Video Streaming over the Internet using Application Layer Multicast

A thesis submitted in fulfilment of the requirement for the degree of Doctor of Philosophy

Bin Rong
B.E., M.E.

School of Computer Science and Information Technology
Science, Engineering, and Technology Portfolio
RMIT University
Melbourne, Victoria, Australia

March 23, 2008
Declaration

I certify that except where due acknowledgement has been made, the work is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program; any editorial work, paid or unpaid, carried out by a third party is acknowledged; and, ethics procedures and guidelines have been followed.

Bin Rong
March 23, 2008
Acknowledgments

The pursuit of a PhD certainly is the most wonderful experience in my life, and it means a great deal of commitment and hard work. Luckily many people offer their invaluable help along the way. My deepest gratitude goes to my two supervisors: Dr. Ibrahim Khalil and Professor Zahir Tari, for their encouragement and support throughout my PhD.

My gratitude to Dr. Fred Douglis, Dr. Zhen Liu, and Dr. Cathy Xia for their guidance and help during my internship at IBM T. J. Watson research center.

I want thank all members of the discipline for so many memorable moments I shared with them. They are: Sandy Citro, Saravanan Dayalan, Islam Elgedawy, Vidura Gamini Abhaya, Nalaka Gooneratne, Malith Jayasinghe, Sakib Kazi Muheymin, Craig Pearce, Mikhail Pereplechikov, Damien Phillips, Hendrik Gani, Alice Wang, Anh Phan, Kwong Lai, Peter Dimopoulos, James Broberg, and Abhinav Vora.

Many people have lent their help during my PhD, and I want to thank all of them, they are Professor Panlop Zeephongsekul, Peter O’Neill and Danne O’Neill, Geoff Warburton, and Don Gingrich.

Finally, I am deeply indebted to my wife Yunyan Ai, my parents Ying’e Yang and Hongfa Rong, for all the sacrifices they have made along the way. Without them, this thesis would have never come into existence.
Credits

Portions of the material in this thesis have previously appeared in the following publications:


- Bin Rong, Ibrahim Khalil, and Zahir Tari, *Reliability Enhanced Large-Scale Application Layer Multicast*, 49th annual IEEE Global Telecommunications Conference (GLOBECOM), San Francisco (USA), November 2006

- Bin Rong, Ibrahim Khalil, and Zahir Tari, *Making Application Layer Multicast Reliable is Feasible*, The 31st Annual IEEE Conference on Local Computer Networks (LCN), Florida (USA), November 2006

- Bin Rong, Ibrahim Khalil, and Zahir Tari, *An Adaptive Membership Management Algorithm for Application Layer Multicast*, International Conference on Networking and Services (ICNS), Silicon Valley (USA), July 2006


This work was supported by an Australian Postgraduate Award.

Note

Unless otherwise stated, all fractional results have been rounded to the displayed number of decimal figures.
Contents

Abstract

1 Introduction
  1.1 Research Questions .................................................. 5
  1.2 Research Contributions .............................................. 6
  1.3 Thesis Structure ..................................................... 7

2 Background
  2.1 Group Communication Model ......................................... 9
  2.2 Multicast Models ..................................................... 9
    2.2.1 Client-server Model ............................................ 10
    2.2.2 Network Layer Multicast ..................................... 10
    2.2.3 Overlay Multicast ............................................... 15
    2.2.4 Content Distribution Networks ................................. 15
  2.3 The Relationship with Peer-to-Peer Technologies ................. 18
  2.4 Survey of Overlay Multicast Protocols .......................... 18
    2.4.1 Mesh-based Protocols ........................................ 19
    2.4.2 Tree-based Protocols ......................................... 23
    2.4.3 Data-driven Protocols ....................................... 27

3 Adaptive Gossip-based Membership Management Algorithm .......... 31
  3.1 Motivation .......................................................... 32
  3.2 Related Work ....................................................... 34
    3.2.1 Group Membership Management ............................... 34
    3.2.2 A Scalable Protocol with a Non-scalable Membership Management Algorithm ........................................... 34
### Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2.3</td>
<td>Gossip-based Algorithms</td>
<td>35</td>
</tr>
<tr>
<td>3.3</td>
<td>Proposed Approach</td>
<td>36</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Terminologies and Metrics</td>
<td>37</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Detailed Algorithm Description</td>
<td>40</td>
</tr>
<tr>
<td>3.4</td>
<td>Analytical Results</td>
<td>43</td>
</tr>
<tr>
<td>3.5</td>
<td>Experimental Results</td>
<td>46</td>
</tr>
<tr>
<td>3.5.1</td>
<td>Simulation Setup</td>
<td>46</td>
</tr>
<tr>
<td>3.5.2</td>
<td>Metrics of Interest</td>
<td>47</td>
</tr>
<tr>
<td>3.5.3</td>
<td>Simulation Results</td>
<td>48</td>
</tr>
<tr>
<td>3.6</td>
<td>Conclusion</td>
<td>54</td>
</tr>
<tr>
<td>4</td>
<td>Resilient Application Layer Multicast</td>
<td>55</td>
</tr>
<tr>
<td>4.1</td>
<td>Motivation</td>
<td>56</td>
</tr>
<tr>
<td>4.2</td>
<td>Related Work</td>
<td>56</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Overlay Multicast Tree Construction</td>
<td>57</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Resilient Overlay Multicast</td>
<td>58</td>
</tr>
<tr>
<td>4.3</td>
<td>Proposed Approach</td>
<td>60</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Rationale Underlying the Proposed Approach</td>
<td>60</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Detailed Algorithm Description</td>
<td>66</td>
</tr>
<tr>
<td>4.4</td>
<td>Analytical Results</td>
<td>67</td>
</tr>
<tr>
<td>4.5</td>
<td>Experimental results</td>
<td>71</td>
</tr>
<tr>
<td>4.5.1</td>
<td>Simulation Setup</td>
<td>72</td>
</tr>
<tr>
<td>4.5.2</td>
<td>Metrics of Interest</td>
<td>72</td>
</tr>
<tr>
<td>4.5.3</td>
<td>Simulation Results</td>
<td>73</td>
</tr>
<tr>
<td>4.6</td>
<td>Conclusion</td>
<td>78</td>
</tr>
<tr>
<td>5</td>
<td>QoS-aware Reliable Application Layer Multicast</td>
<td>82</td>
</tr>
<tr>
<td>5.1</td>
<td>Motivation and Problem Formulation</td>
<td>83</td>
</tr>
<tr>
<td>5.1.1</td>
<td>Network Model</td>
<td>84</td>
</tr>
<tr>
<td>5.1.2</td>
<td>Design Space</td>
<td>86</td>
</tr>
<tr>
<td>5.1.3</td>
<td>Hardness of The Problem</td>
<td>88</td>
</tr>
<tr>
<td>5.2</td>
<td>Related Work</td>
<td>89</td>
</tr>
<tr>
<td>5.3</td>
<td>A QoS-aware Scheme</td>
<td>91</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Building Blocks</td>
<td>92</td>
</tr>
</tbody>
</table>
### CONTENTS

5.3.2 A New Parent Selection Procedure ........................................... 92
5.4 Experimental Results ............................................................... 93
5.4.1 Simulation Setup ................................................................. 94
5.4.2 Metrics of Interest ............................................................... 94
5.4.3 Simulation Results ............................................................... 95
5.5 Conclusion ................................................................................. 98

6 Admission Control for Application Layer Multicast .......................... 102
6.1 Motivation ................................................................................. 103
6.2 Related Work ........................................................................... 104
6.2.1 Deterministic Algorithms ...................................................... 105
6.2.2 Statistical Algorithms .......................................................... 105
6.3 Mathematical Problem Formulation ............................................. 108
6.4 The Proposed Admission Control Protocol .................................. 112
6.4.1 Membership Information Management .................................... 113
6.4.2 A Distributed Admission Control Algorithm ......................... 113
6.5 Analytical Results ..................................................................... 115
6.6 Experimental Results ............................................................... 117
6.6.1 Simulation Setup ................................................................. 118
6.6.2 Metrics of Interest ............................................................... 118
6.6.3 Simulation Results ............................................................... 119
6.7 Conclusion ................................................................................. 128

7 Conclusion .................................................................................... 129
7.1 Membership Management ......................................................... 129
7.2 Reliability Enhancement ............................................................. 130
7.3 QoS-aware Tree Construction ..................................................... 131
7.4 Admission Control ................................................................. 132
7.5 Future Work ............................................................................. 132

Bibliography .................................................................................. 134
List of Figures

2.1 An example of client-server model. ........................................ 10
2.2 An example of Network layer multicast. ................................. 11
2.3 An example of application layer multicast. ............................. 15
2.4 CDNs and application layer multicast. .................................. 17
2.5 An example of Scribe. ....................................................... 22
2.6 The hierarchy of NICE. ..................................................... 24
2.7 The joining process of NICE. .............................................. 25
2.8 The joining process of Overcast. ........................................... 26
2.9 Stream decomposition of Coolstreaming [Xie et al., 2007]. ............. 29
3.1 A simple example of membership view. .................................. 37
3.2 A simple join and gossip. ................................................... 42
3.3 Average hop count and delay of data packets of the proposed algorithm. . 48
3.4 Link stress of data packets of the proposed algorithm. .................. 49
3.5 Link stress of gossip overhead of the proposed algorithm. ............... 50
3.6 SCAMP: Average hop count and delay of data packets. .................. 50
3.7 SCAMP: Link stress of data packets. ..................................... 51
3.8 SCAMP: Link stress of gossip overhead. ................................ 52
3.9 Gossip overhead analysis. ................................................ 53
3.10 Reliability analysis. ....................................................... 53
4.1 Reliability is the problem. .................................................. 56
4.2 Logical hierarchical structure. ............................................. 62
4.3 Make before break switch. ................................................ 64
4.4 Handling newly joined peers. .............................................. 65
4.5 Quality of Service comparison. ........................................... 74
LIST OF FIGURES

4.6 Service disruption comparison. ........................................ 75
4.7 Latency problem of the proposed algorithm. ...................... 76
4.8 Latency of data packets comparison. ................................. 77
4.9 Forwarding comparison. ............................................... 77
4.10 Average bandwidth 30. .............................................. 79
4.11 Average bandwidth 25. .............................................. 79
4.12 Average bandwidth 20. .............................................. 80
4.13 Average bandwidth 15. .............................................. 80

5.1 An example of single-tree-based peer-to-peer media streaming. . 85
5.2 Cumulative QoS comparison. .......................................... 96
5.3 QoS under different lifetime distributions. .......................... 97
5.4 QoS under different delay constraints. ............................... 99
5.5 Rejoin frequency comparison. ...................................... 100

6.1 A simple example illustrating the importance of an admission control algorithm. 103
6.2 A simple example illustrating State-Dependant Markov Decision Process (SD-MDP). ........................................ 109
6.3 An example of single-tree-based peer-to-peer media streaming. .... 116
6.4 How capacity changes with gamma and number of generations. .... 117
6.5 QoS comparison. ...................................................... 120
6.6 QoS under different delay constraints. ................................. 122
6.7 QoS under different lifetime distributions. ........................... 124
6.8 Rejection rate comparison. .......................................... 125
6.9 Rejoin frequency comparison. ...................................... 126
List of Tables

5.1 Notations. .............................................. 83
Abstract

Multicast is a very important communication paradigm, and many applications are built upon multicast, such as Video-on-Demand (VoD), large volume content distribution, teleconference, and many other group communication applications. However, the deployment of multicast at IP layer is very slow, due to development and deployment issues such as ISPs’ lack of incentives to update routers and inter-operability among multicast routing protocols.

Application Layer Multicast (ALM) seems to be a good alternative [Chu et al., 2002], where participating peers organize themselves into a logical overlay network atop the physical links and data is “tunneled” to each other via unicast links. The distinctive feature between IP multicast and ALM is that in ALM, data replication and forwarding functionalities are performed by participating peers (a.k.a. end systems), rather than the routers in Internet Protocol (IP) multicast. This fundamental difference enables ALM to be able to circumvent the development and deployment issues of IP multicast, by exploiting the resources (e.g., CPU cycles, storage, and access bandwidth) at the edge of the network. Nevertheless, it also raises other challenges, as peers are not as stable as routers since they may join and depart the on-going session at will. In this thesis, we address some of the challenges and they are summarized as follows:

- First, most current P2P or ALM streaming systems are equipped with a non-scalable membership management algorithm, greatly hindering their applicability to large-scale implementations over the Internet [Chu et al., 2002; Francis, 1999; Zhang et al., 2002; Pendarakis et al., 2001]: they either rely on a central entity to handle group membership, or simply assume that all group members are visible to each other and flooding is the main mechanism used to disseminate membership-related updates to all participating group members. This implies that they are only applicable to small groups.

- Second, one of ALM’s prominent features, flexibility, has not been fully exploited: moving the multicast functionalities from lower layer (IP layer) to higher layer (Application
layer) can greatly facilitate the integration of Quality-of-Service (QoS) support. The end-to-end philosophy states that it is better to leave those functionalities to higher layers because the heterogeneity among users’ requirements can be handled much better by end users, rather than the network. However, QoS, and in particular, reliability has not been thoroughly addressed in existing ALM schemes.

- Third, good admission control algorithms are essential to the success of any ALM system, due to the fact that in ALM, each peer acts as both a client as well as a server. On the other hand, the heterogeneity among peers, in terms of their computational power, storage capacity, and access bandwidth, further complicates the design of a good admission control.

Several contributions are made to address the aforementioned research challenges, and they are outlined as follows:

- The first contribution is a devised gossip-based membership management algorithm that is able to collect and disseminate membership-related information under high rate of churn, using relatively low communication overheads.

- The second contribution is a reliability-centric multicast tree construction algorithm that greatly enhance peers’ perceived reliability.

- The third contribution is a QoS-aware tree construction algorithm that accommodates the heterogeneity among peers, such as access bandwidth, network distance, and reliability.

- The last contribution is the identification of the admission control problem in this overlay video streaming context.
Chapter 1

Introduction

The Internet has become the primary communication platform, and many applications are built upon it, such as video-on-demand (VoD), live broadcasting, teleconferencing, and large-volume content dissemination. There is a growing need for support of multicast functionality because of the emergence of these group applications.

Multicast is an extension of the original Internet Protocol (IP), that was proposed to overcome the shortcomings of IP protocol, providing efficient multipoint delivery [Deering and Cheriton, 1990]. However, the efforts to support multicast at IP layer have proved to be slow and painful, due to factors such as ISPs’ lack of incentives, limited address space, difficulty to support reliable transmission and congestion control.

Recently, real-time video streaming has become a reality from a dream with the pervasiveness of high-speed broadband networking technologies and powerful Personal Computers (PCs). The emergence of Peer-to-Peer (P2P) and Application Layer Multicast (ALM) technologies make it increasingly possible to deliver video and audio streaming over the Internet.

P2P-based and ALM-based streaming have gained enormous popularity in recent years due to their ability to bypass the development and deployment problems associated with traditional network layer multicast. In both schemes, participating peers store the streaming data and subsequently become supplying peers by streaming to other requesting peers. This fundamental difference makes overlay multicast very appealing as an alternative to traditional IP multicast:

- Timely deployment: No modification nor administration work needs to be performed since the multicast functionality has been shifted to application layer and handled by
participating users, i.e., there is no extra cost incurred for ISPs. Therefore, an overlay network can be easily built and maintained. Consequently, as a common communication platform, many group applications requiring multicast support can be built upon the overlay network. Taking PlanetLab $^1$ as an example, which is a global testbed for new Internet-based applications and currently reaches out to 440 nodes worldwide. New applications can be quickly deployed and validated without modification of the existing Internet architecture.

- **Resource exploitation:** Overlay networks exploit the resources at the edge of the networks, e.g., computational power (CPU cycles), storage, and communication (access bandwidth). Akamai $^2$ and Skype $^3$ are good examples illustrating how to exploit resources at the edge of the network.

- **Flexibility:** Flexibility is a big advantage of overlay networks since various functionalities can be implemented at the application layer, such as Quality-of-Service (QoS) and various network management activities.

Nevertheless, several challenging issues need to be addressed before large-scale implementations.

- **First,** most current P2P or ALM streaming systems are equipped with a non-scalable membership management algorithm, greatly hindering their applicability to large-scale implementations over the Internet [Chu et al., 2002; Francis, 1999; Zhang et al., 2002; Pendarakis et al., 2001]: they either rely on a central entity to handle group membership, or simply assume that all group members are visible to each other and flooding is the main mechanism used to disseminate membership-related updates to all participating group members; this implies they are only applicable to small groups.

- **Second,** one of ALM’s prominent features, flexibility, has not been fully exploited: moving the multicast functionalities from lower layer (IP layer) to higher layer (Application layer) can greatly facilitate the integration of Quality-of-Service (QoS) support. The end-to-end philosophy states that it is better to leave those functionalities to higher layers because the heterogeneity among users’ requirements can be handled much better.

---

$^1$www.planet-lab.org

$^2$www.akamai.com

$^3$www.skype.com
CHAPTER 1. INTRODUCTION

by end users, rather than the network. However, QoS, in particular reliability, has not been thoroughly addressed in existing ALM schemes.

- Third, good admission control algorithms are essential to the success of any ALM system, due to the fact that in ALM, each peer acts as both client as well as server. On the other hand, the heterogeneity among peers, in terms of their computational power, storage capacity, and access bandwidth, further complicates the design of a good admission control.

1.1 Research Questions

Since the early work of YOID [Francis, 1999], a large number of papers have been published on ALM, e.g., Narada [Chu et al., 2002], Host Multicast [Zhang et al., 2002], ALMI [Pendarakis et al., 2001], and so on. However, the assumptions they rely on or the way in which the overlay networks are constructed are not applicable to large-scale implementation over the Internet.

This thesis investigates the feasibility of implementing large-scale video streaming using overlay networks, and various algorithms are devised or proposed to make our scheme work even under the conditions of high rate of churn, heterogeneity among peers, limited bandwidth, and lack of infrastructure support. In particular, the following research questions are raised:

1. Is there a scalable membership management scheme and is there a cost-effective way to do this? Various techniques have been proposed for group membership management purposes [Deering et al., 1994; Ballardie et al., 1993; Haberman and Martin, 2001; Deering et al., 1994; Chu et al., 2002]. However, these proposed techniques are either not applicable to overlay networks or not scalable.

2. Is reliability an inherent problem of overlay streaming? Many overlay multicasting schemes have been proposed [Chu et al., 2002; Francis, 1999; Zhang et al., 2002; Pendarakis et al., 2001], but most of them are concerned with setting up the proper multicast structure on top of the overlay networks and they failed to explicitly take reliability into consideration. Due to ALM’s serverless nature, reliability has a huge impact on users’ perceived Quality-of-Service (QoS). Therefore, we must find a suitable way to deal with it.
CHAPTER 1. INTRODUCTION

3. Can heterogeneity among peers be handled and accommodated in a graceful way? Can existing routing protocols be modified to accommodate this heterogeneity?

4. Is there an effective admission control algorithm for overlay streaming? Can we adapt the existing admission control algorithms, e.g., those admission control algorithms proposed for ATM networks, to the unique overlay streaming environment, i.e., a highly dynamic environment.

1.2 Research Contributions

Bearing the aforementioned questions in mind, we conducted our investigations over the feasibility of large-scale video streaming over the Internet. A number of contributions have been made in answering those research questions raised in the previous section, and these contributions are summarized below:

Membership Management

The first contribution is a devised gossip-based membership management algorithm that is able to collect and disseminate membership-related information under high rate of churn, using relatively low communication overhead. In the proposed algorithm, the parameter settings of the gossip algorithm are fine-tuned by dynamic weight setting throughout the session, in terms of the length of the gossip round and the scope of the gossip targets selection. The tuning process is done in such a way that it reflects the changes and the characteristics of the network, and this makes it possible to significantly reduce the communication and computational overhead. Experimental results show that a maximum of 50% reduction can be achieved in terms of network overhead on core network components, such as backbone links and attached routers, without sacrificing reliability.

Reliability Enhancement

The second contribution is a reliability-centric multicast tree construction algorithm that greatly enhance peers’ perceived reliability. The proposed algorithm first organizes participating peers into a hierarchy in such a way that it reflects their relative stabilities (represented by their “rank”), rather than their geographical proximities or other criteria. Then a multicast delivery tree is constructed out of the hierarchy. In addition, peers periodically update their ranks and make attempts to be connected to more stable peers. In this way, peers
that are potentially more stable, eventually “climb” up and are placed close to the streaming source, and most dynamics caused by ungraceful departure of peers are confined within the lower end of the multicast tree. A minimum reduction of 50% can be achieved in terms of service disruption frequency for most peers, and consequently, peers’ perceived QoS are greatly improved.

**QoS-aware Tree Construction**

The third contribution is a QoS-aware tree construction algorithm that is able to accommodate the heterogeneity among peers, such as access bandwidth, network distance, and reliability. It is built upon our work on reliability enhancement, i.e., peers are organized into a hierarchy according to their potential reliability. The difference lies in a new parent selection algorithm, which is derived from Dijkstra’s shortest path algorithm, taking peers’ access bandwidth, network distance and other realistic QoS parameters into consideration. Extensive simulation reveals that the proposed approach can actually accommodate the inherent heterogeneity, and most of the participating peers are able to receive satisfactory service.

**Admission Control**

The last contribution is the identification of the admission control problem in this overlay video streaming context. It is found that there exists a large performance gap needing to be filled, and this is attributed to the fact that peers are admitted into the system in order of arrival, rather than from a performance perspective. The identified problem is formulated as a stochastic knapsack problem, and an heuristic-based algorithm is proposed to approximate the solution to this stochastic knapsack problem. The proposed admission control algorithm is validated through simulations and is able to reduce the rejection rate by as much as 50%.

**1.3 Thesis Structure**

The rest of the thesis is organized as follows:

- Chapter 2 presents a survey of the related work. Various techniques related to multicast and application layer multicast are presented, putting our work into perspective.

- Chapter 3 focuses on how to maintain the group structure in a highly dynamic environment, i.e., a cost-efficientive membership management algorithm. A new gossip-based
membership management algorithm is presented, together with a detailed mathematical analysis and simulation results.

- Chapter 4 addresses the reliability problem, and the use of peers’ potential reliability (represented by their “rank”) is investigated. A novel reliability-centric tree construction algorithm is proposed in this chapter together with evaluation results.

- Chapter 5 extends the work presented in Chapter 4, taking into account other realistic and important parameters, such as access bandwidth, and network distance. The outcome is a QoS-aware tree construction algorithm. Detailed analysis and validation results are also included in this chapter.

- Chapter 6 investigates the admission control problem in overlay streaming. The problem under consideration is identified and formulated as a stochastic knapsack problem, and a heuristic-based algorithm is presented with satisfactory results, validated and proven using extensive simulations.

- Finally, the whole thesis is concluded in Chapter 7, in which the contributions of this thesis are summarized and future research is discussed.
Chapter 2

Background

Background materials are presented in this chapter, putting our work into perspective. First, the ground communication model is defined. Followed by a review of the state-of-the-art video streaming technologies, including Client-Server model, IP multicast, Application Layer Multicast (or more general Overlay multicast), and Content Distribution Network (CDN).

2.1 Group Communication Model

The focus of this thesis is multicast, and to be more specific, Application Layer Multicast; so the group communication model here is multicast, i.e., there is one sender and many receivers and the detailed model is defined as follows:

A network \((V, L)\), where \(V = \{v_1, v_2, ...v_n\}\) represents the set of nodes. \(L\) is the corresponding link set, where \(l = (v_x, v_y) \in L\) represents the physical link from node \(v_x\) to \(v_y\). It is further assumed that each physical link to be directed, which is the case for most real networks. Nevertheless, all the algorithms presented in this thesis are also applicable to the undirected network model.

2.2 Multicast Models

Various multicast models exist under this common group communication model, and they are broadly classified into four catalogs as described in this section.
2.2.1 Client-server Model

Making use of the client-server architecture is probably the simplest and most straightforward way of realizing multicast over the Internet. Figure 2.1 gives a very simple example of this client-server model, where 1 is the data stream source, and 2, 3, and 4 are prospective receivers, and A, B, and C are routers. As can be seen from the figure, users 2, 3, and 4 are treated as independent users although they are retrieving the same content, and consequently individual connections are setup between the users and the data source. The pitfall of this architecture is clear: the upload bandwidth of the data source has become the bottleneck. This drawback greatly limits its applicability, only to very small groups, and clearly it is not desirable.

2.2.2 Network Layer Multicast

In order to overcome the shortcomings of the aforementioned client-server mode, multicast was proposed, as an extension of the original Internet Protocol (IP), to provide efficient multipoint delivery [Deering and Cheriton, 1990]. It works by sending one and only one copy of each packet along the so-called “multicast tree”, achieving the efficient usage of network resource. Figure 2.2 gives a very simple example of network layer multicast (a.k.a.
CHAPTER 2. BACKGROUND

Figure 2.2: An example of Network layer multicast.

IP multicast), where 1 is the data stream source, and 2, 3, and 4 are receivers. A and B stand for two routers. As can be seen from the figure, only one copy of the data packets are sent from the stream source to router B although two receivers, 2 and 4, are attached to router B. The underlying mechanism is that router B is aware of the existence of receivers 2 and 4, and it automatically replicates the incoming packets and forwards them to receivers 2 and 4 respectively. This multicast model is termed as “network layer multicast (IP multicast)” since all the multicast related activities (e.g., membership management, data replication and forwarding, etc.) are taken care of by routers that operate at IP layer.

Various techniques utilizing network layer multicast can be categorized into three approaches: the reactive transmission approach, the proactive transmission approach, and the hybrid approach. In all three approaches, the unit server bandwidth required to serve one video stream is termed as a channel, and the number of these channels is limited by the server bandwidth. These three approaches differ in how to utilize these channels.

