Peer-to-Peer Live Media Streaming: Design and Implementation

ピアツーピア型ライブメディアストリーミング手法の提案と実装に関する研究

by

POO KUAN HOONG

A DISSERTATION
submitted to the Graduate School of Engineering,
Department of Computer Science & Engineering,
Nagoya Institute of Technology,
in a partial fulfillment of the requirements for the degree of
DOCTOR OF PHILOSOPHY
written under the supervision of
Professor Hiroshi Matsuo

NAGOYA INSTITUTE OF TECHNOLOGY
Nagoya, Japan
February 2008

This work was created using the LaTeX 2$_\varepsilon$ system, MiKTeXv2.5, together with the software WinEdt v5.5, BibTeXMng v5.0, MathType v5.2c and LaTable v0.7.2. The illustrations were created using the software MATLAB v7.0, Microsoft Visio 2003, Adobe Illustrator CS2 and GSview v4.8.

Abstract

Peer-to-peer (P2P) file sharing has become increasingly popular, accounting for as much as 70% of Internet traffic by some estimates. Recently, we have been witnessing the emergence of a new class of popular P2P applications, namely, P2P audio and video streaming. While traditional P2P file distribution applications target elastic data transfers, P2P streaming focuses on the efficient delivery of audio and video content under tight timing requirements. In these applications, each node independently selects some other nodes as its neighbors and exchanges streaming data with neighbors. In this dissertation, we propose and investigate a full distributed, scalable, and cooperative protocol for live video streaming in an overlay peer-to-peer network. Our protocol, termed *P2P Unstructured Live Media Streaming (PALMS)*, makes use of combination of push-pull scheduling methods to achieve high performance (in term of delay, stream continuity, cooperation, etc.). In live P2P streaming, the media stream is a continuous flow of media data encoded from the streaming server. Media content generated must be delivered to participating nodes under a tight time constraint. Nodes should be able to receive media content before the playout deadline or the media content will be considered obsolete and discarded. To address these problems, we propose two methods that are built on an unstructured overlay network. Our first method is based on the combination of push-pull scheduling methods that effectively enhances the delivered streaming quality. Our second method is the extension of the first method which is based on two-layer super-peers unstructured overlay network that consists of super-peers and ordinary peers. The main contribution of our proposed methods is that it effectively reduces the end-to-end streaming delay and in turn results better delivered quality. Furthermore, with the implementation of two-layer based overlay network that consists of super-peers and ordinary peers, PALMS is able to leverage on the heterogeneity of bandwidths and simplify the complexity of transmission service with the existence of super-peers and in turn shows better Quality of Service (QoS). We extended PALMS's push-pull protocol into two-layer super-peer based overlay streaming network - PALMS-SP. The super-peer approach is able to organize the P2P overlay as a trade-off solution that merges the client-server model relative simplicity and the P2P autonomy and resilience to crashes. We have extensively evaluated the performances of PALMS and PALMS-SP. Our experiments demonstrate that PALMS and PALMS-SP achieve good streaming quality. We believe that our proposed methods are viable and efficient methods for adaptive Internet live media streaming application. In the future, we can expect to see a greater proliferation of Internet live media streaming applications that is leverages on the power of connected computers and Internet's capabilities.

Keywords: *peer-to-peer, streaming, overlay, push-pull, super-peer*

抄録

Peer-to-peer (P2P) ファイル共有は急速に普及し，いくつかの推計によれば，インターネットのトラフィックの 70%に達している．特に近年においては，P2P による音声・映像ストリーミングに代表される，新しい種類の P2P アプリケーションが出現している．従来の P2P ファイル分散アプリケーションが柔軟なデータ転送を目標とすることに対して，P2P ストリーミングは，厳密なタイミングが要求される状況下での，音声や映像の効率的な配信に焦点を当てている．これらのアプリケーションでは，各ノードは自律分散的に，他ノードを近傍ノードとして選択し，その近傍ノードとストリーミングデータを交換する。本論文では，オーバーレイ P2P ネットワークにおけるライブ・ビデオストリーミングのための，完全に分散しかつ，スケーラブルな協調プロトコルを提案する．提案手法である P2P Unstructured Live Media Streaming (PALMS) は，ストリーミングにおける遅延，連続性，協調処理などのパフォーマンス向上のために，push-pull スケジューリング手法を適用する. ライブ P2P ストリーミングでは，メディアストリームは，ストリーミングサーバでエンコードされるメディアデータの連続的なフローである．生成されるメディアコンテンツはタイトな時間的制約の下で，参加中のノードに配信されなければならない．各ノードはメディアコンテンツを再生時刻までに受信しなければならず，さもなければ，メディアコンテンツは間に合わなかったものとして破棄される．この問題に対処するために，unstructured 型のオーバーレイネットワーク上に構築される 2 つの手法を提案した．第一の手法は，push-pull スケジューリング手法を組み合わせたものに基づき，配信されるストリーミングの品質を効率的に高めた．第二の手法は，2 層からなる super-peers unstructured 型のオーバーレイネットワーク上に構築した．このオーバーレイネットワークは，スーパーピアと通常のピアから構成される．提案手法の主要な貢献は，エンドツーエンドのストリーミングの遅延を効率的に削減すること，およびその結果としての，良好な品質の配信である．また，PALMS の push-pull プロトコルを，2 層からなるスーパーピア・オーバーレイストリーミングネットワーク PALMS に拡張した．スーパーピアを用いるアプローチにより，クライアント・サーバモデルにおける容易さと，P2P の自律性と障害に対する回復性との，トレードオフを解決する P2P オーバーレイを組織することが可能となった．さらに，スーパーピアと通常のピアから構成される 2 層のオーバーレイネットワークの実装により，PALMS-SP は，帯域の異種性を活用して，スーパーピアの存在を伴う転送サービスの複雑性を簡素化し，より良い Quality of Service (QoS)を示した．PALMS の性能は実験により広範囲に評価し，その結果は PALMS と PALMS-SP が良好なパフォーマンスを達成することを示し．提案手法が，インターネット上の適応的なライブ・メディアストリーミング・アプリケーションに対して，実現可能な効率的手法であることを確認した．提案手法を用いることにより、近いに将来，回線に接続されたコンピュータの処理能力と回線の通信容量にもとづく，インターネットにおけるライブ・メディアストリーミング・アプリケーションの広範囲な普及が期待できる．

キーワード： peer-to-peer, streaming, overlay, push-pull, super-peer

Acknowledgements

Accomplishing a Ph.D is without doubt a significant achievement in one's life. Although it is habitually accredited to a single person, its author, in most cases there is a number of people who have contributed one way or another to achieve this goal. Having completed my thesis, it is time to give credit to some of the people that played a role in my Ph.D.

Undoubtedly, I should start with the person that deserves the biggest share of the credit. I would like to express my deep and sincere gratitude to my supervisor, Professor Hiroshi Matsuo, Matsuo & Tsumura Laboratory, Department of Computer Science and Engineering, Nagoya Institute of Technology. His wide knowledge and logical way of thinking have been of great value for me. I am also grateful for his understanding, encouraging and personal guidance that have provided a good basis for the present thesis.

I also would like to express my deepest appreciation to Professor Akira Iwata and Professor Naohisa Takahashi as my second and third panel of my dissertation. I am deeply thankful for their guidance and wisdom throughout my research period.

During this work I have received assistance, advices, supports, and companionship from many of my university mates for whom I have great regard, and I wish to extend my warmest thanks to all those who have helped me with my work in the Department of Computer Science and Engineering and everyone from Nagoya Institute of Technology, namely Dr. Toshihiro Matsui, Dr. Mauricio Kugler, Heethaka Pradeep Ruwantha de Silva, Jiang Wei, Huang Di, Kothakapu Srikanth Reddy, Ruben L. Menchavez, Rajendran Barath Kumar and Kim Me Ka.

I would also to express my thanks all my Malaysian friends in Nagoya and throughout Japan which have given me unlimited supports and encouragement during my four years of research. A big thank you to Choo Yap Yin, Tan Foong Ming, Seow Phei Shan, Leow Chi Cheng, Yeoh Fei Yee, Low Yu Ping, Puay How Tion, Chan Lih Shing, and Lee Ai Ling.

Life in Japan is challenging and sometimes down right stressful. I would like to thank all my friends back home, Malaysia for providing me a great deal of moral support and belief for all these years I am in Japan. Many thanks to Yap Moi Hoon, Ryan Kawailani Ozawa, Pee Chih Yang, Akhtaruddin bin Hariri, Lee Chin Sin, Tah Jia Ren, Wong Ya Ping, Prof. Chuah Hean Teik, Dr. Jacob K. Daniel, Dr. Ewe Hong Tat, Yu Yee Mei, and many more.

This dissertation is dedicated to my family. I thank my parents, Loo Sek Engan & Poo Hee Yuen, my brother Poo Kuan Meng, and my two sisters, Poo Wae Cheng and Poo Wae Wae for their wisdom of life and unconditional belief in me. Their love, which transcends time and distance, is the eternal source of my strength. I am grateful for all the sacrifices my family has made for me and make life meaningful.

The financial support of the Ministry of Education, Culture, Sports, Science and Technol-

ogy, Government of Japan, for providing me a scholarship during the doctoral course Japan and Hori Information Science Promotion Foundation, Japan, for the grant during the year of 2006 are gratefully acknowledged.

Enjoy a simple life.
Don't complicate it.

Poo Kuan Hoong

We are what we think. All that we are arises with our thoughts.
With our thoughts, we make our world.

The Buddha

Contents

List of Figures

List of Tables

List of Abbreviations

ACK	-	Acknowledgement
ADSL	-	Asymmetric Digital Subscriber Line
ALM	-	Application-Layer Multicast
AS	-	Autonomous System
CDF	-	Cumulative Distribution Function
CDN	-	Content Distribution Network (plural: CDNs)
DHT	-	Distributed Hash Table
DNS	-	Domain Name System
FEC	-	Forward Error Correction
FIFO	-	First In, First Out
FTP	-	File Transfer Protocol
ID	-	Identifiers (plural: IDs)
IP	-	Internet Protocol
kbps	-	kilobit per second
LAN	-	Local Area Network (plural: LANs)
LRU	-	Least Recently Used
MDC	-	Multiple Description Coding
NACK	-	Negative Acknowledgement
P2P	-	Peer-to-Peer
PDF	-	Probability Density Function
QoS	-	Quality of Service
RAM	-	Random Access Memory
RTP	-	Real-time Transport Protocol
RTT	-	Round Trip Time
SHA	-	Secure Hash Algorithm
TCP	-	Transport Control Protocol
TCP/IP	-	Transport Control Protocol/Internet Protocol
TTL	-	Time-to-Live
UDP	-	User Datagram Protocol
VCR	-	Videocassette recorder
VOD	-	Video On Demand
WAN	-	Wide Area Network

CHAPTER 1

Introduction

Failure is the opportunity to begin again,
more intelligently.
Henry Ford.

Peer-to-peer (P2P) file sharing has become increasingly popular, accounting for as much as 70% of Internet traffic by some estimates. Recently, we have been witnessing the emergence of a new class of popular P2P applications, namely, P2P audio and video streaming. While traditional P2P file distribution applications target elastic data transfers, P2P streaming focuses on the efficient delivery of audio and video content under tight timing requirements. Still in its infancy, both live and on-demand P2P streaming have the potential of changing the way we watch TV, providing ubiquitous access to a vast number of channels, personalizing your TV experience, and enabling roaming TV services. For a long time, traditional approaches that are client/server based e.g., Akamai [aka] have been used for streaming multimedia applications over the Internet.

Over the past few years, P2P networks have emerged as a promising approach for distribution of multimedia content over a network. Some P2P network related research is by the following authors [GST03], [HHR+03], [PWC03], [YM04], and [ZLLY05]. One form of P2P network, the peer-to-peer overlay, offers a promising approach to support one-to-many multimedia streaming applications without any special support from the network, called P2P streaming. The basic building blocks for P2P streaming, called *nodes* or *peers*, are no longer passive receivers of data but can act both as clients and servers at the same time. Stream data are simultaneously received, played, and passed to other connected peers. The goal of P2P streaming mechanisms is to maximize delivered quality to individual peers in a scalable fashion despite the heterogeneity and asymmetry of their access link bandwidth. An effective P2P streaming mechanism depends on the effective utilization of the outgoing bandwidth of most participating peers.

This dissertation is an effort to overcome the problem of delivering live streaming media to potentially large number of concurrent clients. Our solution emphasizes on the building an adaptable system for live media streaming. We address two key challenges: the design of a

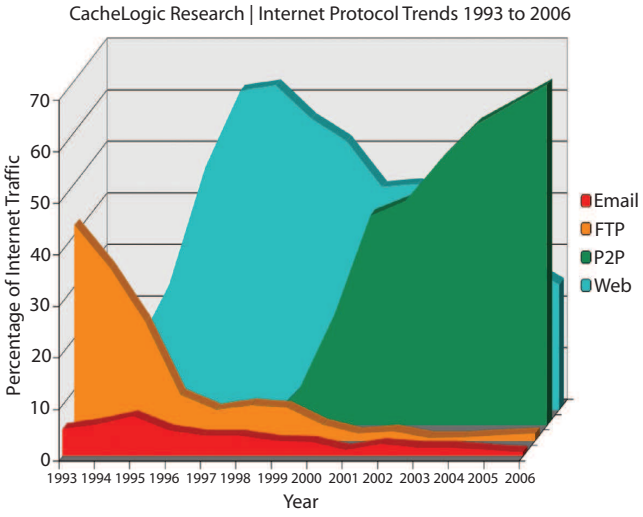


Figure 1.1: An overlay multicast for a P2P media streaming system.

live media streaming system and a set of protocols for efficient live media streaming and good Quality of Service (QoS) of delivered media.

1.1 Motivation

In the last few years, the Internet has been used for an increasing amount of traffic stemming from the emergence of multimedia applications which use audio and video streaming. This increase is expected to continue and be reinforced since access technologies such as Asymmetric Digital Subscriber Line (ADSL) and cable modems enable residential users to receive high-bandwidth multimedia streams. One specific application which will be enabled by future access technologies is live media streaming. Live media streaming allows users to watch a certain video in real time. The challenges of providing live media streaming in the Internet are manifold and require the orchestration of different technologies.

In live media streaming, the media stream is a continuous flow of media data encoded from the streaming server. Media content generated must be delivered to participating nodes under a tight time constraint. Nodes should be able to receive media content before the playout deadline or the media content will be considered obsolete and discarded. The key challenges for a peer in P2P live media streaming applications include:

1. locate supplier peers with the desired media segments/packets before the playout time deadline

2. choose supplier peers that are likely to provide good performance for playback

3. manage parallel download and upload to connected neighbor nodes

4. managing the connections with connected peers in the network due to the dynamicity of peers joining and leaving

1.2 Objectives

The main objective of this dissertation is to "*design bandwidth scalable, robust real-time multimedia streaming overlay P2P protocols for the best-effort Internet.*" In order to make this goal manageable, we identified a number of sub-problems and objectives that naturally lead to the solution of the main problem:

- carry out an extensive performance study of constant-bitrate video streaming in the existing Internet and apply the learned lessons to the design of our streaming protocols;
- analyze the scalability of rate-based congestion control in real-time applications and design new methods that can scale to a large number of users;
- study the performance of real-time end-to-end bandwidth estimation methods and their applicability to multimedia streaming and congestion control.

1.3 Contributions

The major contributions of this dissertation are as follows. We propose an overlay peer-to-peer unstructured streaming model, which works as the unified framework for various overlay based streaming applications. Our application example in this dissertation is the live multimedia distribution. Based on our results obtained from simulations and testing, we exhibit the great potential of overlay-based solution at saving server load and network bandwidth consumption compared to IP-multicast-based solutions. We note that such a foundation works for all kinds of distribution applications such as large-volume file downloading, teleconferencing, etc. Our proposed algorithms can also be employed as part of the evaluation tools to study interesting questions such as the impact of IP routing at constraining the maximum throughput of overlay multicast.

1.4 Topics not covered in this dissertation

Peer-to-peer systems is a recent topic and many new issues arise, as many new possibilities are being discovered, evaluated and tested. There are at least two topics that we consider important in regard of P2P networks, but not covered in this dissertation:

- **Security**: We do not cover basic security services in Peer-to-Peer networks. The techniques that allow avoiding the pernicious effects of a badly behaving (possibly colluding) group of peers are out of the scope of this work. However, where possible, we succinctly address some security issues by suggesting how the P2P system may detect uncooperative peers. The absence of a centralized point of failure in a P2P network makes impossible a centralized *trusted third party* (e.g., a certification authority), which is the base for classic authentication, non-repudiation and confidentiality in the Internet [MOV96], [Sch95]. There are efforts to distribute the certification authority among a number of machines through threshold cryptography. Threshold cryptography allows for the cooperative verification of a signature by different collaborative entities, making the collusion among peers more difficult to some extent. EigenTrust [KSGM03] is a distributed approach for peer reputation in a given P2P system.
- **Copyright**: Copyright law applies to virtually every form of expression that can be captured (or, to use the copyright term of art, "fixed") in a tangible medium, such as

on paper, film, magnetic tape, hard drive, optical media, or (arguably) in RAM. Songs, books, photographs, software, and movies are all familiar examples of copyrighted works. Copyright law reserves certain rights exclusively to the owner of the copyright, including the rights to reproduce, distribute, and publicly perform the work.

The nature of file-sharing technology inevitably implicates copyright law. First, since most digital files are "fixed" for purposes of copyright law (whether on a hard drive, CD, or possibly in RAM), the files being shared generally qualify as copyrighted works. Second, the transmission of a file from one person to another generally results in a reproduction. Copyright owners have also argued that digital transmissions can qualify as a distribution, and possibly a public performance (in the world of copyright law, "public performance" may include the act of transmitting a copyrighted work).

There have been three major court opinions that have applied indirect liability theories to companies that distribute P2P software: A&M Records v. Napster [nap], In re Aimster Copyright Litigation [aim], MGM v. Grokster [mgm]. Unfortunately, these three cases are not entirely consistent in their analyses. The law continues to evolve, and other courts may further muddy the waters in the years to come.

