Media Distribution using Overlay Multicast and Peer-to-Peer Technologies

Dissertation
zur Erlangung des Doktorgrades
der Mathematisch-Naturwissenschaftlichen Fakultäten
der Georg-August-Universität zu Göttingen

vorgelegt von

Jun Lei

aus Huzhou, China

Göttingen 2008

D7
Referent: Professor Dr. Xiaoming Fu.
Korreferent: Professor Dr. Dieter Hogrefe
Tag der mündlichen Prüfung: 17 Juli, 2008

Abstract

The explosive growth of multimedia services and applications (e.g. media streaming) demands an efficient, deployable media distribution system on the Internet. Although native IP multicast is regarded as an efficient way of delivering media streams to a group of receivers, it faces a number of technical and operational issues which have eventually prevented its widespread usage. The aim of this work is therefore to build a scalable, efficient, reliable and incrementally deployable infrastructure for supporting media distribution services.

In this thesis, a new framework named Dynamic Mesh-based overlay Multicast Protocol (DMMP) framework, and two important extensions to the basic DMMP protocol, namely Self-improved DMMP protocol (DMMP+) and Interest-shared Group Management (IGMT) protocol for DMMP, are developed to efficiently serve a large number of concurrent clients with relatively high inbound bandwidth and low start-up delay.

The DMMP framework dynamically manages a two-tier hierarchy, i.e., an overlay core so-called dynamic mesh, and clusters without relying on classic IP multicast. The key idea is to let a number of end hosts get selected and self-organized into the overlay hierarchy, and dynamically maintain such a hierarchy. In comparison to prior application layer multicast protocols, DMMP is more adaptive to group size changes, and provides efficient and reliable media delivery with less control overhead and less packet loss.

DMMP+ extends the basic protocol in the DMMP framework to optimize the data delivery hierarchy. Two self-improvement techniques are designed to gradually optimize the established overlay mesh and clusters. The analysis identifies that the DMMP+ protocol can assist the DMMP framework to be more scalable, reliable and efficient in the sense of providing better data path quality but less control overhead and packet loss.

The IGMT protocol further extends DMMP+ to help the transient or partitioned nodes quickly join/rejoin the group in a highly dynamic environment. Motivated by an experimental investigation on Joost's peer-to-peer management, IGMT allows nodes to maintain interest-shared groups and to establish shortcuts in addition to relying on centralized servers to join the group. The simulation results have demonstrated that IGMT is efficient and resilient to highly dynamic membership changes.

The combination of these new approaches constitutes a coherent and effective media distribution

architecture, which provides a great potential to support large-scale media distribution services. Noticeably, while the key techniques may be jointly used for providing efficient media distribution services, they can be used independently to effectively address scalability, efficiency and resilience issues in peer-to-peer overlay networks.

Zusammenfassung

Der explosionsartige Zuwachs von Multimediadiensten und Applikationen (wie z.B. Media Streaming) erfordert ein effizientes und einsetzbares Mediendistributionssystem im Internet. Obwohl IP Multicast ein effizienter Weg ist Medienströme zu einer Gruppe von Empfängen zu befördern, besitzt es technische und operative Probleme, die letztendlich eine weite Verbreitung verhindert haben. Das Ziel dieser Arbeit ist es darum, eine skalierbare, effiziente, verlässliche und schrittweise einsetzbare Infrastruktur für Mediendistributionsdienste zu entwickeln.

In dieser Arbeit werden ein neues Framework, genannt Dynamic Mesh-based Overlay Multicast Protocol (DMMP) Framework, und zwei wesentliche Erweiterungen zum grundlegenden DMMP Protokoll, und zwar Self-improved DMMP (DMMP+) und das Interest-shared Group Management (IGMT) Protokoll, entwickelt, um eine große Anzahl von Clients gleichzeitig mit relativ hoher Eingangsdatenrate zu versorgen und eine geringe Start-Up Verzögerung der Clients zu erreichen.

Das DMMP Framework verwaltet dynamisch eine Zwei-Stufen-Hierarchie, d.h. einen Overlay Kern, sog. Dynamisches Mesh, und Cluster, ohne das klassische IP Multicast zu verwenden. Die Hauptidee ist es, einige End-Hosts auszuwählen, welche die Overlay-Hierarchie bilden und dynamisch verwalten. Im Vergleich zu früheren Applikations-Layer-Multicast-Protokollen, ist DMMP besser anpassungsfähig an Änderungen der Gruppengröße und bietet eine effiziente und verlässliche Medienverteilung bei geringeren Kontrolloverhead und geringeren Paketverlusten.

DMMP+ optimiert die Datendistributionshierarchie durch Erweiterung des Protokolls im DMMP Framework. Zwei selbst verbessernde Techniken werden entworfen, um das gebildete Overlay-Mesh und die Cluster zu optimieren. Die Analyse zeigt, dass das DMMP+ Protokoll das DMMP Framework skalierbarer, verlässlicher und effizienter macht, in dem Sinne das es einen besseren Datenpfad und weniger Kontrolloverhead und Paketverluste erzielt.

Das IGMT Protokoll erweitert weiterhin DMMP+ dahingehend, dass kurzlebige oder abgetrennte Knoten in hoch dynamisch wechselnden Umgebungen schnell der Gruppe beitreten bzw. erneut beitreten können. Motiviert durch eine experimentelle Untersuchung des Joost Peer-to-Peer-Managements, ermöglicht IGMT den Knoten interessenbasierte Gruppen zu bilden und Shortcuts zu etablieren in Ergänzung zur Nutzung der zentralisierten Server. Die Simulationsergebnisse zeigen, dass IGMT effizient und robust gegenüber sehr dynamischen Wechseln in den Mitglied-

schaften ist.

Die Kombination dieser Methoden ermöglicht eine einheitliche und effektive Architektur, die ein großes Potential zur Unterstützung von umfangreichen Medienverteilungsdiensten bietet. Während die einzelnen Techniken gemeinsam genutzt werden können, um effiziente Medienverteilungsdienste anzubieten, können die Techniken unabhängig von einander genutzt werden, um Skalierbarkeit, Effizienz und Robustheit in Peer-to-Peer Overlay Netzwerken zu erreichen.

Acknowledgements

With respect and gratitude, I express my warm and sincere thanks to Professor Dr. Xiaoming Fu and Professor Dr. Dieter Hogrefe, whose expertise, guidance, support, encouragement, and patience has made this dissertation possible. Their suggestions and comments on my thesis research have vastly improved the quality of this dissertation and provided an enthralling educational experience.

The office staff members of the Institute of Computer Science (Institut für Informatik) have been wonderful. Also, I must admit that I truely enjoyed working with the research group of our entire Institute during the past three and half years.

I am grateful to David Weiss for his assistance with prototyping the components of my thesis. Based on the basic simulation analysis, I am able to specifically improve the performance in regards to efficiency, reliability and scalability of the framework.

I must also acknowledge my committee members: Professor Dr. Dieter Hogrefe, Professor Dr. Xiaoming Fu, Professor Dr. Stephan Waack, Professor Dr. Wolfgang May, Professor Dr. Bernhard Neumair and PD Dr. Hartje Kriete. Their comments and suggestions have improved the thesis.

I am deeply grateful to our NET group members, Ralf Lübben, Niklas Neumann, Niklas Steinleitner, Mayutan Arumaithurai, Lei Shi, John-Parick Wowra, Vicky Kuo, Ingo Juchem and Filippo Ammazzalorso for providing instant feedback on my research and the assistance with writing papers for publication in the past years. I would also thank our Telematik group members, especially, Patryk Chamuczynski, Omar Alfandi, for friendly cooperations and insightful discussions.

Spending three and a half years as a graduate student can be profitable. Several fellow graduate students and friends, Nikunj Modi, Christian Dickmann, Jan Demter, Jan Engelhardt, Lars Reimann, Swen Weiland have made my life colorful and as pleasant as it could possibly be. I really enjoyed the living and studies at the University of Göttingen.

I owe my most sincere gratitude to our colleagues and partners, Carmen Scherbaum de Huaman, Annette Kadziora, Udo Burghardt for the cooperation and support in the last few years.

Very special thanks to my friends, Dr. Jiong Yan, Dai Chen, Dr. Ralf Brauchler, Dr. Xiaoqin Zhang and Rui Zhang for inviting me to join several parties and special events in Göttingen. Their kindness and help make my life fruitful.

Finally and most importantly, I have to make a special mention of my entire family, my father Dr. Sheng Lei, my mother Hongying Zhuang, my grandmotherLihua Wei, and especially my husband Dr. Tianhui Chen for making a *wonderful* difference to my life. I would not be able to complete the PhD study if they didn't give the full support and understanding.

Table of Contents

List of Figures

List of Tables

List of Acronyms

ALM Application Layer Multicast

ARQ Automatic Repeat Request

AS Autonomous system

BREC Buffer-controlled Retransmission-based Error Control

CBT Cored Based Trees

CDN Content Delivery Network

CLN Children Level Node

CPC Candidate Parent Cache

DMMP Dynamic Mesh-based overlay Multicast Protocol

DMMP+ Self-improved DMMP Protocol

DNS Domain Name System

DVMRP Distance Vector Multicast Routing Protocol

E2E (e2e) end-to-end

FEC Forward Error Control

FGS Fine Granularity Scalable

GOP Group of Picture

HARQ Hybrid ARQ

ID Identifier

IGMT Interest-shared Group Management

IP Internet Protocol

ISP Internet Service Provider

LAN Local Area Network

MOSPF Multicast Open Shortest Path First

MPEG Motion Picture Experts Group

MSN Multicast Service Node

NAS Network Attached Storage

OM Overlay Multicast

OSI Open Systems Interconnection

OSPF Open Shortest Path First

P2P Peer-to-Peer

PLN Parent Level Node

PPP Point-to-Point Protocol

PRM Probabilistic Resilient Multicast

PIM Protocol Independent Multicast

QoS Quality of Service

RDP Relative Delay Penalty

RIP Routing Information Protocol

RP Rendezvous Point

RSVP Resource Reservation Protocol

RTCP Real Time Control Protocol

RTP Real Time Protocol

RTSP Real-Time Streaming Protocol

RTT Round-Trip Time

SAP Session Announcement Protocol

SDP Session Description Protocol

SIP Session Initiation Protocol

SAN Storage Area Network

TCP Transmission Control Protocol

TS Transit-Stub

TTL Time-To-Live

UDP User Datagram Protocol

VoD Video on Demand

WLAN Wireless LAN

Chapter 1

Introduction

1.1 Background

In recent years, there is an emerging need to support real-time media streaming over the Internet [1]. However, streaming multimedia traffic to a large number of customers imposes a high traffic load on the network. The high volume of such multimedia traffic alongside timing constraints requires a large-scale, cost-effective media distribution system.

Figure 1.1 shows an example of a traditional architecture for streaming stored videos over the

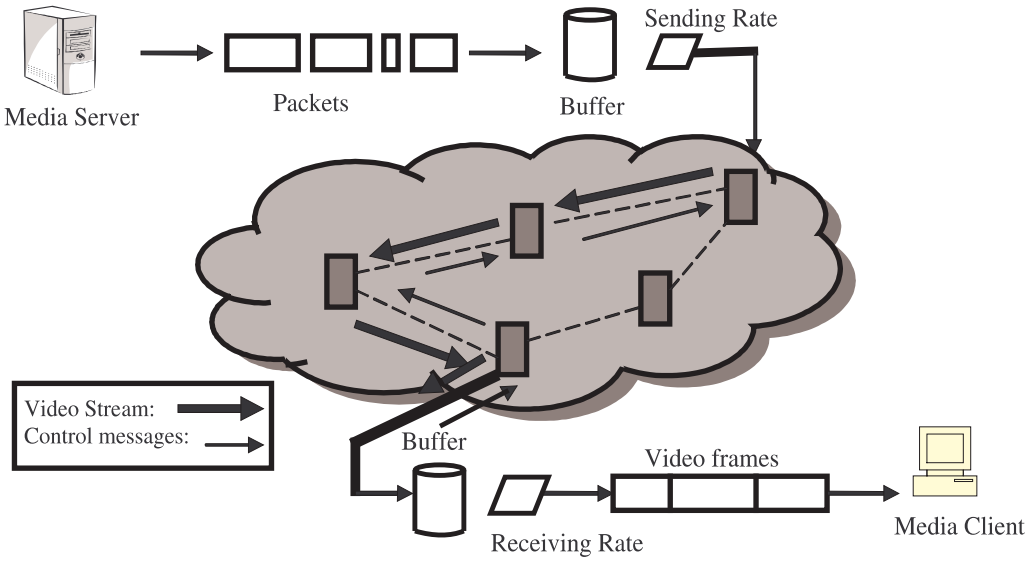

Figure 1.1: Basic Architecture for Video Streaming over the Internet.

Internet. Upon each client's request, the media server retrieves the video and multicasts it over User Datagram Protocol (UDP) at a constant rate which equals to the drain rate at the client side. However, these UDP packets may be dropped or delayed during the transmission over the Internet, e.g. due to network congestion. Thus, a buffering scheme is usually used at the server side before sending the video file into the network in order to control the sending rate to fit the current network status and service requirements. Besides, for efficiency reasons the video must be compressed before it is sent to the network. Once the client receives the compressed video from the network, it needs to decompress the video by its local media player. Before starting to play back the video, the client places the received packets into its own buffer, so-called client buffer, which is set to balance the receiving rate and the playing rate. Finally, the video can be decoded and played back properly at the client.

However, it becomes challenging for today's Internet to rely on such an architecture to deliver videos to a large number of users due to the following reasons.

- The traditional media streaming systems use client-server approaches to allocate a dedicated stream from a media server upon each client's request. However, limited processing power, memory size and limited out-bound network bandwidth of the streaming server causes a limitation on the total number of concurrent clients that the system can support [2].
- As IP multicast is not widely deployed and is not generally available as a service for average end users, most of the existing media distribution systems rely on native unicast protocols for delivering video. However, unicast is recognized as an inefficient way of delivering multimedia services to a large number of clients. It not only wastes the network resources but also raises scalability issues. The scalability issue results from the fact that adding Internet-scale potential users requires a commensurate amount of resource to the supplying server(s).
- Streams need to be transported reliably to the end-user across the Internet. However, the Internet is designed as a *best effort* network without considering application's Quality of Service (QoS) requirements. Communications between two end points are not guaranteed and packets may be lost or delayed if they traverse congested routers or links. Another reliability concern arises from the fact that only one type of entity (i.e. video server) is responsible for all clients. Thus, the server failure may take place due to instant, short- or even long-term overloads.
- The timing constraint makes the system even harder to design. Besides the timing requirements for playing video in time, the media streaming system should be able to detect and recover from failures quickly, so that the service disruption for the affected nodes is minimized.

Therefore, this thesis focuses on overcoming the problem of serving multimedia files to a large number of end hosts distributed across the Internet. A key challenge is to design a system that

is **scalable**, **efficient** and **reliable**, in the sense of being able to efficiently serve a large number of concurrent clients with relatively high inbound bandwidth and low start-up delay. In some circumstances, adaptiveness to available resources along the path from the server is required, as well as resilience to dynamic changes (e.g., network condition changes, membership changes). This thesis proposes protocols that effectively address the above issues, and uses theoretical and simulation-based analysis to evaluate their performance. Furthermore, through extensive experimental studies on a real-life Peer-to-Peer (P2P) Video on Demand (VoD) system inferred from its network traffic, the thesis provides a better understanding of the design requirements for video distribution systems using P2P technologies. Motivated by these requirements, the thesis integrates the Peer-to-Peer technology into the proposed framework to be more resilient and reliable even in highly dynamic scenarios.

The remainder of the chapter is organized as follows. Section 1.2 provides terminologies used in the thesis. The primary contributions of this thesis are briefly enumerated in Section 1.3. Finally, the roadmap of the entire thesis is given in Section 1.4.

1.2 Terminologies

The terminologies used in this thesis are listed as follows:

- Media Distribution – The principle of providing media information and content (e.g., video) over the Internet in the form of products or services.
- Source – The video service provider or sender. It could be a video stored server or some video distributed servers in one service domain (e.g., Autonomous system (AS)), which delivers data traffic to the subscribed video group members.
- Delay Jitter – The variability of packet delays within the same packet stream.
- Content Delivery Network – A set of networked computers cooperate together to transparently deliver content (especially for supporting a large mount of content) to end hosts across the Internet [3].
- Application Level Multicast – In contrast to network layer multicast (i.e., IP multicast), multicasting is performed at the application layer instead of at the network layer. That is, in the same multicast session each end host replicates the received packets and forwards to other end hosts using IP unicast. Therefore, the network routers don't have to upgrade to support multicasting.
- Overlay Multicast – A multicast data delivery scheme depends on end hosts or some infrastructure nodes to form an overlay network for message control and a multicast tree for data delivery.

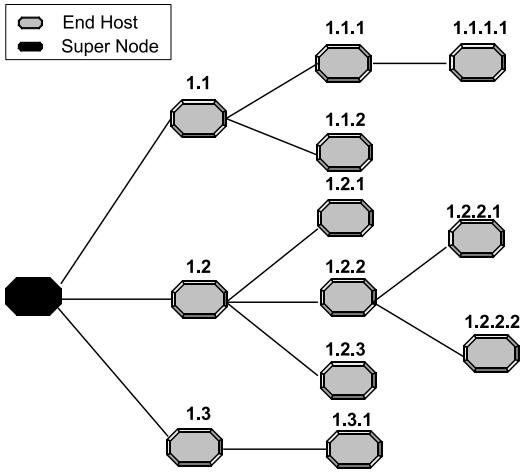

Figure 1.2: Example of Local Cluster.

- Rendezvous Point (RP) – A server or a proxy assists managing group members and stores some required information (e.g., performance related metrics).
- Receivers – They are multicast group members who want to receive the data from the source.
- Mesh – An overlay core, which is responsible for group member management and multicast tree configuration. In this thesis, the mesh is completely formed by selected end hosts.
- Super Nodes – Some end hosts are chosen to manage the multicast group and to relay data from the overlay core to receivers. Currently, only end hosts can serve as super nodes; extensions of this work may specify the case when some routers (e.g., first-hop routers) are used as super nodes.
- Clusters – Relying on each super node, end hosts organize themselves into a core-based multicast tree [4].
- Out-degree – Available connections, namely, the available number of connections that a node can establish.
- Uptime – The time duration from a node joining in a multicast session to its leaving the multicast session.

Within each cluster, there are some terminologies and abbreviations defined as follows:

- Parent – direct upstream node of a node is called the parent of that node, for instance, in Figure 1.2 end host 1.2 is the parent of end host 1.2.1.

- Parent level nodes (PLN) – nodes (exclusive parent) at the same level as the parent of some node, e.g., 1.1 and 1.3 are parent level nodes of 1.2.1 in Figure 1.2.
- Child – the direct downstream node of some node, e.g., in Figure 1.2, 1.2.1 is the child of 1.2.
- Children level nodes (CLN) – nodes (exclusive children) at the same level as children of some node, e.g., 1.1.1, 1.1.2, 1.3.1 are children level nodes of 1.2 in Figure 1.2.
- Siblings – nodes at the same level of a node are called siblings, e.g., in Figure 1.2, 1.1.1, 1.1.2, 1.2.2, 1.2.3 and 1.3.1 are siblings of 1.2.1.

1.3 Contributions

The main contributions of this thesis are shortly enumerated as follows:

- We propose a video streaming architecture based on overlay multicast, as described in Figure 2.9. To fulfill the requirements mentioned in Section 2.2, we propose an innovative two-tier framework based on overlay multicast and caching mechanisms.
- The proposed Dynamic Mesh-based overlay Multicast Protocol (DMMP) framework, as one of the first systematic proposals in this research field, addresses the scalability, efficiency and deployability issues in the existing approaches. Extensive theoretical and simulation analysis proves that DMMP has the potential to support large-scale media applications.
- The DMMP+ protocol extends the DMMP protocol in the framework to optimize the data delivery hierarchy. Two self-improvement techniques are designed to gradually improve the quality of established overlay hierarchy. The analysis establishes that DMMP+ can assist the DMMP framework to be more scalable, reliable and efficient in the sense of providing better data path quality but less control overhead and less packet loss.
- Through exploring the peer-to-peer management mechanisms in Joost, we propose an Interest-shared Group Management (IGMT) protocol for DMMP, which addresses the resilience issue of DMMP+ in highly dynamic scenarios. IGMT is shown to be efficient and resilient to dynamic membership changes by allowing nodes to maintain interest-shared groups and to establish shortcuts in addition to relying on centralized servers to join the group.

Note that in Section 7.2 we will further highlight the scientific achievements accomplished through the entire thesis.

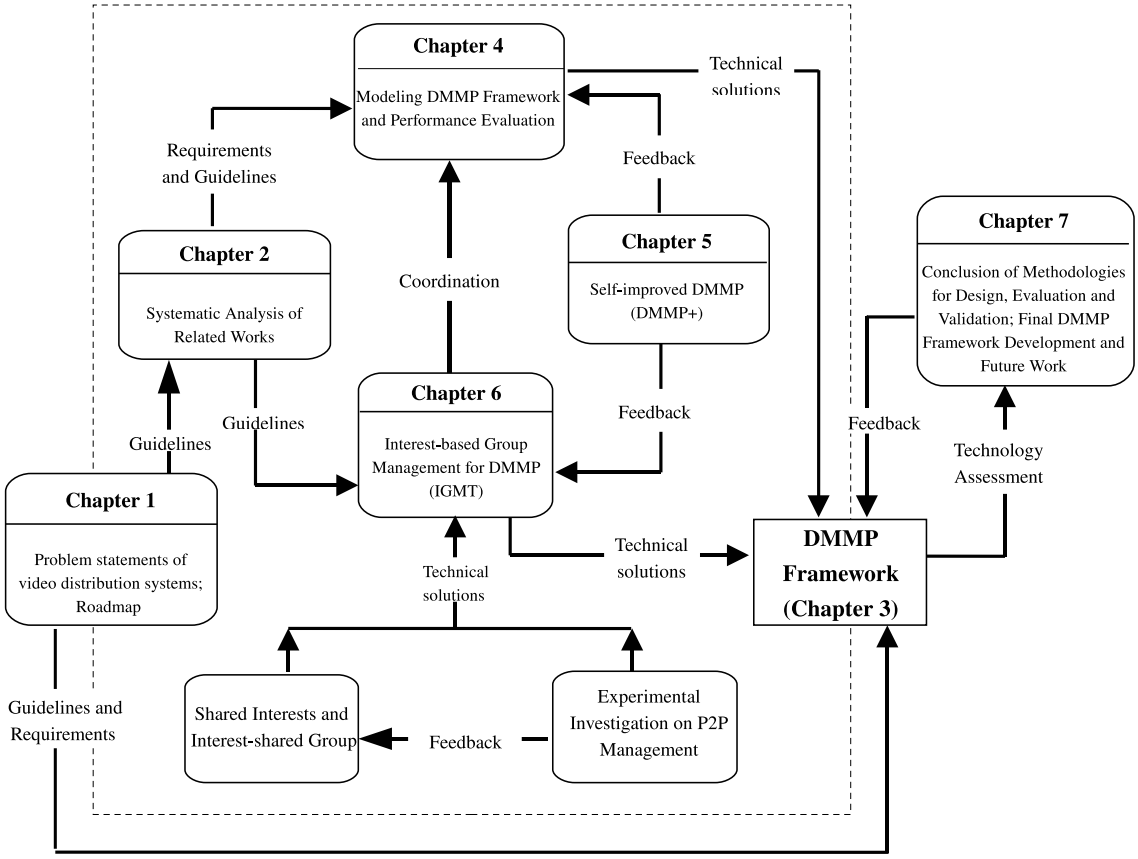

Figure 1.3: Thesis Roadmap.

1.4 Thesis Roadmap

The remainder of the thesis is organized as shown in Figure 1.3.

Chapter 2 presents related works. Motivated by the studies on related works, we propose a Dynamic Mesh-based overlay Multicast Protocol (DMMP) framework in Chapter 3. Chapter 4 illustrates the modeling of the DMMP protocol and evaluates its performance through numerical and simulation-based analysis. During the performance evaluation, we observe that the quality of DMMP-aware data delivery hierarchy may degrade due to dynamic membership changes. Thus, we propose a self-improved DMMP protocol called DMMP+ in Chapter 5, which gradually optimizes the established overlay hierarchy in the DMMP framework. Furthermore, peer-to-peer technologies have shown the capability of being resilient to dynamic group changes and network failures. In order to get a better understanding of peer-to-peer management, Chapter 6 first provides an experimental analysis of Peer-to-Peer (P2P) management in Joost which is one of the first commercial P2P VoD systems. Then, an Interest-shared Group Management (IGMT) proto-

col is developed in order to improve the DMMP+'s resilience and robustness in highly dynamic scenarios. Finally, we conclude the thesis and give an outlook in Chapter 7.

Chapter 2

Related Work

2.1 Introduction

The Internet was not originally designed for supporting multimedia applications. It offers a shared medium and a *best effort* delivery to distribute the multimedia content. Compared with the general Internet services, multimedia applications are typically sensitive to end-to-end (e2e) delay and delay jitter but more tolerant to packet loss. More specifically, it needs consistent bandwidth support, low transmission latency, low network jitter and in-order packet delivery.

There are four challenges in designing a scalable, efficient, and reliable media distribution system to satisfy the aforementioned needs. Firstly, one major issue of video distribution is to perform the streaming in a manner that a sequence of constraints * should be met. Any data that is delayed during the transmission cannot be used at the receiver, that is, the sequence of packets is vital to video streaming. Generally, it can estimate the available bandwidth and adjust the transmitted video bit rate to the available bandwidth. In most cases, there are, however, various bandwidth requirements if a single sender streams data to multiple receivers. Besides the sequence of arriving packets, the variation in e2e delay (i.e., delay jitter) can affect the quality of streaming video. Secondly, the medium for delivering video may vary due to dynamic network changes, which causes instability of the transmission. For instance, wired networks may be affected by network congestion; wireless channels are influenced by both bit errors and bursty errors [5]. When considering the bandwidth

*Consider the time interval between displayed frames to be denoted by Δ, e.g. Δ is 33 ms for 30 frames/s video and 100 ms for 10 frames/s video. Each frame must be delivered and decoded by its playback time; therefore the sequence of frames has an associated sequence of deliver/decode/display deadlines:

1. Frame N must be delivered and decoded by time TN
2. Frame N+1 must be delivered and decoded by time TN $+\Delta$
3. Frame N+2 must be delivered and decoded by time TN $+2\Delta$
4. Frame . . . and so on

constraints, it should thus focus not only on the amount of available bandwidth but also on the consistency and quality of the resources. As explained in Section 1.1, current streaming technology bypasses the timing problem by buffering a certain amount of content before the media file can be really played back. Nevertheless, it causes another problem that users have non-sequential access request [6]. For example, if one user wants to start playing from the middle position of a video, it is difficult to retrieve this specific playing position by buffering. Thirdly, due to the fact that today's Internet is lack of QoS considerations, it makes the transmission of video more challenging. There is no dedicated resource reserved for upcoming or ongoing video transmission. Lastly, there are still some security issues left for media distribution unresolved, such as access control, data confidentiality in inter-domain communications. Rather than attempting to cover the entire spectrum of media distribution research, this chapter will discuss related work that is most relevant to this thesis.

The remainder of this chapter is organized as follows. Section 2.2 identifies the video requirements which are very different from any other applications such as file sharing. According to identified requirements, Section 2.3 presents the media related considerations, and Section 2.4 illustrates some existing media distribution architectures. Section 2.5 gives networking considerations in order to meet the networking requirements mentioned in the Section 2.2. Some security considerations are provided in Section 2.6. Section 2.7 proposes an overlay multicast-based video distribution system, in combination with some caching mechanisms to fulfill the above requirements as well as to overcome the aforementioned challenges. Finally, a short summary is given in Section 2.8.

2.2 Video Distribution Requirements

In accordance to the above four challenges, we classify them into four categories according to system requirements:

- ◊ *Media related requirements:* Before video/audio can be transmitted over a computer network, it must be digitized and compressed. Digitization is necessary because computer networks transmit bits, so all transmitted information should be represented in bits. The need for compression is obvious: uncompressed audio and video consume tremendous amount of bandwidth and storage. At the receiver side, the processed video needs to be processed again, for instance, decompression, decoding, synchronization [7]. [8] have introduced several methods for the formal synchronization requirements in a multimedia environment.

- ◊ *Architectural requirements*: The media distribution architecture plays a key role in the supporting multimedia services. To offer qualified services, it is necessary to consider above QoS requirements into designing the main architectural components including media server, network filtering, network monitoring and the client. For example, using a network filtering can alleviate the impacts on the video quality due to a network congestion. Network mon-

itoring can be used to periodically report the network status, e.g. bandwidth utility, packet loss, delay, which are useful to balance the video sending/receiving rate.

◊ *Networking requirements:* Since Internet's best effort and unicast service model provides neither efficient routing nor QoS guarantees for a high-quality video delivery, different networking control mechanisms have to be taken into consideration. The previous work on networking control mechanisms spans all layers of the Open Systems Interconnection (OSI) networking suite, such as network layer, transport layer and session control layer.

◊ *Security requirements:* Security issues have becoming more and more serious in every research field. Without a surprise, there are still some gaps to be filled in the current multimedia distribution systems. We emphasize that our main goal here is not to design a bullet proof system. Instead, our goal on the security aspect is to provide a simple and efficient solution which makes the video distribution system not worse and in many cases better than today's system.

In the following sections, we present the architectural considerations for supporting video streams over the Internet, while meeting the above-identified requirements. For each aspect, we first identify the major existing problems and then investigate potential solutions.

2.3 Media Related Considerations

2.3.1 Video Compression

Transmission of uncompressed video consumes a large amount of bandwidth. To save the resource and achieve the efficiency of transmission, video must be compressed before transmission. Nowadays, video compression depends on a coding/decoding system to standardize the video types [9] [10] which can be recognized by the receivers. As depicted in Figure 1.1, at the receiver side the encoded video data is decoded and played back in a proper way. Here, we provide a brief and high-level introduction on Motion Picture Experts Group (MPEG) type which is the most common video encoded type and belonging to one of the three open (ISO/IEC) standards [11].

MPEG-1 [12] was originally designed for Video Home System (VHS) quality video on CD-ROM in 1988. Later, MPEG-1 is considered as a major storage format for a group of videos and audio, and it does offer excellent streaming quality for the specific bit-rate it supports.

MPEG-2 [13] was published in 1994 and was used to encode video and audio for broadcast applications. For instance, MPEG-2 is widely used as the format of digital television signals that are broadcast by cable, direct broadcast satellite TV or over the air. It also specifies the format of movies, which can be distributed on standard commercial DVD. Typically, MPEG-2 creates a video stream out of three types of frame data: intra-frames (I), forward

predictive frames (P) and bidirectional predicted frames (B). All of them can be arranged in a specified order called the Group of Picture(GOP) structure.

MPEG-4 [14] [15] was introduced in 1999 and a standard developed specifically for web streaming media, CD distribution, and conversational services. It integrates most of the merits of MPEG-1, MPEG-2 standards and other related standards (e.g. WMV). Therefore, it is capable of representing audio, video, images, graphics and text as separate objects, and can even multiplex and synchronize these objects into scenes. Besides, the MPEG-4 standards provide embedded error resilience capabilities to detect and recover errors, and to visually conceal the impact of errors by embedded error correction mechanisms.

According to encoding strategy, video compression can be classified into scalable and non-scalable video coding [16]. Scalable video encoder compresses a raw video into multiple sub-streams. One of them is base sub-stream and others are called enhancement sub-streams. The base sub-stream can be independently decoded, and provides coarse visual quality; enhancement sub-streams are decoded with the base sub-stream together to provide enhanced video quality. According to the available bandwidth, the receiver adapts to different levels of video quality. A scalable video encoding provides a compromise solution to meet heterogeneous demands of clients. Differently, non-scalable video coding only provides different quality levels of encoded video, such as a high-quality video, a medium-quality video or a low-quality video. Before delivering the requested video to the user, the server selects a certain quality video according to user's requirements or certain service agreement.

More recently, the use of path diversity has been studied as an alternative to provide extra dimension of adaptability to video services. Apostolopoulos et al [17] proposed a means of simultaneously transmitting several sub-streams of the video over different paths, while each sub-streams encodes a partial description of the video. The video can be decoded correctly, even if some of the sub-streams are lost. Nevertheless, it may encounter the redundancy problem, which reduces the transmission efficiency and wastes network resource.

2.3.2 Media Server

After compression, the media server needs to store the pre-processed video data into a selected storage device. For today's high-quality video service, the media server needs to consider two additional aspects: 1) the timing constraints; 2) interactive operations such as play/pause/stop, fast forward/fast backward. Therefore, three elements are mostly involved: a storage system, an operating system and a communication system. But in the following context, we only discuss about the storage systems and operating systems. The communication systems can be considered as the communicator between the session layer QoS control and transport layer protocols. For clarification, detailed information about media distribution architecture and protocol stacks will be respectively shown in Section 2.4 and Section 2.5.2.

Storage systems

The storage system is responsible for creating and storing video content. The video providers use various tools to produce the content. One example is video converter, which adds animation (e.g. advertisement) into certain video format which the media server can stream to the clients. Another example is using production tool which can optimize the video for the efficient delivery over the Internet, depending on the original quality of the material and the capabilities of the client computers.

Furthermore, a storage system for processed video has other requirements including high throughput, high capacity and error-tolerance [18]. Although large disk capability is already available for storing a large amount of data, frequent requests and high-throughput needs a more reliable and flexible storage system. To support large-scale demands, a hierarchical storage architecture can be suitable. Suppose that videos are stored according to the "priority" (e.g. regarding its popularity) in the hierarchical storage architecture. Video files with high popularity, namely, frequently requested video streams are kept on disks or quick access devices; the remainder stays at the automated tape library. Other possible solutions: 1) Storage Area Network (SAN) [19] [20]; 2) Network Attached Storage (NAS) [21] are two examples for supporting large-scale video streaming services. To be resilient to the disk errors, redundant media content should be kept in the storage system, however, it might waste resources if all video files have to be maintained twice. Obviously, there is a trade-off between the reliability of storage and cost of maintenance.

Operating systems

The above discussions concentrate on the hardware requirements for the media severs, whereas application requirements are essential to build and maintain an efficient video management system. In fact, an operating system builds a bridge between the hardware and the applications. It can be used to support interactive operations besides timing constraints. Existing systems can provide acceptable playing mechanisms, but interactive operations like VCR are rather difficult to achieve. Interaction operations require not only an efficient support of media servers but also a high capacity of alleviating the impacts from the network such as network congestion, packet loss. Moreover, at the client side the CPU power, memory support and hard disk space may have influence on the quality of video service. For an effective and flexible operating system, the above issues should be taken into considerations. However, operating system related issues will not be investigated in details as they are application specific requirements.

2.3.3 Video synchronization mechanisms

Besides the server side, some functionalities are necessary to be implemented at the client side, such as video synchronization within a media player. Based on video synchronization, the clients are able to present video streams in the same way as the videos were originally generated at the

media server. The major issues for media synchronization include how to specify the synchronization and where to implement it. Thus, these issues fall into two research branches: intra-media synchronization, and inter-media synchronization. These two types of synchronization have tight connections with three semantic layers of multimedia data: media layer, stream layer and object layer [22].

- *Intra-media synchronization*: It reflects the time relationship between presentation units of one media object. For example, it can represent the time between single frames of a video sequence. To guarantee the video received with required jitter, throughput and latency, the time constraints must be kept same across a single continuous media connection. Without this type of synchronization, the video may pause or stop during the playback.
- *Inter-media synchronization*: It is more complicated than the intra-media type, and concerns about the temporal relationships among different continuous media (e.g. audio and video). A prominent example of inter-media synchronization is the lip-synchronization scenario. Without inter-media synchronization, the movements of the lip of a speaker don't match the presented audio.

Therefore, we suggest using *an integrated method* with both intra-media synchronization and inter-media synchronization in order to accommodate video under the timing constraints. Usually, three aspects are necessary to be considered: 1) normalized clock times which represents a relationship between the clock of the media server and the clocks of the destinations; 2) normalized relative time-stamps which should be preserved among the media data from the media server to the destinations; 3) detection of asynchrony which is based on policies to trigger the synchronization mechanism if the media streams are out of the synchrony.

Basso et al. [23] presented a flexible framework for synchronization of multimedia streams. They utilize two collaborative modules, a transmitter-driven module and a local inter-media synchronization module, to synchronize the incoming streams. Whenever the first module is not enough to guarantee a reliable synchronization (e.g. since the encoder does not know the exact timing of the decoder), the second module provides further assistance to synchronize the media.

2.4 Media Distribution Architecture

Delivering the compressed video while considering aforementioned QoS requirements from a media server to a number of clients heavily depends on the service infrastructure. Four main approaches have been proposed to support media distribution services: Content Delivery Network (CDN), Network Layer Multicast, Application Level Multicasting (ALM) and Peer-to-Peer (P2P) Content Distribution.

2.4.1 Content Delivery Network (CDN)

Content Delivery Networks (CDNs) [3], [24] have evolved to overcome the user perceived QoS issues when users access a remote media content. In general, the CDNs replicate the content from the original media server to a set of cache servers and place some of the cache servers at the network edge, close to any receiver. The cooperation among the cache servers allows distributing video to end users in a reliable and timely manner.

The main idea of using CDNs is to offer fast and reliable applications and services by maximizing the system bandwidth, improving the accessibility and maintaining the robustness through content replication. A CDN may compose the following four infrastructures:

- *Content Delivery infrastructure*: The content delivery infrastructure consists of the original content server and the cache servers that deliver copies of the content to end hosts.
- *Distribution infrastructure*: The distribution infrastructure transports contents from the original server to cache servers and ensures consistency of the video contents in the caches.
- *Request-routing infrastructure*: The request-routing infrastructure is mainly responsible for directing clients' requests to the appropriate edge servers. Thus, it needs to periodically contact with the distribution infrastructure to track the updated content stored in the CDN caches.
- *Accounting infrastructure*: The accounting infrastructure maintains records of client accesses and the usage of the serving CDN servers. The information is used for traffic reporting and usage-based charging.

Peng *et al.* [3] presented an overview of CDNs, which included the critical issues involved in designing and implementing an effective CDN, and gave a survey of selected approaches to address these issues. Vakali *et al.* [24] presented a survey of existing CDN architectures and several popular CDN service providers. Differently, this survey was intended to provide an understanding of the CDN framework and its functionalities. Dilley *et al.* [25] provided an insight into the overall system architecture of the leading CDNs, so-called Akamai, [24], [26]. It gave an comparison of existing content delivery approaches and highlighted the features of the Akamai network infrastructure and its operational functions. Besides, it identified the technical challenges while constructing a global CDN like Akamai. Pathan *et al.* [27] provided a more comprehensive survey in terms of organizational structure, content distribution mechanisms, request redirection techniques, and performance measurement methodologies. Recent studies have focused on how CDNs can support efficient content delivery to large-scale network users.

Besides the efficiency requirement, content replication is designed to improve the scalability of a media delivery system. It has some objectives, such as a reduction of the end-to-end latency for clients, and a reduction of bandwidth occupation on the underlying network links. To achieve these goals, caching and mirroring are commonly used. Nevertheless, they face a more critical

problem of determining replica locations, i.e. how many replicas should be placed and where to place them. Therefore, content replication may not be feasible in the case of supporting a large amount of video data but low resource utilization.

To summarize, the cost of CDNs infrastructure setup and administration is expensive, and in the near future CDNs still face scalability and efficiency issues, if content providers and end users seek to receive high quality content [27]. Therefore, in this thesis CDN-related technologies will not be further investigated.

2.4.2 Network Layer Multicast

Multicast is yet another solution which has emerged as an efficient mechanism for supporting media distribution services. In the past few years, IP multicast was regarded as the most efficient technology for one-to-many, many-to-many or many-to-one data transmission.

In this section, we briefly describe the native IP multicast protocols, namely, Distance Vector Multicast Routing Protocol (DVMRP), Multicast Open Shortest Path First (MOSPF), Protocol Independent Multicast (PIM) and Cored Based Trees (CBT). While IP multicast is not widely deployed due to its technical and operational issues, lessons and experiences learned from these existing approaches are very essential towards building an efficient media distribution system.

Distance Vector Multicast Routing Protocol (DVMRP)

DVMRP is the first multicast protocol proposed in 1988 [28], which extends the unicast distance vector routing protocol *Routing Information Protocol (RIP)* to support multicasting. However, it builds its own multicast routing table based on which it constructs a *reverse path forwarding tree*. Originally, it was assumed that the group members are densely distributed over a network and therefore uses a broadcast and prune mechanism. DVMRP routers flood datagrams to all interfaces except the one that provides the shortest unicast route to the source. If there is no multicast subscribers in a certain subnetwork, the designated router will request its upstream router and accordingly it will be pruned from the tree. Obviously, this approach does not scale well due to its inefficient routing management and data delivery.

The main reason why DVMRP fails to provide multicast services for a large-scale group is because it depends too much on the particular unicast routing protocol, RIP.

Multicast Open Shortest Path First (MOSPF)

Later, the Routing Information Protocol (RIP) was replaced by a link-state routing protocol named *Open Shortest Path First (OSPF)*. The MOSPF protocol was a multicast extension to OSPF and proposed in 1994. In this approach, all routers in a routing domain (e.g. AS) have a complete, up-to-date information of the underlying topology and all group members. The computation of

the shortest path uses Dijkstra's algorithm, but the distribution of the link-state packets relies on a reliable broadcasting mechanism so-called flooding, which is however not scalable for wide area network like the Internet.

MOSPF is incapable of providing large-scale multicast services over the Internet because it still heavily relies on a specific unicast routing protocol and thus it raises a strong concern on the scalability.

Protocol Independent Multicast (PIM)

PIM appears to be the most widespread network multicast protocol [29]. It provides two different modes of multicasting [30]: 1) dense mode (PIM-DM) [31] where the session is used for a high node density; 2) sparse mode (PIM-SM) [32] in which the density is low. PIM-DM utilizes a shared tree, that is, several routers are connected into a data delivery core which is shared by all source hosts. PIM-SM starts with a shared tree as well but it has the ability to switch into a source-specific tree. For both PIM-DM and PIM-SM modes all data packets from the source will be forwarded cross a centralized point, usually called Rendezvous Point (RP).

Note for there are two extended modes, namely Bidirectional PIM and PIM Source Specific Multicast (PIM-SSM). The Bidirectional PIM does not build a shortest path tree and can scale better than PIM-SM because it requires no source-specific state. However, it may have much longer e2e delays than that of PIM-SM. The PIM-SSM protocol builds a single-source tree, offering a more secure and scalable model for supporting a limited amount of applications (such as TV broadcasting) [33].

Cored Based Trees (CBT)

To make IP multicasting more scalable, Core Based Trees (CBT) is proposed to construct a tree of routers [4]. The main difference of core-based trees from other multicasting schemes is that the routing tree comprises multiple "cores". The locations of the core routers are statically configured and other routers are added by extending branches of the tree. Therefore, it is perceived as a spare mode protocol. Unfortunately, it depends on the same root for all source-based distribution and defines a complex algorithm to construct and maintain the shared tree. Because of being lack of deployability, CBT is not any more under current use.

2.4.3 Application Level Multicasting

Unfortunately, applications of network layer multicast (i.e. IP multicast) for worldwide media distribution services remain limited due to several problems, even if many routers could be upgraded to support multicast. Those issues include: a lack of appropriate charging models, no scalable inter-domain routing protocol and little support in access control and effective network

management [34], [35]. To solve these issues, various application level multicast solutions have been proposed in order to move the multicast support out of the network core. They can be largely classified into two categories, namely, Application Layer Multicast (ALM) and Overlay Multicast (OM), due to their differences in overlay construction and membership management for a multicast group.

In a typical ALM approach, end hosts form a virtual overlay network, and multicast delivery structures are constructed on top of the overlay. As an extension to ALM, the OM approach employs some explicit routers as overlay proxies for obtaining and utilizing the knowledge of underlying network topologies. Media distribution systems can benefit from overlay networks as a result of the following characteristics: adaptation, self-organization, fault-tolerance, availability through massive replication, and the ability to construct dynamic meshes and harness large amounts of resources.

Both application layer multicast and overlay multicast have to construct an overlay hierarchy on the top of underlying network topology, and therefore we first discuss the possible ways in which an overlay construction takes place, and then identify the main features of current Application Layer Multicast (ALM) protocols and Overlay Multicast (OM) protocols, which are the two categories of the application level multicast.

Construction of Overlay Hierarchy

The construction techniques for an overlay hierarchy can be classified into five categories: centralized, tree-based, mesh-based, hybrid and special logic structures.

- **Centralized structure**: In this approach, a tree manager node or central controller answers for computing a Minimum Spanning Tree (MST) based on application-specific performance metrics (e.g. end-to-end latency, available bandwidth). For example, ALMI [36] measures round-trip times (RTTs) experienced by group members, since latency is critical for many applications and is also relatively easy to monitor. Although the efficiency of ALMI multicast trees approximates the efficiency of IP multicast trees, ALMI has a limited scalability as it can only support tens of members in a group. Theoretically, with the centralized approach it is easy to perform overlay routing, since all group information can be managed by the tree manager node. However, loops and partitions might still occur, e.g., when some packets are delayed or lost by some members, which will inevitably break up the tree-like connections.
- **Tree-based structure**: Group members self-organize into a tree structure, based on which group management and data delivery will be performed. The main advantages of tree are easy implementation, small maintenance costs and a good scalability. However, tree structure has fundamental limitations both for high bandwidth multicast and for high reliability. The former difficulty results from the fact that bandwidth will be monotonically declined along the tree; for instance, a member in OMNI [37] receives data only from its upstream node and the data reception rate of this member cannot be higher than that of its upstream

node. It is even more difficult in the core network where each Multicast Service Node (MSN) is responsible for data delivery to a whole cluster. That is, any packet loss caused at the upstream part of the tree will reduce the bandwidth available to downstream receivers. The second limitation is caused by single node failures or loops, which can partition the tree and eventually disrupt communications among the members.

- **Mesh-based structure**: In contrast to tree-based structure, a mesh use multiple links between any two nodes [38]. Thus, the reliability of data transmission in a mesh is relatively higher. Before transmission, a link evaluation is usually required to select ”better quality” links from the mesh in order to achieve the efficiency of data delivery. However, the cost of maintaining such a mesh is much larger than maintaining a tree. So far, large-scale groups usually use tree while small or medium-sized groups use mesh.

- **Hybrid structure**: Some approaches such as TOMA [39] propose two-tier overlay multicast architecture, where some service nodes or special proxies are strategically deployed in the overlay network. Besides, group members construct a core-based P2P multicast tree with other end hosts close by. Taking both advantages of tree-like and mesh-based structures, hybrid structure may efficiently deliver the multicast services to large groups. However, it still encounters some difficulties with achieving the flexibility and good performance.

- **Special logic structure**: In this approach, a special logic structure is required to organize the multicast group nodes through (re)mapping. For example, CAN (Content-Addressable Network) [40] maps a virtual d-dimensional space into several zones. In this d-dimensional coordinate space, two nodes are neighbors if their coordinate spans overlap along d-1 dimensions and joint along one dimension.

  Figure 2.1 depicts an example 2-d space with seven nodes. Here, node 4 is a neighbor of node 3 because its coordinate space along *y*-axis overlaps with 3’s and its *x*-axis joints with that of node 3. Moreover, node 1 is not a neighbor of node 3 as its coordinate joints with 3’s both in *x*-axis and *y*-axis. This purely logic neighbor structure is sufficient to route between two arbitrary nodes in the space: A CAN node routes a message by simply greedy forwarding to its neighbor with coordinate closest to the destination coordinate. The special logic structure is assumed to scale better than tree- and mesh-based structure, and requires no explicit routing algorithms. Furthermore, the number of status information kept in each node is reduced via using the logic structure. Nevertheless, the logic structure after mapping may not well utilize the underlying network capabilities.

Application Layer Multicast

Since several researches on ALM protocols have been proposed, in this section we give a systematic survey of some typical application level multicast approaches. We first illustrate their key ideas and then identify the major issues that have not yet been addressed or required for further investigations.

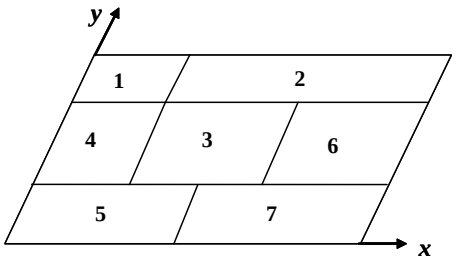

Figure 2.1: Example of 2D Coordinate Space with 7 Nodes.

End System Multicast (Narada)

The End System Multicast (ESM) [41] is entitled to be one of the first application level protocols, which demonstrates the multicast functionality could be implemented at the application layer. The success of ESM owes to that it only focuses on small groups, which evades the scalability and control overhead problems since the group size is limited.

It constructs and refines a source-rooted multicast tree in two steps. First, it builds a fully connected mesh and tries to ensure that the mesh has some desirable performance properties. Then, it utilizes a distance vector routing algorithm to build the spanning trees of the mesh. Each tree can be periodically optimized for each source by adding/deleting overlay links.

To ease the membership management, Narada defines a Rendezvous Point (RP) to maintain the status information of all group members. The information is also used to bootstrap the newly joining members. For example, when a newcomer wants to join the multicast group, it firstly contacts the RP to get a list of group members who have already joined the mesh. From this list, it randomly selects a subset of members and attempts to join as the neighbor of them. The joining procedure continues until at least one of these members accepts the newcomer as their mesh neighbor.

After joining the mesh, the new member immediately starts exchanging refresh messages with its mesh-neighbors. Each member of Narada needs to maintain a list of all other members in the group and to periodically exchange its knowledge of other group members with its neighbors. Distribution of such state information about each member to all other members leads to a relatively high control overhead. Thus, Narada protocol is effective only when the multicast group size is small or medium.

Considering the data delivery plane, Narada constructs spanning trees on the top of the mesh. When a newcomer joins, or when some failures occur in the mesh, a random set of mesh edges are added into the mesh. Similarly, some inefficient links can be dropped when new links can provide higher performance in terms of e2e latency. In this way, Narada can be perceived as a refinement-based protocol that refinements can be periodically made to improve the quality of data delivery paths. To avoid frequent adding/dropping links, the gain of refinement has to be significant before changing a mesh link.

Currently, ESM has been deployed to support media streaming application [42] over the Internet. Due to its deployment in the reality, we will choose it as the deployability benchmark of our performance studies in Section 4.4.

NICE + PRM

NICE [43] proposed a completely different method of overlay construction from Narada. It is a cooperative framework using a distributed algorithm through which nodes are self-organized into a top-bottom hierarchy. Each member must join the lowest layer and a distribution clustering protocol at each layer partitions these members into a set of clusters. Only one node of each cluster can be elected as the leader to join into the next higher layer. Usually, the leader locates geographically at the core of the cluster. Layer zero contains all nodes, while the highest layer contains only one end host. The layered design simplifies the membership management and helps it scale better.

Similar to Narada, NICE uses a RP to help bootstrapping newcomers. A newly joining member firstly contacts with the RP which sends back a list of all members of the highest layer. By probing each of them, the newcomer finds the ”closest” one and contacts it to get a list of all other available cluster members at the lower layer. This process iterates until the new host joins the specific cluster at the lowest layer, namely, layer zero. Furthermore, each leader at any layer must periodically check the size of the cluster. If the cluster size exceeds a pre-defined threshold, the cluster splits itself into two same-sized sub-clusters. If the cluster size is far beyond a pre-defined threshold, it merges with another small-sized cluster. For data forwarding, each member replicates and forwards the received packets to its neighboring clusters (except from which it receives the packet) of which it is a member at that layer. In this forwarding mechanism, a NICE host can have as many as $O(klog_k^N)$ peers along its data path.

For a group size of 32 members, NICE has low link stress, improved or similar end-to-end latencies, and better resilience than Narada [43]. However, each newcomer joining the group must estimate the end-to-end latency from the top layer till the lowest layer. Such a joining mechanism may raise three concerns: 1) the control overhead might be very high; 2) prolonging the packet delivery; 3) being weak to single node failures (e.g. the node at the highest layer).

To overcome the second and third weakness, Probabilistic Resilient Multicast (PRM) [44] was proposed to handle the case especially when there are high packet loss and host failures. PRM aims at improving data delivery ratios by both proactive and reactive mechanisms. The proactive component uses a simple, low-overhead randomized forwarding mechanism. Each overlay node periodically sends a few extra packets along randomly selected overlay edges. Thus, the overlay nodes can receive data through randomly selected edges if there is a failure occurs at a certain part of the multicast tree. In the reactive mode, overlay nodes calculate gaps between the sequence numbers of received packets to detect missing data. In fact, the proposed data recovery mechanism can be applied into other ALM solutions since it is independent from overlay hierarchy. Nevertheless, the proactive mechanism introduces much higher overhead, whereas reactive mechanism requires much longer time to detect and recover from the failure.

Note that NICE is demonstrated to have a good scalability and one of our main targets is to design a scalable media distribution system. Therefore, NICE is selected as the second benchmark for the performance studies in Section 4.4.

HMTP

Host Multicast Tree Protocol (HMTP) [45] is the first hybrid approach of network layer multicast and application layer multicast, which interconnects IP-multicast-enable islands using application layer multicast solutions. Within each island, native IP multicast is used to deliver data. Then, data encapsulated in UDP packets flow from one island to another through tunnels established by some designated members (e.g. proxy).

HMTP belongs to tree-first structure (c.f. Section 2.4.3), where each member tries to find its parent on a shared tree. New member joins the group by searching a partial list of existing members to find a qualified parent (in terms of having a low e2e latency). In this case, each newcomer searches from the root and sets the root as its potential parent. From the root and its direct children, the newcomer chooses the "closest" one as a new potential parent. The procedure continues until the new joining member reaches a leaf node or a node that is closer than any other neighbors. The main idea behind HMTP is, in some sense, similar to the Greedy algorithm [46] in which each member tries to attach to the tree as near as possible so that the end-to-end delay can be alleviated. Unfortunately, the attached parent may not be the best choice for the newcomer since it searches only a small part of the tree.