Reactive Server Transmission Approach

In reactive transmission approach, the server dedicates several channels to serve several requests for the same video arriving closely in time. To further conserve the server bandwidth, two approaches, static multicast and dynamic multicast, have been proposed.
• **Static Multicast Approach**

In the static multicast approach, only one channel is used to serve a batch of requests for the same video arriving closely in time. This approach is also referred to as batching, and all users belonging to the same batch are served using the same multicast tree. That is to say, once a batch of users join the streaming session, a static multicast tree is formed to serve all these users, and the multicast tree remains unchanged throughout the streaming session. The difference between various schemes lies in the policy to select which batch to serve first when a server channel becomes available.

In first-come-first-serve (FCFS), the batch with the longest waiting time is served when server channel is available. The FCFS approach offers fairness by treating each user equally regardless of the popularity of the requested video, however, it yields low system throughput because the batch with fewer user requests may block the batch with more user requests. To address this limitation, in maximum-queue-length-first (MQLF) [Dan et al., 1996], a separate waiting queue is maintained for each video, and the batch with the longest queue is served next. The system throughput is gained at the price of sacrificing fairness since the users in the batch with fewer request may have to wait for a long time before they are served. Maximum-factored-queued-length [Aggarwal et al., 1996b] tried to strive a balance between fairness and system throughput. It extends the MQLF scheme by choosing the batch with longest queue weighted by a factor $\frac{1}{\sqrt{f_i}}$, where $f_i$ is the popularity of the requested video $v_i$. The factor $f_i$ prevents the server from always favoring the more popular videos.

• **Dynamic Multicast Approach**

The dynamic multicast approach extends the static multicast approach to include the newly arriving users, i.e., the multicast tree can be dynamically extended to accommodate newly joined users. In other words, in dynamic multicast approach, the multicast tree grows with the addition of new users. In adaptive piggybacking [Golubchik et al., 1996], the server gradually slows down the delivery rate to a previous user, while speeds up the delivery rate to a new user until they share the same play point in the video. By merging two video streams, the server is able to use only one channel to serve two users at the same time.

Patching [Cai et al., 1999; Carter and Long, 1999; Eager et al., 1999] enables the newcomers to join an on-going session and receive the entire video stream. The newcomers
download and cache the later portion of the video, while the server delivers the missing portion of the requested video stream to the newcomers in a separate patching stream.

**Proactive Server Transmission Approach**

In the proactive transmission approach, users do not make any requests to the server. Instead, the server periodically broadcasts a video clip, e.g., a new stream of the same video is broadcasted every $t$ seconds. This approach can serve a large number of users with minimal server bandwidth while guaranteeing a bounded service delay.

In proactive transmission approaches [Dan et al., 1994; Aggarwal et al., 1996a; Juhn and Tseng, 1997; Hua and Sheu, 1997; Hua et al., 1998; Hu, 2001; Mahanti et al., 2001; Gao et al., 2002], a video is broken into several segments. Each segment is periodically broadcasted on a dedicated channel. It is highly scalable, due to its capability of serving a large number of users with minimal server bandwidth. Existing proactive transmission schemes can be classified into two categories: server-oriented and client-oriented. Server-oriented approaches reduce service delay by increasing server bandwidth, i.e., either broadcast the video at a high data rate to allow the clients to be able to prefetch data into a local buffer, or repeatedly broadcast the video within a short interval. On the contrary, client-oriented approaches achieve the same goal by requiring more client bandwidth, i.e., clients try to concurrently download from several channels so as to minimize service delay.

- **Server-oriented Category**

  Staggered broadcasting [Dan et al., 1994] is the earliest video broadcasting technique. This approach staggers the broadcast starting time evenly across available channels. The starting time difference is referred to as phase offset. Since a new stream of a particular video clip is broadcasted every phase offset, it is the longest service delay. Permutation-based broadcasting [Aggarwal et al., 1996a] divides each channel into $s$ sub-channels that broadcast a replica of the video fragment with a uniform phase delay. This technique reduces the bandwidth at the client side by a factor of $s$. Hua and Sheu [1997] proposed skyscraper broadcasting, where the server bandwidth is divided into several logical channels of bandwidth equal to the playback rate of the video. Each video is further fragmented into several segments, and the sizes of the segments are determined using the broadcast series [1, 2, 5, 12, 25, 52, ...]. Assume the size of the first segment is $x$, this scheme limits the size of the biggest segments ($W$ segments) to $W$. These segments are stacked up to resemble a skyscraper of a width $W$.  

(March 23, 2008)
• **Client-oriented Category**

Harmonic broadcasting [Juhn and Tseng, 1997] initiates the techniques in this category. It fragments a video into segments of equal sizes and periodically broadcasts each segment on a dedicated channel. The channel have decreasing bandwidths following the harmonic series. Clients download segments from all channels concurrently. However, this client-oriented approach has many drawbacks compared with the server-oriented approach. First, the client must a network bandwidth equal to the server bandwidth allocated to the longest video. Second, in order to reduce service delay, it requires adding bandwidth to both server and client.

**Hybrid Server Transmission Approach**

The proactive approaches involve periodic broadcast that is suitable for popular videos. A hybrid approach that combines both on-demand multicast and periodic broadcast may offer better performance. Hua et al. [2002] proposed the adaptive hybrid approach. It periodically measures the popularity of each video based on the distribution of recent service requests, and popular videos are periodically broadcasted using skyscraper broadcasting [Hua and Sheu, 1997].

However, all network multicast based approaches have many drawbacks, especially in two aspects:

- **Development and deployment issue**: Since routers play a crucial part in IP multicast, so the prerequisite of a widely deployed IP multicast is that all routers can support the multicast functionalities, and to be more specific, data replication and forwarding. Unfortunately, not all the existing routers support these functionalities. Furthermore, the inter-operability of routers from different vendors further delays the deployment of IP multicast.

- **Lack of Quality-of-Service (QoS) support**: The phenomenal success of the Internet is largely attributed to the original design philosophy of a dummy IP layer, i.e., it only deals with packets routing. Unfortunately, the lack of QoS support is due to the same reason, and many higher layer functionality (e.g., error, flow and congestion control, reliability) are not supported [Wu et al., 2001].
2.2.3 Overlay Multicast

To address the above mentioned problems of IP multicast, several researchers [Chu et al., 2002] raised the idea of moving up the protocol stack from the network layer to the application layer, clearing the barriers of establishing multicast structure at network layer. In application-layer multicast (ALM), data packets are replicated at end hosts rather than being replicated at routers inside the IP network, and the end hosts form a logical layer atop IP layer. It is interesting to see from Figure 2.3 that receiver 4 is now getting the stream from receiver 3, i.e., receiver 3 now take care of the data replication and forwarding functionalities.

Various overlay multicast schemes are elaborated in detail in the following section due to their close resemblances to our work in many aspects, e.g., network model, and protocol stack.

2.2.4 Content Distribution Networks

IP multicast and overlay multicast represent two extremes of the multicast design spectrum: on one end, multicast is implemented at network layer and is transparent to end users; while in overlay multicast that is on the other end of the spectrum. End users take over the multicast-related functionalities while the network nodes simply relay the packets. Content
CHAPTER 2. BACKGROUND

Distribution Networks (CDNs) try to strive for a balance between these two extremes by deploying a set of geographically distributed gateways over the Internet, e.g., Akamai \(^1\). As can be seen from Figure 2.4(a), end users are served by the nearby gateways that are statically deployed beforehand and have a replication of the content through caching. These gateways \textit{per se} form an overlay network.

\(^1\)www.akamai.com
CHAPTER 2. BACKGROUND

(a) A content delivery network (CDN).

(b) An application layer multicast network.

Figure 2.4: CDNs and application layer multicast.
CHAPTER 2. BACKGROUND

This solution can provide worldwide streaming services. However, its deployment and maintenance costs are too expensive for small content providers.

On the other hand, application layer multicast or overlay multicast does not need any infrastructure-wise support since content is replicated and further disseminated by end users. Figure 2.4(b) clearly shows the ability of overlay multicast to make use of the resources at the edge of the networks. Due to this ability, overlay multicast is applicable to small content providers and supports fast deployment of streaming applications, e.g., video conference.

2.3 The Relationship with Peer-to-Peer Technologies

Peer-to-Peer (P2P) technology has emerged as a very important platform for a wide range of applications, ranging from file sharing (such as Emule 2, Gnutella 3, and Bittorrent 4) to Voice-over-IP (VoIP) (e.g., Skype 5). Its huge success gives an impression that it is quite straightforward to extend P2P technology to video delivery domain. However, the unique and stringent requirements of bandwidth and delay for video streaming raise different challenges to P2P based technologies. These requirement are tight and they must not be violated under any circumstance. On the contrary, delay is never an issue in most file sharing applications, e.g., Emule, Gnutella, and Bittorrent, and it is quite common to spending several hours or even several days to downloading a large file. This clearly is not affordable in the video streaming context.

On the other hand, VoIP, such as Skype, does have the similar real-time requirement. Nevertheless, the high bandwidth consumption characteristics, together with its highly dynamic nature, raise new challenges for the existing P2P technologies. These challenges are identified and demonstrated by reviewing the state-of-the-art overlay multicast schemes in the following.

2.4 Survey of Overlay Multicast Protocols

Because of overlay multicast’s simplicity and its timely deployment characteristics, this thesis will focuses on overlay multicast, and various the sate-of-the-art overlay multicast protocols are surveyed to put our work into perspective.

---

2 www.emule-project.net
3 www.gnutella.com
4 www.bittorrent.com
5 www.skype.com

(March 23, 2008)
The large body of work on application layer multicast generally fall into three approaches: mesh-based, tree-based, and data-driven. Each of them is explained in greater detail in the following.

2.4.1 Mesh-based Protocols

Mesh-based protocols first build a mesh-like topology out of the participating users by modeling users as vertexes and the links between them as edges, and there might be multiple paths connecting a pair of users. Then single or multiple multicast delivery trees are built out of the mesh. It is termed as mesh-based approach because the multicast tree is implicitly embedded in the mesh and the quality of the mesh has a huge impact on the quality of the resulting multicast tree.

There are a lot of tradeoffs that need to be considered in mesh-based approach. For example, the density of the mesh, computational complexity, and the quality of the final multicast tree. On one hand, a denser mesh means there are more alternative paths between users, and this may lead to a multicast tree with low latency. However, more alternative paths also means a larger solution space, and this may lead to a more computationally extensive multicast tree construction scheme. On the other hand, fewer alternative paths means a simpler multicast tree construction would suffice, but at the cost of a longer delay in the resulting tree.

A large body of work on mesh-based approach has been published, and they all focus on different optimization aspects, e.g., delay, link stress, algorithmic complexity, and so on. In order to demonstrate the basic mechanisms underlying the mesh-based approach, two representative protocols are chosen to present here, and they are Narada [Chu et al., 2002] and Scribe [Castro et al., 2002].

**Narada**

Narada [Chu et al., 2002] is the first application layer multicast protocol, and it clearly demonstrated the feasibility of moving multicast functionalities to higher layers. It is targeted at Internet conference applications, where participants can act as both data sources and receivers at the same time.

Each node in Narada maintains a membership list containing the information about a random subset of members, as well as information about the path from the source to itself. A newcomer joins the session by contacting the source, and it is provided with a partial list
of the members that are currently in the session. It then selects one of these members in the partial list using the parent selection algorithm. The membership-related information is maintained through periodical exchange of refresh messages among participating nodes. In this way, the changes in membership due to nodes’ join and departure are eventually propagated to all participants. The actual multicast tree is constructed using the reverse path algorithm [Dalal and Metcalfe, 1978] which works in the following way: a peer, say peer \( i \), upon receipt of the multicast packets from the source \( s \), it forwards the received packets to all the peers that are on the shortest path from \( i \) to \( s \).

Narada constantly makes the effort to improve the quality of the mesh. Each node periodically probes some subset of the nodes it knows to evaluate the overall delay if connected through the probed nodes. If the reduction, in terms of overall delay, is beyond a pre-defined threshold, it drops the current link and chooses to be connected through the newly probed node.

In the meantime, each node calculates the consensus cost of the edges between itself and its neighbors. For all the shortest paths from a node, say node \( u \), to other participating nodes, \( u \) counts the number of them, including link \( l_{uv} \). While node \( v \) does exactly the same. The maximum of these two numbers is the consensus cost, and if it is below a pre-defined threshold, link \( l_{uv} \) is disconnected.

The pre-defined adding and dropping thresholds are nothing but some functions of the maximal and minimal fanout of the participating nodes. In other words, Narada controls the maximal and minimal fanout of all nodes to prevent nodes from becoming bottlenecks because of too many connections.

The partition of the mesh can be detected with the aid of the pre-mentioned periodical message exchange. A node, say node \( u \), suspects its neighbor, node \( v \), is down because it misses several refresh messages from \( v \), and node \( u \) probes node \( v \) immediately to find out the actual state of node \( v \). Once confirmed, node \( u \) will take the appropriate action.

Being the first overlay multicast application, Narada clearly demonstrated the feasibility of moving multicast functionality to higher layers. However, it is not scalable and only applicable to very small groups due to several reasons. First, changes of membership are disseminated to all participating peers and incurs a overhead of \( O(N^2) \). Second, the employment of the reverse path algorithm [Dalal and Metcalfe, 1978] requires each peer to maintain a routing table of size \( O(N) \), i.e., the routing table contains entries corresponding to all the other participating peers. Therefore, the communication and computational overhead greatly hinder its scalability and applicability.
Scribe

Scribe [Castro et al., 2002] concerns only about multicast group management because it is built upon the overlay mesh constructed and maintained by Pastry [Rowstron and Druschel, 2001]. Pastry provides Scribe with the basic routing and content delivery functionalities, and it organizes participating peers in such a way that every peer is tagged with a unique identifier, and peers having similar contents are grouped close to each other. Scribe constructs an overlay multicast tree for each multicast group on top of the mesh built by Pastry. Therefore, it is possible that one node that participates in more than one multicast groups belongs to multiple multicast trees. Upon receipt of a packet, the node simply forwards the packet to all of its children in that specific multicast group. Consequently, those non-leaf nodes are termed as forwarders in Scribe.

In Pastry, each node is identified by using a random NodeId between 0 and $M$. Each NodeId is expressed in base $B$, and its uniqueness is guaranteed with high probability by using common message digest functions. Every node maintains its own routing table based on the leading prefix of the destination NodeId. The routing table at a node with a NodeId of $u = [u_1, u_2, ..., u_l]$ contains $l = \lceil \log_B (M + 1) \rceil$ rows and $B$ columns. The entry at the $r^{th}$ row and $c^{th}$ column represents a destination with a NodeId matching node $u$’s $r - 1$ prefix and has a value $c - 1$ at the $r^{th}$ position. More specifically, the $(r, c)$ entry represents a node $v$ with its NodeId $v = [v_1, v_2, ..., v_l]$, where $v_1 = u_1, v_2 = u_2, ..., v_{r-1} = u_{r-1}$, and $v_m = u_{c-1}$.

The resulting routing table enables quick lookup by checking the maximal prefix matching, and the entry with the maximal match is the NodeId of the next-hop node. It is noticeable that each entry is associated with only one next-hop node while there might be several nodes that meet the prefix matching requirement. Consequently, each node periodically probes each of the prospective next-hop nodes to select the one with the smallest round trip time. In Pastry, the average path length is $O(\log_d M)$ since the packet is one step closer to the destination upon each forwarding.

Each multicast group is associated with an unique key as an identifier. A newcomer joins the session by sending a join request with the group key, and the join request is forwarded until it arrives at an on-tree node that belongs to the same group. In the meantime, all the nodes that have forwarded the join request are automatically turned into forwarders. In other words, the overlay multicast tree in Pastry could be viewed as an aggregation of the individual paths. Noticeably, the loop-free feature that is desirable in any routing schemes is achieved automatically since the distance to the destination is reduced upon each hop.
Figure 2.5 gives a simple example of Scribe, where a base of 2 is used, i.e., $B = 2$, and the group key is 0000. There are 8 nodes and only 4 of them belong to the group, they are represented by the shaded nodes and they are 0100, 0101, 0011, and 0100. Further assume that they join the session in the same order, i.e., 0100 joins first and 0100 is the last one to join the session. Those nodes are located in the centric circles based on the number of matched prefix digits with the group key. When node 0100 (with only one matched digit) joins, it sends a request to node 0001 (with 2 matched digits), and the request is further forwarded to node 0000. Similarly, node 0101 sends its own request to node 0001, and since node 0001 is already an on-tree node, node 0101’s request will not be propagated any further.

Similar to Narada [Chu et al., 2002], nodes in Scribe periodically sends refresh messages to its children. In the case of failure to receive those refresh messages from its parent, the affected node simply assumes that its parent is down and a rejoining process is invoked. Scribe also has a mechanism to remove potential bottleneck in the multicast tree by limiting the number of its children.
CHAPTER 2. BACKGROUND

Scribe is scalable since the size of the routing table at each peer is $O(\log^2 B M)$, where $B$ is the base and $M$ is the size of the multicast group. Nevertheless, the performance of Scribe strongly depend on the key distribution of Pastry, and there are cases that two peers are close in terms of key distribution, but they are actually geographically far apart from each other.

2.4.2 Tree-based Protocols

Tree-based protocols build the multicast tree directly on participating peers, without the aid of a mesh. In tree-based schemes, participating nodes are organized into a tree structure for data delivery purpose, and their relationship is well-defined. The so-called “parent-child” relationship describes the relationship between an upstream node and downstream nodes. Generally, a push-based delivery scheme is employed: upon receipt of a data packet, the corresponding node simply forwards copies of the incoming data packet to all its children.

Tree-based structures are the simplest and most straightforward solution to video delivery over the Internet, and have wide applications. NICE [Banerjee et al., 2002] and Overcast [Janottii et al., 2000] are two representative examples, and we will demonstrate the principle of tree-based approaches using these two protocols.

NICE

NICE [Banerjee et al., 2002] aims at improving the scalability of overlay multicast by organizing peers into a multi-layer hierarchy, where the highest layer contains only one peer and the lowest layer consists of all the participating peers. Peers of the same layer are further grouped into several clusters, and a cluster leader is elected. Those cluster leaders form the groups that are one level up, e.g., layer $L_1$ peers consist of the cluster leaders from layer $L_0$, and so on. The size of the cluster is limited from $k$ to $3k - 1$, where $k$ is some constant.

Figure 2.6 shows an example of the hierarchy of NICE, where the little white circles represent participating nodes and the shaded boxes denote clusters in each layer. There are 8 nodes, i.e., node A, ..., node H, in layer 0, and they are grouped into smaller clusters denoting as $C_0^0$ to $C_0^3$. The leader of each cluster, B, D, F, and H in this case, form the layer one level up. In level 1, nodes are further grouped into small clusters as $C_1^0$ and $C_1^1$. This process of grouping and selecting leaders is repeated until there is only one node in the highest layer, as shown as layer $L_0$ in Figure 2.6.

Peers join the session in a bottom-up fashion. Upon joining, the newcomer, say peer $i$,
selects a cluster from the lowest layer \( L_0 \) to join. The actual joining process works like this: the joining peer \( i \) probes other peers from the highest layer to the lowest layer. Peer \( i \) first knows the existence of the peer, say peer \( j \), belongs to the highest layer by contacting a rendezvous point, then it contacts peer \( j \). Peer \( j \) notifies peer \( i \) all the cluster leaders that are one level down, and peer \( i \) chooses the closest one and queries it the cluster leaders that are reachable from it and are one level down. This process is repeated until peer \( j \) reaches the closest cluster leader that belongs to the lowest layer, and peer \( j \) joins this cluster to finish the joining process. Figure 2.7 clearly demonstrates this joining process.

The multicast delivery tree is constructed implicitly. Upon receipt of a packet, peers simply forward the packet to all the other peers that are in the same cluster. For example, node \( H \) in Figure 2.6 receives a data packet, and it forwards the copies of the incoming data packet to other cluster members, i.e., node \( G \) in \( C_0^0 \) and node \( F \) in \( C_1^1 \). The maximal length of the resulting data delivery path is bound by \( O(\log_k N) \), where \( k \) is the cluster size and \( N \) is the number of participating nodes. Consequently, the maximal node stress defined as the fanout of the node is simply bound by \( kO(\log_k N) \), as the product of the cluster size and the number of layers. NICE can achieve an end-to-end delay of \( \log_k N \). However, since all
Figure 2.7: The joining process of NICE.
the joining peers have to query along the hierarchy, peers belonging to higher layer become
the bottlenecks of the system, and once they are saturated with joining queries, the NICE
system is at the risk of being partitioned.

**Overcast**

Overcast [Jannotti et al., 2000] is designed for bandwidth-intensive applications, e.g., TV-
broadcasting. It focuses on maximizing the bandwidth of the path from the source to prospec-
tive receivers.

A newcomer, say peer $i$, joins the on-going session by contacting its potential parents,
and the source node $s$ is the default potential parent for all joining peers. Then peer $i$
estimates its available bandwidth to source $s$, and also the bandwidth to source $s$ through
each of source $s'$ children. If the bandwidth through any of the children is comparable to the
direct bandwidth to source $s$, then these children are selected and the closest one, measured
in number of hops, becomes the new potential parent and a new round of estimation starts.
This process is repeated until there is no qualified children, and the current parent under
consideration becomes peer $i$'s parent, as shown in Figure 2.8.

![Figure 2.8: The joining process of Overcast.](image)

There are several drawbacks of Overcast. First, Overcast focuses on bandwidth maximiza-
tion and one of the key building block is bandwidth estimation. Overcast simply measures
the download time of a 10K bytes file that is not accurate enough. Second, all joining peers
start with the source $s$, so the traffic concentration on upper layers puts Overcast in the risk
of being partitioned. Third, in the worst case, a joining peer has to contact all the existing peers, leading to a time complexity of $O(N^2)$ where $N$ is the number of participating peers, and this is not desirable for real-time applications, such as video-conference.

### 2.4.3 Data-driven Protocols

Apart from the traditional mesh-based and tree-based approaches, data-driven schemes, approach the problem from another angle [Pai et al., 2005; Xie et al., 2007]. They draw experience from P2P file sharing systems like Bittorrent 6, and let the data availability guide the actual data flow rather than sticking to a well-defined structure, e.g., a tree or mesh.

That being the case, data-driven approaches eliminate the overhead of maintaining a structure. However, it must have a mechanism to realize data delivery in the face of participating nodes’ dynamics. Gossip algorithms [Demers et al., 1988; Birman et al., 1999] are robust and simple. In a typical gossip algorithm, upon receipt of a data packet, it simply chooses a random set of nodes to forward the received packet, and those randomly chosen nodes do exactly the same. The random nature of gossip algorithms make them resilience to random failures, and the decentralized feature make them applicable to distributed applications. However, due to the same fact, a large amount of overhead is incurred as nodes may receive many duplications of the same data packet. Therefore, the simple push-based approach is not applicable to bandwidth intensive video streaming applications.

In order to overcome the aforementioned problems, pull-based approach is adopted by Chainsaw [Pai et al., 2005] and CoolStreaming [Xie et al., 2007], and they are elaborated shortly to demonstrate the mechanism underlying the data-driven approaches.

**Chainsaw**

Chainsaw is a pull-based system, in which data is only sent to those nodes that have requested the data packet. It eliminates the need for global routing algorithms, and participating nodes can easily recover from packet loss by simply requesting for the lost data [Pai et al., 2005].

In Chainsaw, each peer maintains a neighbor table, and each entry of this table contains the list of packets that each neighboring peer has. Upon receipt of a new packet, the receiving peer sends a NOTIFY message to all its neighbors. Each packet is associated with a sequence number, representing its position in the stream, and each peer also maintains a window of interest, reflecting the range of sequence numbers of the packets that it is interested in.

---

6 www.bittorrent.com
Furthermore, each peer has a window of availability, indicating the range of packets that it is willing to share with others.

Each peer starts the requesting process by creating a list of desired packets, representing those packets that it is in search of. Then a REQUEST message is sent based on this desired packets list and its neighbors’ windows of availability. Upon receipt of the REQUEST message, the contacted peer sends the requested packets back to the requesting peers.

There is a clearly resemblance between Chainsaw [Pai et al., 2005] and Bittorrent. Chainsaw eliminates the need for a global routing structure by implicitly constructing an unstructured overlay mesh, based on the request-available relationship between peers. However, it has two major drawbacks. First, whenever a new packet arrives at a peer, that peer has to send the NOTIFY messages to all its neighbors, incurring large amount of overhead. Furthermore, it is not clear from the original paper that whether those NOTIFY messages will be propagated further by the those neighbors that have received the messages. If those neighbors do not propagate those messages, the overhead will grow exponentially with the number of participating peers, and the system’s performance will degrade very quickly with the increase of participating peers. On the other hand, if those neighbors do not propagate those messages, that leads to the second drawback, i.e., it is in doubt that whether Chainsaw can meet the stringent delay requirement of real-time video streaming systems. It is obvious that the performance of the Chainsaw system strongly depends on the availability of data packets, and the availability of the data-availability-related information per se. The availability of new data packets must be disseminated to participating peers as quickly and efficiently as possible. In Bittorrent, peers could wait hours or even days for the completion of the file downloading. On the other hand, the real-time requirement of video streaming raises a great challenge for Chainsaw like systems.

**CoolsStreaming**

A CoolStreaming node typically has three key modules: a membership manager, a partnership manager, and a scheduler [Xie et al., 2007].

The membership manager deals with group and partner management. The joining node must contact the original server to obtain a partial node list, and it subsequently contacts the nodes in this partial list to join the on-going session.

Similar to Chainsaw [Pai et al., 2005], Coolstreaming eliminates the explicit multicast
CHAPTER 2. BACKGROUND

delivery structure by divided into small segments, as shown in Figure 2.9. Each node periodically exchanges its availability information with several neighbors, termed as partners, to retrieve unavailable data, while also supplying data to others at the same time.