The legal issues associated with P2P are quite complex and need to be dealt with on an international scale. There are millions of people with access to the Internet who could all potentially use some P2P technology to directly infringe copyright law and there appears to be quite a large group who do. So it would be a huge task for any artist/retailer/company or even a group of such people to get such a large number of people convicted of copyright infringement.

1.5 Dissertation outline

This research produced several conferences and journal papers and this work is a compilation of those papers' propositions and results. The papers' material was rearranged in a more comprehensive order and a more extensive review of the literature was made.

Chapter 2 presents a review about the basic concepts of streaming protocols. Basic concepts of network model and overlay graph are given to clarify the notation and give the definition of the theorems and rules used along the paper, as well as to introduce about streaming related works and problems. Finally, the technical challenges and research opportunities are presented at the end of the chapter.

Chapter 3 overview of the proposed P2P Unstructured Live Media Streaming (PALMS) architecture. We discuss the various components that form the foundation of distributing live media streaming over unstructured P2P networks. We continue the discussion of the pros and cons of the PALMS distribution model and present the details of some of the components that underlie the architecture.

In order to clarify the experimental framework and simulation setup used in the following chapters, chapter 4 describes the simulation setup, simulation parameters and performance metrics used along the experiments on the following chapters.

Chapter 5 presents the results of simulations on PALMS. We examine the impact of heterogeneous bandwidth and free-riders on the performance of PALMS streaming. We also study the three performance metrics of interest: Delivery quality, Delivery latency and Data overheads. We compare the streaming performance of PALMS with other existing streaming systems.

As the main modification presented in chapter 6 is the introduction of two-layer super-peer overlay network, PALMS-SP. With the addition of super-peers layer and ordinary peers layer, the super-peer approach is able to organize the P2P overlay as a trade-off solution that merges

the client-server model relative simplicity and the P2P autonomy and resilience to crashes. Simulations experiments were conducted and a comparison of the effectiveness and streaming quality of PALMS and PALMS-SP with other existing streaming protocols. Based on the results, the performances of PALMS-SP using the two-layer super-peer overlay network deliver better streaming quality and more robust.

Finally, chapter 7 presents the main conclusions of this research, discuss the experimental results presents suggestions for future work. □

CHAPTER 2

Background and Related Works

The whole problem with the world is that
fools and fanatics are always so certain of themselves,
but wiser people so full of doubts.
Bertrand Russell.

In this chapter, we first provide background information about the basic concepts and terminology of peer-to-peer (P2P) networking, as a preface for understanding how *P2P Unstructured Live Media Streaming (PALMS)* is able to provide better solution for streaming multimedia over the Internet. Next, we discuss the challenges of streaming service over P2P networks as they inherent instability and unreliability. Finally, we look at several streaming protocols and projects that have been proposed as P2P overlay streaming.

2.1 Basic Concepts and Terminology

Although basic knowledge in networks is a basic requirement for the network studies, terms and concepts are lost or misunderstood when classification algorithms are naively applied. This section do not intend to deeply explain statistics, but to refresh and clarify those concepts in order to facilitate the comprehension of the further chapters.

2.1.1 Network Model

We consider a network (V, L), where $V = \{v_1, v_2, ..., v_{|v|}\}$ represents the set of nodes. L is the link set, where $l = (v_a, v_b) \in L$ represents a physical link from node v_a to v_b. We assume each physical link to be directed, which is the case for most real networks. However, all the algorithms presented in this proposal also apply for the undirected network model.

Consider k multicast sessions $M_1, M_2, ..., M_k$. Each session $M_i \subseteq V$. It consists of one server $S(M_i)$ and several receivers $R_1(M_i), R_2(M_i), ..., R_{|M_i|-1}(M_i)$. Normally, receivers grouped in one session have common interests with the data disseminated by the server. A network node $v_i \in V$ can belong to several sessions.

2.1.2 Overlay Graph

Definition

Within each session M_i, an overlay graph $G_i = (M_i, \xi_i)$ is formed. G_i is a directed virtual graph, where an edge $e = (R_m(M_i), R_n(M_i)) \in \xi_i$ represents a data relay path from $R_m(M_i)$ to $R_n(M_i)$. The path corresponds to the unicast route between these nodes in the physical network L. The edge weight of e represents various metrics of this route, e.g., communication cost, available bandwidth, traffic amount, under different application settings. An overlay graph can take the following forms:

- **Complete Graph**. If the physical network L is not partitioned, there exists a route between any pair of nodes, which indicates that a node can relay data to any other nodes of the multicast session, except for the sender. Therefore, the overlay graph can be a complete graph. Such a graph configuration suits well for scenarios where richly connected overlay paths are required, and the session size is limited, such as resilient routing [ABKM01].

- **Overlay Mesh**. Since maintaining a complete graph is not scalable as the size of the multicast session increases, one can choose to construct the overlay graph as only a subgraph of the complete graph, i.e., a mesh [CRZ00].

- **Data Dependency Graph**. In previous overlay graph examples, an edge exists between two nodes simply because we can physically transfer data from one to another. Such a definition is suitable for application scenarios where distribution is simultaneous, such as live broadcasting, teleconferencing, etc. In other scenarios where distribution is asynchronous, such as peer-to-peer file sharing or on-demand streaming, there exists a data path from node R_m to R_n, only when the data requested by R_n is locally available at R_m. In this case, the edge (R_m, R_n) not only represents the data path, but also the data dependency relation between R_m and R_n.

2.1.3 Overlay Graph Management

The overlay graph changes dynamically: (1) when a new receiver arrives, a new node is inserted into the graph; (2) when a receiver quits, its corresponding node is removed from the graph. The management of overlay graph can be *centralized* or *distributed*. In the centralized approach, the manager node of the multicast session, normally the server, has the global view of the overlay graph. In order to maintain such a global view, each new node joining the multicast session must report to the server. In return, the server feedbacks to the new node with information of existing nodes in the overlay graph, and arbitrate the joining node to setup new edges with all, or some of them. Likewise, when a node leaves the multicast session, or fails, the server must be notified to remove edges attached to this node. In the distributed approach, each node only maintains information of its overlay graph neighbors. Nodes are interconnected via various distributed routing mechanisms, such as link-state protocol, distributed hash table, etc. These mechanisms can effectively capture and propagate the dynamic change (node joining and leaving) of the overlay graph to its members.

2.1.4 Peer-to-Peer System

P2P systems build on many concepts from the area of distributed systems: Algorithms for redundancy, load balancing, overlay networks or resilience are used as building blocks of these systems. However in what sense are these P2P systems that are considered to be new?. What

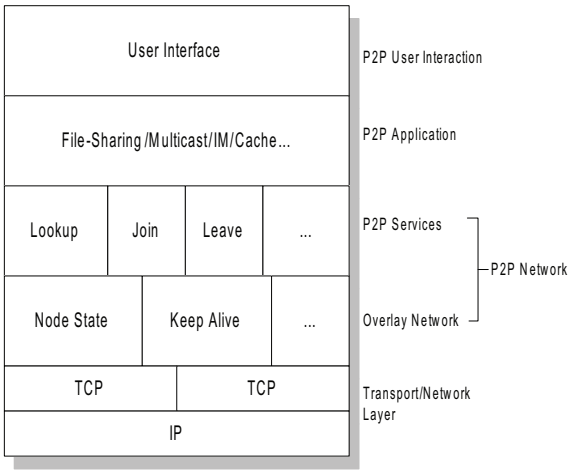


Figure 2.1: The Peer-to-peer Protocol Stack.

are the new characteristics that make them different enough so as to treat them separately from existing distributed systems?.

Systems behaving in a peer-to-peer fashion pre-existed the applications we call *Peer-to-Peer* today. In the very beginning of the Internet, all computers were peers, because no Domain Name System (DNS) existed and a host had to know another in order to establish a communication with the latter. So there are so much of talks and researches related to P2P these days?. This is because the kind of systems we consider in this thesis could not have been imagined when that primitive *"Peer-to-Peer"* Internet existed. Only today we can start to build systems of millions of interconnected hosts. P2P systems are a consequence of the development of the Internet.

It is only nowadays that such a system is feasible. As such, peer-to-peer systems present their own characteristics:

- They are composed of a very large number of hosts, and thus their properties must scale gracefully with an increasing number of participants. Millions of users can be considered a normal scenario
- The computing infrastructure is provided by the users' machines, which can be powerful but often unreliable. Commodity hardware is assumed for all the participating machines. Any given machine (peer) can fail or disconnect at any point in time.
- We consider the justification of the requirement for a minimum bandwidth for some P2P applications to work. However, for most or all peers, bandwidth should be treated in general as a scarce resource. Thus a sense of economy must be applied to all communication not involving the actual exchange of information among users.
- Users are rational. They know that they must collaborate in order to benefit from the system. However, most users will not offer resources without an interest motivating them. In real life P2P networks, a few peers offer resources just for the sake of it, while others may not collaborate at all. In the middle of this range of behaviors we find normal users, which cooperatively build the common infrastructure, motivated by their own interest.

In general, a P2P system is (roughly) composed of a substrate to allow communication among peers, the algorithms locate resources, and an application running on top of the distributed environment. We now define the key elements in a P2P network.

In figure 2.1 we see a detailed P2P protocol stack. Note that all the P2P layers are situated at the application layer of the OSI model. Note that the P2P network is formed by two closely related layers: the Overlay Network and the P2P Services. The *P2P network* is formed by two closely related layers: the *Overlay Network* and the *P2P Services*.

- *Overlay network:* An Application-level routing scheme among the nodes of a distributed application. Packets traveling from one node to another are routed by the network layer (OSI level 3), but the routing decision (i.e., which is the destination node at each hop) is explicitly made at the application level on each end node. This is the reason why overlay networks are also called Application-level networks. A solid example is the *Resilient Overlay Network (RON)* [ABKM01] : a group of multi-homed nodes assure continuous communication even in the presence of network failures, by routing packets through an application-level overlay path that avoids the failed network or link. Overlay networks are often represented by a directed graph $(X; U)$, where X is the set of end nodes in the overlay and U is the set of application-level links among the nodes in X. The $(X; U)$ graph is required to be connected.

- *P2P Services:* Functionality provided by the P2P network for the applications sitting on top. Each instance of a distributed application on each peer interacts with the other instances on other peers through the API offered by the P2P services. Examples of the functions in this API are join a P2P network, or look-up that finds the peer responsible for a given resource in the P2P network.

2.2 Problem Overview

In recent years, Peer-to-Peer (P2P) networking technology has gained tremendous attention from both academy and industry. In a P2P system, peers communicate directly with each other for the sharing and exchange of data [1] as well as other resources such as storage and CPU capacity, each peer acts both as a client who consumes resources from other peers, and also as a server who provides service for others. P2P systems can benefit from their following characteristics: adaptation, self-organization, load-balancing, fault-tolerance, availability through massive replication, and the ability to pool together and harness large amounts of resources. For example, file-sharing P2P systems distribute the main cost of sharing data - bandwidth and storage - across all the peers in the network, thereby allowing them to scale without the need for powerful and expensive servers. However, providing streaming service over P2P networks is still a challenging task because of their inherent instability and unreliability. The two major challenges in providing P2P media streaming are locating supplying peers and maintaining content delivery paths.

2.2.1 Video Streaming Application Background

Streaming is defined as "technique for transferring data (usually over the Internet) in a continuous flow to allow large multimedia files to be viewed before the entire file has been downloaded to a client's computer" [ANS]. Prior to the availability of streaming technique, multimedia content was distributed no differently than any other ordinary files (i.e. text files, executable files). They were all transmitted as "files" using downloading protocols such as ftp and http. Due to the large volume of data associated with a typical multimedia file, a long transmission

[1] for example, see http://www.bittorrent.com

time as well as a large storage space were required before the playback could begin. Furthermore, there was no way for the users to "peek" into the content to see if it is the video they would like to watch. This was often inconvenient, if not unacceptable, to the users due to a long waiting time and a large amount of wasted resources when the content of the video turned out to be something they were not interested in.

Streaming enables near instantaneous playback of multimedia content regardless of their sizes. It is made possible by a steady transmission of data packets in such a way that the users will receive the needed packets moments ahead of the time they must be played back. Streaming reduces the storage space and allows users to "quit" receiving the stream, if not interesting or satisfactory, before the entire file is downloaded.

Streaming media also enables real-time and continuous delivery of video and audio data in a fashion of "flow", i.e., once the sender begins to transmit, the receiver can start playback almost at the same time while it is receiving media data from the sender, instead of waiting for the entire media file to be ready in the local storage. Unlike normal data file, a streaming media file is huge, thus requires high channel bandwidth. Moreover, streaming media also carries stringent demand in the timing of packet delivery. Live streaming captures audio/video signals from input devices (e.g. microphone, video camera), encodes the signals using compression algorithms (e.g. MP3, MPEG-4), and distributes them in real-time. Typical application of live streaming includes surveillance, broadcasting of special events, and distribution of information that have the prime importance in real-time delivery. In live streaming, the server side has the control over the selection of the distribution content and the timing of their streaming. The user involvement is typically limited to joining and leaving the running streaming sessions.

Pre-recorded or stored streaming distributes pre-encoded video files stored at a media server. Sample applications include multimedia archival retrievals, news clip viewing, and distance learning through which students attend classes on-line by viewing pre-recorded lectures. Under stored streaming, when and what title of the video will be streamed are dictated by the user. As such, a great amount of load may be placed on the media server when supporting a large number of asynchronous users (i.e. users whose streaming requests arrive at different times) with diverse interests. Our research focuses on the support of stored streaming.

Streaming brings new challenges to the distribution of multimedia content. Due to its strict on-time delivery requirement of data packets, a mechanism is needed to regulate the flow of packets over the Internet. An active research has been conducted in the areas of rate control, to cope with the time-varying bandwidth availability of Internet [VChS03], [TZ99], buffer management, to overcome delay variations [KSG02], and error control, to reduce the impact of packet loss. Progress in standardization work has produced specifications on key aspects of streaming, such as media encoding [LBCL99], media transport and session control, and media description and announcement.

2.2.2 Multimedia Streaming over TCP/IP

Standard networking applications like email or FTP, which only need reliable non-realtime data transmission, are using TCP on top of lossy IP networks. TCP countervails the problems of the underlying network layers by most notably implementing congestion control and guaranteed retransmission. As an example, for a file transfer it is acceptable, that the amount of time used for this transfer is determined by the available bandwidth. So it will take longer to transfer the same amount of data over a low bandwidth link than over a faster connection.

For multimedia streaming, a stream with a certain bandwidth requirement should ideally never suffer from lacking bandwidth. If it is subjected to a sudden bandwidth reduction and no measures are taken to prevent such a case, the client application stops playing and has to

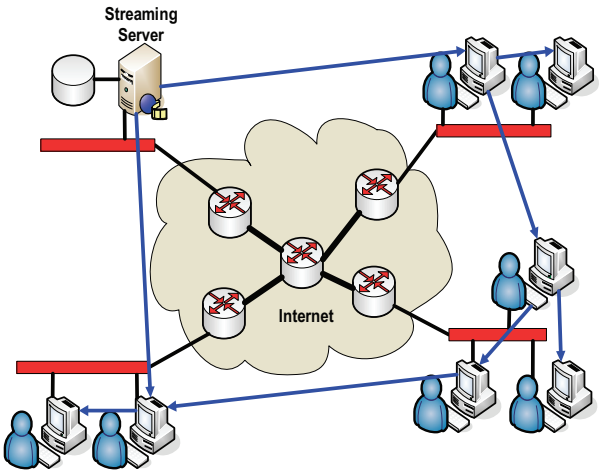


Figure 2.2: An overlay multicast for a P2P media streaming system.

refill the buffer again. TCP, when used for multimedia streaming and subjected to packet loss, sacrifices bandwidth for retransmission, even though some retransmitted packets would be late anyway. Further, TCP congestion control reacts with heavy and discontinuous reduction of the available streamout rate, which also wastes available bandwidth. Multimedia streaming environments cannot compensate those heavy and frequent fluctuations over a longer period of time without running out of buffer sooner or later.

Although attempts were made to use TCP/IP for multimedia streaming [KLW01], [GAA03] the widely accepted approach is to use the real-time transport protocol (RTP [SC96]) on top of UDP/IP, which circumvents most of TCP's unwanted behavior. Using recently introduced extensions on RTP retransmission of lost packets [RLM+02] and more immediate RTP feedback about network behavior [OWS+02], an intelligent server-client software like the one presented in this work, will overcome some of the Internet's deficiencies without suffering from TCP's rigid behavior.

Further note that using UDP also offers the possibility of sending data in a multicast fashion, which means that multiple clients are connecting to the same stream, which hereby severely decreases the network load. Still, for this work, we want to focus on personalized video on demand streaming, so each client can start and pause its stream at will, which is only functional with the unicast scenario.

2.3 Recent Progress of P2P Media Streaming

A simple and straightforward way of P2P streaming implementation is to use the technique of application-layer multicast (ALM). With ALM, all peer nodes are self-organized into a logical overlay tree over the existing IP network and the streaming data are distributed along the overlay tree. The cost of providing bandwidth is shared among the peer nodes, reducing the burden of the media server. In application-layer multicast, data packets are replicated and forwarded at end hosts, instead of at routers inside the network. Compared with IP multicast, application-layer multicast has several advantages. On the one hand, since there is no need for supports from routers, it can be deployed gradually based on the current Internet infrastructure; on the other hand, application-layer multicast is more flexible than IP multicast, and can adapt different distribution demands of various upper level applications.

Thus, how to construct and maintain an efficient ALM-based overlay network has became

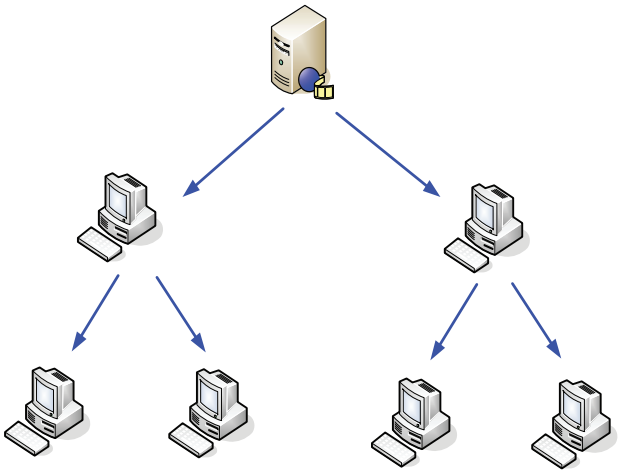


Figure 2.3: A simple example illustrating the basic approach of tree-based overlay multicast.

one of the key problems of P2P streaming research. Figure 2.2 shows the structure of an overlay multicast for a peer-to-peer (P2P) media streaming system. To address this problem, mainly three questions should be answered. The first relates to the P2P network architecture, i.e., what topologies should the overlay network be constructed? The second concerns routing and scheduling of media data, i.e., once the overlay topology is determined, how to find and select appropriate upstream peers from which the current peer receives the needed media data? The third is membership management, i.e., how to manage and adapt the unpredictable behaviors of peer joining and departure?.