Through periodic message exchanges, neighbors' states are updated at each node, which includes the root path information of its neighbors. Keeping the root path information can not only quicken recovery procedure, but also facilitate the recovery procedure. When a member leaves the group, it notifies its parent and children. Upon receiving the leaving notification, each child looks for a new parent from its root path.

For any tree-based multicast protocol, two issues are noteworthy: 1) loop problem; 2) triangle optimization. Instead of applying loop avoidance, HMTP uses a reactive approach called loop detection and resolution. Once a loop is found, the member within the loop stops passing its root path to its downstream nodes. Afterwards, it breaks the loop by leaving its current parent, and rejoins the tree from the root. Obviously, such a mechanism requires a long recovery time since the rejoining procedure starts from the root.

Another issue in tree-based multicast protocols is triangle optimization problem. HTMP develops a heuristic to handle this issue. Figure 2.2 depicts how HTMP can solve the triangle problem. In this example, node X is assumed as a newcomer and it has found the nearest node, say node B, based on a lookup operation. Without performing the triangle optimization, X will attach to node B, as shown in (b). However, the partial tree in Figure 2.2-(b) is not optimized as node X locates between node A and node B, which is also known as triangle problem. Then, HMTP addresses this problem by informing node X the distance between node B and node A, $d(A, B)$. Based on this information, X will try to attach to node A if $d(A, X)$ is smaller than $d(A, B)$. After X attaches to node A, node B may perform a refinement depending on the available out-degree of node A, as

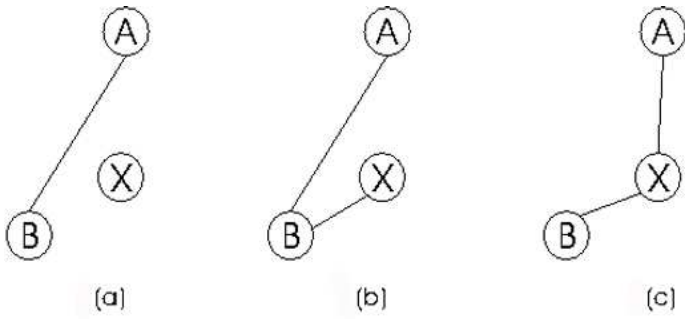

Figure 2.2: Triangle Optimization in HMTP.

shown in Figure 2.2-(c).

Overall, HTMP aims to be a simple and scalable overlay multicast routing protocol. However, it only takes e2e delay as the performance metric to build a shared distribution tree and therefore it cannot support high-bandwidth media distribution services. Furthermore, there is no considerations on heterogeneous capabilities of nodes.

HostCast

HostCast [47] is a typical example of refinement-based protocol, which is very similar to HMTP except for explicit mesh construction and additional bandwidth considerations for multimedia applications. It is designed for supporting single-source delay sensitive applications like media streaming.

Figure 2.3 shows an example of HostCast-aware data delivery tree (Figure 2.3-A) and its corresponding control plane (Figure 2.3-B) respectively. Meanwhile, the solid links represent as primary root paths and dotted lines connect the group members with their backup parents. In order to achieve a quick convergence and a good recovery capability, some classified relationships (e.g. potential parents) are defined among group members.

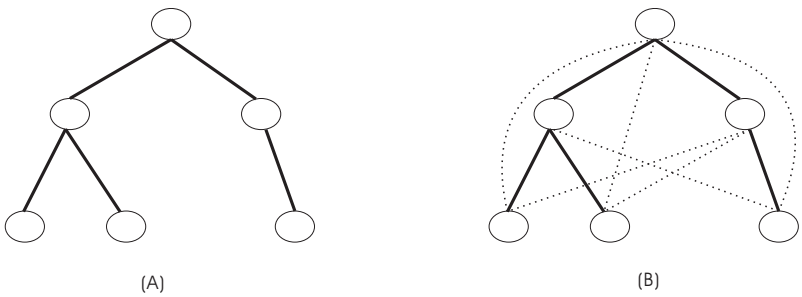

Figure 2.3: Data Delivery Tree and Control Plane in HostCast.

Initially, a new member sends a *Join Request* message to its potential parent, usually the root.

The potential parent replies with its current children's addresses and an indication of acceptance or rejection. If the request fails, this potential parent will be stored in a list called non-potential parent list. The non-potential nodes are sorted according to their e2e delay to the new joining member.

Like in HMTP each HostCast member keeps a member list of its primary root path to avoid loops in the data delivery tree. Each time, the potential parent needs to validate whether the requestor is already located in its primary path or not. Besides using the primary root path, HostCast defines a secondary root path to maintain additional neighboring information.

The outstanding feature of HostCast is deploying a simple method to obtain the knowledge of the underlying topology, which can be used to meliorate the quality of established multicast tree. To gradually find better root paths while keeping the scalability, HostCast uses path estimation method that the root (multicast source) periodically generates a small fixed-size probe packets to its children along the mesh. If a node receives a probe packet from its primary parent, it duplicates the packet and forwards the probe to its own children. Each probe packet carries a timestamp to estimate the delay and a weighted measurement to detect the available bandwidth along the root path. However, since the obtained knowledge is coarse and usually out-of-date in a highly dynamic scenario, the multicast tree is far from optimal.

The second feature of HostCast is relying on two refinement mechanisms to gradually improve the system performance: 1) switching primary parent; 2) substituting primary parent. An end-to-end delay measurement is employed to determine which type of the refinement should be chosen.

For data delivery, HostCast sets up an *overlay routing table* at the application layer of each group member. Whenever a member receives multicast packets, it duplicates and forwards the packets to its children in the data delivery tree. If the maximum number of children which a node can handle is k, the maximum number of peering relationships (primary and secondary), which a node has to maintain (i.e., control overhead) is $k^2 + k$. In reality, the control overhead is quite high even for a small group.

Overlay Multicast Protocols

The reason why application layer multicast is less efficient is that the overlay topology is always "randomly" connected without carefully considering the underlying network topology. A possible solution will be accurately matching the underlying network topology. Nevertheless, we have to consider that gathering more accurate topological information comes with a high cost. To balance the tradeoff between the efficiency of multicast tree construction and estimation costs, topology probing techniques have been introduced into the multicast community. Several techniques are designed to collect the underlying information, for example, using network tomograph [48].

Alternatively, an "overlay backbone" infrastructure is proposed to implicitly gain the knowledge about the network topology. Figure 2.4 shows an example of an overlay multicast framework. Some explicitly deployed intermediate proxies or entities form the overlay backbone to allow

more efficient membership and multicast tree management. The basic concept behind the overlay multicast approaches has been introduced in Chapter 1. In the following sections, we will present three representative overlay multicast protocols.

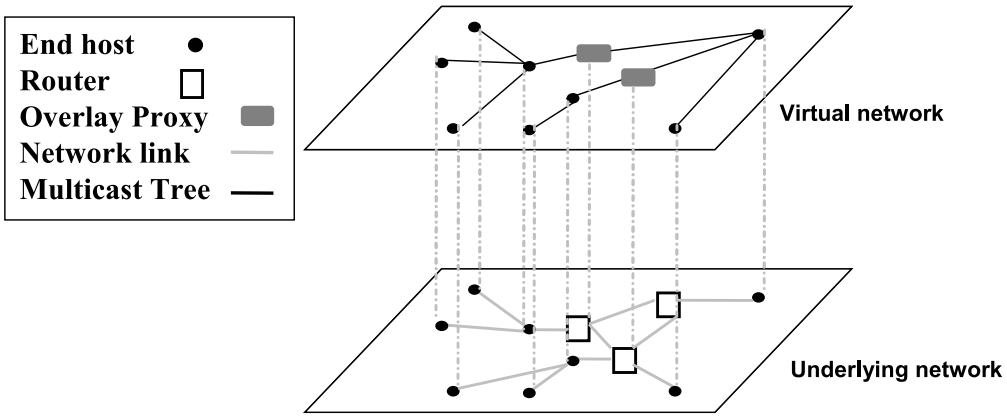

Figure 2.4: High Level Overview of Overlay Multicast.

Overcast

Overcast [49] targets at constructing an bandwidth-efficient distribution tree without knowing precise topology information of the underlying substrate network. It mainly depends on an overlay network to perform the multicast functionality, which consists of a cluster of nodes strategically placed in an existing network fabric. Towards fast convergence, Overcast uses a simple protocol to track the global status of the dynamic distribution tree.

The goal of Overcast's tree algorithm is to maximize bandwidth to the root for all nodes. Different from latency oriented optimization in aforementioned ALM approaches, Overcast tries to place the newcomer as far from the root as possible. In Overcast, the root is responsible for storing the statistical information of each node and updating the information by periodic update messages. Each node periodically contacts its relatives (i.e. *sibling, parent, and grandparent*) in the tree. To avoid loops, each node keeps an ancestor list. If the child fails to contact its parent within a limited time, the parent will assume the child as a "dead" node.

Overcast utilizes a "up/down" protocol to keep track of which nodes are upstream nodes and which nodes are downstream nodes. To limit the control overhead, each node only checks its direct parent node. When a failure occurs, it is, however, not easy for OverCast member to find a new parent. Even by maintaining the whole path from the source to a certain node, the node cannot quickly find a new parent since the capacities of potential parents may have been already occupied.

The frequent message exchanges used for tracking the status of the relatives lead to a high overhead, and therefore Overcast cannot support large-scale services. Besides, its deployability is limited by the availability of the deployed cluster nodes at the network fabric.

OMNI

OMNI [37] is one of the first overlay multicast infrastructures proposed for enabling real-time applications, and thus it puts most concerns about the overlay core construction. The OMNI infrastructure deploys a set of Multicast Service Nodes (MSNs) within a network in order to efficiently support real-time media streaming applications. The OMNI scheme formulates the overlay network construction as optimizing a latency degree-bounded spanning tree problem. They also identified using a linear integer-programming formulation to solve it.

During initialization, each new MSN starts to join the OMNI from the root by firstly measuring the unicast latency between itself and the root MSN. To be adaptive to dynamic network changes, OMNI presents some relevant refinements required for overlay multicast maintenance, including local transformations and probabilistic transformations. To our best knowledge, OMNI is the first proposal which describes adaptive operations in such detailed way, i.e. same layer transfer, different layer swap, despite that some fundamental refinements have been proposed in [45], [47].

Although they claimed the reason why they only consider the overlay latency between the root MSN and each MSN, it is not convincible to rely on such an overlay structure to support high bandwidth media distribution services. Furthermore, the tree structure may not be the best choice for streaming media services since the bandwidth is limited from upstreaming nodes to downstream nodes [50].

TOMA

TOMA [39] is mainly derived from Bi-dirEctional Aggregated Multicast (BEAM) [51] which is used to improve the state scalability of IP multicast in backbone domains. In TOMA, multiple multicast groups can be aggregated at the incoming edge routers in order to share a single distribution tree and de-aggregated at the outgoing edge routers. By doing this, these core routers are fully utilized through establishing a per-aggregated tree instead of a per-group tree. This scheme can not only significantly save network resources but also reduce the cost of building multicast trees for each service.

They proposed a Two-tier Overlay Multicast Architecture (TOMA) to provide scalable, efficient and practical multicast support for multiple group communication applications. In the TOMA architecture, Multicast Service Overlay Network (MSON) is advocated as the backbone service domain, where some overlay proxies are strategically placed to form an overlay network. Within this overlay network, the aggregated multicast approach is adopted to enlarge the resource utility. The main contribution is that TOMA takes advantage of shared overlay domain to support multiple groups instead of a single group. Obviously, there is a trade-off between bandwidth waste and efficient aggregation: the more bandwidth scarifies, the more groups share one tree and thus better aggregation can be achieved. In access network, it simply uses core-based approach to construct peer-to-peer multicast tree.

For group management, an overlay aggregated multicast protocol, OLAMP, is manipulated among overlay proxies. Several control messages are defined for MSON management. Actually, TOMA only concentrates itself on overlay backbone construction and maintenance, like in OMNI. Each member proxy contacts with host proxy when it decides to relay a *join* request for a group. After

performing the group-to-tree matching, the host proxy finds or computes an appropriate tree for the group and sends back to the requester. Similarly, the member proxy sends a *leave* message to the host proxy when there is no end users attached. After triggering the group-to-tree matching, the host proxy may remove the group-tree mapping or remove the whole tree if there is no other groups mapped onto it.

Once an end host sends a packet to the group, the other members of its local cluster including the member proxy will firstly receive the packet. Then, this local member proxy replicates and forwards the packet to other member proxies along the aggregated multicast tree. Thus, TOMA is not a source-specific multicast solution but can be initiated by any source.

The two-tier architecture is regarded as a suitable solution for application level multicast because the constructed multicast tree is more efficient based on the knowledge of underlying topology obtained by overlay proxies, and each local domain limits the control overhead of membership management. However, the overlay backbone is not built on-demand, which degrade performance over time even if the MSON can be correctly maintained. Additionally, TOMA takes little considerations on multicast approach in Access Network, besides the core-based P2P trees within each cluster. Unfortunately, the constructed multicast trees may unavoidably share the bottleneck links in the underlying network, i.e. the ”last-mile” access link at the Internet Service Provider (ISP).

Summary of Application Level Multicast

So far, we provided an overview of Application Level Multicast solutions by introducing some typical approaches. To summarize the above investigations, we list the features of ALM approaches as follows:

- Efficiency of data delivery

  In most cases, the data is delivered from one end host to another by IP unicast. Thus, redundant traffic and prolonged e2e latency are unavoidable. The efficiency of ALM can not compete with IP multicast. Even for the overlay multicast approaches, they can not achieve commeasurable performance as IP multicast because OM actually still depends on ALM (e.g., application layer multicast tree within each cluster).

- Easy implementation

  ALM and OM shift multicast functionality from core routers to end systems and sometimes overlay proxies, without relying on the IP multicast (e.g. IP multicast-enabled routers). Although they are not efficient as IP-based multicast, they can be easily deployed into the current Internet and provide reasonable application performance.

- Scarce knowledge of underlying topology

  Unlike network routers, end hosts have either limited or even no information about the underlying topology. However, it is the key challenge for overlay construction. OM tries

to address it by using overlay proxies to obtain the knowledge about underlying topology. However, this causes other problems such as overlay proxies misplacement, the high cost for overlay proxies management.

- Resilience

  End hosts are not comparatively stable as routers such that it is important to find an effective mechanism to recovery from failures. When we consider failure recovery mechanism, it is even harder to be performed in ALM approaches. As two completely uncorrelated overlay links may traverse several times through the same underlying links, it is difficult to select the backup paths.

- Scalability

  For many applications, multicast is a suitable solution for one-to-many or many-to-many data distribution model. But for some others applications, i.e. real-time news or stock tickets, a solution that scales to large groups is necessary. Although some end system multicast solutions are proposed to overcome the scalable issue, more extensive studies as well as real tests are needed to better understand their properties.

- Capacity constraints

  Compared with routers, end hosts have limited processing power and available bandwidth, which constraints the branching degree in the delivery structure. In addition, the heterogeneity of end hosts make this problem more complex.

- Adaptivity

  In application level multicast sessions, each end host may join or leave the group at will. However, it does not happen in IP multicast because the non-leaf nodes in the delivery tree are routers which do not leave the multicast tree frequent or ungracefully. The dynamic changes may have a great impact on the overlay construction (e.g., a part of the overlay may be partitioned from the entire hierarchy). Therefore, in ALM another challenge is to be adaptive to network condition changes.

Generally, ALM and OM approaches take advantage of overlay networking techniques to address the deployment issues of network layer multicast. Due to its efficiency and incremental deployability, overlay multicast is considered as one of the most promising techniques to support media applications, providing information dissemination across different administrative boundaries (e.g. AS) and various platform for heterogeneous dynamic user groups [52]. Furthermore, we provide four tables in Appendix A to summarize above studies on application level multicast.

2.4.4 Peer-to-Peer Content Distribution

Content distribution on the Internet uses many different service architecture, ranging from centralized server-client mode to fully distributed peer-to-peer mode. The recent widespread use of peer-

to-peer applications such as Gnutella [53], peer-to-peer (P2P) IPTV [54] indicate that peer-to-peer content distribution systems can provide more resilience and higher availability than server-client media distribution systems.

During the last few years, IPTV has been gaining a tremendous popularity, identified by the increasing number of operators that provide media distribution services to residential users. Most of the traditional IPTV systems rely on Content Delivery Network (CDN) or deploying a set of local streaming proxies in every service domain. Relying on CDN networks (i.e., Akamai), Youtube [55] is the market leader in online sharing videos over the Internet. While these systems offer a means for media delivery and streaming, as identified in Section 2.4.1 they also pose a significant performance challenge in terms of scalability and service delay as the number of clients increases. For instance, Saxena *et al.* [56] showed that YouTube might have very high service delays for most popular services since it always retrieves video content at a constant rate at any stages. To solve these issues, P2P technologies have been applied to support IPTV systems, namely, P2P IPTV systems. These systems capitalize receiver's bandwidth to provide services to other recipients and only rely on IP unicast.

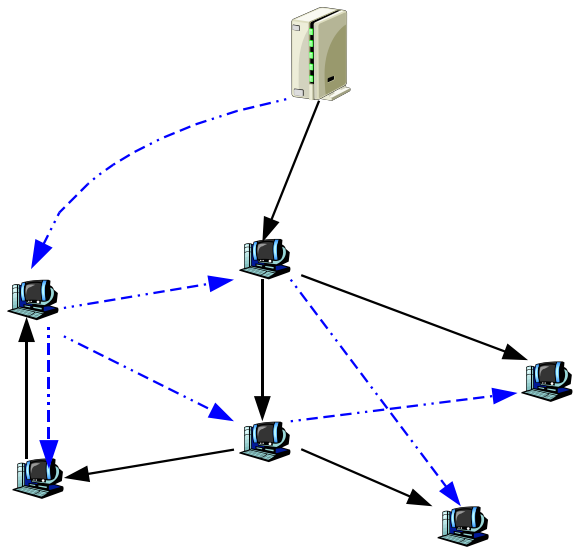

Figure 2.5: Example of Peer-to-Peer Media Distribution System.

Nowadays, the capacity of arbitrary peer (e.g. client's computer) can be comparable to that of a server in terms of processing power, memory size and storage size, that is, a peer can act both as a server and as a client. As shown in Figure 2.5, end hosts act as "server", "client" and "forwarder". Each end host is called a peer and data is replicated at each peer and forwarded to other peers. Different from general P2P file sharing, P2P media distribution system poses more stringent resource requirements for real-time media transmission. A straightforward way of building such a system is to use application level multicast which has been presented in Section 2.4.3.

The main difference between a P2P media distribution system and application level multicast-based media distribution system is peer management. In the former type of system, peers are organized in a more flexible and self-disciplined way, not necessarily a dedicated structure. In ALM systems, peers need to maintain a kind of order or structure (as described in Section 2.4.3) to help transmitting media data among users.

Peer-to-Peer Network Structure

The peer-to-peer network can be categorized into three classes according to the hierarchy structure: 1) unstructured; 2) structured ; 3) hybrid infrastructure.

- *Unstructured:* The placement of video content is completely unrelated to the overlay topology. In an unstructured P2P network, video files need to be located. Searching mechanisms vary from straightforward force methods, such as flooding the network with propagating queries in a depth-first manner till the file is located, to more sophisticated and resource-saving strategies that use random walks and routing indices [57], [58]. Here, routing indices refer to the tables of information about other nodes, which typically provide a list of neighbors that are most likely to be "in the direction" of the video content according to the query.

  Unstructured systems are generally more suitable for highly-dynamic node populations. The representative examples include Napster, Publius [59], Gnutella [53], Kazaa [60], Edutella [61], as well as others. The searching algorithms deployed in these unstructured systems imply the availability, scalability and persistence issues.

- *Structured:* Structured networks have emerged in an attempt to address the scalability problems that unstructured systems were faced with. In structured networks, the overlay topology is strictly controlled and video files are placed at precisely specified locations. Through a mapping between the video content and the location (e.g. node address), queries can be efficiently delivered to the specific nodes with desired video. However, a fundamental weakness of structured systems is that it is difficult to maintain such a structure in case a large number of transient nodes join or leave the system.

  Typical examples of structured systems are Chord [62], CAN [63], PAST [64], and Tapestry [65].

- *Hybrid:* Another category of P2P networks between structured and unstructured are referred as hybrid networks. Although the locations of video content are not completely specified, they may be impacted by routing requirements. A example of such a hybrid network is Freenet [66] in which files are identified by unique binary keys. These keys are generated based on a hash function on a short description of the original owner. Its main focus is taken on security, publisher anonymity, deniability, and data replication for availability.

Peer-to-Peer Content Distribution Systems

In this section, we introduce some selected P2P content distribution systems which are the representatives of aforementioned structures.

Gnutella

Gnutella is a typical example of decentralized unstructured P2P network. Like most peer-to-peer systems, Gnutella builds a virtual overlay network with its own routing mechanism, allowing users to share content with other peers. There is no centralized coordination in the network and users connect directly with each other through the software application that performs both as client and server.

Gnutella relies on an application level protocol to support the communication between servants [67]. It provides four types of messages to manage the group:

Ping A request for an arbitrary host to announce itself.

Pong A reply to the Ping request, which contains the responder's IP and port, and the number and size of the shared files.

Query A lookup request which includes a search string and the speed requirements of the responder.

Query Hits A response to the Query message. It carries the responder's IP, port and speed, the number of matching files, and the indexed results set.

The original Gnutella architecture uses a "flooding" mechanism to deliver *Ping* and *Query* messages. To limit the overspread of the network, each message header consists of a Time-To-Live (TTL) field. Once the value of this field reaches zero, the message is dropped. The scalability issues arose from the fact that using TTL largely limits the area where the message could reach. In order to locate a file in an unstructured system such as Gnutella, nondeterministic searches are the only option since the peers have no way of finding files by guess or random selection.

Chord

Chord is actually a P2P routing and lookup infrastructure that performs a mapping between the file identifiers to node identifiers. Content locations can be implemented on top of Chord by identifying files (*data items*) with keys and storing such pair (*keys*, *data item*) at the node that the keys map to. The keys are assigned to both files and nodes with a deterministic function [62].

The network in Chord can be considered as a ring or a circle. All node Identifier (ID)s are ordered in the "identifier circle" modulo 2^m where m is the key length. Each node is responsible for the key k if its identifier is equal to, or follows k. The node is called the successor node of key k. For example, Figure 2.6 shows that a Chord identifier circle contains three nodes, namely, 0, 1 and 4, where $m = 3$. According to the above definition, key 1 is located at node 1, key 2 is assigned to

node 4, and key 7 at node 0. Through the deterministic function, each node is only required to know its successor on the circle. In this example, queries for key 2 are passed around the circle via its successor node 4.

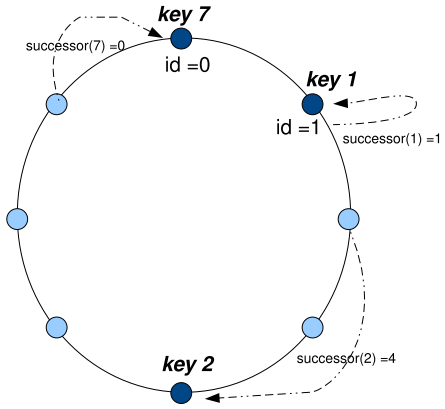

Figure 2.6: Example of Chord Identifier Circle.

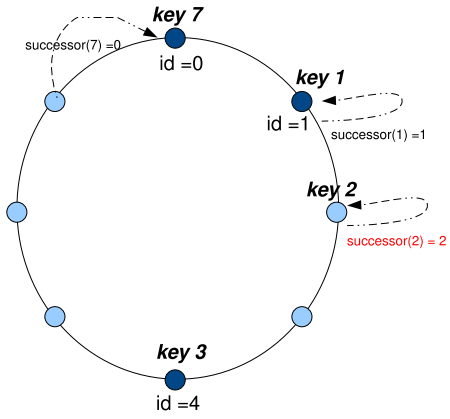

Figure 2.7: Example of Node Joining in Chord.

Chord also handles nodes joining and leaving. When a new node *n* joins the network, corresponding keys previously assigned to *n*'s successor will be assigned to *n*. Suppose node 2 joins the above Chord circle. As depicted in Figure 2.7, node 4 is no longer responsible for key 2 but instead node 2 is assigned with key 2.

Since only one data element per node is guaranteed for the correct routing of queries, the performance may degrade dramatically when routing information is out-of-date or highly-dynamic changes due to nodes joining or leaving the system. To increase the efficiency of lookup algorithm, "finger table" is proposed to maintain additional routing information. In the finger table,

each entry i points to the successor of node $n+2i$. If a node n queries a lookup for key k, finger table searches for the highest node n_h whose ID is between n and k. If such a node exists, the lookup starts again from node n_h. Otherwise, the successor of node n is returned. The time required to complete lookups are $O(logN)$ for N node system in a steady stage.

Freenet

In fact, Freenet is a loosely structured P2P network, which used file identifier and node identifier to generate an estimation of where the content may be located. It uses *chain mode propagation* approach to forward queries among nodes [66].

Each Freenet node maintains its own local database, which is available for the entire network to read and write, and a dynamic routing table containing the addresses of other nodes and the files to be shared among other nodes. To search for a particular file, the user sends a request specifying the *key* and TTL value (similar to the value defined in Gnutella).

Newcomers join the Freenet work by first discovering the one or more existing nodes, and then sends *Data Insert* message in order to insert new files to the network. Any node receiving such a request, checks whether the key already exists. Otherwise, the node looks for the closest key in terms of lexicographic location in its routing table, and forwards the request to the corresponding node. Thus, newly inserted files are placed at nodes with similar keys.

For such a purely decentralized content distribution system, how to obtain a key associated with a specific key remains open. Besides, how to handle a large volume of indirect files is unresolved.

Summary of P2P Media Distribution Systems

P2P systems capitalize receiver's bandwidth to provide services to other recipients as depicted above. However, available bandwidth or processing capacity of the peers is still an issue, which differs a client from a media server. To build an efficient P2P media distribution system, the following aspects are essential to be considered:

- Peer management and distribution hierarchy construction: each requester must find supplying peers in order to get relatively better Quality of Service (QoS).
- Heterogeneous capacity: peers may have limited bandwidth capacity or are unwilling to contribute resources. A recent study [53] revealed that over 85% of the peers do not share any files. To encourage capable peers to contribute more to the network, several incentive mechanisms [68], [69], [70] have been recently discussed in the research committee.
- Dynamic adaptation: peers may join or leave the media session at any time, which is known as the transient nature of peers [71].

2.5 Networking Considerations

2.5.1 Media QoS Control

There have been numerous efforts to provide QoS over the Internet since most of the multimedia applications are inherently QoS-sensitive. For example, rate control and congestion control are desirable for video streaming applications. Congestion control commonly relies on rate control by adapting the sending rate to the available bandwidth of the network. We briefly introduce some major approaches for congestion control and rate control.

Congestion Control

Congestion control is mainly used to prevent packet loss and reduce e2e delays. Most of the current congestion control methods rely on end-to-end mode by using TCP to regulate the rate of video stream to the available bandwidth. Once the network is detected as congested, congestion control mechanisms at the sender side will reduce the sending rate. Additionally, Explicit Congestion Notification (ECN) [72] provides a congestion indication method for incipient congestion. Upon receiving an indication from the network, rate control mechanisms can be applied to regulate the rate of video stream to the available bandwidth. The rate control mechanisms can be classified into three categories: endpoint-based, hybrid rate control and path aggregation.

For the endpoint-based congestion control [73], [74], the source is responsible for adapting the video transmission rate for each video session. All sessions need feedback information about the present network status. Based upon the feedback, the sender respectively regulates the rate of video stream.

For the hybrid rate control, both the sender and receivers sides regulate the rate of videos streams simultaneously. A typical example is a layered multicast scheme [75], in which it adjusts the video bit rate to the available bandwidth at the receiver side. Note that reducing the bit-rate of encoded video to avoid the congestion will consequently cause the reduction of the visual quality. This issue has been extensively studied, for example by Lam et al. [76]; and by Cuetos and Ross [77].

Previous research works have shown that path aggregation [78], [79] may overcome the bandwidth deficiency by an efficient multiplexing and selecting low latency paths fitting into user's requirements. Besides, it can reduce performance degradation due to high path latencies and loss rates. If a Forward Error Control (FEC) [80] strategy can be applied to decouple the transmission of error correction frames from the associated data, it can provide protection against correlated losses.

Fragmentation Issues and Error Control

Applying congestion control may alleviate the impacts (e.g. due to network congestion) on the video quality, other changes of network conditions can still result in the diminished video stream

quality: server or router failures; packets arriving out of order; packet fragmentation. Particularly, the packet fragmentation has been considered as one of the most influential factors. The reason for media fragmentation is that network cannot support more than $1,480$ byte packets while generally, media encoder produced packets of $1,000-4,000$ bytes. So network has to break encoded packets into smaller fragments to be fit for the network requirement. If one of the fragmentations is dropped, the original datagram may have to be fragmented again and retransmitted. This not only prelongs the service delay but wastes the network resources.

Recent works on addressing the fragmentation issue focus on reactive mode which provide a compensation to correct the errors. There are four methods used to recover from video errors: 1) Link-layer error control; 2) retransmission-only error control;; 3) error concealment; 4) error-resilient video coding.

- Link-layer error control includes FEC and Automatic Repeat Request (ARQ) [81]. FEC adds redundant information into the messages, also known as an error correction code. The original message can be reconstructed by using the redundant code when errors occur. This allows the receiver to detect and correct errors without asking the sender for additional data. Differently, in ARQ the receiver notifies the source only when the packets are corrupted and needed to be retransmitted. It mainly relies on acknowledgements and timeouts to achieve the reliable data transmission.

  A variation of ARQ is Hybrid ARQ (HARQ) associated with both *FEC* and *ARQ* to get a better performance, particularly over wireless channels. For example, type-II HARQ sets a constraint on the maximum number of retransmissions for a packet [82], [83], which is pre-defined and fixed. When the receiver detects the loss of packet N under the condition of $T_c + RTT + D_s < T_d(N)$, where T_c is the current time, D_s is a slack term, and $T_d(N)$ is the time when packet N is scheduled for playback, the receiver requests for a retransmission of package N from the sender.

- Retransmission-only error control is identified inappropriate for real-time multimedia applications due to its high latency [84]. However, a Buffer-controlled Retransmission-based Error Control (BREC) takes advantage of the motion prediction loop employed in most motion compensation-based codecs [84]. Such a method does not require any artificial extension of control time and play-out delays, and thus can be selected for supporting interactive media applications.

- Error concealment is proposed by hiding errors from human perception. The key idea behind the proposal is to use redundant error bits with "high-priority" in terms of loss rate and importance to flip the "chosen" error bits. Therefore, it is very useful for one-way systems like video broadcasting which requires no feedback and retransmission of media streams [85], [5].

- Adopting Fine Granularity Scalable (FGS) in MPEG-4 enables a layered and fine granularity scalable bitstream with difference importance at different layers (e.g. base layer,

enhancement layer). However, it causes several problems when delivering enhancement layers in FGS bitstream over an error-prone channel since the enhancement layer can only provide weak error detection capability. Therefore, MPEG-4 includes some error resilience techniques to enable a robust transmission of compressed video over noisy communication channels. An overview of video error resilience techniques is presented in [86], in which error resilience techniques are classified into four categories: 1) encoder based techniques; 2) decoder based techniques; 3) interactive based techniques; 4) proxy based techniques. Ge, Peng et. al. [87] compares different error resilience algorithms for video multicasting on Wireless LANs.

2.5.2 The Protocol Stack for Video Distribution

So far, we introduced the video processing mechanisms, the media service requirements at the server side, as well as the architectural considerations on how to support video distribution. In this section, we present protocols designed for communication between clients and streaming servers. According to different functionalities, we focus on three layers: network layer, transport layer and session layer, as shown in Figure 2.8. Network layer protocols mainly provide a basic network service support. Transport layer protocols are the main communicator between endpoints. Session control protocols are defined to control the delivery of media streams within an established video session.

Network Layer

In the network layer, IP is chosen as the main network layer protocol for video streaming. For the multicast purpose, Internet Group Management Protocol (IGMP) is used between end hosts and their inter-mediate multicast agents like routers to support the management of groups (e.g. creation of a transient multicast group, periodic updating of group membership, addition or deletion of group members). Since the traditional Internet lacks QoS support for the multimedia transmission, a user can depend on Resource Reservation Protocol (RSVP) [88] to request a specific QoS guarantees from the network. Then, the RSVP protocol carries the request through the network, and traverses each node which the network uses to carry the media stream. At each node, RSVP tries to make a resource reservation for the upcoming video stream based on local admission control and policy control.

Transport Layer

Considering the transport layer, Real Time Protocol (RTP) [89] is a transport mechanism often used for real-time media data. It includes two main components: RTP and Real Time Control Protocol (RTCP). RTP supports the media synchronization mechanism by providing *time-stamping, source identification, participant identification and corresponding RTP timestamps*. To ensure

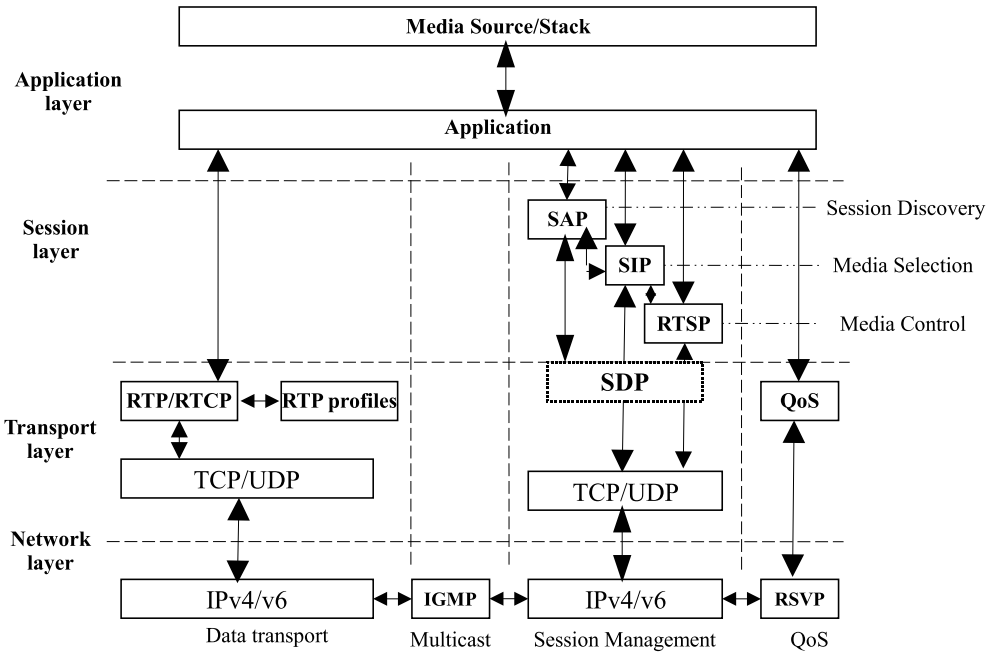

Figure 2.8: Protocol Stacks for Video Streaming

playback successfully, RTP employs sequence numbering to place incoming packets in order. RTCP mainly offers control-related functions to the media source such as congestion control, QoS feedback to an application. Based on the feedback, the sender and the receiver can adjust the transmission rate, and determine the current network status (e.g. local network congestion); re-evaluate the network performance of media distribution.

Session Control Layer

Real-Time Streaming Protocol (RTSP) [90] provides session control for media streaming services. The main function of RTSP is to support VCR-like operations, such as fast forward, rewind, step backward. In the session layer, Session Initiation Protocol (SIP) [91] is used to create, modify and terminate media sessions. SIP relies on Session Description Protocol (SDP) to transmit signaling messages. Moreover, Session Announcement Protocol (SAP) [92] is created to assist establishing multicast sessions, for instance, to carry the relevant session setup information to prospective participants. SAP is always in conjunction with SIP and RTSP protocols. During a multicast streaming of a video file, SAP periodically multicasts video packets containing a description of this session. At the same time, remote participants in the other session directories can use the received description as a start tool to join the session. Thus, SAP can be regarded as a guidance

to the media session since multicast services can be dispersed to broader areas and attract more participants.

2.6 Security Considerations

The last and very important issue in any media distribution system is security. Several research works have been studied in media distribution algorithms and architectural mechanisms, however, very few efforts has been focused on security issues. In general, three security properties are necessary for a deployable video streaming system over the Internet: authentication, authorization, and video confidentiality.

2.6.1 Authentication

Authentication is the fundamental requirement for any services through which source identity, user identity, and integrity of video streams during delivery can be identified. The server needs to authenticate whether the user is exactly a valid receiver and the user needs to ensure the server is a legitimate service provider. To validate the both sides, digital certificates or digital signatures are commonly used.Unfortunately, most of the current approaches focus on one-way authentication (i.e., server to client) using the third party authority. Moreover, it is also important to consider how to protect the integrity of video streams. Otherwise, content pollution attack can be devastating as polluted video may spread through the video distribution system if unsuspecting users download the polluted video into their sharing folder, from which other users may then download it again [93].

2.6.2 Authorization

Even though the users and the server are mutually authenticated, there are still a series of gaps left. For example, how to verify the exact services between the service provider and the users who are now allowed to access the video stream? Which kind of video file is accessible by the arbitrary user? How to charge the users if they request for the service?

To fill in these gaps, authorization is usually required after the authentication phase. In a typical authorization model, the type of services, the duration time of services and the accounting type are elementary components. In addition, it often relies on cryptography that is a process of converting ordinary information (e.g. plaintext) into unintelligible gibberish (i.e., cipher). The detailed operation of a cipher is controlled both by the algorithm and, in each instance, by a key. Based on the cryptography, authorization allows the server and users exchange encrypted keys under a protected tunnel. To strengthen the communication between the server and users, some conditional access systems can be employed, for example, with embedded watermarks in the video stream. These

watermarks provide the complementary authorization even if the user has the permission from the initial access model.

2.6.3 Video Confidentiality

To keep the confidentiality of a video is a critical security aspect for most video applications without this protection uncountable illegal video streams may be produced or replicated. It is also a hard problem since many systems can only prevent "casual" copying via either a serial number or time stamps. For media streaming applications, confidentiality-related elements include the identifiers of clients, the identification of protected video streams and an agreement for a legal video copy. Some protection mechanisms rely on watermarks and confidentiality information embedded in the video stream headers. A more complicated security model for media streaming systems allows the combined use of cryptography, digital signatures, video watermarking and personal information binding for securing digital video streams.

2.7 Overlay Multicast-based Video Distribution Architecture

So far, we identified the requirements for video distribution systems, and presented main challenges in supporting media distribution services, as well as provided the architectural considerations.

Taking all aforementioned statements into consideration, we propose a video streaming architecture shown in Figure 2.9, which involves both sides of a media server and clients. Here, overlay multicast and caching mechanisms are chosen to constitute the media distribution architecture as overlay multicast is a desirable solution because of its efficiency and deployability. The overlay proxies can store some video prefix or special part of video streams, by which clients can directly receive the required video without long-haul delays. The non-cached portions, if needed, can be retrieved from the media server. To be more efficient, we employ caches into the clients' side which can further improve performance in terms of reduced service latency.

In the context of this thesis, we are interested in providing an overlay multicast-based framework that can support a large-scale, efficient, and reliable video transmission. Since we are concentrated on protocol-oriented system design, the software and application specific aspects will not be further explored in the following thesis. These aspects include the media fundamental aspects and media QoS control. Instead, we assume existing approaches can be used to achieve the basic support, and they are not the main focuses of this thesis. For security issues, we intend to provide some rudimentary protection but more serious security considerations are not part of our primary tasks.

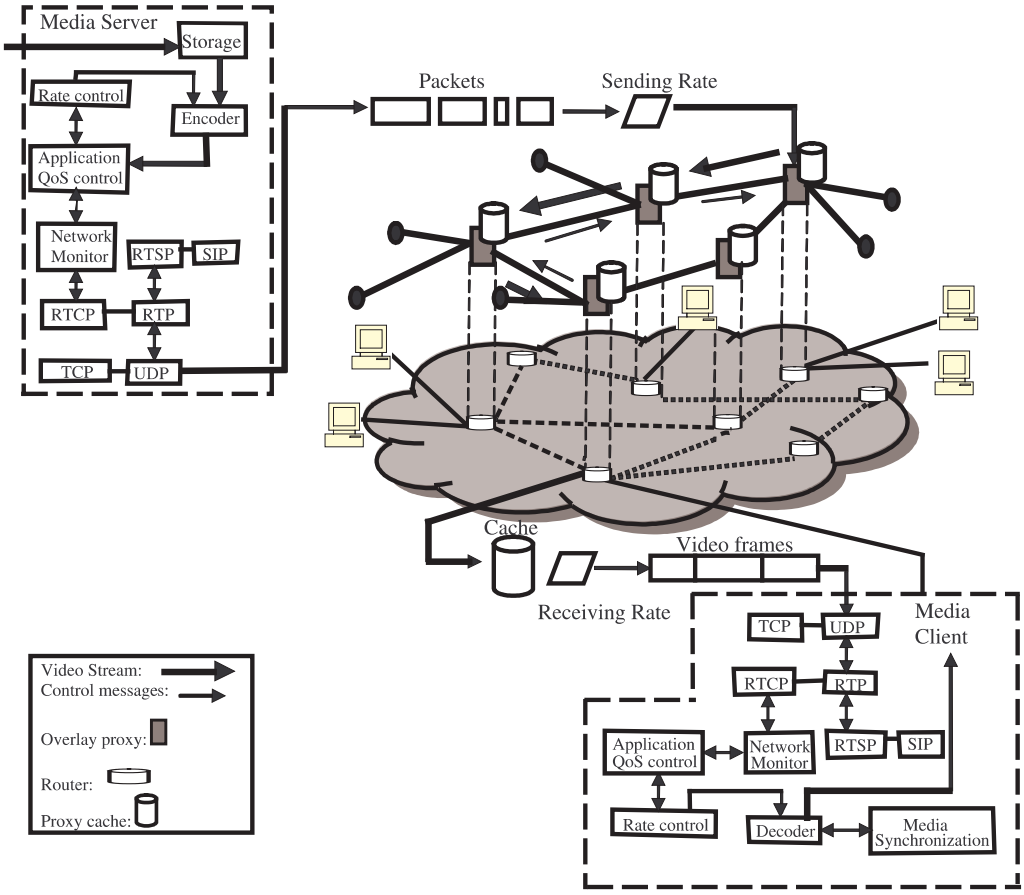

Figure 2.9: Overlay Multicast-based Media Distribution Architecture with Proxy Caching.

2.8 Summary

In this chapter, we first identified the video requirements for supporting media distribution system on the Internet. Then, we presented the architectural considerations in order to meet the aforementioned requirements. Meanwhile, we investigated Content Delivery Network, Network Layer Multicast, Application Level Multicast which includes application layer multicast and overlay multicast, and Peer-to-Peer Media Distribution approaches. Our investigations were comprised of the identification of existing challenges in each classified system, exploration of possible solutions, and a brief summary of these approaches with regards to their advantages and potential weaknesses. Further, we categorized the construction of overlay hierarchy into five types, which facilitated the understanding of application level multicast. Similarly, we classified P2P network structures into three categories, and presented a representative protocol for each category.

The above investigation provided insights towards building a scalable, efficient and reliable media

distribution system. Finally, we proposed an overlay multicast-based video distribution system, in combination with some caching mechanisms to fulfill the above requirements as well as to overcome the above identified challenges.

In the rest of the thesis, we focus on providing an efficient and cost-effective solution for large-scale media distribution services. Keeping the above two-tier architecture (in Figure 2.9) in mind facilitates the understanding of our proposed framework in Chapter 3. However, instead of deploying some dedicated proxies the new framework completely relies on end hosts to self-organize into an overlay hierarchy. However, the proposed new framework can be easily extended with deloying some infrastructure nodes (e.g., proxies).

Chapter 3

A Dynamic Mesh-based Overlay Multicast Protocol (DMMP) Framework

3.1 Introduction

Multicast has emerged as an efficient mechanism for supporting group communications, such as video and audio conferencing, multi-party games and content distribution. Unfortunately, applications of network layer multicast (IP multicast) for worldwide media streaming services remain limited due to its technical and deployment issues [34], [35]. To solve these issues, various non-network layer multicast solutions have been proposed.

Aforementioned application layer multicast and overlay multicast approaches take advantage of overlay networking techniques to address the deployment problems of IP multicast. Among them, overlay multicast is considered as the most promising technologies to support media distribution applications, with its ability to disseminate information across different administrative boundaries and various platforms for heterogeneous, dynamic groups of users [94]. Additionally, the explosive growth of multimedia services and applications over the Internet necessitates streaming media to a large popularity of users. However, with the current technology, it's hard to develop a comprehensive media distribution system due to the following two challenges [1].

First, the total number of concurrent clients the system can support is limited by the resources of the streaming supplier. Second, current media streaming proposals usually have limitations in reliability and scalability. The reliability concern arises from that only one entity is responsible for all clients. The scalability issue is resulted from the fact that adding internet-scale potential users requires the commensurate amount of resources to the supplying server. Meanwhile, aforementioned proposals could not explicitly support media distribution applications (e.g. media

streaming) in a large scale.

Motivated by the studies in Chapter 2, we propose a new overlay multicast framework which manages a dynamic mesh-based overlay core and only involves participating end hosts without relying on the availability of delicately deployed infrastructure nodes, while providing certain degree of efficiency, reliability and resilience. We integrate the capacity classification and locality-awareness into the DMMP-aware overlay hierarchy, which makes the framework more scalable and efficient.

The remainder of this chapter is structured as follows. Section 3.2 summarizes the properties of the DMMP framework, which overcomes the aforementioned two issues. Section 3.3 gives a brief overview of the DMMP framework. Then, Section 3.4 specifies the messages used in the framework. The protocol details are explained in Section 3.5. Section 3.6 illustrates the criteria used to evaluate the capacity of the end host. Further, some considerations on security aspects are presented in Section 3.7. Lastly, we give a short summary in Section 3.8.

3.2 Properties of DMMP

The DMMP framework is organized into a two-tier overlay hierarchy and the mechanisms are introduced to dynamically manage and maintain the hierarchy. The key idea behind DMMP is to let a few end hosts selected and self-organized into an overlay mesh during the multicast initialization phrase and also when group member changes, and dynamically maintain such a mesh. Although routers may also be manually designated (e.g. by ISPs) to construct the mesh, this document initially discusses the approach via end hosts. Specifically, there are four design challenges to be addressed in DMMP:

1. DMMP considers the *heterogeneous properties* of group members by evaluating their available bandwidth during runtime. In this framework, high-capacity nodes which are able and willing to make more contributions to the network are expected to get better performance. This incentive-based mechanism may help maximizing the usage of available bandwidth for the entire overlay tree.

2. *Scalability* is one of the main problems to be solved in the multicast applications. In DMMP, each end-host may act as a potential server for other clients and the number of possible servers increases at the same rate as the end host clients. As Peer-to-Peer (P2P) technologies have been deployed to support various services over the Internet, it is possible that more end hosts resources are available in the network. Once a node joins the DMMP multicast session, additional resources are available to the whole system. Therefore, the DMMP system is scalable as it can potentially support a number of clients.

3. DMMP also considers the *serving quality of media* to end hosts (e.g. *end-to-end service delay*). When constructing the overlay multicast tree, high-capacity nodes are given a high

priority to stay at the higher level of the overlay tree. In return, this allows DMMP to generate the tree as short as possible and accordingly the overall delivery delay could be reduced.

4. DMMP specifically considers the transient nature of end hosts and attempts to prevent incapable or short-lived nodes from staying close to the center of the multicast tree. Consequently, the DMMP overlay structure is relatively *stable and resilient* to dynamic network changes. Those who frequently join or leave the multicast session are expelled from the core of the overlay. Thus, any failure of a single node may result in a transient instability in a small subset of participants, but it will not cause a catastrophe in the whole overlay framework.

Section 3.3.1 and 3.3.2 explain how above features can be achieved in DMMP. Moreover, we evaluate these features through both theoretical and simulation-based analysis in Chapter 4.

3.3 Framework Overview

Since the DMMP framework is proposed to support large-scale media distribution, it consists of two types of functionalities: control plane and data plane. The former is mainly used to manage the DMMP-aware overlay hierarchy, and the latter is designed for delivery of media data to the end hosts.

Although the tree-based overlay structure (cf. Section 2.4.3) is regarded as the most efficient approach for data distribution in a stable network, it is not effective for multimedia distribution under dynamic scenarios. The main reason is that a pure tree structure has difficulties to meet both high bandwidth and high reliability requirements. The first difficulty is caused by the structure as the delivered service quality to downstream hosts is limited by the minimum bandwidth among the upstream connections along the data delivery path from the source. For instance, a member in the OMNI [37] tree receives data only from its upstream node and the data reception rate of this member can not be greater than its upstream node. It is even more difficult in the core network where each Multicast Service Node (MSN) should be responsible for delivering a large amount of data to a whole cluster. Towards reliability, a pure tree is much less robust than a full mesh because a single node failure or a loop can easily partition the entire tree and disable the communications among members.

Different from above proposals, a dynamic DMMP-aware mesh as one of multiple forwarding solutions is introduced to increase the throughput of the whole overlay network. Such a mesh will allow the reception of packets from multiple nodes other than from the single upstream node.

Figure 3.1 presents an overview of the DMMP framework. Here, the control plane composes an overlay mesh and some core-based clusters. Data plane is, then, built on the top of the structured control plane. Meanwhile, the source entity and a set of super nodes form the overlay mesh, through which each super node supervises one cluster.

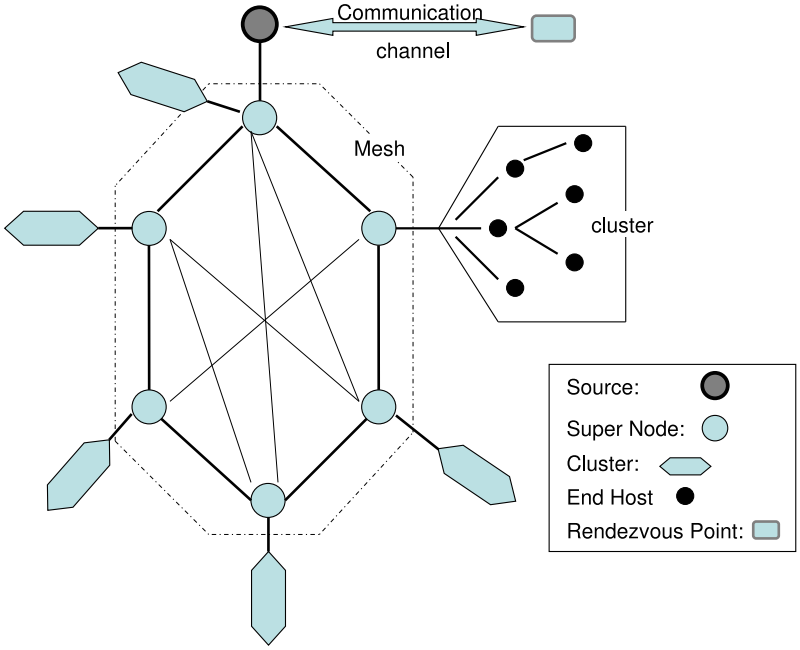

Figure 3.1: Overview of DMMP Framework.

3.3.1 Control Plane in DMMP

Before we explain how to configure the control plane in the DMMP framework, we identify the main reason why DMMP needs to consider the degree bound in streaming applications. Our design philosophy is motivated by the fact that in most media streaming systems available bandwidth resources possessed by a multicast group are insufficient during the runtime, which can be easily observed from the available bandwidth [95]. Therefore, we propose to use a combination of available bandwidth [96] and uptime to represent the capacity of each DMMP-aware host [97]. For example, upon an assumption that the bit rate of media is B and the outbound bandwidth of an end host i is $b(i)$, the total number of connections it can establish is $b(i)/B$ which is also the maximum degree of the end host. Moreover, the usage of available bandwidth in the overlay routing has become possible, based on recent advances in available bandwidth measurement techniques and tools [98], [99], [100]. Obviously, if an application has additional requirements on end-to-end delay or loss rate, these metrics can be jointly considered during the overlay hierarchy construction.

The construction of the control plane is composed by two parts: dynamic-mesh and local clusters. We first present the procedure how the mesh core is configured. For simplicity, the DMMP framework relies on Rendezvous Point (RP) to bootstrap new members.

- Step 1: After an initialization phase, the RP will calculate the out-degree of each end host and classify them into two categories: leaf nodes and non-leaf nodes. If one's out-degree is

less than two, the end host is sorted as leaf node because it can only receive data from the incoming connection.

- Step 2: The information of two-category nodes are respectively stored at the RP. Meanwhile, all non-leaf nodes are placed in the order of their out-degrees and the related information is reported to the source. On receiving the list of ordered non-leaf nodes, the source selects an application-specific number of them as super nodes. Those nodes are expected to have higher capacities as defined in Section 3.5.2, and are used to manage the multicast group. In the initialization stage, high capacity nodes refers to nodes with higher available bandwidth since the current uptime for all members is zero. To efficiently manage the group, the capacities of each super node are also stored at the source and the RP.

- Step 3: After being selected, the super nodes organize themselves into a mesh rooted at the source. The overlay mesh construction is mostly motivated from [101].

As one of the main features of DMMP, it considers the heterogeneous capacities of group members by evaluating their available bandwidth during runtime so that high-capacity nodes (i.e., super nodes) which are able and willing to make more contributions to the network are expected to get better performance. Based on selected super nodes, some core-based clusters will be formed to connect with the mesh nodes, namely, the super nodes.

