W., Mueller, E., "An Augmented Virtuality Approach to 3D Videoconferencing", *Proceedings of 2003 IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR'03)*, pp. 290-291.

[89] Reynard, G., Benford, S., Greenhalgh, C., "Awareness Driven Video Quality of Service in Collaborative Virtual Environments", *Proceedings of 1998 ACM conference on Human Factors in Computer Systems (CHI'98)*, pp. 464-471.

[90] Roca, V., El-sayed, A., "A host-based Multicast (hbm) Solution for Group Communications", *Proceedings of 2001 IEEE International Conference on Networking (ICN'01), July 2001*, pp. 610-619

[91] Rowstron, A., Druschel, P., "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems", *International Conference on Distributed Systems Platforms*, 2001

[92] Sellen, A., "Remote Conversations: The Effects of Mediating Talk with technology" *Human Computer Interaction*, pp. 401-444, 1995.

[93] Sherwood, R., Braud, R., Bhatacharjee, B., "Slurpie: A Cooperative Bulk Data Transfer Protocol", *Proceedings of IEEE conference on Computer Communication (INFOCOM'04)*, to appear

[94] Shi, S.Y., Turner, J.S., Waldvogel, M., "Dimensioning Server Access Bandwidth and Multicast Routing in Overlay Networks", *Proceedings of 2001 ACM International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'01)*, pp. 83-91.

[95] Shi, S.Y., Turner, J.S., "Routing in Overlay Multicast Networks", *Proceedings of 2002 IEEE conference on Computer Communication (INFOCOM'02), volume 3*, pp. 1200-1208.

[96] Singhal, S., Zyda, M. Networked Virtual Environments: Design and Implementation, *ACM Press*, pp. 181-246, 1999

[97] Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H., "Chord: A scalable peer-to-peer lookup service for Internet applications", *Proceedings of 2001 ACM conference on applications, technologies, architectures, and protocols for computer communications (SIGCOMM'01)*, pp. 149-160

[98] Tang, J., Isaacs, E., "Why Do Users Like Video? Studies of Multimedia-Supported Collaboration", *Computer-Supported Cooperative Work: An International Journal*, pp. 163-196, 1993.

[99] Taylor, M.J., Rowe, S.M., "Gaze Communication Using Semantically Consistent Spaces", *Proceedings of 2000 ACM conference on Human Factors in Computer Systems (CHI'00)*, pp. 400-407.

[100] TeamSpeak http://www.teamspeak.org/

[101] Tran, D.A., Hua, K.A., Do, T.T., "A Peer-to-Peer Architecture for Media Streaming", *IEEE Journal on Selected Areas of Communication (JSAC), Special Issue on Advances in Overlay Networks, volume 22, issue 1*, 2004, pp. 121-133

[102] University College London Mbone tools. http://www-mice.cs.ucl.ac.uk/multimedia/software

[103] van Renesse, R., Birman, K., Vogels, W., "Astrolabe: A robust and scalable technology for distributed system monitoring, management and data mining", *ACM Transactions on Computer Systems, 21(2), May 2003*, pp. 164-206

[104] Vertegaal, R., Weevers, I., Sohn, C., Cheung, C., "GAZE-2: Conveying Eye Contact in Group Video Conferencing using Eye-Controlled Camera Direction", *Proceedings of 2003 ACM conference on Human Factors in Computer Systems (CHI'03)*, pp. 521-528

[105] Video Interactive Multipoint: http://www.vianet.com/vi.htm

[106] Vogel, J., Widmer, J., Farin, D., Mauve, M., Effelsberg, W., "Priority-Based Distribution Trees for Application-Level Multicast", *Proceedings of 2003 ACM workshop on network and system support for games (NetGames'03)*, pp. 148-157

[107] Xu, J., "On the Fundamental Tradeoffs between Routing Table Size and Network Diameter in Peer-to-Peer Networks", *Proceedings of 2003 IEEE conference on Computer Communication (INFOCOM'03), volume 3*, pp. 2177-2187

[108] Xu, Z., Tang, C., Banerjee, S., Lee, S.J., "RITA: Receiver Initiated Just-in-Time Tree Adaptation for Rich Media Distribution", *Proceedings of 2003 ACM International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'03)*, pp. 50-59.