Research has shown that it's feasible to support large-scale media streaming over the Internet using a P2P approach but such systems still confront some design challenges:

- *Dynamic uptime.* In P2P networks, peers don't always stay online in the system. Supplying peers might suddenly crash or leave ungracefully. In this case, the requesting peers need to find new supplying peers to replace the failed ones. Therefore, the system should be robust enough to withstand such node failures.

- *Limited and dynamic peer bandwidth.* Unlike powerful video servers, peers have limited bandwidth capacities. Each supplying peer might only be able to support a few requesting peers (or multiple supplying peers are required to support one requesting peer). Also, the available bandwidth of supplying peers might fluctuate unexpectedly. Hence, the system should be able to adaptively adjust each supplying peer's sending rate to keep the streaming quality at requesting peers unaffected.

To deal with these challenges, researchers have proposed various solutions. From the view of network topology, current systems can be classified into three categories approximately: tree-based topology, forest-based (multi-tree) topology, and mesh topology. In the following we give a brief summarization of P2P streaming techniques according to this classification.

2.4 Tree-based Topology

The most intuitive and common way to deliver multimedia content to a large group of peers is to build a single multicast tree, in which all interior nodes and leaf nodes are the peers. As Figure 2.3 shows, an interior node must forward data to other nodes, while a leaf node doesn't

need to forward any received data. An overlay tree among peers can be constructed in either a centralized or distributed manner.

2.4.1 PeerCast

The typical model of tree-based P2P streaming system is PeerCast [DBGM01]. In PeerCast, nodes are organized as a single multicast tree, where the parent provide service only directly to its sons. The node joining and departure strategies used in PeerCast are simple. For node joining, a new node n first request services from the root node S. If the S has enough resources, it provides service for n directly; otherwise, S redirects the request of n to one of its sons. The son then repeats this process, until the parent of n is found. Since each node only maintains the information of its parent and sons, unbalanced tree may be constructed.

Generally, there exist four route selection strategies in PeerCast: random selection, round-robin selection, smart selection according to physical placement, and smart selection according to bandwidth. To achieve a balanced multicast tree, custom routing policy should be chosen carefully for individual peer node.

2.4.2 Overcast

Overcast [JGJ$^+$00]is designed to offer an on-demand delivery of non-interactive, bandwidth demanding, video streaming service to a self-similar community of users through overlay network based multicast. The design goal of Overcast is to build single source multicast trees that maximize bandwidth availability from the root to each overlay node without knowing the details of the underlying network topology. It also tries to respond quickly and efficiently to transient network failures and congestions in the underlying network. Through a centralized lookup mechanism, a client wishing to receive a video streaming service finds a root node of a multicast tree that distributes a desired content. The client requests the root node to help measure the available bandwidth between the client and the root on a direct connection and saves it as a nominal download bandwidth. The client also requests the root node for a list of children nodes and their descendant nodes that are attached to the root node. The client checks the availability of bandwidth from the root through each one of every descendants and determines which overlay nodes can offer the same amount of bandwidth as the nominal download bandwidth. The most distant node, in terms of tree hierarchy, from the root that satisfies the bandwidth requirement will be selected as the node to which the client will attach itself in the multicast tree. On-demand service is supported by the notion of archive. Each overlay node buffers data it forwards in archive and distributes the archival index to participating overlay nodes in a multicast tree. A user can request the starting point, such as the beginning, when it joins an archival group of a particular tree. Since root node holds both the content as well as the distribution tree information, communication failures (e.g. link down, root node down) can jeopardize the entire distribution system. To cope with this problem, Overcast performs root node replications and forwards the streaming requests to replicated nodes in round-robin fashion. It also employees a series of replication nodes cascaded in front of the root node so that any one of them can overtake the responsibility of the root if necessary. In order to cope with non-root node failures, each child node maintains a list of ancestors so it can attach to a next higher level parent.

2.4.3 NICE

Peers in NICE [BBK02] systems cooperatively organize themselves into a logical hierarchy of clusters. As Figure 2.4 shows, clusters are managed into multiple levels using distributed

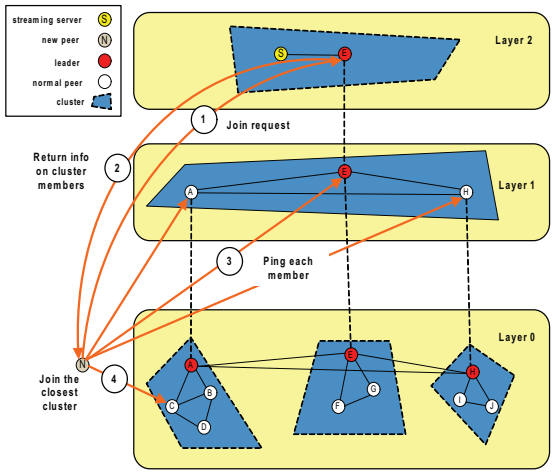


Figure 2.4: A hierarchical overlay approach for P2P Streaming.

algorithms. Each cluster has a cluster leader that's responsible for monitoring its cluster membership and is a member of a cluster in the upper level. Hence, some peers are cluster leaders in multiple levels. Cluster sizes can be between k and $3k$ (k is a system parameter) and are maintained using merge and split algorithms for bounding the out-degree of each peer. There's only one cluster in the topmost level where the source of the media resides.

To join the system, a newcomer first contacts the rendezvous point (the leader of the topmost cluster in Figure 2.4, labeled "F"). With that cluster's peer list attached in the leader's reply, the newcomer measures the distances between itself and all the peers in the list. Then, it selects the closest peer, which is a cluster leader in the lower level. After that, the newcomer sends another request to that leader. Again, that closest leader replies with its cluster member list. The probing process repeats until the new client finds its appropriate position in the architecture. By this successive probing, nearby peers are grouped together, making the data transmission based on that structure efficient.

This approach distributes the peer-management load over all the system's peers. For example, in contrast to $O(N)$ states maintained by the central server in a directory-based scheme, each peer in Zigzag only maintains $O(\log N)$ states. It also eliminates the single point of failure. Because the searching is based on the overlay structure, the maintenance of the structure among peers is critical to the searching performance. The control protocol for overlay maintenance hence must be resilient to node failure. In NICE, for instance, members in the same cluster periodically exchange heartbeat messages to detect member failure. However, such maintenance protocols can complicate the system implementation.

2.4.4 ZigZag

ZIGZAG [THT04] is another tree-based P2P streaming system which can construct more balanced multicast tree. As shown in Figure 2.5, ZIGZAG organizes receivers into a hierarchy of bounded-size clusters and builds the multicast tree based on that. The connectivity of this tree is enforced by a set of rules, which guarantees that the tree always has a height $O(\log_k N)$ and a node degree $O(k^2)$, where N is the number of receivers and k is a constant. Furthermore, the effects of network dynamics such as unpredictable receiver behaviors are handled gracefully without violating the rules. This is achieved requiring a worst-case control overhead of $O(\log_k N)$ for the worst receiver and $O(k)$ for an average receiver.

Other tree-based P2P streaming systems also include Overcast [JGJ+00], and Bayeux

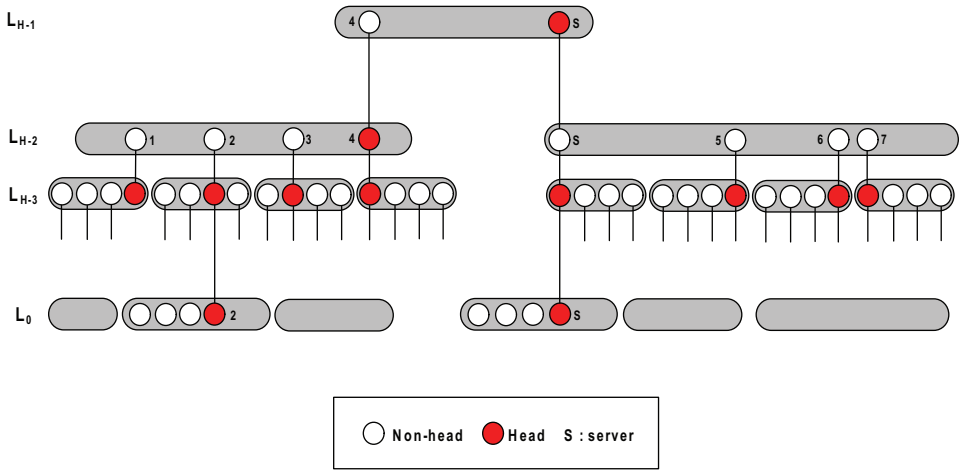


Figure 2.5: Administrative organization of Zigzag peers.

[ZZJ01].

Nevertheless, a tree-based structure suffers from several disadvantages:

- *It isn't fair.* All the leaf nodes don't need to forward data, but interior nodes are required to forward data to at least two children nodes (otherwise, the data paths become long). The number of leaf nodes increases much faster than the number of interior nodes. Due to the fact that there is only the interior nodes carry the forwarding load, the system load distribution becomes unbalanced.

- *It's fragile and prone to severe service disruption.* Each node is connected to its only parent with a single link. If the parent suddenly halts or the link is broken because of congestion, the child node and all its descendants immediately suffer from data shortage (that is, a buffer underflow) and a recovery scheme becomes necessary.

- *An interior node might not be able to offer high bandwidth video streaming to its children because of its limited bandwidth.* Furthermore, bandwidth is guaranteed to be monotonically decreasing as it goes down the tree; therefore, a node many hops away from the source might not receive enough bandwidth even though its parent has large outgoing bandwidth. Besides, for interactive VOD systems, it's even more challenging to maintain the overlay tree due to frequent VCR operations. Hence, supporting efficient interactive VOD streaming in P2P networks is still an active research topic.

Due to these weaknesses, the tree-based multicast approach is only suitable to small group streaming applications such as videoconferencing or multiparty gaming.

2.5 Forest-based Topology

Conventional tree-based multicast is inherently not well matched to a cooperative environment. The reason is that in any multicast tree, the burden of duplicating and forwarding multicast traffic is carried by the small subset of the peers that are interior nodes in the tree. Most of the peers are leaf nodes and contribute no resources. This conflicts with the expectation that all peers should share the forwarding load.

To address this problem, forest-based architecture is beneficial, which constructs a forest of multicast trees that distributes the forwarding load subject to the bandwidth constraints of the

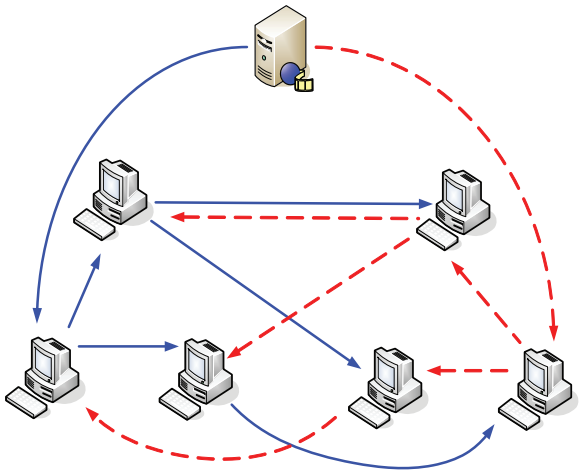


Figure 2.6: A simple example illustrating the basic approach of forest-based overlay multicast.

participating nodes in a decentralized, scalable, efficient and self-organizing manner. Figure 2.6 shows the basic approach of forest-based overlay multicast.

2.5.1 SplitStream

A typical model of forest-based P2P streaming system is SplitStream [CDK+03]. Figure 2.7 shows the basic approach of SplitStream. The key idea of SplitStream is to split the original media data into several stripes, and multicast each stripe using a separate tree. Each of the stripes has an identifier stripe ID starting with a different digit. Each stripe is multicast in its own designated tree. Peers join as many trees as there are stripes they wish to receive and they specify an upper bound on the number of stripes that they are willing to forward. To achieve better forwarding load balancing among nodes, SplitStream assigns each node to be an internal node at most in one overlay tree and leaf nodes in all other trees. It uses Pastry, a DHT protocol, to locate parents and assign positions of nodes in each tree.

A node's parent in a tree is the first node in the routing path from that node to the node with a node ID equal to the tree's stripe ID. Hence, the DHT routing protocol (Pastry) [RD01] intrinsically determines the locations of each node's parents in SplitStream. The multicast tree is then implicitly constructed by merging the paths from all the nodes to the root. This algorithm can potentially violate the degree bounds of interior nodes, and SplitStream applies several heuristic methods to redistribute the data-forwarding load among nodes. Using DHT protocol to locate supplying peers takes advantage of well-developed DHT protocols, which are scalable and offer good load balancing.

The challenge is to construct this forest of multicast trees such that an interior node in one tree is a leaf node in all the remaining trees and the bandwidth constraints specified by the nodes are satisfied. This ensures that the forwarding load can be spread across all participating peers. For example, if all nodes wish to receive k stripes and they are willing to forward k stripes, SplitStream will construct a forest such that the forwarding load is evenly balanced across all nodes while achieving low delay and link stress across the system.

Striping across multiple trees also increases the resilience to node failures. SplitStream offers improved robustness to node failure and sudden node departures like other systems that exploit path diversity in overlays. SplitStream ensures that the vast majority of nodes are interior nodes in only one tree. Therefore, the failure of a single node causes the temporary loss of at most one of the stripes (on average). With appropriate data encodings, applications

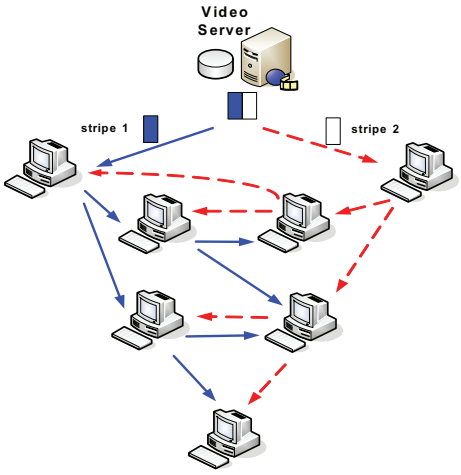


Figure 2.7: A simple example illustrating the basic approach of SplitStream.

can mask or mitigate the effects of node failures even while the affected tree is being repaired.

Besides SplitStream, there are many other forest-based systems. Examples include building mesh-based tree (Narada and its extensions and Bullet [KRAV03]), leveraging layered coding (PALS [RO03]), and multiple description coding (CoopNet [PWCS02]).

2.6 Mesh Topology

In conventional tree-based P2P streaming architectures, at the same time a peer can only receive data from a single upstream sender. Due to the dynamics and heterogeneity of network bandwidths, a single peer sender may not be able to contribute full streaming bandwidth to a peer receiver. This may cause serious performance problems for media decoding and rendering, since the received media frames in some end users may be incomplete.

In forest-based systems, each peer can join many different multicast trees, and receive data from different upstream senders. However, for a given stripe of a media stream, a peer can only receive the data of this stripe from a single sender, thus results in the same problem like the case of single tree.

Multi-sender scheme is more efficient to overcome these problems. In this scheme, at the same time a peer can select and receive data from a different set of senders, each contributing a portion of the streaming bandwidth. In addition, different from the multi-tree systems, the sender set members may change dynamically, due to their unpredictable online/offline status changes, and the time-variable bandwidth and packet-loss rate of the Internet. Since the data flow has not a fixed pattern, every peer can send and also receive data from each other, thus the topology of data plane likes mesh. The main challenges of mesh topology are how to select the proper set of senders and how to cooperate and schedule the data sending of different senders. Figure 2.8 illustrates the example of a mesh topology network.

Examples of mesh-based multi-sender P2P streaming system include CollectCast [HHR+03], GnuStream [JDXB03], and DONet (CoolStreaming) [ZLLY05].

2.6.1 CollectCast

CollectCast puts its emphasis mainly on the judicious selection of senders, constant monitoring of sender/network status, and timely switching of senders when the sender or network fails or seriously degrades. CollectCast operates entirely at the application level but infers and exploits

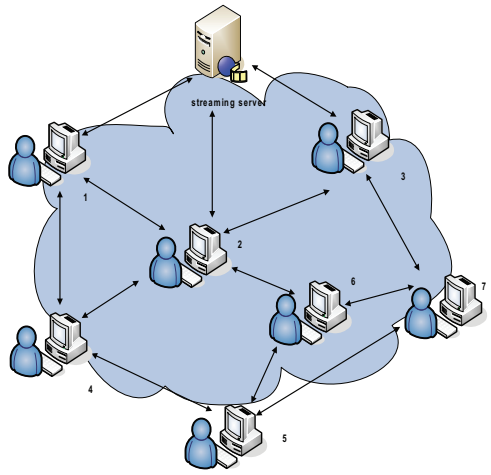


Figure 2.8: A simple Illustration of a mesh topology.

properties (topology and performance) of the underlying network. Each CollectCast session involves two sets of senders: the standby senders and the active senders. Members of the two sets may change dynamically during the session. The major properties of CollectCast include the following: (1) it infers and leverages the underlying network topology and performance information for the selection of senders. This is based on a novel application of several networks performance inference techniques; (2) it monitors the status of peers and connections and reacts to peer/connection failure or degradation with low overhead; (3) it dynamically switches active senders and standby senders, so that the collective network performance out of the active senders remains satisfactory.

2.6.2 GnuStream

GnuStream is a receiver-driven P2P streaming system which is built on top of Gnutella. It features multi-sender bandwidth aggregation, adaptive buffer control, peer failure or degradation detection and streaming quality maintenance. GnuStream is aware of the dynamics and heterogeneity of P2P networks, and leverages the aggregated streaming capacity of individual peer senders to achieve full streaming quality. GnuStream also performs self-monitoring and adjustment in the presence of peer failure and bandwidth degradation.