- Step 1: After constructing the overlay mesh, the next step is to form core-based clusters. Each non-super node will firstly consult its local cache for super node candidates. If there is no suitable candidate, it queries the RP immediately. Then, the requester caches these newly received candidates, from which it selects the best one based on e2e latency measurements. If there are multiple super nodes which can provide similar e2e latency for the node, one of them with higher out-degree will be chosen.

- Step 2: Those non-super nodes sharing the same super node will form a local cluster. The cluster formation is initiated by the super node which answers for informing the RP and contacting the source. Generally, certain number (due to the super node's available bandwidth) of end hosts with higher capacity will be selected as its immediate children. This operation guarantees that the multicast tree within each cluster meets the bandwidth need of media streaming applications.

- Step 3: Afterwards, direct children of super nodes choose some nodes with higher capacities (i.e., out-degree, e2e latency) as their children. This selection method will expedite the convergence of the tree and alleviate the average latency.

The iteration will continue until all cluster members join the tree, and accordingly the control hierarchy is constructed for the multicast group . For the sake of resilience, each node in the local cluster should keep some information of its relatives in the local cache. In this chapter, these entities (i.e., parent, PLN, child, CLN and siblings, as seen in Section 1.2) are denoted as relatives.

3.3.2 Data Plane in DMMP

Compared with existing application level multicast approaches, DMMP is designed to be more stable, efficient and applicable to support large-scale groups without relying on predetermined intermediate nodes in the network and potentially get better performance. While the current DMMP framework is designed for single-source multicasting, it can be easily extended to support multi-source distribution using decentralized servers (e.g. CDN in Section 2.4.1).

In order to overcome the two challenges identified in Section 3.1, media streaming task in DMMP is accomplished through the following two phases: (1) an on-demand overlay core (or called mesh) is established to achieve the optimized performance; (2) based on the structured mesh, several clusters are formed to connect with selected mesh members, namely, super nodes.

- DMMP distributes the task of group management and data delivery to the super nodes (which construct the on-demand overlay mesh), which can alleviate the server bottleneck at both available serving bandwidth and management cost. Moreover, it alleviates the risk of one entity dependent reliability by distributing the overload to a set of decentralized nodes.
- Based on the structured mesh, several clusters are formed to connect with selected mesh members. DMMP applies the concept of locality-awareness (e.g. aggregated clusters) into the group management so that it can dramatically reduce the control overhead and complexity of the overlay maintenance.

Basically, a source-based DMMP architecture consists of a sender, several receivers, one or many Rendezvous Points and Domain Name Systems (DNSs). Typically, a respective data channel between two entities is established by exploiting the existing protocol stacks such as UDP/Internet Protocol (IP) or Transmission Control Protocol (TCP)/IP. The data channels utilize IP unicast according to the underlying IP transport scheme.

Figure 3.2 depicts an example of data delivery within a DMMP-aware cluster. In this example, data is firstly replicated into three copies, respectively delivered from the super node to its direct children 1.1, 1.2, and 1.3 using IP unicast. Similarly, node 1.1 replicates copies of data according to the number of their children (e.g. two copies), sending to node 1.1.1 and 1.1.2. Before long, node 1.1.1.1 receives the copy of data from its parent 1.1.1. In the next iteration, the receiver will similarly make copies and deliver to its children.

In addition, as required in real-time media streaming services a sequence of media packets should be transmitted with minimal communication delay and maximum bandwidth support. Therefore, DMMP tries to meet both requirements.

3.3.3 An Example of DMMP Overlay Hierarchy

Let us explain how to construct DMMP overlay hierarchy by a basic example. In Figure 3.3, a corresponding communication channel between the source and RP is built. Basically, a source-

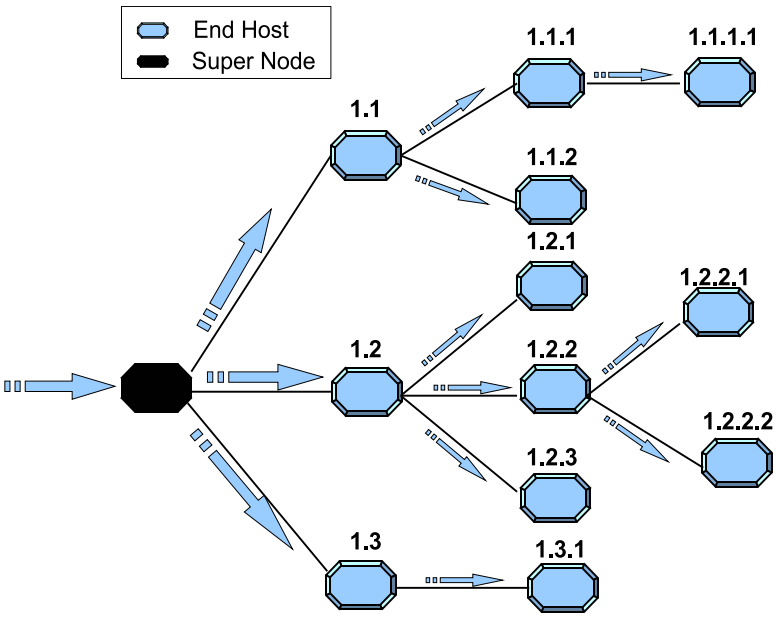

Figure 3.2: Example of Data Delivery in DMMP.

based DMMP framework consists of a sender, several receivers, one or many RPs and DNSs.

Assuming that it is the first time to construct the overlay hierarchy, then:

Step 1: When obtaining a list of group members from the RP, the source selects six end hosts as super nodes. In Figure 3.3, they are listed as A, B, C, D, E, and F. Those end hosts are actually used to manage the multicast group and relay data from the source to other receivers. This classification can alleviate the routing burden at the media source by using super nodes to perform data delivery to the end hosts.

To illustrate the super node selection mechanism in a simple way, we assume that the capacity of each host is linearly distributed. Thus, we assign the capacity of each end host c_i as follows:

$$c_i = \frac{b_i}{\sum_{i=1}^{N} b_i} + c \cdot t_i. \tag{3.1}$$

where, b_i is the available bandwidth of node i, N is the total number of group members and c is a constant. Moreover, t_i starts to calculate the time duration from a node joining in a multicast session to its leaving, or called uptime. Again for the same reason mentioned in Section 3.3.1, we consider the available bandwidth during the super node selection. Obviously, if an application has additional requirements on end-to-end delay or loss rate, those metrics could be jointly considered in expression (3.1) during the overlay hierarchy construction. The proposed equation is very flexible, which can be adjusted according to application-

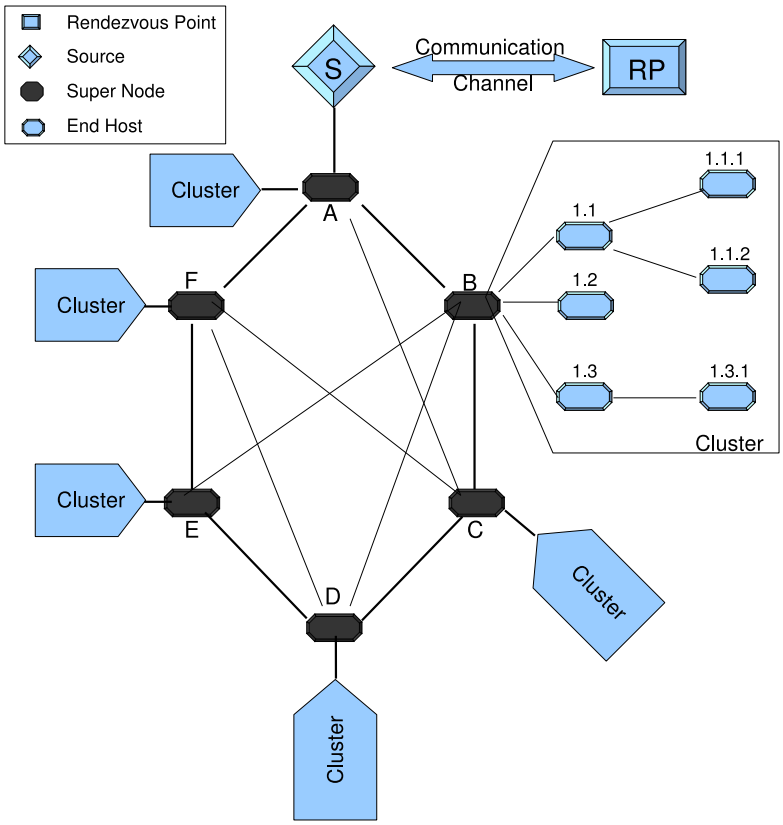

Figure 3.3: Example of DMMP Overlay Hierarchy.

specific requirements. In the initialization stage, nodes with higher bandwidth support will be naturally selected as super nodes since the current uptime is zero.

Step 2: After selection, super nodes self-organize into an overlay mesh rooted at the source. In addition, the source is regarded as a member of the DMMP-aware mesh since if the hosts are located at the same access network as the source, they receive packets directly from the source instead of from other overlay nodes. By doing this, DMMP-aware overlay tree can save the delivery time and the network resource.

Step 3: During the cluster creation procedure, each non-super node firstly consults its local cache for super node candidates. If there are no suitable candidates, it queries the RP immediately to obtain some new candidates, from which it chooses the best contributors based on e2e latency measurements. For example, node 1.1, 1.2, 1.3, 1.1.1, 1.1.2 and 1.3.1 all choose A as their super node. It is very important for multimedia application to consider the locality-awareness since it can save resources for the network providers when a large amount of traffic traverse within local networks. Recently, there have been extensive discussions and

contributions on P2P traffic optimization through locality-aware management [102][103]. Therefore, ISPs and end users can both benefit from locality-aware clusters.

Step 4: Then, node 1.1, 1.2, 1.3, 1.1.1, 1.1.2 and 1.3.1 form a local cluster. The cluster formation is initiated by the super node A which is responsible for informing the RP and contacting the source. Due to the A's available bandwidth, three end hosts, 1.1, 1.2 and 1.3, with larger capacity are selected as its immediate children.

Step 5: Since the capacity of super node A is exhausted, it responds to node 1.1.1, 1.1.2 and 1.3.1 with its immediate children and an indication of rejection. These requesters then send *Join* requests to 1.1, 1.2 and 1.3. In this case, node 1.1 accepts the requests from node 1.1.1 and 1.1.2, and node 1.3 takes node 1.3.1 as its child. Generally, requesters with higher out-degree are likely to be accepted as the children. If there are multiple acceptances, the end host attaches to the one which is "near" to it due to the e2e latency.

Step 6: The iteration will continue until all end hosts confirm their positions, and at the same time the control hierarchy is initially constructed for the overlay multicast group.

Essentially, DMMP can be regarded as a hybrid approach of application layer multicast and overlay multicast, which attempts to support one-to-many media streaming applications having hard real-time requirements [1]. The preliminary ideas of DMMP have been proposed in [97] and the ongoing Internet draft [104]. The detailed description of the DMMP protocol is illustrated in Section 3.5.

3.4 DMMP Messages

DMMP handles tasks related to overlay hierarchy management, multicast tree configuration and maintenance. It uses a common format to carry both data and control packets as shown in Figure 3.4.

Here, *version* refers to the current version of the DMMP protocol. The *tree version* field is used to prevent loops and partitions from the multicast tree. Since the multicast session tree is initialized and controlled by the RP, a loop free topology may be generated. Moreover, since tree update messages are independently disseminated to all group members, there is possibility that some messages might be lost and received out-of-order by different group members. These members may reply *Refresh* messages with updating their capacities. All these events could cause loops and tree partitions. In order to avoid these failures, the RP will assign a monotonically increasing version number to each newly generated multicast tree.

The *Option* fields in the header defines various types of operation messages as shown in below. Besides, the *Session ID* and *Source ID* are generated by the RP and guaranteed to be collision free in each multicast session. Moreover, the *Sequence Number* is used to identify the received media packets. For future usages, *Reserved* is left for possible extensions.

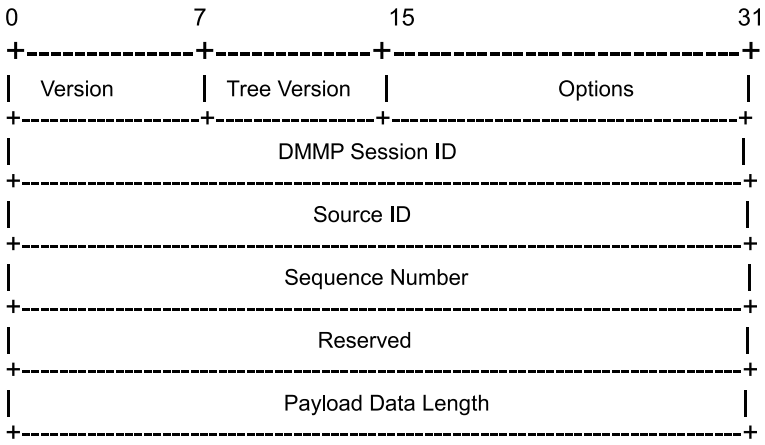

Figure 3.4: DMMP Packet Header Format.

In DMMP, there are seven pairs of control messages. Each pair of control messages will be exchanged between the DMMP-aware entities in a request-and-response way.

- Subscription Request and Response - Group members get the address of RP from the DNSs.
- Ping_RP Request and Response - During bootstrapping, each member of the group gets a list of available super nodes from the RP, containing at least one active node.
- Join Request and Response - A newly joining member sends request in order to join the multicast session and gets corresponding information from active group members.
- Status Request and Report - To request the status reports from neighbors or relatives, and accordingly to send reports to them.
- Probe Request and Response - To probe whether the target node is still active or not.
- Inactive Report and Response - To inform the other group members that the target node is inactive.
- Refresh Request and Response - To maintain the overlay hierarchy, they are used to periodically update the capacities (such as uptime, out-degree) of group members.

To adapt to dynamic network changes, each end host maintains the overlay core by periodically updating its capacities. For example, it periodically exchanges Refresh messages with its neighbors. If node A cannot receive this message from its neighbor, suppose node B, within the Refresh timer, node A will send a Probe request to node B. If there is still no Response returned, node B

Table 3.1: DMMP Messages

Messages	From	To	Operation
Subscription Req.	Group Member	DNS Server	Initialization
Subscription Resp.	DNS Server	Group Member	
Ping_RP Req.	Group Member	RP	Bootstrapping
Ping_RP Resp.	RP	Group Member	
Join Req.	Newcomer	Group Member	Member Join
Join Resp.	Group Member	Newcomer	
Status Req.	Group Member	Group Member	Cluster Member Monitoring
Status Resp.	Group Member	Group Member	
Probe Req.	Group Member	Group Member	Probing
Probe Resp.	Group Member	Group Member	
Inactive Req.	Leaving Node	Group Member	Member Leaving
Inactive Resp.	Group Member	Leaving Node	
Refresh Req.	Group Member	Group Member	Update Information
Refresh Resp.	Group Member	Group Member	

will be confirmed to be inactive by a certain time. Then, the Status report, indicating node B being inactive, will be used to inform the rest of group members.

Table 3.1 lists the DMMP messages according to the associated DMMP operational phases.

Although TCP provides a generic protocol for a guaranteed, in-order delivery of stream-based messages, this reliability comes at a price in the performance. Besides, the communication pattern in DMMP is strictly in a request-response mode, and most messages have a small fixed maximum size. Thus, it is preferred to encapsulate all DMMP messages over UDP to provide the required delivery guarantees without extra network burdens.

3.5 DMMP: Protocol Details

All DMMP-aware nodes and the source are assumed to be able to know the IP address of the RP. Also, once a source node starts the multicast session, a direct communication channel will be established between the source node and the RP. In this way, the necessary information like the capacities of group members and active super nodes could be exchanged between them during the lifetime of the overlay multicast session. Furthermore, it reduces the control overhead at the source. Obviously, a DNS namespace is necessary to maintain the RP information for a specified multicast group. It is possible that multiple RPs serve for the same multicast group, e.g. for load balancing and fault tolerance. However, for simplicity, the DMMP protocol initially considers the case where there is only one RP involved.

Before the initialization, each group member and the source send out Subscription request containing the specified group name and domain name, to the DNS for the address of associated Rendezvous Point. Since RP does not participate in the data forwarding, the location of RP has no significant impact on the performance of data distribution. If there is no existing RP which is serving for this multicast group, DNS will allocate a new one for this multicast group based on application requirements. Otherwise, the RP's address will be sent back.

3.5.1 Initialization

During the initialization phase, we assume that certain application related software has been distributed to the prospective DMMP-aware entities for the DMMP control and data transport models.

Before the multicast session starts, the source and RP must be ready to give response to requests from DMMP-aware end hosts. The source and RP will take no further reactions to any DMMP requests once the session stops. The active session time should be the period from the service starts until it stops. Then the out-of-band channel between the RP and source should be active during this active session so that the source can monitor the current status of memberships. However, the detailed mechanism for implementing this out-of-band bootstrapping is out of the scope of this document.

Moreover, session-related information should be obtained before the session starts and all prospective group members use out-of-band bootstrapping mechanism to get necessary information, for instance, Group ID and location of RP including the port number serving for certain sessions before the application begins. Then DMMP-aware entities can start receiving data after they join the overlay hierarchy.

3.5.2 Super Node Selection

During the mesh construction, the selection of the super nodes can ensure that a newly joining member is able to quickly find its appropriate position in the multicast tree using a very small number of queries such as Join request. As the super node selection is the first step towards the overall mesh establishment, this section gives more details about how to select super nodes.

To select super nodes for better performance while maintaining scalability, the following distribution requirements need to be taken into account.

- Connections: Super nodes have relatively higher capacities and are expected to be strong to perform additional tasks such as resources control, load balance and fault tolerance.
- Number: To be more efficient, the number of active super nodes is no more than one hundred, otherwise it may cause high control overhead and high stress [101]. Assuming that each super node can manage, in average, hundreds of cluster members, it is sufficient to

support totally more than thousands of end hosts. Otherwise, multiple sources will be deployed into media streaming by multiple overlay multicast sessions. To balance the tradeoff between the efficiency and the reliability, it is reasonable to select an application-specific number of super nodes to construct the overlay mesh. For example, in the case of 110 end hosts, it may need 10 super nodes if the required ratio is 10%. In this case, 10 end hosts with higher capacity will be chosen as super nodes.

- Downstream: To be adaptive to bandwidth requirements, super nodes should not serve more than K non-super nodes as its immediate children, where K is respectively determined by the available out-degree of each super node and service specifications.

In order to deal with factors from a large-scale and dynamic network environment, the following three conditions are outlined in addition to above requirements.

- Stability: The unstable network status is the main reason why current multimedia streaming services cannot guarantee required QoS. Thus, super nodes should be relatively stable because, otherwise, its cluster members are easily partitioned from the tree.
- Resilience: Super nodes are responsible for detecting dynamic changes and for handling them quickly, e.g., one super node leaves the group ungracefully, which should be detected by at least one of other active nodes. Then, a new super node should be quickly selected to replace the leaving super node. The time for detection and recovery process is also constrained by certain service requirements.
- Security: Super nodes should be fundamentally invulnerable to common attacks; otherwise, they will easily disrupt the multicast service by forwarding wrong/polluted messages or failing to accept correct information from other members.

Following above instructions, it is not difficult to select some required number of super nodes from a overlay multicast group.

3.5.3 Member Joining

To be resilient to dynamic changes, DMMP specifies how to handle events like a member joining/leaving.

If a new member wants to join the multicast session, it first checks its local cache for super node candidates. If there are no suitable candidates in the cache, it requests the RP for the addresses of the source and super node candidates. After receiving the source address and a list of active super nodes, it caches their capacities, i.e. uptime, out-degree. Then, it will measure the e2e latency between them and itself, and sends the Join Request message to some super node which can provide smaller e2e latency.

On receiving Join Request from a newcomer, the super node will check its current out-degree. If it is possible to accept the newcomer to as its immediate child, the super node will respond with an indication of acceptance. In this case, the information about newly joining nodes will only be propagated to the existing children of this super node since no child of the new member exists. When the super node cannot accept it as its immediate child, it will redirect this Join Request message to its active children with the largest available out-degree. If one of them responses to the super node, this response will be relayed to the new member. If there are more potential parents, the new member selects the one with smallest tree depth as its parent. If there are multiple potential parents at the same depth, it chooses the best one in terms of their uptime. Once finding the appropriate parent, the new member starts data delivery. At this time, the information about the new member will be propagated from the parent to its PLN, siblings and the super node. The process will be terminated until the new member finds its position and accordingly updates its related information at corresponding nodes.

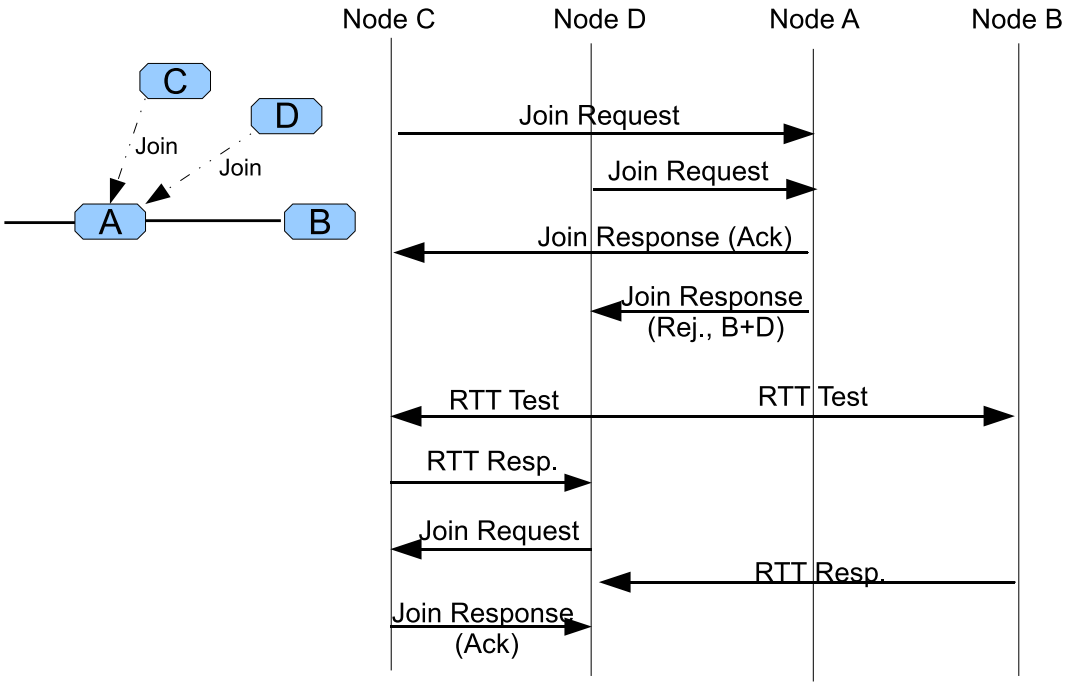

Figure 3.5: Example of DMMP-aware Joining Algorithm.

To illustrate above Join algorithm in a simple way, we give an example in Figure 3.5. Suppose node *A* has maximum degree of three and it already uses one degree for connecting with node *B*. Further, we assume that node *C* has higher capacity than node *D*. At the same time, node *C* and *D* attempt to join the multicast group. Both of them have obtained a list of randomly selected super nodes from the RP. Further, both have independently selected node *A* as their potential parent. Then, the join procedure will be performed as follows.

Step 1: Node *C* and node *D* independently send *Join Request* to node *A*. Since node *A* still has some available degree left, it can decide to take one node as its child.

Step 2: Node *A* doesn't accept node *D* as it child because node *C* has higher capacity than node *D*. Therefore, node *C* receives a *Join Response* with notification of acknowledgment, whereas node *D* receives a rejection in the *Join Response*. To help node *D* quickly join the group, the *Join Response* from node *A* contains the addresses of node *B* and *C*.

Step 3: Upon receiving the rejection as well as node *A*'s available children, node *D* sends *RTT Test* messages respectively to node *B* and *C*. It is possible to use other measurement metrics, for instance, the length of the shortest path in the underlying topology. Nevertheless, we believe round-trip measurement is a lightweight, efficient approach.

Step 4: Once node *D* receives a *Join Response*, it is confirmed to join the group. There is no necessity for node *D* to wait for other responses. As it receives the response from node *D* earlier than any other responses, the e2e delay between node *C* and node *D* is definitely shorter than that of others.

In case the newcomer fails to find an appropriate position in any existing clusters to meet application requirements, it can sell itself as a potential super node and report its own capacities to the RP. Regarding its capacity and the current number of super nodes, it could be entitled as a super node. In this way, end hosts have more flexibility to get optimal services, which will be left for future development due to the difficulty in the group management.

3.5.4 Refresh Information

In DMMP, each member is responsible for maintaining the overlay hierarchy, by periodically sending Refresh message. The Refresh mechanism in the overlay mesh has a little difference from that in the local clusters. To efficiently manage the overlay hierarchy, both active and passive models are utilized in DMMP.

Within each cluster, end host starts to exchange Refresh message with its PLNs, siblings and CLNs once it joins the cluster. In addition that each member has to periodically update its information, members in the local cluster are able to request refresh message from their relatives, e.g., PLNs or CLNs. This operation guarantees the reliability and the stability of the overlay hierarchy.

For the Refresh message in the mesh, each super node sends its current information to all mesh members including the source. Once receiving updated information, the source will correspondingly update the information at the RP. If one mesh member stops receiving Refresh message from another beyond the Mesh_Refresh_Timer, it assumes this neighbor to be either inactive or leaving. In order to confirm the status, it may initiate a Probe message as stated in Section 3.4.

3.5.5 Member Leaving

In most cases, two situations stand for a member leaving the group, either gracefully or ungracefully. In each cluster, the graceful leaving member should at least send an Inactive Request to its parent or one of its children. After receiving the confirmation, it can leave the group gracefully. Then the notified node will propagate this Inactive message to its relatives so that they can update their service membership tables. For an ungraceful leaving case, the Inactive status will be detected by periodically exchanging Refresh messages. If any member within the cluster, say *p*, fails to receive a Refresh Report message from one of its required relatives, say *q*, within the refresh timeout Refresh_Timer, then *p* sends a redundant Probe Request message to *q*. If there is still no Probe Response message returned, *p* assumes *q* to be inactive and propagates this Status_Inactive message throughout the whole cluster. Afterwards, one of it children with relatively high capacity will replace its place, and other children will accordingly change their positions. Nevertheless, ungraceful leaving may cause the crash of whole multicast tree. DMMP is able to handle different situations by detecting the failures and recovering quickly from them as shown in Section 3.5.8.

Compared with the handling in the local cluster, the operation is even tougher in the core mesh since all its cluster members are partitioned from the tree. When there is no end hosts connecting to the leaving super node, no further changes to the overlay are required. Before a super node gracefully leaves the group, it must recommend a replacement leader for the cluster it owns and inform other super nodes in the overlay mesh before leaving.

Take an example to explain how the partition is handled. Similar to the algorithm described in [101], each super node is assumed to store a *refresh table* with one entry for each other super node. Suppose a super node sn_i has detected a partition and the entry e is used to perform the following operations. First of all, a probe request is sent to super node sn_j with the IP address $e.ip_j$. If no response is received, the function *handlePartition()* is called. Most likely, sn_i does not receive an update from sn_j before sn_j has left the group. In a worst case, sn_i needs $t_m \cdot d$ seconds to know the departure of sn_j, where t_m represents the refresh timer and d is the diameter of the mesh. This approach can guarantee that mesh partitions are repaired quickly without causing additional mesh partitions.

To detect unannounced leaves, DMMP relies on the periodic Refresh message exchanges. If the failed peer happens to be a super node, the overlay hierarchy has to be repaired, which will be depicted in Section 3.5.8.

3.5.6 Member Rejoining

In DMMP, members may be unreachable due to network condition changes or membership changes. In these cases, it necessary for the partitioned peers to rejoin the multicast session. To facilitate the rejoin procedure, we propose to rely on relatives (e.g. sibling, PLN) as mentioned in Section 1.2.

To explain the procedure in details, we take an example of relying on Parent Level Node (PLN) to

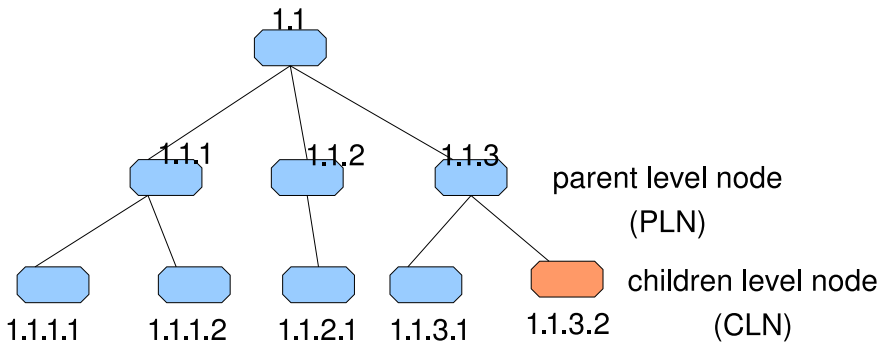

Figure 3.6: Member Rejoining Procedure.

rejoin the group. As shown in Figure 3.6, node 1.1 is stored as grandparent for the Children Level Node (CLN). To be more robust to dynamic changes, especially to the membership changes, we propose to periodically exchange messages between the relatives. Suppose that node 1.1.3.2 just joins the group, it immediately sends a request to its parent, namely, node 1.1.3. By that time, node 1.1.3 has already stored its parent, node 1.1, in its cache. Therefore, it is quite easy for node 1.1.3.2 to get the grandparent stored in its cache. However, two possibilities may still happen while requesting the grandparent's address from its parent. One possibility is that the parent, node 1.1.3, may leave ungracefully before it receives a request from node 1.1.3.2. The second possibility is that the grandparent 1.1 leaves before the request from node 1.1.3.2 can be arrived. For simplicity, we don't consider the possibility that the grandparent may leave after it responses to node 1.1.3.2 since the its information will anyway be updated to the Children Level Node (CLN).

For the first case that node 1.1.3 has left before it can notify node 1.1.3.2, the solution is straightforward: node 1.1.3.2 is actually partitioned from the multicast tree and it needs to rejoin the group through normal joining procedure. Once it rejoins the group, it can send request to its new parent node. As long as the parent is available, it is not difficult for node 1.1.3.2 to get information of its grandparent. However, for the second case, node 1.1.3.2 either sends a request to node 1.1.3 again in order to get a new grandparent, or waits for any notification about re-construction of the cluster.

3.5.7 Data Delivery Control

After the multicast tree configuration, the new member will ask its immediate parent to send the data. Generally, parent nodes will delete data in their local cache after they have forwarded them to their children. If the parent still holds the data, the new member can quickly get the data from it. If the parent has not received data yet, either it waits until the parent forwards the data after receiving, or it directly inquires the super node to deliver the data. The former option is preferred in DMMP as its overhead is likely much lower than the latter one. Upon receiving the data, the new member will firstly forward them to its parent if its parent hasn't received the data yet. If the

parent has deleted the data, it will then ask its siblings to forward data directly. In order to alleviate the redundant data transmission, the new member needs to wait for a certain while before it asks for the data from its siblings.

Different from the procedure for joining as a cluster member, the new member may join the multicast session as a super node. In this case, it will firstly ask its neighbors in the overlay mesh to send the data. In addition, it may query the data from its children when they are already in its local cluster. If any of them receives the data, this new super node will also get the data. A third possibility is to let the new member directly ask the source to send the data. To alleviate the control overhead, it is recommended that this new node waits for a certain while until one of its mesh neighbors receives the data.

3.5.8 Failure Recovery

The maintenance of a multicast tree in DMMP faces a key issue because all non-leaf nodes in the tree are end hosts who are more likely to fail than routers and may join/leave the tree at will. It does not happen in IP multicast since non-leaf nodes in the delivery tree are routers which do not leave the multicast tree without notification. Thus, one design challenge in DMMP is to efficiently reconstruct the overlay multicast tree after a node's departure.

If one non-leaf node leaves the group ungracefully, its downstream nodes will be inevitably affected. Two possible means can alleviate such impacts: one is to reduce the possibility of failures; the other is to reduce the number of possible affected nodes. In practice, however, the first way might be very difficult since end hosts may leave the group at will. For the second option, DMMP proposes a proactive mechanism by periodically pushing high-capacity nodes to higher levels of the tree by capacity comparison mechanism. Detailed mechanisms are described in Section 5.1.

Thereby, it is very likely that long-lived super nodes and their immediate children form a stable and efficient cluster core after a certain time. The longer a node remains in the multicast session, the more it becomes attaching to other long-lived nodes with similar uptime. In other words, higher capacity nodes form a well-connected core with relatively more bandwidth support and being more stable, whereas peers with less available bandwidth and shorter uptime will be placed out of the core as possible.

In order to improve the performance of DMMP, especially when there are high packet losses or host failures, a reactive recovery technique is also used after failure detection. Recovery from failures regarding a member crash is similar to handling a member leaves. The difference is that surviving members usually do not receive prior notification of a crash. Thus, Refresh message is periodically exchanged between each member and its neighbors. It is even more difficult for super nodes to maintain the cluster since all of its local members are partitioned from the multicast tree and can not receive the multicast data until it is repaired.

In the local cluster, each immediate child of the super node must find a backup parent in advance, either the source or a group member. Once the super node leaves the group, its children try to

Table 3.2: Selection Criteria

Criteria	Involved Operations
Out-degree.	Differentiation of non-leaf nodes from leaf nodes
	Super node selection
	Tree construction within clusters
	New member joining the group
	Failure recovery mechanisms
E2E delay	Non-super nodes attach to super nodes to form clusters
	New member joining the group
Uptime	New member joining the group
	Failure recovery mechanism

contact with their alternative parents to rejoin the multicast tree. This approach can facilitate the recovery process and strengthen the reliability of the overlay hierarchy. In addition, cluster members periodically estimate their relatives (i.e. PLNs, CLNs) within the cluster and evaluates the number of losses that it shares with these nodes. While in the core mesh, each super node maintains state information about all other mesh members, no additional discovery of nodes is necessary. Using this mechanism, packet delivery ratios can be increased with a high probability. To handle different scenarios of failures, more mechanisms need to be defined in the near future.

3.6 Selection Criteria

End host based overlay multicast is more sensitive to dynamic network changes since end hosts may join or leave the group at will. It would be even harder for DMMP to manage and maintain such an overlay mesh because super nodes may leave the group ungracefully as well. To address the instability of mesh, uptime is chosen as an assisted criterion to strengthen its maintenance. Once an end host joins in the overlay multicast tree, its uptime starts to calculate from zero until its leaving. Besides, there are several metrics used in DMMP as the criteria of the capacity, which will be specified as follows.

As shown in Table 3.2, out-degree is the main criterion to select the super nodes from end hosts. Then, non-super nodes select one super node which locates “near” to it based on the estimated e2e latency. During the tree construction within clusters, nodes with higher out-degree are likely to join in the tree at the high level. Regarding new members joining procedure, out-degree, e2e latency and uptime are all taken into considerations. To keep the stability of the overlay hierarchy, out-degree and uptime are chosen as comparison metric to enhance the stability of the overlay multicast tree. Furthermore, out-degree and uptime are regarded as the main selection criterion for searching alternative nodes during the failure recovery.

3.7 Security Considerations

The main goal of the DMMP framework is not to offer a bullet proof system, but to provide a simple and efficient media distribution solution. Therefore, we present a few thoughts on the most important security requirements for the DMMP framework. The studies outlined below could be viewed as a starting point towards better solutions for overlay construction and maintenance that we will develop in the near future.

First of all, the security considerations should be taken during the super node selection (cf. Section 3.5.2). In DMMP, the current preference is to rely on an authority center which qualifies the trust level of participated end hosts. Once an end host obtains a security certificate (e.g. digital certificate) from the authority center, it is entitled to be selected as a super node. Otherwise, this end host has been temporarily considered as an unqualified candidate for being a super node.

Besides, within each cluster we propose to use *Cluster Key*, *Group Key* and *Private Key* as the security scheme to manage the cluster members in a safe way. For example, the following three items are considered within the key management scheme:

- **key establishment**: how to create keys is the fundamental problem, which form the basis of key management. Usually, there are two ways, namely, stateful and stateless configuration which may be application specific and we need to decide whether the key establishment mechanism is able to accommodate different ways of configuration.
- **key distribution**: the importance of key distribution is distributing small-size but critical messages (e.g. keys, virus signatures) to a large number of overlay nodes that are organized into trees, meshes, or other types of overlay structures. For instance, we could employ conventional public key techniques that pairwise keys are established between two nodes. Symmetric key techniques may also work, but have several limitations, such as limited scalability for large groups, no flexibility for adding new members.
- **key storage**: either centralized or decentralized storage can be further studied. Another technique called probabilistic key pre-deployment [105] can be studied to reduce the storage overhead, however, currently it can only provide threshold security in case that a certain number of colluding members can greatly jeopardize the secure links shared between other members in the system.

3.8 Summary

In this chapter, we focused on specifying and identifying the properties of the DMMP framework, which can overcome the identified challenges. The proposed DMMP framework has a few major differences from existing works presented in Chapter 2.

- We proposed a dynamic two-tier architecture with one mesh core and multiple clusters, which was composed all by end hosts. Here, "dynamic" refers to that DMMP-aware mesh is resilient to dynamic changes (e.g. member joining/leaving).
- The proposed overlay hierarchy does not need any infrastructure upgrade, and therefore can be deployed into the current Internet. DMMP relies on IP unicast to deliver the packets through decentralized users.
- To construct the DMMP overlay hierarchy, we combined the available bandwidth and up-time to represent the capacity of each node. In fact, the idea was motivated from economic philosophy and incentive mechanisms that long-staying capable clients who are willing to contribute more to the network, most likely get better service than others. Moreover, the heterogeneity has been specifically considered in DMMP.
- To form the cluster, "locality-awareness" was taken into considerations. That is, nearby end hosts were converged into the same cluster. Such a concept brings two major benefits: a) the serving delay will be reduced; b) reduction of the control overhead and the complexity of the overlay maintenance.

Chapter 4

DMMP Modeling and Performance Evaluation

In this chapter, we model the DMMP framework (as presented in Chapter 3) and evaluate the performance through both theoretical and simulation studies. Most importantly, we attempt to identify the efficiency of media data delivery, the service quality (e.g. data path quality) experienced by the end hosts, and the reliability (e.g. being resilient to dynamic changes) of the DMMP framework, against IP multicast and other ALM approaches.

The rest of the chapter is organized as follows. Section 4.1 gives a theoretical analysis on how it is possible to support four properties (seen in Section 3.2) in the DMMP framework. The analysis is comprised of: 1) the required number of non-leaf nodes for DMMP-aware tree construction; 2) analysis on the tree depth; 3) resilience to dynamic changes; and 4) analysis of convergence time. The performance metrics are introduced in Section 4.2. Section 4.3 discusses impacts of the related metrics on the above theoretical analysis. The second part of the analysis, we focus on the performance evaluation under various simulation scenarios, especially, in highly dynamic circumstances with a large amount of heterogeneous end hosts. Our methodologies and experimental setup are illustrated in Section 4.4. Meanwhile, the major ideas of DMMP are implemented in OMNeT++ simulator (introduced in Section 4.4.1). Section 4.5 presents the main performance results obtained from two simulation scenarios. Section 4.6 summarizes the chapter.

4.1 Theoretical Analysis on DMMP Properties

One important property of DMMP is the ability to support the heterogeneity of the node capacities. Currently, we consider the out-degree as the primary metric, which is noted as the number of the outgoing multimedia sessions that a node can establish. For example, on the assumption that the bit rate of media is B and the outbound bandwidth of an end host i is $b(i)$, the total number of

sessions it can establish is $b(i)/B$ which is also the maximum degree of the end host. Meanwhile, the knowledge of available bandwidth in overlay routing is nowadays regarded as acquirable, based on recent advances in available bandwidth measurement techniques and tools [98]. However, in a heterogeneous environment like Internet, only a small number of end hosts can provide extra out-degree, while a large number of them can only receive data from incoming sessions, so-called leaf nodes.

One question immediately arises: how many non-leaf nodes are required when constructing the overlay multicast tree for a given sized group and a given topology of network? To answer this question, we firstly estimate the required non-leaf nodes which can provide extra out-degrees for other nodes in terms of different multicast groups, to form the multicast tree in each cluster.

Remark that the following theoretical analysis is DMMP-specific only, we do not compare the results with other solutions. In contrast, the simulation studies will consider the comparisons with IP multicast and other ALM solutions.

4.1.1 Required Number of Non-leaf Nodes

Constructing an overlay multicast tree can be modeled as a degree-constrained spanning tree problem. For the convenience of our discussion, one cluster case is taken as an example to explore the possibility of constructing the DMMP-aware overlay multicast tree. We assume that m end hosts participating in the cluster in which the percentage α of end hosts are non-leaf nodes. Out-degrees for i ($1 \leq i \leq [\alpha \cdot m]$) non-leaf node is n_i. That is, $[(1-\alpha) \cdot m]$ end hosts could only perform as leaf-nodes as they can hardly provide extra out-degree for other nodes. These leaf-nodes are planned to be placed at the bottom of the overlay multicast tree as possible because they can just receive the services instead of making any contribution to the network. Unless otherwise stated, in the remaining sections, above notations are kept in the same meaning.

Observed from [106], it is possible to compose an overlay multicast tree if and only if $n_i \geq 1$ and $\sum_{i=1}^{m} n_i \geq 2m$. Since m-node spanning tree has $m-1$ edges and each edge results in two degrees used, one for each node. Then, based on our assumption we have the following inequality,

$$(1-\alpha) \cdot m + \sum_{i=1}^{|m \cdot \alpha|} n_i \geq 2(m-1), \tag{4.1}$$

as $[(1-\alpha) \cdot m]$ end hosts have only one out-degree. Besides, we also set the maximal out-degree of arbitrary node as $n_{max} \leq k$ where k is upper limit of the out-degree for an arbitrary node. If we assume the average out-degree of non-leaf nodes is $\overline{n}$, due to $0 < \alpha < 1$ the average out-degree $\overline{n}$ will have the following constraints.

$$2 - \frac{5}{m+2} \leq \overline{n} \leq k, \tag{4.2}$$

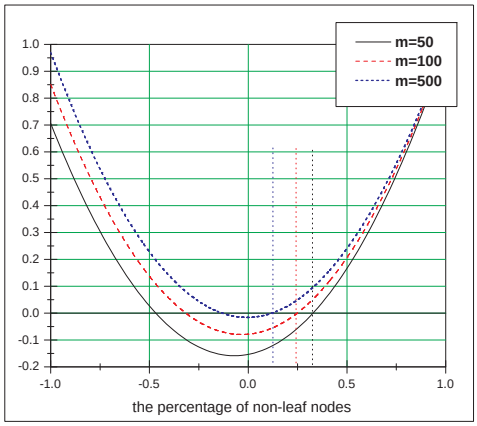

Figure 4.1: Ratio of Required Non-leaf Nodes.

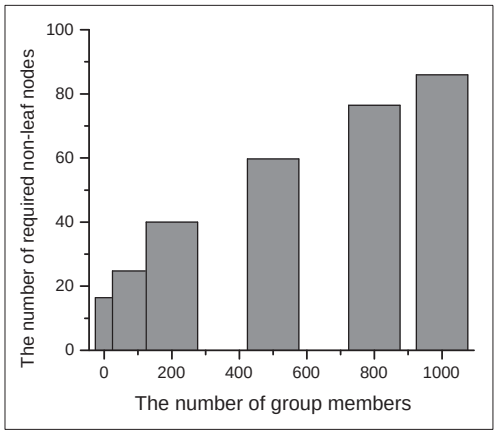

Figure 4.2: Nr. of Required Non-leaf Nodes.

That is, $\overline{n}$ which requires almost two extra out-degrees. Nowadays, it is quite common for a subscriber to have two extra out-degree (e.g. 256 kbps available bandwidth) as many Internet content providers use the Windows Media Audio 9 codec to deliver media content at 64 kbps or 128 kbps. Besides, while the total number of group member increases, the required $\overline{n}$ keep constant.

More specifically, we assume that the distribution of n_i is an arithmetical series, that is, $n_i = [n_1 + \Delta \cdot (i-1)]$ and $n_i \leq k$. For brevity, we set $\Delta = 0.25$ since we assume the group members have heterogeneous capacities. To make Equation 4.2 come into existence α should satisfy the following two inequalities (suppose $n_1 = 2$):

$$m^2 \cdot \alpha^2 - (8m \cdot n_1 - 9m)\alpha - 8m + 16 \geq 0. \tag{4.3}$$

$$m^2 \cdot \alpha^2 - m \cdot \alpha + 8n_1 \cdot m \cdot \alpha - 8k \cdot m \cdot \alpha \geq 0 \tag{4.4}$$

As long as two conditions are satisfied,

$$(\alpha + \frac{7}{2m})^2 + \frac{15}{4m^2} - \frac{8}{m} \geq 0, \tag{4.5}$$

$$m \cdot \alpha - 8k + 15 \leq 0 \tag{4.6}$$

the overlay tree can be constructed.

Figure 4.1 and 4.2 interprets the relationship between the minimal number of required non-leaf nodes and the minimal out-degree in terms of different group. Let we take $m = 500$ as an example,

α should satisfy the condition $\alpha \geq 0.12$. That is, it is possible to form the overlay multicast tree for 500 end hosts if there are at least 60 non leaf-nodes with minimal out-degree two. In this case, a large number - nearly 440 - leaf nodes exist in the network, which is quite accord with the common situation over the today's Internet.

Based on the condition of $0 \leq \alpha \leq 1$, Figure 4.3 depicts the impacts of k on α. The larger K is, the allowed maximal α could be. Besides, when the number of group becomes larger, the impact of k on α becomes less. It is reasonable since in a large group end hosts may have large different in their capacity even if the out-degree is constrained. Remark that Section 4.5.1 further discusses the impacts of k-interleaved spanning tree on the DMMP mesh.

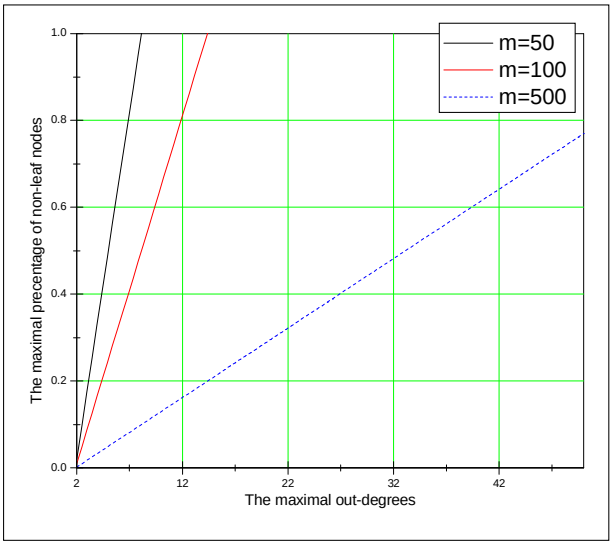

Figure 4.3: Impacts of K-spanning tree.

From above analysis, it should be possible to construct the DMMP-aware overlay multicast tree to satisfy bandwidth constraints of media streaming applications although end hosts have different capacities (e.g., available bandwidth support). Since DMMP targets at providing an efficient and resilient multicast solution for large-scale media streaming applications, it should optimize the overall delay besides satisfying the bandwidth requirement.

4.1.2 Tree Depth

We believe there is a need to reduce the overall delay of the multicast tree, which can be easily observed from the time-constraints of media streaming systems, for instance, a packet arriving after its scheduled play back time is useless and considered as lost. The question arises concerning

how to optimize the overall delay of DMMP-aware multicast tree. Observing from the overlay tree construction, the overall delay in DMMP can be broken down into three parts. The first-mile delay from the source to each super node can be alleviated by choosing super node close to the source. However, it can also be achieved by deploying multiple servers within each service domain. The second-mile delay of transporting packets from each super node to its cluster can be alleviated by attaching to the multicast tree with lower e2e delay between the super node and local end host. This objective is already taken into consideration (e.g., cluster formation phase).The last-hop delay of forwarding packets from overlay upstream nodes to overlay downstream nodes has a great impact on the e2e delay of each node. This is very important bottleneck to overcome, and mechanisms for shortening such delay are the topic of this section.

Hence, the objective of reducing the overall delay can be regarded as constructing the multicast tree within each cluster as short as possible. It is also noted that the overall delay from the top to the bottom of the tree is somehow proportional to the depth of the tree. In DMMP, a mechanism is proposed that nodes with larger capacity would be assigned to the higher level in each cluster. This seems reasonable as more end hosts could attach to the tree at each level, and the tree depth would be shortened. In addition, those nodes staying at the higher level are likely to get better performance if they are willing to contribute more to the overlay applications.

In reality, it is, however, not so optimal since some leaf nodes may have already occupied the positions at the higher level of the tree. In this case, some high out-degree nodes could only be attached to the initial tree at the lower level. To explain the tree depth problem more explicitly, in the following subsections we discuss the issue concerning two cases.

The Worst Case

In the worst case, all leaf nodes will attempt to join the tree at the higher level or they have already occupied these places. Theoretically, at least one non-leaf node should stay at each level of the tree; otherwise, it is impossible to support multicast sessions for downstream nodes. One example mechanism attaching to the tree without invitation allows nodes to join in the tree once they receive an answer from one of the group members [107]. This approach would create deep graphs with tree depth of $[(1-\alpha)\cdot m/(\overline{n}-2)]$ but fast join operations and less cost of tree construction.

The Best Case

In contrast to the worst case, the best situation is that all nodes with higher out-degree try to occupy the positions at the higher level of the multicast tree so that all leaf nodes can only be placed at the bottom level. For example, the approach of attaching to the tree with best invitation supposes that a newly joining node waits for all responses from the requested nodes until it finds the best one [107]. This approach would create wide graphs with a low worst-case tree depth $log_{\overline{n}-1}^{m(1-\alpha)}$ but slow join operations and high cost of tree construction as well.

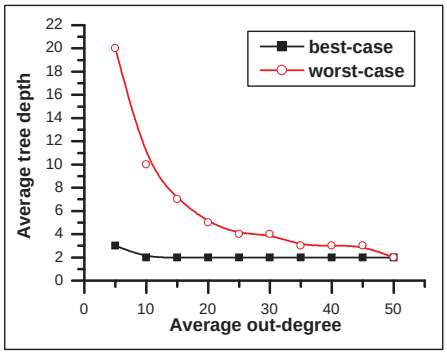

Figure 4.4: Tree Depth at N=100.

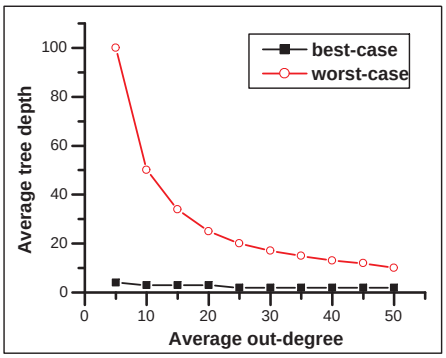

Figure 4.5: Tree Depth at N=500.

Accordingly, we make use of the above analysis to derive the result of tree depth issue concerning the best case and worst case in terms of $m = 100$ and 500. In Figure 4.4 and Figure 4.5, the tree depth decreases dramatically between out-degree 5 and 10 concerning the worst case. The difference between the best case and the worst case is huge, especially when the group is large. This is because some early-joined leaf nodes may have already taken up the higher position of the tree. How to cope with this situation? A self-refinement mechanism is proposed to periodically optimize the established DMMP clusters as described in Section 5.1. Upon expression(1), nodes either with definitely higher bandwidth support or having joined in the multicast session for a long time will be switched to or kept staying at the higher level of the tree.

4.1.3 Resilience to Dynamic Network Changes

One cause for current multimedia streaming services which cannot guarantee required QoS occurs mainly from unstable network status. Compared with IP multicast, overlay multicast approaches usually are more perceptive to dynamic network changes, e.g., nodes leave the group just after a short time, which are also called transient nodes.

How can DMMP achieve the above objective? To detect node failures (e.g., leaving accidentally), REFRESH message is periodically exchanged between each group member and its relatives (e.g., parent, sibling). If a node fails to receive a REFRESH message from one of its required relatives in a certain time, it will send a PROBE message to the relative. If there is no response message returned, the relative is confirmed to be inactive. One of its children with higher capacity will replace its place, and other children will correspondingly change their positions. Consequently, the information will be updated in the source and associated nodes. However, if the leaving node is a super node, it will be even more difficult since all its cluster members are partitioned from the tree. One possible solution is that each immediate child of the super node must find a backup parent list. Once the super node leaves, these children try to contact their alternative parents to rejoin the tree [97].

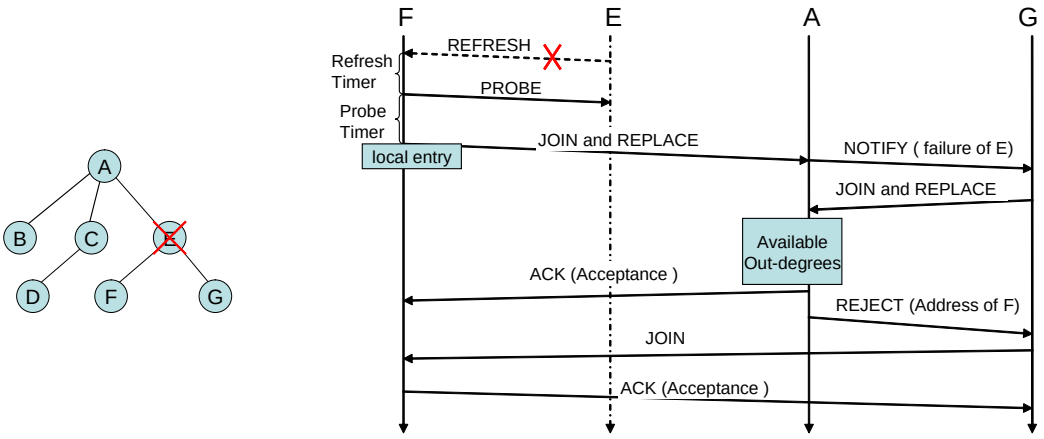

Figure 4.6: Example of Message Flows During Failures.