Single stream of blocks with Sequence number [1,2,3,...13]

Four sub-streams {S1,S2,S3,S4}

S1 1 5 9
S2 2 6
S3 3 7
S4 4 8

Combine and decompose

Figure 2.9: Stream decomposition of Coolstreaming [Xie et al., 2007].

The incorporated scheduling algorithm enables Coolstreaming to meet the stringent playback time requirement, and the actual content delivery is achieved by using a hybrid push and pull scheme. The whole video is divided into sub-streams, as shown in Figure 2.9. Each node subscribes to a sub-stream by connecting to one of its partners (acts as its parent) using a single request (pull), and once the connection is setup, the requested node (its parent) pushes all the data blocks to its children in a continuous fashion (push), achieving timely and continuous segment delivery. However, it still suffers from the same problem, i.e., how to determinate the data availability information in a timely and efficient way.

There are some other protocols working under this scheme as well, such as YOID [Francis, 1999] and Host Multicast [Zhang et al., 2002]. Both of them use the tree-first overlay network construction algorithm, and they all use hybrid multicast data delivery, combining the traditional multicast and application layer multicast schemes together. But again the high control overhead is the major problem associated with them. Scalable ALM [Banerjee et al., 2002] tried to solve this scalability problem by organizing the group members in a hierarchical way, but it lacks the topology-awareness ability because it relies on short-term
measurement (end-to-end delay) to construct the overlay network. Furthermore, the hierarchical structure and aggregating process bring inaccuracy to the information available to the managing component, making efficiently management more difficult. Very similar to YOID and Host Multicast, topology-aware Overlay [Kwon and Fahmy, 2002] is another example, and it makes use of the underlying network topology to build the overlay network. Several other works have also been done related to application-layer multicast, including CAN [Ratnasamy et al., 2001], ALMI [Pendarakis et al., 2001].

However, Quality of service (QoS) has not been paid enough attention by these protocols. Quality of service (QoS) is crucial for multimedia applications, and there are four major difficulties associated with QoS guaranteed services. First, the diversity of the services puts different QoS constraints on the network, such as delay, delay jitter, loss ratio and bandwidth. Second, the future integrated services networks will carry both QoS-based and best-effort traffic, which makes the performance optimization more complex. Last but not least, the network undergoes dynamic changes because of the load fluctuations, and members’ free join and leave. Furthermore, the ever-growing size of the network makes it very difficult to gather the most up-to-date state information of the network to support efficient routing and information delivery [Chen and Nahrsted, 1998]. To realize wide-area application layer multicast, QoS standards (bandwidth, delay, delay jitter, and packet loss probability [Wang and Hou, 2000]) have to be assured. In addition, a lot of works need to be done to optimize the network to achieve efficiency and reliability at the same time, and this is out of the scope of this thesis.
Chapter 3

Adaptive Gossip-based Membership Management Algorithm

The very first step for any group communication to take place is to have an efficient and robust group membership management algorithm, i.e., a method to define a group and to maintain this group in the presence of dynamics, due to members’ joining and departure. There is no exception for Application Layer Multicast (ALM), and this chapter focuses on the membership management perspective, in particular, how to do it in a cost-effective way.

It is difficult to have a scalable and efficient membership management algorithm in the Peer-to-Peer (P2P) context where there is no central entity that could potentially facilitate the execution of such a membership management algorithm. Each member, or end system, is equivalent to any other and acts as a server as well as a slave. Since there is no central entity to handle membership management related tasks, peers have to rely on themselves and flooding is sometimes the only choice. Most existing membership management algorithms impose large amount of overhead on networks. The crux is how to find a cost-effective membership management algorithm, given the inherent dynamics and distributed nature of P2P networks.

The answer to this challenge and the contribution of this chapter is a new gossip-based membership management algorithm. This algorithm captures the changes in the network and adjusts the parameter settings dynamically, bringing adaptivity to reduce overhead. Simulation results indicate that the proposed gossip-based membership management is effective. A maximum of 50% reduction can be achieved in terms of network overhead on core network components, such as backbone links and attached routers, without sacrificing reliability.
3.1 Motivation

In this section, group membership management is defined, followed by discussion of problem formulation, bringing our research into perspective.

Definition of a Group Membership Management Algorithm

To put it simple, a group membership management algorithm needs to have at least the following two functionalities:

- a means to identify and distinguish each group member, e.g., IP address and port number could serve this purpose; otherwise, there is no way for members to communicate with each other.
- the ability to collect and disseminate membership-related information, e.g., the presence of a new member, or the failure of an existing member.

A good membership management algorithm is vital to the success of group communication, and the quality of a membership management algorithm could be judged by the following two criteria:

- **Reliability**: The failure/departure of participating nodes must be detected quickly and the remaining nodes must be notified of this topology change in a timely fashion. In other words, the membership management algorithm should remain functional even in a highly dynamic environment.
- **Scalability**: The overhead should not grow linearly with the number of participating nodes, and the resulting membership management algorithm should handle nodes joining and departure at a minimized cost, accommodating large number of nodes.

What is the Problem?

In traditional IP multicast, group membership management is done in a transparent way: both sender(s) and receivers register with routers. Routers take care of all the membership management-related activities, e.g., track active receivers and keep membership information up-to-date. Nevertheless, this scheme implicitly makes use of the fact that most routers are very stable, and could keep running for quite a long period of time without failure. Nonetheless, the unique feature which distinguishes between native IP multicast and Application
Layer Multicast is that with ALM there are no device like routers set aside to manage group membership. On the contrary, in a dynamic environment, in particular, Peer-to-Peer (P2P) networks and Application Layer Multicast (ALM henceforth), there is no central server and the overlay is built on-the-fly, normally in a distributed way. This raises the need for a robust and scalable membership management algorithm. These two requirements, together with the inherent dynamics of P2P networks, make it a great challenge to design a cost-effective membership management algorithm for P2P networks.

A straightforward and easy way is to make use of the client-server architecture. Some centralized servers are responsible for tracking all the membership information. But realistically, the ALM group is formed on-the-fly and changes very frequently. It is difficult, if not impossible, for any server to maintain a full list of the members in a dynamic large-scale network. Therefore, a fully distributed membership management algorithm is a necessity in this case.

Epidemic or gossip-based algorithms are good candidates [Demers et al., 1988], and a gossip-based membership management algorithm has been published [Ganesh et al., 2003]. It disseminates membership information in an epidemic way, that is, every member periodically picks some other members at random to send the membership information. This approach lacks flexibility and imposes the same amount of overhead on the network regardless the characteristics of the network. This non-adaptability greatly hinders its applicability in an ever-changing environment like P2P networks.

According to Sripanidkulchai et al. [2004], most applications in ALM are short lived, with an average of 3.3 requests from a single IP address during a session. In such a highly dynamic environment, the major concern is how to capture and communicate these changes among the remaining users in a timely and efficient manner, and also how to balance network overhead, computational complexity and network performance. This is exactly the contribution of this chapter: a new gossip-based membership management algorithm that associates each participating user with a weight, representing the probability that it will be chosen as the gossip target, according to its access bandwidth and other realistic parameters; in the mean time, peers’ weights are constantly adjusted, reflecting the dynamic characteristics of the underlying overlay network.
CHAPTER 3. ADAPTIVE GOSSIP-BASED MEMBERSHIP MANAGEMENT ALGORITHM

3.2 Related Work

Many research papers have been published, both in traditional multicast context and the new overlay multicast environment. This section surveys the related work and puts our work into perspective.

3.2.1 Group Membership Management

Group membership management protocols are crucial to the success of multicast because they provide applications with dynamic membership information. There are two types of membership management mechanisms: local group management [Haberman and Martin, 2001] and global multicast routing [Deering et al., 1994]. In a traditional network layer multicast scheme, a local group management algorithm enables multicast routers to be aware of the presence of group members within their local networks by letting every participating member register to the router. Hence, it only applies to LAN or several LANs [Haberman and Martin, 2001]. In contrast, the global multicast routing mechanism learns of the existence of the members by exchanging membership information among the routers distributed across wide-area networks [Deering et al., 1994; Ballardie et al., 1993]. The most common local group management mechanism is Internet Group Management Protocol (IGMP) [Haberman and Martin, 2001]. It periodically updates membership information by using a query/reply model. However, none of these protocols are suitable for large P2P networks or ALM, either due to large overhead or the chance of a central point of failure. For example, PIM [Deering et al., 1994] builds a shared multicast distribution tree centered at a rendezvous point. It suffers from traffic concentration and the possibility of a central point of failure. In Narada [Chu et al., 2002], a mesh was built among participating group members, with each member maintaining a full list of the other group members, rendering a large amount of overhead, in the order of $O(n^2)$, making it inapplicable to large-scale applications. The increasing popularity of ALM requires a new membership management algorithm.

3.2.2 A Scalable Protocol with a Non-scalable Membership Management Algorithm

Since the early work of YOID [Francis, 1999], a large body of work has been done on ALM, e.g., Narada [Chu et al., 2002], Host Multicast [Zhang et al., 2002], ALMI [Pendarakis et al., 2001], etc. Nevertheless, they each made the same assumption that all the participating members are visible to each other; in other words, every node should keep track of all the
other nodes since there is not a central entity that does it for them. For a network consisting of \( n \) nodes, each node needs to devote \( O(n^2) \) storage space for membership information. Even worse is the communication and computational overhead. Whenever a peer joins or quits the session, the relevant information is flooded throughout the entire network, incurring an overhead of \( O(n^2) \). In a highly dynamic environment, like ALM, the logical links forming the overlay will quickly become saturated because of this “membership update storm”.

Even though the protocols per se are scalable, the large amount of control overhead used for membership management limits its use to only a small group of users. Therefore, a scalable group membership management algorithm has to be in place to facilitate the implementation of ALM.

Presently, there are two ways to address this problem. One takes advantage of the reliability and scalability offered by gossip-based algorithms, where each node only needs to keep a random partial view of the system, rather than a complete picture of the system [Birman et al., 1999; Lin and Marzullo, 1999; Demers et al., 1988; Ganesh et al., 2003]. On the other side of the design spectrum is the approach of first constructing a Harary graph, then messages are flooded over the constructed Harary graph [Lin et al., 2000]. The crux of the second approach relies on the optimality properties of the constructed Harary graph: which can strike a good balance between the number of messages and the achievable reliability. Nevertheless, global membership information is still needed to build the associated Harary graph, limiting its applicability. In contrast, due to its inherent simplicity and scalability, gossip-based algorithms have been widely used. Therefore, we turn our attention to gossip-based algorithms.

### 3.2.3 Gossip-based Algorithms

Gossip-based algorithms seem to offer scalability and reliability [Birman et al., 1999; Lin and Marzullo, 1999; Demers et al., 1988]. They work by spreading packets among members in a random way, just like the way in which some infectious diseases spread. It could be summarized as follows: in each gossip round, of a fixed time interval, each member chooses a fixed number of members (termed gossip selected targets or gossip target selection), uniformly at random, to send a copy of the messages that it has received. Reliability is achieved, due to this protocol’s randomized nature, at the expense of redundant messages in the network.
CHAPTER 3. ADAPTIVE GOSSIP-BASED MEMBERSHIP MANAGEMENT ALGORITHM

Directional Gossip

Directional Gossip [Lin and Marzullo, 1999] is targeted at remedying the shortcoming of traditional gossip algorithms by trying to reduce the communication overhead. The directional gossip algorithm works like this. Each LAN is associated with a gossip server, which only knows of its immediate neighbors in WAN. The gossip server prunes links from the multicast tree so as to minimize overhead. However, they did not specify how to maintain this structure and elect this gossip server in the original paper.

Bimodal Multicast

Bimodal Multicast [Birman et al., 1999] tried to combine the efficiency of tree-based multicast with the reliability and scalability associated with gossip-based algorithms. Messages are disseminated in two phases. First, messages are delivered using unreliable multicast. Then, packet losses are recovered using gossip-based anti-entropy. However, as pointed in the original paper, only membership service needs the full membership of the multicast group, i.e., it relies on a non-scalable membership management algorithm.

SCAMP

SCAMP [Ganesh et al., 2003] is a distributed gossip-based membership management algorithm. Each node maintains a partial view of the system of size $\log(n)$, and periodically exchanges membership-related information with other peers. Nevertheless, the major drawback is its lack of flexibility, both in terms of time and space. Each member gossips to a fixed number of “neighbors” in a fixed time interval. This means that it could not adapt to the ever-changing network dynamics, imposing roughly the same amount of overhead on the network regardless of the current characteristics of the network.

3.3 Proposed Approach

The algorithm proposed in this chapter offers a remedy for the pitfalls of SCAMP [Ganesh et al., 2003], and a detailed description of the protocol is given in this section. The proposed algorithm works in control plane, decoupled from data delivery and manages the membership in a fully distributed fashion. Before delving into the details of the algorithm, some relevant terminologies and metrics are explained.
CHAPTER 3. ADAPTIVE GOSSIP-BASED MEMBERSHIP MANAGEMENT ALGORITHM

3.3.1 Terminologies and Metrics

For the sake of simplicity, an ALM session with a single sender is considered here. All participating nodes join the group freely. These nodes and the connections between them form a multicast overlay on top of the IP layer. Each node keeps partial membership information about some of the participants. Similar to the idea of [Ganesh et al., 2003], two separate lists are used. InView contains the nodes that know the node and OutView contains the nodes that it knows. As an example, in Figure 3.1, node 4’s InView contains node 1, 2 and 3; its OutView includes node 5, 6 and 7. Each entry in the list corresponds to one participating peer, named as “neighboring node”, and it contains the basic Quality of Service (QoS) attributes of that “neighbor”, such as the session “ID”, residual bandwidth (download and upload bandwidth depends on the context) and so on. Using two sets of membership container facilitates the management of membership information, as this reduces the lookup time and simplifies programming complexity. In addition, it facilitates the quick replacement of departing nodes so as to minimize the service disruption caused by nodes’ departure.

\[\text{Figure 3.1: A simple example of membership view.}\]

- **Graceful departure**: The departing node informs all the nodes in its InView of its departure and picks up some of the nodes from its OutView to replace itself, if possible, taking the QoS parameters into consideration. This step reduces the impact of node departure.

- **Non-graceful departure**: Each node periodically sends its latest status to the nodes in its OutView as an update or heartbeat message. If one node, say node \(i\), has not heard from its neighbor, node \(j\), for a while, it will contact node \(j\) to check the status of \(j\). Node \(i\) will delete node \(j\) from its InView if it has not received the confirmation within a predefined timeout period, and an update message about this change will also be sent.
Nodes periodically send their current QoS parameters to their neighbors, and the time interval between consecutive updating is termed a “gossip round”, which is adjusted dynamically as well. In each round, the receiving node gossips to others in a pseudo-random way by choosing the targets preferentially, according to a metric defined later. Upon receipt of an update message, each node preferentially picks up some nodes from its OutView lists to pass the message, similar to the way in which some infectious disease spreads. Eventually, most of the nodes will get this information. However, it is not expected that all the nodes will have the same view after several rounds of gossiping. The reason is that the network is in a highly dynamic state throughout the multicast session. The overhead generated by letting all the nodes have very accurate and the most up-to-date information about the entire network is too high. From another point of view, it is inefficient to get the data packets from other nodes that are very far away. For example, a user located in Australia is more willing to get data locally, rather than from another user in Europe. So, the information contained in the gossip message has different meanings for different users, and only part of the population may be interested in a particular piece of information. In order to account for this varying importance or meaning, the “goodness” of a node is defined as follows:

\[ m = C \times \frac{B_i \times B_j}{D^2} \] 

(3.1)

where \( C \) is the number of “children” it has, i.e., the number of nodes that are currently receiving data packets from it; \( B_i \) and \( B_j \) stand for the residual bandwidth of node \( i \) and \( j \) respectively, and \( D \) is the delay between the nodes, or simply the network distance. The idea is very simple, the goodness or desirability of a node is proportional to its available bandwidth, and is inversely proportional to its delay. In other words, a “good” node should have high bandwidth and/or low delay, e.g., the node near the sender has a higher energy, thus has a higher influence on others. In fact, this metric of “goodness” is borrowed from physics. In field theory, two objects have an attraction force of:

\[ F = G \times \frac{m_i \times m_j}{r^2} \] 

(3.2)

where \( G \) is a constant, \( m_i \) and \( m_j \) are the mass of the two objects \( i \) and \( j \) respectively, and \( r \) is the distance between them. Comparing Equation 3.1 with Equation 3.2 shows that the bandwidth in ALM context is comparable to mass in physics: the larger the bandwidth, the
bigger the “attraction” between nodes. In addition, the Euclidean distance in Equation 3.2 is replaced by network distance in Equation 3.1, and the meaning is self-explanatory. The rationale behind this metric is two-fold:

- It has been proven that the scale-free [Barabasi and Albert, 1999] property of the network favors information spreading, especially if those nodes with higher degree, or named as hubs, are reached by the gossip message, and the gossip message propagates through the network at a very high speed [Barthelemy et al., 2004]. The proposed algorithm makes use of this property. By directing the gossip message to the prospective nodes who might be interested in the information, faster convergence and overhead reduction are achieved without scarifying the reliability associated with gossip-based algorithms.

- It has been shown [Gnu, 2000] that nodes connected by a 56Kbps modem are unable to handle more than 20 queries per second, corresponding to a network of about 1000 nodes. If these nodes fail, the network may become fragmented. Therefore, the control overhead has to be distributed unevenly according to the different capacities of nodes, reflecting the heterogeneity of the network.

As stated before, gossip round and gossip target selection are two important parameters that determine the behavior of a gossip-based algorithm. Therefore, the ability to adjust these two parameters plays a crucial role in reducing the overhead, and the pre-defined metric serves this purpose.

- One use is in the time domain, adjusting the gossip round interval. Each node periodically calculates its so called “energy” or “goodness”. $B_i$ is the “download bandwidth” of the current node, and it is the smaller one of the streaming rate and the node’s residual download bandwidth. For example, a node connected by an ADSL connection with an uplink speed of 128Kbps and download speed of 512Kbps joins a multicast session running at 2Mbps. Clearly, the “download bandwidth” is, therefore, $\min(512K, 2M) = 512Kbps$. $B_j$ is the residual uplink bandwidth of the node itself, and it is 128Kbps in this case. $D$ is the network distance from the sender to this node itself. A node with a higher “energy” can be interpreted as having high upload bandwidth and/or low delay, and can potentially offer better service to other nodes. It indicates that more nodes will be interested in the information about it. Therefore, it ought to have a shorter time interval between consecutive gossip rounds. De-synchronizing the
gossip rounds between all the participating nodes can also avoid the message explosion problem.

- The other use of the metric is in the space domain, adjusting the gossip target selection. Upon receipt of the gossip message, each node picks up some nodes from its OutView to pass the message. The metric will act as an index to direct the message to the nodes for which this message might be useful. In Equation 3.1 $C$ is the “children” account; the more nodes it serves, the more responsibility it should take, i.e., it needs to have more information to cope with the failure in the future; $B_i$ is the residual upload bandwidth of the neighboring node; $B_j$ is the residual download bandwidth of that node. $D$ is the delay between itself and the neighboring node. It indicates the desirability of the potential path. As stated before, the gossip message has different meaning for different nodes. For example, when an event is streamed at 512Kbps, a node with a download bandwidth of 1Mbps may not be interested to know the information about another node with an available upload bandwidth of just 56Kbps. The idea is to direct the gossip message to those nodes with lower delay, higher bandwidth and higher child count. It has been shown that once the highly connected nodes, called hubs, have received the message, it will spread over the entire network at a very high speed [Barthelemy et al., 2004].

3.3.2 Detailed Algorithm Description

This section describes the proposed algorithm in detail. Before moving to the full details of the algorithm, several assumptions need to be explained in order to justify the applicability of the proposed approach:

- When a node joins for the first time, certain information about one or several on-tree nodes will be provided by some out-of-band mechanism like bootstrap.

- Each node holds an approximation of the magnitude of the delay between the sender and itself and what is the current sending rate.

These assumptions are also used in other protocols, and delay can be measured actively by pinging, or passively by checking the timestamps of received packets. The proposed protocol consists of three building blocks:
1. Join: A newly-joined node is provided with an initial list by the bootstrap mechanism, and it sends a **JOIN** message containing its own quality of service (QoS) demands (e.g., bandwidth and/or delay) to the target nodes selected from the initial list. If this joining process succeeds, a confirmation message will be sent back to the joining node and the joining node will be able to receive data; if the joining process fails, a reply packet will be sent back as well, containing the contacted node’s current QoS attributes, and all the intermediate nodes will check this packet and decide whether to update or not (depending on whether the delay and residual bandwidth have changed or not). If necessary, some data packets will be sent back as well, to reduce the joining latency. The joining node is provided with several nodes to contact by the bootstrap algorithm. It contacts all of them until one of them sends back the acknowledgement to finish the joining process.

A simple example is illustrated in Figure 3.2. The joining node, node 9, sends a **JOIN** message to node 5; node 5 accepts the joining request and sends back a confirmation message. All the intermediate nodes (node 8) will update the entry associated with the contacted node (node 5).

2. Gossip: Each node periodically calculates its current “energy” and adjusts the gossip interval accordingly. With an initial gossip interval \( t \), the next gossip interval is calculated using a normalization \( t' = \frac{E}{\sum E} \), where \( E \) represents the calculated “desirability”, and it is normalized over \( N \) rounds. When the next gossip round starts, it sends a gossip message containing its current QoS attributes to some nodes chosen pseudo-randomly according to the probability \( p = \frac{E}{\sum E} \), where “energy” is normalized over all the members in the list. Upon receiving the message, those receiving nodes will update their neighboring list if necessary and pass the message to some nodes chosen based on the metric as well. A count is associated with the gossip message, indicating how many nodes have saved and propagated this message. On one hand, it is important to make sure that this message has been processed by some nodes already. On the other hand, it can be used as a last resort for scope control to reduce the volume of gossip messages. Algorithm 1 summarizes the gossip target selection process, where line 4 calculates the selection probability for each in-list member and line 5 describes how to send the message to each member with the calculated probability. Algorithm 2 details the gossip round adjustment procedure, where line 6 calculates the length of the next gossip round and line 7 sets the gossip round accordingly.

41 (March 23, 2008)
CHAPTER 3. ADAPTIVE GOSSIP-BASED MEMBERSHIP MANAGEMENT ALGORITHM

Algorithm 1 Gossip Target Selection

1. if receipt the gossip then
2. update its membership list
3. for every node \( n_i \in \text{OutView} \) do
4. \( p_i = \frac{E_i}{\sum_i E_i} \)
5. Send\((n_i, p_i)\) //send the message to node \( n_i \) with a probability of \( p_i \)
6. end for
7. end if

3. Cope with Failure: By sending and receiving these gossip message, each node updates its neighboring list. When random failures occur, the affected nodes can select the most suitable parent-to-be from its updated neighboring list, and resume the service without huge disruption.

To summarize, the idea of the proposed algorithm is to reduce the membership management overhead by adjusting the gossip interval and gossip target selection dynamically.
CHAPTER 3. ADAPTIVE GOSSIP-BASED MEMBERSHIP MANAGEMENT ALGORITHM

Algorithm 2 Gossip Round Adjustment

1: if it is time to gossip then
2: for every node \( n_i \in \text{OutView} \) do
3: \( p_i = \frac{E_i}{\sum_i E_i} \)
4: Send\((n_i, p_i)\)
5: end for
6: \( t' = t \frac{E}{\sum_i E_i} \)
7: adjust the gossip interval
8: end if

3.4 Analytical Results

In this section, detailed mathematical analysis is presented, including an analysis of the network size and the converging speed of the proposed gossip-based algorithm. The analytical modeling provides the theoretical basis for the simulation parameters setting and a general framework for analyzing the performance of the proposed approach.

Network Size

The presence of participating peers implicitly defines a multicast group. Peers may join the current session at any time and the arrival of these peers is modeled as a Poisson distribution with rate \( \lambda \). Furthermore, peers’ sojourn time follows an exponential distribution with rate \( \mu \). This modeling strategy has been used by classical teletraffic analysis, and is also in good agreement with the recent measurement-based study of the peer-to-peer networks [Saroiu et al., 2002].

This chapter focuses on the group membership management aspect, and it is in the control plane that is orthogonal to the data delivery issues. Therefore, for the sake of simplicity, it is assumed that all incoming peers are admitted into the group without imposing any admission control. Since there is no rejection, this particular peer-to-peer ALM network can be modeled as an infinite server Poisson queue with parameters \( \lambda \) and \( \mu \). Let \( M(t) \) represent the total number of nodes in the system that have arrived by time \( t \), and \( N(t) \) denote the number of nodes in the system at time \( t \). Obviously, \( N(t) \) is the conditioned form of \( M(t) \) [Ross, 1970].

\[
P\{N(t) = j\} = \sum_{n=0}^{\infty} P\{N(t) = j|M(t) = n\} e^{-\lambda t} \frac{\lambda t^n}{n!}
\]

(3.3)

Noticeably, the probability that a node arrives at time \( x \) will remain in the system at
time $t$ is $1 - e^{-\mu(t-x)}$. Therefore, given that $M(t) = n$, the probability of an arbitrary peer still being present in the system at time $t$ is

$$p = \frac{1}{t} \int_0^t (1 - e^{-\mu(t-x)})dx = \frac{1}{\mu t} (1 - e^{-\mu t})$$  \hspace{1cm} (3.4)$$

It is easy to see that

$$P\{N(t) = j| M(t) = n\} = \binom{n}{j} p^j (1 - p)^{n-j} e^{-\lambda t} \frac{\lambda^n}{n!}$$  \hspace{1cm} (3.5)$$

Combined with Equation 3.4,

$$P\{N(t) = j\} = \sum_{n=j}^{\infty} \binom{n}{j} p^j (1 - p)^{n-j} e^{-\lambda t} \frac{\lambda^n}{n!}$$  \hspace{1cm} (3.6)$$

$$= e^{-\lambda t} \frac{\lambda^j}{j!} \sum_{n=j}^{\infty} \frac{(\lambda(1-p))^{n-j}}{(n-j)1} = e^{-\lambda tp} \frac{\lambda^j}{j!}$$  \hspace{1cm} (3.7)$$

In other words, $N(t)$ has a Poisson distribution with mean

$$\lambda \int_0^t (1 - e^{-\mu x})dx = \lambda tp = \frac{\lambda}{\mu} (1 - e^{-\mu t})$$  \hspace{1cm} (3.8)$$

when $\mu t$ approaches $\infty$, the network size becomes $\frac{\lambda}{\mu}$. This means that the network will reach an equilibrium state even under high dynamics. Nodes join and leave, but the total number of nodes remains roughly the same. So a network with a fixed size is simulated in our experiments.