GnuStream has the following salient features:

- **Integration with P2P lookup substrate:** GnuStream leverages Gnutella as its lookup substrate, making it readily deployable in the current Gnutella P2P network environment.

- **Multi-sender aggregation:** Instead of relying on one single sender, GnuStream distributes streaming load among multiple peer senders.

- **Receiver data collection:** The receiver coordinates the arrival of different media data segments, and reconstructs media data in their original and continuous order before feeding them to the media player.

- **Detection of peer status change:** GnuStream uses periodic probing and soft states to detect any changes in the status of peer senders, including peer disconnection, failure, and bandwidth degradation.

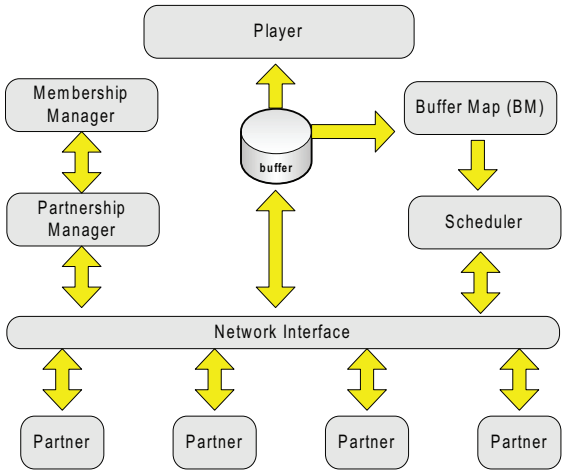


Figure 2.9: Illustration of the architecture DONet Buffer Map.

- **Recovery from failure or degradation:** If a current peer sender is detected as suffering from failure or bandwidth degradation, GnuStream will migrate all or part of its streaming load to another peer sender or a standby peer sender.
- **Buffer control:** To accommodate the dynamic set of peer senders and the end-to-end network congestion, GnuStream implements a suite of buffer control mechanisms which involves more concurrency and scheduling complexity than the traditional buffer control mechanisms in client-server streaming.

2.6.3 CoolStreaming/DONet

Recently, CoolStreaming/DONet implemented a multi-sender model by introducing a simpler and straightforward data-driven design, which does not maintain an even more complex structure. The core of DONet is the data-centric design of streaming overlay, and the Gossip-based data schedule and distribution algorithm.

In DONet system, the video stream is divided into segments of uniform length, and the availability of the segments in a node's buffer is represented by a buffer map (BM). Figure 2.9 illustrates the architecture of DONet node buffer map. Each node continuously exchanges its BM with other peers (or partners). Upon the receipt of BMs from multiple partners, the node assigns the requested data segments to each of the partners according to their data availability and available uploading bandwidth.

In the data-centric design of DONet, a node always forwards data to others that are expecting the data, with no prescribed roles like father/child, internal/external, and upstreaming/downstreaming, etc. In other words, it is the availability of data that guides the flow directions, while not a specific overlay structure that restricts the flow directions. This data-centric design is suitable for overlay with high dynamic nodes.

Gossip algorithms have recently become popular solutions to multicast message dissemination in P2P systems [HHL06]. In a typical gossip algorithm, a node sends a newly generated message to a set of randomly selected nodes; these nodes do similarly in the next round, and so do other nodes until the message is spread to all. The random choice of gossip targets achieves resilience to random failures and enables decentralized operations. Similar to the related work [GKM03], DONet employs a gossiping protocol membership management. The data schedule and distribution method used in DONet is also partially motivated by the gossip concept. It

Approach	Allow Optimization	Resilient to Node Failure	Multiple Suppliers	Load Balancing	Achievable Transmission rate	Implementation
Tree-based	Yes	Poor	No	Medium	Medium	Easy
Forest-based	No	Good	Yes	Good	High	Difficult
Mesh	No	Good	Yes	Good	High	Easy

Table 2.1: Comparison of between various approaches for content delivery

uses a smart partner selection algorithm and a low-overhead scheduling algorithm to intelligently pull data from multiple partners, which greatly reduces redundancy. Experiments show that, compared with a tree-based overlay, DONet can achieve much more continuous streaming with comparable delay. Research has also shown that the average overlay path length from the source to a node is $O(\log N)$ [ZLLY05]. This preserves the media data's freshness, which is crucial to a live media streaming application.

The gossip protocol lets each peer in the system retrieve data from multiple parents and, at the same time, serve multiple children. Compared to tree-based protocols, this approach greatly improves resource utilization and load balancing. In addition, the service's stability is also enhanced because of the redundancy of service providers. Unfortunately, because of the partner selection process's randomized nature, the quality of overlay delivery paths such as bandwidth and delay can't be optimized or even guaranteed.

2.7 Problems related to Locating Supplier Peers

Locating supplying peers is a challenge in a P2P media streaming system because each requesting peer must find supplying peers with enough bandwidth and preferably low latency to achieve a good service quality. Some common techniques for locating supplying peers in such systems include a centralized directory, hierarchical overlay structure, distributed hash table (DHT) based approach, controlled flooding, and gossip-based approach.

2.7.1 Centralized Directory

The simplest and most commonly used method for locating peers is to maintain a centralized directory of all peers in a directory server [2]. All required peer information, including its network address, available bandwidth, fan-out degree, and starting point of access (for video-on-demand [VoD] systems), are in a directory server with a well-known IP address. The server can also keep the global overlay topology among peers. Figure 2.10 demonstrates the flow of control messages in such a system. In this system, a new requesting peer's request is first directed to the directory server. Upon receiving a user request, the directory server selects the most suitable supplying peers from the stored peer list for the new user, according to its network address and the requested media. For example, a peer with large available bandwidth and a network location close to the requesting peer is chosen by the directory server to serve the requesting peer. If a user wants to leave the system, he or she must signal the directory server with a **LEAVE** message to clear his or her entry in the directory. The advantages of this approach are its easy implementation and simple deployment. Centralization also greatly simplifies the join mechanism and consequently makes the join and leave procedures quick.

The advantages of this approach are its easy implementation and simple deployment. Centralization also greatly simplifies the join mechanism and consequently makes the join and leave

[2] such as http://www.ppstream.com or http://www.pplive.com

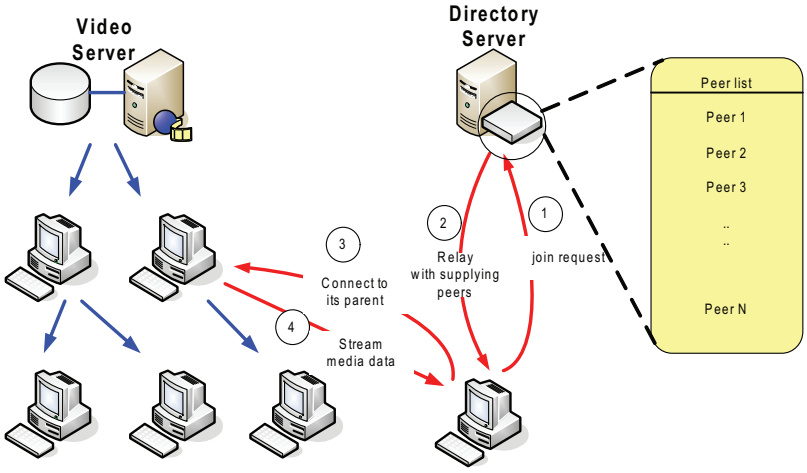


Figure 2.10: A centralized directory approach for P2P Streaming.

procedures quick. However, given N peers in the system, the directory server must maintain $O(N)$ states, which might overload the server when N is large. Furthermore, if a peer is unable to send a **LEAVE** message to the directory server (such as during a node failure), its state remains in the directory. To handle this problem, the peers must refresh their status at the server using periodic *keep-alive* (or *heartbeat*) messages or similar mechanisms. Transmitting these $O(N)$ keep-alive control messages also incurs high-bandwidth consumption at the server. Another system weakness is that the directory server becomes a single point of failure. While the directory server is down, users can no longer join the system. Researchers [PWC04] have argued that the directory server is usually also the source of data - such as for a video server. Hence, if the server (source) fails, it might not matter whether the directory is down

2.7.2 DHT-based Approach

Another distributed approach for locating supplying peers is based on a DHT [CDK+03], a common P2P searching technique, usually for locating nodes that store a desired object (such as a file) in P2P networks [SMK+01], [RFH+01], [RD01]. In DHT, each peer is assigned a peer ID by hashing its own IP address using a common known hash function such as SHA-1 [18095] and each object is also associated with a key in the same space of peer IDs by hashing the object itself. The peer with an ID equal to the hashed key is responsible for storing the object's location (or the actual object). The primitive functions **PUT** and **GET** are available in DHT, as in a conventional hash table data structure. With an object's hashed key, a query for the object is routed through several nodes in the DHT to the node responsible for storing the object. There's a routing table maintained at each node in the DHT, based on which peers route queries. The routing load is also evenly distributed over all the peers in the DHT.

Related work has proven that query messages are routed through only $O(logN)$ nodes for each lookup, and each node only needs to maintain $O(logN)$ states in its routing table. Also, the load of routing requests is evenly distributed over all the peers in the network. Furthermore, using a well-developed DHT protocol simplifies the system's design and implementation. The system then can focus on the media streaming functionalities and needn't deal with the complicated peer management.

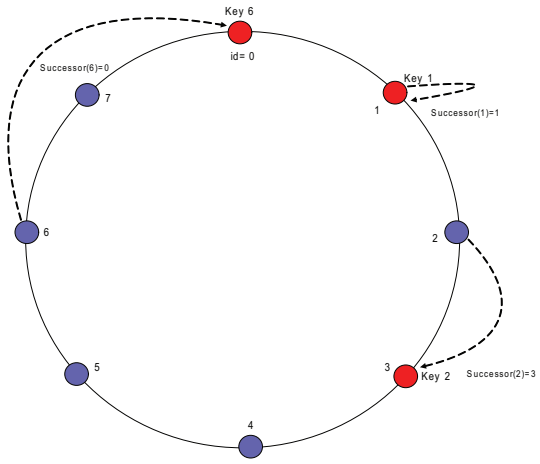


Figure 2.11: A Chord identifier circle consisting of the 3 nodes 0, 1 and 3. In this example, key 1 is located at node 1, key 2 at node 3 and key 6 at node 0.

Chord

Chord [SMK+01] is a peer-to-peer routing and location infrastructure that performs a mapping of file identifiers onto node identifiers. Data location can be implemented on top of Chord by identifying data items (files) with keys and storing the (*key, data item*) pairs at the node that the keys map to.

In Chord nodes are also identified by keys. The keys are assigned both to files and nodes by means of a deterministic function, a variant of consistent hashing [KLL+97]. All node identifiers are ordered in an "identifier circle" modulo 2^m. Key k is assigned to the first node whose identifier is equal to or follows k in the identifier space. This node is called the successor node of key k. The use of consistent hashing tends to balance load, as each node receives roughly the same number of keys.

The only routing information required is for each node to be aware of its successor node on the circle. Queries for a given key are passed around the circle via these successor pointers until a node that contains the key is encountered. This is the node the query maps to. When a new node n joins the network, certain keys previously assigned to n's successor will become assigned to n. When node n leaves the network, all keys assigned to it will be reassigned to its successor. These are the only changes in key assignments that need to take place in order to maintain load balance.

Only one data element per node needs be correct for Chord to guarantee correct (though slow) routing of queries. Performance degrades gracefully when routing information becomes out of date due to nodes joining and leaving the system, and availability remains high only as long as nodes fail independently. Since the overlay topology is not based on the underlying physical IP network topology, a single failure in the IP network may manifest itself as multiple, scattered link failures in the overlay.

To increase the efficiency of the location mechanism described above-which may in the worst case require traversing all N nodes to find a certain key-Chord maintains additional routing information, in the form of a "finger table". In this table each entry i points to the successor of node $n + 2i$. For a node n to perform a lookup for key k, the finger table is consulted to identify the highest node n_o whose ID is between n and k. If such a node exists, the lookup is repeated starting from n_o. Otherwise, the successor of n is returned. Using the finger table, both the amount of routing information maintained by each node and the time required for

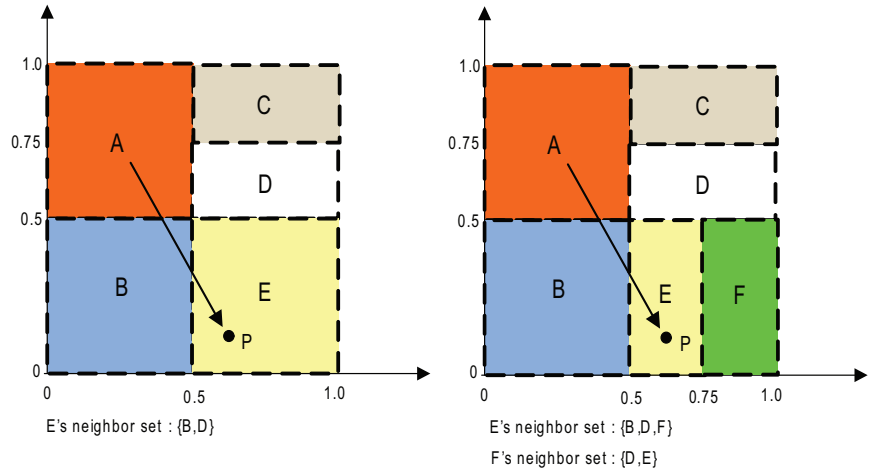


Figure 2.12: CAN: (a) Example 2-d $[0, 1] \times [0, 1]$ coordinate space partitioned between 5 CAN nodes. (b) Example 2-d space after node F joins.

resolving lookups are $O(logN)$ for an N-node system in the steady state. Figure 2.11 shows a chord identifier circle consisting of three nodes.

CAN

The CAN ("Content Addressable Network") [RFH+01] is essentially a distributed, internet-scale hash table that maps file names to their location in the network, by supporting the insertion, lookup, and deletion of (*key, value*) pairs in the table.

Each individual node of the CAN network stores a part (referred to as a "zone") of the hash table, as well as information about a small number of adjacent zones in the table. Requests to insert, lookup or delete a particular key are routed via intermediate zones to the node that maintains the zone containing the key.

CAN uses a virtual d-dimensional Cartesian coordinate space (see Figure 2.12) to store $(keyK, valueV)$ pairs. The zone of the hash table that a node is responsible for corresponds to a segment of this coordinate space. Any key K is therefore deterministically mapped onto a point P in the coordinate space. The (K, V) pair is then stored at the node that is responsible for the zone within which point P lies. For example in the case of Figure 2.12(a), a key that maps to coordinate (0.1, 0.2) would be stored at the node responsible for zone B.

To retrieve the entry corresponding to K, any node can apply the same deterministic function to map K to P and then retrieve the corresponding value V from the node covering P. Unless P happens to lie in the requesting node's zone, the request must be routed from node to node until it reaches the node covering P.

CAN nodes maintain a routing table containing the IP addresses of nodes that hold zones adjoining their own, to enable routing between arbitrary points in space. Intuitively, routing in CAN works by following the straight line path through the Cartesian space from source to destination coordinates.

2.7.3 Binning

Binning [RHKS02] is a landmark based peer positioning scheme to identify a set of server peers that are close to the requesting peer. One of the challenges of P2P discovery service is how to locate the nearest server peers among possibly many and globally spread peers listed at an

	Level 5	Level 4	Level 3	Level 2	Level 1
Entry 0	07493	x0493	xx093	xxx03	xxxx 0
Entry 1	17493	x1493	xx193	xxx13	xxxx 1
Entry 2	27493	x2493	xx293	xxx23	xxxx 2
Entry 3	37493	x3493	xx393	xxx33	xxxx 3
Entry 4	47493	x4493	xx493	xxx43	xxxx 4
Entry 5	57381	x5493	xx593	xxx53	xxxx 5
Entry 6	67493	x6493	xx693	xxx63	xxxx 6
Entry 7	77493	x7493	xx793	xxx73	xxxx 7
Entry 8	87493	x8493	xx893	xxx83	xxxx 8
Entry 9	97493	x9493	xx993	xxx93	xxxx 9

Figure 2.13: Illustration of a neighbor map held by Tapestry node with ID 67493. Each entry in this table corresponds to a pointer to another node.

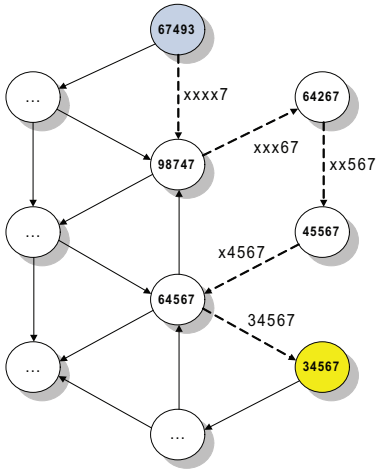


Figure 2.14: Illustration of an example of Tapestry mesh routing example, showing the path taken by a message originating from node 67493 and destined for node 34567.

overlay node. Binning organizes nearby peers in a group known as a bin. The peer to bin assignment is done as follows. Each peer measures the distance to several landmark nodes (Binning suggests 8 to 10 landmark nodes to cover the entire Internet) that are scattered throughout the Internet and order the measured results in the closest to the farthest ranking. Those peers that have the same order of landmark nodes will be placed in the same bin. The idea behind this logic is that peers that have the same or similar landmark-distance-order should reside in the same region of the network. When a peer requests a content from a CAN zone, it first measures the distance to each landmark and determines its bin order. This bin order is attached to the query from the peer to the CAN node. The destination CAN node returns only those server peers that have the same or similar bin orders to the requesting peer.

Tapestry and Pastry

Tapestry is a peer-to-peer, wide-area decentralized routing and location network infrastructure developed at University of California at Berkeley by Zhao et al [ZHJK04]. Tapestry is based on a peer-to-peer overlay routing infrastructure offering efficient, scalable, location-independent routing of messages directly to nearby copies of an object or service using only localized resources. Tapestry forms an overlay network that sits at the application layer (on top of an Operating System). If Tapestry is installed on different network nodes it will allow any one node to route messages to any other node running tapestry, given a location and a network independent name. Nodes in a Tapestry network can also advertise location information about data it possesses in a specific format understood by other nodes running tapestry. This special format allows the other nodes to find and access this data easily and efficiently, given that they know the data name. The main focus of Tapestry is on the routing that minimizing message latency and maximizing message throughput.