More concretely, Figure 4.6 shows the flow of messages when node E fails. In this scenario the failure of node E is detected by one of its children, say F since E fails to answer the *PROBE* from F. Then, node F will search its local entry for the parent node of E, that is, node A. It sends *JOIN and REPLACE* request directly to A and A will notify other children of E. Once notified by A, other children, for example G, will send *JOIN and REPLACE* to A as well. After a certain while, A will choose the higher capacity node to replace E. Here, node A choose F as the candidate to replace E. Accordingly, A accepts the request from F and reject the request from G. Moreover, the address of the replacement (e.g. address of F) will inserted in the *REJECT* message. Upon receiving the address of the new parent, rest of the children (e.g. node G) will re-join the multicast tree by requesting the new parent, F.

Nevertheless, whenever one non-leaf node leaves the group, its downstream nodes will be inevitably affected. We believe that two possible means can alleviate such impacts: one is to reduce the possibility of failures; the other is to reduce the number of possible affected nodes. In practice, however, the first way might be very difficult since end hosts may join/leave the group at will. For the second option, DMMP proposes a proactive mechanism by periodically pushing high-capacity nodes to higher levels of the tree. Meanwhile, we combine *uptime* with available out-degree as the capacity of each node, which is depicted in expression (4.1) to strengthen the maintenance of the overlay hierarchy.

Thereby, it is very likely that long-lived super nodes and their immediate children form a stable and efficient cluster core after a certain time. The longer a node remains in the multicast session, the more possibility it has to attach to other long-lived nodes with similar *uptime*. In other words, higher capacity nodes form a well-connected core with relatively more bandwidth support and being more stable, whereas peers with less available bandwidth and shorter uptime will be placed out of the core as possible. In addition, the newcomers who have higher capacities could “climb” from the bottom to a higher level after some switching stages. For example, a newcomer at the

lower level could switch with its parent if its capacity exceeds (over a predefined threshold) the current parent. Here, an appropriate threshold will be defined to avoid unnecessary switching since if the child has a smaller bandwidth support, it will be ultimately placed below the parent.

To summarize, stable nodes with higher bandwidth support are likely placed at the higher level of the DMMP-aware tree regardless of dynamic changes. Moreover, a node is encouraged to contribute more resources or longer service time to the network in tradeoff for a better service quality. These design options and their detailed implementations will be studied in the following simulation analysis.

4.1.4 Analysis of Convergence Time

The convergence time in DMMP greatly relies on the tree depth and the percentage of non-leaf nodes. Suppose non-leaf nodes have similar attaching time at a_1 and leaf nodes at a_2. According to our basic mechanism that high capacity nodes have the high priority to join the tree, we get $a_2 < a_1$. Moreover, we suppose that each level takes the similar convergence time t for the tree construction. That is, the convergence time of each level of the tree can be presented as follows: $t_i = t_{i-1}$ where i represents the ith level of the tree.

Two extreme cases are involved: best case and worst case, which have been shown in the previous subsection. At the best case, all incapable nodes are placed at the bottom of the tree. Therefore, the convergence time will be:

$$t_b \leq (log_{\overline{n}-1}^{m(1-\alpha)} - 1) \cdot t \cdot \overline{n} \cdot a_1 + a_2 \cdot m(1-\alpha) \cdot t \tag{4.7}$$

In case the positions at the higher level has already been occupied by the leaf nodes, the convergence time becomes:

$$t_w \geq \alpha \cdot m \cdot t \cdot a_1 + a_2 \cdot (\overline{n} - 2)[(1-\alpha) \cdot m/(\overline{n} - 2)] \cdot t \tag{4.8}$$

namely,

$$t_w \geq t \cdot m \cdot (\alpha \cdot a_1 + (1-\alpha) \cdot a_2) \tag{4.9}$$

If the multicast group is large (e.g., $m > 50$) and α satisfies the condition in inequality 4.5, it is very obvious that $t_b << t_w$. It also indicates that our mechanism is very useful, by pushing the high capacity nodes to the high level of the multicast tree. Nevertheless, the heterogeneous capacities of the underlying end hosts and different methods of constructing the multicast tree may have a great impact on the convergence time. However, it is not the main target of our proposal and additionally in reality due to dynamic membership changes the convergence time varies at any time. Therefore, this metric will not be discussed in the following simulation experiments.

So far, we have identified the possibility of constructing DMMP-aware overlay multicast tree and discussed some performance related factors, e.g. out-degree, tree depth, convergence time. However, there are some parameters closely related with the underlying network topology, which may have a great impact on the performance.

4.2 Performance Metrics

Through the in-depth investigations of video distribution systems in Chapter 2, we expect to extend this study for use as general guideline for application level multicast protocol design, validation and evaluation. This includes, but is not limited to, validation of protocol correctness and robustness, evaluation of overhead, efficiency, scalability and other performance metrics. The focus of our evaluation will be placed on existing metrics, such as stress, stretch, convergence time and control overhead [108], [109].

Stress: We refer to the number of identical copies of a packet carried by a physical link as the stress of a physical link [101].

To get a better understanding of this concept, in Figure 4.7 we show two examples of application layer multicast and one for network layer multicast, all on the same topology of routers and end hosts. Meanwhile, square nodes represent routers and circular nodes are end hosts.

Let us assume that each link on the topology is of 1 unit length. According to the definition of *Stress*:

- Case (1) Max Stress = 1; Average Stress = 1;
- Case (2a) Max Stress = 3, which is the link from end host A to router R1; Average Stretch = 1;
- Case (2b) Max Stress = 2, which is the link from end host A to router R1; Average Stretch = $(\frac{3}{3} + \frac{6}{4} + \frac{3}{3})/3 = 1.67$

Delay Performance: It is evaluated by stretch or Relative Delay Penalty (RDP).

$$RDP = \frac{\text{Overlay Delay}}{\text{Unicast Delay}} \tag{4.10}$$

End-to-End Delay: The e2e delay experienced at each DMMP-aware end host i is $d_i = t_r - t_s$, where t_s is the time that the source sends out the data, and t_r represents the time when "same" data is received at i.

Tree Cost: It can be defined as follows.

$$\text{Tree Cost} = \frac{\text{Overlay Tree Cost}}{\text{IP Multicast Shortest Path Tree Cost}} \tag{4.11}$$

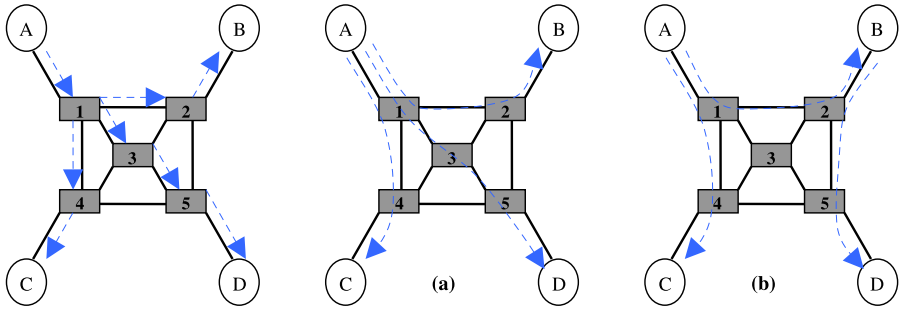

Figure 4.7: Examples of Application Layer Multicast and Network Layer Multicast.

Control Overhead: It refers to the average number of bytes of control packets.

Convergence time/speed of the protocol: It is a process and a measurement, respectively, of the adaptation of a computer network to unplanned changes in its topology or structure.

Loss Rate: Most of the multimedia applications rely on UDP to transmit the data, which does not guarantee a reliable delivery. It might be often happen that the source sends out the video packets to the group, however, some of the group members could not receive the data due to the bad network status (e.g. network congestion).

We measure the loss rate in accordance with its definition in [43]: for each group member i, the loss rate can be evaluated through the following equation:

$$L_i = \frac{N_t - n_t}{N_t} \tag{4.12}$$

Here, N_t is the total number of data sent from the source to the group within a timer t, and n_t stands for the number of induplicated data that member i.

The further evaluation in the rest of the thesis is performed both theoretically and experimentally. Theoretical evaluation is conducted using an analytical model and a stochastic model, while experimental analysis relies on network simulations. Using network simulations, we will study and evaluate protocol correctness and robustness according to different network models, e.g. Georgia Tech (GT) network model [110], [111].

4.3 Discussions on Performance Metrics

Towards validating our theoretical analysis presented above, we describe initial results consisting of a series of performance evaluations on a proposed model. Scalability, efficiency, resilience

and QoS are major concerns for application level multicast Since DMMP is designed to support multimedia services, we consider the bandwidth consumption, the efficiency of delivery (e.g. the consumption of control traffic), packet loss rate, and latency. Specifically, we focus our efforts on discussing the metrics of **stress**, **control overhead**, **loss rate**, and **data path length**, which have been introduced in the above section.

- **Stress**: The stress of a link in the underlying network refers to the number of copies sent over the link (in both directions). The stress of a router is the number of forwarded copies of a packet. It mainly measures the additional load on a network link, and therefore it closely related with the efficiency of resource utilization and scalability of the protocol. Generally, we would like to keep the stress on all links as low as possible. For instance, the stress for any network level multicast tree is one.

  Model: Since we are interested in the asymptotic nature of the metric, we assume a very large number of end hosts are uniformly distributed in the network. Thus, the DMMP-aware clusters will have similar properties, e.g., will have the similar number of cluster members, k, and the same cluster radius.

  DMMP builds the data delivery plane directly on top of the overlay hierarchy as shown in Figure 3.1. A distance vector protocol runs on top of the core mesh and within each cluster data is top-down forwarded across the DMMP-aware tree. Thus, the number of links that connect super nodes of cluster C_i to their respective cluster members is given by p_i which is no more than their out-degrees. The number of packet copies is determined by the number of downstream nodes ($\leq n_i$). The average link stress would be:

$$\overline{\lambda} \leq \frac{\sum_{i=1}^{L} p_i + L[1 + \sum_{j=1}^{k\cdot\alpha}(n_j - 1)]}{N}. \tag{4.13}$$

  where N is the total number of links (nodes) in the network, L notes the number of super nodes and n_j is the used out-degree of non-leaf nodes j. Since the member population is large and network is uniformly populated with the members, let us make a fluid approximation on the expression 4.13 based on the inequality 4.2. Therefore, the average link stress of DMMP will be:

$$\overline{\lambda} \leq 1 + \frac{p+1}{k}. \tag{4.14}$$

  Let p denote the average out-degree of super nodes, we assume $p = k \cdot (1 - \alpha)$, as the number of leaf nodes in the DMMP is much larger than the number of non-leaf nodes. Then inequality 4.16 becomes:

$$\overline{\lambda} \leq 2 + \frac{1}{k} - \alpha. \tag{4.15}$$

Reconsidering the condition in Equation 4.4 into the above inequality, we could roughly get the average link stress of DMMP.

Figure 4.8 presents the initial comparison results of the average stress between DMMP and NICE, a well-known protocol which assumes to have a good scalability. Meanwhile, the value of k in DMMP usually varies with the value of N. But the value of k in NICE is predefined (usually $k = 3$) and will not change corresponding to the group size. In contrast to NICE (the average stress is $k^2/(k-1)^2$ [112]), the stress of DMMP keeps at fairly small values (always below 1.9) regardless of the group size. It means DMMP could achieve better performance in terms of resource efficiency and scalability. However, if we set $k = N^{1/2}$, the value of stress is larger than the first setting $k = 3$. We believe it might be caused by improper value of k, which also implies choosing values of k has a great impact on the performance of DMMP and should be taken into consideration in a DMMP implementation when the value of N changes.

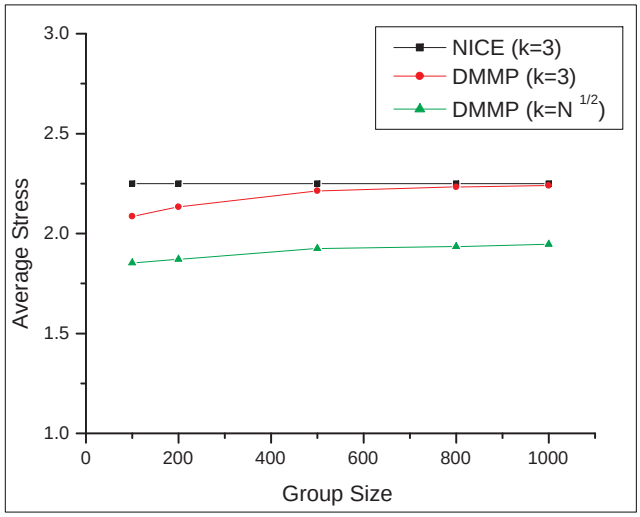

Figure 4.8: Comparison of Average Stress.

Besides above theoretical analysis, we further measure *stress* through two simulation scenarios defined in [101]. Each end host j counts the last-hop stress based on the equation:

$$s(j) = \frac{1}{N} \cdot \sum_{i=0}^{n} n(i, j) \tag{4.16}$$

where $n(i, j)$ represents the number of copies of packet i ($1 < i < N$) which is transmitted via j. Apparently, the last-hop stress caused by DMMP is nearly two if the number of group members is large.

- **Control Overhead**: DMMP is mainly used to efficiently transmit media data to end hosts. Hence, it is important to reduce the bandwidth consumed by control traffic. The control overhead of DMMP is caused by establishment and maintenance of the control plane. There are several possibilities of measuring the control overhead, for instance, counting the number of exchanged control messages. In the following performance evaluation in Section 4.4, we intend to measure the amount of bandwidth which is consumed by the control traffic.

- **Loss Rate**: While media application may be tolerant to higher loss rate than other generic applications, it is nevertheless desirable to achieve a high efficiency of media delivery. In DMMP, ungraceful leaving and network congestions may cause packet loss. For the first case, the data delivery topology can be temporarily partitioned, which requires some time to recover from the failures. In the following evaluation, packet loss rate of node i is calculated as follows:

$$l_i = \frac{N_j - n_{i,j}}{N_j} \tag{4.17}$$

  where N_j is the number of total packets of media session j sent from the source and $n_{i,j}$ is the number of packets received by the end host i.

- **Data Path Length**: Instead of measuring end-to-end (e2e) latency, we consider data path length which measures the distance between the source and a group member. The *data path length* pl_i of the group member i is defined as the number of physical links that a packet traverses till reaching node i. The high value of pl_i does not necessary mean high latency, but it implies high jitter and high possibility of packet loss.

4.4 Performance Evaluation through Simulations

We intend to evaluate and validate our proposals through the use of current powerful network tools, such as ns-2 [113] or OMNeT++ [114]. Under some carefully designed test scenarios (e.g. initialization phase, dynamic membership changes), the performance of proposed protocols can be intensively measured and compared against different design options.

4.4.1 Network Simulator

In this section, we introduce the network simulators and focus on illustrating OMNeT++ [114] which is used throughout the thesis to develop our proposed protocols.

Network Simulator Types

There are two major types of network simulator: *Customized Simulator* and *Common Simulator*. Customized simulators are designed for analyzing one particular protocol in a certain scenario. Apparently, it is difficult to compare network protocols with customized simulators unless all required protocols are implemented in the same way. For instance, NICE and Narada are implemented using customized simulators and therefore we cannot directly compare them with our simulations. Differently, common simulators can provide generic simulation framework which is independent from implementing protocols. With such a framework, protocols can be comparable since they are using the same underlying modules and interfaces. Hence, in this thesis *common simulator*, more specifically, OMNeT++ [114] is used.

Before introducing OMNeT++, in the following subsection we summarize the properties of common simulators based on our experiences.

- *Discrete event system:* Such a system changes its state at discrete points and each state change is called "event". Nothing happens between two events. At an initial phase, all events are scheduled and kept in a global data structure (e.g. event queue). Besides, each event receives a timestamp indicating when it is supposed to happen.
- *Efficiency:* Simulation of a large networks may take a long time and consume a large amount of memory. In order to simulate complex networks, the programming language in the simulator is very important.
- *Extensibility:* The advantage of using common simulator is that various of protocols can be implemented on top of the simulation framework. In this sense, the underlying module, common interfaces and comprehensive documentation make the extensibility easier.
- *Visualization:* Most common simulators can create an animation of the simulation model, which allows user to observe the protocol behavior. It facilitates debugging because visualization can help users to get an intuitive understanding of the implemented protocols.
- *Statistic Support:* To evaluate the performance of a certain protocol, collecting statistics is necessary. Common simulators explicitly offer an easy interface for recording statistics data.

We choose OMNeT++, one of the most popular, object-oriented common simulators, for our following simulation experiments. As it is written in C++ and uses a discrete event processing engine, its efficiency is rather predictable.

OMNeT++

OMNeT++ is free for academic non-profit use. There is also a commercial version called OMNEST. OMNeT++ has a much broader focus than nsnam [113]. Any system that can be modeled as a dis-

crete event system can be simulated. OMNeT++ is mostly used for computer network simulation, but it could also be used for e.g. analysis of hardware architectures. It consists of a *simulation kernel*, a *simulation library*, *component libraries* and *user interfaces* [114, pages 211–212].

- The simulation kernel mainly handles the discrete event processing. It supports distributed simulation.
- The simulation library offers support for common simulation tasks. It includes, for example, random number generators and containers, as well as classes for gathering statistics. We will elaborate a bit on the statistics support: *Output vectors* are collections of (time, value) pairs, which are recorded over the course of a simulation run. For example, assume that packet round trip times are measured regularly in a simulation. Then all the individual measurements could be stored in an output vector. The output vector writes the data to a file. The data can later be plotted using *plove*, a tool that comes with OMNeT++. The format of files generated by plove are very simple, which makes post-processing using external tools rather easy. An *output scalar* stores a single scalar value and a description string. Scalars are typically recorded at the end of a simulation run. Example: one could count the lost packets over the course of a run, and record the total number as a scalar at the end. The tool *scalars* can be used for post-processing.
- There are two alternative user interfaces; the text-based, non-interactive *Cmdenv* for batch execution, and the richer graphical user interface *Tkenv*[*]. Tkenv does not only provide animation, but also additional debugging and tracing support. Most notably, it is possible to inspect all simulation objects, such as messages, modules, parameters (see below) or output vectors, at the run time.
- The component libraries contain mostly the protocol implementations. OMNeT++ is completely independent from these libraries; in fact, it does not come with any component libraries. For example, the *INET framework* provides the essential Internet protocols. In nsnam, the basic Internet protocols are an integral part of the simulator itself, in contrast. Simulation objects are wrapped in modules. These modules can be arbitrarily combined to build more sophisticated modules. Component libraries consist of a number of related modules.

OMNeT++ simulation models are implemented as follows: *simple modules* are implemented as C++ classes, they are not composed of other modules. *Compound modules* contain other modules, which can be simple modules or compound modules. They are described using the *NED* language, which is a simple compiled programming language with a syntax similar to C. OMNeT++ provides a compiler that translates NED code into C++ code. That means, there is also C++ code for compound modules, but it is usually not written manually. All modules of a simulation are included in the *system model*, which is a module hierarchy rooted at the system model. For network simulation, the system model usually represents a network.

[*]Tkenv is based on the graphical user interface toolkit *tk* [114].

To observe large-scale network behaviors, setting up the testbeds with at least thousands of nodes is hardly feasible. Therefore, using the network simulator OMNeT++ is a good way of demonstrating the efficiency and robustness of a network protocol.

In the following analysis, we seek to evaluate the performance of DMMP via simulations in OMNeT++. We start by presenting the protocol stacks created in our simulation, which is illuminated by Section 2.5.2. Then, we introduce the fundamental items in the DMMP protocol design, and describe the network topology we use for our evaluation, as well as the parameters we choose for configuring the simulations. In the following experiments, an Intel Core 2 Duo machine with 6 GB of RAM was used.

4.4.2 Protocol Stack

In our simulation model, the physical layer stack has not been considered very detail since our focus is on the application level multicast. Thus, the bit error model is not implemented in the underlying network since the physical properties are not our focus either. When a frame is sent over a link, we assume that the transmission time relies on the link's bandwidth capacity and propagation delay. We use PPP as the default link layer protocol. Nonetheless, the implementation of the network layer is rather complicated since we have to extend the existing network layer in OMNeT++ in order to collect measurement data (e.g. stress in Section 4.2) by routers.

Figure 4.9 depicts the protocol stack of an DMMP-aware end host. PPP module is used at the link layer which connects to an access router. On top of the PPP module, there are *dmmp_networkLayer* for the network layer, namely *dmmp_ip*, and *dmmp_udp* for the transport layer. An overlay protocol *dmmp_overlay* sits on top of the *dmmp_udp*, which receives data from application layers such as "Tier 1" module and then passes to the transport layer.

Different from general protocol stack at the end host, we place an overlay layer between the application layer and the transport layer. In fact, the *dmmp_overlay* is extended from *BaseOverlay* which is the basic overlay module implemented in OverSim. Further, the *BaseOverlay* module is derived from *cModule* which is the base class for all simple modules in OMNeT++ [114]. As DMMP is proposed to support multimedia applications, we implemented a preliminary module representing a video streaming application. Such a module is only used at the *DMMP source* side. Once an initial control topology is constructed, the RP notifies the *Multimedia Application* module which generates data at a constant bit rate (e.g., r kbps) and starts the DMMP multicasting session. To avoid IP fragmentation and quicken the simulation procedure, we use a small bit rate with 64 kbps, 128 kbps and 256 kbps. Suppose it sends three data messages each second, every data message is sent with $\frac{64}{3}$ kbps, $\frac{128}{3}$ kbps or $\frac{256}{3}$ kbps.

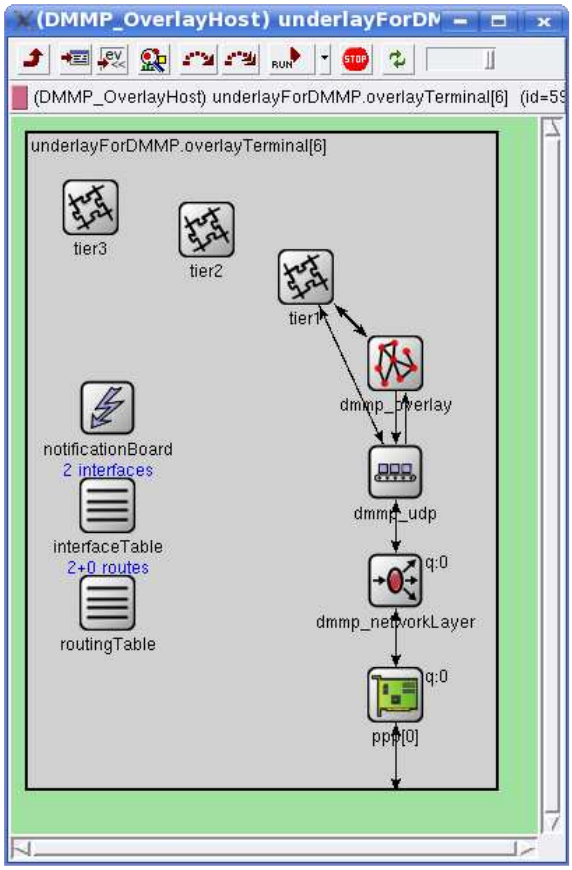

Figure 4.9: The Implemented Protocol Stack of DMMP-aware End Host.

4.4.3 Protocol design

According to the above protocol stack, the DMMP protocol is designed with considerations on 1) application layer (multimedia application); 2) dmmp overlay; 3) transport layer (DMMP_UDP); 4) network layer (DMMP_IP); and 5) link layer (PPP). As it can be seen from Figure 4.10, the "DMMP Source" and "DMMP Member" are two main classes in the DMMP simulation. The *DMMP Source* receives the video packet from the application layer, attaches with DMMP header, and then distributes it to its mesh neighbors. The mesh neighbors are implemented within the *DMMP Member* class, as well as other DMMP-aware end hosts. The IP addresses of DMMP-aware members are stored as *string*; while additional information is stored in the *MemberInfo*. These additional data includes the maximum degree of each member, available degree of mesh members, and the measured distance towards the source. Moreover, *MemberMap* is used to manage the information stored in the *MemeberInfo*. In the following simulations, the *DMMP Member* relies on several *MemberMap* triggers to keep track of the following data.

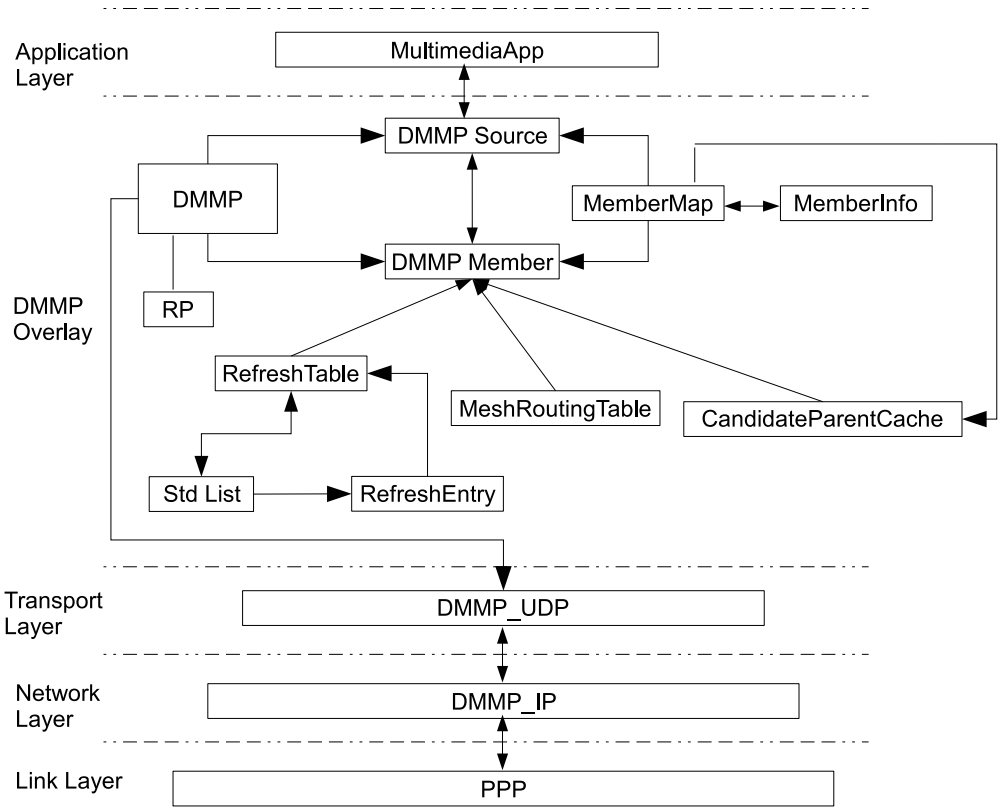

Figure 4.10: Datagram of DMMP Protocol Design.

(1) The children in the local cluster (number, IP address, etc.).

(2) The mesh neighbors (IP address, available degrees).

(3) The super nodes (IP address, number, maximum degrees).

(4) The buffered joining requests.

(5) The candidate parents.

The information of category (2) and (3) is only used by super nodes since there is no need to other group members to remember all other super nodes. However, the candidate parents are required by all non-super nodes. If a non-super node wants to rejoin the multicast session, the information of candidate parents can be very helpful. Besides, each DMMPMember maintains a *RefreshTable* (seen in Figure 4.10) that stores a number of *RefreshEntries* and a list of potentially unreachable super nodes.

4.4.4 Network Topology

In the following experimental scenarios, we use NED-oriented topology algorithm to create the underlying network topology. Such a topology generator is easy and efficient [114]. In our simulation, the underlying network is configured with two types of routers: backbone routers and access routers. Each access router is connected with one backbone router. End hosts are placed dynamically into the topology graph, depending on the churn model. Every end host is connected only with one access router via Point-to-Point Protocol (PPP), which is randomly chosen when the end host is created. However, such a topology algorithm is often used for generating a small number of routers. In order to increase the number of underlying routers, a sophisticated algorithm with dynamic router placement (which is written in C++ code) has to be used.

An alternative way is to use script languages like *perl* or *awk* to generate a complicated NED file. However, to generate large-scale networks using the script languages may be inefficient.

4.4.5 Data Model

In all the experiments, we model the scenarios with one multimedia stream source multicasting to the group. We chose a single media server to be the data source generating a constant bit rate data. In fact, DMMP framework can support multiple source media distribution, however, we focus on single source-based multicasting to evaluate the fundamental ideas. The reason is that a single source-based multicasting can be easily extended to support multiple source-oriented multicast, for example, using CDN infrastructure or aggregated multicast hierarchy (e.g., using aggregated multicast trees [51]).

4.4.6 Initial Parameter Settings

As we have discussed several application-specific parameters for DMMP in Chapter 3, we define them with the same values for the following experiments.

- Source Bit Rate: The data source generates a constant bit rate data and sends to the group. The value of bit rate varies from 64 kbps to 256 kbps.
- Link Capacity: The link capacity used in the following simulations is heterogeneous. We set totally 14 types of the channels with the variants of delays and data rate. The delay varies from 0ms to 15ms, and the data rate varies from 128 kbps to 100 Mbps, which is in accord with current link capacity distribution.
- Timeouts: T_{min} and T_{max} are used to determine how aggressive links can be added to repair potential partitions. As known that each DMMP-aware super node stores a *refresh table* with one entry for each other super node. Each entry includes an *IP address*, a *timestamp* and a *sequence number*. If an entry is not updated more than T_{min} seconds, the entry is

Table 4.1: Maximum Degree Distribution

Maximum Degree	Probability
1.0	0.4
3.0	0.2
7.0	0.2
10.0	0.1
14.0	0.1

copied to a queue. All entries are remained up to $T_{max} - T_{min}$ seconds. If the timer exceeds, a partition is assumed and the function of *handlePartition* is called.

Fiber optic cables are used for the links between routers with a propagation delay of five milliseconds and a bandwidth of 1.0 Gbps. For consistence, in the following experiments for the underlying network model, without explicit explanation all links between the access routers and the attached end-hosts have the same propagation delay and follow the same bandwidth distribution as described above. Besides, in the following scenarios all links between an access router and its attached end hosts have the same propagation delay and bandwidth. For the bandwidth distribution, the end hosts attached to the same access router have the same maximum degree. As we assume that there are a large number of *free-riders* [104] over Internet, we set the maximum degree "ONE" of the access network with the probability of 0.4. The maximum degree distribution is listed in Table 4.1.

Besides, we rely on a respective model to estimate the available bandwidth of each end host. Then, the estimated bandwidth is used to calculated the out-degree (cf. Section 4.1.1) and capacity (cf. Section 3.3.3). Though there are other possibilities of estimating available bandwidth, for example, probing based estimation, we currently use the straight-forward way to measure the available bandwidth. For an accurate bandwidth measurement, it can be considered in the future development of the simulation.

4.4.7 Data Collection

The performance metrics listed in Chapter 3 are measured through the following ways in our simulations:

- **Stress**: Since the overlay network dynamically changes, the link stress changes over time. Therefore, it requires to track each data packet sent from the source. As indicated in [101] that the number of links/routers should be counted with the ones that actively participated in data transmission. By modifying the basic model of IP routers in OMNeT++, DMMP-aware routers are able to record the number of forwarded messages. Thus, the *router stress*

can be easily measured. In the following measurement, the load of the end host is calculated as the stress of the last-hop link, so-called *link stress*.

- **Control overhead**: The control overhead is mainly caused by establishing and maintaining the DMMP-aware overlay network. To be consistent with the commonly used metrics, for example in [39], the control overhead is counted with the number of control messages.
- **Packet loss rate**: As shown in Chapter 2 that media applications are more tolerant to the packet loss than to the delay. Nevertheless, if the data delivery tree is partitioned for a long time the video service is unavoidably affected. Thus, we numerate the received video packets at each end host and accordingly determine the loss rate as indicated in Section 4.2. In the simulation, the number of packet lost can be perceived via the gap between the sequence numbers within two subsequent received packets. For instance, if the difference between the sequence number in the newly received packet and that in the last received packets is two, two packets have been lost.
- **Data path length**: To be consistent with the metrics used in [43], the data path length is recorded in a hop counter in each data packet. Once a router or an end host forwards the packet, the hop counter increases by one. Till the packet arrives at the dedicated end host, the data path length is finally remembered as the total hop counter.

Remark that we do not measure the *e2e latency* and *delay performance* (cf. Section 4.2) due to the following two reasons: 1) optimization of the e2e latency is not the first aim of DMMP, though it has already been considered during the cluster formation; 2) we measure the data path length instead, which can be regarded as a more reasonable metric for evaluating the efficiency of application level multicast since one overlay link may traverse several underlying physical links.

4.5 Simulation Results

In this section, we present the evaluation results in two envisioned scenarios: 1) dynamic scenario, which is defined to observe the impacts of different parameters on the performance of DMMP in a dynamic environment; and 2) NICE scenario, which focuses on comparing the performance of DMMP against NICE and Narada. The first scenario considers dynamic member joining and leaving since DMMP is expected to be resilient; the second scenario is specifically designed for the purpose of comparison with NICE and Narada.

4.5.1 Scenario 1: Dynamic Membership Changes

The first scenario, the *dynamic scenario*, specifically considers membership changes, since it is more realistic to design the scenarios with periodic membership changes. Thence, we used a typ-

ical churn model which is provided by a *ParetoChurn* generator following the Pareto distribution [115]. In brief, Scenario 1 consists of two subsequent phases:

- Join Phase: It is a short join phase within 100 seconds when a large number of DMMP members join the multicast session and none of them leaves the group.
- Membership Changes Phase: Members frequently join and leave the multicast session, reaching a stable equilibrium. That is, the total size of the multicast group hardly changes during the runtime of the simulation. For leaving members, both graceful and ungraceful cases are considered. For simplicity, members leave ungracefully with probability of 0.5.

Parameter Settings

The paper [97] indicates that a large amount of end users over the Internet do not have enough upstream bandwidth to support media distribution in the overlay multicast. Therefore, we set these incapable end-hosts with maximum degree of one and distribution probability of 0.4 (cf. Table 4.1). Besides, nowadays peer-to-peer applications have shown the potential that some end hosts can have a relatively higher upstream bandwidth to become super nodes. In this scenario, we simulated heterogeneous capacities of end hosts with the maximum degree distribution as listed in Table 4.1.

We used OverSim to build the underlying network with $1,500$ backbone routers and $1,000$ access routers. Besides, we configured the following parameters to identify the impacts of the size of constructed DMMP-aware mesh.

- Super node degree: 5, which is the maximum degree of super nodes.
- Target Overlay Terminal Num: $1,000$, which is the final group size of end hosts.
- Lifetime Mean: $1,200$ seconds, which is the mean value of life time of each end host.
- Refresh Timer: 1.5 seconds, which is the interval of sending refresh messages among members.

The Impact of K

As we mentioned in Section 4.1.1 that k may have a great impact on the overall performance. Here, k specifically refers to the number of interleaved spanning trees that the DMMP-aware mesh contains. Therefore, the total number of mesh links is about $n \times k$, where n is the number of super nodes. In the following analysis, the number of super nodes is set as 30 since the number of end hosts is quite large in this scenario (i.e., up to $1,000$). Note that the impacts of different numbers of super nodes have been investigated in the second part of this scenario.

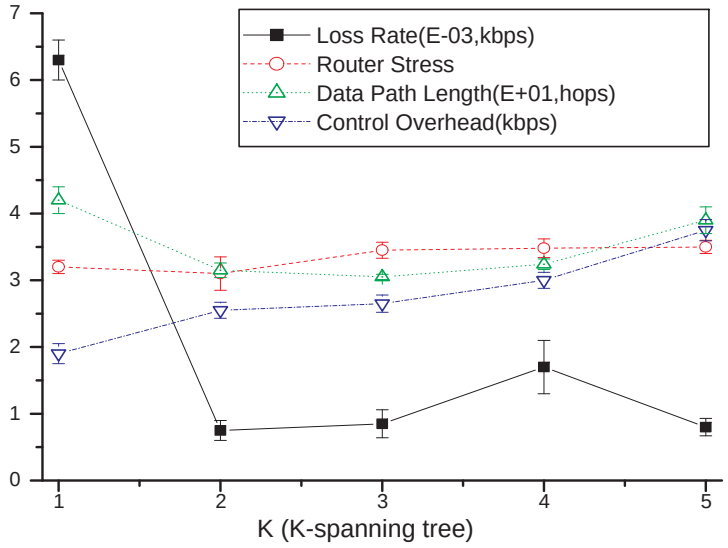

Figure 4.11: Impacts of K-spanning Tree.

In Figure 4.11, the control overhead increases from 2 kbps to 3.5 kbps as the number of mesh links increases, while the loss rate drops dramatically from 6.3 kbps to 0.7 kbps if there are 2-spanning tree in the mesh instead of 1-spanning tree. It is quite understandable since 1-spanning tree is actually a pure tree, not a mesh, which is anyway less reliable than a mesh.

For the data path length, the impacts of k are controversial: 1) the higher mesh density, the higher possibility of building good reverse path trees; 2) the mesh density is high, super nodes consume more bandwidth for maintaining reverse path forwarding. Then, the available bandwidth left for forming clusters may be much less, which leads to higher depth of local clusters. That explains why the data path length increases when k becomes three or four. Most likely, it is the same reason why the loss rate slightly increases between $k = 3$ and $k = 4$. Nevertheless, the loss rate is still quite low (around 1.5 kbps). When k changes from four to five, the loss rate drops again. We suspect that with the same selected number of super nodes (i.e. 30) in our case 5-spanning tree has reached the highest stability within the DMMP mesh.

The router stress slightly increases in the range between two and four because the high data loss rate causes a high number of redundant retransmission. However, the results with much higher value of k are not shown in the figure because the control overhead and data path length dramatically increases but the other two values are kept stable. From above observations, we can ascertain that k does have great impacts on the performance of DMMP-awareness overlay framework. For a comparatively better performance, it is suggested to use $k = 2$ or $k = 3$. In the following simulations, we, therefore, choose either $k = 2$ or $k = 3$, depending on the maximum number of super

nodes. For example, if the number of super nodes is more than 20 the value of k could be 3 in order to ensure the reliability of the mesh core.

The Number of Super Nodes

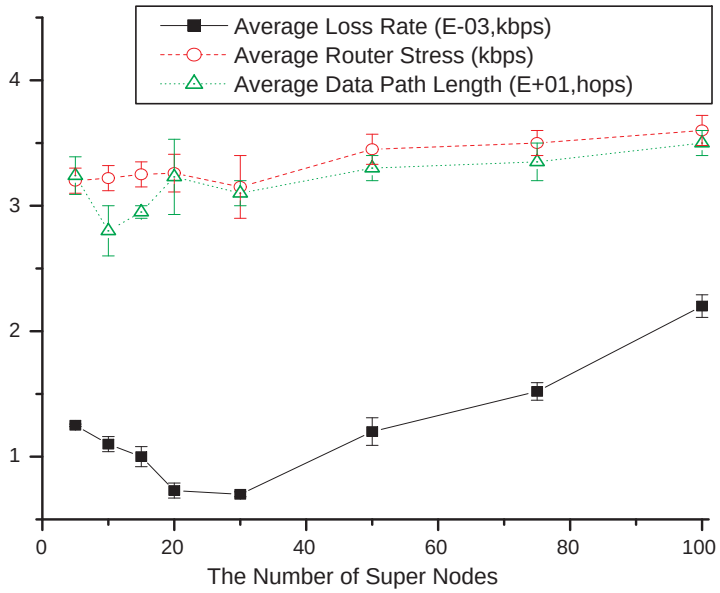

Figure 4.12: Impacts of Mesh Size.

In order to identify the impacts of the number of super nodes, we measured average loss rate, average router stress, average path length, and average control overhead. As depicted in Figure 4.12, when the maximum number of super nodes is low, such as five, the loss rate is 70% higher than that of 20. It is reasonable since when the number of super nodes is low, the clusters get very large due to the limited size of the mesh core. Moreover, large clusters are vulnerable to partitions if there are frequent membership changes.

Similarly, when the number of super nodes is less than 20 the router stress and data path length are higher (10%) than the case when the number of super nodes is 30. Such a result proves that a small mesh size leads to: 1) relatively unreliable overlay network, which eventually causes more redundant data retransmission of DVMRP (cf. Section 2.4.2); 2) deeper cluster trees, since the available number of clusters is less. However, if the number exceeds 30 both the router stress and the data path length linearly increase. We expect that the data path length can be shortened if the mesh size gets larger However, in this case, we conjecture that the established mesh core is not optimized due to the dynamic group changes. Because of the above reasons, we further developed

mesh self-improvement mechanisms (in Section 5.1.1) to optimize the mesh links through periodically adding efficient links and removing inefficient links. The optimized performance is shown in Section 5.2.

Differently, the average loss rate firstly decreases at the value of 30 and then increases linearly. It is because with 40 super nodes for $1,000$ end hosts the established DMMP-aware mesh core could be very reliable. Nevertheless, if the number of cluster members (which is equal to the number of super nodes) becomes higher than 40 the complexity of maintaining such a large-size mesh increases dramatically. This eventually causes high loss rate at the clusters. In addition, if some super nodes happen to leave ungracefully, the loss rate can be easily affected.

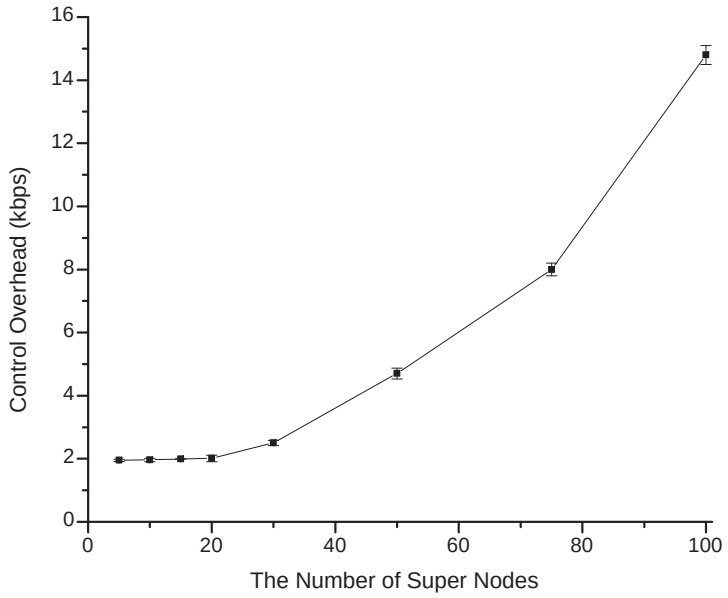

Figure 4.13: Impacts of Mesh Size on Control Overhead.

Due to different range sizes of the measurement, we separately plot the average control overhead with regards to the number of super nodes in Figure 4.13. Without a doubt, the control overhead increases in accordance with the number of group members. However, as long as the number of super nodes is smaller than 50 the control overhead is quite acceptable (less than 5 kbps). Here, for simplicity we used a relatively small source bit rate, 64 kbps. Thence, the control overhead less than 10% is acceptable, namely, less than 6.4 kbps. It mainly owes to the locality-awareness used in forming the DMMP clusters, through which the refresh messages are exchanged within each cluster. If the number of super nodes is larger than 50, it exhibits quadratic growth. The main cause is that the maintenance of such a mesh core requires a large amount of refresh messages among super nodes.

According to above observations, the number of super nodes between 20 and 30 is preferred in the following simulations as router stress, data path length, and control overhead are quite low; the loss rate is acceptable. For instance, in Scenario 2 we intend to choose 30 (at least no more than 40) as the maximum number of selected super nodes.

The Number of End Hosts

DMMP targets at supporting large-scale media distribution applications, and therefore it is necessary to evaluate the impacts of the number of end hosts. We measured the performance regarding the average control overhead, the average router stress and the average data path length, if the number of clients varies between 128 and 2, 048.

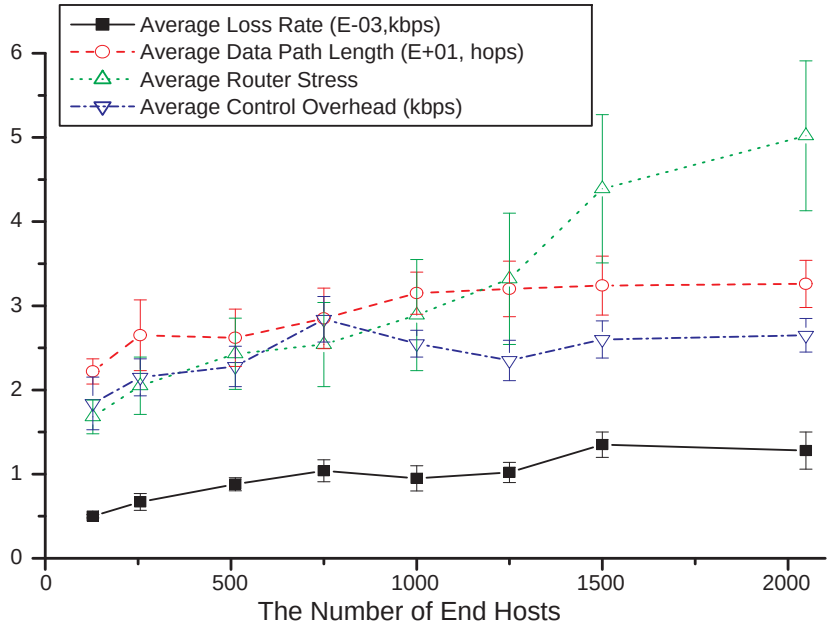

Figure 4.14: Impacts of the Number of End Hosts.

We repeated our experiments using the same underlying network topology to observe how the number of end hosts could have impacts on the performance. As shown in Figure 4.14, the average data path length, average loss rate and as well as the average control overhead are kept in a very stable way. Especially, for 2, 048 end hosts the control overhead increases less than 30% compared with the case when the number of end hosts is only 128. Moreover, it remains relatively constant along with the increasing group size. Remark that the number of super nodes has been chosen with different variables in order to be adaptive to the group size. For example, for 128 end hosts we

chose 10 as the maximum number of super nodes, whereas we set 30 when the group size reached 2,048. That may also result in some increment in both control overhead and data path length. As we presented in Section 4.5.1, with increasing number of super nodes the control overhead and data path length accordingly grow. However, the value of data path length changes very slowly though the depth of the data delivery tree unavoidably grows with larger group sizes. The above analysis indicates the DMMP has the potential to support large-scale media applications with considerations on both expanding available bandwidth and alleviating e2e service delay.

Nevertheless, when the group size gets larger than 1,200 the router stress increases in a very impressive manner. The main cause probably comes from the impact of underlying network size since 2,500 routers (with 1,500 access routers and 1,000 backbone routers) might not be capable of supporting such large-scale groups. The conjecture is identified in subsequent section.

The Impact of Underlying Network Size

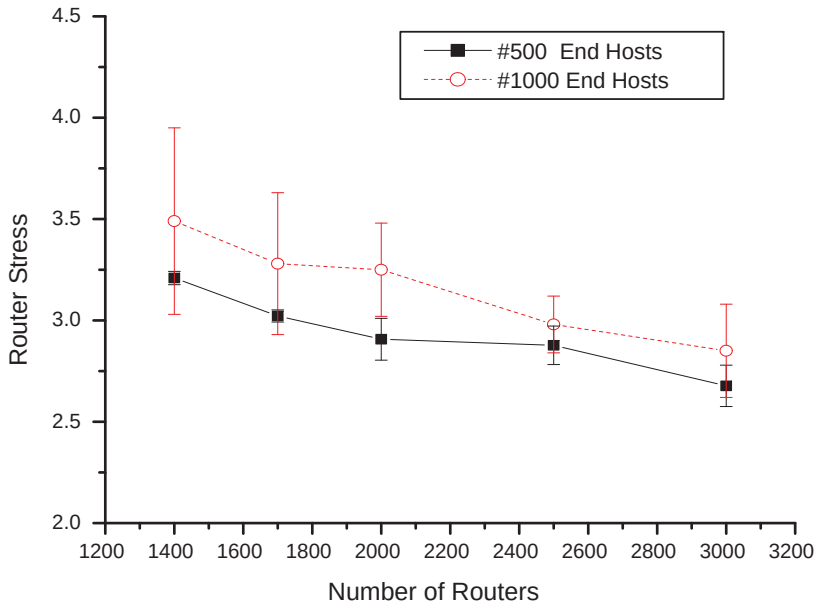

Figure 4.15: Impacts of the Underlying Network Size.

Unfortunately, Figure 4.14 implies that the router stress significantly increases with the increasing number of the group size. However, we cannot draw a statement that the efficiency of data delivery in DMMP is low since the number of underlying network size has a great impact on the performance. To validate our inference, we further measured the impacts of the size of underlying network topology in Figure 4.15. We tested 500 and 1,000 end hosts with various types of

underlying network topologies:

- 3000: 1,800 backbone routers and 1,200 access routers.
- 2500: 1,500 backbone routers and 1,000 access routers.
- 2000: 1,200 backbone routers and 800 access routers.
- 1700: 1,000 backbone routers and 700 access routers.
- 1400: 800 backbone routers and 600 access routers.

Figure 4.15 indicates that the router stress decreases dramatically when the underlying network size increases. For example, the router stress drops 30% from the number of the routers 3,000 to that of 1,400 for the same group size of 500. For the group size of 1,000, the router stress drops even more than 35% from the network size of 1,400 to the size of 3,000. Therefore, we can ascertain that the number of underlying topology can greatly affect the overlay multicast performance. In fact, it is understandable since with a larger underlying network the traffic overload can be more evenly distributed over available routers. As shown in [43], all topologies used in the simulations had 10,000 routers.

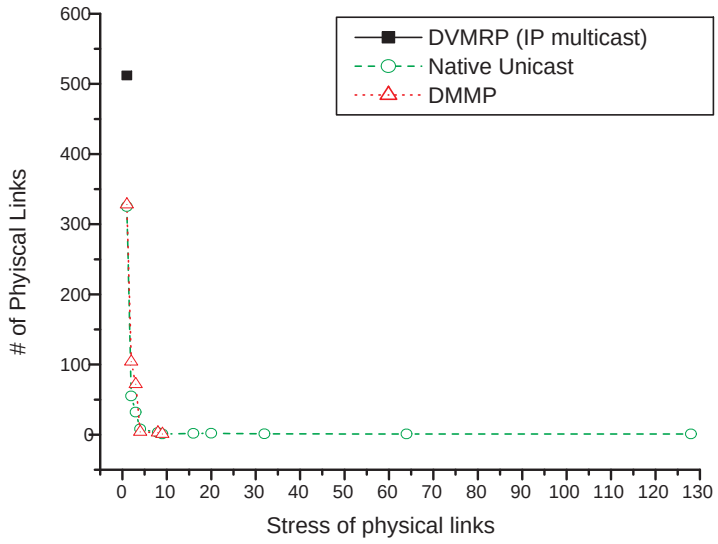

Figure 4.16: The Number of Physical Links With a Given Stress vs. Stress.

Besides, we studied the variation of physical link stress under DMMP and compared the results with that of DVMRP (shown in 2.4.2) and naive unicast in Figure 4.16. We used the similar model

with a group size of 128 members in [101]. Here, the horizontal axis represents stress and the vertical axis represents the number of physical links with a given stress. The stress of any physical link is one for DVMRP, shown as a solitary dot. Under both native unicast and DMMP, most links have a small stress, which is expected to be so. However, the significant difference lies in the tail of the plots. For the native unicast, one link may have to transmit 127 duplicated copies of the same packet (since the stress may reach 127). Though there are quite a few links having a stress above 120, native unicast may heavily overload the underlying network. DMMP, however, distributes the stress more compactly, and no physical link has a stress larger than nine. Since the traffic in the DMMP framework is well-dispersed, most of the routers experienced stress with five or even less.

4.5.2 Scenario 2: Comparison with ALM Approaches

In Scenario 2, we intend to compare the performance of DMMP with that of NICE which is claimed to have a good scalability, and Narada which is one of the first application level multicast protocols. Very careful readers might notice that, we haven't considered some overlay multicast approaches (e.g. OMNI, TOMA) as mentioned in Section 2.4.3. This is because the performance of these approaches largely depends on the delicately deployed infrastructure nodes (e.g. MSN). Different from their design philosophy, DMMP-aware nodes are self-organized to support media distribution system, without any special infrastructure support from the underlying network. For fairness and reasonability, NICE and Narada are chosen as two benchmarks for the comparison analysis, which are self-organizing systems as well. Further, most recent systems [39], [37] origin from above two systems, and additional features in the new systems are not essential for the following comparisons. Other systems such as [116], [117] are not feasible for the general topology configuration in following simulations.

In our simulation, we reproduce the scenario implemented with the same module called *DMMP_NICEChurn* following the description in [43]. It can generate a churn of arrival and departure rate of the end hosts during a runtime. In brief, the NICE churn model includes three subsequent phases:

- Join Phase: During the first 200 seconds, a set of 128 end hosts uniformly join the multicast session.
- Stabilization Phase: Within $1,800$ seconds the DMMP-aware overlay is kept stabilized. There are no membership changes during this phase, which is the same period mentioned in [43].
- Leaving Phase: After the stabilization, 16 randomly selected members leave over 10 seconds. This phase repeated four more times at 100 second intervals. To verify the resilience of DMMP, ungraceful leaving has been simulated during the four-time leaving phases, and therefore the total simulation time is $2,400$ seconds.

Simulation Setup

We relied on OverSim topology generator to configure the underlying network with 5,000 routers with an average node degree between 3 and 4. In our simulation setup, we were unable to simulate with with topology of 10,000 routers or larger since the OverSim configuration for 5,000 routers and 1,024 end hosts consumed more than 6 GB of the memory. However, the results from our simulation can be still comparable to that of NICE or Narada since the following metrics are measured in the "average" and "percentage" way. For example, the average router stress refers to the average duplicated numbers of the same packet traversed through each of the underlying routers. Furthermore, we emphasize that the purpose of this performance evaluation intends to show the capability and potentials of DMMP in supporting large-scale media services.