It is obvious that the dynamics of the network has an impact on the performance of any membership management algorithm: the higher the dynamics, the more difficult it is for the membership management algorithm to work properly. Consequently, the dynamics of the network, as a very important parameter, should be represented and reflected in the validation of the proposed approach. Half-life [Liben-Nowell et al., 2002] was proposed to capture the dynamic aspect of the network, which is the time taken by the system to change half of the nodes. Clearly, the length of half-life is a direct measurement of the dynamics of the network, and we validate our algorithm under different half-life settings.
CHAPTER 3. ADAPTIVE GOSSIP-BASED MEMBERSHIP MANAGEMENT ALGORITHM

Gossip Spreading

A simple recursive probability describes the converging speed of the algorithm. Let \( G_0(x) \) represent the generating function for a scale-free network, then:

\[
G_0(x) = \sum_{0}^{\infty} p_k x^k = c \sum_{1}^{\text{max}} k^{-\alpha} x^k
\]

(3.9)

where \( p_k \) is the probability that a randomly chosen node has a degree of \( k \). The degree distribution follows a power-law distribution and assume there is a minimum degree 1 and a cutoff degree \( \text{max} \). \( c \) is a normalization constant and subject to the requirement \( (G_0(1) = c \sum_{1}^{\text{max}} k^{-\alpha} = 1) \). When a node \( n_k \) with degree \( k \) generates a gossip message, potentially, only a fraction of the nodes will be interested in this message, which is expressed as:

\[
E[k] = \sum_{0}^{k} p_k = \sum_{0}^{k} k^{-\alpha}
\]

(3.10)

Let \( R_k(t) \) represent the number of nodes that have received the information about node \( n_k \), and \( S_k(t) \) denote the number of nodes that are not aware of it. \( R_k(t) \) forms a Markov chain with a sample space of \( \{1, \ldots, n\} \). Now \( S_k(t) \) can be expressed using a simple probability expression [Pittel, 1987]

\[
S_k(t + 1) = S_k(t)(1 - \frac{1}{n})^{n(1-S_k(t))}
\]

(3.11)

When \( n \) is large, it leads to:

\[
S_k(t + 1) = S_k(t)e^{-1}
\]

(3.12)

This indicates that the population of the susceptible nodes, i.e., nodes that are not aware of the information, shrinks exponentially, and if we normalize over all the degree distribution, we have:

\[
< r > = \sum_{0}^{\text{max}} p_k \ln(E[k]n) = \sum_{0}^{\text{max}} k^{-\alpha} \ln(\sum_{0}^{k} k^{-\alpha} n) = \frac{\alpha}{k^{\text{max}} - 1} \ln(n) + o(k)
\]

(3.13)

By intentionally choosing the node with higher degree, the information can be spread over the whole network in \( O(\ln(n)) \) time unit. Let \( t_k \) denote the average gossip overhead for
node \( n_k \), and \( S_k(t) \) can be related to \( t_k \) as:

\[
\frac{S_k(t)}{N} = e^{-t_k}
\]  

(3.14)

This can be explained heuristically as the probability for any node fails to receive all the messages \( N \times t_k \) is \((1 - \frac{1}{N})^{Nt_k} = e^{-t_k}\). Rewriting Equation 3.14 leads to:

\[
t_k = -\ln\left(\frac{S_k(t)}{N}\right) = -\ln(1 - E[k])
\]  

(3.15)

The average gossip overhead can be calculated over all the nodes:

\[
<t> = -\sum_k p_k \times \ln(1 - E[k])
\]  

(3.16)

In other words, the gossip overhead will not grow linearly with the size of the group but it only grows logarithmically with it.

### 3.5 Experimental Results

This section describes the simulation setup and explains the simulation results. The simulation results show that the proposed algorithm successfully remedies the drawbacks of SCAMP [Ganesh et al., 2003].

#### 3.5.1 Simulation Setup

In order to simulate a realistic and meaningful network, 600000 nodes, representing 600000 peers were generated using the GT-ITM topology generator [Zegura et al., 1996]. Peers are randomly distributed among the generated topology. Peers join the on-going session following a Poisson distribution and their lifetime are drawn independently and identically from an exponential distribution.

In terms of network dynamics, peer-to-peer networks are quite different from the traditional network, since most of the nodes are expected to get data from others, instead of from the data source directly. Hence, nodes’ perceived QoS is not truly independent on that of each other. The behaviour of some nodes will affect that of others, such as the nodes who are taking the responsibility to forward data. The definition of network dynamics have to be redefined, taking the properties of peer-to-peer network into account. Half-life [Liben-Nowell et al., 2002], which is the time taken by the system to change half of the nodes, has been
OmNet++ lacks the ability to control the achievable confidence level with required precisions. Therefore, Akaroa [aka] is employed to control the duration of our simulation to achieve a 95% confidence level, and it is able to analyze the observations at runtime and decides when to start collecting data and when to stop the simulation as enough observations have been collected.

3.5.2 Metrics of Interest

This section introduces several performance metrics, used to characterize, validate, and compare the proposed approach with SCAMP [Ganesh et al., 2003].

Latency penalty

The flexibility of ALM is gained at the price of a performance penalty. Data packets are relayed at end systems rather than by routers, and this leads to a longer end-to-end latency. On the other hand, video streaming has a stringent requirement of end-to-end delay. Therefore, special consideration needs to be taken to meet the required end-to-end delay. This latency is captured and presented by using hop counts, both for data packets and the gossip message, in all the simulations.

Link Stress

Link stress is defined to be the number of identical copies of a packet (data packet and gossip message) traversing over a physical link, and it is a direct measurement of the efficiency of the algorithm. In general, the gossip overhead could be measured by this link stress quantity.
3.5.3 Simulation Results

Since video streaming has a stringent requirement of end-to-end delay, and in ALM, the data forwarding job is taken by individual peers, so average hop count and delay of the data packets is of interest. As can be seen from Figure 3.3, the average hop count and delay increase proportionally with half-life time. In other words, they decrease proportionally with the network dynamics. This can be explained by the fact that when the network is more stable, a higher number of nodes will take the responsibility to replicate and relay the data, and consequently, more nodes will get data locally rather than from the source directly. On the other hand, when the network is highly dynamic, it is difficult for the nodes to get data locally, and they have no choice, forcing them to turn to the source directly. It is noticeable that when the average hop count equals one, most of the nodes will contact the data source directly, and the corresponding half-life is around 60 seconds, representing a highly dynamic network. In reality, we would expect a network to have a much more stable behavior.

Figure 3.3: Average hop count and delay of data packets of the proposed algorithm.
Figure 3.4 shows the link stress for data packets. It is necessary to point out that the traces labeled as tier1, tier2 and tier3 routers represent the core network, local region network, and local access network respectively. The trend clearly shows that stress for local access networks remains relatively constant. This can be interpreted that the stress is in proportion to the number of nodes participating the session, so is relatively constant. Link stress for the local region network decreases slightly when the network becomes more stable, compared with the link stress for the core network, which has an approximately 44% decrease. The explanation for the phenomenon is that when the network becomes more stable, more nodes will get data locally instead of from the data source, and thus some of the data packets will bypass the core network.

Figure 3.5 displays the link stress of the gossip messages. We investigate the link stress for core, local regional and local access networks, represented by tier-1 router, tier-2 router, and tier-3 router respectively in the figure. The link stress for local access networks remains roughly the same except for small fluctuations, and this can be viewed as the fluctuation of the number of participating nodes. It is interesting to see that there is a crossover point corresponding to a half-life time of around 180 seconds. At first, when the network is highly dynamic, the link stress for the core network is twice that of the local regional network. This
CHAPTER 3. ADAPTIVE GOSSIP-BASED MEMBERSHIP MANAGEMENT ALGORITHM

Figure 3.5: Link stress of gossip overhead of the proposed algorithm.

Figure 3.6: SCAMP: Average hop count and delay of data packets.
indicates that when the network is not very stable, in order to exchange information, the gossip messages have to disseminated across the local regional and core network. When the network becomes more stable, the link stress for the local regional network outgrows that of the core network after the crossover point. This indicates that nodes are only exchanging information locally in a more stable network. That means our algorithm is adaptive, and it can confine the gossip messages to local regions, reducing the load of the core networks, and scalability is achieved. As can be seen from Figure 3.6, the average hop count and delay of the data packets in SCAMP follow the same trend shown in Figure 3.3, increasing with the stability of the network.

![Figure 3.7: SCAMP: Link stress of data packets.](image)

When it comes to the link stress of data packets in the well-known SCAMP, Figure 3.7 shows the same pattern with that of our algorithm, except for the slight differences in tier2 and tier3 routers. This shows that the superiority of our algorithm as it can make use of the hierarchical structure and discover the local resource more efficiently. Hence the nodes can get data locally instead of from the nodes which are far away, preserving the backbone bandwidth.
However, there is a big difference, between the proposed algorithm and SCAMP, in the link stress for gossip message overhead. If we compare Figure 3.5 with Figure 3.8, it can be concluded that our algorithm uses the network resource more efficiently. When the network becomes more stable, all the nodes can capture this change and exchange information locally. However, in the case of SCAMP, this change can not be sensed by the nodes, and consequently, the communication overhead remains roughly constant proportional to the number of participating nodes.

Due to its adaptive nature, the proposed algorithm is effective for reducing the gossip overhead. On the other hand, the mathematical analysis presented in section 3.4, in particular Equation 3.16, shows that the gossip overhead grows logarithmically with the network size. Figure 3.9 compares the simulation and analytic result in terms of gossip overhead, where the $x$-axis is the network size, which is simply the number of participating peers; and the $y$-axis is the number of gossip overhead packets, averaged over all peers. The trend is very clear, as the gossip overhead grows very slowly with the number of participating peers, indicating that our algorithm is efficient.

A good and effective membership management algorithm of ALM should enable most participating peers to remain connected, despite the inherent dynamics of ALM. On the
CHAPTER 3. ADAPTIVE GOSSIP-BASED MEMBERSHIP MANAGEMENT ALGORITHM

**Figure 3.9: Gossip overhead analysis.**

**Figure 3.10: Reliability analysis.**
other hand, due to ALM’s unique nature peers receive multicast data packets from other participating peers, rather than from the data source directly. Therefore, the reliability of the ALM network can be measured by the average Quality-of-Service (QoS) perceived by each peer. In addition, this thesis focuses on VoD-like video stream applications, i.e., we can assume that there is a Constant-Bit-Rate (CBR) traffic source emitting packets at a constant data rate. Ideally, the amount of any peer’s received portion of the video packets should be proportional to its lifetime. Therefore, the Quality-of-Service (QoS) perceived by any peer can be reflected by the ratio of the amount of received video packets to the lifetime. Figure 3.10 shows that the proposed algorithm is effective, i.e., most peers are able to remain connected and the average QoS is consistently above 80%.

3.6 Conclusion

The effect of network dynamics on the performance of group membership management was investigated in this chapter. It is very difficult to maintain the group structure in a highly dynamic environment, and it is especially the case for P2P ALM: there is no central entity that could facilitate the membership management task, and peers have to maintain their own “neighboring list”.

The crux is how to find a cost-effective way to perform the membership management task because simple techniques like flooding is only applicable to small groups. Gossip-based algorithms strike a balance between operational overhead and the achievable reliability. However, traditional gossip-based membership management algorithms had a scalability problem due to their lack of adaptability: they could not adapt to the ever-changing environment and impose the same amount of overhead on the underlying physical networks. As a remedy, an adaptive gossip-based membership management algorithm is proposed in this chapter. It adapts to changes in the network, confining the communication overhead to local regions by dynamically adjusting the length of the gossip round and probability of gossip target selection. Therefore, it can use the network resource more efficiently.

This algorithm has been analyzed mathematically and validated by simulation. Simulation results show that a 44% reduction can be achieved in terms of gossip message overhead on core network components, without sacrificing reliability. Focus is shifted from the control plane to data plane in the next chapter, in particular, how to improve the reliability perceived by all peers.
Chapter 4

Resilient Application Layer Multicast

Reliability is one of the major concerns for most real-time applications, and there is no exception for Application Layer Multicast (ALM). Furthermore, it is a much more serious issue in ALM because participating users may join and depart at will. The overlay network, built on-the-fly, is highly dynamic. To make things worse, single-tree-based multicast structures are commonly used in real-world applications, due to their efficiency. In these single-tree-based schemes, users’ departure will cause severe service disruption to all the downstream users. In this chapter, we exclusively focus on single-tree-based ALM because of its pervasive use in real-time applications. We propose a new algorithm to enhance reliability for these single-tree-based ALM. The proposed algorithm organizes participating peers into a hierarchy in such a way that it reflects their relative stabilities (represented by their “rank”), rather than their geographical proximities or other criteria. It improves the reliability by a minimum of 20% without sacrificing scalability. Detailed mathematical analysis was carried out to validate the proposed approach. This includes the modeling of both the multicast tree and the corresponding logical hierarchy, together with the reliability analysis of the resulting multicast tree. The analysis demonstrates that the proposed approach is effective in terms of improving reliability for ALM. Simulation results reveal that our algorithm outperforms the benchmark algorithm and further demonstrates that making ALM reliable is feasible.
CHAPTER 4. RESILIENT APPLICATION LAYER MULTICAST

4.1 Motivation

Single-tree-based structure is commonly used in ALM due to its efficiency. Therefore, reliability has become the major concern in ALM because those end hosts that are responsible for all the data replication and forwarding tasks are not as stable as routers. For example, in Figure 4.1, nodes 3, 4, 5, and 6 are receiving data from node 2. If node 2 departs all of a sudden, all the downstream nodes, i.e., 3, 4, 5, and 6 will experience service disruption. This simple example clearly illustrates that ALM protocols need to be carefully designed and reliability requires special consideration.

![Figure 4.1: Reliability is the problem.](image)

However, none of the previous work described in the following section has addressed the reliability problem.

4.2 Related Work

There is a large body of work on ALM since Chu et al. [2002]’s original paper, addressing various problems related to ALM. This section briefly reviews some of the related work and puts our work into perspective.
4.2.1 Overlay Multicast Tree Construction

In ALM, end systems act as routers and are responsible for data replication and forwarding, forming a logical overlay on top of the physical topology. This logical topology could be further divided along two planes according to their functionalities: one performing group membership maintenance called control plane; the other concerns about the actual data packet delivery termed data plane [Rong et al., 2006c; 2005]. Various overlay multicast tree construction algorithms can be roughly classified into 4 categories: tree-first, mesh-first, DHT-first, and implicit approaches [Birrer and Bustamante, 2007].

Tree-first Approach

In tree-first approach, participating peers directly build an overlay multicast dissemination tree by choosing their “parents” [Jannotti et al., 2000; Francis, 1999; Pendarakis et al., 2001].

Mesh-first Approach

On the other end of the spectrum, mesh-first approach builds a overlay multicast tree out of a partially connected mesh. The actual multicast tree can be explicitly built by running multicast routing algorithms, e.g., the reverse shortest path spanning trees algorithm [Deering and Cheriton, 1990]; or it can be implicitly pinpointed based on data availability [Moscibroda and Rejaie, 2007; Zhang et al., 2005].

DHT-first Approach

DHT has been widely used to build structured Peer-to-Peer (P2P) applications, and it has also been the underlying mechanisms of several P2P streaming applications [Castro et al., 2002; Rowstron and Druschel, 2001]. It organizes participating peers in such a way that every peer is tagged with a unique identifier, and peers having similar contents are grouped close to each other. In the meantime, messages are associated with keys, and they are routed to the node that is possibly closest to the associated keys. The actual mapping from message to key and the associated table look up is realized by maintaining a routing table at each peer: nodes in the same row, e.g., row \( r \) share the first \( r \) digits. The closeness of two nodes could be measured by the extent to which the prefix of their identifiers overlap. Scribe [Castro et al., 2002] is an illustrative example of this DHT-first approach, and it is built on top of Pastry [Rowstron and Druschel, 2001]. The multicast in Scribe is actually the union of Pastry routes.
Implicit Approach

The last category is the implicit approach: the overlay multicast tree is built as a by-product out of some optimization operations. Nice [Banerjee et al., 2002] is a typical example, where participating peers are organized into clusters based on their end-to-end delay. Each cluster is of the size between $d$ and $3d - 1$, where $d$ is the degree of the network. A leader, which is the peer having a minimal-maximum network distance to all other peers in the same cluster, is elected. Although a logical hierarchy is defined. However, the actual multicast delivery tree is not explicitly defined and is only implicitly constructed by the packet forwarding rules.

4.2.2 Resilient Overlay Multicast

The prominent distinction between native multicast and overlay multicast lies in where the routing functionalities are performed. In ALM, participating end systems take the responsibility; while routers handle routing-related tasks in native multicast. The biggest impact of this paradigm shift is on system performance, specifically, reliability. Consequently, much of the subsequent work has been devoted to reliable overlay multicast, and they can be broadly divided into two groups: redundancy-based and construction-oriented.

Redundancy-based Approach

Redundancy, temporal, spatial, or content-wise, has been widely used to provide reliability in native multicast [Towsley et al., 1997; Levine and Garcia-Luna-Aceves, 1998]. A typical usage of temporal redundancy is retransmission, a well-studied technique for protocols working on network layer, in which the sender keeps re-transmitting the current data packet until it receives an acknowledgement (ACK) from the receivers (negative acknowledge based or NACK works in the same spirit.) Spatial redundancy is exploited by sending the same packets along multi-paths, in the hope that failures of these paths are independent of each other. The last one could be called “content-redundancy-based”, e.g., Forward Error Control (FEC) or erasure-code based, where the content has certain redundancy, and the receiver does not need to receive the complete message [Floyd et al., 1997; Paul. et al., 1997; Yavatkar et al., 1995]. Nevertheless direct porting of these algorithms to ALM should be carefully designed. Otherwise, the explosion of ACK related messages or redundant messages will make the multicast delivery structure congested, causing undesirable service disruptions.

Silber et al. attempted to make P2P network reliable by augmenting the overlay structure. Links are added either randomly or according to predefined rules [Silber et al., 2004].
Nevertheless, it is not specifically designed for single-tree-based ALM systems and it has two shortcomings. First, the improved connectivity cannot solve the problem caused by node failure. Second, the augmented network ends up with a mesh, rendering another routing layer on top of this mesh.

Probabilistic Resilient Multicast [Banerjee et al., 2003] tried to utilize redundancy to achieve reliability. Similar to Silber et al. [2004]'s idea, random links are added between peers. But it takes one step further by allowing peers to randomly forward packets to others, in the hope that the potential packet loss will be masked pro-actively. Random forwarding may prevent random failure of peers. Nevertheless, this uncoordinated forwarding imposes large overhead on the overlay network, with the danger of saturating peers with low bandwidth and/or small buffer.

Building multiple trees is a commonly used approach to offer extra reliability, where several overlapping multicast trees are built, and only part of the streaming content is delivered along each tree. For example, Magellan [Birrer and Bustamante, 2005] delivered the video streaming packets over a forest instead of a single tree, where every peer contributes to at least one of the trees by acting as a “primary peer”, which is essentially a non-leaf node and serves other peers in this particular tree. However, how to partition the streaming content and the associated peers is a difficult question. Furthermore, the maintenance of these multiple trees and the coordination among active peers requires considerable amount of overhead.

Wong et al. proposed an algorithm to organize participating peers into subtrees for the purpose of failure recovery. These subtrees are constructed in such a way that the network distance between a particular peer and its recovery peers is small, aiming to offer fast recovery [Wong et al., 2004]. However, it failed to take bandwidth and other realistic parameters into consideration, limiting its applicability.

Construction-based Approach

Since end systems are inherently unstable, a natural way to make overlay multicast reliable is to pro-actively build a reliable overlay multicast delivery tree by minimizing the effect of inter-dependency among peers. For example, the so-called minimum depth algorithm [Sripanidkulchai et al., 2004; Padmanabhan et al., 2003; Birrer and Bustamante, 2006] tried to build a multicast tree with minimum depth with the aim that, upon node departure, only a small fraction of peers will be affected. However, its applicability strongly depends on the
upload bandwidth distribution of participating peers and the actual arrival pattern: building a minimum depth tree implicitly makes the assumption that most peers have abundant upload bandwidth. Therefore, a flat or fat tree can be built by letting non-leaf nodes to serve as many as children as possible. Furthermore, the actual arrival pattern of peers has a huge impact on it, and if peers with poor upload access bandwidth arrive first, it is difficult to build a minimum depth tree out of it.

Most importantly, this pro-active construction-based approach aims to solve the reliability-related problem once-and-for-all. This philosophy is very suspicious in a P2P context, as the overlay is highly dynamic and the churn rate is very high. Even if a perfectly reliable multicast tree can be built in the initial stage, it is subject to constant damages and the optimal structure quickly becomes irrelevant.

In summary, considering the pros and cons of various approaches, one would like to conclude that the overlay multicast tree should be optimized incrementally. Bearing this consideration in mind, we propose the lifetime-based approach which will be explained in the following section.

4.3 Proposed Approach

The proposed approach is described in detail in this section, together with an explanation of the rationale underlying the proposed approach.

4.3.1 Rationale Underlying the Proposed Approach

In order to build a reliable ALM system, it is worth examining the characteristics of single-tree-based multicast algorithms. The existing tree-based reliable multicast algorithms organize the receivers into a hierarchy according to their geographic proximities. However, the assumption of a relatively stable network does not hold anymore when it comes to ALM. So, the most important criterion while choosing the upstream nodes is the relative stability of the nodes. For example, a node with a residual lifetime of 60 minutes would be relatively stable enough for a node with a residual lifetime of 30 minutes. This clearly indicates that participating nodes’ lifetime is worth further investigation.

The proposed approach is based on the observation that peers’ lifetime follow a heavy-tailed distribution whose property differs greatly from the commonly used exponential distribution.
Heavy-tailed Distributed Service Time

It has been shown that the lifetime of participating peers in P2P networks follow a power-law distribution [Bustamante and Qiao, 2004] which can be mathematically expressed as:

\[ P\{X > T\} = T^{-\alpha} \]  

where \( X \) is a random variable, standing for the expected lifetime of the peer, and \( T \) represents time. Equation 4.1 is the complementary cumulative distribution function (CCDF). Random variables following a power-law distribution have some unique properties [Harchol-Balter, 2002]:

- A decreasing failure rate, which can be derived from Equation 4.1.

\[ \eta(T) = \alpha T^{-\alpha - 1} \]  

- As lifetime distribution is heavy-tailed, a small fraction of the peers dominate the majority of the lifetime, i.e., only few peers remain alive throughout the whole session.

Because it has “memory”, a conditional probability can be derived from Equation 4.1:

\[ P\{X > b|X' = a\} = \left(\frac{b}{a}\right)^{-\alpha} \]  

where \( X \) is the variable denoting the lifetime of the peer, and \( X' \) is simply its current “age”. Equation 4.3 shows that the remaining lifetime of a peer is strongly correlated to its “history”. A peer’s residual lifetime depends on many factors, such as, the quality of the streaming program, delay and bandwidth guarantees. Intuitively, if a user is satisfied with the service, it is most likely that the user will spend more time in ALM. So, it is clear that this heavy-tailed lifetime distribution can be exploited to tackle the reliability problem. Consequently, participating peers are organized into a logical hierarchy, which is different from the multicast tree structure, and their difference will be explained in the following section.

Logical Hierarchy

At any given time, every peer is mapped into a logical layer using the rank calculation procedure (presented shortly) with its “rank” adjusted dynamically. Peers choose their “parents”
according to this rank and opt to be attached to peers with higher rank, i.e., peers that are potentially more stable.

Figure 4.2 shows a simple example of this hierarchical structure that contains three layers. The multicast delivery structure, which is a tree in this case, is shown on the left, and the corresponding logical hierarchy is shown on the right. This logical hierarchy represents the logical relationship among peers, based on their relative stability. The multicast tree is the graphical representation of the physical connections among participating peers. In Figure 4.2, peer 1 acts as the multicast source, peers 2, 3, 4, 5, and 6 are end users that are currently receiving data from peer 1. The highest layer, layer 2, consists of the multicast source only (with a rank of 2), represents the most stable group. Layer 1, with peer 2 (with a rank of 1), is the second most stable group. Consequently, layer 0 is the most dynamic or the most unstable group. That is to say, peers with higher rank are potentially more stable.