Tapestry uses adaptive algorithms with soft state to maintain fault tolerance in the face of changing node membership and network faults. Its architecture is modular, consisting of an extensible upcall facility wrapped around a simple, high-performance router. Therefore, we can see that Tapestry allows nodes the ability to share data, thereby creating their own P2P system.

Figure 2.13 shows an example neighbor map maintained by a node with ID 67493. As shown in figure 2.13, each node maintains a neighbor map. The neighbor map has multiple levels, each level l containing pointers to nodes whose ID must be matched with l digits (the x's represent wildcards). Each entry in the neighbor map corresponds to a pointer to the closest node in the network whose ID matches the number in the neighbor map up to a digit position. For example, the 5^{th} entry for the 3^{rd} level for node 67493 points to the node closest to 67493 in network distance whose ID ends in ..593. Messages are therefore incrementally routed to the destination node digit by digit, from the right to the left. Figure 2.14 shows an example path taken by a message from node with ID=67493 to node ID=34567. The digits are resolved right to left as follows: $xxxx7 \rightarrow xxx67 \rightarrow xx567 \rightarrow x4567 \rightarrow 34567$.

Tapestry is based on the location and routing mechanisms introduced by Plaxton [PRR97], in which they present the Plaxton mesh, a distributed data structure that allows nodes to locate objects and route messages to them across an arbitrarily-sized overlay network while using routing maps of small and constant size. The Plaxton mesh uses a root node for each object, which serves to provide a guaranteed node from which the object can be located. When an object o is inserted in the network and stored at node n_s, a root node n_r is assigned to it by using a globally consistent deterministic algorithm. A message is then routed from n_s to n_r, storing at all nodes along the way data in the form of a mapping (object id o, storer id n_s). During a location query, messages destined for o are initially routed towards n_r, until a node

Approach	Features
Chord	A scalable peer-to-peer lookup service. Given a key, it maps the key to a node
CAN	Scalable content addressable network. A distributed infrastructure that provides hash-table functionality for mapping file names to their locations.
Tapestry	Infrastructure for fault-tolerant wide-area location and routing.
Pastry	Infrastructure for fault-tolerant wide-area location and routing.

Table 2.2: Comparison of between various approaches for DHT-based approach

is encountered containing the (o, n_s) location mapping.

The advantages of Plaxton mesh include:

1. simple fault handling by its potential to route around a single link or node by choosing a node with a similar suffix, and
2. scalability (with the only bottleneck existing at the root nodes)

While its limitations include:

1. the need for global knowledge required for assigning and identifying root nodes and
2. the vulnerability of the root nodes.

Pastry [RD01] is a scheme very similar to Tapestry, differing mainly in its approach to achieving network locality and object replication. It is employed by the PAST large-scale persistent peer-to-peer storage utility.

2.7.4 Gnutella

Gnutella's searching process is based on controlled flooding of the query over the overlay mesh built among the system's peers. The query from the requesting peer is sent to all its neighboring peers. Upon receiving the query, the peer rebroadcasts the query to all its neighbors except the one that sent the query. The query message is associated with a time-to-live (TTL) value. Each broadcasting by a peer decreases the TTL by one. This query broadcasting continues until the TTL becomes zero. The peers who receive the query and possess the requested object will reply with the query's origin, indicating the requested object's presence. Depending on the degree of connectivity among peers, the flooding of queries can generate a lot of network traffic. Besides, objects located out of the search scope (which the TTL determines) wouldn't be found in the system.

One example of Gnutella P2P search system is GnuStream [JDXB03].

2.8 Technical Challenges and Opportunities

Though some successes have been made in recent years, especially with the introducing of mesh-based approaches, there are still challenging problems and open issues need to be overcome in P2P live media streaming.

The main problem results from the heterogeneity of the underlying IP networks. There exist mainly two types of heterogeneities in the current Internet: heterogeneous receivers and asymmetric access bandwidths. In a P2P based live media streaming system, for each individual peer the receiving capability is decided by its downlink bandwidth, however for the whole system the total available bandwidth is decided by the sum of the uplink bandwidths of all the participated peers. Under this situation, same and perfect QoS is hard to be guaranteed for

Approach	Scalability	Single Point Failure	Search Guarantee	Server States	Peer States	Implementation
Centralized Directory	Low	Yes	Yes	$O(N)$	$O(1)$	Simplest
Hierarchical overlay structure	High	No	Yes	$O(1)$	$O(logN)$	Most Difficult
DHT-based	High	No	Yes	$O(1)$	$O(logN)$	Medium

Table 2.3: Comparison of between various approaches for locating supplier peers

all of the participated peers. For example, if the access bandwidth of a peer is less than the average bit rate of the media stream it requires, or the sum of the uplink bandwidths of all upstream peers who provide data for this peer is less than the average bit rate, then random packet-losses may occur ether during the network or at the buffer of upstream peers. This may lead to incorrect decoding at the client side even partial data have been received, which means not only the waste of bandwidth resources, but also degraded media reconstruction qualities. The most hopeful solution to this problem is to provide self-adaptive QoS for each individual peer according to the current network conditions, at the same time the total available uplink bandwidths of all peers are utilized as full as possible. To satisfy this objective, three main issues should be addressed:

2.8.1 Content aware media data organization

Current P2P streaming systems focus mainly on network topology and protocol design, but pay rare attention to the media contents carried over the network. In fact, since streaming media have their distinct characteristics from normal data file, good performance can be achieved only when both the characteristics of media coding and networking are considered together perfectly. While scalable coding techniques hold promise for providing network adaptive media transmission, they are yet to be deployed in today's mainstream media codec. A promising solution is to partition the current non-scalable coded media data based on content analysis, and reorganize them into another form with scalable capability to some extent, so that selective and priority-based schedule strategies can be used while transmission.

2.8.2 Priority-based media data delivery mechanism

For the quasi-scalable media data prepared above, efficient transmission and control mechanisms should be invented to guarantee that the minimal decodable media units (for example, a video frame or slice) can be transmitted to the receiver in a restrict order based on their priorities. This implies that every data unit received by a peer at any time is intact and decodable independent of any still un-received media data. By this way, no waste of bandwidth is involved and free-error and fluent media experience can be obtained even in the case of worse network conditions.

2.8.3 QoS adaptive multi-source and layered media data schedule algorithm

Based on the above content aware data organization and priority-based delivery mechanism, efficient data schedule algorithms are needed to retrieve data from multiple senders in order to maximize the overall bandwidth utilization of the whole network and minimize the average media reconstruction distortion of all users. Compared with conventional P2P streaming systems which simply partition a streaming media into a series of data blocks and schedule each data block as the minimal transmitting unit, the scheduling model of this system and its solutions are more complicate to establish and resolve.

2.9 Summary

This section provided selected literature review on recent progress of peer-to-peer streaming systems that try to achieve data path scalability and search path scalability.

From the view of network topology, current systems can be classified into three categories approximately: tree-based topology, forest-based (multi-tree) topology, and mesh topology. Most of the peer-to-peer streaming systems were built on top of overlay network based multicast.

Under tree-based topology, we reviewed PeerCast, Overcast, NICE and Zigzag. NICE and ZIGZAG aim to support live-streams with small payload to a large number of users. PeerCast's goal is to support live-streams among dynamically changing peers. Under forest-based (multi-tree) topology, we reviewed SplitStream, Narada and Bullet. The key idea of SplitStream is to split the original media data into several stripes, and multicast each stripe using a separate tree. While under mesh topology, we reviewed CollectCast, GnuStream and CoolStreaming/DONet. In mesh topology, the data flow does not have a fixed pattern, every peer can send and also receive data from each other in an inter-connected overlay network.

We also reviewed the problems commonly faced by peers in a peer-to-peer streaming system - locating supplier peers. We reviewed common techniques for locating supplying peers in such systems include a centralized directory, hierarchical overlay structure, distributed hash table (DHT) based approach, controlled flooding, and gossip-based approach.

Finally we discussed the technical challenges and opportunities for peer-to-peer streaming. We reviewed the three main issues faced by peer-to-peer streaming system which are related to content aware media data organization, priority-based media data delivery mechanism and QoS adaptive multi-source and layered media data schedule algorithm. □

CHAPTER 3

PALMS : Design and Implementation

All successful people have a goal. No one can get anywhere unless he knows where he wants to go and what he wants to be or do.
Norman Vincent Peale.

3.1 Introduction

From an user's perspective, there is little difference between a live stream and an on-demand video, since they have more or less the same appearance and seem to work in the same way. But these two kinds of streaming are very different from a server/system and end-user point of view, since they are based on quite different assumptions and requirements, thus lead to different solutions.

On-demand streaming deals with the transmission of *finite-sized* media files, which are received by the clients and usually played during the download; the media files themselves are not sensitive to playout delay, since the events they represent aren't "live" any more, but have been recorded, edited and stored somewhere for public consumption.

With that, it can be said that this problem is only marginally more complex than standard file-sharing, having the additional need of a certain (very loose) order in the reception of the data chunks so that the user can start watching the file as soon as it's possible. Moreover, we can also point out from our experience that most users don't really care about how soon a movie can be watched, as long as it can be enjoyed when and each time the users want. We also feel that these needs for on-demand streaming are already - or will soon be, as the available bandwidth increases with the development of latest technology, thus can be satisfied by the existing file-sharing applications.

Live streaming, on the other hand, deals with the transmission of practically infinite-sized media streams, which are to be played by the clients upon reception: the streams are sensitive to playout delay, i.e. the time skew between the transmission at the source and the reproduction

by the users' program, and this delay should be upper-bounded to avoid having users play outdated information.

For the reasons given above, we choose to concentrate our research on problems related streaming live media using peer-to-peer technologies, instead of investigate the on-demand streaming problems.

3.2 Objectives and Design Principles

Our main goal for PALMS (P2P Unstructured Live Media Streaming) is to create an unstructured, self-organizing overlay network with simple network construction and maintenance mechanisms, and the ability to deliver high-bandwidth data streams across a highly volatile and transient node population. In designing PALMS, we also make an effort to reduce the constraints to be satisfied by any involved node, so to avoid the exclusion of significant subsets of users with asymmetric connections lacking a sufficient upstream bandwidth, and moreover we take into the consideration of the contribution of each peer to the system. A feature we wish to develop that is resilient towards dynamics of peers arrival and departure, which is also known as *churn* and ability to serve low-rate-contributing peers normally, as long as the system capacity is under-utilized, and to gently decrease their service level when scarcity appears.

To achieve these goals we choose to apply the following design principles:

- all nodes in the population, besides the source, are equal, but they serve (and are served) according to their resources, characteristics and past behaviour
- since the transience in the network and the population may be very big, we do not attempt to keep a global system state, nor we wish to maintain large-scale infrastructures like distribution trees
- we accept that only a local view of the network can be up-to-date at a given time - to reduce the control traffic among nodes - so we try to exploit the properties of local- scale, ephemeral and loosely-delimited node aggregations (local meshes) to obtain the desired global behaviour
- the system should be compatible with (but not depend upon) advanced encoding methods such as Forward Error Correction (FEC)

With the latest achievement of BitTorrent [Coh03], we try to develop peer-driving algorithms that is able to achieve similar results in the field of reliable static file transfer. We focus our research attention mainly on the "network and data management" level, which takes care of the stream data selection, data replication and node association processes.

In this chapter, we present a detailed overview of the PALMS's architecture. We also will discuss the various components that form the foundation of distributing live media streaming over unstructured P2P networks. Later part of this chapter, we continue the discussion of the pros and cons of the PALMS distribution model and present the details of some of the components that underlie the architecture.

3.3 PALMS : System Overview

PALMS is based on data-driven and receiver-based unstructured overlay media streaming. It is designed to operate in scenarios where the nodes have heterogeneous and variable bandwidth resources. For the ease of exposition, we refer to the media source as the *streaming server*

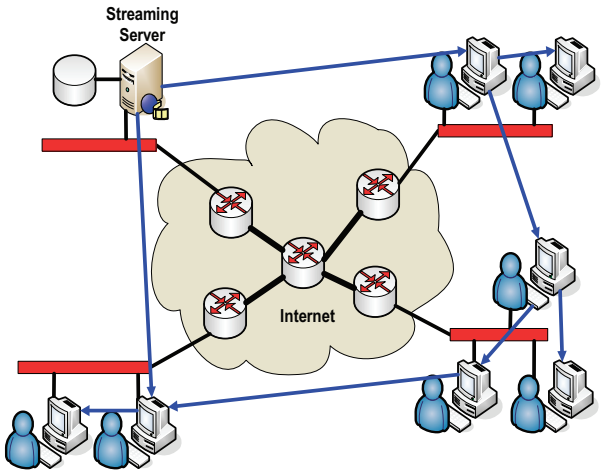


Figure 3.1: Illustration of an overlay multicast for a peer-to-peer (P2P) media streaming system.

and receivers as *clients*. The term *peers* and *nodes* are interchangeable, and refer to the all the clients. Peers in PALMS implement data exchange policies that enforce and reward node cooperation through a score-based incentive mechanism. The incentive mechanism encourages cooperation among participating nodes and allows gradual improvement in the streaming of media content and in turn improves the delivered quality of streaming.

PALMS consists of four major components: (i) **overlay construction mechanism**, which organizes participating peers into an overlay; (ii) **membership management** (iii) **streaming scheduling mechanism**, which determines the delivery of content from the streaming source to individual nodes through the overlay; and (iv) **incentive mechanism**, which determines service received by a peer as a function of the bandwidth contributed by the peer. In the following subsections, we describe the components in PALMS.

3.3.1 Overlay Construction

In PALMS, nodes are functionally identical. They are free to exchange control information and media content data from the stream. Participating nodes form a randomly connected directed graph overlay network as shown in Fig. 3.1. Each peer maintains a certain number of connected nodes that are known as *neighbors*. Each node receives media content from a certain number of parent nodes and relays the content to a certain number of child nodes. *Incoming* and *outgoing* degree denotes the number of connected parents and child nodes respectively.

The basic task of the overlay construction mechanism component for each node is to be in charge of finding appropriate neighbors for each node through the gossip method so that the application layer network can be successfully built up. To join the streaming session, a new peer contacts the bootstrapping node, (streaming server in the case of PALMS) to learn about other participating peers upon arrival. This could be regarded as the login process. The bootstrapping node returns a list of randomly selected peers that can potentially serve as parent nodes. The new peer contacts these potential parent nodes to determine whether they are able to accommodate a new child node. This is by determining whether the parent node still has enough allocation slots on the outgoing degree. The peer also maintains a target incoming degree. If a new peer cannot identify a sufficient number of parents from the reported list, it will contact the bootstrapping node again to obtain another list of potential parent nodes. A successful new peer is registered with the bootstrapping node and can be selected as the parent

Input:
Upon subscription of a new subscriber s on a contact node *contact*

{*The subscription of s is forwarded to all the nodes of view*}
for all nodes $n \in PartialView_{contact}$ **do**
Send $(n, s, forwardedSubscription)$;
end for
{*c additional copies of the subscription s are forwarded to random nodes of view*}
for $(j = 0; j < c; j++)$ **do**
Choose randomly $n \in PartialView_{contact}$
Send $(n, s, forwardedSubscription)$;
end for

Figure 3.2: SCAMP subscription management algorithm

node by other peers. Each new joined node synchronizes the local clock with the bootstrapping node during login process.

All the nodes will self-organize into an unstructured mesh. Each node has a member table that contains a list of neighbor nodes obtained from bootstrapping node. The information in member tables is encapsulated into a UDP packet and exchanged among neighbors periodically. Each node updates its member table in accordance with the member table sent by its neighbors. Each node sends a periodical *heartbeat* message to update its neighbors. If a node does not update its neighbors periodically, it will be removed from the member table. Once a node leaves, it will broadcast a "leave message" to all its neighbors. The nodes that receive this message will delete the respective node from its member table as well. Therefore, the failure of any neighbors can be detected by constantly monitoring periodical messages from neighbors.

In order to locate a better neighbor, which has higher uplink, a peer in PALMS periodically replaces the neighbor with the least contribution by selecting nodes with higher scores. This operation helps each node maintain a stable number of partners in the presence of node departures, and it also helps to discourage the existence of free riders within the network.

3.3.2 Membership Management

PALMS employs membership management protocol similar to SCAMP [GKM01]; a peer-to-peer membership management for gossip-based protocols. Probabilistic gossip-based dissemination protocols have recently emerged as an attractive alternative and provide good scalability and reliability properties. In these protocols, each member is in charge of forwarding each message to a set of other, randomly chosen, group members. This proactive use of redundant messages provides a mechanism for ensuring reliability in the face of node crashes and high packet loss rates in the network. It can also be shown that the load on each node increases only log-arithmically with the size of the group, so these algorithms are scalable. SCAMP is a simple, fully decentralized, and self-configuring membership management protocol.

SCAMP consists of mechanisms for nodes to subscribe (join) and unsubscribe (leave) from the group and for nodes to detect and recover from isolation. The partial views at nodes evolve in response to changing group membership in a fully decentralized way.

In SCAMP, each node maintains two lists, a *PartialView* of nodes it sends gossip messages to and an *InView* of nodes that it receives gossip messages from, namely nodes that contain its node-id in their partial views. If a node i decides to keep the subscription of node j, it places the id of node j in its partial view. It also sends a message to node j telling it to keep the

Input:
{*n receiving s adds it with the probability*
$p = 1/(1 + sizeofPartialView_n)$}
with probability $p = 1/(1 + sizeofPartialView_n)$}
  if $s \notin PartialView_n$ **then**
    $PartialView_n = PartialView_n + \{s\}$
  else
    Choose randomly $n \in PartialView_n$
    Send $(n_i, s, forwardedSubscription)$;
  end if

Figure 3.3: SCAMP handling a forwarded subscription algorithm

node-id of i in its *InView*.

Figure 3.2 and 3.3 show the algorithm for subscription management and forwarded subscription management. SCAMP membership management protocol only requires local information available at the node treating the subscription request. If new nodes join by sending a subscription request to a member chosen uniformly at random from existing members, then the system configures itself toward partial views of size $(c + 1)log(n)$ on average. Here, n is the number of nodes in the system and c is a design parameter.