The number of these end hosts in the multicast group varied between 8 and 512 for different experiments. In our simulations, in order to be comparable with NICE and Narada we only modeled loss-less links: there is no data loss due to network congestion, and no notion of background traffic. However, any data packet is considered as lost whenever DMMP fails to provide a valid path from the source to a receiver, or a duplicated data packet is received through different paths. Besides, we configured the following parameters for the simulation set up:

- Super Node Max Num: 30 is selected, since in this scenario DMMP framework is expected to support a large number of end hosts. With regard to the group size, the maximum size of super node is accordingly changed. That is, the maximum size of super node is bounded to no larger than 10% of group size when the group size is smaller than 300.
- Target Overlay Terminal Num: (8, 512), which is the total group size of end hosts. Since the maximum number of end hosts can be support by Narada is 512, we only test up to 512 end hosts in this scenario.
- Graceful Leaving Ratio: 50% of leaving is ungraceful leaving with 0.2 seconds of graceful leaving delay. Within this delay, these nodes are supposed to notify the neighbors and waits for the leaving notification.
- Refresh Timer: 5 seconds, which is set with the same value of *HeartBeat* period for NICE.

Data Path Quality

Figure 4.17 and 4.18 show the router stress and link stress for different protocols as the group member size evolves. For each metric, we present both the mean valuNICE and Naradae and the standard deviation. Note the aggregated results for NICE and Narada are obtained from [43] which specifically presents the router stress and link stress using 10,000 routers. In our DMMP implementation, we only used 5,000 routers due to the limitation of hardware support.

As explained in Chapter 3, DMMP members aggressively find good points of attachment (e.g. high bandwidth support and relatively low e2e latency) to join the group in the overlay topology.

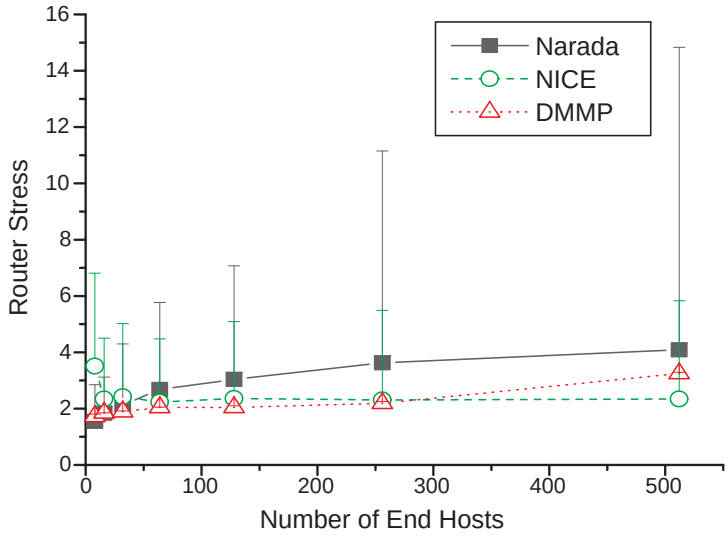

Figure 4.17: Comparison of Router Stress.

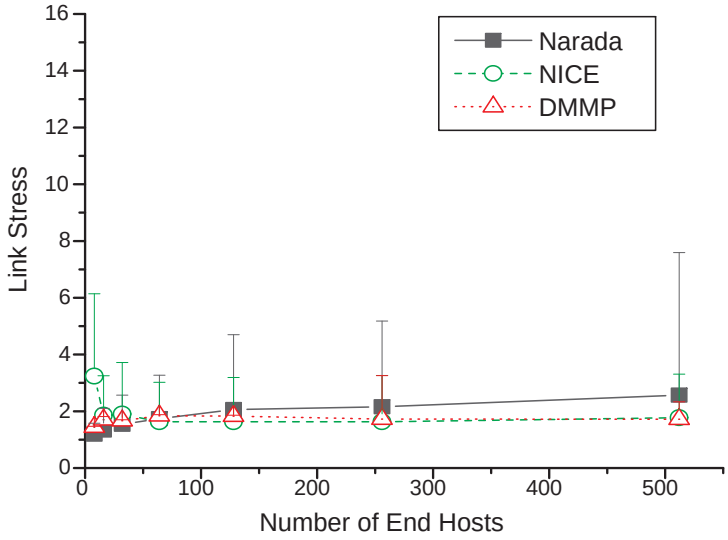

Figure 4.18: Comparison of Link Stress.

Because of the locality-awareness considerations, the nearby members in the DMMP framework are very likely grouped into the same cluster, and therefore the router stress and link stress would

be kept relatively low. Moreover, some high capacity nodes perform as super nodes which can distribute media data in a highly decentralized way. Therefore, the quality of data path in DMMP is expected to be relatively high and kept stable.

As expected, the router stress of DMMP is quite stable (as the deviation of stress for DMMP is very less), comparing with NICE or Narada. However, NICE eventually has lower router stress (30% less than DMMP when the number of end hosts is 512). The main cause for the quality degradation is that the established DMMP-aware overlay mesh may not be optimized during the runtime. Therefore, we propose a self-improved protocol for DMMP in Section 5.1.1 to improve the quality of service hierarchy. Nevertheless, we can argue that DMMP only relies on half of the routers to achieve competitive performance as proved in Section 4.5.1 that the underlying network size has a great impact on the performance, especially for the router stress,

Another fact is that the performance of NICE is much worse than DMMP and Narada when the number of group size is less than 50 and the performance of Narada degrades very quickly when the group size increases. However, DMMP is more adaptive to the variance of group size. Moreover, the link stress of DMMP is very competitive to that of NICE although we used only $5,000$ routers to afford the service. It is also interesting that the link stress is kept quite constant even when the group size evolves. We believe that the dynamic mesh-based overlay hierarchy is efficient for distributing media data to a large number of end hosts.

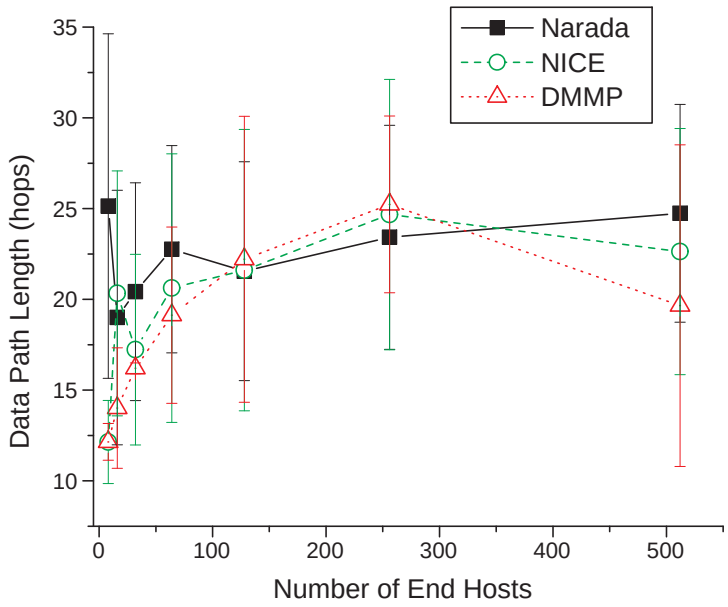

Figure 4.19: Comparison of Data Path Length.

Figure 4.19 plots the average path length and the deviations for Narada, NICE and DMMP with different group sizes. As DMMP does not target at optimizing the delivery path length to receivers, such a result is acceptable. In some cases, DMMP has shorter data path length (e.g. when the group size is lower than 100). In other cases, the data path length of DMMP is a little higher than Narada and NICE (e.g. when the group size is larger than 128). Surprisingly, with 512 end hosts DMMP has shorter data path length compared with the performance offered by NICE or Narada. It is mainly owing to super node selection strategy in our DMMP framework. When the group size gets larger, the number of super node is accordingly increased (cf. Section 4.5.2). With more super nodes, end hosts have higher probability of finding better serving parent nodes who can provide shorter delivery paths.

The last important issue observed from the figure is that the data path length of DMMP is relatively unstable. We believe that established DMMP clusters may not be optimized due to the membership changes, similar to which has been mentioned in "the impacts of the number of super nodes" in Scenario 1. Therefore, in Section 5.1.2 we propose and develop self-improvement mechanisms for the DMMP framework to optimize the established overlay clusters.

Resilience and Control Overhead

DMMP is designed to be resilient to dynamic changes. To investigate the impacts of end host changes, we present results observed from the third part of our scenario: starting from simulation time $2,000$ seconds, a set of 16 members leave the group over the 10-second period. In those leaving cases, there are 50% of ungraceful leaving (cf. Section 3.5.5). This procedure is repeated four times and simulation ends at $2,400$ second. Note we reduce the number of leaving member to 25% of the existing members when the group size is less than 64. For example, when the group size is 16 each time only four members leaves the group instead of 16.

Figure 4.20 shows the mean value and the standard deviation of control overhead at the access links of the end hosts. Each symbol in the plot represents the average value of the control traffic in form of $kbps$ sent and received by the group members. Since the control overhead for NICE and DMMP is very similar, we draw another Figure 4.21 with considering only NICE and DMMP. Obviously, the control overhead of Narada is much higher than both NICE and DMMP. For example, in order to maintain a group with 500 end hosts the control overhead exceeds 200 kbps. The simulation results again prove the statement mentioned in Section 2.4.3 that Narada is only useful for small-size or medium-size multicast group. Otherwise, the control overhead to maintain a large group will overwhelm the network resource.

When the group size is less than 256, the control overhead of DMMP is comparatively higher than that of NICE because of the large number of selected super nodes. The frequent message exchanges among mesh members result in a bit higher control overhead.

However, the average control overhead decreases when the group size is more than 350 stemming from the fact that we only selected the same number (i.e. 30 as identified in Section 4.5.1) of

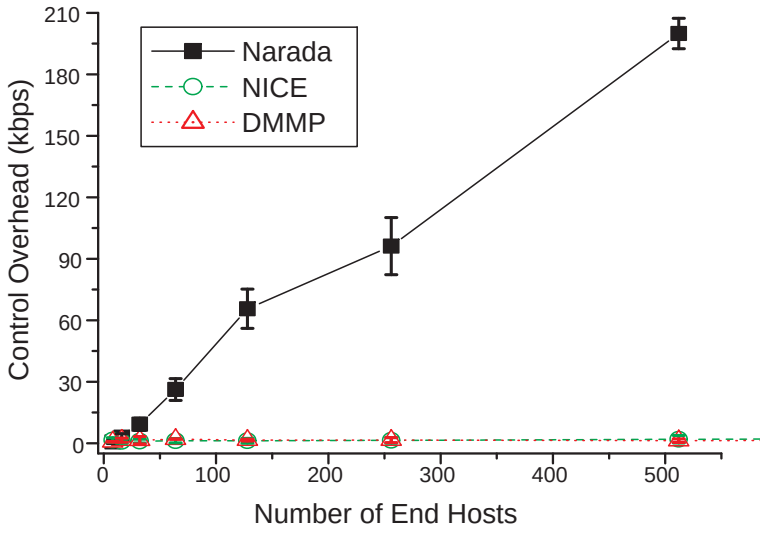

Figure 4.20: Comparison of Control Overhead.

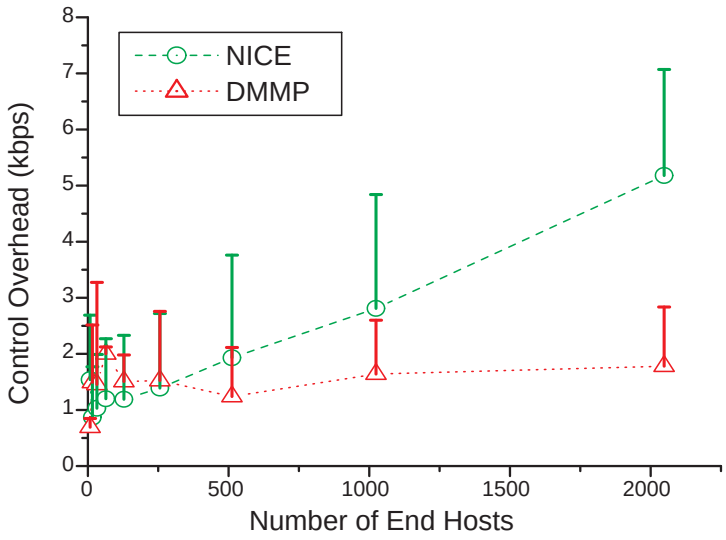

Figure 4.21: Comparison of Control Overhead.

super nodes for both group size of 256 and 512. Apparently, for the former group size more than 10% group members are acted as super nodes, which increase the average cost of maintaining the

mesh core. However, for the latter the number of super nodes is less than 6%, and thus the average control overhead drops a little. It is the same reason why the the data path length is not stable and the average loss rate increases more when the group size is 512. Alternatively, the data path length and the loss rate can be optimized through carefully selected number of super nodes and shortening the refresh intervals (current setting is five seconds). If do so, the control overhead will unavoidably increase since more message exchanges are required to maintain the mesh core.

When the group size is larger than 350 the average control overhead of DMMP decreases 20%, whereas the control overhead of NICE increases more than 60%. The fact is resulted from the following three reasons. Firstly, if a new member wants to join the NICE overlay, it must start from the basic layer. If there are a small number of existing users, the layered hierarchy contains a limited number of layers and control overhead is not high. When the group size gets larger, the NICE layer becomes much higher. In order to join the group, the message exchanges from the basic layer till the highest layer could be very high. Second, each NICE cluster needs to periodically track its size (cf. Section 2.4.3), and based on the detected size it either splits itself or merges with another small cluster. Such an operation is useful to keep the hierarchy stable. However, if the group size is very large the control overhead for such an operation is unavoidably high. Third, if there are several leaders (which are the members located in the core of the layers) happen to leave at the same time, the control overhead and loss rate grows rapidly because all partitioned nodes need to rejoin the session from the basic layer again. It can explain why the control overhead of NICE increases so fast when the number of group size becomes large.

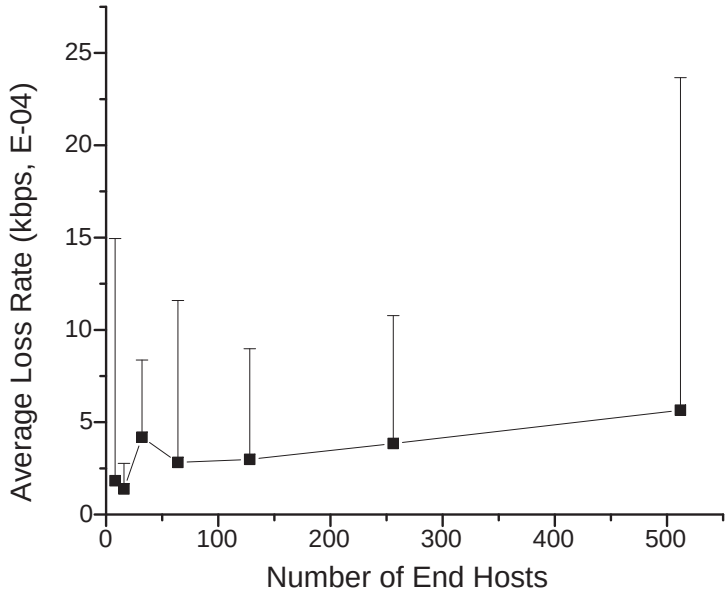

Figure 4.22: Comparison of Average Loss Rate.

Particularly, we measured the average loss rate during the four-time leaving phases. Nonetheless, the packet loss rate is not evaluated in NICE due to the unavailability of the measurement model in NICE. As shown in Figure 4.22, the average loss rate increases dramatically when the group size evolves to 32 since each time there are more than 8 end hosts (25%) leave the group. Even when those group members receive the data, they may not be able to deliver the data to its neighbors before the leaving. Nonetheless, the average packet loss in DMMP framework is still low. Note that there is no data loss introduced by links, network congestions or background traffic. The results in this scenario may not be completely realistic. However, we still can argue that the stable packet loss rate implies the potential resilience of DMMP for supporting large-scale media services.

Observing the above results from the multiple experiments over different group sizes, we can conclude as follows.

- The data path quality of DMMP is much better than that of Narada, and very competitive to that of NICE with much less underlying network support.
- DMMP is more adaptive to different group sizes than both NICE and Narada since the efficiency of data delivery in term of router stress and link stress of DMMP are maintained relatively stable when the group size grows.
- The control overhead and average loss rate varies in a relatively stable way as the deviations of both metrics are much less than that of NICE and Narada. Thus, DMMP has the potential of being resilient to dynamic changes in the circumstance of large-scale group size. While the control overhead of NICE and Narada increases in an impressive way when the group size gets larger than 300, the control overhead of DMMP is less affected and kept stable.
- It is essential to balance the tradeoff between control overhead, and high quality of data delivery because maintenance of a large-size mesh core strengthens DMMP-aware overlay but increases the cost, especially the control overhead.

4.6 Summary

In Chapter 3, we proposed a novel Dynamic Mesh-based overlay Multicast protocol (DMMP) framework to overcome delivery efficiency, scalability and deployment issues for supporting media streaming applications over the Internet. Under this framework, some selected nodes in a physical region self-organize into an overlay mesh, which is dynamically maintained according to their resource availability and willingness of contributions to the network. Note that the DMMP framework currently considers source-specific multicast [118], any-source multicast is left for future studies.

The theoretical analysis proves that it is possible to construct such an overlay hierarchy for DMMP although there are a large number of leaf nodes (i.e., over 40%) in the network. Secondly, the tree

depth has a great impact on the overall latency of the multicast tree, and hence we construct multicast tree within each cluster as short as possible. To address the instability and unreliability aspects of end hosts, DMMP periodically pushes high-capacity nodes to the higher level of the tree. Moreover, our preliminary analytical results show good stress for large multicast groups. Our analysis is based on the assumption of a large member population uniformly distributed in the network. However, in practice, the uniformity assumption may not hold.

Therefore, we implemented DMMP protocol over OMNeT++ and designed two scenarios to evaluate the performance of DMMP (e.g., stress, control overhead). Through the first scenario based analysis, it has been validated that the number of interleaved spanning trees, the number of super nodes, the number of joined end hosts, and the underlying network size have a great impact on the performance of the DMMP framework.

Based on the simulation results of the second scenario, we ascertain that DMMP can achieve a much better performance than Narada, and a very competitive performance as NICE. In terms of the stress, control overhead and loss rate, DMMP has a very similar performance as that of NICE with much less underlying network support. Secondly, DMMP framework is more adaptive to group sizes than Narada and NICE, and the performance of DMMP is relatively stable even when the group size gets larger. Thus, we can argue that the dynamic two-tier hierarchy can be regarded as an efficient way of supporting large-scale overlay networks. Lastly, it is noted that there is always a tradeoff between having a high quality of data delivery and a high control overhead. Even so, DMMP has been benefit from the locality-awareness and dynamic mesh-based overlay construction strategy to balance the quality of service and the cost in an efficient way.

Chapter 5

Self-improved DMMP Protocol (DMMP+)

Chapter 3 introduced a two-tier overlay hierarchy consisting of a mesh core and several clusters. Through performance analysis in Chapter 4, the DMMP framework is identified to have the potential of supporting large-scale media distribution services. Nevertheless, the data path quality may be instable and the average loss rate may largely increase because of dynamic membership changes (e.g. new member joining). In this chapter, we intend to extend the basic DMMP protocol with self-improvement algorithms to provide a better service delivery hierarchy, regarding its scalability, efficiency and reliability. The proposed the self-improvement mechanisms are actually protocol-independent. They can be applied to any overlay hierarchy, such as tree-based, mesh-based structure (cf. Section 2.4.3). In this thesis, we deploy them into the DMMP framework to clarify how the self-improvement mechanisms can be used to improve the quality of the data delivery hierarchy.

The remainder of the chapter is organized as follows. Section 5.1 presents two self-improvement techniques proposed by the DMMP+ protocol, namely, mesh self-improvement and cluster self-improvement. Section 5.2 focuses on proving the effectiveness of the DMMP+ protocol. Meanwhile, we introduce a different topology generator - GT-ITM [119] - to produce a more realistic Internet-like topology. For the performance analysis, two aspects are considered through the simulation, i.e., data path quality, control overhead and packet loss. The comparisons in Section 5.2 validate the correctness of our hypothesis that self-improvement mechanisms not only enhance the reliability but improve the efficiency of the data delivery hierarchy. Section 5.3 gives a short summary of this chapter.

5.1 Self-improvement Mechanisms

The DMMP framework relies on a dynamic mesh to distribute media data to a large number of end hosts. However, the constructed mesh may be sub-optimal, because (i) initial neighbor selection during a member joining the group is given limited topology information; (ii) dynamic member changes due to group member joining or leaving; (iii) underlying network conditions, such as routing, traffic load may vary. Motivated by the refinement studies in [45] and [47], DMMP+ provides two self-improvement mechanisms in order to gracefully enhance the performance of the DMMP framework (e.g., to expand available bandwidth over the network).

5.1.1 Mesh Self-improvement

In DMMP+, the mesh members periodically exchange messages with each other in order to track the dynamic changes of other mesh members. As we mentioned in Section 4.5.1, the number of interleaved spanning tree has a great impact on the efficiency of the data delivery. However, due to dynamic membership changes the quality of the established mesh may degrade. DMMP+ allows an incremental improvement of mesh quality by adding additional high-performance links and dropping low-performance links.

Addition of Mesh Link

Mesh members probe each other at random and new links can be added depending on the perceived *utility* gain. Here, *utility* is defined in Algorithm 5.1.1 to reflect the mesh quality. Following this algorithm, members continue to monitor the *utility* of the existing links, and drop links which are perceived as useless. Our target here is to provide a good-quality mesh which can ensure between any pair of mesh members, the paths along the mesh can provide better performance comparable to the performance provided by the unicast path between them. To illustrate the idea, we provide an example of adding useful links between a pair of DMMP mesh members.

Let v be a super node, v randomly selects a non-adjacent super node, say u, and sends a request for u's routing table as well as it's current available mesh degree. On receiving the routing table, v evaluates the utility of the link $\{u, v\}$, namely, $U(u, v)$. Suppose latency is used as the routing metric, v measures the Round-Trip Time (RTT) between u and v. In this case, the utility $U(u, v)$ is calculated in the algorithm 5.1.1.

The above algorithm is similar to the one proposed in [101], however, different from their design function we extended the algorithm with *gain* considerations: v evaluates how the performance of its routes could be significantly changed if link $\{u, v\}$ is added. That is, we propose a utility threshold to evaluate whether the adding link between them is desirable. Such a threshold depends on the number of existing super nodes, and the available mesh degree of u and v. If $U(u, v)$ is above the threshold, v should send a request, "AddLinkMsg" in our implementation, to u if u still

Algorithm 5.1: Utility for Mesh Self-Improvement

for for member v **do**
  $U(u,v) = 0$
  CL = current latency between u and v along the mesh improvement
  NL = new latency between u and v along the mesh
  if new edge u-v is added
  if $NL < CL$ **then**
    $U(u,v) += \frac{CL - NL}{CL}$
  end if
end for

have available mesh degree. The complete message flow is shown in Figure 5.1.

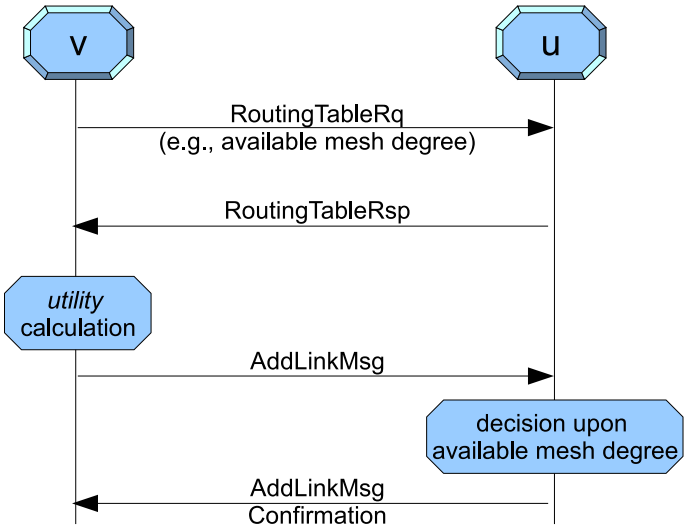

Figure 5.1: Example of Adding Mesh Links.

For simplicity, u will not refuse to add the link but at the same time if there is a super node, say s, which adds a link $\{u, s\}$, the super node u breaks the connection to a non-super node if possible. If it fails, since u's consensus threshold increases u soon drops a link anyway.

Deletion of Mesh Link

Besides periodical adding links to the mesh, each mesh member periodically considers to drop an inefficient link. Dropping a link is easier than adding an extra link since it may require no message exchange. For example, v updates the last routing updates received from its neighbors

and computes the consensus cost of each link, such as $\{u, v\}$ as described in [101]. However, in Narada each group member can be selected as a source. In our case, currently we only consider one source and therefore, it can be useful to focus on optimizing the routes from receivers to the source. During our implementation, it is not wise to drop a link which is along the shortest path between v and the source, or between u and the source. Otherwise, it impacts on all optimal links of v and u. As a complimentary to [101], we propose the approach as follows.

Suppose c is the minimum value of the computed consensus costs. If c is below the consensus threshold that relies on v's current available mesh degree and the number of super nodes, the corresponding link $\{u, v\}$ is dropped. To drop the link, v notifies u by sending a "DropLinkMsg", and both of their super nodes update their internal database, in particular, the routing tables. If the "DropLinkMsg" is somewhere lost, v sends the message again when it receives the next refresh message from u.

5.1.2 Cluster Self-improvement

We suggest the self-improvement mechanisms since a cluster member having a higher capacity than its parent node can be promoted. Basically, the direct children and their parent can swap their positions. After promotion, the former child becomes the new parent and its former parent may become the current child. However, there are still some factors affect the mechanism:

- the number of nodes involved in the promotion
- the reliability of participated nodes
- after promotion, the re-construction of the existing tree may be complicated since the promoted child may have not enough bandwidth to accept all existing end hosts as its children.

For simplicity, we describe the idea by taking the following example. In Figure 5.2, suppose that node 1.1.2 has much higher capacity than its parent 1.1 based on the capacity comparison. Then, node 1.1.2 sends a promotion request to node 1.1. After a certain proof (e.g. authority check), node 1.1 acknowledges the request and sends back a status report which contains the address of node s. Here, it is necessary that node 1.1 waits till node s has received the breakup request. Otherwise, the join request from 1.1.2 may arrive earlier, which will cause a loop in the overlay tree.

Then, node 1.1 breaks the connections with node s and 1.1.2. However, node 1.1 keeps node s as its backup parent in case node 1.1.2 is leaving or unreachable. Moreover, node s considers node 1.1 as it temporary child. At the same time, node 1.1.2 contacts node s and notifies node 1.1 to be its child. Once node 1.1 receives the notification and rejoins the tree as the child of node 1.1.2, it may break the connection with node 1.1.1 if node 1.1.2 still has available capacity. In the following example, node 1.1.2 can support at least three children. Therefore, after the first swap, the node 1.1.1 requests to join as one child of node 1.1.2.

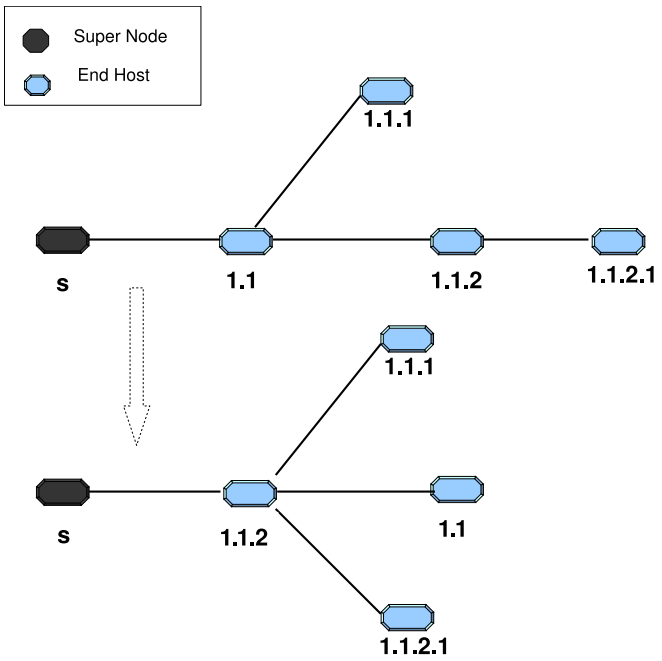

Figure 5.2: Example of Cluster Self-improvement.

The message flow of the promotion example is shown in Figure 5.3.

Due to our strategy of constructing the overlay hierarchy, nodes at the bottom level of the overlay are either transient nodes, leaf nodes or new comers. The above cluster improvement algorithm allows the newcomers who have higher capacities could “climb” from the bottom to a higher level after some switching stages. In reality, it is very important for new customers who have high capacity and willing to share their resource can get better quality of service. For example, a newcomer at the lower level could switch with its parent if its capacity exceeds (over a predefined threshold) the current parent. Nevertheless, an appropriate threshold (as defined in Section 5.2.1) should be chosen to avoid unnecessary switching since if the child has a smaller bandwidth support, it will be ultimately placed below the parent. The main goal of doing this is to reduce the impacts of frequent changes in the overlay so that only a small part of the overlay multicast tree will be affected and needs to be re-constructed after dynamic changes.

5.1.3 Extended Messages

To support self-improvement mechanisms described above, we need to extend the messages defined in Section 3.4. Table 5.1 lists the extended DMMP messages.

Here, *RoutingTableReq* and *RoutingTableResp* messages are used to obtain the information for

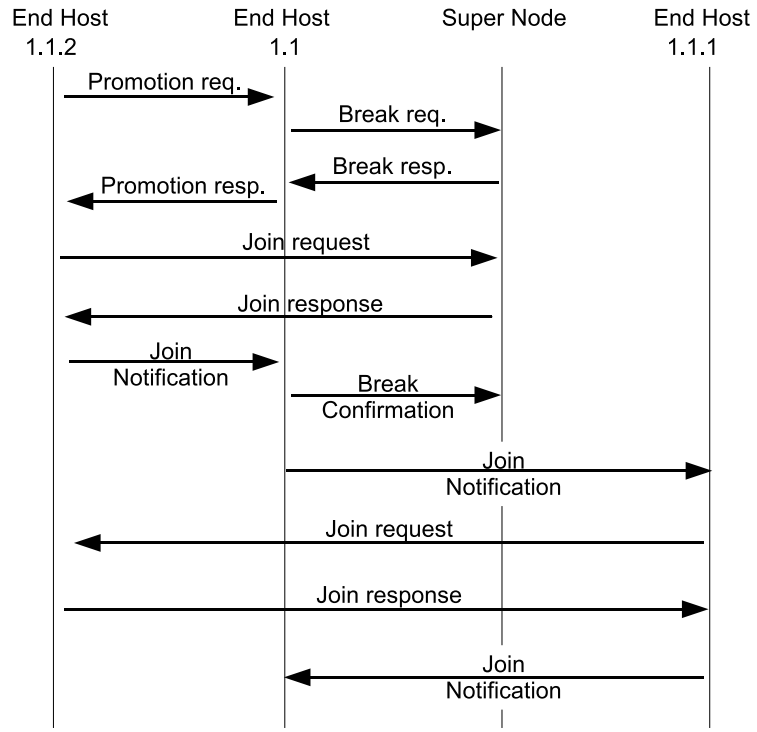

Figure 5.3: Message Flow of Self-improvement.

evaluating random links. If the operation of adding links is decided by one mesh member, it sends *AddLinkMsg* for adding the evaluated link to another mesh member. Similarly, *DropLinkMsg* is used to request for dropping unuseful links. Note the first six messages are extended to support mesh self-improvement mechanisms. Therefore, they are only exchanged between mesh members.

For the cluster self-improvement mechanisms, the last six messages are defined. Meanwhile, *Promotion Request* and *Promotion Response* messages are used to determine whether the swap between the parent node and the child node is allowed. Once the promotion is confirmed, the child node tries to break the existing links with other nodes through *Break Request* and *Break Response* messages. In addition, *Join Notification* is used to notify the partitioned nodes to rejoin the group. Then, these partitioned nodes can request the new parent for joining the group. When such a joining procedure fails, they can follow the initial join phase (cf. Section 3.5.3).

5.2 Performance Evaluation

As observed in Section 4.5.1, the established mesh core may not be optimal due to the frequent member joining/leaving. Moreover, Section 4.5.2 implies the quality of path length may be also

Table 5.1: Extended DMMP Messages for Self-improvement Mechanisms

Messages	From	To	Operation
RoutingTableReq.	Mesh Member	Mesh Member	Links Evaluation
RoutingTableResp.	Mesh Member	Mesh Member	
AddLinkMsg	Mesh Member	Mesh Member	Adding Links
AddLinkMsg Confirmation	Mesh Member	Mesh Member	
DropLinkMsg	Mesh Member	Mesh Member	Dropping Links
DropLinkMsg Confirmation	Mesh Member	Mesh Member	
Promotion Req.	Cluster Member	Cluster Member	Promotion
Promotion Resp.	Cluster Member	Cluster Member	
Break Req.	Cluster Member	Cluster Member	Break Connection
Break Resp.	Cluster Member	Cluster Member	
Break Confirmation	Cluster Member	Cluster Member	
Join Notification	Cluster Member	Cluster Member	Member Join

affected. Therefore, we implement the mesh self-improvement mechanisms to gradually enhance the mesh performance. For the same reason, we implemented the self-improvement in each cluster. In this section, we validate the effectiveness of the two self-improvement mechanisms through measuring the data path length, stress, loss rate and control overhead.

5.2.1 Simulation Setup

In the following simulations, we use two approaches to build the underlying network topologies: 1) NED-oriented topology generation; 2) GT-ITM generator. As mentioned in Section 4.4.4, the NED-oriented topology generator is easy and efficient but less realistic; whereas the latter is commonly a representative abstraction of the real Internet.

Besides the common parameters defined in Section 4.4.6 and Section 4.5.2, the following parameters are configured for the performance evaluation of DMMP+.

- Target Overlay Terminal Num: the value varies between 128 and $2,048$.
- Threshold for Promotion: threshold = $100,000$ bit = 100 kb. As the multimedia session sends data at a constant bit rate of 64, 128 and 256 kbps, we set rather low bandwidth values for the end hosts. It is possible to support up to 2 Mbps bit rate, however, the speed of simulation drops dramatically.
- Refresh Timer: 3.5 seconds. It is used to periodically trigger the message exchanges among

cluster members.

- Refresh Mesh Interval: 2.5 seconds. It defines the period for refreshing the mesh core.
- Utility Interval: 5.0 seconds, which is a period used to consider adding/deleting random links.

In the following two comparisons, we use the similar scenario used in Section 4.5.1. The entire set of members join in the first 200 seconds, and we run the simulation for another 1,800 seconds to allow the topology to be stabilized. That is, the performance of DMMP+ will be evaluated through two phases: joining phase and stabilization phase.

5.2.2 Scenario 1: NED-oriented Topology

To be consistent with simulation results in Chapter 4, we first rely on NED-generated topology to configure the underlying network. As identified that Narada does not scale well (which can only support up to 512 end hosts), in this scenario we only perform the experiments and compare with NICE. We measured the router stress, link stress and control overhead regarding different group sizes.

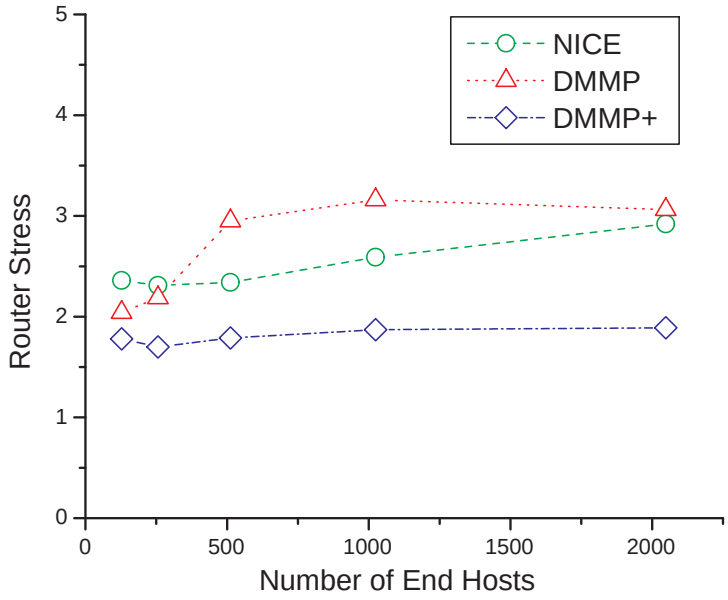

Figure 5.4: Comparison of Router Stress.

Stress

Figure 5.4 shows the comparison of router stress with various group sizes. Compared with NICE, DMMP+ has much less router stress (about 20%). Besides, the performance of DMMP+ is quite stable regardless of the group size changes. When the group size gets larger, the router stress caused by NICE increases whereas the performance of DMMP+ still keeps stable. In a stabilized circumstance, the self-optimized DMMP-aware mesh and clusters can provide high efficiency of data delivery by adding efficient links and deleting inefficient links (cf. Section 5.1) and as well as promoting high capable nodes near the overlay core. Differently, when the group size becomes large NICE has even deeper layered hierarchy and more separated clusters within each layer, which enlarges the possibility of redundant transmission over the routers.

Similar to which has been observed in Scenario 4.5.2, the router stress caused by DMMP is relatively stable. However, when the group size becomes larger than 300 the performance of DMMP degrades and becomes worse than that of NICE. It is mainly because of duplicated transmission through unoptimized overlay hierarchy.

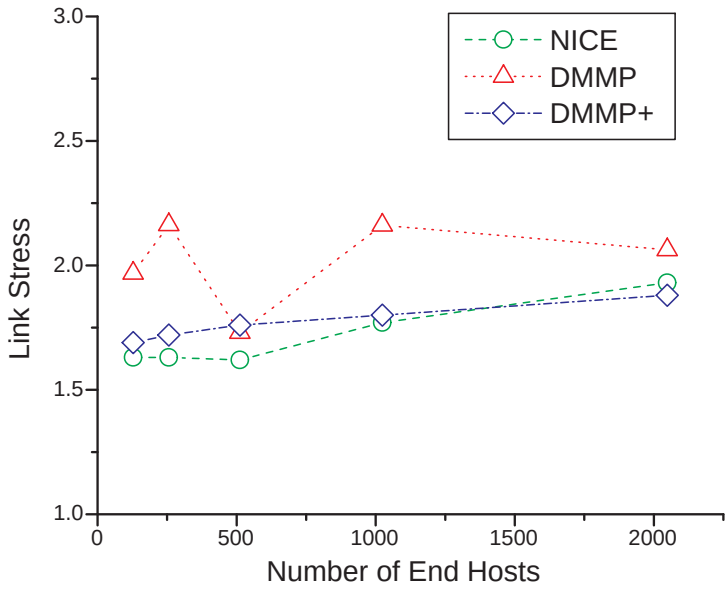

Figure 5.5: Comparison of Link Stress.

Figure 5.5 depicts the comparsion of link stress among NICE, DMMP and DMMP+. Obviously, DMMP+ can achieve much better performance than DMMP. Moreover, due to the gradually optimized overlay hierarchy the link stress of DMMP+ is quite stable. Such an observation exactly demonstrates that the DMMP+ protocol is very helpful to improve the stability and efficiency of

the data delivery in the DMMP framework.

Compared with NICE, DMMP+ has comparatively less link stress though its underlying network is configured with 5,000 routers. Especially when the group size is less than 1,000, the router stress of the DMMP+ protocol is up to 15% less than that of NICE. When the group size is larger than 1,500, the performance of NICE is a bit better than DMMP+. However, as identified in Section 4.5.1 that the link stress is impacted by the underlying topology, we can argue that the link stress caused by DMMP+ is very competitive and stable in contrast to the performance of NICE.

Control Overhead

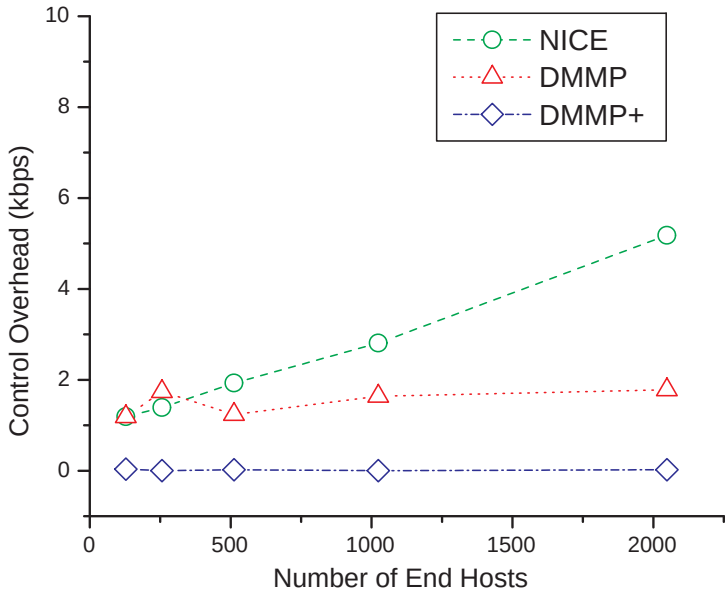

Figure 5.6: Comparison of Control Overhead.

We measured the control overhead with regards to different group sizes for NICE, DMMP and DMMP+. The control overhead used to maintain the NICE overlay hierarchy is much higher than both DMMP and DMMP+. As mentioned in Section 4.5.2, the control overhead of NICE is expected to become serious in a highly dynamic environment. However, in current stable scenario NICE still has a quite high control overhead when there is no dynamic changes in the group during the stabilization phase. The main reason causes such a phenomenon is that maintenance of multi-layered hierarchy is very costly especially when the group size is large. Differently, because of the dynamic mesh-based clusters, the control overhead in the DMMP framework is constraint within the locality, except for few frequent message exchanges in the mesh core.

Furthermore, the control overhead caused by DMMP+ is even less than DMMP because the optimized overlay hierarchy can reduce the message exchanges for joining the multicast session. Suppose high capacity nodes have been promoted to the high level of the hierarchy, it is easier for newcomers to find available parents to join the group.

Therefore, based on the above observations we can conclude: 1) DMMP+ can achieve more efficient data delivery than NICE and DMMP; 2) control overhead caused by the DMMP and DMMP+ protocols is stable and much less than that of NICE, even when the group size becomes large. 2) DMMP+ can largely enhance the performance of DMMP in a relatively stable environment.

5.2.3 Scenario 2: GT-ITM Topology

Due to the limitation of the topology generation algorithm in OverSim that it is less realistic (as mentioned in Section 4.4.4), in Section 4.5 we can only manually configure the underlying routers. In order to further identify the scalability of DMMP and as well as the effectiveness of our proposed self-improvement mechanisms, we attempt to specifically evaluate the performance of DMMP and DMMP+ under Internet-like topology. For example, the GT-ITM topology generator [119] produces a representative abstraction of the real Internet.

In the following experimental phase, the simulations are conducted based on the *Transit-Stub Domain Model*, an abstract representation of today's Internet created using the topology generator GT-ITM [119]. The generated initial network topology is then translated to NED files which can be used for our simulations. The translation mainly refers to the relevant parameters required to build the underlying network. For example, the number of routers (e.g. backbone router) is now defined by GT-ITM, instead by manual settings in a NED topology.

GT-ITM

Different from the topology generated in Section 4.5, we use GT-ITM to generate a large-scale Internet like topology. Before providing the detailed analysis, we introduce GT-ITM topology generator and explain the effort of converting the GT-ITM generated topology to supporting our simulation.

GT-ITM is built on top of the Stanford GraphBase (SGB) [120], a platform of structures and routines for representing and manipulating graphs. However, the topology generated by the original GT-ITM topology generator cannot be directly used for OverSim simulation. Therefore, we converted the necessary information of GT-ITM network into the OverSim-based topology.

There are three relevant specifications in GT-ITM, which are used to rebuild the same topology mentioned in [43].

- **itm** - to create flat random graphs and two forms of hierarchical graphs. The generated *.sgb* files are GT-ITM specific files.

- **sgb2alt** - to convert the above generated *.sgb* files into a *.alt* file which can be easily parsed by other tools.
- **alt2ned** - to generate *.ned* file based on the above *.alt* file.

To support our simulations, we define a new script to evoke the generation of *.ned* file which can be recognized by OverSim simulator. Based on generated *.ned* file, the required knowledge (e.g. the number of backbone routers) of the underlying topology will be used produce the the network definition (i.e. UnderlayforDMMP) for DMMP. For the value of data bit rate or propagation delays can be modified in *alt2ned*.

There are five graph models supported by GT-ITM:

- Pure random model: it is not a reflection of a real internetworks but it is attractive for its simplicity and straight-forwarded testing of networking problems.
- Waxman: is one of the most common random graph models, with the probability of an edge from u to v given by:

  $$P(u,v) = \alpha e^{-d/(\beta L)} \tag{5.1}$$

  where $0 < \alpha, \beta < 0$, d is the Euclidean distance from u to v.

  - Waxman 1: $L = \sqrt{2} \times scale$ and scale is the maximum distance between any two nodes. An increase in α will cause an increase of the number of edges of the graph, while an increase of β will increase the ratio of long edges relative to shorter edges.
  - Waxman 2: It provides an addition of the factor $radius = k\epsilon$ to control the number of edges in the graphs that are generated, given a k. Here, ϵ represents the desired average node degree.
- Exponential: it uses the following equation:

  $$P(u,v) = \alpha e^{-d/(L-d)}. \tag{5.2}$$

  The probability of an edge in this model decreases exponentially with the distance between the two vertexes.
- Locality: it partitions the edges into discrete categories based length, and assigns a different edge probability for each category. For each category, one parameter *radius* to define the boundary:

  $$P(u,v) = \begin{cases} \alpha & \text{if } d < L \times radius \\ \beta & \text{if } d \geq L \times radius \end{cases}$$

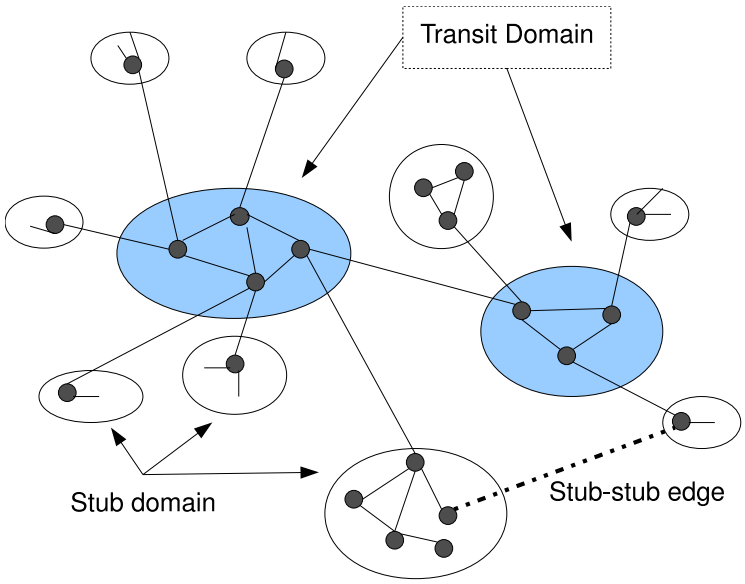

Figure 5.7: Transit-Stub Domain Structure.

However, neither type of above models captures the hierarchy that is present in the real internetworks. Therefore, a Transit-Stub (TS) model is preferred to produce hierarchical graphs by composing interconnected transit and stub domains. As shown in Figure 5.7, a random connected graph is constructed, in which one node represents an entire transit domain and then each node in the graph is replaced by another connected random graph, which finally composes the backbone topology of one transit domain. Moreover, it is required to have some number of additional edges between pairs of nodes, for example, one from a transit domain and one from a stub domain, or one from each of two different stub domains. In this way, the random graphs are all connected into a full graph.

The purpose of the following experiments is to further identify the usefulness of our proposed DMMP+, as well as its scalability. To configure the underlying network, we used $2,500$ routers as the compile of such a large topology generated by GT-ITM consumes up the 6 GB memory. The average router degree was set to between 3.0 and 4.0. We measured the router stress, link stress, control overhead as well as the average loss rate with regards to different group sizes.

Data Path Quality

To evaluate the data path quality, we compared the performance of DMMP+ and DMMP in terms of router stress, link stress and data path length. Note that the following results are different from the ones in Section 5.2.2. Here, we only configured $2,500$ routers due to the hardware limitations.

Stress

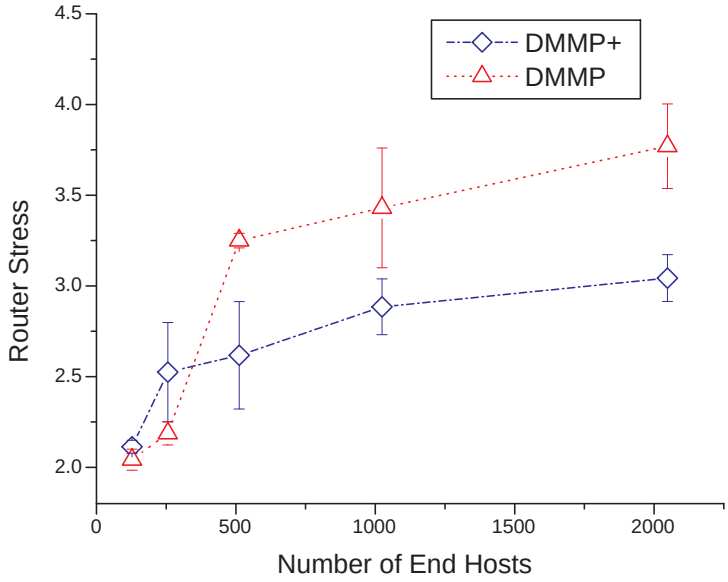

Figure 5.8: Comparison of Router Stress.

Figure 5.8 and 5.9 show the stress on the network routers and links. For each metric, we present both the mean value and the standard deviation.

When the group size is larger than 300, DMMP+ has much less router stress (max. 25%) that of the basic DMMP protocol. DMMP suffers from redundant packet copies through unoptimized overlay hierarchy especially when the group size is large. When the group size is less than 300, DMMP can achieve 15% less router stress than DMMP+ because periodic adding and deleting links within each mesh and parent-children swapings within clusters may cause more duplicated transmission of media packets. It eventually causes redundant copies through the routers.

For the link stress, DMMP+ has better (max. 75%) performance when the group size is larger than 550. The locality algorithm bound the impact of self-improvement within each cluster, and therefore there are few impacts on the network routers than that of links. When the group size is less than 550, the link stress of DMMP+ is higher than DMMP. It is mainly caused by that the switching positions among users, such as generating more duplicated data delivery. Differently, when the group size gets larger (e.g. more than $1,000$) the optimized overlay hierarchy can eventually provide much better service, which comprise the impacts of performing self-improvement mechanisms.

In a relatively stable environment, the quality of the DMMP overlay hierarchy can be gradually improved by the DMMP+ protocol. When the group size gets larger, the self-improvement mechanisms help more in alleviating the redundant packet transmission among the links. That is why

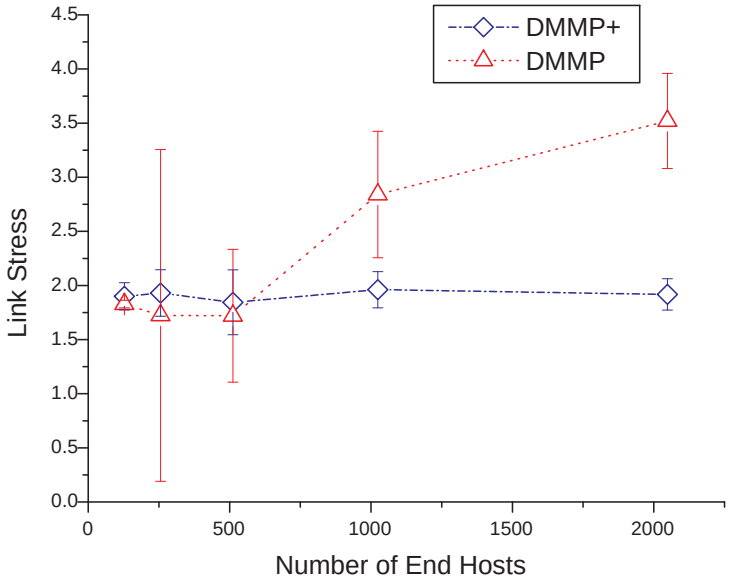

Figure 5.9: Comparison of Link Stress.

the link stress and router stress can be kept stable and low even when the group size is larger than thousands.

Data Path Length

Further, we measured the average data path length of DMMP and DMMP+ in Figure 5.10. Without the self-improvement, the average data path length performs very unstable, especially when the group size is smaller than $1,000$, because the established overlay hierarchy is not optimized due to the member joining phase. DMMP+ performs better than DMMP and the data path length gradually increases. The result is predictable since the overlay mesh and clusters become stable and efficient in expanding more available bandwidth. The more high capacity nodes are promoted to the higher level of the overlay hierarchy, the shorter the overlay multicast tree becomes (as it has been demonstrated in Section 4.1.2). Accordingly, the data path length can be shortened.

Unfortunately, when the group size increases the data path length of DMMP+ grows fast due to two reasons. First, we used the same number of super node for the large-size group. It is hard to find optimal (e.g., less e2e latency) super nodes to join the multicast session with small number of available nodes. Second, in the current work we have not simulated the super node dynamic joining procedure in order to avoid the complexity of the overlay management. Otherwise, the data path length can be easily optimized. We can still argue that the above result is acceptable as optimization of the data path length is not our main target.