Separating the logical hierarchy from the multicast tree facilitates construction of the multicast tree. In our proposed algorithm, reliability is the most important criterion when constructing the multicast tree. Then minimum spanning tree or shortest path tree based algorithms are employed to build the multicast tree, aiming to minimize the network delay or network usage. Therefore, a logical hierarchy is needed to reflect the relationship among all the peers, in terms of their relative stability. For example, in Figure 4.2, although peer 4 is closer to peer 3 as compared with peer 2, in terms of geographical proximity. Peer 3 still chooses peer 2 as its “parent” because of peer 2’s higher rank, given the assumption
that peer 2 is “older” than peer 4 in the current streaming session. This logical hierarchical structure, constructed and maintained using the distributed algorithm described in the next section, has the following properties:

- Each peer only belongs to a single layer (according to its rank) at any given time.
- Peer’s “rank” represents their relative stability, i.e., higher rank indicates higher stability.
- A newly joined peer is mapped into layer 0, since its stability is unknown.
- For a peer belonging to layer \(i\), it must have migrated through layer 0, layer 1,..., layer \(i-1\).
- The logical hierarchy has no more than \(\log_2 T\) layers, where \(T\) is the duration of the multicast session. For example, for a session of 1000 minutes or equivalent to 16 hours, the logical hierarchy consists of a maximum of 10 layers.

**Rank Calculation**

Every peer maintains a partial list, i.e., the peers with higher rank that may be contacted when its current parent fails. This list is constructed and updated using the gossip-based membership management algorithm [Rong et al., 2005; 2006c]. “Rank” is calculated according to peer’s “age” and is used for “migration”, which is the “parent switching process” that enables peers to join a potentially more stable group.

Time is divided into logarithmic sized bins for the sake of rank calculation. For example, time can be slotted into bin 1: 0 – 1; bin 2: 1 – 3; bin 3: 3 – 7; and so on. A peer belongs to bin \(i\), implying that it has a minimum “age” of \(d\):

\[
d = \sum_{k=0}^{i-1} 2^k = 2^i - 1
\]

In other words, if a peer’s lifetime reaches \(d\), it will be updated to a higher layer \(i\).

**Migration Through Layers**

Every peer calculates its rank periodically. If a peer reaches the age of \(d = 2^i - 1\), it will update its rank from \(i-1\) to \(i\), and make an attempt to migrate itself to a higher layer, namely layer \(i\), by attaching to a layer \(i+1\) peer.
The actual migration process can be described as “make before break switch”, which means setting up the new connection before disconnecting from the current one. The reason for doing so is that the attempt to join a prospective new “parent”, when the current parent is no longer available, may not be successful immediately. This avoids unnecessary service degradations although there are some duplicate packets for a short period of time. Figure 4.3 demonstrates a simple example of this “make before break switch” procedure. The top half illustrates that a new connection has been setup with the prospective new “parent” while the old one still remains functional. The bottom half depicts the new logical hierarchy after the completion of the switching process. Noticeably, peers 5 and 6 have left and peers 7 and...
8 are newcomers to the session. The newly joined peers will start from the lowest layer (i.e., layer 0), and this again explains why we separate the logical hierarchy from the multicast tree. By employing this logical hierarchy, most of the activities, i.e., peers’ joining and departing only occurs within the lower end of the logical hierarchy. In other words, relatively stable peers will eventually “climb up” the logical hierarchy and take more responsibilities. The less stable peers, corresponding to the “leaves” in the multicast tree, will have less children, and their departures will not affect many peers.
4.3.2 Detailed Algorithm Description

A detailed description of the proposed algorithm is presented in this section. In order to justify the algorithm, an assumption is made: the newly joined nodes will be provided with an initial set of nodes to which they may contact, via a well-known Rendezvous Point. The proposed algorithm consists of three building blocks: handling newly joined peers, logical hierarchy maintenance, and recovering from failures.

Handling Newly Joined Peers

The newly joined peers are mapped into the lower end of the logical hierarchy. They are provided with an initial list, by a bootstrap procedure, to join the current streaming session. The newcomer knows the existence of several existing on-tree peers from the initial list. It sends a `Join` request to each of them respectively. Upon receipt of the replies, the joining peer selects the best candidate based on network distance, bandwidth, or any other criterion. This is summarized in Algorithm 3, in which `dist()` is used to calculate the network distances between two nodes.

Logical Hierarchy Maintenance

Every peer is mapped to a certain layer in the logical hierarchy at any given time and it periodically makes an attempt to migrate to a higher layer. Intuitively, each peer is interested in the information about peers with higher “rank”, which may be the peer’s new “parent” once their current “parent” fails, or for migration purposes. This backup list is maintained at each peer, and it is constructed and updated by a gossip-based membership management algorithm [Rong et al., 2005; 2006c], which disseminates membership management related information in an epidemic fashion. Similar to the maintenance of a routing table, `Heartbeat` and `soft timer` are used to ensure the “freshness” of entries in the backup list. Algorithm 4 summaries this migration process, where `rank()` is used to calculate a peer’s rank.

Recovering from Failures

Peer can detect the departure of the current parent when it fails to receive either the `Heartbeat` message or the data packets. Once this happens, it initiates a `Rejoin` process, which is very similar to Algorithm 3. It can quickly resume receiving data once it finds the new “parent”.

### Algorithm 3: Handling Newly Joined Peers

1. Send a `Join` request to each of the existing on-tree peers from the initial list.
2. Upon receipt of the replies, select the best candidate based on network distance, bandwidth, or any other criterion.
3. Update the logical hierarchy with the newly joined peer and its parent.

### Algorithm 4: Logical Hierarchy Maintenance

1. Every peer periodically makes an attempt to migrate to a higher layer.
2. Maintain a backup list of peers with higher “rank”, which may be the peer’s new “parent” once their current “parent” fails, or for migration purposes.
3. Update the backup list by using a gossip-based membership management algorithm.
4. Use `Heartbeat` and `soft timer` to ensure the “freshness” of entries in the backup list.

### Algorithm 5: Recovering from Failures

1. Detect the departure of the current parent when it fails to receive either the `Heartbeat` message or the data packets.
2. Initiate a `Rejoin` process, which is very similar to Algorithm 3.
3. Resume receiving data once it finds the new “parent”.
CHAPTER 4. RESILIENT APPLICATION LAYER MULTICAST

Algorithm 3 Joining process

For a newly joined node \( n \)
Get initial list \( List \) using bootstrap
if \( i \in List \& i.rank = 1 \) then
    send Join to \( i \)
end if
Upon receipt reply from node \( i \)
if \( \text{dist}(n,i) < \text{min} \) then
    \( \text{min} = \text{dist}(n,i) \)
end if
Send Confirmation to peer \( k \) with the minimum \( \text{dist}(n,k) \)

Algorithm 4 Migration process

For node \( n \)
if \( \text{rank}(n) = n.rank + 1 \) then
    Get the potential parent \( List \)
    while \( i \in List \& n.rank = i.rank - 1 \) do
        if \( \text{dist}(i,j) < \text{min} \) then
            \( \text{min} = \text{dist}(i,j) \)
        end if
    end while
    send Join to peer \( k \) with the minimum \( \text{dist}(n,k) \)
end if

4.4 Analytical Results

An analytic modeling is presented in this section to mathematically analyze the performance of the existing algorithm and the proposed algorithm.

Network Size

Suppose that peers join an ALM session according to a Poisson distribution with an arrival rate \( \lambda \), and a service time following a power-law distribution. This ALM network can be modeled as a Poisson queue with infinite servers because all the peers will be served eventually, and the admission control related problem is investigated in Chapter 6.

Let \( M(t) \) represent the total number of nodes in the system that have arrived by time \( t \), and \( N(t) \) denote the number of nodes in the system at time \( t \). Obviously, \( N(t) \) is the conditioned form of \( M(t) \) [Ross, 1970].
CHAPTER 4. RESILIENT APPLICATION LAYER MULTICAST

\[ P\{N(t) = j\} = \sum_{n=0}^{\infty} P\{N(t) = j | M(t) = n\} e^{-\lambda t} \frac{(\lambda t)^n}{n!} \] (4.5)

Noticeably, the probability that a node arrives at time \(x\) still remains in the system at time \(t\) is \(1 - G(t - x)\). Therefore, given that \(M(t) = n\), the probability that an arbitrary peer still being present in the system at time \(t\) is

\[ p = \frac{1}{t} \int_{0}^{t} (1 - G(t - x))dx = \frac{1}{t} \int_{0}^{t} (1 - G(x))dx \] (4.6)

It is easy to see that

\[ P\{N(t) = j | M(t) = n\} = \binom{n}{j} p^j (1 - p)^{n-j} e^{-\lambda t} \frac{(\lambda t)^n}{n!} \] (4.7)

Combining Equation 4.6 with Equation 4.7,

\[ P\{N(t) = j\} = e^{-\lambda tp} \frac{(\lambda tp)^j}{j!} \] (4.8)

In other words, \(N(t)\) has a Poisson distribution with a mean of \(\lambda \int_{0}^{t} (1 - G(x))dx\). Recall that peers’ lifetime follows a power-law distribution, as expressed in Equation 4.1, Equation 4.8 can now be rewritten as:

\[ E[N(t)] = \lambda \int_{0}^{t} (1 - G(x))dx \] (4.9)

Recall that Equation 4.1 is the complementary cumulative form of the lifetime distribution, and its corresponding cumulative distribution function (CDF) is:

\[ P\{X \leq T\} = 1 - T^{-\alpha} \] (4.10)

when \(\alpha = 1\), Equation 4.9 becomes:

\[ E[N(t)] = \lambda [ln(t) - 1] \] (4.11)

which means that the network size will increase with the multicast session length, and it will be used as an important simulation parameter during simulation.
CHAPTER 4. RESILIENT APPLICATION LAYER MULTICAST

Logical Hierarchy Analysis

The logical hierarchy is an important building block in our algorithm and it deserves further investigation. As stated in the previous section, peers join an ALM session according to a poisson distribution with an arrival rate \( \lambda \), and the service time follows a power-law distribution. This can be modeled using a \( M/G/\infty \) queue with an arrival rate \( \lambda \) and the service time in the form of cumulative distribution (CDF) \( G \). The proposed algorithm classifies peers according to their lifetime. As described in Section 3.2.1, time is divided into logarithmic sized bins and a peer belongs to level \( i \), if its current lifetime falls in the interval \((b_{i-1}, b_i]\).

Peers’ service time is independent and identically distributed (i.i.d.) and the binning process splits the single \( M/G/\infty \) queue into \( l \) independent \( M/G/\infty \) queues with independent Poisson arrival rates and service times. The arrival rate to the \( i^{th} \) bin, corresponding to the \( i^{th} \) queue is:

\[
\lambda_i = \begin{cases} 
\lambda & i = 1 \\
\lambda_i = \lambda [1 - G(b_{i-1})] & i > 1
\end{cases}
\]

where \( G(x) \) is the service time distribution for the corresponding layer and it is:

\[
G_i(x) = \begin{cases} 
0 & x \leq b_{i-1} \\
\frac{G(x)}{G(b_i) - G(b_{i-1})} & b_{i-1} \leq x \leq b_i \\
1 & x > b_i
\end{cases}
\]

Since the service time follows a Pareto distribution, then the arrival rate to each layer \( i \) also follows a Pareto distribution. According to Equation 4.9, the average number of peers in queue \( i \), corresponding to bin \( i \) or layer \( i \) is:

\[
E[N_i(t)] = \lambda [1 - G(b_{i-1})] \int_{b_{i-1}}^{b_i} \left(1 - \frac{G(x)}{G(b_i) - G(b_{i-1})}\right) dx \tag{4.12}
\]

It is difficult to analyze the \( M/G/\infty \) queue directly. Fortunately, Little’s formula applies to any kind of queueing systems:

\[
N_i = \lambda_i W_i \tag{4.13}
\]

where \( N_i \) is the average number in the \( i^{th} \) queue, \( \lambda_i \) is the arrival rate for queue \( i \), and \( W_i \) is the average time peers spend in the queue. So the question is how to find \( W_i \). \( W_i \) consists of two parts: the first part contains the peers whose lifetime fall into bin \( i \) and bin \( i \) is the final...
destination for them, and they are denoted as \( w_{in} \); the second part consists of peers whose lifetime are longer than the size of bin \( i \) and they will continue their journey afterwards, and they are denoted as \( w_{out} \). Then \( W_i \) can be expressed as:

\[
W_i = \left[ \frac{G(b_i) - G(b_{i-1})}{1 - G(b_{i-1})} \right] E[w_{in}] + \left[ \frac{1 - G(b_i) - G(b_{i-1})}{1 - G(b_{i-1})} \right] E[w_{out}] \tag{4.14}
\]

\[
E[w] = \begin{cases} 
\int_{b_{i-1}}^{b_i} x dG(x) = ln(b_i) - ln(b_{i-1}) = ln 2 & \text{for } w_{in} \\
\lambda^i & \text{for } w_{out}
\end{cases}
\]

when \( \lambda \gg 1 \), we have \( N_i = 3\lambda \), which means there are enough peers in layer \( i \) to support the peers in layer \( i - 1 \).

Reliability Analysis of the Multicast Tree

The generation of the multicast tree can be modeled as a discrete-time Markov chain, or a branching process. Following the conventional terminology, nodes are classified into different generations [Feller, 1968].

Let \( X_n \) represents the size of the \( n^{th} \) generation, having a mean of \( \gamma \) and an variance of \( \sigma^2 \). For simplicity, assuming that \( E[X_0] = 1 \), and a simple recursive relationship exists:

\[
E[X_n] = \gamma E[X_{n-1}] \tag{4.15}
\]

\[
Var[X_n] = \sigma^2 E[X_{n-1}] + \gamma^2 Var[X_{n-1}] \tag{4.16}
\]

Since \( E[X_0] = 1 \), Equation 4.15 and Equation 4.16 can be solved:

\[
E[X_n] = \gamma^n \tag{4.17}
\]

\[
Var[X_n] = \begin{cases} 
\frac{n\sigma^2}{\gamma - 1} & \text{if } \gamma = 1 \\
\sigma^2 \gamma^{n-1} \frac{\gamma^n - 1}{\gamma - 1} & \text{if } \gamma \neq 1
\end{cases}
\]

when \( \gamma > 1 \), the branching process will continue and has a total number of

\[
\sum E[X_n] = \frac{\gamma^{n+1} - 1}{\gamma - 1} \tag{4.18}
\]

Denoting the average depth of the multicast tree as \( E[d] \) and solving Equation 4.17, we have:
\[ E[d] = \frac{\ln\{(\gamma - 1)E[x_n] + 1\}}{\ln\gamma - 1} - 1 \]  

(4.19)

Taking the characteristics of ALM into consideration, stability of the multicast tree is a good index for reliability, i.e., peers’ perceived QoS strongly depend on the stability of the multicast tree. Consider a single branch of the tree rooted at node 2, which is the branch on the left hand side of Figure 4.2. Each node is associated with a probability \( p_i \) that denotes the probability that it will survive the next time interval \( \Delta t \), and it can be expressed as:

\[
p_i = \begin{cases} 
  p_1 = P\{t_i > \Delta t\} \\
  p_i = P\{t_i > \Delta t|t_1 > \Delta t, t_2 > \Delta t...t_{i-1} > \Delta t\}
\end{cases}
\]

where \( p_1 \) represents the survival probability of the 1\(^{st} \) generation, and \( p_i \) of the \( i^{th} \) generation.

Let \( D_i = \sum_{j>i+1} E[X_j] \) represents the total number of descents of node \( i \), and it is easy to see that \( D_i \) is a monotonically decreasing function, i.e., \( D_1 > D_2...D_{n-1} > D_n \). Assuming that the survival probabilities are i.i.d., and the stability calculation can be formulated as a “perfect match” problem, which maps the probability \( p_i \) to \( D_i \), and the objective function can now be formulated as:

\[
\max \sum_{j=1}^{n} p_j D_j 
\]

(4.20)

Given that \( D_i \) is a monotonically decreasing function, it is well known that only when \( p_i \) is also a monotonically decreasing function, Equation 4.20 is maximized. This confirms the intuitive idea underlying our algorithm.

### 4.5 Experimental results

Extensive simulations have been carried out to validate the proposed algorithm. Since there is no previous work that completely resembles our work, several previous work [Silber et al., 2004; Birrer and Bustamante, 2005] has been combined as a benchmark algorithm. We chose them because of their resemblance with the proposed approach in certain key areas, such as, failure recovery and reliability enhancement.
4.5.1 Simulation Setup

A discrete event simulator was developed for simulation purposes. Furthermore, in order to simulate a realistic and meaningful network, 600000 nodes, representing 600000 peers, were generated using the GT-ITM topology generator [Zegura et al., 1996]. Peers are randomly distributed among the generated topology. Peers join the on-going session following a Poisson distribution and their lifetime are drawn independently and identically from a Pareto distribution. This heavy-tailed lifetime distribution has been reported by several researchers [Bustamante and Qiao, 2004].

A 95% confidence level is achieved by using Akaroa [aka] to control the duration of the simulation at runtime. That is to say, the simulation stops as directed by Akaroa [aka] when enough observations have been collected.

4.5.2 Metrics of Interest

To facilitate our experimental comparison, we need to measure the performance of both the proposed algorithm and the benchmark algorithm. Several performance metrics are defined for this purpose.

Quality of Service

Assume there is a Constant-Bit-Rate (CBR) traffic source emitting packets at a constant data rate, which is the playback rate $S$. Ideally, the amount of any peer’s received portion of the video packets should be proportional to its lifetime. Therefore, the Quality-of-Service (QoS) perceived by any peer can be reflected by the ratio of $\frac{\text{Amount of received video packet}}{S \times \text{lifetime}}$.

Frequency of update

The migration operation described in Section 4.3.1 involves switching parent. Although the actual handoff process is “make-before-break”, i.e., the old connection is torn down only after the new connection is ready to use, frequent invocation of this update is still not desirable. Therefore, the average frequency of update is a very important parameter that needs to be investigated.
CHAPTER 4. RESILIENT APPLICATION LAYER MULTICAST

Frequency of Rejoin

In ALM, peers join and depart the on-going session at will and without any notice, e.g., users may shut down or quit the streaming session at any time. The proposed approach tries to reduce peers’ frequency of rejoin by organizing peers according to their relative stabilities. However, even this pro-active approach can not completely prevent service disruption for all the peers. Consequently, the frequency of rejoin would be a very interesting parameter to measure the effectiveness of the proposed approach.

Latency penalty

The flexility of ALM is gained at the price of a performance penalty. Data packets are relayed by end systems rather than routers, leading to a longer end-to-end latency. On the other hand, video streaming has a stringent requirement of end-to-end delay. Therefore, special consideration needs to be taken to meet the required end-to-end delay. This latency is captured and presented by using hop counts in all the simulations.

4.5.3 Simulation Results

This section gives a detailed description of the simulation results, clearly demonstrating the effectiveness of the proposed approach.

The Proposed Algorithm is Effective

Figure 4.5 compares the proposed approach with the benchmark approach, in terms of peers’ perceived QoS. It clearly shows that the proposed algorithm is very effective as peers’ perceived QoS remains very stable and over 90% most of the time. On the contrary, peers experience a very poor QoS most of the time in the benchmark counterpart. The success of the proposed approach is due to its reliability-oriented incremental optimization: peers are organized into a hierarchy according to their relative stability, and peers periodically update their ranks and migrate to a higher layer by attaching to potentially more stable peers. The benefits of this reliability-oriented arrangement are two-fold: first, this periodical adjustment essentially “push” all the reliable peers to the top end of the overlay multicast tree, leaving less reliable nodes at the bottom end of the tree, i.e., most of the dynamic activities are confined to the bottom end of the tree and the departure of these nodes will not cause a severe service disruption for their downstream nodes. Second, nodes are placed in the hierarchy
in an inter-leaving way, e.g., a peer having a residual lifetime of 40 minutes is a desirable “parent candidate” for another peer who only wants to participating in the session for 20 minutes, i.e., peers’ contribution are maximized at various stage of their lifetime (reflected and captured by their “rank”). The first benefit is clearly shown through the comparison of the proposed approach with the benchmark algorithm. The second effect is best illustrated by the observation that a peers’ perceived QoS increases with the length of the streaming session in our algorithm. We observe that as with the increase of the session length, there are more peers in the system and the chances of interleaving between peers are also increased, leading to a better QoS. In a word, this reliability-oriented approach is effective, in terms of reliability enhancement.

Why the Proposed Algorithm is Working

Most importantly, the reliability improvement is not achieved at a price of high network overhead. Figure 4.6 shows that for the compared algorithm, the number of service disruptions grows linearly with the session length, and the explanation is that the dominating long-lived peers choose their parents based on network distance rather than stability, triggering frequent rejoin actions and a increasing number of service disruptions. The longer
they live, the larger is the number of service disruptions. On the other hand, our algorithm uses a logical hierarchy with a maximum of $\log_2 T$ layers, where $T$ is the session length, to dynamically improve the stability of the multicast tree, i.e., the frequency of update only grows logarithmically with the session length. This is the reason why the service disruption frequency and migration frequency, or even the sum of them, remain relatively small over a long session.

![Graph showing service disruption comparison.](attachment:image.png)

**Figure 4.6: Service disruption comparison.**

**Latency is the Price**

Everything has its pros and cons, so does our algorithm. The success in reliability enhancement is gained at the price of a longer end-to-end latency. Figure 4.7 gives a simple example of the latency problem, where node $S$ is the data source, and nodes 1, 2, 3, 4 and 5 are participating peers. As demonstrated in the figure, although peer 4 is much closer to the
data source, peer 5 chooses peer 3 as its “parent” due to peer 3’s high stability, leading to a 3 hops path from data source to peer 5, compared to a 2 hops path if peer 4 were selected as peer 5’s “parent”.

![Figure 4.7: Latency problem of the proposed algorithm.](image)

Figure 4.7 demonstrates this trend. For the benchmark algorithm, since random links are added over time [Silber et al., 2004], path diversity can be better exploited with the increase of the session length. The network size grows with session length, as explained in Section 4.4, leading to the decrease of the average end-to-end delay. On the other hand, the average packet delay in our algorithm grows very quickly with the session length since peers choose their parents primarily based on reliability rather than geographical proximity. When the overlay network becomes more stable, the average logical level $L = \frac{\ln((\gamma-1)E[X_\alpha]+1)}{\ln\gamma-1} - 1$ increases as well, leading to longer paths.

Figure 4.9 depicts the relationship of another very important metric, the average number of times a data packet has been forwarded by other peers, with the increase of session length. It roughly shares the same trend with the average end-to-end delay that is shown in Figure 4.8. The prominent feature of ALM is that the data packets are replicated and forwarded by end systems rather than routers, so the average number of times a data packet has been forwarded by other peers is an interesting aspect to be investigated. In the benchmark algorithm, when the network turns out to be more stable, more peers may get data from other peers nearby,
CHAPTER 4. RESILIENT APPLICATION LAYER MULTICAST

Figure 4.8: Latency of data packets comparison.

Figure 4.9: Forwarding comparison.
CHAPTER 4. RESILIENT APPLICATION LAYER MULTICAST

leading to a decrease of the number of forwarding. On the contrary, in our algorithm, when the network becomes more stable, the average level of hierarchy increases as well, i.e., more peers are getting data packets from other peers that are potentially more stable.

**Can We Do Better on Latency?**

Longer end-to-end delay apparently is not desirable for real-time video streaming, but unfortunately it is the price paid for reliability enhancement. The natural question is: “can we do better, i.e., can we improve reliability without sacrificing too much on end-to-end latency?”

The latency problem arises from the fact that peers are organized solely according to their stability, and network latency has not been taken into consideration. Integrating network latency into the proposed algorithm might lead to the solution to the network latency problem.

Bearing this idea in mind, a preliminary investigation is carried out, and the results are presented from Figure 4.10 to Figure 4.13. The results are encouraging, indicating it is possible to tackle the latency problem. In all these four figures, the x-axis represents the cumulative percentage of peers, and y-axis is the average number hops of data packets. Assume peers’ upload bandwidth is drawn independently and identically from a heavy-tailed distribution with a maximum of $x$, and we vary this maximum achievable upload bandwidth $x$ to conduct the simulation. In reality, low-end video can be broadcasted at around 300 Kbps. If we normalize peers’ upload bandwidth by this rate, the common high-bandwidth users with a 10 Mbps upload bandwidth will have a maximum achievable upload bandwidth of 30 units. It is worth pointing out that due to the heavy-tailed nature, most peers still have a small upload bandwidth, and only a handful of them have the maximum achievable upload bandwidth of $x$.

A closer examination of Figure 4.10 to Figure 4.13 draws two observations: first, it is possible to increase reliability without sacrificing too much on end-to-end latency; second, the improvement over end-to-end delay strongly depends on participating peers’ average upload bandwidth distribution.

4.6 Conclusion

The reliability issue, as an important research question in ALM, is identified and investigated in this chapter. An adaptive algorithm is proposed to enhance reliability for application layer multicast. It makes use of the heavy-tailed lifetime distribution, building the hierarchical
CHAPTER 4. RESILIENT APPLICATION LAYER MULTICAST

Average No. of hops

Percentage of peers (%)

Figure 4.10: Average bandwidth 30.

Average No. of hops

Percentage of peers (%)

Figure 4.11: Average bandwidth 25.

(March 23, 2008)
Average No. of hops

Figure 4.12: Average bandwidth 20.

Percentage of peers (%)

Figure 4.13: Average bandwidth 15.
CHAPTER 4. RESILIENT APPLICATION LAYER MULTICAST

multicast tree dynamically. Peers are organized into a logical hierarchy according to their relative stability (captured by their “rank”). Peers periodically check and update their ranks by attaching to other peers in higher layer. Eventually, more potentially stable peers are “pushed” to the top end of the overlay multicast tree and most of the dynamics are confined to the bottom end of the tree, leading to reliability improvement.

Nevertheless, reliability, measured as average QoS perceived by each peer, is improved at a small penalty of longer delay. However, as long as the delay is consistent, the gain in terms of reliability outperforms the penalty. It has some other attractive features, such as simplicity of implementation, fast recovery and self-organization. It has been validated by simulation experiments. The simulation results shows that it improves reliability by a minimum of 20%, as shown in Figure 4.5.