3.3.3 Streaming Scheduling

PALMS employs a swarm-like content delivery mechanism that is similar to BitTorrent [Coh03]. The main advantages of swarming content delivery are its ability to effectively utilize the outgoing bandwidth of participating peers and its robustness against the churn.

The streaming scheduling mechanism of each node is responsible for exchanging packets with all its neighbors. Swarm-like content delivery is incorporated in PALMS. As a parent, each peer periodically generates a report i.e., *buffer map* of its newly received packets and sends it to its child nodes. As a child, each peer periodically requests a subset of required packets from each parent based on the reports received. The pull mode is deployed to fetch absent packets from its parent nodes and in turn tries its best to deliver packets requested by the neighbors. Packets requested by the pull mode are determined by the packet scheduling algorithm, which is much similar to the data-driven approach in DONet [ZLLY05]. Peer selection for PALMS depends on the rank ordering of the score-based incentive mechanism.

The buffer uses a slightly modified sliding window concept to allow for the flexibility required by PALMS streaming system. There is a fixed-width window of packets, which is allowed to slide forward if it contains at least a certain percentage of data chunks (depending on the robustness and tolerance of the encoding). An ideal case in which a node received an ordered stream of packets from his neighbors at exactly the same rate required by the player (which is in turn a parameter coming from the media encoding technique used) would show a window scrolling at the same rate of the stream.

Every node also maintains a *window of interest*, which is the set of sequence packets that the node is interested in acquiring at the current time. Figure 3.4 illustrates the fundamental concept of the sliding window. A sliding *window of availability* contains the list of segments available for each node. This is the information for the *buffer map* shared with other neighbor nodes. The node slides its window of interest forward over time as new packets stream in. If a packet has not been received by the time it "falls off" the trailing edge of the window, the node will consider that packet lost or obsolete and will no longer try to acquire it. Figure 3.5

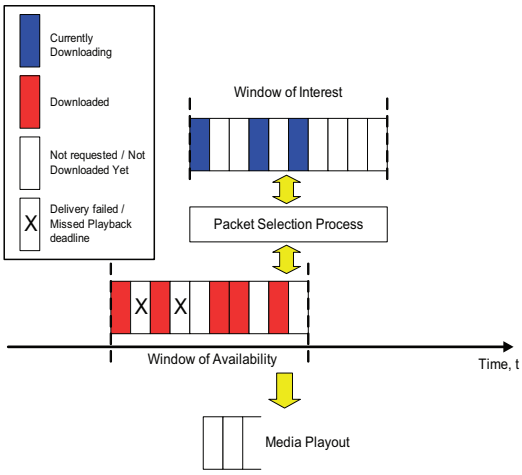


Figure 3.4: Illustration of data buffer for PALMS node

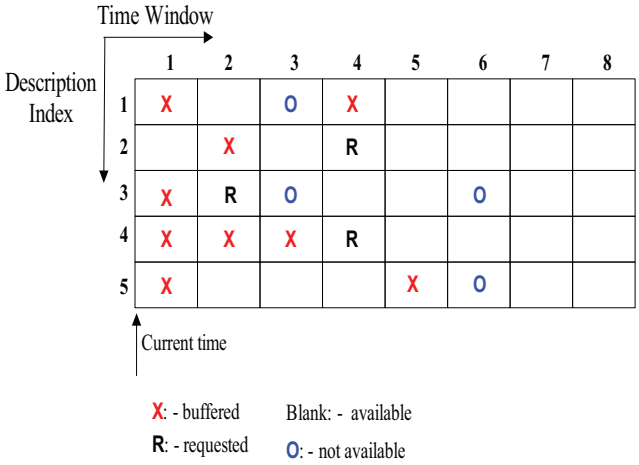


Figure 3.5: Buffer State for a PALMS node at a given time

shows the buffer state of a node at a given time.

To accommodate the bandwidth heterogeneity among peers, the content is encoded with Multiple Description Coding (MDC). Basically, MDC organizes the streaming content into several sub-streams where each sub-stream can independently decoded. The use of MDC for video streaming has been widely studied. Padmanabhan *et al.* propose that introducing redundancy can provide robustness in media streaming [PWC03]. The delivered quality to each peer is proportional to the number of independent sub-streams it receives. With MDC coding, each peer is able to receive the proper number of sub-streams that are delivered through the combination push-pull streaming mechanism.

3.3.4 Incentive mechanism

PALMS, just like any other P2P content delivery system, works on the premise that peers share resources in order to increase the total capacity of the P2P system. In the case of PALMS, it derives bandwidth from participating peers who operate independently of each other. A mechanism that creates favorable incentives for all peers to contribute resources and thus guards against bandwidth starvation in PALMS is needed to sustain peer interest in sharing

bandwidth.

We believe that peer selection for receiving packets at child nodes offers an unique opportunity to tackle both the free-riders and the streaming Quality of Service (QoS) challenges in a synergistic manner. We propose a score-based incentive mechanism that provides service differentiation in peer selection for P2P streaming. Our proposed incentive mechanism is an extension work of [HC04]. Contributors to the system are rewarded with flexibility and choice in peer selection. Free-riders are given limited options in peer selection, if any, and hence result low quality streaming.

We consider that PALMS consists of rational users who choose their contribution level in order to maximize their individual utility. The contribution level x_i of user i is converted to score S_i, which in turn mapped into a percentile rank R_i. The scoring function used in PALMS is based on the ratio of amount bytes uploaded over bytes download. Peer selection depends on the rank ordering of the requestors and candidate suppliers. For example, a peer selection scheme may allow a user to select peers with equal or lower rank to serve as suppliers. The outcome of the peer selection process is the realized quality of the streaming session. User utility U_i is a function of the streaming session quality Q and the contribution cost C:

$$U_i(x_i) = \alpha_i Q(x_i) - \beta_i C(x_i), \tag{3.1}$$

where α_i and β_i define the values of streaming quality and contribution cost to user i.

To evaluate delivery quality and quantify the performance of the media streaming system, we define quality, Q of a streaming session as:

$$Q = \frac{\sum_{i=1}^{T} V_i}{T} \tag{3.2}$$

where T is the number of packets in a streaming session and V_i is a variable that takes value 1 if packet i arrives at the receiver before or on its scheduled play-out time, and 0 otherwise. The quality is different from throughput because it considers the deadline of each packet. Basically, the parameter Q captures other performance parameters such as packet delay, packet loss and jitter.

Delivery quality can be expressed as a function of contribution, score, or rank. The quality function is system dependent. However, delivery quality should exhibit the following properties: (i) delivery quality is monotonically non-decreasing in user score, (ii) Q_{MAX} represents the highest possible quality provided by the system, (iii) Delivery Quality, Q has non-negative initial value, For example, best-effort delivery quality, $Q_{BestEffort}$=$Q(S_i$=$0)\geq 0$.

When a new node first joins the system, it begins with a score of zero and receives best-effort service $Q_{BestEffort}$=$Q(S_i$=$0)$=0. The quality of this service may vary from system to system, and vary as a function of system load. For example, a supplier node may choose to serve a node through push method with a lower score only when it is idle. Thus, best-effort service quality can be highly unpredictable and often results in lower quality. In order to improve performance and receive better quality than best-effort, a node is required to earn it by contributing to the system and in turn improve its score.

The score is a discrete variable and thus the probability density function (pdf) is defined only where the score has a meaningful value. In order to compute the percentile rank, the cumulative distribution function (cdf) of the scores is calculated. The cdf is defined as:

$$F(S) = \sum_{i=S_{low}}^{S_{high}} f(i) \tag{3.3}$$

where f is the pdf of the score. The relationship between the percentile rank and the score is provided by cdf. The percentile is obtained by dividing the cdf by the total number of nodes. The scores of all nodes are kept at the streaming server.

We would also like to point out that with systems like PALMS, it is a time sensitive traffic system. Free-riders cannot afford to wait for more time, since each packet has a certain lifetime. In other words, time constrained data distribution provides stronger incentives to peers to discourage the existence of free-riders.

3.4 PALMS : Scheduling Algorithms

The algorithms presented in this section make up the core of the PALMS system. They determine how each node chooses its partner for data exchange, how data packets to be sent are chosen and scheduled, which data packets are to be requested from each connected neighbor, and an incentive mechanism to encourage contribution of data received.

3.4.1 Analysis of Pure Pull Method

We analyzed the detailed process of pure pull method to provide insight into related issue. Basically, the pull component in PALMS is similar to the data-driven approach in DONet [ZLLY05]. Each node in PALMS periodically exchanges *buffer map* of media packets with neighbors. Based on information gathered from *buffer map*, a node then schedules which packet is to be retrieved from which neighbor accordingly.

In the pull mode of PALMS, when a packet goes from one node to another, the following three steps are executed as shown in Figure 3.6. First, the sender receives packets from a connected neighbor and stores them in buffer. (In this case, the sender is node X while the receiver is node Y). Sender X informs receiver Y about the packets stored in buffer by sending a *buffer map* packet. Second, if receiver Y needs this packet, a request is sent to the sender. Third, sender X will deliver all the requested packets to receiver Y. As depicted in Figure 3.6, at least three end-to-end delays are involved in these steps. As a result, the delivery of most packets will have extra delays for a one hop distance. We use δ_1, δ_2 and δ_3 to denote the intermittent waiting time. The total average latency for a packet transmitted in one hop T_{1hop} can be approximately computed as $\delta_1+\delta_2+\delta_3+3\delta_{xy}$, where δ_{xy} is the average end-to-end delay between nodes.

In a nutshell, the pure pull approach displays extra end-to-end latency for packet delivery. In order to improve packet delivery ratio, we propose the combination of push-pull approach. By incorporating the push-pull approach, we expect the following two significant improvements: *(i)* reduce the end-to-end delay observed at the end user node *(ii)* improve the delivery ratio of a media packet to its receiver before the playback deadline.

3.4.2 Pure Push Mechanism Trade-Off

The prime objective of the push mechanism is to quickly distribute a data block to a certain number of peers, in order to fuel the subsequent pull-based exchanges. As we have argued before, such a mechanism is needed due to the long delays of purely pull-based approaches; the pushing phase brings the data block into the vicinity of virtually all peers. While there already exist several solutions both in literature and in practice, many of these systems fail to take all the aforementioned criteria into account. This may partly be explained by the fact that some of the optimization goals are inherently antagonistic. For example, a low delay can be achieved by having each peer immediately notifies its neighbors of arrival of new packets and at the same

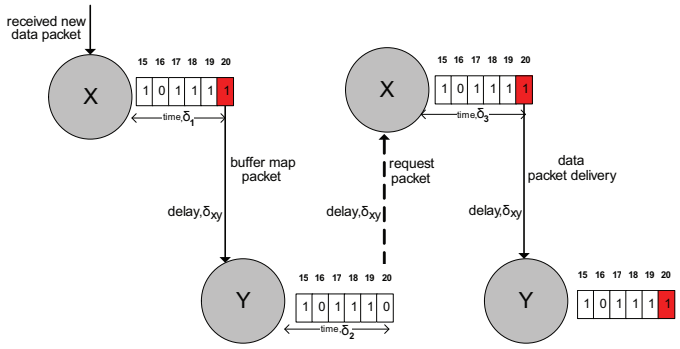


Figure 3.6: Illustration of one hop delay using pure pull method

time, forward all incoming data blocks to its neighboring peers through the push mechanism. Unfortunately, such a naive solution results in a significant overhead, as a peer may receive the same block repeatedly from different neighbors. Alternatively, peers could request missing blocks *explicitly*. Node can wait until dozens of packets arrived before inform its neighbors. This scheme is referred to as pulling since all peers have to initiate the transmission of data blocks towards themselves. While a pull-based approach circumvents the problem of receiving duplicates, it comes at the cost of intolerable latencies, as notifications and requests have to be sent back and forth. Hence, there is a trade-off between overhead and efficiency in term of live media streaming.

3.4.3 Scheduling Algorithm : Pull Mechanism

The main algorithms used for peer selection for pull and push mechanisms are an altruistic algorithm. The prime objective of the pushing component is to quickly distribute a data block to a certain number of peers, in order to fuel the subsequent pull-based exchanges. As we have argued before, such a mechanism is needed due to the long delays of purely pull-based approaches; the pushing phase brings the data block into the vicinity of virtually all peers.

The algorithm for pull methods is similar to the heuristic used in DONet [ZLLY05] and BitTorrent [Coh03]. The main purpose of the pull method is to request the rarest packets among those that are locally available, and to distribute the request across different possible suppliers. The pull algorithm is shown in Figure 3.7.

Using the information gathered from the *buffer map* exchanged among neighbor sets, packets that are rarest across the neighborhood are requested with higher priority than more common ones. Packets with the same number of suppliers are randomly requested from one of the neighbors that can provide them. This is to limit the load on any single peer. Based on the rank of individual nodes, a node is only allowed to pull packet from a supplier that has the same or lower score.

3.4.4 Scheduling Algorithm : Push Mechanism

The pull mechanism is the process of packet delivery by a neighbor after a request is made by a node. Inspired by the work conducted by [BLBS06], the push mechanism for PALMS employs two simple techniques too. The push mechanism for PALMS consists of a proactive component where each node pushes data forward, and a reactive mechanism that is triggered by NACKs.

Basically, the push mechanism sends a packet relay to neighbors as soon as the packet is received. Each node works under pure pull mode at the initial stage after joining the streaming network. After that, based on the traffic from each neighbor, the node will subscribe to the

Input:
$bw[k]$: bandwidth from neighbor k
$num\_suppliers$: number of suppliers
$bm[i]$: buffer map of connected node i
$rank[i]$: percentile rank of connected node i
$deadline[j]$: deadline of packet j
$expected\_set$: set of packets to be pulled
$set\_neighbors$: number of neighbors of the node

Scheduling :
for packet $j \in expected\_set$ **do**
Make $num\_suppliers = 0$
for $l \in \{1..set\_neighbors\}$
• Calculate T_j, Time for Transmitting packet j :
$T_j = deadline[j]$ - $current\_time$
$num\_suppliers = num\_suppliers + bm[l, i]$
end for
end for
if $num\_supplier$=1
• packets when there is only one potential supplier
for $j \in$ {1..expected_set}
$supplier[j] \leftarrow \arg_r\{bm[r, i]=1\}$
end for j
else
• packets when there are more than one potential suppliers
for $j \in$ {1..expected_set}
for $r \in$ {1..num_suppliers}
• Find r with the highest $bw[r]$ and enough
available time $t[r, j]$
$supplier[j] \leftarrow \arg_r\{$ bw[r] > bw[r'],
t[r',j] > t[r,j],$rank[j] \leq rank[r]$, r,$r' \in set\_suppliers\}$
end for
end for
end if
Output $supplier[j]$
Do Pull packets from $supplier[j]$
Do Update *Buffer Map*

Figure 3.7: Pull method heuristic scheduling algorithm

Input:
$set\_neighbors$: number of neighbors of the node
$bm[i]$: buffer map of connected node i
$rank[i]$: percentile rank of connected node k
$deadline[k]$: deadline of packet k
$expected\_set$: set of packets to be pushed

Scheduling :
for packet $k \in expected\_set$ **do**
for $l \in \{1..set\_neighbors\}$
• Find Packet with the highest time-stamp :
$T_k = deadline[k]$ - $current\_time$
end for
end for
for $receiver \in \{1..set\_neighbors\}$
• Roulette Wheel Selection for receiver
receiver's with higher $rank[i]$ are given higher probability
end for
Output $receiver[k]$
Do Push packet to $receiver[k]$

Figure 3.8: Push method heuristic scheduling algorithm

pushing packets from its neighbors accordingly at the end of each time interval. Due to the delay that might occur in a pure pull method, a push mechanism helps to increase the delivery ratio of packet to receiver nodes. Moreover, due to the unreliability of the network link or a neighbor failure, some of the packets are lost during transmission. An overlay node can detect missing packet using gaps in the packet sequence numbers. This information is used to trigger NACK-based retransmission through the next interval of push mechanism. Thus, with the help of the push mechanism, packets are pushed and received at the receiver nodes at a second time interval.

A good selection strategy is required to distribute the packets through the push mechanism. This is to ensure that every node pushes different packets in order to reduce redundancy. Push packets should also take into account the requests from neighbor nodes. The push algorithm is shown in Figure 3.8.

For push mechanism packet scheduling, each neighbor node tries to allocate different packets into the *Push Packet Map*, PPm to be pushed. A *Push Packet Map*, PPm consists of node id and packet sequence number. A simple rank based roulette wheel selection scheme is applied for the next time interval for each node to push the available segments. Packets with the highest time-stamp or *least sent* will be given higher priority to be allocated into the *Push Packet Map*, PPm. Each node keeps a counter of how many times each packet is sent. Packets with the least number of times sent will be chosen. In addition to that, packets that required retransmission based on NACKs received will be allocated into the *Push Packet Map*, PPm too.

3.5 PALMS Framework

In this section, we present a generic system architecture for the implementation of PALMS. Fig. 3.9 depicts the system framework of a PALMS node. *Network Interface* is used to broadcast

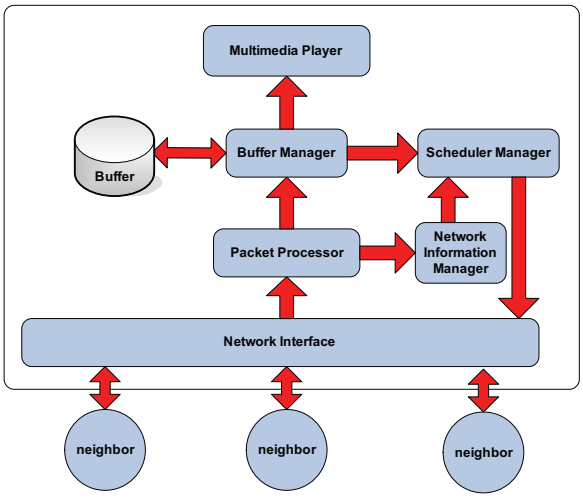


Figure 3.9: A generic system architecture for a PALMS node

and receive packets. *Multimedia Player* decodes multimedia data received and displays it on the screen. Apart from the above components, there are four key modules in the system.