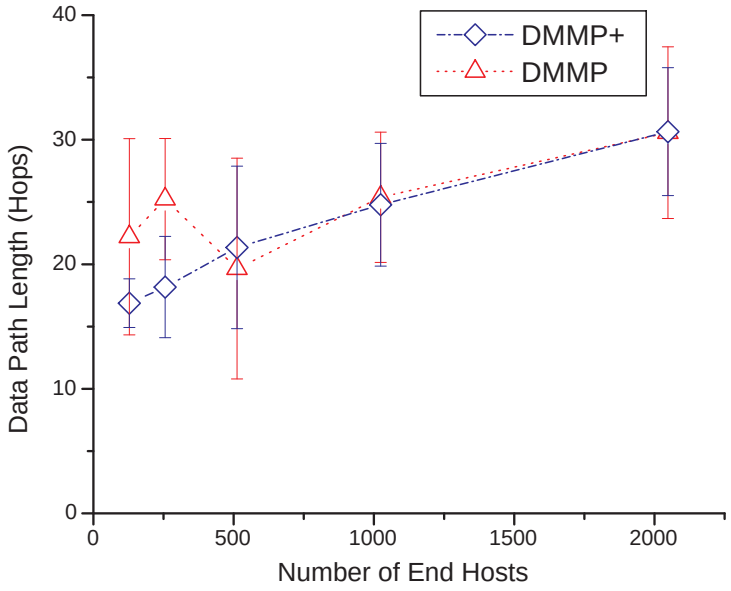

Figure 5.10: Comparison of Data Path Length.

Control Overhead and Loss Rate

With the requirements of additional position switchings, DMMP+ is expected to have higher control overhead and higher loss rate than DMMP. Surprisingly, as shown in Figure 5.11 the overhead of DMMP+ is quite comparable to that of DMMP. We believe that the locality-awareness switching helps reducing the overhead of maintaining the entire overlay hierarchy in a stable surrounding. Although additional message exchanges are required for switching positions, the cluster is kept more stable and efficient for delivering data to downstream nodes than the basic DMMP framework without self-improvement considerations. Similarly, the mesh core becomes more efficient and stable of delivering data to the non-super nodes. Therefore, the loss rate is insensitive to the changes of the group size.

Additionally, we evaluated the packet loss in this scenario. The packet loss rate is not measured in NICE due to the same reason explained in Section 4.5.2. Note that the packet loss is only caused by the duplicated packets or tree partitions. Since there is no frequent changes of the group members, and the loss rate is kept very stable for both DMMP+ and DMMP. Moreover, the loss rate caused by DMMP+ is less than that caused by DMMP. It is quite reasonable since the overlay hierarchy becomes more reliable due to the self-optimizing algorithms conducted in both the mesh core and the clusters.

Another important fact observed from Figure 5.12 is that when the group size is larger than $1,000$

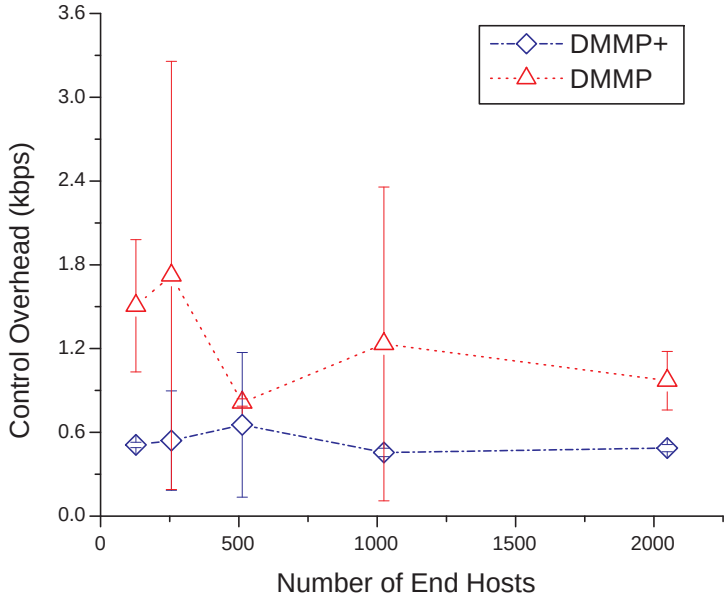

Figure 5.11: Comparison of Control Overhead.

the packet loss rate performs not worse than the case with small group size (e.g. when the group size is 500). As the packet loss is only resulted from duplicated packets or tree partitions, it implies that there are less duplicated packets traversed through the overlay network. Therefore, the link efficiency is accordingly improved because of the self-improvement mechanisms. Such a result also conforms the analysis in Figure 5.9 that the link stress keeps very stable and even drops a little bit when the group size becomes larger than $2,000$.

5.3 Summary

To improve the quality of service delivery hierarchy, self-improved DMMP protocol (DMMP+) was proposed to periodically optimize the established DMMP-aware overlay mesh and clusters. For instance, we define a *utility* threshold in the the mesh self-improvement to evaluate whether the adding link between super nodes is desirable. This threshold can dramatically reduce the cost of unnecessary link changes.

Under the two-phase experimental scenario, DMMP+ has been validated to further enhance the performance of DMMP in terms of scalability, reliability and efficiency. The simulation results have testified that self-improvement mechanisms can largely help optimizing the DMMP clusters in reducing the packet loss and control overhead.

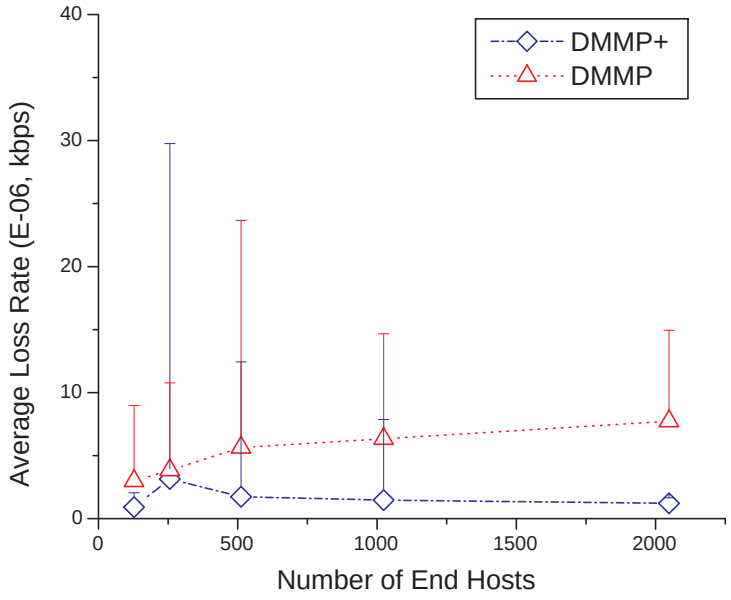

Figure 5.12: Comparison of Loss Rate.

The performance of DMMP+ is even better than that of NICE with much less router stress and much less control overhead. For link stress, DMMP+ can achieve very competitive performance

Overall, we ascertain that DMMP+ can assist the DMMP framework to be scalable, stable and efficient in supporting large-scale media distribution services. Nevertheless, DMMP+ cannot largely optimize the data path length when the group size is large. One possible solution is to implement the dynamic super node joining mechanism, however, it may make the peer management more complicated. Another possibility is to combine the e2e latency into the super node selection algorithm, which may increase the control overhead and the service latency.

Chapter 6

Interest-shared Group Management (IGMT) Protocol for DMMP

6.1 Introduction

As demonstrated in Chapter 4, DMMP can potentially support large-scale media distribution services, however, it may raise the issue of resilience due to the instinct nature of application-level multicast approaches. Peer-to-peer technologies have provided the opportunity of addressing resilience issues owing to its unstructured overlay construction and dynamic maintenance. For instance, several mesh-pull P2P architectures have been successfully deployed for P2P media streaming systems (e.g. IPTV) [54]. While existing studies mostly focus on peer-to-peer or overlay protocol design based on simulation under various topological constraints, experimental studies on a real-life P2P media distribution system will provide valuable information. Thence, we perform a comprehensive analytical and performance study on Joost, which is one of the first commercial peer-to-peer (P2P) Video-on-Demand (VoD) systems distributing various forms of video over the Internet. Motivated by the above investigations and requirements for P2P management in media distribution systems, we propose Interest-shared Group Management (IGMT) protocol for DMMP. The IGMT protocol is applied to DMMP+, which has extended the DMMP+ protocol in the framework to be more scalable, stable and efficient in supporting large-scale media distribution services.

The rest of the chapter is organized as follows. Section 6.2 presents an investigation on peer-to-peer management mechanisms used in Joost. The experimental analysis includes peer management schemes in terms of time pattern, bandwidth consumption and locality considerations. In Section 6.3 we propose an Interest-shared Group Management (IGMP) protocol. It is applied to DMMP-aware clusters to reduce the e2e service latency and to improve the resilience of the DMMP framework when end hosts within the same interest-shared group are available.

6.2 Investigations on P2P Management in Joost

We choose Joost as the target for the study of Peer-to-peer VoD services due to the following reasons. Firstly, as one of the earliest and best-known commercial peer-to-peer VoD products, Joost has the potential to become popular following a successful story of Skype. It offers high-quality and comprehensive VoD services, for instance, the current version (Beta 1.0) supports an instant on-demand video without any need for additional set top box. Furthermore, it is provided as a freeware without releasing source code, although it is known to be built on top of several open software such as Mozilla/xulrunner [121]. These facts may provide us some means to understand some particular behaviors of Joost clients, however, except for the limited knowledge of the used open software, the underlying P2P architecture and detailed mechanisms/techniques used in Joost, like when Skype was new, are still unrevealed.

Getting deep insights into various aspects of Joost has been challenging because the Joost architecture and many technologies it uses are proprietary. In particular, in order to understand its performance, we had to collect a large amount of data and analyze media streaming behaviors and peer management behaviors. Through numerous experiments, our study of Joost brings lights to the Peer-to-Peer performance and design issues of media distribution services to the public Internet. The detailed information of the experimental analysis of Joost system is provided in Appendix B.

As the performance aspect plays an important role in P2P VoD user adoption, we envision three typical usage scenarios and use them to study more closely the behaviors and performance of locality awareness, bandwidth capacity and peer management.

6.2.1 Experiment Setup

There are three main Joost server sites: the USA, Europe and Asia [122]. Since all mechanisms and technologies are assumed to be used in the same way, our experiments explore European site also due to the authors' location in Germany (GMT+01:00).

All experiments were performed between September 2007 and February 2008 at various geographical locations. We used six Windows XP SP2 machines for setting up an experimental testbed. They were equipped with the same processing power and connected to 100 Mbps full-duplex university LAN. Note that we also used Windows Vista and MAC OS X machines for the testing, however, their results haven't shown significant difference from the following results. Since December 2007, Joost has been open to public although it was still called Beta version (Beta v1.0) during the time of our experiments.

For data collection, we used Wireshark [123] and Omnipeek [124]. Tools like WhereIsIP [125] were used to perform reverse country, city and ISP lookups for an IP address when Omnipeek failed to return a DNS PTR record.

6.2.2 Experimental Methodology

Some typical scenarios are designed to examine the peer management aspect as it is the most important aspect. To facilitate our measurements, we analyze the major factors in which the peer management may involve.

- *Time pattern:* The different user distribution during a day or a week may have great impacts on the performance (e.g. contributions from peers highly depend on the number of peers).
- *Upload and download Capacity:* The peer management can be benefit from the efficient bandwidth usage if peer is given some incentives to contribute more to the network.
- *Popularity impacts:* The number of users may be largely determined by the popularity of the programs.
- *Locality considerations:* One of the main challenges in P2P VoD system is the efficient allocation of the available resources. Thus, it is generally desirable that data exchange be made preferably between nodes that are placed 'close by' in the underlying network to reduce the redundant usage of long-haul network links and to save local resources for network providers.

Designed Scenarios

As the performance aspect plays an important role in P2P VoD user adoption, we envision three typical usage scenarios and choose data-driven analysis on time pattern, bandwidth consumption and locality considerations to reveal the P2P mechanisms used in Joost. The results may provide more valuable information to ISPs, network administrators and content owners for a better understanding of the requirements for building and managing a P2P VoD system.

Before getting a deep analysis, this section provides a high-level description of the Joost analysis model we adopt in this investigation. As shown in Figure 6.1, there are two planes in the Joost client model: control plane and service plane. For control plane, only user authentication and channel management is performed using a classical client-server model, and all further signaling is performed on the P2P overlay. Thus, Joost users' information (e.g. contact list, status) are entirely distributed and decentralized among peers, which allow scalability on the one hand and cost-effectiveness on the other hand. Moreover, distributed peer managers (super nodes named in [122]) need to maintain the P2P overlay and help managing group members.

Joost provides high-quality VoD services as well as some value-added services, such as instant messaging. The communication between clients is established using traditional end-to-end IP paradigm, but Joost relies on other public nodes to ease the traversal of symmetric NATs and firewalls. It can be identified through the following experiments.

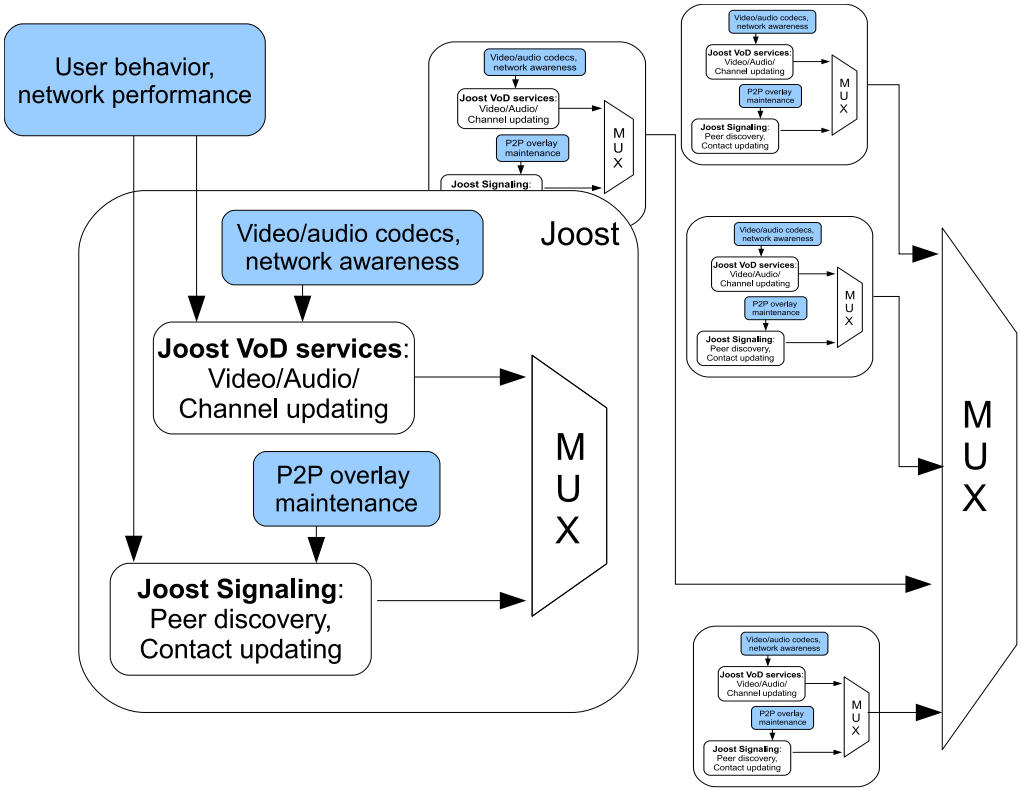

Figure 6.1: Synopsis of The Joost Client Model.

We used six Windows XP machines as described in Section 6.2.1. Three of them were configured with public IP addresses and the other three were located behind a NAT/Firewall (N/F-behind nodes). All of them were connected to 100 Mbps full-duplex university LAN.

- *Scenario 1:* To get a broader view of the time pattern, we monitored public nodes and NAT nodes over a period of three weeks (7-28 January, 2008) and captured over 78 GB data. Those test nodes were equipped with the same processing power and bandwidth support. We repeatedly ran the same channel at both nodes. Once the channel was finished playing, we emptied the local cache and re-started playing.
- *Scenario 2:* We randomly chose programs ranking in the "most popular programs" and unpopular programs with the same length. In this experimental scenario, a public node and a N/F-behind node continuously ran popular and unpopular programs over two-week period (14-28 January, 2008). We captured 24 GB data.
- *Scenario 3:* Two test nodes were located behind a Network Address Translator (NAT) and configured with non-routable, private IP addresses. One of them started to randomly choose one channel. After a short period (e.g. 5 minutes), the other node selected the same channel.

As media data is encapsulated in UDP (cf. Section B.2.3), we only captured UDP packets under the first scenario. Based on the knowledge that the packet size of media data is 1104 bytes and the traffic coming from peer managers are used for controlling, we further isolated media data from other control traffic. Nevertheless, Joost encrypts all UDP payloads, therefore, our analysis on the collected data is restricted to IP and UDP header fields including the source and destination IP address and port, and packet lengths.

6.2.3 Experimental Results

Some initial experiments [126] indicate that Joost relies on a plenty of dedicated infrastructure nodes (e.g. content servers) to distribute video. However, since December 2007 during our experiments, the contributions of peers largely increased and varied according to the time/life pattern. The following experiments illustrate our recent findings.

Time Patterns – NAT/FW-behind Node

Figure 6.2 and Figure 6.3 illustrates typical segments of first scenario results of N/F-behind node. They plot the average percentage of media contributions from Joost servers (cf. Section B.2.1) and peers. We separate weekday trace from weekends trace since they have different distributions. The deviations identify that both traces follow the similar pattern except for two time slots in the weekdays (16:00 and 22:00, 11 January, Friday) when there were a great number of peers contributed to the test nodes. In fact, it is understandable since by that time the weekends had already started in some European countries. Note that our experimental location was in Germany. To verify our conjecture, we further traced these contributed peers. Among the total peer contributions (63.3% in average), 38% of the media data was contributed by European peers and 25.3% was transmitted from the US.

The second main observation from above two figures is that Joost servers delivered a majority portion of media data to Joost clients during the entire week. However, we observed an interesting phenomenon in the weekday trace. There were two user peeks, 6:00 and 12:00, that a lot of contributions (50 – 60%) were from other peers instead of content servers. According to what has been observed in [127] that the number of users drops gradually during the early morning (0:00-7:00) and climbs up to a peak when users are in noon break (12:00-14:00), the number of Joost users is expected to be the least in both time slots. In other words, if the contributors were located in Europe, it would be against the daily life pattern since the first time slot (4:00-6:00) will be sleeping time and 10:00-12:00 is working time. In contrast to the weekday trace, during 0:00-2:00 in the weekends there was a user peek, which is possible that most of the contributions came from European countries.

To discover the reasons and identify our conjecture, we further analyzed these user peeks. As shown in Figure 6.4, from 4:00-6:00 most of the data was contributed from the U.S. and 10:00-12:00 European peers contributed the most (60-65%) but US peers still contributed over 20%. It

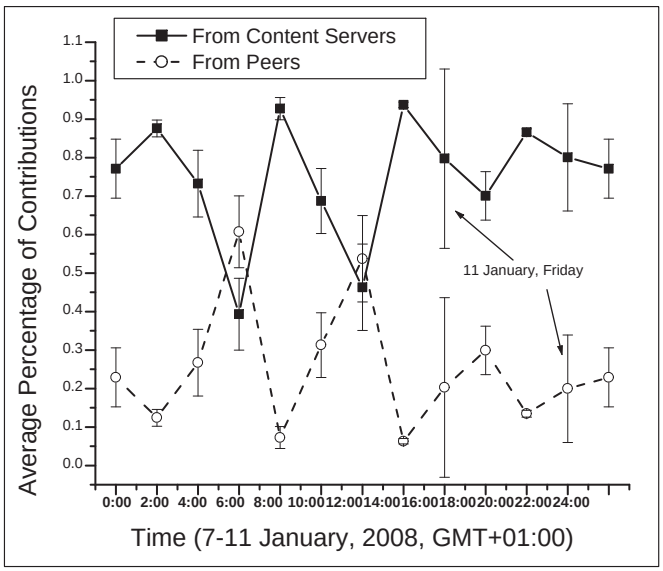

Figure 6.2: Weekday Trace of NAT/FW-behind Node.

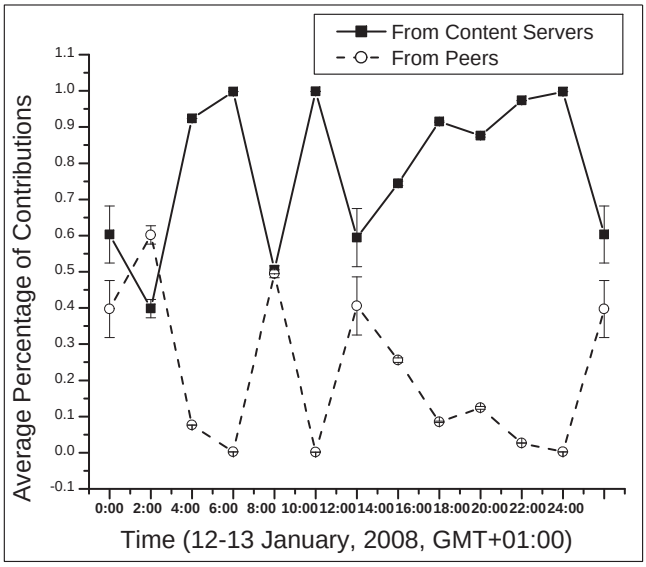

Figure 6.3: Weekend Trace of NAT/FW-behind Node.

indicates that inter-continental links (between the U.S. and Europe) are often used regardless of the number of local peers. For the third slot 0:00-2:00, peers from the U.S., Europe and others

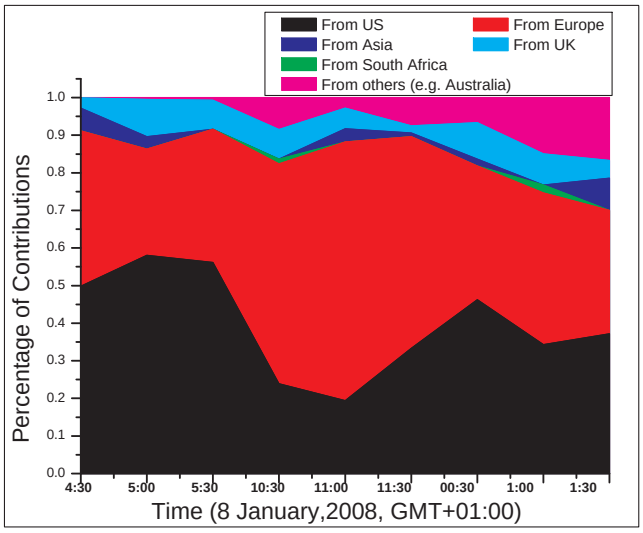

Figure 6.4: Timeslot Trace of NAT/FW-behind Node.

shared the similar portion of the contributions.

As network operators struggle to control the overall usage of bandwidth, it would be advisable that P2P service provider could constrain the media data transmission within a locality without frequent use of inter-continental links. Otherwise, the inter-continental bandwidth usage will become a non-marginal issue for the network providers. To verify whether locality-awareness has been considered in Joost peer management, we studied its performance additionally in Section 6.2.3.

Time Pattern – Public Node

Since public nodes and N/F-behind nodes ran the same channel, the available contributions were assumed to be similar. Surprisingly, public IP address configured nodes relied great heavily on the Joost content servers. As depicted in Figure 6.5 and 6.6, most of the time content servers contributed over 60% of the media data. Comparing with Figure 6.2 and 6.3, we conjecture that Joost uses a peer management algorithm similar to the one used in Skype that easily reachable nodes (e.g. public nodes) with high capacity are used to relay traffic for other peers, so-called super nodes in Skype. Actually, the difference of upload throughput between public nodes and N/F-behind nodes is significant (see Section 6.2.3). Besides, we believe that the available bandwidth, performance (e.g. CPU, memory size) are considered in the peer selection phase, which has been identified in [126].

If we consider the deviation of the weekly trace in Figure 6.5 and 6.6, except for the time during 4:00-6:00, the rest hourly trace followed the similar pattern. To reveal the cause of difference,

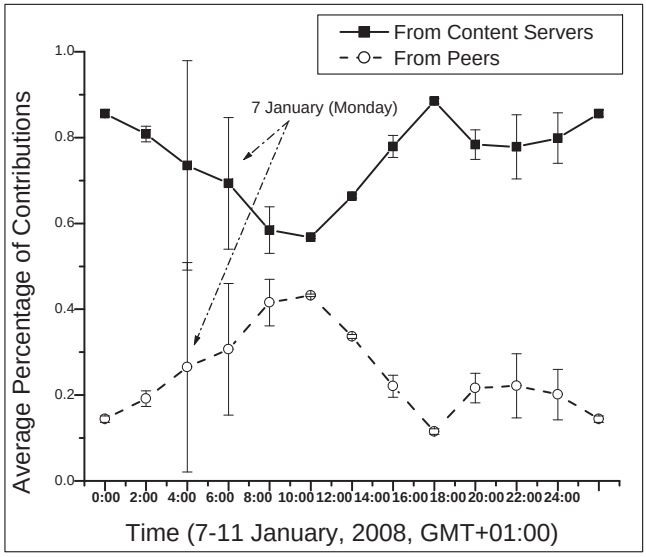

Figure 6.5: Weekday Trace of Public Node.

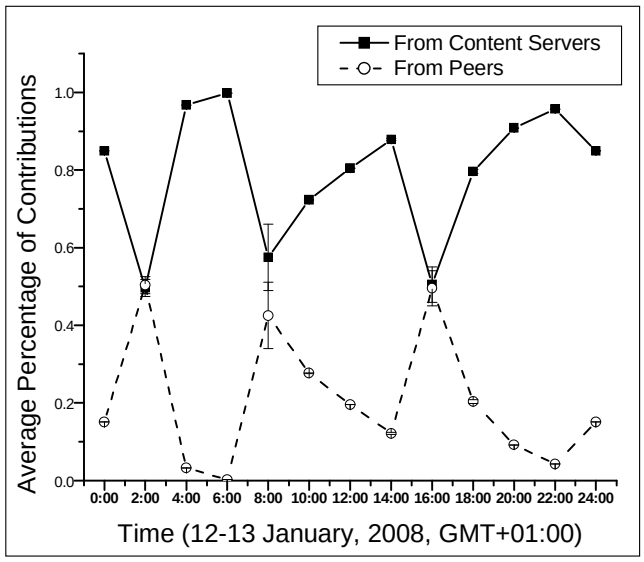

Figure 6.6: Weekend Trace of Public Node.

we tracked this particular time duration. It was noticed that on average 26.4% (total 49.3%) of contributions came from the US (local time: 20:00 - 1:00), 17.5% from Europe and rest of peers

(e.g. Australia (GMT+10)) contributed 5.43%. Again, it conforms to the above observation that most contribution were transmitted through inter-continental links.

Furthermore, the weekend trace performed quite differently from weekday trace. Particularly, the time slots of 2:00 and 17:00 respectively reached the peer contribution peeks (over 50% of the contributions). Among these contributions, European peers transmitted on average 34.72% of the media data, US peers forwarded 11.0% and others contributed 6.48%. Nevertheless, Joost servers still contributed 47.8% which is much higher than any of the above contributions.

Lastly, the number of contributing peers is expected to be even higher at weekend nights but for both public and N/F-behind nodes most of the contributions came from content servers. As observed in [126], the possible explanation is that the local Joost users in Germany were limited likely due to the current programs are mostly only in English without German subtitles. If the Joost client can select preferred language in the secondary audio tracks, it will attract more clients during their relaxing time. The other explanation could be that most Internet TV fans are night owls since for both public node and N/F-behind node between 0:00 and 2:00 on weekends the peer contributions reached the highest peak.

Bandwidth Consumption

Having isolated the UDP packets, we examined the average throughput of public node and N/F-behind node for each period of 1000 minutes through four weeks. Figure 6.7 shows that the public node's upload throughput is on average 67% higher than that of the N/F-behind node although they have the same capacity regarding its bandwidth support and processing power. Such an observation suggests that public node is likely chosen as relaying nodes for other peers. Moreover, the average download throughput of the public node is 15% higher than that of the N/F-behind node since most of the data was directly transmitted from content servers. The reason why the throughput of the public node was not stable is that the content servers may not be able to contribute with the same amount of media data when simultaneously serving a large amount of peers.

We observed that the N/F-behind node downloaded and uploaded media data in a fairly constant speed compared with the public node. The download throughput of the N/F-behind node was 438 kbps, that is, 200 MB per hour, and 22 MB per hour for uploading. For public nodes, the average download throughput was 493 kbps, namely, 225 MB for one hour, and 68 MB per hour for uploading. Through the experiments, we found that the average percentage of control traffic among the total traffic was 15%. Thus, public nodes only need to support 580 kbps downlink capacity and 84 kbps for uplink although they were connected to 100 Mbps full-duplex university LAN. Thus, we suggest that Joost can be a little aggressive especially to high-capacity node when they are available.

Unfortunately, as explored in [126] low capacity nodes could be hardly supported in the Joost system since current Joost system only provides the same quality for any video. We would suggest using layered or adaptive mechanisms for more efficient video distribution. For example, servers

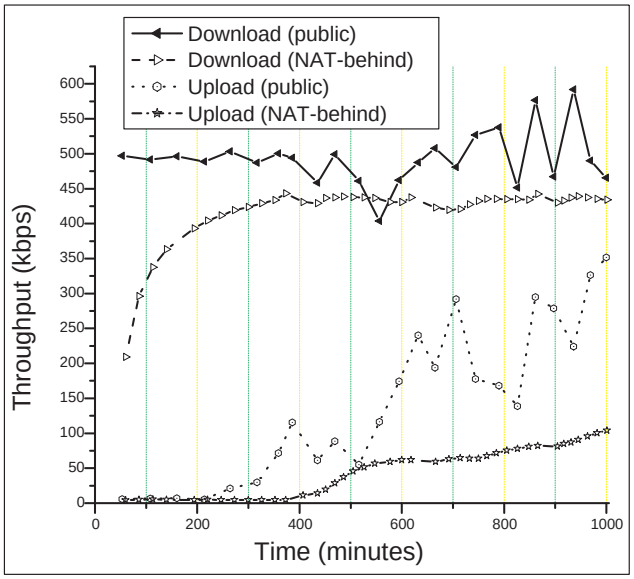

Figure 6.7: Throughput of Public and NAT/FW-behind Node.

or some high-capacity nodes are responsible for transmitting the basic layer of the encoded video and other available peers could be used to transmit the enhanced layers in order to improve the video quality. Although it might introduce some complexities into the peer management, the Joost system could support more users including some low-capacity nodes without wasting network resources.

Popularity Impacts

We assume that the pubic nodes actively participate in relaying media data for other peers. Hence, we only traced two public nodes running different types of programs as defined in Scenario 2.

As with many P2P media streaming applications, the number of users is largely determined by the popularity of the program. For popular channels, the upload throughput should be much higher than that of unpopular channels. What is observed in Figure 6.8 is consistent with the speculation that popular channel node's upload throughput was much higher (over 150%) than that of unpopular channel node.

For popular channels, at the initial phase the throughput increased dramatically to reach the throughput peek 85 kbps. Then, it decreased till 35 kbps and increased again till 60 kbps and kept relatively low. The near constant throughput during the late stage suggests that Joost P2P system scales well since more contributors are able to forward media data after they receive the data.

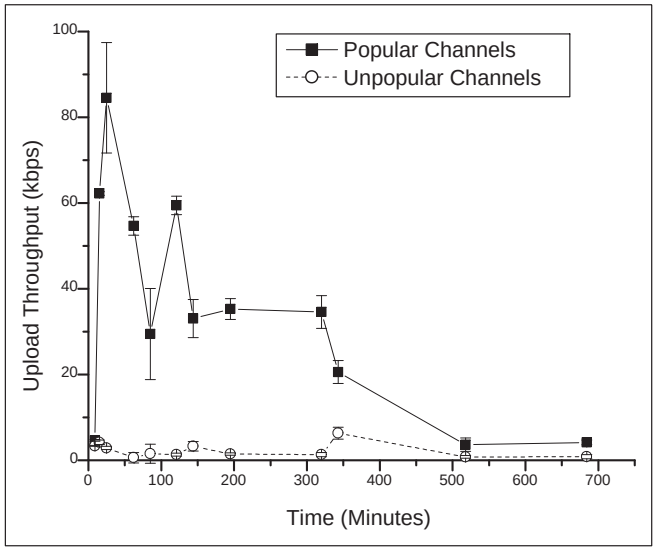

Figure 6.8: Upload Throughput of Popular Channels vs. Unpopular Channels.

During our experiments, the throughput of unpopular channels is always low (in average 2.34 kbps) and the unpopular channel node comes into the stable phase much earlier than the popular channel node. We conjecture that there are much less requests in the system for the unpopular programs.

Locality Considerations

After three-day repeated experiments of Scenario 3, we analyzed the collected data from both test nodes. Although the two test nodes were watching the same channel and geographically locating near each other, the second test node only received in average 1.3% of the data from the first node (96.2 MB out of approximately 7.4 GB). Therefore, we ascertain that the locality-awareness is not well designed in the Joost system, for instance, the topological locality may have not considered in the peer management. It would waste a large amount of resource if media data is transmitted from remote peers instead of from available nearby peers.

To further evaluate the locality-awareness in Joost peer management, we parsed the IP address of contributed peers from which our test nodes received data. In summary, we identified 1210 distinct peers which provided inneglectable contents to our test nodes. These peers were located in over 54 countries. Of all the data collected from the test nodes, 45% (547) came from European countries, 24% (293) came from United States, 8.2% (99) came from Asian countries, 7.9% from South America, 3.6% from other countries. Besides, 45 IP addresses were not traceable and therefore we marked them as “unknown”.

Figure 6.9 shows that the major sources of peers are Europe and United States. Moreover, sources of JCs from Germany were 130 (19% of Europe). Since our host was located in Germany, we suspect that the geographical distance (e.g. from specific continent) may have been considered in Joost. For example, the prefix awareness may have been considered during the peer selection. Note that a high-level (geographical locality) is not enough for peer management since resource can be still wasted for being transferred to a remote user in the same geographical location.

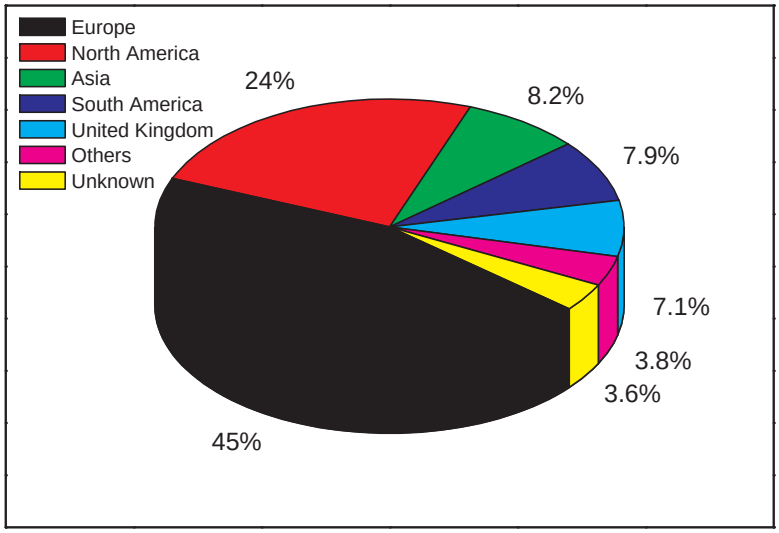

Figure 6.9: Geographic Location Distribution.

As with many P2P media streaming applications, the number of users is largely determined by the popularity of the program. For popular channels, the upload throughput should be much higher than that of unpopular channels. What is observed in Figure B.7 is consistent with the speculation that popular channel node's upload throughput was much higher (over 150%) than that of unpopular channel node.

For popular channels, at the initial phase the throughput increased dramatically to reach the throughput peek 85 kbps. Then, it decreased till 35kbps and increased again till 60 kbps and kept relatively low. The near constant throughput during the late stage suggests that Joost P2P system scales well since more contributors are able to forward media data after they receive the data.

During our experiments, the throughput of unpopular channels is always low (in average 2.34 kbps) and the unpopular channel node comes into the stable phase much earlier than the popular channel node. We conjecture that there are much less requests in the system for the unpopular

programs.

In order to further investigate the locality awareness in Joost, we measured the RTT from the receiving peers to the transmitting peer by OmniPing [128]. We conducted this experiment in parallel with the ping experiment by using WhereIsIP [125] to determine the number of hops between our Joost client and the transmitted peers.

Table 6.1: Locality Experiments with RTT, Hops and Data

Host	Hops	RTT (ms)	Data (%)
Host 1	11	19.20	0.01
Host 2	12	113.63	0.02
Host 3	14	110.13	0.07
Host 4	13	97.397	0.48
Host 5	21	134.14	0.66
Host 6	16	128.22	0.95
Host 7	19	128.97	6.36
Host 8	16	147.14	7.22
Host 9	22	186.27	8.64
Host 10	11	416.68	8.25
Host 11	18	182.47	10.15
Host 12	21	56.19	18.07
Mean	16.17	143.36	5
Median	17.5	131.18	3.655
Standard Deviation	3.848	93.873	5.458
Correlation to Data	0.5228	0.2173	

Table 6.1 summarized the results of the experiments that tested peer connected with University LAN over 1 Hour 30 minutes. We firstly selected 19 peers who contributed most data to our test node. Meanwhile, there were 7 Joost content servers (7 out of 19). Then, we traced the rest 12 peers with their RTT and hop counts. As depicted in the Table, the hop counts varies ranging from 11 to 22, and the largest contributor has large hop counts. All above hosts show a weak positive correlation between RTT and the amount of transferred data. Thus, we conclude that Joost selecting peers is unlikely based on topological locality. Otherwise, the result should show strong negative (over 0.7) correlation between them.

6.2.4 Lessons from Experiments

So far, we have studied peer management in terms of time pattern, bandwidth consumption and locality considerations with some envisioned typical scenarios. Our studies demonstrate that with some dedicated infrastructure server the current Internet is capable of meeting performance requirements of high-quality VoD services. Although large-scale P2P VoD systems are potentially

deployed in today's Internet, the performance could be improved, for example, based on the following observations:

- The current Joost system relies on an overlay network deployed with a set of centralized content servers as identified in Sections 6.2.3 and 6.2.3, which may still raise the scalability issue in the near future. Nevertheless, it is very useful to separate media distribution from controlling peer-to-peer hierarchy, which makes the Joost VoD system relatively stable and scalable. Therefore, in our DMMP implementation we could deploy a set of dedicated servers which are responsible only for managing the multicast group. However, in this thesis we only rely on some RPs to help managing the multicast group.
- As indicated in Sections 6.2.3 and 6.2.3, we believe that public nodes with high capacity may be selected as main relaying nodes for other peers. However, in our experiments their uplink capacity usage is still quite low (on average 84 kbps). Thus, the Joost system could be aggressive to these nodes, together with certain incentive mechanisms to encourage them contribute more to the network, which may help the system overcome the scalability issue. It proves the usefulness of our considerations on heterogeneous capacity during the construction of DMMP-aware hierarchy.
- Section 6.2.3 identified that the geographical distance may have been considered in the peer management of Joost. However, the lower-level locality-awareness (e.g. topological locality) may still be missing in the peer management. Besides, the inter-continental links are often used to transmit media data regardless of the number of local users, which may overload the network provider's costs. If the P2P service could be AS-/network level locality awareness, it would be beneficial for both customers and service providers. Again, through a real world measurement the idea of applying he locality-awareness into the DMMP-aware clusters has been validated.
- Our observations on popularity impacts in Section 6.2.3 indicate that each client has a certain interest of watching IPTV, for instance, a specific interest of selecting channels/programs. These observations motivate the following studies on "interest-shared group management for DMMP".
- Joost currently provides each client with the same quality of video. This may result in an inefficient resource utilization if some clients are unable to support the desired video quality. Hence, layered video or adaptive mechanisms could be introduced into Joost. In this work, the layered video scheme has not been scheduled in order to avoid complexity of peer management in the system.

Based on what has been explored in the above investigations, we intend to contribute our efforts on optimizing the locality-awareness in the peer management of the DMMP framework. Suppose some clients have similar interests in watching IPTV, we propose an interests-shared peer-to-peer management (IGMP) protocol for DMMP with considering shared interests and locality-awareness in Section 6.3.

6.3 Interest-shared Grouping Management (IGMT) Protocol

In this section, we propose the IGMT protocol to further improve the performance of DMMP+, especially for the DMMP-aware cluster formation. Actually, the similar idea has been applied to file sharing systems [129]. They contributed the efforts to quickly resolve content queries in P2P systems. However, different from their work we focus on quickening the joining and rejoining procedure through two algorithms, namely, *interest-shared group discovery* and *shortcuts establishment*.

6.3.1 Shared Interests

Our design philosophy differs from existing P2P media distribution work that we seek to a light-weight, efficient and decentralized grouping algorithm to improve the performance of DMMP membership management. Through above experiments, a powerful principal can be identified, if a peer has watched a particular channel of one category, it is very likely that the peer will select the other channels in the same category. These peers form an *interest-shared group*. We propose an interest-shared group management protocol, namely IGMT, establishing interest-based shortcuts, that efficiently exploits interest-shared groups for membership management. It is assumed that peers who have some common interests can create shortcuts to each other. Without waiting for the long-delay response from the RP or dedicated super nodes, the peer can use these shortcuts to quickly join the group and receive media data. If the shortcut fails to piggyback the required media, peers resort to using the basic joining mechanism described in Section 3.5.3. The shortcuts between peers can be considered as an *open structure* on top of the DMMP framework.

To clarify the above interest-shared grouping concept, Figure 6.10 gives an example. Suppose the peer in the middle is looking for channels C2, C3 and C4. Node A, B and D have one matching channel C2. Node C and E have two matching channels C2 and C3, whereas node E has all three channels. Therefore, the node E and the requesting peer share the same interests, where the interest represents the group of matching channels, $C2, C3, C4$. Our following studies focus on identify these peers, and establish shortcuts among them for efficient media distribution. Remark that the shortcuts will be established only when the two peers are nearby, for example, they are located in the same cluster in DMMP+.

In fact, such an mechanism is protocol-independent, that is, it is not limited to be used in DMMP+, but any other application level multicast or P2P media distribution system. However, in this Chapter we take DMMP+ as an example to validate the idea of using interest-shared group management.

6.3.2 Interest-shared Group

We aim at providing interest-shared grouping with considerations of simplicity and scalability. Relying on the local learning strategy, peers are able to detect potential parent nodes. The discovery

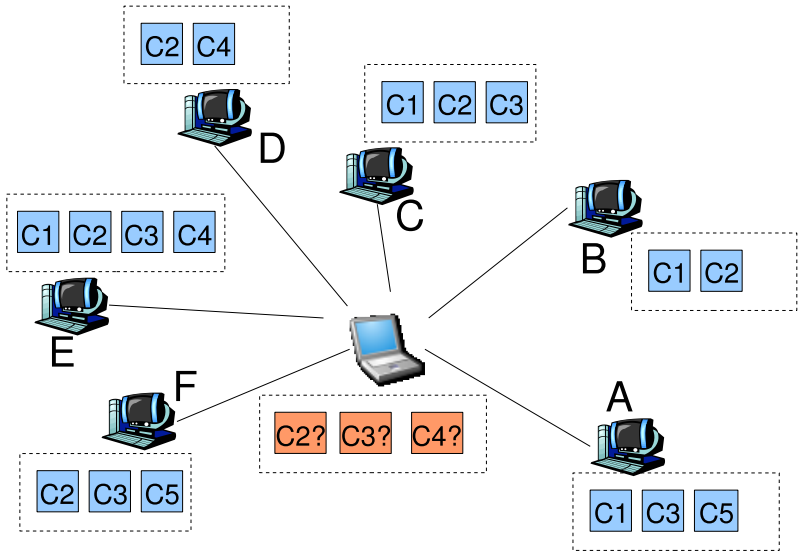

Figure 6.10: Example of Interest Shared Group.

algorithm should be light-weight, and additionally it has to be adaptive to dynamic changes.

Thus, we present two algorithms to achieve the above goal: 1) shared interest group discovery; 2) shortcuts establishment. Through these two algorithms, an open structure is built on top of DMMP framework, which can improve the performance of DMMP with regards to efficiency and resilience.

Group Discovery

We use the following heuristic to detect the shared interest group: peers are belonging to the same group if they have shared interest of selecting channels which we are looking for. When a new peer joins the media session, it first queries its Candidate Parent Cache (CPC), which may contain some addresses of peers who replied its requests in the past, to obtain potential parent nodes. If the cache is empty or those cached peers are not active (through probing), the new peer can ask the RP (landmark) for a list of candidate parent nodes, which is performed in the same way as described in Section 3.5.3 or 3.5.6.

Here, we suppose that there are some candidate parent nodes from its local cache are available. The new peer can try to connect them by directly sending the joining query to them. There are three possible responses:

- **Accepted**: If the candidate parent still has vacant out-degrees and the capacity (e.g. available out-degree) of the requester is acceptable, a response with acceptance will be sent to the newcomer. Here, the candidate parent node checks the capacities of the requester in

order to determine which one can be accepted if there are more than one request at the same moment.

- **Rejection with a candidate list**: Either the requested candidate parent has no spare out-degree or the capacity of the re questor does not meet the requirement (e.g. the re questor is a leaf-node), the request will be rejected. However, the candidates can forward the request to their active children which still have spare out-degrees. The iteration continues till the request is either dropped due to timeout or accepted by certain peers.
- **Rejection only**: Either the candidate parent has no active children or all children have occupied, there is no implication of possible candidates for the requester. The only way the newcomer can join the group is to request directly from the RP.

Take an example to illustrate the interest-shared member joining procedure in Figure 6.11.

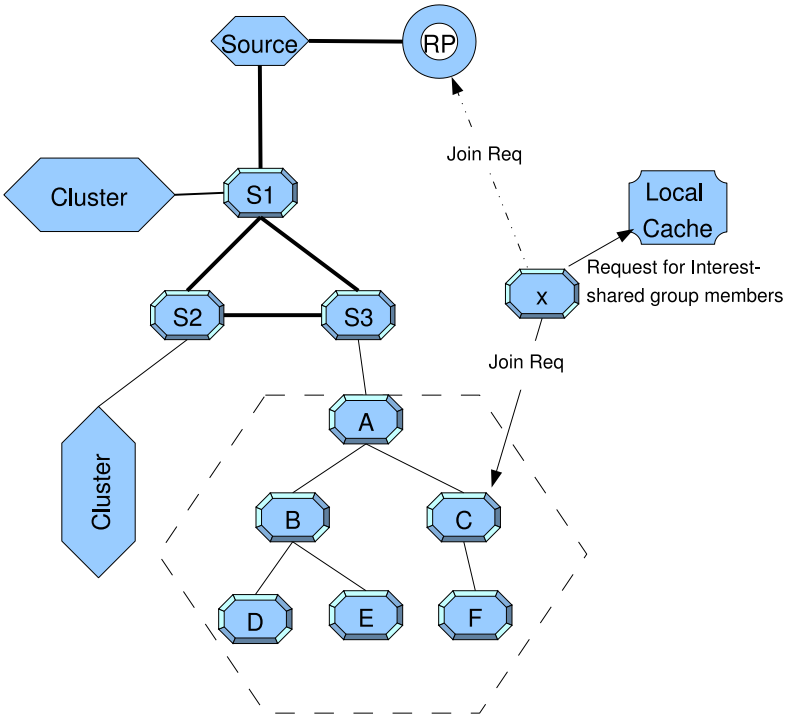

Figure 6.11: Shared Interest-based Joining Procedure.

Suppose node *x* wants to join the multicast session, node *C* firstly checks its CPC for interest-shared group members. If there are some records in the cache, for instance, node *C* is stored in the cache. Node *x* sends a request to node *C* in order to join the group. . There are three possibilities after node *C* receives such a request:

- If node *C* has available degree, it can take the request from node *x* in case there is only one join request (within the waiting timer) to node *C*.
- If there are multiple join requests received by node *C*, node *x* can be accepted as one of its children as long as node *x* has relatively higher capacity than other requesters and node *C* has vacant capacities.
- If node *x* has less capacities than other requesters or node *C* has limited capacity left, it will receive a rejection response and possibly with the address of node *F* if node *F* still have available degrees.
- Upon receiving a rejection from the interest-shared group member *C*, node *x* stores the candidate *F* into its *Candidate Parent Cache* for the use of next joining procedure. The interest-shared grouping algorithm ends in order to avoid the possible flooding attacks. The node *x* will join the group following the general procedure and there is no shortcuts established for it.

However, such a procedure may take a long time. Therefore, we suggest that the each peer sends requests to its potential candidate parents as well as to the RP at the same time. In addition, we bound the maximum number of candidate parent nodes who have the common interest with the newcomer in order to avoid message flooding over the entire overlay network.

There are some alternatives to discover interest-shared group members. For example, more interest-shared members can be discovered through exchanging knowledges among neighboring peers after the peer joins the multicast session. Once the peer joins the group and starts to exchange information with its relatives (e.g. parent, siblings), the additional information about the interests of these peers is also exchanged within the same cluster. By this way, the addresses of nearby peers who have the similar interests are stored in the Candidate Parent Cache (CPC). Once there is a failure, the peer can quickly rejoin the session by requesting the interest shared group members.

Shortcuts Establishment

Given that there are multiple responses from shared group members, with whom the requester should establish the shortcut? Without a doubt, there can be multiple shortcuts accepted by the requester, however, it also raises the cost of the group management. In the current design, we consider one shortcut by the following algorithm: 1) ranking the possible shortcuts based on the perceived capacity; 2) selecting the most useful shortcuts from the top of the list till it joins the group. The shortcuts can be ranked based on many metrics, such as similarity of interests, e2e latency or available capacity. We use a combination of available bandwidth and the similarity of shared interests to perform the selection.

Recall the equation 3.1 in Section 3.3.3, we combine the similarity of shared interests instead of uptime into it if the node newly joins the group:

$$c_i = \frac{s_i \times b_i}{\sum_{i=1}^{N} b_i} \tag{6.1}$$

where s_i refers to the similarity of interest, in form of percentage. Using the same example in Figure 6.10, the similarity of each node is listed as follows:

- node A, node B, and node D: $\frac{1}{3} \approx 33\%$
- node C and node F: $\frac{2}{3} \approx 66\%$
- node E: $\frac{3}{3} = 1$

Further, we suppose the available bandwidth for each of above nodes are as follows.

- node A: $b_a = 56$ kbps
- node B, E: $b_b = b_c = 28$ kbps
- node C, D, F: $b_d = b_e = b_f = 18$ kbps

Therefore, according to the equation 6.1 we can calculate the capacities: $c_a = 0.1124$, $c_b = 0.0562$, $c_c = 0.0723$, $c_d = 0.036$, $c_e = 0.1688$, and $c_f = 0.0723$. Then, the above perceived capacities are accordingly ranked in the node x's local cache. If x receives multiple acceptance from node A, C, E, for example, it might join as E's child since node E has relatively high capacity regarding the shared interest and available bandwidth.

In Figure 6.12, we show the Pseudo code to determine E given the candidate parents in the local cache. The algorithm is used to determine the best shortcuts for all possible active shortcuts and selects five of them with the highest capacities.

As mentioned in Section 5.1, DMMP periodically self-optimize the mesh and clusters by promoting high capacity node to the core of the overlay hierarchy. Besides, Section 5.2 proves that self-improvement can efficiently improve the performance. Therefore, such a mechanism is desirable to be extended to support interest-shared grouping. For clearance, we present pseudo code for the shortcuts establishment algorithm in Figure 6.12.

Through periodic refresh exchanges among the relatives, IGMT member is able to detect more available interest-shared group members. As mentioned in Section 3.3.3, our proposed capacity algorithm can be easily extended for considering additional requirements. Thus, the capacities of interest-shared group members can be updated as follows:

Shortcuts establishment Algorithm

```
1   Enumerate all possible shortcuts ranked in
the local cache: S^1, S^2, ... S^n.
2   S^max= null, E=0
3   for each S^i, 1≤i≤n do
4   set S^max = S^i,  if S^i >S^max
5   endfor
6   for 0<j<6
7   if E_j <S^max,  then
8   E_j = S^max
9   endfor
10  return E
```

Figure 6.12: Pseudo Code for Selecting the Best Shortcuts.

$$c_i = \frac{s_i \times b_i}{\sum_{i=1}^{N} b_i} + c \cdot t_i. \tag{6.2}$$

where t_i refers to the uptime of end host i (cf. Section 3.3.3). Above algorithm provides an extra consideration on the time duration of a group member. The equation 6.2 can be extended to combine other metrics, e.g. delivery delay. For brevity, we use the combination of available bandwidth, shared-interests and uptime to select parent candidates for the partitioned nodes.

In order to avoid message flooding, we set a threshold for the maximum group member (i.e. five) in the cache. Therefore, when a member attempts to rejoin the group due to some failure occurred in the multicast tree, it can directly request these five members. As indicated in Section 6.3.2, the request might not be accepted. Nevertheless, to establish these shortcuts does not harm the joining procedure even if the requests are rejected.

6.3.3 Extended Messages

We need to extend the messages defined in Section 3.4 to support the interest-shared group management described above. Table 6.2 lists the extended DMMP messages.

Actually, the *Refresh Request* and *Refresh Response* messages are extended from the ones in Table 3.1. These two extended messages are used to exchange the knowledge of interests in addition to user's capacity as defined in Section 3.4. Besides, in order to periodically update the user's cache we define a *cpCacheRefreshTimer* to trigger the refresh information among interest-shared group members.

Table 6.2: Extended DMMP Messages for Interest-shared Group Mechanisms

Messages	From	To	Operation
Grandparent Request	Cluster Member	Cluster Member	Probing Relatives
Grandparent Response	Cluster Member	Cluster Member	
Relatives Request	Cluster Member	Cluster Member	Interest-shared Group
Relatives Response	Cluster Member	Cluster Member	
Refresh Request	Interest-shared Group Member	Interest-shared Group Member	Refresh Information
Refresh Response	Interest-shared Group Member	Interest-shared Group Member	

Moreover, to support IGMT operations the *MemberMap* and *MemberInfo* need to be extended from Section 4.4.3. During shared group discovery, the *DMMP+ Member* keeps track of the following additional data.