However, longer end-to-end delay is not desirable for real-time applications. In addition, we have not taken other properties of the peer into account, such as, connection bandwidth, computational power and storage capacity. Extending the proposed algorithm to accommodate these properties and making our algorithm QoS-aware are the topics of next chapter.
Chapter 5

QoS-aware Reliable Application Layer Multicast

Reliability of ALM has been addressed in the previous chapter. Both the mathematical analysis and simulation results show that reliability (reflected by peers’ perceived QoS) can be greatly improved. However, this reliability gain is obtained at the expense of a longer end-to-end delay, which is not desirable for real-time video streaming. This tradeoff must be carefully engineered for the pervasive deployment of video streaming using ALM. Fortunately, preliminary simulations conducted in the previous chapter suggest that it is possible to strive for a better balance between reliability enhancement and end-to-end delay.

In order to have a thorough investigation, reliability of ALM is isolated from other important parameters, e.g., peers’ access bandwidth and the end-to-end delay between peers in the previous chapter. Furthermore, the heterogeneity of peers’ access bandwidth, computational power, storage capacity, and other parameters further complicates the design of realistic ALM applications.

In a word, making our algorithm QoS-aware is crucial to the success of video streaming using ALM. This chapter investigates along this research line: two parameters, bandwidth and delay, are singled out due to their importance to real-time applications. The QoS-aware aspect is integrated into our previous work by deriving a new parent selection algorithm, which modifies Dijkstra’s algorithm to accommodate both bandwidth and delay and reduce the end-to-end delay. Experimental results are promising, indicating that a better balance can be found: reliability should not come at the expense of end-to-end delay.
5.1 Motivation and Problem Formulation

Previous research on ALM mainly focus on constructing a multicast over the overlay mesh [Jannotti et al., 2000; Francis, 1999; Pendarakis et al., 2001; Deering and Cheriton, 1990; Moscibroda and Rejaie, 2007; Zhang et al., 2005; Castro et al., 2002; Rowstron and Druschel, 2001]. Not too much work has been devoted to the Quality-of-Service (QoS) aspect, in particular reliability related aspects of ALM. Our previous work [Rong et al., 2006a:b] took an initial step towards this direction. However, reliability is gained at the price of a longer end-to-end delay. The preliminary results presented in the previous chapter suggest that reliability could be improved without violation of the delay bound. This section identifies and pinpoints the problem. For clarity, Table 5.1 lists all the notations encountered in the formulation of the problem.

Table 5.1: Notations.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( L )</td>
<td>number of hierarchy</td>
</tr>
<tr>
<td>( X_i )</td>
<td>number of peers in layer ( i )</td>
</tr>
<tr>
<td>( B_i^U )</td>
<td>aggregate upload bandwidth of layer ( i ) peers</td>
</tr>
<tr>
<td>( B_i^D )</td>
<td>aggregate download bandwidth of layer ( i ) peers</td>
</tr>
<tr>
<td>( b_i^u )</td>
<td>upload bandwidth of peer ( i )</td>
</tr>
<tr>
<td>( b_i^d )</td>
<td>download bandwidth of peer ( i )</td>
</tr>
<tr>
<td>( \beta_i )</td>
<td>buffer size of peer ( i )</td>
</tr>
<tr>
<td>( \mu_i )</td>
<td>playback rate of peer ( i )</td>
</tr>
<tr>
<td>( S )</td>
<td>CBR rate</td>
</tr>
<tr>
<td>( c_i )</td>
<td>( c_i = \min{b_i^d, \mu_i, S} )</td>
</tr>
<tr>
<td>( d_i )</td>
<td>delay of peer ( i )</td>
</tr>
<tr>
<td>( \tau_i )</td>
<td>delay jitter of peer ( i )</td>
</tr>
</tbody>
</table>
CHAPTER 5. QOS-AWARE RELIABLE APPLICATION LAYER MULTICAST

5.1.1 Network Model

Before diving into the details of the problem, it is worth spending some time understanding the characteristics of single-tree-based ALM systems, in particular the ALM based on our previous work [Rong et al., 2006a;b], as this will help us to gain a better understanding of the problem per se.

In a single-tree-based ALM, participating users form a logical layer on top of the network layer. This virtual network can be modeled as a directed graph \( G = (V, E) \), where \( V \) is the set of participating nodes and \( E \) is the set of edges. This virtual network keeps evolving with the dynamics of participants as participating users may join and leave at will. In order to account for this membership dynamics, the network model can be refined as \( G_k = (V_k, E_k) \), where \( G_k \) represents the network after the \( k^{th} \) event; the \( k^{th} \) event could be the addition or removal of a node. For ease of explanation, let \( |V_k| \) denote the number of participants and \( |E_k| \) the number of edges.

In \( G_k \), each node \( v_i \) has a lifetime \( E[L_i] \), following a common distribution \( F(x) \). In addition, each participant has its own QoS requirements expressed by a constraint vector \( \overrightarrow{q} \); each vector consists of \( m \) constraints \( q_i, i = 1, ..., m \). Each link is associated with a link weight vector \( \overrightarrow{w} \) containing \( m \) types of weight \( w_i \), for \( i = 1, ..., m \).

A multicast tree is a tree spanning all participants, rooted at the streaming source node \( S \). This multicast tree could be viewed as a set of paths connecting \( s \) to each of the participants \( v_i \). In a single-tree-based P2P media streaming system, participating peers are dynamically organized into a hierarchy of \( L \) layers: layer 0 consists of the CBR source only, layer 1 contains the peers that are attached to the CBR source directly, and layer 2 peers receive streaming data from layer 1 peers and forward the received streaming data to layer 3 peers in turn, and so on. Figure 5.1 depicts a simple example, in which node 0 belongs to layer 0, nodes 2 and 7 are layer 1 nodes, nodes 3, 4, and 8 belong to layer 2, and layer 3 contains nodes 5 and 6.

The generation of the multicast tree can be modeled as a discrete-time Markov chain, or a branching process [Feller, 1968]. Following the abused terminology, nodes are classified into different generations, corresponding to peers belonging to different layers. Let \( X_n \) represents the size of the \( n^{th} \) generation, it has a mean of \( \gamma \) and an variance of \( \sigma^2 \). For example, in Figure 5.1, nodes 2 and 7 are layer 1 nodes and they constitute the first generation, so \( X_1 = 2 \) in this case.
Simple recursive relations exist among $X_n$:

$$E[X_n] = \gamma E[X_{n-1}] \quad (5.1)$$

$$Var[X_n] = \sigma^2 E[X_{n-1}] + \gamma^2 Var[X_{n-1}] \quad (5.2)$$

Since the data source is the only node in the 0th generation, that is to say that $E[X_0] = 1$. Given this initial condition, Equation 5.1 and Equation 5.2 can be solved:

$$E[X_n] = \gamma^n \quad (5.3)$$

$$Var[X_n] = \begin{cases} n\sigma^2 & \text{if } \gamma = 1 \\ \sigma^2 \gamma^{n-1} \frac{\gamma^n - 1}{\gamma - 1} & \text{if } \gamma \neq 1 \end{cases}$$

the branching process will continue if and only if when $\gamma > 1$, and has a total number of

$$\sum E[X_n] = \frac{\gamma^{n+1} - 1}{\gamma - 1} \quad (5.4)$$

Denoting the average depth of the multicast as $E[d]$ and solving Equation 5.3, we have:

$$E[d] = \frac{ln\{(\gamma - 1)E[X_n] + 1\}}{ln\gamma - 1} - 1 \quad (5.5)$$
where $\gamma$ represents the fan-out of this ALM tree: for a given generation $n$, increasing $\gamma$ will increase the total number of peers that can be accommodated. On the other hand, for a given total number of peers, increasing $\gamma$ results in a decreased average depth $E[d]$. In reality, $\gamma$ strongly depends on the average upload bandwidth of participating peers: the larger the average upload bandwidth, the bigger is $\gamma$.

### 5.1.2 Design Space

Quality-of-Service (QoS) normally refers to the capability of a network to provide better service to selected network traffic, and a better service can be quantified using the requirements of cost, bandwidth, delay, delay jitter, and reliability, etc. By QoS-aware ALM, we mean in our scheme, QoS has been explicitly addressed. This section identifies some of the key QoS aspects that needed to be taken care of.

**Delay or Delay Jitter, That is The Question**

A basic media streaming system works as follows: given a CBR source, the streaming server disseminates the streaming data at a rate of $S$ bps; the client can buffer up to $\beta$ seconds of streaming media, and is be able to play the streaming media at a playback rate of $\mu$ bps. Applying this simple model to P2P environment implies that the maximum accumulated delay allowed is $\beta/\mu$ seconds, and the detailed exploration is as follows: a layer $i$ peer $p_i$ just joins the streaming session and sends a media request to its parent, a layer $(i-1)$ peer $p_{i-1}$.

Assume that the network distance between a layer $(i-1)$ peer and a layer $i$ peer is denoted as $d_i$ and the corresponding delay jitter is $\tau_i$, and the accumulated delay is $\sum_{k=1}^i d_k + \sum_{k=1}^i \tau_k$, so the initial delay is $2d_i + \beta/c_i$, where $c_i$ is the minimal one of the upload bandwidth of peer $p_{i-1}$, the download bandwidth of peer $p_i$, and the CBR rate $S$, i.e., $c_i = \min(b_{i-1}^u, b_i^d, S)$. Therefore, peer $p_i$ is always lagging behind the CBR source:

$$\sum_{k=1}^i d_k + 2d_i + \beta/c_i + \sum_{k=1}^i \tau_k, \quad \forall i,$$

where $\sum_{k=1}^i d_k + 2d_i + \beta/c_i$ is predictable once the peer is attached to an existing on-tree peer, and it could be easily dealt with by changing peer $p_i$’s start playback time. However, the latter part $\sum_{k=1}^i \tau_k$, can change dynamically and may have a great impact on the perceived experience of peer $p_i$:

1. Case of $\sum_{k=1}^i \tau_k < \beta/\mu$
CHAPTER 5. QOS-AWARE RELIABLE APPLICATION LAYER MULTICAST

In this case, the delay jitter along the path will not have any impact on user experience. Nevertheless, it is important to make sure that the accumulated delay jitter after the migration operation never exceeds the threshold of $\beta/\mu$.

2. Case of $\sum_{k=1}^{i} \tau_k \geq \beta/\mu$

Let $c_i$ as defined before:

- If $\sum_{k=1}^{i} \tau_k < \frac{\beta}{B_i - \mu}$
  This implies that, if the achievable downloading rate of peer $p_i$ is greater than its playback rate, this could mask the fluctuation of network delay along the path.

- If $\sum_{k=1}^{i} \tau_k \geq \frac{\beta}{B_i - \mu}$
  It implies the degradation of peer $p_i$, and if $c_i > \mu$, peer $p_i$ could pre-fetch some of the video packets to mask the fluctuation; otherwise, there is no way to meet the playback deadline.

On the other hand, only two-way real-time applications need stringent delay requirement and one-way streaming on the Internet can tolerate up to a ten to fifteen seconds of delay. Therefore, delay jitter rather than delay itself needs special consideration when constructing the multicast delivery tree.

Reliability

Taking the characteristics of ALM into consideration, stability of the multicast tree is a good index for reliability. Consider a single branch of the tree rooted at node 2, which is the branch on the left hand side of Figure 5.1. Each node is assigned a probability $p_i$, denoting the probability that it will survive for the next time interval $\Delta t$, and it can be expressed as:

$$p_i = \begin{cases} p_1 = P\{t_i > \Delta t\} \\ p_i = P\{t_i > \Delta t | t_1 > \Delta t, t_2 > \Delta t...t_{i-1} > \Delta t\} \end{cases}$$

where $p_1$ represents the survival probability of the 1st generation, and $p_i$ of the $i^{th}$ generation. Let $D_i = \sum_{j>i+1} E[X_j]$ represents the total number of descents of node $i$, and it is easy to see that $D_i$ is a monotonically decreasing function, i.e., $D_1 > D_2...D_{n-1} > D_n$. Assuming that the survival probabilities is i.i.d., and the stability calculation can be formulated as a “perfect match” problem, which maps the probability $p_i$ to $D_i$, so the objective function is:
Given that $D_i$ is a monotonically decreasing function, it is well known that only when $p_i$ is also a monotonically decreasing function, Equation 5.7 is maximized.

It is worth noting that a peer, say peer $i$, its reliability $p_i = \prod_{k=0}^{i-1} p_k$, i.e., its reliability strongly depends on all the peers along the path from the stream source to its immediate parent. Since the survival probability $p_k \leq 1$, it is obvious that reducing the average multicast depth could increase the average survival probability, and this could be achieved by increasing the average upload bandwidth as demonstrated in Section 5.1.1.

**Problem Formulation**

*Multiple Constrained Multiple Criteria Multicast (MCMCM): given $G_k$, find a multicast tree of maximum reliability:*

$$\text{Max } \sum_{i=1}^{n} p_i D_i$$

(5.7)

while the total cost is minimized:

$$\text{Max } \sum_{i=1}^{V_k} r_i$$

(5.8)

subject to all the constraints, i.e., for each path $P(s,v_i)$ connecting $s$ and $v_i$, $i = 1, ..., |V_k|:

$$\sum_{e \in P(s,v_i)} w_i(e) \leq q_i, \text{ for } i = 1, ..., m$$

(5.10)

where $\sum_{e \in P(s,v_i)} w_i(e)$ is the sum of additive weight $w_i$ over all the edges of $P(s,v_i)$.

$$\min_{e \in P(s,v_i)} w_i(e) \geq q_i, \text{ for } i = 1, ..., m$$

(5.11)

where $\min_{e \in P(s,v_i)} w_i(e)$ refers to the bottleneck weight $w_i$ over all the edges of $P(s,v_i)$.

**5.1.3 Hardness of The Problem**

The aforementioned multiple constrained multicast problem is essentially a multiple constrained routing problem, which is well-known NP-complete [Wang and Crowcroft, 2002].
i.e., its exact solution cannot be found in polynomial time.

Facing a NP-complete problem, two alternative approaches exist for finding sub-optimal answers: sophisticated approximation algorithms or heuristic solutions. An $\epsilon$ approximation algorithm guarantees to find a solution within $1 + \epsilon$ of the optimal solution, at the expense of its running time growing linearly with $1/\epsilon$. Generally speaking, approximation are complicated to design and implement. On the other end of the design spectrum, heuristic-based algorithms are much simpler in terms of design and implementation. However, there is no guarantee of solution quality.

Decision has to be made between a sophisticated approximation scheme and a much simpler heuristic algorithm. Several aspects need to be considered to aid decision making:

- The overlay network formed by participating users is highly dynamic, and most peers are short-lived [Sripanidkulchai, Ganjam, Maggs, and Zhang, 2004]. The question is whether it is worth performing extensive computation to find an optimal solution through complete search of the solution space under such a high churn rate: by the time an optimal solution is found, the peers involved may already leave the session, not to mention the prohibitively large search space, sometimes it takes several hours to do a complete search [Garey and Johnson, 1990].

- Exercising computationally intensive algorithms may overload the end systems, which are normally personal computers that are not equipped with powerful CPUs; furthermore, users may run other applications on the same end system, i.e., the available computational resource could be limited. Overloading the end systems may further decrease the service capacity of the network, rejecting more requesting peers: the overloaded users may shut down the computer or leave the on-going streaming session.

- Long setup delay will be experienced by some users since they are geographically distributed, and if they are rejected by their first choice, they have to turn to other users or decide not to join the streaming session.

Taking all these considerations into account, a greedy-based algorithm, derived from Dijkstra’s shortest path algorithm, is proposed and will be presented in Section 5.3.

5.2 Related Work

This section briefly reviews some of the related work, in particular, previous work on QoS-aware multicast routing, as our work is ultimately concerned with constructing a overlay
multicast delivery tree. Various work on multicast routing can be classified into the following categories:

**Shortest Path Tree**

The simplest way to construct a multicast tree is to view a multicast tree as a union of the shortest paths from the source to each individual receiver. The classic Bellman-Ford and Dijkstra algorithms belong to this category [Cormen et al., 1990]. Both of them run in polynomial time and produce the exact solution to the problem. Whether the link metric is hop count or delay, both produce a least-hop tree or least-delay tree. Due to their simplicity and elegance, our algorithm is based on Dijkstra’s algorithm to approximate the optimal multicast tree.

**Minimum Spanning Tree**

The shortest-path-tree based algorithms optimize the individual delay from each receiver’s point view, but the sum of them, i.e., the total delay, is not optimal. Minimum-spanning-tree based schemes can remedy this shortcoming. An equivalently elegant algorithm is presented by Prim [Cormen et al., 1990], and a distributed algorithm has also been given by Gallager et al. [Gallager et al., 1983]. Prim’s and Dijkstra’s are both greedy algorithms in the sense that in each step choices are made in a greedy way: they always add the least cost node to the tree.

**Steiner Tree**

If the total cost incurred is the primary interest, Steiner-tree-based algorithms is the answer; however, it is NP-complete [Gilbert and Pollak, 1968]. Various approximations and heuristic algorithms have been reported.

Most multicast routing algorithms are based on the aforementioned ideas, some of them being singled out because of their resemblance to our algorithm where QoS is explicitly considered.

**KPP Heuristic**

Kompella et al. [1993] proposed the KPP heuristic for finding a minimal cost multicast tree, subjected to delay constraints. The idea is to first construct a closure graph $G'$ out of the original graph $G$, then Prim’s algorithm is employed to build a minimum spanning tree of
the closure graph $G'$. The closure graph $G'$ is constructed in such a way that each edge in $G'$ corresponds to a delay-constrained cheapest path in the original graph $G$; the actual conversion is realized using dynamic programming. Its running time is $O(V^3 \Delta)$, where $\Delta$ is the delay bound; the running time growing very quickly with the number of nodes involved.

**Bauer’s Algorithm**

Bauer et al. considered the multicast routing problem under a slightly different setting: a constraint is imposed on the outgoing capacity of each node and the problem is formulated as a node-degree-constrained Steiner tree problem [Bauer and Varma, 1997]. They proposed the so-called *shortest path heuristic with iteration (SPH-R)* heuristic: it begins with an arbitrary node and in each step an edge closest to the partial tree is added; this operation is repeatedly applied to each node and the solution is the one with the minimal cost. Its time complexity is $O(MV^2)$, where $M$ is the number of edges.

**Jia’s Distributed Algorithm**

A distributed algorithm has been proposed by Jia [1998]. It makes use of the assumption that the least-cost path between two nodes is always the least-delay path as well. It begins with the source, and the source constructs a least-delay path $P$ from itself to a node, say node $i$; if the associated delay bound is met, node $i$ is marked as on-tree node included into the partial tree. Source $S$ then picks up another receiver $j$, whose least-delay path incurs the least cost, and sends a *setup* message to it. Upon receipt of the *setup* message, all the intermediate nodes updates the information. Its reported time complexity is $O(2M)$. However, the assumption it relies on is questionable.

To conclude, the aforementioned algorithms are either too complicated or rely on unrealistic assumptions. Therefore, direct porting of those algorithms to our scheme may not be very efficient. On the other hand, our scheme is based on our work presented in Chapter 4. Therefore, it makes sense to extend our work on reliability to accommodate other QoS parameters, and it is presented in next section.

### 5.3 A QoS-aware Scheme

This section describes in detail of how to make our previous scheme QoS-aware. Since it is based on our previous work, it is worth spending some time presenting its building blocks and explaining how the QoS-aware part is integrated into our previous work. Essentially,
peers are dynamically organized into a hierarchy according to their relative stabilities, which is captured and represented by their “rank” using the rank calculation procedure. Peers periodically check their ranks and initiate parent switching operation if necessary.

5.3.1 Building Blocks

The proposed algorithm consists of three building blocks: handling newly joined peers, logical hierarchy maintenance, and recovering from failures respectively. Furthermore, in order to justify the algorithm, an assumption is made: the newly joined nodes will be provided with a basic set of nodes to which they may contact, either by a well-known Rendezvous Point or by some out-of-band bootstrap procedure.

Handling Newly Joined Peers

The newly joined peers are mapped into the low end of the logical hierarchy. They are provided with an initial list, by the aforementioned bootstrap procedure; the newcomer contacts the nodes on the list and chooses to be attached to one of the on-tree nodes to join the ALM session.

Logical Hierarchy Maintenance

Each peer is mapped into a certain layer in the logical hierarchy at a given time and it will periodically make an attempt to migrate to a higher layer. Similar to the way in which a routing table is maintained, heartbeat and soft timer are also used to ensure the “freshness” of entries in the backup list, and the corresponding routing information is disseminated in a gossip way. When choosing the prospective new ”parent”, stability is used as the most important criterion, other factors are considered as well, such as bandwidth, network latency.

Recovering From Failures

Peer could detect the departure of its current parent by failing to receive either the Heartbeat message or the data packets. Once this happens, it initiates a rejoin process and contacts another peer in its backup list, trying to be reconnected to the network.

5.3.2 A New Parent Selection Procedure

A closer examination of those building blocks reveals that, in the heart of those blocks, lies a very important procedure: the parent selection procedure. Each peer uses the parent selec-
tion procedure to find its prospective parents-to-be, for either failure recovery or migration purposes. The stringent delay and bandwidth requirements could be dealt with in a much better way if the parent selection procedure could find high quality parents-to-be (with high upload bandwidth and low accumulated delay). Furthermore, the parent selection procedure is where the QoS aspects can be taken care of. A modified Dijkstra algorithm, executed on each peer in a distributed fashion, finds proper “parents-to-be” for each peer, subjected to bandwidth and delay constraints. Algorithm 5 describes this algorithm, and it finds itself maximum bandwidth paths to all the nodes with a higher “rank”, subject to a delay bound. To speedup the computation, Fibonacci heap [Cormen et al., 1990] is employed, leading to a time complexity of $O(V \log V + E)$.

### Algorithm 5: Modified Dijkstra (executed on peer x)

begin
  $S = \emptyset; \overline{S} = N$
  $b[i] = \infty$ for each peer $i \in N$
  $b[x] = 0; \text{pred}[x] = \text{NULL}$
  while $|S| < n$ do
    let $i \in \overline{S}$ such that $b[i] = \max\{b[j] : j \in S\}$
    subject to $\text{rank}[i] \geq \text{rank}[x]; \text{delay}[x][i] \leq \text{Threshold}$
    $S = S \cup \{i\}; \overline{S} = \overline{S} - \{i\}$
    for each $j \in N(i)$ do
      if $b[j] > \max\{b[j] : j \in S\}$ then
        $b[j] = \max\{b[j], \min\{\text{pred}, c_{kj}\}\}$ and $\text{pred}[j] = k$
      end if
    end for
  end while
end

Intuitively, Algorithm 5 starts at the peer itself, $x$, it grows a virtual multicast delivery tree $T$ gradually, that ultimately spans all peers that meet the rank and delay constraints; in addition, the bandwidth of the virtual links between those “on-tree” peers to peer $x$ are maximized. Therefore, the resulting “parents-to-be” list consists of those qualified “on-tree” peers.

### 5.4 Experimental Results

A discrete-event simulator was developed, and extensive simulation has been conducted to experimentally validate the proposed algorithm. Since there is no previous work that com-
pletely resembles the proposed algorithm, we modified and combined three major previous work [Birrer and Bustamante, 2006; Silber et al., 2004; Sripanidkulchai et al., 2004], for benchmarking purpose.

5.4.1 Simulation Setup

In order to simulate a realistic and meaningful network, 600000 nodes, representing 600000 peers were generated using the GT-ITM topology generator [Zegura et al., 1996]. Peers are randomly distributed among the generated topology. Peers join the on-going session following a Poisson distribution and their lifetime are drawn independently and identically from a Pareto distribution. This heavy-tailed lifetime distribution has been reported by several researchers, including [Bustamante and Qiao, 2004].

It is worth noting that simulations of telecommunication networks are often computationally intensive and often requires long runs in order to obtain results at a desired level of precision. Akaroa [aka] is employed with OmNet++ to dynamically control the simulation length in order to achieve a 95% confidence level.

5.4.2 Metrics of Interest

This section introduces two performance metrics, used to characterize and validate the proposed approach.

Quality of Service

As mentioned several times in this thesis, a Constant-Bit-Rate (CBR) traffic source emitted packet at a constant data rate, which is the playback rate $S$. Ideally, the amount of any peer’s received portion of the video packets should be proportional to its own lifetime; therefore, the Quality-of-Service (QoS) perceived by any peer would be completely reflected by the ratio of $\frac{\text{Amount of received video packet}}{\text{Self lifetime}}$. This ratio is denoted as “QoS” in all the figures presented in this section, taking on values between 0 and 1.

Frequency of Rejoin

In single-tree-based ALM, participating peers are subjected frequent service disruptions due to either the departure of its current parent or the failure to meet the real-time requirement. The affected peer immediately invokes the parent selection procedure, and attempts to be re-connected to the session through a qualified parent. Nevertheless, no matter how quick is
this re-connecting process, the affected peers suffer from this service disruption. Therefore, the frequency of rejoin is another important metric, reflecting the resulting ALM network from another perspective. It is very clear that the lower the frequency of rejoin, the better is the resulting ALM network.

5.4.3 Simulation Results

The main results are depicted in the following 4 figures, i.e., from Figure 5.2 to Figure 5.5. In all figures, the \( x \)-axis denotes the percentage of peers, and the \( y \)-axis represents the cumulative QoS. A QoS of 1 means that the peer’s perceived QoS is perfect, i.e., there is no packet loss due to network dynamics. It is worth pointing out that the simulation were carried out under different bandwidth settings: we normalized all peers’ bandwidth (upload and download bandwidth) by the sending rate, and the trace labeled as “5” means peers’ bandwidth is drawn from a power-law distribution with a maximum of 5. The reason is that we found out in the simulation that the cumulative QoS is highly sensitive to different bandwidth distributions, and this should give us an idea how will the proposed algorithm perform under different bandwidth settings. In reality, low-end video can be broadcast at around 300 Kbps. On the other hand, most peers have relative low bandwidth, and only a few peers, like campus users, have an access bandwidth of 10 Mbps, which is about 30 times the broadcast rate.