[109] Xu, Z., Mahalingam, M., Karlsson, M., "Turning Heterogeneity into an Advantage in Overlay Routing", *Proceedings of 2003 IEEE conference on Computer Communication (INFOCOM'03), volume 2*, pp. 1499-1509.

[110] Yahoo SuperWebcam Service: http://messenger.yahoo.com/messenger/superwebcam/

[111] Zhang, R., Hu, Y.C., "Borg: A Hybrid Protocol for Scalable Application-Level Multicast in Peer-to-Peer Networks", *Proceedings of 2003 ACM International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'03)*, pp. 172-179

[112] Zhang, B., Jamin, S., Zhang, L., "Host Multicast: A Framework for Delivering Multicast to End Users", *Proceedings of 2002 IEEE conference on Computer Communication (INFOCOM'02), volume 3*, pp. 1366-1375

[113] Zhao, B.Y., Huang, L., Stribling, J., Rhea, S.C., Joseph, A.D., Kubiatowics, J.D., "Tapestry: A Resilient Global-Scale Overlay for Service Deployment", *IEEE Journal on Selected Areas of Communication (JSAC), Special Issue on Advances in Overlay Networks, volume 22, issue 1*, 2004, pp. 41-53.

[114] Zhu, Y., Li, B., Guo, J., "Multicast With Network Coding in Application-Layer Overlay Networks", *IEEE Journal on Selected Areas in Communication (JSAC), Volume 22, No 1*, January 2004, pp. 17-120

[115] Zhuang, S.Q., Zhao, B.Y., Joseph, A.D., Katz, R.H., Kubiatowicz, J. D., "Bayeux: An Architecture for Scalable and Fault-tolerant Wide-area Data Dissemination", *Proceedings of 2001 ACM International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'01)*, pp. 11-20

APPENDIX

$\in$: belongs to
$\forall$: for every
$\varnothing$: empty set
|: such that
$a \succ b$: b is parent of a
$a \vdash b$: b is ancestor of a
{a,b}: directed edge connecting a to b
dist(a,b): total cost of all edges on path between a and b
c(a,b): cost of edge between a and b
outflow(b): available outflow of b
outflow_max(b): maximum outflow of b

MOTOR-Dynamic(s,t) $O(N^3)$

Input: given source $s \in V$, target $t \in V$, max_change: the maximum allowable number of changes and a link cost c(a,b) for every link connecting $a \in V$ and $b \in V$

Output: join target t to the tree Ts of source s, such that outflow_max(v) is respected for all nodes $v \in V$ with number of changes in topology <=max_change. Returns TRUE if successful

$A = \{q \mid q \in Ts, outflow(q)>0\}$
if $A \neq \varnothing$:
    $Ts \mathrel{+}= \{p,t\} \mid p$ has $\min[dist(s,p)+c(q,t)] \quad \forall p \in A$
    Return SUCCESS
$\forall a \in V, a \neq s$:
$B=\{(p,k,j) \mid p \in Ts, p \in Ta, k \in Ta, j \in Ta, j \succ p$ in Ta, $k \nvdash p$, $outflow(k)>0\}$
if $B \neq \varnothing$ and max_change>2:
    choose (p,k,j) with $\min[c(k,j)] \quad \forall (p,k,j) \in B$
    $Ta \mathrel{-}= \{p,j\}$
    $Ta \mathrel{+}= \{k,j\}$
    $Ts \mathrel{+}= \{p,t\}$
    Return SUCCESS
$C=\{b \mid b \notin Ts, outflow(b)>1\}$
if $C \neq \varnothing$ and max_change>3:
    choose r with $\min[c(s,r)] \quad \forall r \in C$
    $Ts \mathrel{-}= \{s,w\} \mid w$ has $\min[c(s,w)] \quad \forall w \succ s, w \in Ts$
    $Ts \mathrel{+}= \{s,b\}$
    $Ts \mathrel{+}= \{b,w\}$
    $Ts \mathrel{+}= \{b,t\}$
    Return SUCCESS
if max_change>4:

```
        d=Relocate_spare_outflow(c,s,t)
        if d≠NULL:
                attach(s,t,d)
                return SUCCESS
return REJECT_REQUEST
```

MOTOR (s,t,B,p) $O(N^8)$

Input: Given a complete graph G=(V,E), a maximum out-flow outflow_max(v) for all nodes $v \in V$, a target set Rs for each source, a requested bitrate for each (source,target) tuple fw(s,t) and a set of trees Tk each rooted at a source k and spanning its targets.