- The relatives in the local cluster (number, IP address, etc.).
- The interest-shared group members (number, IP address, etc.).
- The candidate grandparents.

Note that all above information is exchanged only within the local clusters. The *relatives* are the potential interest-shared group members for a certain peer. Based on the probability of shared interests, some *interest-shared group members* will be maintained in each member's local cache.

6.3.4 Rejoining Procedure

In this section, we only take the rejoining procedure as an example to evaluate the benefits of the interest-shared group management mechanisms. The interest-shared group mechanism can be also used for the newly joining members. However, we believe such a mechanism will be more useful when some members are partitioned from the tree due to either failures of their upstream nodes or failed promotions during the self-improvement procedure.

Recall the rejoining procedure proposed in Section 3.5.6, there are several possibilities that the partitioned member cannot efficiently rejoin the group. One typical example is that the negotiated backup "parent" node leaves the session without notification. In such a case, the node requires to rejoin the group starting from requesting the available super nodes. However, the interest-shared group management can facilitate the rejoining procedure:

- The member checks the local cache for interest-shared group members. Note in most cases as described in Section 3.5.6, when a member rejoins the group, it needs to check its local cache for the candidate parents or available super nodes.
- The member needs to compute the capacity of the interest-shared group members and selects some high capacity nodes.
- The member needs to exchange the knowledge of shared interests after it rejoins the multicast session. Although the periodic refresh message as well as some self-improvement mechanisms require message exchanges, the additional information about shared interests causes extra overhead but maybe marginal.

Therefore, we expect the following message exchanges when a member wants to rejoin the multicast session. Figure 6.13 shows the expected message flows for the rejoin procedure. Here, *x* is the partitioned member which wants to rejoin the multicast session; *cPc* represents the potential parents in its candidate parent cache; *M* represents the interest-shared group member.

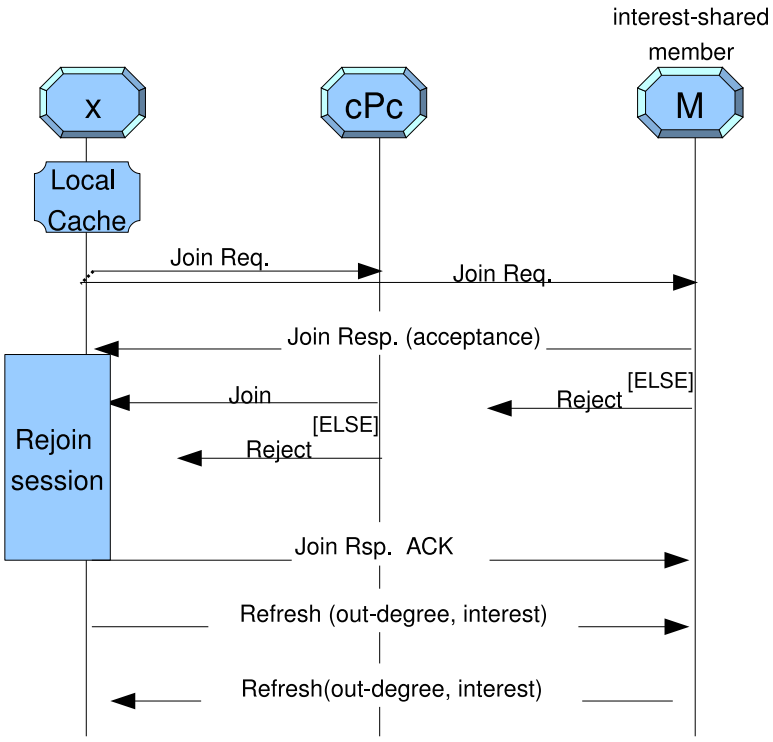

Figure 6.13: Message Flow for Interest-shared Rejoin.

In order to quickly recover from failures, IGMT allows partitioned users send *Join Request* to the potential parents (stored in the cpcCache) and interest-shared group members at the same time.

Certainly, the optimal cases are some of the interest-shared group members are online. The requester can directly rejoin the session through the available interest-shared member. Nevertheless, if it fails to join the session, it can still get a chance of rejoining the tree via the candidate parents as mentioned in Section 3.5.6. The advantage of relying on interest-shared group members to rejoin the group is straight-forward: the high interest similarities between the requesters and the group members, high available bandwidth, and relatively longer online time assure the high availability and good quality of service.

After rejoining the group, it is optional for the end host to testing the e2e delay through *RTT estimation*. Under our assumption, the interest-shared group members are expected to be reachable within the locality. For triggering the rejoining procedure, we simply assume that "early arrive early serve" concept is applicable in our algorithm. That is, the requester waits for a certain timer to determine from which peer the acceptance. If the Internet-shared group members move far from the original locations, the e2e delay is supposed to be long and can be detected by receiving the *Join Resp.*. Note that we haven't considered the nomadic or mobility case that the same user may access to the system from a remote or a very different location. However, with the above algorithm we can easily handle these particular situations.

6.4 Performance Evaluation

Beside above description, in the following section we use the simulation to evaluate the benefits of interest-shared group management. For simplicity, we only implemented the interest-shared idea into the rejoining procedure. The complete implementation is left for the future study.

6.4.1 Simulation Setup

As described above, interest-shared group management is especially useful for handling dynamic membership changes. Therefore, we intend to test the performance of IGMT in a highly dynamic scenario, namely ParetoChurn scenario (cf. Section 4.5.1). We configure the following parameters for the simulation set up:

- Super Node Max Num: For different group size, the maximum size of super node is accordingly changed. The configuration is very similar to the one in Section 5.2.1.
- Target Overlay Terminal Num: the value varies between 128 and 2048.
- Graceful leaving: 50% of leaving is ungraceful leaving with 0.2 seconds of graceful leaving delay.
- Refresh Timer: 3.5 seconds.
- cpCacheRefreshTimer: 3.5 seconds.

- lifetimeMean: 10,000 seconds.
- deadtimeMean: 2,500 seconds.

Besides, we still use *ParetoChurn* model to configure the dynamic changes. It consists of two subsequent phases as described in Section 4.5.1. According to the definition of Pareto distribution, the availability of each node is:

$$\frac{\text{lifetimeMean}}{\text{lifetimeMean} + \text{deadtimeMean}} = 0.8. \tag{6.3}$$

Actually, above scenario is highly dynamic. Suppose there are 1,000 end hosts within each Churn turn over 200 end hosts join/leave the session.

Besides, the configured underlying network was composed of 4,000 routers. Meanwhile, we used 2,500 backbone routers and 1,500 access routers.

6.4.2 Simulation Results

We use the original DMMP protocol (DMMP), self-improved DMMP protocol (DMMP+) as two benchmarks for the following comparisons. The following analysis includes the measurement of *control overhead*, *data path length* and *packet loss rate*. As we target at being resilient during dynamic changes, above three metrics are reasonable choices for the performance evaluation.

Note the results in the following measurement are different from which have been identified in Section 4.5.2. In the following scenario, nodes frequently join or leave the group following the Pareto distribution. Therefore, the requirement of being resilient is much strict for the protocols.

Control Overhead

Firstly, we compared the average control overhead caused by three protocols. Figure 6.14 depicts the mean value and deviation of the average control overhead measured for DMMP, DMMP+ and IGMT. Among them, DMMP has stable performance because there are only a few *keep alive* messages required. DMMP+ performed not well due to two reasons: 1) DMMP+ needs to periodically optimize the overlay hierarchy, the control overhead increases dramatically when the number of end hosts gets larger; 2) in most cases, partitioned members in DMMP+ rejoin the group through a top-down probing procedure, which increases the control overhead especially in a highly dynamic scenario.

Compared with DMMP+, IGMT achieves quite reasonable performance. The result is predictable since IGMT members can quickly rejoin the group through interest-shared group members maintained in their cache. Therefore, the message exchanges between the candidate parents and the

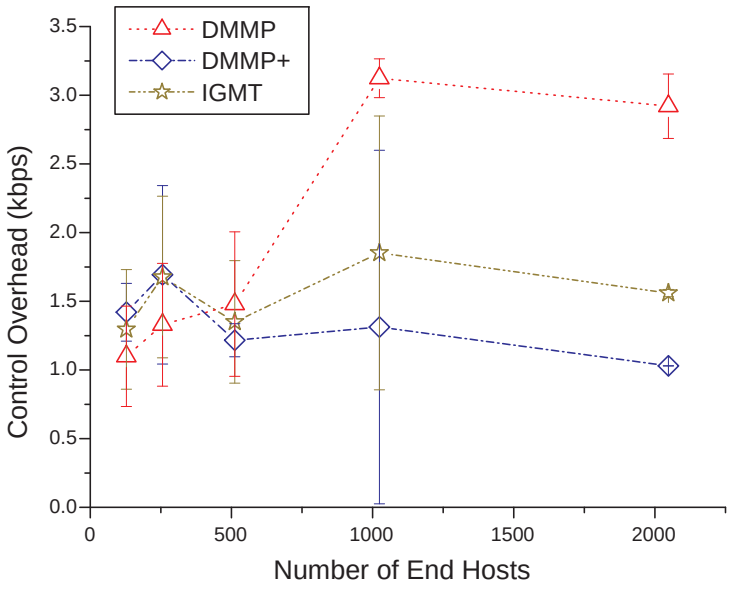

Figure 6.14: Comparisons of Control Overhead.

partitioned members can be largely reduced. When the group size gets larger than 1,000, more benefits of using interest-shared group mechanism can be gained.

However, the control overhead introduced by IGMT is higher than DMMP due to periodic message exchanges for the self-optimizing mechanisms. When the number of group members is larger than 550, the performance of IGMT becomes much better than that of DMMP+ though it still causes higher control overhead than DMMP. Again, it proves that the DMMP framework can benefit from interest-shared group management in a highly dynamic environment for large-scale group users.

Data Path Length

Secondly, we measured the average data path length for three protocols in Figure 6.15. Surprisingly, the original DMMP performed the best out of three protocols. Through a close investigation, we found the reason that DMMP framework considered e2e delay when constructing the clusters, and afterwards no optimization was performed. Thus, the data path length is optimized during the initialization phase according to the e2e delay measurement. Differently, the self-improvement mechanisms promote high capacity nodes the overlay core in order to improve the resilience and reliability of DMMP framework. As a matter of fact, these promoted nodes may not located near the underlying network core and therefore the service deliver path has been sacrificed. In addition, IGMT created similar data path length as DMMP+. Such a result further backs up our hypothesis

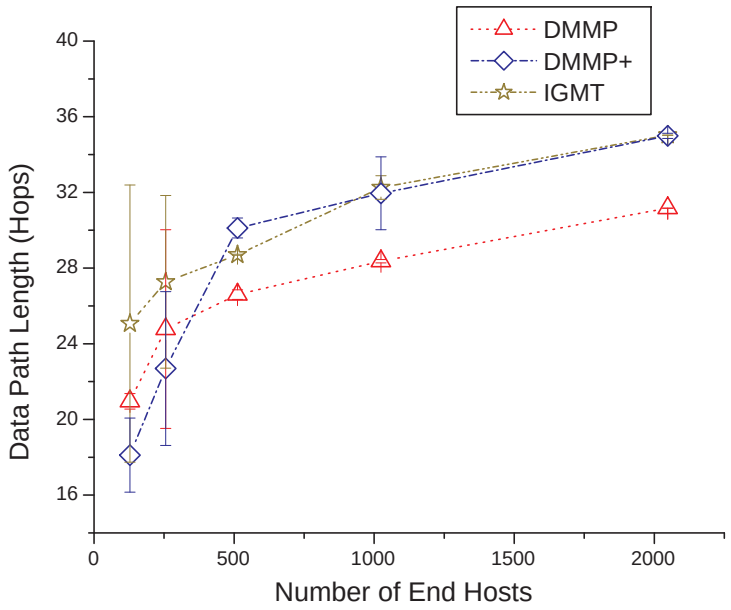

Figure 6.15: Comparisons of Data Path Length.

that interest-shared shortcuts do not harm the performance of DMMP+ and additionally quicken the rejoining procedure.

Hence, we believe that the self-improvement cannot help optimizing the data path length in a highly dynamic scenario. However, the result is still acceptable as optimization of the service path length was not our main target (which has been already mentioned in Section 4.5.2).

Packet Loss Rate

Figure 6.16 shows the comparisons of average packet loss rate introduced by the above three protocols. As expected, the average packet loss of DMMP was much higher than DMMP+ and IGMT since the established overlay hierarchy might not be optimized during membership changes. One cause is resulted from that several redundant packets could be transmitted through the unoptimized overlay core. Moreover, the frequent membership changes caused more instability of data transmission in DMMP.

Compared with DMMP+, IGMT achieved very competitive performance or even less. In IGMT, partitioned members could rejoin the group in a fast and efficient way through the available shortcuts. If partitioned member can rejoin the multicast session in a short time, the packet loss can be alleviated.

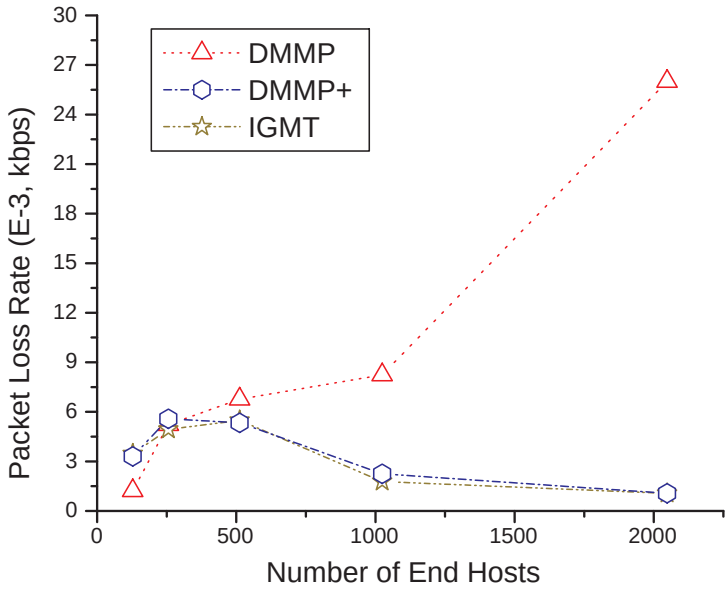

Figure 6.16: Comparisons of Packet Loss Rate.

Overall, IGMT can achieve quite good performance by greatly alleviating the control overhead during rejoining phase and reduce the packet loss rate. Though the data path length caused by IGMT is slightly longer (25%) than that of DMMP, we can argue that our main target was not optimizing the service path length. Otherwise, we can provide less path length by estimating e2e delay before joining or rejoining the multicast session. Nevertheless, it can cause much longer joining/rejoining delay and more packet loss.

6.5 Summary

In this chapter, we firstly extensively investigated the peer-to-peer management mechanisms in one of the first P2P VoD systems, so-called Joost. Based on the experimental results, we summarized the important issues in designing P2P media distribution systems. Also, we observed one important fact that each client has a certain interest of watching IPTV, for instance, a specific interest of selecting channels/programs.

Motivated by the above observation, we proposed Interest-shared Group Management (IGMT) protocol based on the assumption that a large number of users have similar interests of watching videos. The proposed IGMT protocol is light-weight, efficient and decentralized, independent from media distribution systems.

We deployed IGMT into the DMMP-aware clusters, which can largely help the partitioned members to quickly join the group. Through two algorithms, namely shared interest group discovery and shortcuts establishment, an open structure is built on top of the DMMP framework. Besides, we extended the capacity formula to assist partitioned peers to quickly rejoin the multicast session.

The advantage of relying on interest-shared group members to rejoin the group is straight-forward: the high interest similarities between the requester and the group members, high available bandwidth, and relatively longer online time assure the high availability and good quality of service. Importantly, the locality-awareness has been considered during the shortcuts establishment. In this way, these shortcuts do not degrade the quality of established overlay hierarchy.

Through the simulation analysis, three important aspects have been observed:

- The IGMT protocol has achieved better performance than DMMP and DMMP+, regarding the control overhead and the packet loss rate. However, due to the highly dynamic changes IGMT may not fully optimize the data path length.
- When considering the data path length only, DMMP can perform better than DMMP+ and IGMT. It is mainly because the established overlay has taken the e2e delay into consideration. Nevertheless, DMMP causes the highest packet loss rate because no further optimization is performed in the highly dynamic scenario.
- The control overhead of DMMP+ is much higher than that of DMMP but the packet loss rate is much less. It again proves our statement (in Section 4.6) that there is always a tradeoff between having a high quality of data delivery and much more control overhead .

Chapter 7

Conclusion and Future Work

To conclude the work, this chapter summarizes the thesis, enumerates the major contributions with detailed information, and briefly outlines the future work.

7.1 Thesis Summary

Chapter 1 formulated the existing issues in the traditional media streaming system when supporting a large-scale multimedia service over the Internet. Then, the aim of designing a "scalable", "efficient" and "reliable" media distribution framework was highlighted. Finally, it outlined the major contributions and listed the structure of the entire thesis.

Chapter 2 firstly identified the requirements of today's video distribution systems. Secondly, we carefully provided the architectural considerations on how to configure such systems. Meanwhile, we overviewed the relevant prior works on four major media distribution systems: 1) content delivery network; 2) network layer multicasting; 3) application level multicasting; and 4) peer-to-peer media distribution. With respect to each of them, we identified the existing challenges, explored some typical solutions, as well as summarized their advantages and potential weaknesses. Lastly, an overlay multicast-based two-tier media distribution architecture was proposed to meet aforementioned requirements.

Chapter 3 proposed and developed a dynamic mesh-based overlay multicast protocol framework, so-called DMMP. The DMMP framework allows a few end hosts selected and self-organized into an overlay mesh during the multicast initialization phase and also when group member changes, and dynamically maintain such a mesh. The DMMP protocol was illustrated in details, including its properties, messages, and other related features. Moreover, two self-improvement mechanisms were described to further optimize the DMMP-aware overlay hierarchy, especially for handling dynamic membership changes (e.g. member joining/leaving).

Overall, there are four merits achieved in DMMP:

- DMMP is a dynamic two-tier framework constructed by a mesh core and clusters, which is comprised all by end hosts. Here, "dynamic" means that DMMP-aware mesh is resilient to dynamic changes (e.g. member joining/leaving).
- The proposed overlay hierarchy does not need any infrastructure upgrade, and therefore can be deployed into the current Internet. DMMP relies on IP unicast to deliver the packets through decentralized users.
- We combine the available bandwidth and uptime to represent the capacity of each node. In fact, the idea is motivated from economic philosophy that long-staying capable clients who are willing to contribute more to the network, likely get better quality of service than others. Moreover, the heterogeneity of the end hosts has been specifically considered in constructing the DMMP overlay hierarchy.
- To form the cluster, "locality-awareness" is taken into considerations. That is, nearby end hosts are converged into the same cluster. Such a concept brings two major benefits: a) reduction of the serving delay from the server to each receiver; b) reduction of the control overhead and the complexity of the overlay maintenance.

Chapter 4 evaluated the DMMP protocol through both theoretical and simulation-based analysis. The simulation of DMMP protocol was presented as "a proof of the concept" of the proposal. The performance evaluation is two-fold: 1) dynamic scenarios, where group members join or leave the group at will. It identified that k (k-interleaved tree), the number of super nodes, and the number of the end hosts have great impacts on the performance. 2) comparison with ALM approaches in terms of *stress*, *control overhead*, *data path length*, and *packet loss rate*. DMMP performed much better than Narada and competitively good as that of NICE with much less underlying network support. Particularly, DMMP was more adaptive to different group sizes than both NICE and Narada. The efficiency of the data delivery regarding *router stress* and *link stress* were maintained relatively stable when the group size grows.

Chapter 5 proposed a self-improved DMMP protocol (DMMP+) with two self-improvement mechanisms. DMMP+ was designed to gradually optimize the established overlay mesh and clusters. We used the GT-TIM topology generator to configure the underlying network, which can produce a representative abstraction of the real Internet. The performance analysis on *data path quality* and *control overhead and packet loss* ascertained that self-improvement assisted DMMP framework to be scalable, stable and efficient in supporting large-scale media distribution services.

Chapter 6 presented an experimental investigation on one of the first commercial Peer-to-Peer (P2P) VoD systems, Joost. Our interests fell into the peer management which was essential for optimizing the DMMP-aware overlay in highly dynamic circumstances. Through a close investigation on the Joost traffic, we inferred peer management in terms of *time pattern*, *bandwidth consumption*, *popularity impacts* and *locality considerations*. The experimental observations provided insights toward how to build and maintain a P2P overlay in an efficient way. Motivated by above observations, we proposed an interest-shared group management (IGMT) protocol for

DMMP, which optimized the joining and rejoining procedures by establishing shortcuts among interest-shared group members. The simulation results have further demonstrated that IGMT can alleviate the control overhead and packet loss rate during the rejoining procedure.

7.2 Contributions

To summarize the main contributions of this work, we enumerate the efforts which have been contributed through the entire thesis:

- Identifying the video distribution requirements for today's Internet facilities a better understanding of how to build an efficient, scalable and reliable media distribution system. An overlay multicast-based architecture is proposed, which meets most of the identified requirements as well as the deployment needs. This architecture mainly relies on a self-organized overlay network built on top of the IP network to distribute the media service.

- The state-of-the-art of content delivery network, network layer multicast, application level multicast and peer-to-peer media distribution systems is surveyed. It includes the identification of existing challenges in each classified system, exploration of selected solutions, and as well as a brief summary of their advantages and potential weaknesses. The comprehensive analysis on the related works provides a design basis towards a reasonable media distribution system.

- The Dynamic Mesh-based overlay Multicast Protocol (DMMP) framework is proposed, as one of the first systematic proposals in this research field, which addresses the scalability, efficiency and reliability issues in the existing approaches. Extensive theoretical and implemental analysis has proved that DMMP protocol has the potential to support large-scale media application. In particular, the DMMP framework is adaptive to group sizes as the performance is kept relatively stable even when the group size gets larger. Compared with NICE and Narada, DMMP can provide efficient media delivery with less control overhead and less packet loss ratio.

- The self-improved DMMP protocol so-called DMMP+ with two self-improvement mechanisms has been proposed and extensively evaluated. The simulation analysis has demonstrated its usefulness in optimizing the performance of the DMMP framework with respect to the scalability, stability and efficiency of the data delivery hierarchy.

- The analytical and experimental study on Joost peer-to-peer management mechanisms provides insights on the how to construct and maintain a more efficient overlay hierarchy in P2P VoD systems. Since these systems can manage peers in a highly dynamic circumstance, the lessons learned from above investigations illuminates the possible ways of optimizing the DMMP protocols, particularly, to be more resilient during dynamic membership changes.

- Motivated by the above investigations on Joost, the interests-shared group management (IGMT) protocol is developed. It is a light-weight, efficient, and protocol independent scheme. It has been identified to that IGMT can alleviate the control overhead and improve the resilience of the DMMP framework (integrated with DMMP+) since peers having similar interests can create shortcuts to one another. In a highly dynamic scenario, such a protocol can quicken the joining/rejoining procedure through these shortcuts.

7.3 Future Work

The major issues on: (a) scalability, (b) QoS guarantee, (c) resilience and (d) security, form the important building blocks for overlay multicast approaches. In this thesis, we mainly concentrated on addressing the scalability, QoS (refers to available bandwidth and e2e serving delay), and the resilience issues. While security aspect has not been taken into serious considerations, our proposed framework can rely on existing security mechanisms to provide basic protection as shown in Chapter 3.

In the next step, the DMMP framework will study necessary extensions, where e2e QoS, security provision and application adaptability will be likely involved. In the following section, related open areas of research are discussed that present fruitful avenues for the future work:

7.3.1 QoS Provisioning

Most of the existing overlay multicast routing protocols such as [130], [131], [43] have been proposed with varying results regarding performance, cost and implementation. However, most of them are QoS-oblivious; they simply use the available best-effort unicast routing protocols to find paths from the sender to the receivers without considering each member's service requirements. On the other hand, constructing efficient routes among various end hosts is still a big challenge for the following reasons.

- Constructing better performance routes between end hosts using trees built on top of overlay networks can increase the stress (one performance metric to evaluate the efficiency of multicast protocols) on the underlying physical network, since multiple copies of the data may traverse a same physical link.
- End-to-end (e2e) latency of data transported along an overlay path between pairs of end hosts is significantly larger than that in IP-layer routing. Several approaches [132], [133], [134] have been focusing on carefully matching the overlay topology to the underlying physical network in order to reduce the link stress and e2e delay. However, few of them have actually attempted to construct QoS-constrained communicating paths between end hosts.

- Algorithms (e.g. [130], [134]) used for constructing efficient, large-scale multicast trees as well as reducing link stress and latency are typically very complex.

Besides, according to the requirements of media streaming applications [135], [2], the following design issues are typically important for overlay multicast solutions:

- Host heterogeneity: end hosts may vary widely with respect to their capacities, such as CPU power and available bandwidth. Media streaming applications are bandwidth-intensive so that we need to consider the significance of heterogeneity in bandwidths of multicast members.
- Tree construction: multicast nodes may have a wide range of available bandwidth, which can result in a large number of tree shapes under various overlay construction methods.
- Resilience and reliability: it is important for media streaming to detect and recover from failures quickly so that the disruption of service is minimized to those nodes affected downstream. Although media streaming has no constraint requirements on reliability, packet error recovery mechanisms should be performed in a best-effort manner.

To provide QoS using overlay networks, effective and efficient QoS overlay routing, QoS overlay monitoring and QoS overlay restoration are desired. It has been a lot of research efforts conducted in the area of QoS routing, whereas, relatively few efforts have been devoted to QoS monitoring and QoS restoration. Besides, the research in these three areas has progressed independently and inconsistently, which causes further inefficiencies in the utilization of resources.

7.3.2 Security Issues

Security could be an additional consideration since the nature and dynamics of overlay construction in overlay multicast also involve some issues of security protection. Overlay multicast approaches may also involve related issues concerning Denial of Service (DoS) and security protection. For instance, authentication and access control are essential to ensure only legitimate nodes can join the multicast session, as using an untrusted overlay node to deliver services will certainly threaten service availability. In addition, the overlay network has to protect data confidentiality and integrity for overlay nodes and authorized users since data are now relayed by “intelligent” end hosts not ”dumb” routers.

However, security issues in overlay multicast have been received little attention so far. Gothic [136] proposes group access control architecture for secure IP multicast and IP anycast. It assumes that the key server has the global knowledge about the topological locations of each group member in the multicast session. The access control scheme is designed for overlay multicast and can not hold the above assumption since it is not realistic to conjecture each member’s location on the underlying topology.

Cryptographic access control [137] may be a possible solution, which is substantially based on asymmetric cryptography. That is, objects (e.g. messages, data) are encrypted with a private key, and can only be decrypted by someone who holds the public key. The main advantage of doing this is that the access control is inherently distributed, so there is no need to coordinate the state among end hosts and media servers in the system. It also matches the architectural requirements of the overlay multicast system. For example, there are three different functional entities in DMMP, namely, source, super node and cluster member. Based on the cryptographic access control, we may define different types of keys (e.g. group key, multicast session key, cluster key). If a message is encrypted with the group key, the cluster member owning a correct cluster key can decrypt such an object. Once a cluster member leaves the multicast session, only one key - cluster key - needs to be changed and therefore network situation changes (such as multicast members joining/leaving) within a local cluster will not have any impact on other clusters.

To protect data transmitted between group members cryptographic algorithms are commonly used. In order to control such cryptographic algorithms, some key management mechanisms are deployed to encrypt and decrypt information between the communicating entities. A key management scheme has been extensively studied in the context of secure IP multicast. However, the stateful protocols (e.g. LKH [138], ELK [139]) are no longer fit for securing overlay multicast, in which a member must correctly receive all the encryption keys through all previous re-keying operations to decipher the current group key. However, overlay nodes may fail or leave the multicast session at a significant rate, which causes difficulties to receive all the encryption keys for an overlay node. Differently, the stateless protocol (e.g. OFT [140]) may be applied into the overlay multicast system since it only needs a legitimate user to receive the keys in the current re-keying operation to decode the group key. That is, it allows the flexibility for the overlay multicast applications (e.g. allows users to go offline very frequently), but has much higher communication overhead than stateful protocols.

7.3.3 Application Adaptability

Unlike normal data transfer, a streaming media file is huge, thus requires high channel bandwidth. Moreover, streaming media also introduces stringent demand in the timing of packet delivery. Besides QoS for data delivery, adaptability of different media streams may be also considered for optimizing overall network utilization. This is because the stored video is pre-compressed at a certain rate, which may not match the available bandwidth in the network. For example, an attractive solution uses cumulative layering, in which a raw video sequence is compressed into several non-overlapped layers [141]. There is a base layer, which contains the most important features of the video. Additional layers, called enhancement layers, contain complementary information that progressively refines the reconstructed video quality. Accordingly, different layers can serve the receivers with heterogeneous capacities.

Appendix A

Summary of Non-Network Layer Multicast Approaches

Table A.1 summarizes some selected application level multicast protocols and overlay multicast protocols as presented in Section 2.4.3.

The *Target Application* column refers to as which type of applications the proposed multicast solution is due to support. The *Multicast tree construction* column means the main differences at forming the multicast trees for each of multicast solutions. *Recovery mechanism* mainly refers to approaches used to recovery from any failure such as member leaves ungracefully, intermediate node dies without notification. We also compare the application requirements and evaluation criteria such *Scalability Measures*, *QoS considerations*, *Adaption* and other metrics. Then we point out the main contributions of the certain multicast solution compared to existing approaches.

Table A.1 and A.2 summarize four application layer multicast protocols regarding their characteristics and main contributions. Similarly, Table A.3 and A.4 focus on four overlay multicast solutions.

Table A.1: Application Layer Multicast Protocol Summary Table-1

Protocol	Characteristic	Comment
ESM(Narada)	Target application	Supporting small or medium-sized group
	Tree construction	Refinement-based mesh-first
	Optimization Objective	End-to-end latency
	Optimization Technique	Adding and dropping of links depending on perceived gain in utility
	Routing Algorithm	Distance vector routing algorithm
	Group Formation	Peers self-organize into a mesh and each node shares group information with neighbor nodes
	Recovery Mechanism	While detecting the existence of a partition, Narada repair it by adding at least one overlay link to reconnect the mesh
	Scalability Measures	No considerations on node degree and link stress
	QoS considerations	Dynamic optimization to the mesh by performing end-to-end latency measurements
	Adaption	Out-of-band bootstrap; random request to join the group; leaving information is propagated to the rest of group members along the mesh
	Data Delivery	Distance vector protocol on top of the mesh & a new routing cost (transient forward)
	Main contributions	It firstly demonstrates multicast functionalities can be performed at application layer
NICE	Target application	Low bandwidth large-scale data streaming applications
	Tree construction	Cluster-based source-specific tree
	Optimization Objective	Low control overhead and low latency for large groups
	Optimization Technique	Hierarchical clustering
	Routing Algorithm	Hierarchical cluster-based trees
	Group Formation	Assign members into different layers; hosts in each layer are partitioned into a set of clusters
	Recovery Mechanism	Each survivor of the cluster independently selects a new leader if the leader of the cluster leaves the group
	Control Overhead	Max: $O(klog_k^N)$, average: $O(k)$f if the number of members in a cluster is k
	Scalability Measures	No considerations on node degree
	QoS considerations	Cluster attachments by identifying the closest member in the super-cluster
	Adaption	Cluster splitting and merging operations;
	Data Delivery	Hierarchical structure implicitly defines data delivery paths
	Main contributions	It uses hierarchical structure to support low bandwidth multi-sender applications with a very large member populations

Table A.2: Application Layer Multicast Protocol Summary Table-2

Protocol	**Characteristic**	**Comment**
HMTP	Target application	Low cost tree for multi-sender applications
	Tree construction	Refinement-based tree-first protocol
	Optimization Objective	End-to-end delay (member-to-member RTT)
	Optimization Technique	Parent-switching by re-running the join procedure
	Routing Algorithm	Bi-directional shared distribution tree is used to interconnects IP-multicast-enabled islands
	Group Formation	All Designated Members self-organize into a bi-directional shared tree
	Loop Problem	Maintenance of root path
	Triangle Problem	A heuristic of triangle optimization
	Scalability Measures	No considerations on node capabilities
	Control overhead	$O(k)$, if the maximum number of children a node can handle is k
	Recovery Mechanism	Surviving members detect the failure by noticing missing periodic refresh message, and repair the tree by running the repair algorithm
	QoS considerations	HMTP uses member-to-member round-trip time as the only distance metric in tree building
	Main contributions	It firstly interconnects IP-multicast-enabled islands by using application layer multicast solutions
HostCast	Target application	Delay-sensitive applications
	Tree construction	Refinement-based tree-first protocol
	Optimization Objective	Root latency
	Optimization Technique	Switch-parents + parent-children swap
	Routing Algorithm	HostCast builds a single source-rooted multicast tree using measurement-based approach to optimize the bandwidth and end-to-end delay between the source and the various group members
	Group Formation	Group members self-organize into the source-based overlay tree
	Loop Problem	Primary root path + secondary root path
	Triangle Problem	Condition-constrained solution
	Recovery Mechanism	HostCast picks up secondary parents or other nodes along the primary root path as candidate parent nodes
	Scalability Measures	distance (end-to-end delay)
	Control overhead	$O(k^2)$, if the maximum number of children a node can handle is k
	QoS considerations	Measurement-based approach to derive the overlay path QoS conditions and provide QoS to multicast users
	Main contributions	It uses a simple probe technique to obtain the underlying topology; periodic refinements are used to improve the performance

Table A.3: Overlay Multicast Protocol Summary Table-1

Protocol	Characteristic	Comment
Overcast	Target application	Single source streaming media content distribution
	Tree construction	Single-source multicast tree
	Optimization Objective	To maximize bandwidth to the root for all nodes
	Optimization Technique	Constantly revaluation of positions in the tree
	Routing Algorithm	A single source-rooted multicast tree is built, using end-to-end measurements to optimize bandwidth between the source and various group members
	Group Formation	Single source multicast; each node joins group at a Overcast node
	Loop Problem	By acquiring the knowledge of nodes' ancestors
	Triangle Problem	no
	Recovery Mechanism	"Up/down" protocol is used to keep track of nodes up and down the tree
	Scalability Measures	Scalability of the root enables to handle a large amount of service requests
	QoS considerations	Bandwidth constraints
	Adaption	Maintenance of global status at the root of the distribution tree
	Data delivery	Data are moved between parent and children using TCP streams
	Main contributions	implementing a protocol to handle dynamic changes of the distribution tree
OMNI	Target application	Single source media streaming applications
	Tree construction	A set of Multicast Service Nodes(MSNs) are deployed in a network, acting as overlay relay agents
	Optimization Objective	low-latency overlay paths
	Optimization Technique	Local transformations + probabilistic transformations
	Routing Algorithm	Degree constrained average latency algorithm based source-rooted spanning tree
	Group Formation	Joining from the root by measuring the unicast latency between itself and the root
	Loop Problem	No
	Triangle Problem	No
	Recovery Mechanism	One child of departing MSN is prompted up to replace its position if the maximum subtree latency is reduced most
	Scalability Measures	Degree bounded directed spanning tree
	QoS considerations	Overlay latency
	Adaption	MSNs periodically adjust their positions to adapt to network conditions
	Data Delivery	From Multicast core to subtrees
	Main contributions	Modeling degree bounded directed spanning tree to fit for large scale data distributions

Table A.4: Overlay Multicast Protocol Summary Table-2

Protocol	Characteristic	Comment
TOMA	Target application	Large-scale multicast services
	Tree construction	Deploying Multicast Service Overlay Network (MSON) as the backbone service domain
	Optimization Objective	Network resource
	Optimization Technique	Aggregated tree + dynamic group-tree matching algorithm
	Routing Algorithm	Shortest IP-layer path
	Group Formation	Aggregated tree within MSON + cluster formation outside MSON
	Loop Problem	No
	Triangle Problem	No
	Recovery Mechanism	not specified
	Scalability Measures	Aggregated multicast trees
	QoS considerations	Average percentage bandwidth overhead
	Adaption	Handling Otype messages and maintaining a multicast routing table
	Data Delivery	Multicast distribution trees are built on top of the MSON
	Main contributions	Aggregated multicast approach enables multiple groups share one delivery tree within the backbone
oStream	Target application	On-demand asynchronous media streaming
	Tree construction	K-array tree
	Optimization Objective	Required server bandwidth
	Optimization Technique	Hierarchical stream merging + asynchronous multicast
	Routing Algorithm	Source-based minimal spanning tree
	Group Formation	Tree construction based on request for media + adjustments to time of request
	Loop Problem	No
	Triangle Problem	No
	Recovery Mechanism	Recovery from node leaving is performed locally
	Scalability Measures	Server capacity for buffering data
	QoS considerations	Data buffering at relay nodes and at end host
	Adaption	Join procedure based on partial knowledge of the tree
	Data Delivery	Either from media server or from other end hosts
	Main contributions	Take advantage of strong buffering capabilities of end hosts

Appendix B

An Experimental Analysis of Joost P2P VoD System

B.1 Introduction

In the recent few years, IPTV has gained a tremendous popularity in the operators and users as well as a lot of attention from the research community. For residential users, such a service is often provided in conjunction with Video-on-Demand (VoD) and may be bundled with other Internet services such as Voice over IP (VoIP). Traditionally, when a client user selects a program, a point-to-point unicast connection is established between a decoder (aka *set top box*) and delivering media server. Most of current VoD services mainly rely on content distribution networks (CDNs) [142] or local streaming proxies to increase system scalability and to alleviate the delay experienced by end users. However, their system performance and deployment still becomes a key challenge as the number of clients increases. Especially, if a flash crowd [143] occurs, servers can be easily overloaded.

To address above issues, peer-to-peer technologies (e.g. swarming [144]) have been recently employed to support VoD services. However, a P2P VoD system is rather challenging to design than any other P2P media streaming systems [54] because, in addition to providing low playback delays, the system allows users arriving at arbitrary time to watch videos. The heterogeneous arrivals reduce sharing opportunities and increase the complexity of video distribution mechanisms. Besides, in order to support VoD functions (e.g. backward) system requires a certain local space to store the downloaded video. Thus, another issue is how to allocate such a storage and use the storage in an efficient way to support VoD functionalities. Therefore, it is essential to understand how to design a VoD architecture that scales smoothly to support a large number of users, while maintaining high video quality and reasonable operational costs. It is also critical for ISPs, network administrators, and content owners to consider the service requirements for supporting P2P VoD systems.

Before the investigations, we first raise some questions about the Joost system:

- *What is the Joost architecture?* What are the key components in such an architecture? Which functions are performed by these components? How these functions can be achieved?
- *How are the characteristics of Joost network traffic?* Which kind of protocols does Joost use? What is the fraction of outgoing and incoming data traffic? What is the fraction of network traffic that a peer receives is control traffic?
- *What are the characteristics of peer behaviors?* At what rates does a peer download from and upload to its partners? How are the partnerships different for a University LAN client and a DSL residential client?
- *How are the Peer-to-Peer technologies used in Joost?* How does the peer selection performed in Joost? During the peer selection, has Joost considered locality? Whether heterogeneity is considered? How about the fairness of contribution? Has Joost considered peer adaption during dynamic changes?
- *How about the performance with different network conditions?* How about the performance of high capacity nodes, if they can offer high network access speed? If the Joost client suffers from low, unstable network conditions, will the performance dramatically degrade?

We are more interested in the peer management mechanism, and therefore we seek to answer the last two main questions by data-driven analysis (cf. Chapter 6). The answers to the first three questions are shown in this chapter.

The rest of the chapter is organized as follows. In Section B.2 we infer the Joost system, such as the server architecture, protocols used in the system. Section B.3 describes the key components of Joost software. Section B.4 discusses key Joost functions like installation, bootstrapping, reconnection, channel switching and VoD functionalities. In Section B.5, we review related work. Finally, we conclude this chapter and plan future work in Section B.6.

B.2 Charting the Joost System

Joost [145], created by N. Zennström and J. Friis, co-founders of Skype [146] and Kazaa [60], is one of such systems for providing high-quality and comprehensive VoD services using P2P TV technologies. The current version offers 20,000+ TV shows through 400+ channels. Based on the experiments [126] and information in [122], we infer the Joost system architecture using a top-down approach: 1) abstract a high-level hierarchy from the overall architecture; 2) investigate the functionalities performed by each component of the hierarchy.

Figure B.1 shows five types of servers, one Joost peer manager*, and various Joost clients. There are other servers taking charge of value-added services, for example, instant chat service (*scd.joost.com*) and advertisement server (*lux-cdn-lo-4.joost.net*). Because the fundamental functions are our focus, these additional servers have been omitted from the discussion.

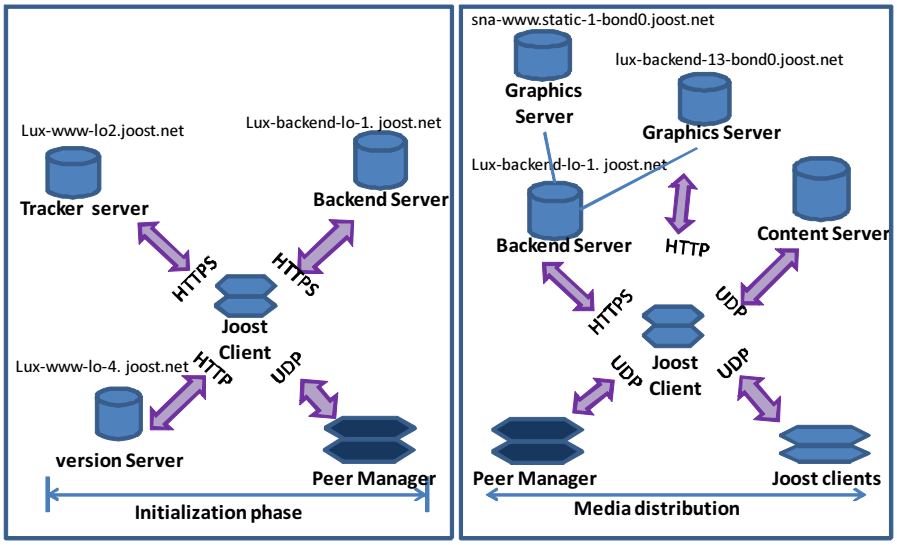

Figure B.1: Joost Architecture.

B.2.1 Joost Servers

Identifying the Joost servers facilitates our understanding of the peer management mechanisms because their behaviors can be differentiated from that of arbitrary peers. In this paper, we use the term of peer and client interchangeably.

As shown in Figure B.1, *lux-www-lo4.joost.net* is the version server that is responsible for checking the current version of the software during login. For instance, JCs sent HTTP 1.1 GET requests for getting the latest software version. The second type of server is called tracker server (i.e. *lux-www-lo2.joost.net*) whose sole responsibility is to keep track of its group members and helps bootstrapping new peers.

Channel management in Joost differs largely from any other P2P VoD system (e.g. PPLive [147]) as it builds an API for the channel list using XULRunner that provides Joost clients more interactive experience. Therefore, except for channel list management (e.g. updates) Joost system has to handle on-screen menus involving some animations overlayed on top of video. Because

*Although peer manager is named as super node in [122], in our experiments we identify that its functionality is peer management. It is not responsible for relaying/forwarding media data to other peers. Therefore, we call it peer manager in this paper.

of the complexity of channel management, the channel management is not performed by a single server in Joost, but a server cluster. That is, the backend server, *lux-backend-lo-1.joost.net*, answers for controlling channel list requests and keeping load balance among cluster servers, whereas other cluster servers perform particular tasks (e.g. channel graphs downloading). Some of them can be named channel graphic servers. For instance, there were two servers, *lux-backend-13-bond0.joost.net* and or *sna-www.static-1-bond0.joost.net* from which a Joost Client (JC) downloaded the channel graphics instead of directly from the backend server. Nevertheless, the scalability might be a major concern for the future development if the number of users dramatically increases in a short period.

The last type of Joost server is content server. Joost did distributedly deploy a serious number of content servers over the network. During our experiments, we observed the following server sites: (1) 4.71.105.0/24 (sna-Itsnode-*x*-bondx-*x*.joost.net); (2) 4.71.174.0/24 (IPsoft); (3) 212.187.185.0/24 (Icy-Itsnode-*x*-bondx-*x*.joost.net). Here, *x* varies from 0 to 10. The first and third IP address site is owned by Level 3 Communication INC [148] which has been selected by Joost to support on demand Internet TV. The second IP space group belongs to IPsoft service provider.

B.2.2 Peer Manager

Another major different design from other P2P networks (e.g. Skype) or overlay multicast solutions [149] is that all peer managers in Joost are only used for controlling and helping new peers find available contributing peers. Based on above understanding, the traffic from peer managers can be easily extracted from the overall traffic in Section 6.2.3.

In fact, it is quite efficient and reasonable that Joost's peer management is isolated from the media distribution. According to [150], some universities have already banned Skype from their campuses, while some other universities and government agencies require that their users disable supernode functionality to avoid relaying traffic outside the stub Autonomous Systems (ASs). We conjecture that Joost designers have taken the issue into consideration. Moreover, strategically deployed peer managers not only ease the membership management but also improve the reliability of transmission. For example, if a super node in Skype leaves ungracefully, all the other peers relying on it will be unavoidably affected. To summarize, Joost super nodes perform the following three basic functions in most cases.

- After bootstrapping JCs first contact super node, which directs clients to available peers. Peers are either JCs or Joost content servers.
- For on-demand video functions, super nodes periodically exchange some small UDP packets with clients. We believe that these UDP packets are used for peer management, such as keep-alive probing.
- Additionally, channel switching requires the JC to talk to the super node. At that time, super

node most likely helps it finding available peers to fetch the new media data.

B.2.3 Protocols

Joost is rather a complicated system, to evaluate the efficiency of video transmission it is necessary to identify the protocols used for control traffic and for media data traffic.

Table B.1 depicts the main protocols used in the Joost. All video packets are encoded in UDP and the size is exactly 1104 bytes. Using UDP for video transmission is not reliable, however, it quite cost-effective and time-efficient especially for large-scale peer-to-peer networks. Besides the media transmission, peers frequently communicate with each other by sending UDP probes (64 bytes). During the channel switching, peers contact peer managers by sending UDP packets. Since April 2007, when port number 4166 was assigned by IANA as the official UDP port used for Joost, all media data and some control messages (e.g. peer management) are sent from Joost servers through 4166. Tracing these specific port numbers facilitates our following experiments.

Table B.1: Main Protocols in Joost System.

Protocol	Functionality	Packet Size
UDP	video distribution	1104 bytes
	content probe (peer to peer)	~ 64 bytes
	channel switching	< 1000 bytes
	VoD interactions	< 150 bytes
HTTP	software version	
	client → server	~ 64 bytes
	server → client	< 500 bytes
	channel management	
	client → server	~ 64 bytes
	server → client	<= 1518 bytes
HTTPS	administrative management	
	client → server	64 bytes
	server → client	< 500 bytes

HTTP is used for checking the software version and updating the channel list during bootstrapping and initialization phases. When the JC browses the channel category, the channel graphs are downloaded in real time through HTTP from the graphics servers.

When the JC re-connects to the Joost system, HTTPS is used for the administrative management duties which include checking software version, channel list updating, obtaining trackers.

B.3 Key Components of Joost Software

According to our following experiments, during its operation a Joost client performs one or more of the following actions: listen on particular ports for incoming traffic; store media data into its local cache; maintain a table of other peers called a host cache; use Advanced Video Codec (AVC); determine if it is behind a NAT or firewall; and functions required by additional features, such as instant messaging. This section discusses the key components involved in these actions.

B.3.1 Ports

During the installation and bootstrapping, Joost client contacts some HTTP/HTTPS servers initially. It will be further described in Section IV.

Upon the first initialization, the Joost client (JC) randomly chooses a port number through which the JC can subsequently communicate with other peers and Joost servers. Such a port (noted as *JC_P*) is usually some high port (e.g. 57929). Once the port number is determined during the first run, subsequent media transactions will always use this port no matter the JC restarts or reboots. Besides, the Joost client listens on this port for incoming requests from other peers. To send media data to other Joost clients, the JC also uses this port.

In April 2007, port number 4166 was assigned by the Internet Assigned Numbers Authority (IANA) [151] as the official TCP and UDP port used for Joost. Since then, all media data and some of the control messages (e.g. peer management) are sent through 4166 from Joost servers (seen Section V for detailed information). Looking into the specific port number facilitates our following experiments.

To summarize the above analysis, the different ports used in Joost network traffic are depicted in Table 1.

Table B.2: Ports in Joost System

Protocol	Joost Server	Joost Client	Super Node
HTTP	80	HTTP port	
HTTPS	443	HTTPS port	
TCP	4166	*JC_P*	4166
UDP	4166	*JC_P*	4166

Here, *super node* is not a Joost Client, but a delicately deployed entity mainly for peer management and peer lookup purposes in Joost, as described in Section V.

B.3.2 Video Codecs

Joost claims to use "an H.264 codec for video encodings (aka AVC, aka MPEG-4 Part 10, aka ISO/IEC 14496-10) called CoreAVC, created by CoreCodec" [152]. However, in our experiments we did not observe H.264 as shown in Figure B.2. We conjecture that RTP dynamic (payload type: 96-127) is the codec H.264 (payload type: 99) for Joost video encoding as the freely available analyzers we found were unable to distinguish it. Furthermore, it was observed that Joost used G.711, G.726, G.728, G.729, G.723.1 and GSM for audio codecs (Figure B.2), which allow frequencies between 8,000-90,000 Hz to pass through. These codecs have been developed by ITU-T [153]. We conjecture that Joost followed the RTP specification [154] for the implementations.

Protocol	Percentage	Bytes
UDP	89,862%	2.648.613.744
RTP Dynamic	6,282%	185.161.606
HTTPS	0,707%	20.849.981
RTCP	0,491%	14.481.126
G.711	0,430%	12.662.605
IMAP	0,218%	6.415.720
G.729	0,202%	5.939.730
G.726	0,198%	5.826.491
G.723.1	0,197%	5.816.021
RTP	0,196%	5.774.642
G.728	0,196%	5.768.628
H.263	0,195%	5.741.342
H.261	0,195%	5.735.162
GSM	0,194%	5.721.785
ARP Request	0,140%	4.134.996
SAP Request	0,063%	1.848.128
ICMP Dest...	0,049%	1.447.486
TCP	0,042%	1.251.630
HTTP	0,041%	1.194.632

Figure B.2: Example of Protocols in Joost System.

B.3.3 Local Video Cache

A JC for Windows XP users stores the media data in its local cache as "anthill_cache" at System_Disk(e.g. C:)\\ Documents and Settings\< XP user>\Application Data\Joost\ anthill\anthill_cache. For Windows Vista users, local cache is stored in System_Disk:\\uers\<Vista User>\AppData\ Roaming\Joost\anthill. Joost claims that just like a "Skylib" (Skype library) enabling voice and chat services on the P2P layer, Joost runs on a media streaming library the company has nicknamed "Anthill". Here, Anthill [155] is an agent-based peer-to-peer system to support the media distribution services. A brief overview of Anthill is shown in Appendix.

The cache size depends on which and how long programs have been played. Each time a new program is chosen, the size of the cache will automatically increase. In our experiment, it was more than 2 GB. Therefore, we believe that user's system resources will be significantly occupied if the JC continues to watch different channels.

If we assume that the local cache did completely store the played video, the JC should watch the old program directly from the local cache. However, when we had disabled the Internet connection, the program surprisingly stopped, even if the particular program has been watched 1 minute ago. We guess that although some media data have been stored locally, it still requires a kind of codec from the remote server or a encryption key (e.g. AES key) authorized by the Joost server to access the video file. To prove these conjectures, we made additional experiments.

We launched a new channel and at that moment, the local cache was empty. After the whole channel was watched, the size of cache file grew up to 1.7 GB and the average download speed was 518 kbps. If we turned off the JC and restarted it, the download speed was dramatically dropped down to 11 kbps when the same channel was watched. Moreover, the size of local cache increased only 5.88% during the second watching time.

B.3.4 Host Cache

Similar to what was observed in the Skype analysis [146] and [156], host cache is a list of Joost super nodes IP address and port pairs that JC builds and refreshes periodically. The JC for Windows XP stores the host cache as an XML file "shared.xml" in System_Disk:\\Documents and Settings\<XP User>\Application Data\Joost\anthill. A Joost client for Windows Vista stores it in System_Disk:\\users\<Vista User>\AppData\ Roaming\Joost\anthill.

B.3.5 NAT and Firewall

We detected that a random port was configured at the first login time and kept in use for the subsequent media transmission. As video packets are sent over UDP (as shown in Section V), we conjecture that Joost uses a modified STUN [157] protocol to determine the type of firewall and NAT it may behind, similar to what was observed in [146]. The NAT and firewall traversal related information is stored in the share.xml file.

More information related with STUN is stated in Appendix.

B.4 Joost Functions

All the experiments were performed for Joost version Beta 1.0. Joost was installed on Windows XP and Windows Vista machines. The Windows XP was Intel Pentium Dual-Core 1.73 GHz processor

with 1.00 GB RAM. The Windows Vista was equipped with AMD Althon X64 processor with 1.00 GB RAM.

B.4.1 Installation

One Joost server was involved in installation phase: lux-backend-lo-1.joost.net (89.251.4.75). The client sent a HTTP 1.1 GET request to this Joost server and downloaded a SQLite [158] file (zelos2.sqlite) which is the initial channel list. See Appendix B for complete messages.

This channel list is stored in System_Disk:\\Program files\Joost\defaults\profile\zelos2.sqlite, currently fixed to 1.35 MB size and 33 channels). Clearly, SQLite is used for the Joost channel database management, which is a self-contained, embeddable, zero configuration SQL database engine. Figure B.9 shows a snapshot of the initial Joost channel list.

HTTP messages involved in Joost installation

This section shows the message dump of HTTP 1.1 GET request that a JC sent to backend.joost.net (89.251.4.175) and the responses it received.