Bandwidth Has an Impact on QoS

Comparing Figure 5.2(a) with Figure 5.2(b) reveals two important observations. First, QoS of peers is highly sensitive to bandwidth: increasing average bandwidth, in particular the upload bandwidth, could enhance the cumulative QoS. The trend is very clear in both Figure 5.2(a) and Figure 5.2(b). The explanation is pretty straightforward: with the increase of bandwidth, the total depth of the multicast tree could be reduced, so does the failure rate, as has been explained in Section 5.1.2. Second, the proposed algorithm outperforms the benchmark by as large as 50%. When the average upload bandwidth is relatively small, denoting by traces labeled as “5”-“8”, both algorithms perform roughly the same, and our algorithm is slightly better. The reason is that when most of the peers have limited upload bandwidth, there is no way to optimize the multicast tree by reducing the average depth. However, we see a transition at “9” in Figure 5.2(a) and at “15” in Figure 5.2(b); the cumulative QoS improves dramatically in both figures. This shows that increasing average bandwidth could
Figure 5.2: Cumulative QoS comparison.
enhance QoS. Furthermore, if combined with the idea of relative stability, as in the proposed algorithm, the cumulative QoS can be further improved. This is because there is no positive correlation between bandwidth and relative stability, and consequently, a peer with a high bandwidth does not necessarily lead to a high stability. Therefore, increasing bandwidth alone is not enough in terms of QoS enhancement. For higher bandwidth, e.g., the traces labeled as “10”-“30” in Figure 5.2(a), our algorithm can maintain the QoS very close to 1 for most of the peers. It is worth emphasizing “30” means that the maximum achievable upload bandwidth is 30 times the broadcast rate. However, most peers’ bandwidth are relatively very low since the bandwidth follows a power-law distribution. This implies that our algorithm works even for peers with relative low bandwidth, which is the case in most real-world applications.

**It Works for Other Lifetime Distributions As Well**

![Figure 5.3: QoS under different lifetime distributions.](image)

It is interesting to note that our algorithm is applicable not only to a heavy-tailed lifetime distribution, but also to other lifetime distributions. As can be seen from Figure 5.3, our algorithm performs surprisingly well for uniform and exponential distributions. The graph labeled $exp - X$ means the lifetime is drawn from an exponential distribution with a mean of
The trend is very clear that with the increase of the mean, as the achievable cumulative QoS begins to approach to that of a heavy-tailed lifetime distribution marked as “Zipf”.

**Delay Does Affect QoS**

Figure 5.4 shows the impact that different delay constraints have on peers’ cumulative QoS. The graph labeled “X” stands for the corresponding delay constraint is X hops. It can be seen from Figure 5.4(a) that the proposed algorithm is not as sensitive to delay as that of the benchmark algorithm. The proposed algorithm achieves desired results, and 90% of the peers have satisfactory QoS, even with a very tight delay constraint, and this is vital for the success of real-world real-time video streaming service. The reason is that delay has been deliberately taken into consideration in our parent selection procedure, detailed in Section 5.3.2. In comparison, Figure 5.4(b) shows that the benchmark algorithm suffers from lack of consideration for delay.

**Why Does the Proposed Algorithm Work**

Figure 5.5 reveals the reason why the proposed algorithm greatly outperforms the benchmark algorithm. As can be seen from Figure 5.5, the rejoin frequency of most peers can be kept very low, causing as few as 1 or 2 service disruptions for most of the peers. This is achieved as in the proposed algorithm, reliability is singled out and all peers are dynamically organized into a hierarchy according to their relative stability with the aim to minimize service disruption frequency so as to enhance reliability. On the contrary, Figure 5.5(b) shows that in the benchmark algorithm, the rejoin frequency could be as high as 1000, i.e., some peers will experience 1000 service disruptions throughout the session. As stated in Section 5.1.3, most peers are short-lived, only a handful of peers remain active throughout the session, and those long-lived peers will suffer from very frequent service disruptions if they do not choose their parents by taking reliability into consideration.

**5.5 Conclusion**

Quality of service (QoS), especially reliability, has become the prominent concern in Application Layer Multicast. Each peer’s perceived QoS is heavily related to that of each other. The proposed algorithm makes use of this unique characteristic, dynamically organizing peers into a hierarchy according to their relative stability. Both mathematical and experimental
Figure 5.4: QoS under different delay constraints.
CHAPTER 5. QOS-AWARE RELIABLE APPLICATION LAYER MULTICAST

(a) The proposed algorithm

(b) Benchmark algorithm

Figure 5.5: Rejoin frequency comparison.
analysis validates our algorithm’s effectiveness in terms of reliability enhancement. The proposed algorithm works for a heavy-tailed lifetime distribution, and is also applicable to other lifetime distributions. Most importantly, the proposed algorithm closely resembles the real-world video streaming in the sense that it incorporates delay and bandwidth, the two most important parameters, into consideration. Extensive simulation shows that the proposed algorithm even works under tight bandwidth and delay constraints, indicating its feasibility in real-time video streaming applications.
Chapter 6

Admission Control for Application Layer Multicast

The previous two chapters attempted to address two very important issues of Application Layer Multicast: reliability and Quality of Service (QoS). Simulation results are encouraging, indicating that the proposed algorithms are effective. However, performance of those algorithms can be enhanced further. To be more specific, when most peers have limited upload capacity, the achievable QoS is only around 70%. We have also learned that peers’ perceived QoS is highly sensitive to bandwidth distribution, in particular, peers’ upload bandwidth distribution.

It is quite natural to extend our research to another vital issue of ALM, i.e., admission control. The justification is that in ALM, a peer acts as both a server and a client, and there is no infrastructure to contribute extra bandwidth to support ALM. When most peers have limited upload bandwidth, the available resource, in the form of upload capacity, becomes scarce, and an effective admission control becomes vital to the success of ALM, since a decision has to be made on which peer to accept or which one to reject, and this decision will have a profound effect on latecomers. The admission control problem is ultimately simplified to an optimization problem, and to be exact, a stochastic knapsack problem. Due to its inherent complexity, a heuristic-based algorithm is proposed. Although simple, it is quite effective and successfully bridges the performance gap, and a minimum of 20% improvement can be achieved.
CHAPTER 6. ADMISSION CONTROL FOR APPLICATION LAYER MULTICAST

6.1 Motivation

Since QoS-related problems have been addressed in the previous two chapters, the crux of the problem now is to find a good admission control algorithm to further boost the performance. The importance of a good admission control algorithm can be demonstrated by a very simple example in Figure 6.1. For example, two peers, 2 and 3, simultaneously want to join the existing session by being attached to an on-tree node 1. Further assume their download bandwidth are 2 and 1; and their upload bandwidth are 2 and 1 as well. As shown in Figure 6.1(a), if peer 3 is admitted before peer 2, peer 2 will be blocked; on the contrary, if peer 2 is admitted first, both peer 2 and peer 3 will be accepted, as depicted in Figure 6.1(b).

Therefore, a good admission control is vital to the success of any Application Layer Multicast (ALM) applications. Furthermore, as shown by Kunwadee et al., most applications in ALM are short-lived, and there are an average of 3.3 requests from a single IP address during a session [Sripanidkulchai, Ganjam, Maggs, and Zhang, 2004]. Under such a high churn rate, it is desirable for the admission control algorithm to have the following properties.

- **Fast**: Decision, in terms of whether or not to accept the incoming peer, needs to be made in real-time.

- **Efficient**: In order to support real-time decision making, the algorithm should be efficient, i.e., it should be as computationally simple as possible.

- **Effective**: The admission control algorithm still needs to be effective, i.e., the algorithm should be able to improve the Quality of Service (QoS) perceived by all peers.
CHAPTER 6. ADMISSION CONTROL FOR APPLICATION LAYER MULTICAST

- **Distributed:** Due to the inherent distributed nature of Peer-to-Peer (P2P) systems, the algorithm should easily lend itself to a distributed implementation.

Bearing all these requirements in mind, this chapter investigates the use of admission control techniques to further boost the performance of ALM systems. The actual admission control problem, in the context of ALM, is mathematically formulated as an optimization problem, in particular, as a stochastic knapsack problem. Since the admission control algorithm is built on top of our previous work, it needs to be integrated into the protocol described in Chapter 5. Therefore, it differs from the standard stochastic knapsack problem formulation [Ross and Tsang, 1989] by a great deal, and will be explained in greater detail in Section 6.3.

Due to the problem’s inherent computational intractability, a heuristic-based algorithm is proposed. It first tags each peer with a cost and reward function, based on its stability and bandwidth (both download and upload bandwidth); then these cost and reward functions are mapped and fitted into the standard stochastic knapsack problem formulation [Ross and Tsang, 1989]. Thereafter, peers are ordered and admitted into the ALM system in a greedy way. This heuristic-based approach is experimentally verified to be effective, in spite of its simplicity.

6.2 Related Work

Admission control is not a new topic: it has been studied extensively and there is a large body of published work [Ferrari and Verma, 1990; Anderson et al., 1992; Wrege et al., 1996; Lee et al., 1996; Knightly and Shroff, 1999]. Admission control refers to the ability of the system to ensure the desired QoS for existing users (i.e., the users that have been accepted) by rejecting some of the prospective users and reserving sufficient resources for existing ones. In a multimedia or Video-on-Demand (VoD) context, two major resources are worth special consideration: storage and bandwidth. Therefore, most admission control work focuses on these two aspects. Nevertheless, with the advance of technology, storage (mainly in the form of disk) is no longer a primary concern. Hence, our focus here is on admission control policy over bandwidth. Generally speaking, admission control algorithms can be classified into two categories: deterministic and statistical algorithms.
6.2.1 Deterministic Algorithms

Deterministic admission control policies can ensure guaranteed Quality of Service (QoS) for the existing users, in the sense that the continuous playback requirement is never violated and no packets are dropped or delayed beyond the pre-defined delay bound [Anderson et al., 1992; Wrege et al., 1996]. This deterministic approach is pessimistic, in the sense that it always focuses on the worst-case scenario: a significant number of the prospective users would be rejected on the assumption that the acceptance of them might result in the violation or degradation of the QoS of the existing users. In other words, the system resources are generally under-utilized, resulting in a low throughput.

On the other hand, the main resource in P2P systems, in the form of aggregated upload bandwidth, is already scarce; if the admission control algorithm could not lead to efficient usage of it, the service capacity would be further diminished. Therefore, a better admission control algorithm is necessary.

6.2.2 Statistical Algorithms

Due to the persistence of vision effect, small delay and rate variation of the streaming video are not humanly perceptible, i.e., hard real-time requirement is not necessary. Soft real-time service is perfectly acceptable. Hence, there is a tradeoff between QoS as perceived by users and utilization: delay and other QoS bounds can be slightly violated, trading utilization from system perspective [Knightly and Shroff, 1999]. A large body of work has been developed along this research track because of its ability to boost system utilization. They are further divided into four classes: the differences between them are mainly the way in which they specify the traffic characteristics of the source.

Average and Peak Combinatorics

Lee et al. [1996] proposed to use average and peak rate to approximate the packet loss probability in a bufferless environment.

\[
f_j(x) = \begin{cases} 
1 - p_{on,j} & x = 0 \\
p_{on,j} & x = r_{pk,j} \\
0 & \text{otherwise}
\end{cases}
\]

where an on-off source \(j\) is fully characterized by its peak rate \(r_{pk,j}\) and average rate \(r_{av,j}\); it either transmits at its peak rate when it is on or does not transmit at all when it is
idle. \( p_{on,j} = \frac{r_{av,j}}{r_{pk,j}} \) is the probability that it is on. Taking these peak and average rates into
consideration, the packet loss occurs when the aggregate arrival rate is greater than the link
capacity. Lee et al. [1996] further developed efficient methods to compute the aggregate
arrival rate and the corresponding loss probability.

Ferrari and Verma [1990] deployed the similar idea of using peak and average rate to
capture the characteristics of the traffic source, but using \( r_{av,j} \) to represent the long-term
average rate, or worst-case rate over any interval of length \( l \). Based on this, they showed how
to calculate the probability of violating the delay bound for Earliest Deadline First schedulers
based on the proposed long-term average rate \( r_{av,j} \).

**Additive Effective Bandwidth**

Effective bandwidth is another way to characterize the traffic source, and several schemes
have been proposed [Guerin et al., 1991; Elwalid and Mitra, 1993; Kesidis et al., 1993; Chang,
1994]. The effective bandwidth is so-called because each flow independently reserves its own
share of bandwidth that is between its average and peak rate. This effective bandwidth is a
function of both the required loss probability \( P_l \) and the associated flow’s stochastic features,
e.g., its auto-correlation function, peak and average rate, etc. The admission control is based
on the derived effective bandwidth \( E_j(P_l) \):

\[
\sum_{j=1}^{N} E_j(P_l) < C \quad (6.1)
\]

where \( N \) is the number of flows sharing the link and \( C \) is the capacity of the link.

There are several published papers targeted at improving the effective bandwidth scheme
by using a number of interrelated techniques, e.g., envelope processes [Chang, 1994], large
deviations theory [Kesidis et al., 1993], and eigenvalue decomposition of Markovian flows [El-
walid and Mitra, 1993].

**Engineering the Loss Curve**

Loss probability has a close relationship with buffer size, and it is worthwhile to investi-
gate this relationship. Generally speaking, the loss probability is in an inverse exponential
proportion to the buffer size:

\[
P_l = e^{-\delta B} \quad (6.2)
\]
where $B$ is the buffer size. The crux is how to fit the loss curve into the experimental observations.

Shroff and Schwartz [1998] proposed the so-called hybrid scheme, stating that when the arrival process could be modeled as a general Markov modulated arrival process and with a Markov modulated fluid source with arrival rate $\lambda$, and further defining $\pi_i$ as the probability of being in state $i$, the actual loss probability can be described as:

$$
P_l = \frac{\sum_i \lambda_i \pi_i (1 - \frac{C}{\lambda}) e^{-\delta B}}{\lambda}
$$

(6.3)

where $B$ is buffer size and $C$ is the capacity of the link.

**Maximum Variance Based Approaches**

Denote $X_t$ as the aggregate arrival rate from a large number of sources, and $X_t$ is expressed as:

$$X_t = \sum_j A_j[s - t, s] - C_t
$$

(6.4)

and the associated tail probability is

$$P(Q > B) = P(\sup X_t > B)
$$

(6.5)

the idea of Maximum Variance Based Approaches is if $X_t$ is Gaussian, explicit formula may be derived for maximum variance of $X_t$:

$$\sigma^2_B = \max_t \frac{\text{var} X_t}{(B - E(E_t))^2}
$$

(6.6)

To maximize $P(X_t > B)$ is equivalent to maximizing the variance $\sigma^2_B$. Therefore, the time instance that maximizes the variance $\sigma^2_B$ will achieve the maximal value for $P(X_t > B)$ [Choe and Shroff, 1998; Kim and Shroff, 2001].

All the aforementioned admission control schemes are primarily designed for ATM, and most of them focus on finding computationally efficient methods to calculate the characteristics of traffic source, e.g., using effective bandwidth. A simple admission control test is carried out to check whether the available resources is sufficient for the incoming request, then a decision is made based on this test.

When considering Constant Bit Rate (CBR) Video on Demand (VoD), the playback rate
and delay-bound are sufficient to characterize the streaming source. Hence the focus should be an efficient and effective admission control scheme that is superior to the previous admission control test, rather than focusing on finding efficient computational methods for capturing traffic source’s features. Detailed description of the problem is given in the following section, alongside some related work on the stochastic knapsack problem.

6.3 Mathematical Problem Formulation

This admission control and resource sharing problem has been formulated as a stochastic knapsack problem [Ross and Tsang, 1989]. For the purpose of clarification, it is worth introducing the classical knapsack problem first.

The classical knapsack problem is defined as follows [Martello and Toth, 1990]:

*Given a knapsack of volume $V$ and objects taken from $K$ different classes; furthermore, once selected, each object from class $k, k = 1, \ldots, K$, will consume $b_k$ of the total volume and reap of a reward of $R_k$. The aim is to pack the knapsack in such a way so as to maximize the total reward.*

This classical knapsack problem has been extensively studied although it is a well-known NP-hard problem, and many solutions, including both exact and approximation algorithms, have been proposed [Martello and Toth, 1990]. In particular, a very simple solution exists when the knapsack volume $V$ is an integer multiple of the object volume $b_k$: first order all the objects in a non-decreasing fashion, according to the so-called profit-to-volume ratio $\frac{R_k}{b_k}$, then fill the knapsack in order.

This multiple integer requirement is relatively easy to be satisfied in reality by modeling the shared bandwidth as the knapsack, and the available bandwidth as the corresponding knapsack volume. For example, most low quality VoD is delivered at 300 kbps, and the shared bandwidth could be multiple times this playback rate. In other words, the static version of the knapsack problem is easily solved. Unfortunately, there is a very high churn rate in most P2P systems, i.e., peers keep joining and departing during the session, adding another layer of complexity to the already NP-hard problem. Consequently, the static knapsack problem has now changed to its stochastic counterpart:

*The system is best described using a vector $n = (n_1, n_2, \ldots, n_k)$ where $n_k$ is the number of class $k$ objects in the system. Objects arrive in the system according to a Poisson distribution with an arrival rate of $\lambda_k$; once accepted, they will consume $b_k$ of the knapsack volume, and release this $b_k$ resources when they depart after a random sojourn time with a mean*
of $\mu_k^{-1}$. The previous once-off reward $R_k$ has now changed to the reward rate $r_k$, and obviously $r_k = R_k \mu_k$.

This stochastic knapsack was first defined by Ross and Tsang [1989], and they pointed out that it is indeed a Markov Decision Process (MDP) problem. Furthermore, due to its inherent complexity, they claimed that they could not establish structural properties for the cases where there are more than two object classes. Little work has been done since the problem was first identified nearly 20 years ago. Only small number of papers have been published, including Ross and Tsang [1989]'s original paper. Furthermore, no satisfactory solution has been found except the time-consuming dynamic programming approach [Puterman, 1994]. There is another branch of research focusing on the stochastic knapsack. Nevertheless, they focus on the problem settings in which objects have random sized volume and reward, and it is quite distinct from our problem.

\begin{figure}[h]
\centering
\begin{tikzpicture}
\node (1) at (0,0) {1};
\node (2) at (1,1) {2};
\node (3) at (2,1) {3};
\node (6) at (2,0) {6};
\draw (1) -- (2);
\draw (1) -- (3);
\draw (2) -- (6);
\draw (3) -- (6);
\node at (-0.5,0.5) {6(1)};
\node at (0.5,0.5) {2};
\node at (1.5,0.5) {3};
\node at (2.5,0.5) {6(6)};
\node at (-0.5,1.5) {6(1)};
\node at (0.5,1.5) {2};
\node at (1.5,1.5) {3};
\node at (2.5,1.5) {6(6)};
\end{tikzpicture}
\caption{A simple example illustrating State-Dependant Markov Decision Process (SD-MDP).}
\end{figure}

The problem we consider here is further complicated by the fact that the knapsack vol-

(March 23, 2008)
CHAPTER 6. ADMISSION CONTROL FOR APPLICATION LAYER MULTICAST

Volume $V$ is per se a random variable, compared to the static one in the original stochastic knapsack problem. The reason is very clear: the resources that is modeled as the knapsack in our problem, is in the form of bandwidth, in particular, the upload bandwidth. The prominent feature of P2P system is that there is no infrastructure support and all of the upload bandwidth is contributed by participating peers. Therefore, the problem we consider is actually a State Dependant Markov Decision Process (SDMDP), i.e., the current decision on which peer to accept has a long-term impact on the performance of the system. Figure 6.2 gives a simple example and demonstrates this impact, where node 1 is the streaming source with an upload bandwidth of 6, and nodes 2 and 3 are prospective users. The digits beside each node indicate its download (upload) bandwidth. Assume a reward equating to the node’s download bandwidth will be realized if it is accepted, and further assume that nodes 2 and 3 want to join the session at the same time. If node 2 is accepted first, there is no room for node 3, and the total reward collected now is 6 as shown in Figure 6.2(b). On the other hand, as demonstrated in Figure 6.2(c), if node 3 is accepted first, node 2 can be accommodated as well, and the reward now is 12 instead of 6. This simple example clearly shows that a good admission control algorithm is necessary for P2P streaming systems.

A Precise Definition of the Problem

State Dependant Markov Decision Process (SDMDP): The system is best described using a vector $n = (n_1, n_2, ..., n_k)$ where $n_k$ is the number of class $k$ peers in the system. Peers arrive in the system according to a Poisson distribution with an arrival rate of $\lambda_k$; once accepted, they will consume $b_k$ of the knapsack volume, and release this $b_k$ resources when they depart after a random sojourn time $\mu_k^{-1}$. Let $B_k$ represent the blocking probability of class $k$ peers, then the acceptance of a class $k$ peer leads to a reward of $r_k$, and apparently $r_k = R_k \mu_k$. The aim is to admit incoming peers in such a way so as to maximize the expected long-run reward:

$$\text{Max} \sum_{i=1}^{K} \lambda_k (1 - B_k) R_k \quad (6.7)$$

Hardness of the Problem

This State Dependant Markov Decision Process belongs to the general Markov Decision Process (MDP). It has been shown that it is analytically intractable and computationally complex to solve the MDP problems exactly [Puterman, 1994]. Along this research line, Ross
and Tsang proposed to solve the MDP problem using a standard Linear Programming (LP) approach: dynamic programming equations are derived from the original stochastic knapsack problem, and standard value or policy iteration techniques are employed to solve the problem numerically [Ross and Tsang, 1989]. On the negative side of the problem, Jordan and Varaiya [1994] have shown that the optimal solutions to MDP problems generally lack a well-defined structure, i.e., there is no systematic way to solve those problems. Therefore, the solutions to MDP problems are problem-dependant, and only ad-hoc approaches can be employed in the context of individual problem formulation.

This lack of a well-defined structure for an optimal admission control policy naturally leads to the other end of the solution spectrum: using heuristic and approximation algorithms to obtain good suboptimal solutions, trading solution quality for time complexity. Gavious and Rosberg took this approach and derived a performance bound for a heuristic-based approximation policy [Gavious and Rosberg, 1994]. In order to simplify the problem, Ross and Tsang refined the problem and only focused on one particular class of policy: the so-called Coordinate Convex policies. Under this assumption, the stochastic knapsack problem becomes tractable and relatively easy to solve: a product form solution can be derived recursively for the steady state distributions. Unfortunately, it is only applicable to several special cases. In other cases, the solution produced by this approach is anything but optimal [Jordan and Varaiya, 1994].

The fact that the problem considered here is analytically intractable argues the necessity of a good approximation algorithm, based on the four properties described in Section 6.1. The justifications for a fast and distributed approximation algorithm are:

- The overlay network formed by participating users is highly dynamic, and most peers are short-lived [Sripanidkulchai, Ganjam, Maggs, and Zhang, 2004]. The question is whether it is worth performing extensive computation to find an optimal solution through complete searching of the solution space under such a high churn rate: by the time an optimal admission control policy is found, the peers involved may have already left the session. Not to mention the prohibitively large search space, sometimes it takes several hours to do a complete search.

- Performing extensive computation may overload the end systems, which are normally personal computers that are not equipped with powerful CPUs. Overloading the end systems may further decrease the service capacity of the network, resulting in the rejection of more requesting peers: users of the overloaded systems may shut down the
• Long setup delay will be experienced by some users since they are geographically widely distributed, and if they are rejected by their first choice, they have to turn to other users or choose not to join the streaming session.

• It is very difficult, if not impossible, to design a central entity to handle all the membership related activities: track the dynamics of peers, collect statistics of all peers (e.g., bandwidth, delay and etc.), and maintain a lookup list containing all of the active peers, and so on. Therefore, admission control decision has to be made based on local information only: the incoming peers only get a sample of the existing peers to which they may send their connection requests of connecting. The contacted peers make the decision on whether to accept or reject the connection request based on its current status, e.g., available upload bandwidth, without considering others and any future events. In a word, a distributed algorithm is highly desirable in such a circumstance.

Bearing all these considerations and tradeoffs in mind, a distributed heuristic-based algorithm is proposed and presented in the following section.

6.4 The Proposed Admission Control Protocol

This section gives a detailed description of the proposed distributed algorithm. This algorithm is mainly an admission control algorithm working on top of our previous work presented in Chapter 4 and Chapter 5. Before diving into the full details of the admission control algorithm, it is worth briefly refreshing some of our previous work.

For the purpose of enhancing reliability, peers are organized into hierarchical layers according to their current “age” and time is divided into logarithmic-sized bins; once a peer’s age reaches the pre-defined threshold, it starts looking for potentially more stable “parents” and migrating to a higher layer, reflecting the fact that it could be more stable [Rong et al., 2006a;b].

The proposed distributed admission control takes place on two occasions: for the incoming peers who wish to join the session and for peers issuing a rejoin or migration request, i.e., it happens when decisions need to be made on which peer to accept or which one to reject.
6.4.1 Membership Information Management

As explained in Section 6.3, it is quite difficult for each peer to have up-to-date and complete knowledge of all the other active peers; on the other hand, a distributed protocol ought to work with a distributed membership management algorithm. An adaptive gossip-based membership management algorithm is deployed to disseminate membership related information in a distributed fashion: each peer maintains an “inView” and “outView” list representing the nodes that know of it and the nodes it knows respectively; each node chooses several nodes in a pseudo-random way in each gossip round to distribute membership related information [Rong et al., 2006c; 2005]. On average, each peer maintains two lists sized $O(ln(n))$, for the purpose of being connected and facilitating parents switching.