Output: find tree Ts for every source $s \in V$, such that outflow_max(v) is respected for all nodes $v \in V$ and $\sum c(e)$ is minimized for $e \in Ts$ for every $s \in V$.

```
if Assign_outflow()=FALSE:
        ∀ s∈D, |Rs|>|Ts|:
        Swap_rejected_requests(s,Rs)

Low_cost_tree(Ts)  ∀ s∈V
```

Assign_outflow() $O(N^8)$

Input: Given a complete graph G=(V,E), a maximum outflow outflow_max(v) for all nodes $v \in V$, a target set Rs for each source and a request flow weight fw(s,t) for each source-target pair (s,t). fw(s,t) for s=t is infinite.

Output: a set of trees Tk rooted at each source $k \in V$ and spanning its targets Rk such that outflow_max(v) is respected for all nodes $v \in V$ and if $i \succ j$ then $fw(i,s) \leq fw(j,s)$ for $i,j \in Ts$ for all $s \in V$. Returns TRUE if successful.

```
A={s | s∈V, outflow_max(s)> ∑fw(s,i) ∀ i∈Rs}
D={s | s∈V, s∉A}

∀ s∈A, v∈Rs :
        add edge={s,v} to Ts with flow weight fw(s,v)
outflow(s)=0 ∀ s∈D
for all s∈D in descending order of (outflow_max(s)- ∑fw(s,i) i∈Rs):
        outflow(s)=outflow_max(s)
        avail_outflow=outflow_max(s)+∑outflow(k), k∈Rs
        if(avail_outflow<∑fw(s,m) ∀ m∈Rs):
          if ∃ r∈V | r∉Rs, r≠s, outflow(r)≥ (fw(s,n)+fw(s,m))  m,n∈Rs:
                avail_outflow+=outflow(r)
```

```
            Rs+={r}
            fw(s,r)=Max(fw(s,n),fw(s,m))
        Total_outflow=outflow_max(s)
        Ts={s}
        min_flow = Min(fw(s,i) i∈Rs)
        while(Total_outflow>= min_flow & |Rs|>|Ts|) do:
            find u | u∈Ts with Max(outflow(j)) ∀j∈Ts)
            P={p | p∈Rs, fw(s,p)≤ outflow(u), fw(s,p)≤ fw(s,u)}
            Choose q∈P with Max(fw(s,q)) and use greater outflow(q) for tie-break
            add edge={u,q} to Ts with flow weight fw(s,q)
            Rs - = {q}
            Total_outflow+=[outflow(q)- fw(s,q)]
            Min_flow = Min(fw(s,i) ∀ i∈Rs)

∀ s∈D, |Rs|>|Ts|:
        Insertnode(s,t) ∀ t∈Rs, t∉Ts
if ∃ s∈D, |Rs|>|Ts|:
        Return FALSE
else Return TRUE
```

InsertNode(s,t) $O(N^5)$

Input: given source s∈V and target t∈V (t∉Ts) and flow weight fw(s,t)

Output: insert new target t to the tree Ts of source s with flow weight B, such that outflow_max(v) is respected for all nodes v∈V.

```
∀c | c∉Rs, outflow_max(c)- outflow(c)≥ fw(s,t):
        d=Relocate_spare_outflow(c,s,t)
        if d≠NULL:
            attach(s,t,d)
```

Relocate_spare_outflow(c,s,t) $O(N^4)$

Input: given a source s∈V, a node c∈V with outflow_max(c)- outflow(c)>=fw(s,t)

Output: relocate spare outflow of c to a node d∈Rs, such that outflow_max(v) is respected for all nodes v∈V. Returns a node d with outflow(d)≥fw(s,t) if successful, NULL otherwise.

```
d=NULL
∀ d | d∈Rs OR d=s, d≠c:
        if can_relocate(c,d,s,t)=TRUE: Return d
if ∃ a,d | a∉Rs, a≠c, outflow_max(a)-outflow(a)<fw(s,t), can_relocate(c,a,s,t)=TRUE,
d∈Rs OR d=s, d≠c,d≠a, can_relocate(a,d,s,t)=TRUE:
        Return d
```

---

can_relocate(c,d,s,t) $O(N^2)$

Input: given a node $c \in V$ with outflow(c)$\geq$fw(s,t) and a node $d \in V$, $c \neq d$