In case it was the first time after installation, the client sent a HTTP 1.1 GET request containing the URI of the resources. The requested file is zelos2-0.12.zip which can be unzipped into zelos2-0.12.sqlite as described in Section 6.

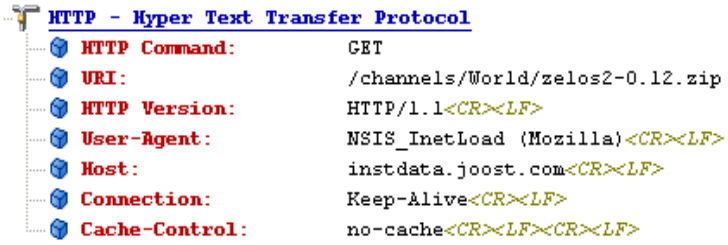

Figure B.3: Joost Installation: HTTP GET.

Then, there was a 200 OK response received by the client for the above GET request.

At that moment, the local cache, node identity and the listening port number through which the client will communicate with other peers were not yet configured. We found that there was no local cache file, no share.xml file in which node identity and port would be configured. In this chapter, we use the term of peer and client interchangeably.

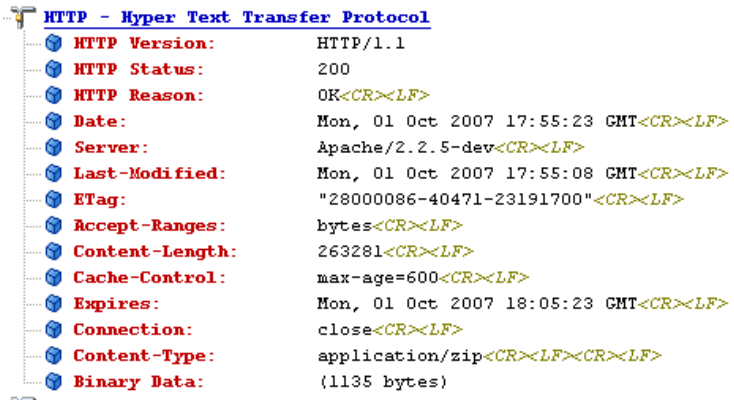

Figure B.4: Joost Installation: HTTP OK.

B.4.2 Bootstrapping

Totally, three Joost servers and two Joost super nodes were responsible for the bootstrapping procedure, by which the listening port was configured and the channel list was updated (System_Disk:\\ Documents and Settings\<XP User>\ Application Data\Joost\Profiles*.default\zelos2.sqlite, 1.76 MB, 45 channels).

Firstly, the JC communicated with lux-www-lo-2.joost.net server (89.251.2.85) over HTTPS. We conjecture that it is a kind of tracker server. In case the newcomer contacts the tracker, it will receive some available super node addresses and possibly some content server addresses. Then, a HTTP GET request was sent to lux-www-lo4.joost.net (89.251.2.87) server for getting the latest software version. The snapshot of these HTTP messages is shown as follows.

HTTP messages involved in Joost initialization

During the bootstrapping procedure, JC sent a HTTP 1.1 GET request to instdata.joost.com which is actually the version server (89.251.2.87). The current software version is 0.13.0 (Beta 1.0). The fragmentation of the HTTP GET request is shown as follow.

```
HTTP Command: GET
URI:          /?version=0.13.0
HTTP Version: HTTP/1.1
Host:         instdata.joost.com
User-Agent:   Mozilla/5.0 Windows; U; Windows NT 5.1; ...)
```

Figure B.5: Joost Initialization: HTTP GET.

Then, a 200 OK response was sent from the version server to the client.

```
HTTP Version:  HTTP/1.1
HTTP Status:   200
HTTP Reason:   OK
Server:        Apache/2.2.5-dev
Cache-Control: max-age=600
```

Figure B.6: Joost Initialization: HTTP OK.

Besides, the lux-backend-lo-1.joost.net server (the same server involved in the installation) was also involved in bootstrapping the client by sending packets over HTTPS.

Finally, JC started to contact some of Joost super nodes, for instance, lid-snode-1-eth0.joost.net (89.251.0.16), lid-snode-2-eth0.joost.net (89.251.0.17) and lux-snode-1-bond0.joost. net (89.251.4.71), possibly to obtain the list of other available clients and begin transacting video contents. Before long, the running JC has already started communicating with other peers besides Joost severs.

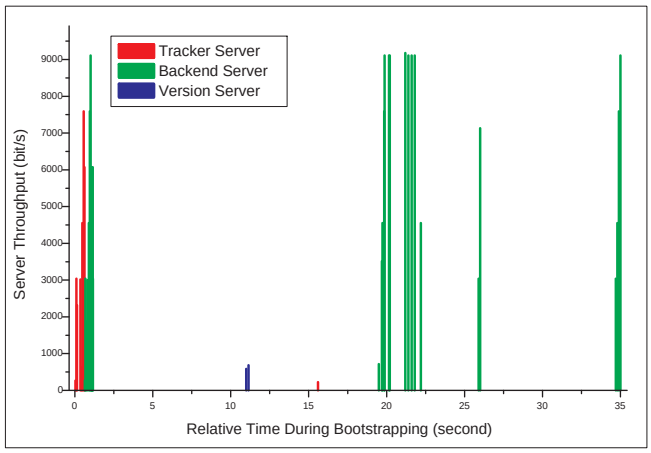

Figure B.7: Joost Server Throughput during Bootstrapping.

Figure B.7 shows the throughput of above three Joost servers during bootstrapping. At the very beginning, the tracker server (lux-www-lo-2.joost.net) helped bootstrapping the new client. Clearly, after a short period the tracker server was not involved in the subsequent communication. Then, the backend server (lux-backend-lo-1.joost.net) appeared and continuously sent a large amount of data to the client, we guess, in order to update the channel list. At some point of the bootstrapping procedure, the version server (lux-www-lo4.joost.com) checked the version of Joost software.

B.4.3 Reconnection

We restarted the JC and attempted to observe the peer behaviors during client reconnection. The above-mentioned initialization process occurred again with two exceptions. One is that the version server might not appear if the interval was short and the other exception is the port number that was negotiated when first connecting to the Joost network was stored in the share.xml file and reused. If the version server really appeared, checking software version used HTTPS instead of HTTP.

Furthermore, the JC attempted to communicate with peers from which it has downloaded content previously. This was done by sending some small UDP probe (64 bytes) to other clients, which in turn would reply with another small UDP probe (64 bytes). Afterwards, media data was continuously downloaded from some connected clients.

HTTP messages involved in Joost reconnection

This section shows the message dump of HTTP 1.1 GET request that a JC sent to channel graphs server and the responses it received.

The HTTP GET containing the URI of requesting Compressed Scalable Vector Graphics File (.svgz) is shown below. At that moment, the channel list management was redirected to lux-backend-13-bond0. joost.net (89.251.4.153).

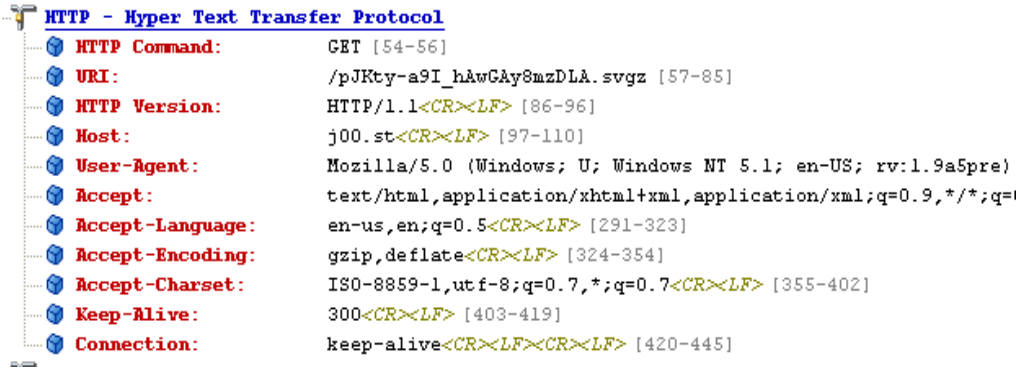

Figure B.8: Joost Reconnection: HTTP GET.

Then, there was an OK response received by the client for the above GET request.

The snapshot of the Joost channel database in Figure B.9.

URN	ID	Title	Description	Logo
urn:tv:banquet.theve	104000g	Banquet		
urn:tv:alliance_atlan	450055	Alliance Atlantis Sci-		
urn:tv:thomas_lucas	246000i	SpaceRip: Space, S		
tag:clouseau.thever	client-popular	Popular		
urn:tv:zelos:hodgepo	hodge-podge	Joost Links	Joost Links are shor	
urn:tv:ministry_of_so	065001u	Ministry of Sound TV	What's current in clu	https://thumbs.ops.thevenic
urn:tv:reuters_us.the	069000h	Reuters	Reuters brings you t	https://thumbs.ops.thevenic
urn:tv:banquet.theve	104000g	Banquet	Banquet, The Cham	https://thumbs.ops.thevenic
urn:tv:northone.thev	01900b6	Fifth Gear Shortcuts	Popular award-winni	https://thumbs.ops.thevenic
urn:tv:tvp.thevenice	024000h	Joost Suggests	Welcome to Joost! H	https://thumbs.ops.thevenic
urn:tv:viacom_mtv_i	0860002	MTV	MTV needs no intro	https://thumbs.ops.thevenic
urn:tv:universal_bug	136005g	Bugeye Music	Live concert footage	https://thumbs.ops.thevenic
urn:tv:alliance_atlan	0450055	Alliance Atlantis Sci-	Check out the finest	https://thumbs.ops.thevenic
urn:tv:viacom_paran	0910077	Paramount Pictures	Disfrutan de esta co	https://thumbs.ops.thevenic
urn:tv:viacom_paran	0910079	Paramount Pictures	Geniessen Sie eine	https://thumbs.ops.thevenic
urn:tv:viacom_paran	0910078	Paramount Pictures	Retrouvez une colle	https://thumbs.ops.thevenic
urn:tv:otf.thevenicep	020006n	Off the Fence Docs	Docs on demand - s	https://thumbs.ops.thevenic
urn:tv:mondo_media	203001g	Mondo Mini Shows	Mondo Mini Shows	https://thumbs.ops.thevenic
urn:tv:gong.thevenic	01300a7	GONG	GONG is the first an	https://thumbs.ops.thevenic
urn:tv:aardman.thev	0440001	Aardman Animations	Multi award-winning	https://thumbs.ops.thevenic
urn:tv:tv5.thevenice	1660045	TV5MONDE PLUS	TV5MONDE is well	http://j00.st/MclwcwUL7Gc
urn:tv:the_onion.the	163000f	Onion News Networ	The Onion News Ne	https://thumbs.ops.thevenic
urn:tv:indivisual.thev	0590003	The Soccer Channe	Relive the greatest g	https://thumbs.ops.thevenic
urn:tv:warnermusicg	08200ma	Best of Today	The Best of Today E	https://thumbs.ops.thevenic
urn:tv:warnermusicg	08200m9	Stadium Rockers	Stadium Rockers is	https://thumbs.ops.thevenic

Figure B.9: Initial Channel List.

B.4.4 Channel Switching

We observed that at the moment of channel switching the JC contacted firstly with some super nodes. These super nodes IP addresses and ports have been obtained from the tracker server (lux-www-lo-2.joost.net) during the initialization. For example, the JC contacted lid-snode-2-eth0.joost. net (89.251.0.17) and lux-snode-1-bond0.joost.net (89.251.4.71) over UDP. From these super nodes, the JC may obtain some address lists including related content servers and possibly some other clients who were watching the same channel but ahead of this particular client. Once the JC received such a list, it attempted to contact them by immediately sending UDP requests. At the same time, other super nodes continued to send the client available address lists. When the selected channel started playing, the JC periodically exchanged messages with these super nodes

over UDP. We believe that the super nodes are responsible for redirecting clients to content servers or peers during channel switching. Moreover, they periodically exchange messages with clients, possible for peer management or acquiring keying materials to eventually watch the video stream.

In the current version of Joost, the function of local video buffer is not supported. That is, when the client pauses the video it stops downloading. There are some claims that Joost should have a small amount of buffer in order to avoid the stuttering and temporary freezes [152]. However, observing from the fact that most of users frequently switch channels, the current solution may save resources in case of short-term switching as it does not maintain local buffers.

B.4.5 VoD Functionalities

Unlike file sharing or live media streaming, each JC is more "selfish" in the sense that it only cares about contents after its current playing position, which is often different from other peers. The peer can only download from those whose playback positions are ahead, or from who have already watched the program. Instead, itself can help peers which join later. However, as each Joost client can change its playback position at any time, which differs from many other P2P streaming systems, it becomes difficult to optimize the overall VoD system. For example, the "rarest-first" strategy [144] in BitTorrent is not applicable here.

As a result, the VoD aspect attracted our particular interests. After repeating several experiments, we come up with the following conclusions.

First, in Joost system each media file was broken down into fixed-time chunks and each chunk is encrypted. During our experiments, if the fast forward interval was smaller than 5 seconds the JC may continuously play without waiting. However, if the interval is large it took 5 – 10 seconds to start playing. To illustrate our observation, we suppose that each media file is divided into multiple 10-second play time chunks, but the exact size of the chunk is unknown. As shown in Figure B.10, each chunk includes an anchor which is a dedicated marker for encrypted media data similar to I-frame in MPEG [11]. When a seek is triggered in a client (i.e., control bar is moved to a backward position), the client will always search for the closest anchor in the local video cache if it is already downloaded. Otherwise, it firstly sets a new anchor and requests new data from other peers.

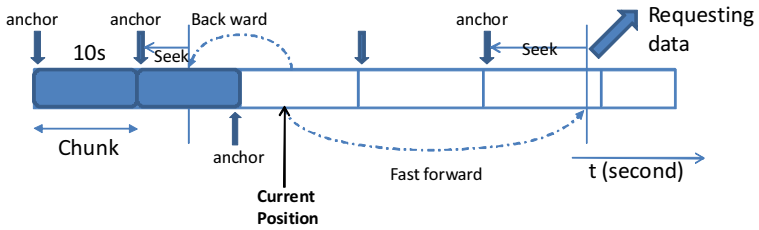

Figure B.10: On-demand Video Functions.

Second, if the JC drags the control bar into any specific position, it communicates with one of the super nodes, for example, lid-snode-2-eth0.joost.net (89.251.0.17) or lid-snode-1-eth0. joost.net (89.251.0.16) in all our experiments.

To prove that those super nodes support VoD functionalities, we traced the first super node during the periodic (every 20 seconds) actions of "fast forward" (10-minute period of video) within the same program. As shown in Figure B.11, each time the JC dragged the control bar, there was a large amount of traffic sent from the super node. Otherwise, the traffic from the super node was quite low compared to the "fast forward period". By analyzing the traced data, we found that UDP was used to carry the traffic and the average received packet size was 137 bytes and the average size of sent packets was 141 bytes (all below 150 bytes). Therefore, we suppose that these packets are only used for control, not for media transmission. Furthermore, we conjecture that the updated lists, which contains information about peers having already received the on-demand contents, are encoded in these packets.

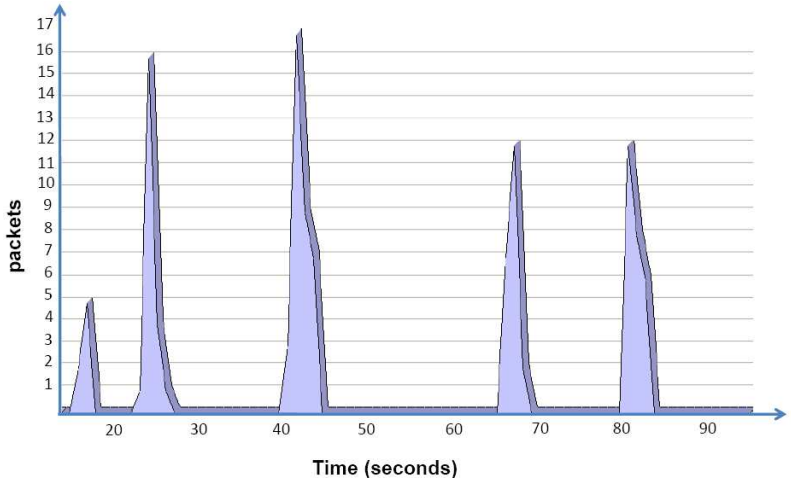

Figure B.11: VoD Functionality.

B.5 Related Work

Peer-to-Peer IPTV architecture requires a minimal infrastructure support and can offer the possibility of rapid deployment at low cost. In terms of simultaneous users, one of the most successful IPTV deployments has employed P2P streaming architecture. Hei *et al.*[54] provided an overview of P2P streaming system (e.g. PPLive [147]) and characterized P2P IPTV behavior and traffic profiles at packet, connection and application levels. Among most popular IPTV services, Video-on-Demand (VoD) provides video, audio and data service triggered by users' selection. However, most of existing work about P2P VoD systems was concentrated on the protocol design and the implementation [159], [160], [161]. Different from them, we provide a real measurement analysis

on Joost functions, peer selection and locality awareness. Furthermore, the Joost architecture is quite different from, even more complex than, media streaming architecture described in [54]. In Joost, each JC is more "selfish" since it only cares about the content behind its current playback position. The VoD functionalities give the users more flexibilities, and hence make the system more difficult to analyze.

Joost uses some similar P2P technologies as used in Skype which critically depends on the a peer-to-peer network formed by super nodes. Any participating node initially is a standard node, and some of them will be promoted to super nodes according to a number of factors including spare bandwidth and public reachability. Baset *et al.* [146] analyzed various aspects of the Skype protocol such as login, NAT and firewall traversal, call establishment, media transfer, codecs and conferencing under three network setups. In general, the paper provided a detailed analysis of Skype user experience and peer behaviors. Guha *et al.* [156] analyzed node dynamics and churn in Skype's peer-to-peer overlay. Further, it identified that Skype was fundamentally different from earlier P2P systems like P2P file sharing networks. There are three main differences between Skype and Joost. First, Joost architecture requires more than a login server. Second, Joost super nodes are not responsible for relaying traffic to standard nodes. Third, as observed in [146] the voice packet size varied between 40 and 120 bytes, however, the Joost video packet size was much larger (1104 bytes). Joost analysis may help to understand how P2P technologies for such VoD services should be provisioned.

Hall *et al.* [162] provided a measurement study of Joost in May, 2007. This paper explained an understanding of Joost's application behavior, network behavior, and peer behavior. However, there are several major differences between their work and our work. First, their experiments were taken based on Joost version 0.9.2 which is already out-of-date. Differently, our experimental studies were performed by Joost beta 1.0 which is more stable and integrated version. Second, through our analysis we inferred the Joost architecture and key components, however, [162] did not provide such information. Third, we designed three typical scenarios in order to further investigate the performance of locality awareness, bandwidth capacity and peer selection. Nevertheless, [162] only examined the locality awareness through three experiments. Lastly and more importantly, we analyzed the Joost VoD functionalities which are the main difference from other media streaming systems. Therefore, we can argue that we provide the first comprehensive analysis of Joost P2P VoD service.

B.6 Summary and Conclusions

Joost is one of the first commercial P2P VoD systems which can provide high quality on-demand TV based on P2P technologies. Unlike live media streaming system, each VoD client is more "selfish" in the sense that it only cares about contents after its current playing position, which is often different from other peers. The peer can only download from those whose playback positions are ahead, or from who have already downloaded the program. Instead, itself can help peers which join later than itself. However, as each client can change its playback position at any time, which

differs from many other P2P streaming systems, it becomes difficult to optimize the overall VoD system. For example, the “rarest-first” strategy [144] in BitTorrent is not applicable here. In this chapter we have made a first step towards discovering various aspects of the Joost functions and behavior by analyzing the network traffic and by being acquainted with some of the open software used in Joost. Without a surprise, Joost and Skype have some P2P mechanisms and supporting techniques in common.

This Appendix provides a first trial on investigating the Joost peer behaviors and media distribution mechanisms. Current Joost P2P code maybe neither AS-level aware nor end-to-end latency aware for the peer selection. However, the exact peer lookup and selection techniques that Joost used for peer management is still not clear. Our guess is that it uses a combination of swarm techniques in BitTorrent and prefix awareness.

Bibliography

[1] M. M. Hefeeda and B. K. Bhargava, “On-Demand Media Streaming Over the Internet,” The 9th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS’03), 2003.

[2] K. Andreev, B. M. Maggs, A. Meyerson, and R. K. Sitaraman, “Designing Overlay Multicast Networks For Streaming,” The 15th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA’03), San Diego, California, USA, 2003.

[3] G. Peng, “CDN: Content Distribution Network,” Technical Report TR-125, Experimental Computer Systems Lab, Department of Computer Science, State University of New York, Stony Brook, NY, 2003.

[4] A.-J. Ballardie, P.-F. Francis, and J. Crowcroft, “Core Based Trees,” ACM SIGCOMM Computer Communication Review, Vol. 23, No. 4, pp.85-96, 2003.

[5] X. Ding and K. Roy, “A Novel Bitstream Level Joint Channel Error Concealment Scheme for Realtime Video over Wireless Networks,” in Proc. of IEEE INFOCOM’04, 2004.

[6] Y. Cui, B.-C. Li, and K. Nahrstedt, “oStream: Asynchronous Streaming Multicast in Application-Layer Overlay Networks,” IEEE Journal on Selected Areas in Communications, Special Issue on Recent Advances in Service Overlay Networks, Vol. 22, No. 1, pp. 91–106, 2004.

[7] J.-F. Kurose and K.-W. Ross, “Computer Networking – A Top-Down Approach Featuring the Internet,” Addison Wesley, ISBN 0-321-26976-4, pp. 631–632, 2005.

[8] T.-D. Little and A. Ghafoor, “Synchronization and storage models for multimedia objects,” IEEE Journal on Selected Areas in Communication, Vol.8, No.3, pp.413-427, 1990.

[9] T. Sikora, “MPEG Digital Video Coding Standards,” Digital Electronics Consumer Handbook. McGraw Hill, 1997.

[10] M. Ghanbari, “Video Coding-an introduction to standard codecs,” The Institution of Electronical Engineers, 1994.

[11] C. Herpel and A. Eleftheriadis, "MPEG-4 Systems: Elementary Stream Management," Image Commmunication Journal, Tutorial Issue on the MPEG-4 Standard, Vol. 15, No. 4, pp.299-320, 2000.

[12] L.-G. Didier, "MPEG: a video compression standard for multimedia applications," Communications of the ACM, Vol. 34, No. 4, pp. 46–58, 1991.

[13] P.-N. Tudor, "MPEG-2 Video Compression," Electronics and Communication Engineering Journal, tutorial, 1995.

[14] F. Pereiera and E. Touradj, "The MPEG-4 Book," Prentice Hall, 2002.

[15] A.-N. Netravali and B.-G. Haskell, "Digital Pictures: Representation, Compression, and Standards," Plenum Press, Chapter 3, 1995.

[16] D.-P. Wu, Y.-W. Hou, W.-W. Zhu, and et al., "Streaming Video over the Internet: Approaches and Directions," IEEE Trans. on Circus and Systems for Video Technology, Vol. 11, No. 2, pp.282-300, 2001.

[17] J.-G. Apostolopoulos, T. Wong, W. Tan, and et al., "On multiple description streaming with content delivery networks," in Proc. of IEEE INFOCOM, 2002.

[18] J. Gemmell, H.-M. Vin, D.-D. Kandlur, and et. al., "Multimedia storage services: a tutorial," IEEE Computer Magzine, Vol. 28, No. 5, pp. 40–49, May 1995.

[19] A. Guha, "The evolution to network storage architectures for multimedia applications," in Proc. IEEE international Conf. on Multimedia Computing and Systems, pp. 68–73, 1999.

[20] D.-H. Du and Y.-J. Lee, "Scalable server and storage architectures for video streaming," in Proc. IEEE International Conf. on Multimedia Computing and Systems, pp. 62–67, 1999.

[21] G.-A. Gibson and R.-V. Meter, "Network attached storage architecture," Communications of the ACM, Vol.43, No.11, pp.1060-3425, 2000.

[22] R. Steinmetz and K. Nahrstedt, "Multimedia: Computing, Communications and Applications," Upper Saddle River, NJ: Prentice Hall, 1995.

[23] A. Basso and J. Ostermann, AT&T Labs research, August 2003. [Online]. Available: http://www.patentstorm.us/patents/6602299/claims.html

[24] V. Vakali and G. Pallis, "Content Distribution Network: Status and Trends," IEEE Internet Computing, IEEE Computer Society, Vol. 7, No. 6, pp.68-75, 2003.

[25] J. Dilley, B. Maggs, J. Parikh, H. Prokop, and B. Weihl, "Globally Distributed Content Delivery," IEEE Internet Computing, IEEE Computer Society, Vol. 6, No. 5, pp. 50-58, 2003.

[26] "Akamai Technologies," www.akamai.com.

[27] A.-M. K. Pathan and R. Buyya, "A Taxonomy and Survey of Content Delivery Networks," Grid Computing and Distributed Systems (GRIDS) Laboratory, University of Melbourne, Parkville, Australia, 2006.

[28] D. Waitzman, C. Partridge, and S. Deering, "Distance Vector Multicast Routing Protocol," Network Working Group, November 1998. [Online]. Available: http://www.rfc-archive.org/getrfc.php?rfc=1075

[29] Agilent Technologies, "Testing IP Multicast," Technical Paper, 2002.

[30] D. Estrin, D. Farinacci, A. Helmy, and et al., "Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification," Network Working Group, June 1997. [Online]. Available: http://www.rfc-archive.org/getrfc.php?rfc=2117

[31] A. Adams, J. Nicholas, and W. Siadak, "Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised)," RFC 3973, 2005.

[32] B. Fenner, M. Handley, H. Holbrook, and I. Kouvelas, "Protocol Independent Multicast - Spare Mode (PIM-SM) Protocol Specification (Revised)," RFC 4601, 2006.

[33] S. Bhattacharyya, "An Overview of Source-Specific Multicast (SSM)," RFC 3569, 2003.

[34] K. Almeroth, "The evolution of multicast: From the MBone to inter-domain multicast to Internet2 deployment," IEEE Network, 2000.

[35] C. Diot, B. Levine, J. Lyles, H. Kassem, and D. Balensiefen, "Deployment issues for IP multicast service and architecture," IEEE Network, 2000.

[36] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel, "ALMI: An Application Level Multicast Infrastructure," San Francisco, CA, USA, 2005.

[37] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, and S. Khuller, "OMNI: An Efficient Overlay Multicast Infrastructure for Real-time Applications," Computer Networks, Special Issue on Overlay Distribution Structures and their Applications, Vol. 50, No. 6, pp. 826-841, 2006.

[38] S. Banerjee, B. Bhattacharjee, C. Kommareddy, and et al., "Scalable Application Layer Multicast," SIGCOMM'02, 2002.

[39] L. Lao, J.-H. Cui, and M. Gerla, "TOMA: A Viable Solution for Large-Scale Multicast Service Support," In IFIP Networking 2005, May 2005.

[40] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content-addressable Network," SIGCOMM'01, August 2001.

[41] Y.-H. Chawathe, S. G. Rao, and H. Zhang, "A case for end system multicast," In ACM SIGMETRICS, 2001.

[42] ESM. [Online]. Available: http://esm.cs.cmu.edu/

[43] S.Banerjee, B. Bhattacharjee, C. Kommareddy, and et al., "Scalable Application Layer Multicast," SIGCOMM'02, 2002.

[44] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan, "Resilient Multicast using Overlays," IEEE Transactions of Networking, Vol. 14, No. 2, pp.237-248, 2005.

[45] B. Zhang, S. Jamin, and L. Zhang, "Host multicast: a framework for delivering multicast to end users," in Proc. IEEE INFOCOM'02, New York, USA, 2002.

[46] P.-E. Black, "Greedy Algorithm," in Dictionary of Algorithms and Data Structures, U.S. National Institute of Standards and Technology, 2005.

[47] Z. Li and P. Mohapatra, "HostCast: A New Overlay Multicast Protocol," in Proc. of IEEE ICC'03, 2003.

[48] Y. Tsang and R. Nowak, "Optimal Network Tomography," in Proc. of IEEE INFOCOM 2005 Student Workshop, Miami, FL, 2005.

[49] J. Jannotti, D.-K. Gifford, and K.-L. Johnson, "Overcast: Reliable Multicasting with an Overlay Network," In Proc. of OSDI'00, San Diego, California, 2000.

[50] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh," SOSP'03, Bolton Landing, New York, USA, 2003.

[51] J.-H. Cui, L. Lao, D. Maggiorini, and M. Gerla, "BEAM: A Distributed Aggregated Multicast Protocol Using Bi-directional Trees," in Proc. of IEEE ICC2003, Anchorage, Alaska, USA, May 2003.

[52] X. Fu and J. Crowcroft, "GONE: an Infrastructure Overlay for Resilient, DoS-Limiting Networking," in Proc. of ACM NOSSDAV, Newport, Rhode Island, USA, May 2006.

[53] S. Jun and M. Ahamad, "Incentives in BitTorrent Induces Free Riding," in SIGCOMM'05 Workshop, 2005.

[54] X. Hei, C. Liang, Y. Liu, and K. Ross, "A Measurement Study of a Large-scale P2P IPTV System," IEEE Transactions on Multimedia, Vol. 9, No. 8, pp.1672-1687, 2007.

[55] YouTube, http://www.youtube.com.

[56] J. Liebeherr, M. Fidler, and S. Valaee, "A Min-Plus System Interpretation of Bandwidth Estimation," in Proc. of IEEE INFOCOM, May 2007.

[57] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and replication in unstructured peer-to-peer networks," in Proc. of the 16th ACM International Conference on Supercomputeing (ICS'02), 2002.

[58] B. Yang and H. Garcia-Molina, "Improving search in peer-to-peer networks," in Proc. of the 22th International Conference on Distributed Computing Systems (ICDCS'02), 2002.

[59] M. Waldman, R. AD, and C. LF, "Publius: A robust, tamper-evident, censorship-resistant web publishing system," in Proc. of the 9th USENIX Security Symposium, 2000.

[60] Kazaa, http://www.kazaa.com/.

[61] W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmer, and T. Risch, "Edutella: A p2p networking infrastructure based on rdf," in Proc. of the 12th International Conference on World Wide Web., 2003.

[62] I. Stocia, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," SIGCOMM'01, San Diego, California, USA, 2001.

[63] S. Ratnasamy, P. Francis, M. Handley, and R. Karp, "A scalable content-addressable network," in Proc. of SIGCOMM, 2001.

[64] P. Druschel and A. Rowstron, "Past: A large-scale, persistent peer-to-peer storage utility," in Proc. of the 8th Workshop on Hot Topics in Operating Systems, 2001.

[65] B. Zhao, J. Kubiatowicz, and A. Joseph, "Tapestry: An infrastructure for fault-tolerant wide-area location and routing," in Tech. Rep. UCB/CSD-01-1141, Computer Science Division, University of California, 2001.

[66] I. Clarke, O. Sandberg, and B. Wiley, "Freenet: A distributed anonymous information storage and retrieval system," in Proc. of the Workshop on Design Issues in Anonymity and Unboservability, 2000.

[67] M. Jovanovic, "Modeling large-scale peer-to-peer networks and case study of Gnutella," Master Thesis, Department of Electronical and Computer Engineering and Computer Science, University of Cincinnati, 2000.

[68] M. Feldman, K. Lai, I. Stoica, and J. Chuang, "Robust Incentive Techniques for Peer-to-Peer Networks," in Proc. of ACM EC04, May 2004.

[69] K. Park, S. Pack, and T. Kwon, "Climber: An Incentive-based Resilient Peer-to-Peer System for Live Streaming Services," the 7th International Workshop on Peer-to-Peer Systems (IPTPS), 2008.

[70] M.-J. Freedman, C. Aperjis, and R. Johari, "Prices are Right: Managing resources and incentives in peer-assisted content distribution," the 7th International Workshop on Peer-to-Peer Systems (IPTPS), 2008.

[71] M. Bawa, H. Deshpande, and H. Garcia-Molina, "Transience of Peers & Streaming Media," ACM SIGCOMM Computer Communications Review, Vol. 33, No. 1, pp.107-112, 2003.

[72] A. Kuzmanovic, S. Floyd, and K.-K. Ramakrishnan, "Adding Explicit Congestion Notification (ECN) Capbility to TCP's SYN/ACK Packets," Internet Draft, draft-ietf-tcpm-ecnsyn-05.txt, 2008.

[73] T. Montgomery, "A Loss Tolerant Rate Controller for Reliable Multicast," Technical Report, NASA-IVV-97-011, 1997.

[74] M. Kalman and B. Girod, "TCP-like congestion control for layered multicast data transfer," in Proc. of IEEE INFOCOM'98, 1998.

[75] Q. Guo, Q. Zhang, W. Zhu, and et al., "Send-adaptive and receiver-driven video multicasting," IEEE International Symposium on Circuits and Systems, Sydney, Australia, May 2001.

[76] L.-S. Lam, J.-Y. Lee, S.-C. Liew, and W. Wang, "A Transparent Rate Adaption Algorithm for Streaming Video over the Internet," 18th International Conference on Advanced Information Networking and Applications, AINA'04, 2004.

[77] P.-D. Cuetos and K.-W. Ross, "Adaptive Rate Control for Streaming Stored Fine-Grained Scalable Video," Network and Operating System Support for Digital Audio and Video (NOSSDAV), pp. 3–12, May 2002.

[78] S. Varadarajan, H.-Q. Ngo, and J. Srivastava, "An adaptive, perception-driven error spreading scheme in continuous media streaming," in Proc. of Distributed Computing Systems for Video Technology, pp. 475–483, 2000.

[79] D. Wu, Y.-T. Hou, W. Zhu, and et al., "On end-to-end architecture for transporting MPEG-4 video over the Internet," IEEE Trans. on Circus and Systems for Video Technology, Vol. 10, pp. 923–941, 2000.

[80] M. Luby, L. Vicisano, J. Gemmell, and et al., "Forward Error Correction (FEC) Building Block," RFC 3452, Network Working Group, 2002.

[81] S. Lin, D.-J. Jr., and M.-J. Miller., "Automatic-Repeat reQuest error-control schemes," IEEE Communications Magazine, Vol. 22, No. 12, pp. 5–17, 1984.

[82] Q. Zhang and S.-A. Kassam, "Hybrid ARQ with selective combining for fading channels," IEEE J. Select. Areas Commun., Vol.17, Taipei, Taiwan, pp. 867–880, May 1999.

[83] P. Bucciol, E. Masala, and J.-C. Martin, "Perceptual ARQ for H.264 video streaming over 3G wireless networks," IEEE Communications Society, Vol. 3, pp.1288-1292, 2004.

[84] C.-H. Wang, R.-I. Chang, J.-M. Ho, and et al., "Rate-sensitive ARQ for real-time video streaming," in Global Telecommunications Conf. (GLOBECOM), 2001.

[85] A. Narula and J.-S. Lim, "Error concealment techniques for an all-digital high-definition television system," Proc. SPIE Conf. Visual Commu. And Image Proc., pp. 304–314, 1993.

[86] Y. Wang and Q.-F. Zhu, "Error control and Concealment for video communications: A Review," Proc. of the IEEE, Vol. 86, No. 5, pp.974-997, May 1998.

[87] P. Ge and P.-K. Mckinley, "Comparisons of Error Control Techniques for Wireless Video Multicasting," 21st IEEE International Performance, Computing, and Communications Conference, 2002.

[88] R. Braden, L. Zhang, S. Berson, and et al., "Resource Reservation Protocol (RSVP)Version 1 Functional Specification," RFC 2205, 1997.

[89] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-time Applications," RFC 1889, Network Working Group, 1996.

[90] H. Schulzrinne, A. Rao, and R. Lanphier, "Real Time Streaming Protocol (RTSP)," RFC 2326, Network Working Group, 1998.

[91] J. Rosenherg, H. Schulzrinne, G. Camarillo, and et al., "SIP: Session Initiation Protocol," RFC 2974, 2002.

[92] M. Handley, C. Perkins, and E. Whelan, "Session Announcement Protocol," RFC 2974, 2000.

[93] F. Benevenuto, C. Costa, M. Vasconcelos, and et al, "Impact of Peer Incentives on the Dissemination of Polluted Content," in Proc. of the 2006 ACM symposium on Applied computing, 2006.

[94] T.-V. Nguyen, F. Safaei, P. Boustead, and C.-T. Chou, "Provisioning overlay distribution networks," Computer Networks, Vol. 49, No. 1, pp.103-118, 2005.

[95] G. Tan, S. Jarvis, and D. Spooner, "Improving Fault Resilience of Overlay Multicast for Media Streaming," IEEE International Conference on Dependable Systems and Networks (DSN-2006), Philadelphia, USA, 2006.

[96] N. Hu and P. Steenkiste, "Evaluation and Characterization of Available Bandwidth Probing Techniques," IEEE JSAC Special Issue in Internet and WWW Measurement, Vol. 21, No. 6, pp.879-894, 2003.

[97] J. Lei, X. Fu, and D. Hogrefe, "DMMP: A New Dynamic Mesh-based overlay Multicast Protocol Framework ," in Proc. of IEEE Consumer Communications and Networking Conference - Workshop on Peer-to-Peer Multicasting (P2PM 2007), 2007.

[98] M. Jain and C. Dovrolis, "End-to-end available bandwidth: measurement methodology, dynamics, and relation with tcp throughput," IEEE/ACM Transaction on Networking Vol. 11, No. 4, pp.537-549, 2003.

[99] J. Strauss, D. Katabi, and F. Kaashoek, "A measurement study of available bandwidth estimation tools," in Proc. of ACM SIGCOMM Conference on Internet Measurement, 2003.

[100] J. Liebeherr, M. Fidler, and S. Valaee, "A Min-Plus System Interpretation of Bandwidth Estimation," in Proc. of IEEE INFOCOM, May 2007.

[101] Y.-H. Chu, S.-G. Rao, and H. Zhang, "A case for End System Multicast," SIGMETRICS, 2000.

[102] H. Xie, Y. Yang, A. Krishnamurthy, Y. Liu, and A. Silberschatz, "P4P: Provider Portal for Applications," in Proc. of Sigcomm, 2008.

[103] F. B. D.R. Choffnes, "Taming the Torrent: A Practical Approach to Reducing Cross-ISP Traffic in P2P Systems," in Proc. of IEEE Sigcomm, May 2008.

[104] J. Lei, X. Fu, X. Yang, and D. Hogrefe, "A Dynamic Mesh-based overlay Multicast Protocol (DMMP)," Internet Draft, draft-lei-samrg-dmmp-03.txt, 2008.

[105] D. Liu and P. Ning, "Establishing Pairwise Keys in Distributed Sensor Networks," in Proc. of 10th ACM Conference on Computer and Communications Security (CCS'03), Washington D.C., October 2003.

[106] M. Yang and Z. Fei, "A Proactive Approach to Reconstructing Overlay Multicast Trees," IEEE INFOCOM'04, 2004.

[107] G. Carsten, K.-H. Vik, and P. Halvorsen, "Multicast Tree Reconfiguration in Distributed Interactive Applications," 2nd IEEE International Workshop on Networking Issues in Multimedia Entertainment (NIME'06), 2006.

[108] L. Lao, J.-H. Cui, and M. Gerla, "A Framework for Realistic and Systematic Multicast Performance Evaluation," Elsevier Journal of Computer Networks, Vol. 50, No. 12, pp.2054-2070, 2006.

[109] M. Castro, M.-B. Jones, A.-M. Kermarrec, A. Rowstron, M. Theimer, H. Wang, and A. Wolman, "An Evaluation of Scalable Application-level Multicast Built Using Peer-to-peer Overlays," in Proc. of IEEE INFOCOM'03, San Francisco, California, USA, 2003.

[110] K. Calvert, M. Doar, and E. Zugura, "Modeling Internet Topology," IEEE Communications Magazine, Vol. 35, No. 6, pp.160-163, 1997.

[111] E.-W. Zegura, K. Calvert, and S. Bhattacharjee, "How to Model an Internetwork," in Proc. of IEEE INFOCOM'96, San Francisco, CA, 1996.

[112] S. Banerjee and B. Bhattacharjee, "Analysis of the NICE Application Layer Multicast Protocol," UMIACS Technical Report TR 2002-60 and CS-TR 4380, 2000.

[113] ns-2 simulator, http://www.isi.edu/nsnam/ns/.

[114] OMNeT++, http://www.omnetpp.org/index.php.

[115] O. Herrera and T. Znati, "Modeling Churn in P2P Networks," Simulation Symposium, 2007.

[116] Z. Zhang, S. Chen, Y. Ling, and R. Chow, "Resilient Capacity-Aware Multicast Based on Overlay Networks," in Proc. of IEEE ICDCS, May 2005.

[117] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: High-Bandwidth Multicast in Cooperative Environments," Bolton Landing, New York, USA, 2003.

[118] S. Bhattacharyya, "An Overview of Source-Specific Multicast (SSM)," IEEE Network, 2003.

[119] K. Calvert and M. Doar and E.-W. Zegura, "Modeling Internet Topology," IEEE Communications Magazine, Vol. 35, No. 6, pp.160-163, 1997.

[120] D. Knuth, "The Stanford GraphBase: A Platform for Combinatorial Computing," Addison-Wesley, 1994.

[121] J. O. Source, http://opensource.joost.net/.

[122] J. N. Architecture, http://www.scaryideas.com/video/2362.

[123] Wireshark, http://www.wireshark.org.

[124] WildPackets, http://www.wildpackets.com/products/omnipeek/overview.

[125] WhereIsIp, http://www.jufsoft.com/whereisip/.

[126] J. Lei, L. Shi, and X. Fu, "An experimental analysis of joost peer-to-peer vod service," Technical Report No. IFI-TB-2007-03, Institute for Computer Science, University of Goettingen, Germany, ISSN 1611-1044, 2007.

[127] H. Yu and D. Zheng and B.-Y. Zhao and W. Zheng, "Understanding User Behavior in Large-scale Video-on-Demand Systems," in Proc. of EuroSys'06, 2006.

[128] OmniPing Professional, http://www.manasoft.com/manasoft/omniping_pro.aspx.

[129] S. Kim and E.-A. Fox, “Interest-based User Grouping Model for Collaborative Filtering in Digital Libraries,” ICADL 2004, pp. 533-542, 2004.

[130] Z. Xu, C. Tang, and Z. Zhang, “Building topology-aware overlays using global soft-stat,” in the 23th international conference on distributed Computing Systems (ICDCS), Rhode Island USA, May 2003.

[131] M. Yang and Z.-M. Fei, “Cooperative Failure Detection in Overlay Multicast,” Networking 2005, May 2005.

[132] C. Wong, M. Gouda, and S. Lam, “Secure Group Communication Using Key Graphics,” in Proc. of SIGCOMM 1998, Vancouver, British Columbia, 1998.

[133] A. Perrig, D. Song, and D. Tygar, “ELK, a new protocol for efficient large-group key distribution,” in Proc. of IEEE Symp. On Security and Privacy’01, Oakland CA, May 2001.

[134] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas, “Multicast Security: A Taxonomy and Some Efficient Constructions,” in Proc. of IEEE INFOCOM’99, May 1999.

[135] J. Lei, X. Fu, I. Juchem, and D. Hogrefe, “Architectural thoughts and requirements considerations on video streaming over the Internet,” Technical Report No. IFI-TB-2005-06, 2005.

[136] P. Judge and M. Ammar, “Gothic: A Group Access Control Architecture for Secure Multicast and Anycast,” in Proc. of IEEE INFOCOM’02, New York, USA, 2002.

[137] A. Harringto and C. Jensen, “Cryptographic Access Control in a Distributed File System,” in Proc. of SACMAT’03, Como, Italy, 2003.

[138] C. Wong, M. Gouda, and S. Lam, “Secure Group Communication Using Key Graphics,” in Proc. of SIGCOMM’98, Vancouver, British Columbia, 1998.

[139] A. Perrig, D. Song, and D. Tygar, “ELK, a new protocol for efficient large-group key distribution,” in Proc. of IEEE Symp. On Security and Privacy’01, Oakland CA, May 2001.

[140] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas, “Multicast Security: A Taxonomy and Some Efficient Constructions,” in Proc. of IEEE INFOCOM’99, May 1999.

[141] J. Liu, B. Li, and Y.-Q. Zhang, “Adaptive Video Multicast over the Internet,” IEEE Computer Society, Vol. 10, No. 1, pp.22-33, 2003.

[142] J.-M. Almeida, D.-L. Eager, M. Frris, and M.-K. Vernon, “Provisioning Content Distribution Networks for Streaming Media,” in Proc. of INFOCOM’02, 2002.

[143] I. Norros, B. Prabhu, and H. Reittu, "Flash crowd in a file sharing system based on random encounters," in Proc. of ICST/ACM workshop on Interdisciplinary systems approach in performance evaluation and design of computer & communications sytems, 2006.

[144] B. Cohen, "Incentives build robustness in bittorent," in 1st Workshop on the Economics of Peer-2-Peer Systems, Berkley, CA, 2003.

[145] Joost, http://www.joost.net/.

[146] S.-A. Baset and H. Schulzrinne, "An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol," in Proc. of INFOCOM'06, Barcelona, Spain, 2006.

[147] PPLIve, http://www.pplive.com.

[148] L. . Communications, http://www.level3.com/.

[149] C.-L. Abad, W. Yurcik, and R.-H. Campbell, "A survey and comparison of end-system overlay multicast solutions suitable for network-centric warfare," Battlespace Digitization and Network-Centric Systems IV, Vol. 5441, pp.215-226, 2004.

[150] R. Paul, "More universities banning Skype," 2006. [Online]. Available: http://arstechnica.com/news.ars/post/20060924-7824.html

[151] Internet Assigned Numbers Authority (IANA), http://www.iana.org/.

[152] Joost Support Forum, http://www.joost.com/support/faq/Technology.html.

[153] ITU Telecommunication Standardization Sector (ITU-T), http://www.itu.int/ITU-T/.

[154] H. Schulzrinne and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control," RFC 3551, Network Working Group, 2003.

[155] O. Babaoglu, H. Meling, and A. Montresor, "Anthill: A Framework for the Development of Agent-Based Peer-to-Peer Systems," 22nd IEEE International Conference on Distributed Computing Systems (ICDCS'02), 2002.

[156] S. Guha, N. Daswani, and R. Jaine, "An Experimental Study of the Skype Peer-to-Peer VoIP System," in Proc. of IPTPS'06, 2006.

[157] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy, "STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)," RFC 3489, Network Working Group, 2003.

[158] R. Hipp, "SQLite," http://www.sqlite.org/.

[159] T. Do, K. Hua, and M. Tantaoui, "P2vod: providing fault tolerant video-on-demand streaming in peer-to-peer environment," In Proc. of IEEE ICC 2004, Paris, France, 2004.

[160] S. Guha, N. Daswani, and R. Jaine, "Peer-assisted vod: Making internet video distribution cheap," in Proc. IPTPS 2007, 2007.

[161] Y. Guo, K. Suh, J. Kurose, and D. Towsley, "P2cast: peer-to-peer patching for video on demand service," Multimedia Tools Application, vol. 33, No. 2, pp.109-129, 2007.

[162] Y.-J. Hall, P. Piemonte, and M. Weyant, "Joost: A Measurement Study," Carnegie Mellon University, May 2007.

Appendix C

Curriculum Vitae

Personal Data

Name:	Jun Lei
Date of Birth:	13 August 1979
Place of Birth:	Zhejiang Province, China
Nationality:	Chinese
Address:	Institute for Informatics George-August-Universität Göttingen Lotzestrasse 16-18 37083 Göttingen Germany

Research Interests and Topics

- Overlay Networks, Peer-to-Peer (P2P) Networking
- Multicast Routing
- Mobility (global and localized mobility management)
- Internet QoS
- Network Security

Education and Stages

July 2008	Dr.rer.nat	University of Göttingen, Germany
	Supervisor:	Prof. Dr. Dieter Hogrefe Prof. Dr. Xiaoming Fu
	Research Topics:	Overlay Multicast Systems Peer-to-Peer Networking Localized Mobility Management Media Streaming Quality-of-Service (QoS)
April 2004	Master of Science	Zhejiang University, China
	Supervisor:	Prof. Dr. Jianrong Tan (Member of the Chinese Academy of Engineering)
	Research Direction:	Mechanical Design and Theory CAD & CG
July 2001	Bachelor Degree in Science	Hangzhou Normal University, China
	Topic:	Mathematics Education

Research Experience

Nov. 2004 - present	Research Assistant	Institute of Computer Science at University of Göttingen, Germany
May. 2003 - Mar. 2004	Research Assistant	DingLi Computer Graphics Company Zhuhai, China
Oct. 2002 - Mar. 2003	Research Assistant	P&T department in UTStarcom Hangzhou, China
Mar. 2001 - Jun. 2003	Research Assistant	CAD & CG lab at Zhejiang University Hangzhou, China

Publication

Journal and Conference

1. J. Lei and X. Fu, “Evaluating the benefits of introducing PMIPv6 for localized mobility management”, in Proc. of the International Wireless Communications and Mobile Computing Conference 2008 (IWCMC’08), Grete, Greece, IEEE, August 2008.

2. N. Neumann, X. Fu and J. Lei, “Inter-domain Handover and Data Forwarding between Proxy Mobile IPv6 Domains”, Internet Draft (work in progress), July 2008.

3. J. Lei, X. Fu and D. Hogrefe, “Dynamic Mesh-based Overlay Multicast Protocol (DMMP)”, draft-lei-samrg-dmmp-03, work in progress, February 22, 2008.

4. J. Lei, L. Shi and X. Fu, ”An Experimental Analysis of Joost Peer-to-Peer VoD Service”, Technical Report No. IFI-TB-2007-03, Institute for Informatics, University of Göttingen, Germany, ISSN 1611-1044, Oct. 2007.

5. D. Le, J. Lei and X. Fu, “A New Decentralized Mobility Management Scheme for IPv6-based Networks”, in Proceedings of the 3rd ACM International Workshop on Wireless Multimedia Networking and Performance Modeling (WMuNeP07), in conjunction with the 10th ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile System (MSWiM07), Chania, Crete Island, Greece, ACM Press, New York, USA, Oct 2007.

6. J. Lei, X. Fu and D. Hogrefe, “Comparative Studies on Authentication and Key Exchange Methods for 802.11 Wireless LAN”, Computer & Security, Elsevier, Vol. 26/5, pp 401-409, August 2007.

7. J. Lei and X. Fu, ”Evaluating the Benefits of Introducing PMIPv6 for Localized Mobility Management”, Technical Report No. IFI-TB-2007-02, Institute for Informatics, University of Göttingen, Germany, ISSN 1611-1044, June 2007.

8. J. Lei, X. Fu and D. Hogrefe, “DMMP: A New Dynamic Mesh-based Overlay Multicast Protocol Framework”, in IEEE Consumer Communication and Networking Conference 2007 – Workshop on Peer-to-Peer Multicasting (P2PM’07), Las Vegas, Nevada, USA, IEEE Communication Society, January 2007.

9. J. Lei, X. Fu and D. Hogrefe, ”DMMP: A New Dynamic Mesh-based Overlay Multicast Protocol Framework”, Technical Report No. IFI-TB-2006-05, Institute for Informatics, University of Göttingen, Germany, ISSN 1611-1044, January 2006.

10. J. Lei, I. Juchem, X. Fu and D. Hogrefe, ”Architectural Thoughts and Requirements Considerations on Video Streaming over the Internet”, Technical Report No. IFI-TB-2005-06, Institute for Informatics, University of Göttingen, Germany, ISSN 1611-1044, November 2005.

11. J. Lei, D. Hogrefe and J. Tan, “Video Image-based Intelligent Architecture for Human Motion Capture”, ICGST International Journal on Graphics, Vision and Image Processing (GVIP), Volume 5, May 2005.

Master Thesis : The technology and implementation of motion surveillance-based video capture and analysis

Abstract: The development of computer science not only improves the whole levels of the science and technology, but also promotes the development of information in the economy and society. Based on the development of the multimedia software of DingLi telecommunication company in Zhuhai, this thesis focuses on exploiting, investigating and implementing the human motion surveillance and analysis system based on video image collection.

Professional and Scientific Activities

* August 12-14, 2008, European MING-T project meeting, Beijing, China.
* August-September, 2008, DAAD-PPP sponsored visting research staff at Columbia University, New York, USA.
* May 28-30, 2008, the 18th International workshop on Network and Operating Systems Support for Digital Audio and Video (Nossdav'08), Brauschweig, Germany.
* October 23-26, 2007, 7th ENABLE project meeting, Ottobrunn, Munich, Germany.
* September 12-14, 2007, L3S Research Workshop, Goslar, Germany.
* July 11-13, 2007, 6th ENABLE project meeting, Waterford, Ireland.
* April 9-11, 2007, 5th ENABLE project meeting, Brunel University, London.
* January 23-26, 2007, 4th ENABLE project meeting, University of Murcia, Spain.
* January 11-13, 2007, IEEE Consumer Communication and Networking Conference 2007 – Workshop on Peer-to-Peer Multicasting (P2PM'07), Las Vegas, Nevada, USA.
* August 11-13, 2006, AIT's Summer School on Next Generation Computing Systems (NGCS), Athens Information Technology, Athens, Greece.

Awards

- August 2006, "Excellent Summer School Student", AIT's Summer School on Next Generation Computing Systems (NGCS).
- July 2002, "The 2st Class Scholarship of Graduate in 2001-2002"
- July 2001, "Excellent Graduate Student"
- July 2000, "Shuping Spiel Scholarship"; "Excellent Student of Graduate in 1999-2000"; "The 1st Class Scholarship of Graduate in 1999-2000"
- July 1999, "Excellent Cadre of Graduate in 1998-1999"; "The 1st Class Scholarship of Graduate in 1998-1999"

- July 1998, ”The 1st Class Scholarship of Graduate in 1997-1998”, ”Excellent Student of Graduate in 1997-1998”