6.4.2 A Distributed Admission Control Algorithm

The actual admission control algorithm consists of two phases: a bidding phase and a resource allocation phase. It mimics the actual bidding process happening everyday in real-life: the requesting peers send bids to prospective parents and the contacted parents make the admission control decision independently in a distributed fashion.

Parent Selection and Bidding

The very fist step of the proposed admission control algorithm is the parent selection procedure: when it is time for migration through parent switching or its current parent fails, each peer independently searches for a prospective parents-to-be. A modified Dijkstra algorithm, executed on each peer in a distributed fashion, finds proper “parents-to-be”, subjected to bandwidth and delay constraints. Algorithm 6 depicts this parent selection procedure: it is derived from the famous Dijkstra’s shortest path algorithm and searches parent-to-be in a greedy way. Its running time is $O(V \log V + E)$, where $V$ is the number of nodes and $E$ is the number of edges.

Intuitively, Algorithm 6 starts at the peer itself, $x$, it grows a virtual tree $T$, that ultimately spans all peers that meet the rank and delay constraints. In addition, the bandwidth of the virtual links between those “on-tree” peers to peer $x$ are maximized. Therefore, the resulting “parent-to-be” list consists of those “on-tree” peers.

The requesting peer sends bidding messages to all candidate parents-to-be, containing its own lifetime, upload and download bandwidth capacities, requesting connection to these peers for rejoining or parents switching purposes.
Algorithm 6 Modified Dijkstra (executed on peer x)

\begin{verbatim}
begin
S = ∅; S = N;
b[i] = ∞ for each peer i ∈ N;
b[x] = 0; pred[x] = NULL;
while |S| < n do
    let i ∈ S such that b[i] = max{b[j]; j ∈ S};
    subject to rank[i] ≥ rank[x]; delay[x][i] ≤ Threshold;
    S = S ∪ {i}; S = S − {i};
    for each j ∈ N(j) do
        if b[j] > max{b[j]; j ∈ S} then
            b[j] = max{b[j], min{b[k], c[kj]}} and pred[j] = k;
        end if
    end for
end while
end
\end{verbatim}

Distributed Admission

Upon receipt of these bidding messages, the contacted peer makes the admission decision. Take peer i as an example, its available upload bandwidth \( b_u[i] \) is equivalent to the capacity of the to-be-packed knapsack; it sorts the received bids according to the ratio \( \frac{b_u[i] - \min\{S, b_d[j]\}}{\min\{S, b_d[j]\}} \), and admits peers in a nondecreasing order according to the same ratio. Algorithm 7 gives the details, where \( S \) is the playback rate. As has been explained in Section 6.3, the reason that algorithm 7 works is that when the knapsack volume \( V \) is an integer multiple of the object volume \( b_k \), the greedy algorithm will always achieve the optimal solution. In reality, the widely used Multiple Description Coding (MDC) [Goyal, 2001] enables users to have the freedom to choose the descriptions that best match their access bandwidth, justifying the integral assumption. On the other hand, the ratio \( \frac{b_u[j] - \min\{S, b_d[j]\}}{\min\{S, b_d[j]\}} \) could be viewed as the modified profit-to-volume ratio, where \( \min\{S, b_d[j]\} \) is the actual download rate of peer \( j \), and \( b_u[j] - \min\{S, b_d[j]\} \) is the effective profit gain if peer \( j \) is accepted; in other words, it is the additional upload bandwidth contributed to the system. Admitting peers according to this ratio will enable ALM to favour peers that can boost the capacity of the system. Rejecting some users with asymmetric bandwidth distribution (e.g., ADSL users) may reserve the scarce upload bandwidth for more profitable users (e.g., campus users), leading to the rapid expansion of system capacity and, as a compensation, more asymmetric users may be admitted later on.
Algorithm 7 Admission Control Procedure (executed on peer \( x \))

begin
    sort biddings according to ratio \( \frac{b_x^u - \min\{S, b_j^d\}}{\min\{S, b_j^d\}} \)
    while \( b_x^u - \min\{S, b_j^d\} \geq 0 \) do
        admit peer \( j \);
    end while
end

6.5 Analytical Results

A multicast tree is a tree spanning all participants, rooted at the streaming source node \( S \). This multicast tree could be viewed as a set of paths connecting \( s \) to each of the participants \( v_i \). In a single-tree-based P2P media streaming system, participating peers are dynamically organized into a hierarchy of \( L \) layers: layer 0 consists of the CBR source only, layer 1 contains the peers that are attached to the CBR source directly, and layer 2 peers receive streaming data from layer 1 peers and forward the received streaming data to layer 3 peers in turn, and so on. Figure 6.3 depicts a simple example, in which node 0 belongs to layer 0, nodes 2 and 7 are layer 1 nodes, nodes 3, 4, and 8 belong to layer 2, and layer 3 contains nodes 5 and 6.

The generation of the multicast tree can be modeled as a discrete-time Markov chain, or a branching process. Following the conventional terminology, nodes are classified into different generations [Feller, 1968].

Let \( X_n \) represents the size of the \( n^{th} \) generation, it has a mean of \( \gamma \) and a variance of \( \sigma^2 \). For simplicity, assuming that \( E[X_0] = 1 \). Simple recursive relations exist:

\[
E[X_n] = \gamma E[X_{n-1}]
\] (6.8)

\[
\text{Var}[X_n] = \sigma^2 E[X_{n-1}] + \gamma^2 \text{Var}[X_{n-1}]
\] (6.9)

Recall that \( E[X_0] = 1 \), Equation 6.8 and Equation 6.9 can be solved:

\[
E[X_n] = \gamma^n
\] (6.10)
Figure 6.3: An example of single-tree-based peer-to-peer media streaming.

\[
Var[X_n] = \begin{cases} 
  n\sigma^2 & \text{if } \gamma = 1 \\
  \sigma^2\gamma^{n-1}\frac{\gamma^n-1}{\gamma-1} & \text{if } \gamma \neq 1
\end{cases}
\]

the branching process will continue if and only if when \( \gamma > 1 \), and has a total number of

\[
\sum E[X_n] = \frac{\gamma^{n+1}-1}{\gamma-1}
\]  

(6.11)

Denoting the average depth of the multicast as \( E[d] \) and solving Equation 6.10, we have:

\[
E[d] = \frac{\ln((\gamma - 1)E[X_n] + 1)}{\ln\gamma - 1} - 1
\]  

(6.12)

\( \gamma \) represents the fan-out of this ALM tree: for a given generation \( n \), increasing \( \gamma \) will increase the total number of peers that can be accommodated; for a given total number of peers, increasing \( \gamma \) results in a decreased average depth \( E[d] \). In reality, \( \gamma \) strongly depends on the average upload bandwidth: the larger the average upload bandwidth, the greater the value of \( \gamma \). Figure 6.4 clearly demonstrates how service capacity changes with \( \gamma \) and the
average number of generations.

![Diagram showing how capacity changes with gamma and number of generations.](image)

**Figure 6.4:** How capacity changes with gamma and number of generations.

In order to maximize the capacity of an ALM system, for a given delay bound, increasing $\gamma$ will increase the achievable bandwidth, as illustrated in Figure 6.4. It is not hard to verify that increasing $\gamma$ will decrease the average depth. On the other hand, for a peer $n$, its stability relies on the stability of upstream ancestors, and it could be simplified as $\prod_{j=1}^{n} p_{j}$, where $p_{j}$ is the stability of its ancestor peer $j$ and is a conditional probability, which in turn depends on its ancestors as well. Nevertheless, since $p_{j} \leq 1$, reducing the average depth will reduce the average failure rate. This explains why previous work had worked by constructing a minimum depth tree [Birrer and Bustamante, 2006; Silber et al., 2004; Sripacondkulchai et al., 2004].

### 6.6 Experimental Results

A discrete-event simulator was developed, and extensive simulation has been conducted to experimentally validate the proposed algorithm. Since there is no previous work that completely resembles the proposed algorithm and this admission control algorithm is built on top of our previous work presented in Chapter 4 and Chapter 5, we compared two scenarios:
i.e., the cases with and without admission control.

6.6.1 Simulation Setup

In order to simulate a realistic network, 600000 nodes, representing 600000 peers were generated using the GT-ITM topology generator [Zegura et al., 1996]. Peers are randomly distributed among the generated topology. Peers join the on-going session following a Poisson distribution and their lifetime are drawn independently and identically from a Pareto distribution. This heavy-tailed lifetime distribution has been reported by several researchers, including [Bustamante and Qiao, 2004].

In the meantime, in order to achieve a 95% confidence level, Akaroa [aka] is used to control the simulation duration in a dynamic fashion, i.e., the simulation length is determined by Akaroa based on the number of observation collected.

6.6.2 Metrics of Interest

This section introduces several performance metrics, used to characterize, validate, and compare the case when admission control is used with the case without admission control.

Quality of Service

As mentioned several times in this thesis, a Constant-Bit-Rate (CBR) traffic source emits packets at a constant data rate, which is the playback rate \( S \). Ideally, the fraction of any peer’s received portion of the video packets should be proportional to its own lifetime; therefore, the Quality-of-Service (QoS) perceived by any peer would be completely reflected by the ratio of

\[
\text{Quality of Service (QoS) = } \frac{\text{Amount of received video packet}}{S \cdot \text{lifetime}}
\]

This ratio is denoted as “QoS” in all the figures presented in this section, taking on values between 0 and 1.

Number of Rejected Peers

The goal of an admission control algorithm is to optimize the use of the available resources. The proposed admission control algorithm is no exception. A direct reflection of this efficiency is the number of peers that are rejected; and this rejection rate is referred to as “Number of rejected peers” in the figures presented shortly in this section.
CHAPTER 6. ADMISSION CONTROL FOR APPLICATION LAYER MULTICAST

Frequency of Rejoin

Due to the reason described in Section 6.6.3, the efficiency with which the available resources can be used is a useful metric. The “Acceptance Rate” is applicable to the newcomers, i.e., for incoming peers; however, it is irrelevant to the existing users. Hence, the successful rejoin rate is a natural choice, as it represents the probability that a peer’s rejoin action ends in success.

6.6.3 Simulation Results

The main results are depicted in the following 5 figures. In Figure 6.5, Figure 6.6, and Figure 6.7, the $x$-axis denotes the cumulative proportion of peers; the $y$-axis represents the cumulative Quality of Service (QoS). A QoS of 1 means the peer’s perceived QoS is perfect, i.e., there is no packet loss due to network dynamics. It is worth pointing out that the simulations were carried out under different bandwidth settings: we normalized all peers’ bandwidth (upload and download bandwidth) by the sending rate, and the trace labeled “5” means peers’ bandwidth is drawn from a power-law distribution with a maximum of 5. We found out in the simulation that the cumulative QoS is highly sensitive to different bandwidth distributions. This should give us an idea how the proposed algorithm will perform under different bandwidth settings. In reality, low-end video can be broadcasted at around 300 Kbps. On the other hand, most peers have relatively low bandwidth, and only a few peers, like campus users, have an access bandwidth of 10 Mbps, which is about 30 times the broadcast rate. In Figure 6.8 and Figure 6.9, the $x$-axis represents the relative bandwidth: for example, a relative bandwidth of “5” means the upload bandwidth is drawn from a heavy-tailed distribution with a maximum of 5, i.e., most peers have a small upload bandwidth, and only a handful of them have the maximum achievable upload bandwidth of 5.

QoS is Sensitive to Upload Bandwidth

As expected, the cumulative QoS is very sensitive to the available upload bandwidth. Figure 6.5 clearly demonstrates that increasing the average upload bandwidth improves the achievable QoS. There are transition points both in Figure 6.5(a) and Figure 6.5(b): when most peers have relatively low bandwidth, denoted by traces labeled as ”5”-“8”, there is no way to optimize the multicast tree by reducing the average depth, and peers are forced to form long chains. This structure is highly vulnerable to random failures along the multicast “chain”. With the increase of bandwidth, the total depth of the multicast tree can be
Figure 6.5: QoS comparison.
CHAPTER 6. ADMISSION CONTROL FOR APPLICATION LAYER MULTICAST

Reduced, so does the failure rate, as explained in Section 6.5 and demonstrated by traces labeled as “9”-“30”. Comparing Figure 6.5(a) with Figure 6.5(b) reveals that with the aid of an admission control algorithm, there is a slight improvement in cumulative QoS with low upload bandwidth distributions; on the other hand, there is an improvement as large as 20% when most peers have relative large upload bandwidth. In particular, as shown by traces labeled as “10”-“30” in Figure 6.5(a), the majority of peers can receive the streaming data with minimal interruptions. It is worth pointing out that the trace labeled as “30” means the maximum achievable upload bandwidth is 30, which is equivalent to 10 Mbps. Since peers’ upload bandwidth distribution is heavy-tailed, i.e., most peers still have a relatively low bandwidth, and only a handful of them are “campus users” that have abundant upload bandwidth. The enforced admission control policy gives preference to users with high upload bandwidth. This leads to the fast amplification of system capacity and reduce the average multicast tree depth. According to the analysis done in Section 6.5, this will improve the cumulative QoS. A comparison between Figure 6.5(a) and Figure 6.5(b) shows that the proposed admission control algorithm is very effective, although it is very simple and intuitive.

Delay Bound also Affects the Cumulative QoS

As formulated in Section 6.3, the cumulative QoS is subject to two major QoS related constraints: bandwidth and delay bound. The effect of bandwidth, in particular upload bandwidth, has been investigated, and this subsection considers how varying the associated delay bound may affect the cumulative QoS. For simplicity, we use the metric of hop count to represent different delay bounds: e.g., in Figure 6.6, a graph labeled “X” stands for a delay bound of X, i.e., packets that travel more than X hops will be discarded. As shown in Figure 6.6, the cumulative QoS does vary with different delay bounds, but it is not as sensitive to upload bandwidth. The reason is that delay, as a very important parameter, has been deliberately taken into consideration in the parent selection procedure; as long as there is sufficient upload bandwidth, most peers can find alternative routes satisfying the associated delay bound. Nevertheless, the admission control algorithm does slightly improve the cumulative QoS. This can be attributed to the fact that the proposed admission control algorithm effectively reduces the chance of all peers being rejected, thus leading to an improvement of the cumulative QoS.
CHAPTER 6. ADMISSION CONTROL FOR APPLICATION LAYER MULTICAST

![Graphs showing Quality of Service under different delay constraints with and without admission control.](image)

(a) With admission control

(b) Without admission control

*Figure 6.6: QoS under different delay constraints.*
How Lifetime Distribution Reacts to An Admission Control Algorithm

It is very interesting and worth investigating how peers’ lifetime may affect and react to the proposed admission control algorithm. Figure 6.7 shows that the proposed admission control algorithm can greatly enhance peers’ perceived QoS under different lifetime distributions. The ability of the admission control algorithm to fast amplify system capacity is very beneficial under most lifetime distributions: a maximum of 20% improvement can be achieved.
Figure 6.7: QoS under different lifetime distributions.
CHAPTER 6. ADMISSION CONTROL FOR APPLICATION LAYER MULTICAST

Figure 6.8: Rejection rate comparison.

(a) With admission control

(b) Without admission control
Figure 6.9: Rejoin frequency comparison.
Admission Control Does Reduce the Number of Rejections

The main purpose and the most appealing part of an admission control algorithm is its ability to optimize use of the resources, maximizing the number of peers accepted and ensuring the desired QoS for those accepted users. Figure 6.8 illustrates that the proposed admission control algorithm can greatly reduce the chance of rejection for participating peers. Comparing Figure 6.8(a) with Figure 6.8(b) reveals that the system equipped with the proposed admission control algorithm can decrease the frequency of rejection by an order of magnitude. This can be attributed to the ability of the proposed algorithm to make efficient use of the scarce upload bandwidth resource: by accepting peers with abundant upload bandwidth and rejecting some of the peers with poor upload bandwidth, the system’s capacity is quickly enhanced, leading to less overall rejections.

The overlay formed by participating users is highly dynamic: users join and depart at will, the overlay is constantly changing in the sense that once an on-tree peer leaves, all of the downstream nodes will initiate the rejoin operation, trying to re-connect to the multicast tree; in the meantime, peers periodically migrate to higher layer by switching parents [Rong et al., 2006b;a]. At any instance, there are many parent switching and rejoining activities. The proposed admission control is also effective in decreasing the chance of rejection for these rejoining operations. Figure 6.9 clearly demonstrates this effect. A close examination of Figure 6.9(a) and Figure 6.9(b) shows that the admission control algorithm is able to reduce the number of rejections for peers’ rejoining by as much as 50%. It is interesting to compare Figure 6.8(b) with Figure 6.9(b), the difference once again proves that our admission control algorithm is effective: due to peers’ periodical parent switching operation, the multicast tree is constantly under adjustment. Despite the participating peers’ high churn rate, due to the optimization characteristics of the admission control algorithm, every parent switching or rejoining operation represents an improvement to the multicast tree. The cumulative effect is that the rejection rate of the rejoining is much lower than the rejection rate when peers first join the session.

To conclude, the proposed admission control algorithm is quite effective, although it is only a simple heuristic-based approximation of the optimal solution. Nevertheless, the effectiveness and simplicity make it very promising. Intergraded into our previous work, it further bridges the performance gap mentioned in the beginning of this chapter.
6.7 Conclusion

An important research question, the admission control problem, is raised and formulated in the context of Peer-to-Peer (P2P) streaming. The similarity and difference to other related problems is clearly presented; in particular, the similarity to a closely-related problem, the stochastic knapsack problem, is identified. The identified admission control problem is NP-hard and there is no well-defined structure of the optimal solution: i.e., it is computationally very difficult to obtain the exact optimal solution.

In the P2P paradigm, the system considered is highly dynamic and the admission decision has be to be made in a real-time fashion. Taking the complexity of the problem into consideration, an heuristic-based approximation algorithm is proposed. A new profit-to-volume ratio is derived according to the unique characteristics of the P2P streaming system. Then the requesting peers are ordered and admitted in a non-decreasing way according to this newly-defined ratio.

Extensive simulation was undertaken to validate the proposed algorithm. Simulation results demonstrate the effectiveness of the algorithm. Integrated with our previous work, the proposed algorithm successfully bridges the performance gap, paving the way for an effective implementation of the P2P streaming paradigm.
Chapter 7

Conclusion

This thesis investigated Application Layer Multicast or Overlay Multicast as an alternative to IP multicast to offer timely deployment of video streaming over the Internet. The issues of membership management, reliability enhancement, QoS, and admission control have been addressed respectively. While being a solution to the original development and deployment problems with IP multicast, overlay multicast are facing many new challenges. To be more specific, the following research questions have been addressed in this thesis:

1. What is a cost-efficient way to maintain the multicast group formed by participating users under the conditions of high rate of user churn and heterogeneous users?

2. Can reliability perceived by participating users be improved with a manageable overhead?

3. Can heterogeneity among peers be handled and accommodated in a graceful way? Can we fit QoS into the existing schemes?

4. Can better admission control be achieved in the highly dynamic environment of overlay multicast?

7.1 Membership Management

Overlay multicast falls into the group communication model, and the very first step for any group communication application is to have a membership management scheme. The membership management scheme concerns how the group is defined and maintained, in the highly dynamic overlay multicast environment, as participating users join and depart at will.
CHAPTER 7. CONCLUSION

So the timely notification of the changes of users’ joining and departure is vital to the success of any overlay multicast application. Furthermore, the overhead incurred by this membership management scheme should be manageable, i.e., it should be done in a cost-efficient way.

Gossip-based membership management algorithms were identified as a suitable method to overcome the aforementioned difficulties: its random nature in terms of gossip target selection enables it to cope with the random failures caused by users’ dynamic join and departure; its distributed nature and its inherent simplicity lead itself easily to the overlay multicast context. However, the redundancy resulting from random gossip target selection seems like a double-edged sword: it brings in the resilience and enables the membership management scheme to work properly even in a highly dynamic environment; it also incurs a large amount of overhead, in particular on the core network elements, e.g., routers of the backbone links as identified in the simulation.

It is believed that optimization, in terms of overhead reduction, can be obtained. For example, the disseminated gossip messages containing users’ information have varying importance for different users. Due to the stringent delay requirement, it is not optimal or even impossible for a user, say user \( a \), located in Australia to obtain the streaming data from another user, say user \( b \), in US. This implies that gossip user \( b \)'s information to user \( a \) is a waste of the resources. Based on this belief, a single index is used to reflect this varying importance among individual users, taking delay, bandwidth (both upload and download bandwidth), and other parameters into consideration. A new gossip-based algorithm is proposed in Chapter 3, making use of the index. The idea is to choose gossip target preferentially based on the single index, which \( \text{per se} \) is dynamically adjusted, rather than the uniform random approach in traditional gossip-based algorithm.

Experimental results show that a maximum of 50% reduction can be achieved in terms of network overhead on core network components, such as backbone links and attached routers, without sacrificing resilience. It shows the reduction in terms of member management overhead is achievable, and the devised membership management is suitable for overlay multicast.

7.2 Reliability Enhancement

Overlay multicast, especially single-tree-based schemes, suffers from the inherent dynamics. Due to its severless nature, participating peers fetch data from existing on-tree ones, and this “parent-children” relationship is modeled as a multicast tree structure, which is quite
vulnerable to peers’ dynamic departures or failures.

It is discovered that peers’ lifetime follows a heavy-tailed distribution and exhibits the Used-Better-Than-New (UBTN) feature, and this seems to be a good starting point to enhance reliability for overlay multicast. Since the UBTN behavior simply states that a peer with a longer lifetime tends to be more reliable, and based on this fact, a reliability-centric multicast tree construction algorithm is proposed in Chapter 4.

In order to reflect participating peers’ lifetime qualitatively, a rank, derived from its lifetime, is associated with each peer. Participating peers are organized into a hierarchy in such a way that it reflects their relative stabilities (represented by their “ranks”), rather than their geographical proximities or other criteria; then a multicast deliver tree is constructed out of the hierarchy. In addition, peers periodically update their ranks and make attempts to be connected to more stable peers. In this way, peers that are potentially more stable eventually “climb” up and are placed close to the streaming source, and most dynamics caused by ungraceful departure of peers are confined within the lower end of the multicast tree.

A minimum reduction of 50% can be achieved in terms of service disruption frequency for most peers, and consequently, peers’ perceived QoS are greatly improved. This proves that it is possible to improve reliability for overlay multicast schemes.

### 7.3 QoS-aware Tree Construction

The reliability-centric multicast tree construction algorithm proposed in Chapter 4 can be further improved by taking various Quality-of-Service (QoS) parameters into consideration, such as access bandwidth, network distance, and reliability.

It is built upon our previous work on reliability enhancement, i.e., peers are organized into hierarchy according to their potential reliability; the difference lies in a new parent selection algorithm, which is derived from Dijkstra’s shortest path algorithm, taking peers’ access bandwidth, network distance and other realistic parameters into consideration. It works by finding the prospective parents for each peer with maximum upload bandwidth while satisfying the delay and reliability requirements. The algorithm is presented in detail in Chapter 5.

Extensive simulation reveals that the proposed approach can actually accommodate the inherent heterogeneity, and most of the participating peers are able to receive satisfactory service.
7.4 Admission Control

It is found, through our work presented in Chapter 5, that there exists a large performance gap that needs to be filled, and this could be attributed to the fact that peers are admitted into the system in order of arrival, rather than from a performance perspective.

The uniqueness of overlay multicast is that the data replication and forwarding functionalities are all undertaken by participating peers. In other words, all the resources, such as bandwidth, CPU cycles, storage, and so on are contributed and shared by all participating peers. The availability of those resources has a huge impact on the performance of the resulting overlay multicast applications. For example, admitting a large number of peers with low upload bandwidth at the early stage of the streaming session greatly limits the availability of bandwidth, and it may lead to a poor performance. The difficulty is that peers join and leave the session at will, the decision to admit a particular peer might have a huge impact for the latecomers, and this is formulated as a stochastic knapsack problem. It is an NP-hard problem, and due to its inherent complexity and the stringent delay requirement, an heuristic-based algorithm is proposed to approximate the solution to this stochastic knapsack problem in Chapter 6. The idea is to decompose the original stochastic problem into smaller problems, and let each peer send bids to prospective parents, and let each parent solve an instance of the knapsack problem.

The proposed admission control algorithm is validated through simulation and is able to reduce the rejection rate by as large as 50%, and this proves that this it is possible to further boost the performance of our overlay multicast scheme.

7.5 Future Work

While this thesis addressed several of the issues of ALM, it was not possible to explore all aspects of the issues that have been addressed in this thesis. Instead, they are described here as future work.

The reliability-centric multicast tree construction algorithm presented in Chapter 4 is built upon the assumption that peers’ lifetime exhibits the Used-Better-Than-New (UBTN) feature. Although it is shown that it works for other lifetime distributions as well through simulation, the simulation results also show that it has degraded performance under memoryless lifetime distribution, such as exponential distribution, especially those with a smaller expected lifetime. It is believed that the algorithm could be improved to account for memory-
less lifetime distributions.

The performance of the parent selection scheme presented in Chapter 5 heavily depends on the membership management algorithm: the more accurate and timely information it can provide, the better is the performance of the parent selection scheme. It would be very interesting to investigate the tradeoffs between them, such as overhead, efficiency, etc.

Due to the inherent computational complexity, the admission control problem investigated in Chapter 6 is only provided with an heuristic-based solution. Although it is simple and effective, a more sophisticated scheme would be worth investigating in pursuit of further performance boost.

Last but not least, all the work presented in this thesis were validated through simulation and mathematical analysis. Although doing so is sufficient to capture the characteristics of the problem and validate the effectiveness of the proposed approaches. It would be valuable to apply the research outcomes to real-world implementation, e.g., PlantLab. This is the major future work that we consider needs to be carried out.
Bibliography


BIBLIOGRAPHY


135 (March 23, 2008)


BIBLIOGRAPHY


139 (March 23, 2008)
BIBLIOGRAPHY


BIBLIOGRAPHY