1) Buffer Manager: This module manages the video data received and stores it in *Buffer Memory*. When a video packet is received from the *Packet Processor* module, it then stores and orders the data in *Buffer Memory*. In addition, *Buffer Manager* module supplies video data to *Multimedia Player* and *Scheduler Manager* for display and broadcast, respectively.

2) Neighbor Information Manager: Neighbor Information Manager stores information on connected neighbors. It helps the node maintain a partial view of other overlay nodes. Based on information obtained from buffer map, which is periodically exchanged between a node and its neighbors, *Network Information Manager* keeps track of the current list of active nodes. It also helps the node to establish and maintain partnership with other nodes.

3) Scheduler Manager: The primary function of this module is to broadcast buffer map packets periodically. It constructs a buffer map packet that indicates the descriptions received and the connected nodes with the descriptions. Then, the packet is broadcast through the *Network Interface*. Moreover, *Scheduler Manager* handles the push-pull mechanism of video packets and constructs video packets for broadcast operations with suitable video data supplied from *Buffer Manager*. Packets are sent through the push mode to qualified nodes based on the score-based incentive mechanism.

4) Packet Processor: The functions of this module are to extract information from packets (buffer map packets and video packets) received from *Network Interface* and deliver it to other modules. Upon processing a buffer map packet, the module provides neighborhood information to *Network Information Manager*. If a video packet is received, it extracts the required information and passes to *Buffer Manager* and *Scheduler Manager* module.

3.6 Advantages of the PALMS approach

In the following section, we present the advantages and disadvantages of PALMS push-pull scheduling approach with respect to the live streaming system.

3.6.1 Scalability

One of the main issues for live streaming system is the number of participating nodes in a system. However, PALMS is based on data-driven receiver-based approach P2P system that is inherently scalable. Similar to nodes in a P2P system, if each node is sharing part of the load, more nodes mean not only more demand but also more capacity. By contrast, if a service runs on a central host, more nodes will eventually mean that more resources need to be added at the host. If new host resources aren't added, the service breaks or slows to a crawl or suffers in some other way.

3.6.2 Dealing with heterogeneity

PALMS employs a swarm-like content delivery mechanism similar to BitTorrent. The main advantages of the swarming content delivery are its ability to effectively utilize the outgoing bandwidth of participating peers and its robustness against the dynamics of peer leaving or joining (or churn). The swarm-like content delivery incorporates push and pull mechanisms, for content requesting and delivering. Each peer in PALMS periodically reports its newly received packets to its neighbor peers. As a receiver peer, each peer periodically requests a subset of required packets from each supplier peer based on the reported available packet at each supplier peer and its available bandwidth. To accommodate the bandwidth heterogeneity among peers, in PALMS, the content is encoded with MDC. MDC organizes the streaming content into several sub-streams where each sub-stream can be independently decoded. The delivered quality to each peer is proportional with the number of independent sub-streams that it receives.

3.6.3 Reliable Streaming

The heuristic approach for the push and pull mechanisms allow each peer chooses itself its neighborhood according to the data it wants. Moreover this mechanism makes the message possible in order to be transmitted to all receivers without a clearly defined topology built in the overlay. As PALMS employs the data-driven and receiver based approach, which does not really structure the overlay, could be less sensitive to dynamicity of the streaming hosts, thus it is be the best approach to use for live P2P streaming protocols. With layered coding like MDC, data is encoded in several layers and it is necessary to receive at least the main layer. The other layers will only improve the quality of reception. With MDC, it is almost like layered coding but it is not necessary to receive a particular layer.

3.7 Disadvantages of the PALMS approach

3.7.1 High volume of traffic

PALMS is based on data-driven receiver-based P2P overlay network. Similar with swarm-like content delivery mechanism of BitTorrent-style networks where all participants share resources in an unstructured P2P networks. These approaches, although similar in nature, each have their own distinct disadvantages, especially when considered in relation to a scientific research community utilizing volunteer resources. However, the swarm-like content could leads to high volume of traffic. A client could send consecutive requests for packet lists to connected neighbors. A possible solution for this problem would be to have a group peers to monitor the requests made by connected nodes.

3.7.2 Lack of monitoring

Due to the distributed nature of unstructured overlay network of PALMS, lack of monitoring is one of the disadvantages of the PALMS approach. PALMS unstructured overlay networks have multiple peers that send multiple traffic to other peer. This may introduces extra data overhead for retransmits, communication and redundancy if no proper monitoring system is employed.

3.8 Summary of PALMS architecture

In this chapter, we presented an overview of the PALMS architecture, its service model and the network underlying the architecture. We described details of how PALMS clients discover and join a streaming session, locate appropriate neighbor nodes, and attach to them to participate in the streaming session. The PALMS architecture relies on unstructured overlay application level mesh network to provide an efficient and robust live media streaming distribution. Our architecture makes use of collaborative push and pull media connected from connected nodes.

The incentive mechanism in PALMS provides flexibility to select suppliers to the cooperative users to improve the streaming quality. With the incentive mechanism, the system selects best suppliers for each session and ensures that each supplier has high availability so that the suppliers do not fail often. PALMS employs rank-based incentive mechanism in order to achieve cooperation through service differentiation.

In Chapter 4, we will discuss the simulation framework to perform extensive simulations to study the performance of PALMS. Then in Chapter 5, we will evaluate and discuss the simulation results based on performance metrics and comparison with other existing streaming protocols. □

CHAPTER 4

Simulations Framework

These are my principles. If you don't like them, I have many others.
Groucho Marx.

In this chapter, we perform extensive simulations to study the performance of PALMS. Simulations on the algorithms' behavior test for different network sizes, bandwidth distributions, streaming rates, and number of free-riders using Network Simulator, ns-2 [ns2].

4.1 Simulation Software

Network Simulator, ns-2 is a discrete event simulator targeted at networking research. ns-2 provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks.

ns-2 began as a variant of the REAL network simulator in 1989 and has evolved substantially over the past few years. In 1995 ns-2 development was supported by DARPA through the VINT project at LBL, Xerox PARC, UCB, and USC/ISI. Currently ns-2 development is support through DARPA with SAMAN and through NSF with CONSER, both in collaboration with other researchers including ACIRI. ns-2 has always included substantial contributions from other researchers, including wireless code from the UCB Daedelus and CMU Monarch projects and Sun Microsystems.

Network Simulator, ns-2 software is freely available at The Network Simulator - ns-2 homepage[1].

4.2 Simulation Setup

We have conducted extensive experiments with our PALMS prototype. In this section, we first describe the simulation setup for PALMS in the Network Simulator, ns-2 environment. Sim-

[1] http://www.isi.edu/nsnam/ns/

ulations on the algorithms' behavior test for different network sizes, bandwidth distributions, streaming rates, and number of free-riders.

4.2.1 Video Data

The length of the video is 120 minutes (a typical length for a movie).

4.2.2 Video Coding

We used a video stream that is Multiple Description Coding (MDC) encoded with 5 descriptions. MDC was originally developed at Bell Laboratories, having specifically circuit switched networks in mind. The idea was to transmit data over multiple (telephone) lines where in case of a line failure it should still be possible to decode the remaining data, though this would result in a reduced quality. This method was called channel splitting. The original bit stream is partitioned into different so-called descriptions of the one source. Receiving one or more of the source descriptions allows the source image to be reconstructed to a prescribed quality level.

MDC builds on Forward Error Concealment methods; i.e. the mechanisms to deal with errors (respectively reduced quality) are already implemented in the coding process. Therefore redundant information is encoded with each descriptor so that it is possible to decode each of the descriptors separately. This is called fractional repetition of core data. The protection of the core data can be higher (Unequal Error Protection) to ensure that a descriptor can be decoded. Any additional descriptor then enhances the presentation quality. Thus, descriptors carried in different streams do not build on each other and therefore do not need to be prioritized.

MDC's coding technique which fragments a single media stream into n independent sub streams ($a \geq 2$) referred to as descriptions. The packets of each description are routed over multiple, (partially) disjoint paths. In order to decode the media stream, any description can be used, however, the quality improves with the number of descriptions received in parallel. Built on Forward Error Concealment, MDC is to provide error resilience to media streams. Since an arbitrary subset of descriptions can be used to decode the original stream, network congestion or packet loss - which are common in best-effort networks such as the Internet - will not interrupt the stream but only cause a (temporary) loss of quality. The quality of a stream can be expected to be roughly proportional to data rate sustained by the receiver.

MDC is a form of data partitioning, thus comparable to layered coding as it is used in MPEG-2 and MPEG-4. Yet, in contrast to MDC, layered coding mechanisms generate a base layer and n enhancement layers. The base layer is necessary for the media stream to be decoded, enhancement layers are applied to improve stream quality. However, the first enhancement layer depends on the base layer and each enhancement layer $a + 1$ depends on its subordinate layer a, thus can only be applied if a was already applied. Hence, media streams using the layered approach are interrupted whenever the base layer is missing and, as a consequence, the data of the respective enhancement layers is rendered useless. The same applies for missing enhancement layers. In general, this implies that in lossy networks the quality of a media stream is not proportional to the amount of correct received data.

For simplicity, we assume that all descriptions have the same constant bit rate of 100 Kbps. Therefore, the rate of the full quality version of the stream is 500 Kbps.

4.2.3 Peer Parameters

The incoming access link bandwidth for all peers are set to 500 Kbps. The incoming access links of all peers are set to 500 Kbps so that each peer can easily receive the full quality playback

Upload Bandwidth	SN1	SN2	SN3	SN4	SN5
128 kbps	5 %	15 %	10 %	50 %	25 %
256 kbps	10 %	80 %	10 %	25 %	25 %
512 kbps	40 %	5 %	80 %	25 %	25 %
0 kbps	45 %	0 %	0 %	0 %	25 %

Table 4.1: Scenarios for comparing different upload bandwidth under PALMS

rate. The playback starts 10 seconds after receiving the first packet. The buffer length is set to 30 seconds. In all our experiments we use a *heartbeat* period of 5 seconds for all simulated protocols. The interval for the next round of push mechanism is set for every 10 seconds. In order to examine the effects of aggregate available resources, bandwidth heterogeneity and free-riders, the outgoing bandwidth of individual peers can be set to one of four values : 128 Kbps, 256 Kbps, 512 Kbps and 0 Kbps. By controlling the distribution of peers across these four groups, we can control the heterogeneity of outgoing access link bandwidth, the percentage of free-riders that exist in the system i.e., with outgoing bandwidth of zero, which in turn determines the aggregate outgoing bandwidth i.e., system capacity for a given scenario. The distribution of 1000 peers across different groups is shown in Table 4.1.

4.2.4 Network Topology

Topology is generated by using Georgia Tech Internetwork Topology Models (GT-ITM) generator [Zeg96], a package for generating and analyzing graph models of internetworks. The GT-ITM topology generator can be used to create flat random graphs and two types of hierarchical graphs, the N-level and transit-stub. Several types of information are associated with nodes and edges to augment the basic topology. Each node has a string label that indicates properties of the node: an identifier indicating whether it is a transit or stub node, a global identifier for the domain to which it belongs, and a domain-local identifier.

For stub nodes, the label also indicates the identifier for the primary transit node where the stub domain is attached. Each edge has a routing policy weight that can be used to find routes that follow the standard domain-based routing outlined earlier. That is, a shortest path found using the routing policy weights will traverse transit domain(s) if and only if the two endpoints are in different domains.

The Transit-Stub generation software is written in C, and uses the Stanford GraphBase [2] for representation and manipulation of graphs. The GT-ITM release includes the necessary libraries from the Stanford GraphBase (SGB), thus the user need not download or understand this code to generate and analyze graphs. GT-ITM also contains other graph generation methods (e.g., Waxman's) for comparison and to use as intradomain topology.

In respect to simulation for PALMS, the delay on the access links are randomly selected between 5 ms to 25 ms. □

CHAPTER 5

Simulation Results and Discussions

By three methods we may learn wisdom:
First, by reflection, which is noblest;
Second, by imitation, which is easiest;
and third by experience, which is the bitterest.
Confucius.

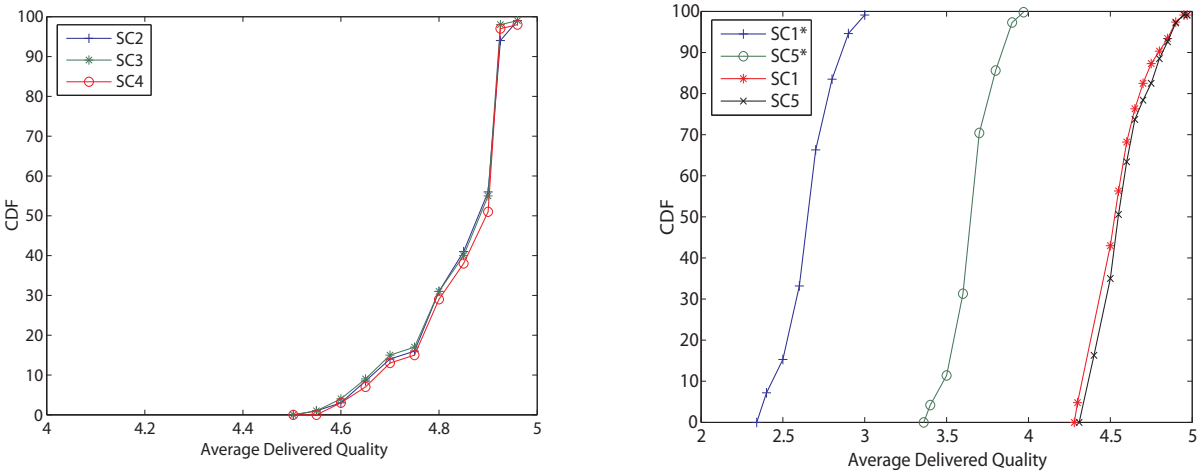
The goal of PALMS is to overcome the streaming performance failures and Quality of Service (QoS). In this chapter, we present an evaluation of how well PALMS meets these goals, and show that PALMS are beneficial to live media streaming applications because PALMS can successfully improve the quality of streaming. In many cases, PALMS can also improve the latency and throughput.

5.1 Results and Discussions

We have examined the impact of heterogeneous bandwidth and free-riders on the performance of PALMS streaming. We also study the three metrics of interest: Delivery quality, Delivery latency and Data overheads. We compare the push-pull protocol performance of PALMS with two existing streaming protocols - DONet [ZLLY05] and Chainsaw [PKT+05]. Both DONet and Chainsaw employ pure pull mechanism. DONet employs a rarest-first strategy as the block scheduling method, and select suppliers with the most surplus bandwidth and enough available time first. Chainsaw uses a purely random strategy to decide what blocks to request from neighbors.

5.1.1 Effects of Heterogeneous Bandwidth

We first examine the impact of heterogeneous uplink bandwidth on the performance of PALMS. The first question is: *"How the delivered quality of high bandwidth peers is affected by the degree of bandwidth heterogeneity and the percentage of low bandwidth peers?"*.



(a) Effect of heterogeneous upload bandwidths on PALMS (b) Effect of free-riders on delivered quality for PALMS

Figure 5.1: Simulation results on the effect of heterogeneity bandwidths and free-riders

In the experiments, we focus on three scenarios, SN2, SN3, and SN4 as shown in Table 4.1. We also examine the correlation between the delivered quality (in terms of number of description) and contributed resources vis-a-vis outgoing bandwidth of participating peers. Figure 5.1(a) depicts the CDF of delivered quality and utilization of access upload bandwidth among participating peers for these three scenarios. Based on the results obtained, figures show that the degree of heterogeneity in upload bandwidth does not affect the distribution of delivered quality of participating peers in PALMS.

5.1.2 Effect of Free-riders

We also investigate the impact of free-riders on the performance of PALMS. For the experiments, we focus on scenarios SN1 and SN5. We set the number of free-riders in the system as roughly 25% and 45% of the total number of participating peers. We examine the performances of PALMS without the implementation of the score-based incentive mechanism. Figure 5.1(b) shows that the presence of free-riders significantly reduces the delivered quality. We are aware that the scenario with free-riders can be viewed as a special case for bandwidth heterogeneity. Thus, the significant drop in delivered quality as the result of free-riders was rather surprising since the earlier result showed that heterogeneity of bandwidth does not have a major effect on performance.

As we take a closer examination of our results, they revealed that the free-riders affect the connectivity of the overlay. The explanation for such behavior is because free-riders do not have any connection to any neighbors and their presence in the overlay can affect the connectivity and content exchange between other connected nodes. This in turn limits the delivered quality to other participating nodes. Nevertheless, participating peers are not completely disconnected from the mesh network. The presence of free-riders affects the distribution of content which in turn affects the buffer requirement at each peer. Thus, while a traditional file sharing system can be sustained with low level of cooperation, a P2P streaming system cannot provide high streaming quality to its users if only a small fraction of users cooperate. In short, our results show that the presence of free-riders can significantly affect the connectivity of participating peers in the overlay network which in turn prevents content swarming among peers and thus

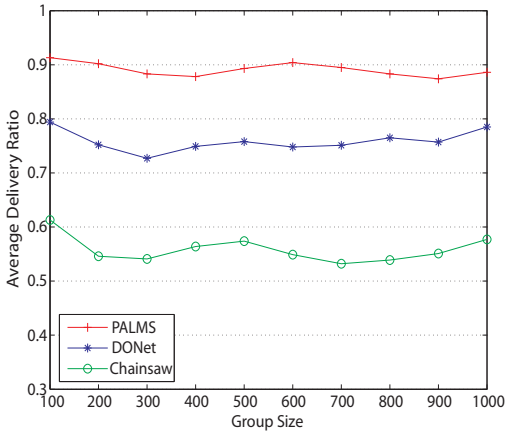


Figure 5.2: Delivery Ratio as function to Group Size.

limits the delivered quality to subset of peers. This shows that there is a need for mechanism to ensure proper connectivity among participating peers and reduce the number of free-riders in the network.

5.1.3 Effectiveness of the Incentive Mechanism

We compare the effectiveness of the incentive mechanism for PALMS with the existence of free-riders. Basically, the incentive mechanism provides flexibility to select suppliers from cooperative users to improve the streaming quality. Figure 5.1(b) shows the system performance under situations with and without the proposed incentive mechanism. Based on the results, most peers have substantially higher quality of media streaming as compared to the system that does not have an incentive mechanism. We also observe from the results of our experiments that the proper selection of peers is important for a P2P streaming session. Random peer selection yields unpredictable quality, which might be acceptable in file sharing, but not in a streaming session. On the other hand, quality-aware peer selection can provide stable and predictable quality which is a pre-condition for video applications.