Output: see if it is possible to transfer the spare outflow of node c to node d such that outflow(d)$\geq$fw(s,t). Returns TRUE if successful.

```
if ∃ e | e∈V, d∈Te, c∈Te, |Te|>2:
        G={g | g≻d,  g,d∈Te}
        if ∃ H⊆G | H≠∅, outflow(d)+removed_flow)≥ fw(s,t), outflow(c)≥removed_flow,
                        fw(e,c) ≥fw(e,h) ∀ h∈H OR c ⊢ d
                        (where removed_flow = ∑fw(e,h), h∈H):
                remove edge={d,h} from Te, h∈H
                if c ⊢ h in Te & outflow(c)+removed_flow>=fw(e,d):
                        remove edge={f,d} from Te d≻f
                        add edge={f,h} to Te with flow weight fw(e,d)
                        add edge={c,d} to Te with flow weight fw(e,d)
                        Return TRUE
                else if c ⊬ h:
                        add edge={c,h} to Te, h∈H with flow weight fw(e,h)
                        Return TRUE
Return FALSE
```

attach(s,t,c) O(N)

Input: given source $s \in V$, a target $t \in V$ and a node $c \in Rs$ such that outflow (c) $\geq$fw(s,t)

Output: attach target t to the tree Ts of source s, taking advantage of spare outflow of c

```
if c∈Ts:
        if fw(s,c) ≥fw(s,t)
                add edge={c,t} to Ts with flow weight fw(s,t)
                Return TRUE
        else if fw(s,p) ≥fw(s,t) & outflow(p) ≥(fw(s,t)-fw(s,c)), c≻p in Ts:
                add edge={c,t} to Ts with flow weight fw(s,t)
                fw(s,c)=fw(s,t)
                outflow(p) -=fw(s,t)-fw(s,c)
                Return TRUE
else if c=t:
```

$\forall$ h$\in$Ts | fw(s,h) $\geq$fw(s,t):
G={g | g$\succ$h}
if $\exists$ K$\subseteq$G | K$\neq\emptyset$, [outflow(h)+$\sum$fw (s,k)] $\geq$fw(s,t) &$\sum$fw (s,k)$\leq$outflow(c)$\forall$k$\in$K:
remove edge={h,k} in Ts $\forall$k$\in$K
add edge ={h,c}to Ts with flow weight fw(s,c)
add edge={c,k} to Ts with flow weight fw(s,k) $\forall$k$\in$K
Return TRUE
Return FALSE

Swap_rejected_requests

Input: a source s$\in$V, priority for each request p(s,t) from source s to receiver t and Rs the receiver set of source s

Output: a tree rooted at source s and spanning a subset of Rs with lower priority requests rejected

$\forall$ s$\in$V such that |Rs|>|Ts|:
find u such that p(s,u)>=p(s,t) for all t,u$\in$Rs AND t,u$\notin$Ts,
Q={q$\in$Rs such that p(s,q)<p(s,u)}
$\forall$ q$\in$Q:
Rs=Rs-{q}
Assign_outflow(Rs)
$\forall$ t$\in$Q:
Insertnode(s,t)

Low_cost_tree () $O(N^3)$ (modified version of Compact Tree in [94])

Input: Given a complete graph G=(V,E), a tree Ts rooted at source s∈V, spanning its targets Rk and edge cost c(a,b) for every a,b∈E .

Output: a tree Tmin rooted at each source s∈V and spanning its targets Rk such that outflow_max(v) is respected for all nodes v∈V and ∑c(b) is minimized for all b∈Tmin.

```
for each r∈Ts
    for each v∈Ts
        di(v)=c(r,v)
        p(v)=r
    Tmin = (W={r}, L={})
    while(W≠V)
        let j∈V-W be vertex with smallest di(j)
        if outflow_max(j)>outflow(j) OR j is the last remaining receiver in Rk
            u=j
        else
            u∈V-W that has outflow(u)>0 with smallest di(u)
        W=W+{u}
        L=L+{u,p(u)}
    for each v∈W-{u}
        di(v)=Max{di(v),dist(u,v)}
    for each v∈V-W
        di(v) = ∞
        for each q∈W
            if outflow(q)>0 AND c(v,q)+di(q)<di(v)
            p(v)=q
```