5.1.4 Delivery Quality and Scalability

Content delivery among peers is performed using the combination of push scheduling coupled with pull requesting by child peers. Each peer receives content from all of its connected neighbors and provides content to all of its neighbors peers in the overlay. The requested packets from each neighbor are determined by a packet scheduling mechanism. Given a peer's playout time as well as the available content and available bandwidth among its supplier peers, this receiver-driven packet scheduling mechanism should select requested packets from each supplier peer in order to maximize its delivered quality, *i.e.,* accommodating in-time delivery of requested packets while effectively utilizing available bandwidth from all parents.

Maintaining continuous playback is a primary objective for streaming applications. To evaluate continuity, we define ***delivery ratio*** to represent the number of packets that arrive at each node before playback deadline over the total of number of packets encoded. We set the streaming rate as 500kbps. Figure 5.2 shows the average delivery ratio for PALMS in comparison to DONet and Chainsaw. From the result, we can observe that the performances for PALMS and DONet remain almost the same when group size increases. This is an indication

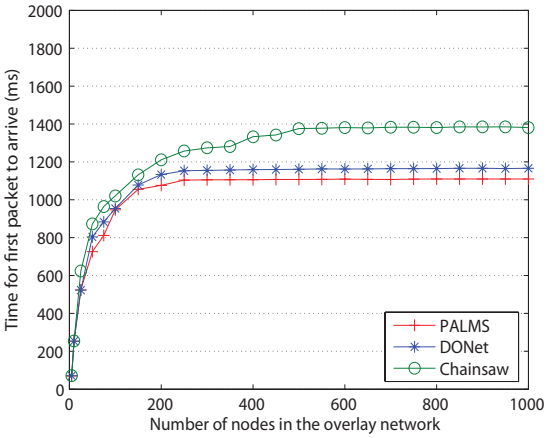


Figure 5.3: Comparison on the average time for arrival first packet.

that the performance of swarming based protocols or data-driven protocols is not affected by group size. In other words, swarming protocols have a good scalability. However, Chainsaw method decreases more in comparison to PALMS and DONet. As shown in Fig. 5.2, PALMS has 20% gains compared to DONet and over 45% gains compared to Chainsaw. We also note that the delivery quality of PALMS scales with the number of participating nodes and almost independent of the overlay network size. This is because the availability information from buffer maps are only locally exchanged. In fact, a larger overlay network size often leads to better playback continuity due to the increasing degree of cooperation.

5.1.5 Delivery Latency

Typically, users of live P2P streaming applications need to receive a stream of good quality, *i.e.* a continuous stream. In other words, protocols should limit packet losses which happen when clients leave the networks and depending peers have to recover the stream. Since it is about live streaming, protocols should minimize the duration between the moment where the clients enter the network and the moment they receive their first packet of the stream. This is all the more true if a client asks for the stream consequently to departures of peers. Thus, we compare approaches used by these protocols according to the average time for the arrival of first packet. In these simulations, we want to know the delays it takes to receive the first packet of the stream according to the protocols and their different implementations. We varied the number of clients from 1 to 1000.

Figure 5.3 shows the result of the distribution of latency experienced by data packets at the different overlay nodes. Note that all protocols suffer an increase in average time of first packet arrival, stabilize, then stay relatively constant with the number of nodes. The increase is well identified and is due to the implementation of swarming protocols for PALMS and DONet. PALMS and DONet have an average time to first packet shorter than all implementations of Chainsaw. It means that relatively PALMS and DONet are better than Chainsaw according to these metrics. This is due to the approach used by PALMS and DONet to establish peering relationships.

Group Size	Control Overheads (Control Traffic/Video Traffic) PALMS	DONet	Delivery Ratio PALMS	DONet
100	0.0173	0.0161	0.91	0.79
200	0.0175	0.0163	0.90	0.75
300	0.0183	0.0171	0.88	0.73
400	0.0184	0.0173	0.88	0.75
500	0.0196	0.0182	0.89	0.76
1000	0.0232	0.0204	0.89	0.79
2000	0.0244	0.0232	0.90	0.79
4000	0.0296	0.0270	0.90	0.80

Table 5.1: Comparison of Control Overheads for PALMS and DONet

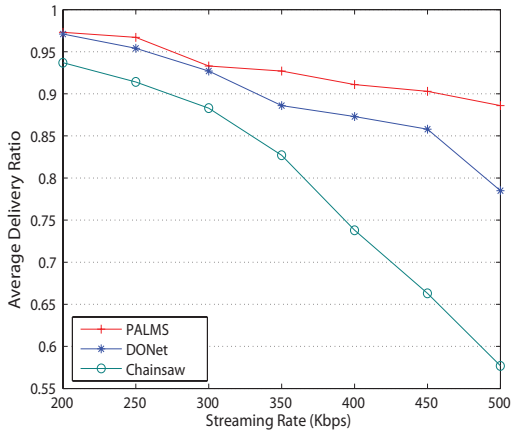


Figure 5.4: Average Delivery Ratio as a function to Streaming Rate. Group Size is 1000.

5.1.6 Data Overheads

PALMS manages the membership for all nodes in a fully distributed fashion. Each node keeps partial information about other nodes, and gossips these information to preferentially chosen neighbors. Group membership management protocols are crucial to multicasting. They provide applications with the dynamic membership information. Due to large overhead, traditional protocols are not suitable for large-scale P2P networks. As PALMS employs employs a light-weight gossip protocol, most control messages in PALMS are for exchanging data availability information. The number of nodes in a group thus becomes a key factor to the control overhead.

We compare the overheads of PALMS to DONet to achieve different delivery ratios. Table 5.1 shows that PALMS incurs very low additional data overheads to achieve relatively high delivery ratios. The control overheads at different overlay nodes increase log-arithmically with the increase in group size. The control overheads for PALMS are higher due to the additional messages such as *Push Packet Map* messages and NACKs. However the amount of increase at each overlay node is essentially minor, less than 3% of the total traffic. We believe the data overheads for PALMS can be further reduced by increasing the window size. We also observe that the data delivery ratio of PALMS is high across various group sizes.

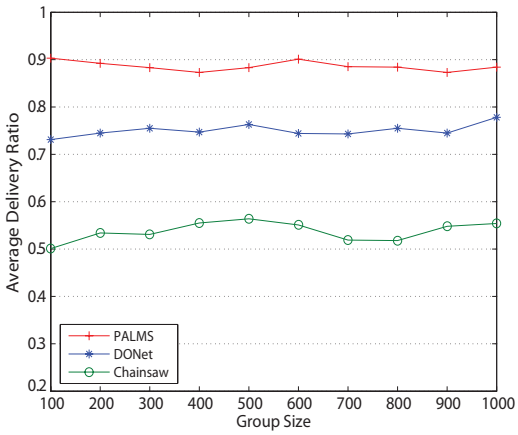


Figure 5.5: Average Delivery Ratio for different group size with 20% free-riders and streaming rate 500kbps. (Group Size 1000).

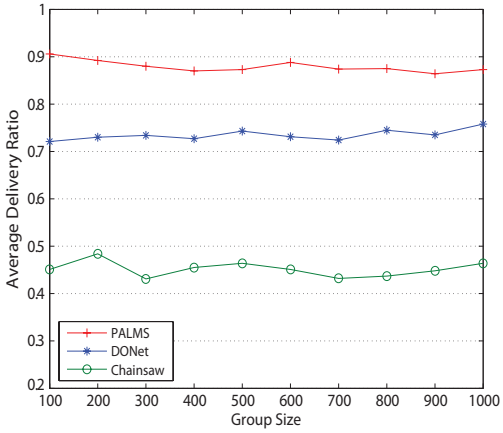


Figure 5.6: Average Delivery Ratio for different group size with 50% free-riders and streaming rate 500kbps. (Group Size 1000).

5.1.7 The impact of Free-riders

We compare the performance of PALMS, DONet and Chainsaw with the existence of free-riders. We set the number of free-riders as 20% and 50% of the total number of connected nodes. The streaming rate is set as 500kbps. Figure 5.5 and figure 5.6 show the average delivery ratio as a function to group size. As expected, the average delivery ratio for PALMS are significantly better than DONet and Chainsaw for both cases.

5.1.8 Performance under Stable Environment

We examine the performance of PALMS in comparison to DONet and Chainsaw under stable environment. We set all the nodes to join in an initialization period of around 1 minute, and then persist in the lifetime of the streaming for 120 minutes, a typical length for a movie. Figure 5.2 and 5.4 show the average delivery ratio as a function to group size and streaming rate. As mentioned earlier, we can see that the average delivery ratio basically remains almost the same when the group sizes increase.

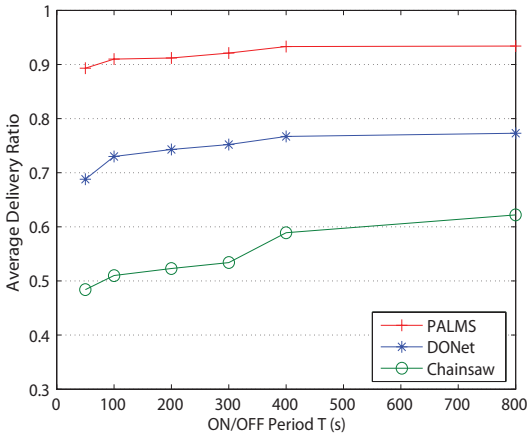


Figure 5.7: Average Delivery Ratio as a function to ON/OFF Period, T(s). Group Size is 1000.

5.1.9 Performance under Dynamic Environment

The high connectivity of our network topology and the flexible choice of neighbors allows to build up and exchange buffer map information quickly. Several scenarios have been considered in which a large fraction of peers leaves simultaneously. It turns out that it is easy to maintain the topology and to recover even from such massive concurrent network changes.

In Figure 5.7 we show the average delivery quality with dynamic node joining, leaving and failing. Most parameters settings are similar to that in the previous experiment for stable environment. For this experiment, we set each node changes its status according the ON/OFF model. The node actively participates the overlay during the ON period, and leaves (or fails) during the OFF period. Both ON and OFF periods are exponentially distributed. A severe network failure is assumed where all the peers crash without notice (no "leave message"). Figure 5.7 shows that a shorter ON/OFF period leads to a lower delivery ratio. However, the overall delivery ratio for PALMS is higher in comparison to DONet and Chainsaw because the additional push mechanism is able to help to recover from a vast majority of losses. Note that Chainsaw displays the poorest performances. Due to the fast repairing process, our system also copes well with membership changes occurring continuously over time.

5.2 Summary

In this chapter, we presented PALMS, a P2P system for live media streaming. Our systems' innovative features are the usage of the combination push-pull protocol and the presence of score-based incentive mechanisms to encourage cooperation among connected nodes.

We also examine the issues and challenges in offering P2P streaming under PALMS. In particular, we focus on the impact of outgoing bandwidth heterogeneity and free-riders on the performance of P2P streaming. We identified that P2P streaming is able to effectively accommodate the heterogeneity of uplink bandwidth, but the presence of free-riders could significantly affect the connectivity of the overlay and reduce the feasibility of data exchange among nodes. This result significantly degrades the delivered quality to a subset of peers in the system.

To successfully deploy PALMS streaming services, we proposed push-pull score-based incentive mechanism to address the issue of delivery quality, delivery latency and free-riders. We conducted simulations and we showed that a push-pull score-based incentive mechanism

achieves cooperation through service differentiation. In this framework, the contribution of a user is converted into a score and mapped into a rank, and the rank provides flexibility in peer selection that determines the quality of a streaming session. Cooperative users earn higher rank by contributing their resources to others, and eventually receive high quality of streaming. Free-riders have limited choice in peer selection, hence receive low quality streaming.

We evaluated the performance of PALMS in comparison to DONet and Chainsaw. Our simulations conducted over ns2 demonstrated that PALMS delivers quite a good playback quality even under formidable network conditions and the existence of free-riders. Our study shows that there are multiple motivating factors for having an incentive mechanism in a P2P media streaming system. First, the streaming quality is poor if the level of cooperation is low even when the network is not heavily congested. Second, unlike traditional file sharing, cooperation from a few altruistic users cannot provide high quality streaming to its users in a large system. We show that a rank-based incentive mechanism achieves cooperation through service differentiation. Cooperative users earn higher rank by contributing their resources to others, and eventually receive high quality streaming. Free riders have limited choice in peer selection, hence receive low quality streaming. □

CHAPTER 6

Extensions of PALMS

We are what we repeatedly do.
Excellence, then, is not
an act, but a habit.
Aristotle.

The previous chapter introduced and examined PALMS - P2P Unstructured Live Media Streaming model. PALMS is a live media streaming system that is based on mesh topology and the combination of push-pull scheduling protocol. Based on simulations conducted, it is shown that PALMS is able to provide reasonable streaming quality and robust to formidable network conditions and the existence of free-riders. Moreover with the usage of the combination push-pull protocol and the presence of score-based incentive mechanisms, PALMS is able to encourage cooperation among connected nodes.

A distinct, but related problem regards roles that nodes may assume: original P2P systems were based in a complete "democracy" among nodes. The common assumptions of "everyone is a peer" is generally applied. However, physical hosts running P2P software are usually very heterogeneous in terms of computing, storage and communication resources, ranging from high-end servers to low-end desktop machines. The super-peer paradigm is an answer to both issues. The super-peer approach to organize a P2P overlay is a trade-off solution that merges the client-server model relative simplicity and the P2P autonomy and resilience to crashes. The need for a super-peer network is mainly motivated by the fact to overcome the heterogeneity of peers deployed on the Internet. A super-peer connected with some ordinary peers has sufficient CPU power, bandwidth, and storage capacity and plays a role of a controller. A ordinary peer has the same ability of other ordinary peers have. Authors [YGM03] proposed some design guidelines and fundamentals characteristics are discussed. A mechanism to split node clusters is proposed and evaluated analytically. Super-peer solutions proved to be effective solutions in the real world. Applications like Kazaa (FastTrack) [kaz] and Skype [sky] are two outstanding examples. However, their actual protocols are not publicly available and thus it is difficult for other protocols to make comparison in terms of designs and performances.

Our proposed protocol, PALMS-SP is a super-peer based two-payer P2P overlay network that focuses on the monitoring aspect to improve the latency between peers and delivered streaming quality of live media streaming. We incorporate the work in [ZLLY05] by considering a combination of push-pull methods, rather than pure pull methods for the streaming mechanism. Our main objective is to reduce the end-to-end delay and in turn enhances delivered streaming quality.

6.1 PALMS-SP : System Overview

PALMS-SP is based on data-driven and receiver-based unstructured two-layer super-peer based overlay media streaming. It is designed to operate in scenarios where the nodes have heterogenous and variable bandwidth resources. For the ease of exposition, we refer to the media source as the *streaming server* and receivers as *ordinary peer*. The term *peers* and *nodes* are interchangeable, and refer to the all the ordinary peers. We consider a network consisting a large collection of nodes. The network is highly dynamic; new nodes may join at any time, and existing nodes may leave, either voluntarily or by *crashing*.

PALMS-SP consists of three major components: (i) **overlay construction mechanism**, which organizes participating peers into a two-layer super-peer based overlay; (ii) **streaming scheduling mechanism**, which determines the delivery of content from the streaming source to individual nodes through the overlay; and (iii) **super-peer management mechanism**, which determines which nodes may switch role at will from a ordinary peer to super-peer status. In the following subsections, we describe these components in PALMS-SP.

6.2 Overlay Construction

In PALMS-SP, nodes are functionally identical. They are free to exchange control information and media content data from the stream. Each peer maintains a certain number of connected nodes that are known as *neighbors*. Each node can potentially communicate with every other node in the network. Each node receives media content from a certain number of neighbor nodes and relays the content to a certain number of neighbor nodes. Nodes are heterogenous: they differ in their computational, storage capabilities, and bandwidth. Nodes may act as super-peers or ordinary nodes. Each super-peer SP is associated with a *capacity* value $max(SP)$ that represents the maximum number of ordinary nodes associated to a super-peer SP.

The basic task of the overlay construction mechanism component for each node is to be in charge of finding appropriate super-peer and neighbors for each node through the gossip method so that the application layer network can be successfully built up. To join the streaming session, a new peer contacts the bootstrapping node, (streaming server in the case of PALMS-SP) to learn about super-peers and other participating peers upon arrival. Streaming server is selected as streaming server persists during the lifetime of streaming and its identifier/address is universally known. This could be regarded as the login process. The bootstrapping node returns a list of selected super-peers that can potentially serve as parent nodes. The new peer contacts these potential super-peers to determine whether they are able to accommodate a new child node. This is by determining whether the super-peer still has enough allocation slots on the outgoing degree. In the case of PALMS-SP, each peer is associated to exactly one super-peer. The number of child nodes associated to a super-peer is pre-determined. As shown in Figure 6.1 and 6.2, an overlay network consists of two layers, namely *ordinary peers layer* (lower) and *super-peer* (higher) layers. The ordinary peer and super-peer layers are composed of a set of ordinary peers and a set of super-peers, respectively. A collection of a super-peer SP

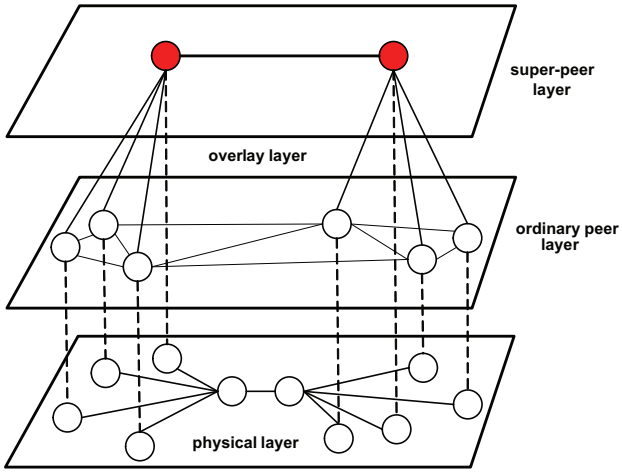


Figure 6.1: PALMS-SP : Two-layer overlay network composed of ordinary peer and super-peer layers

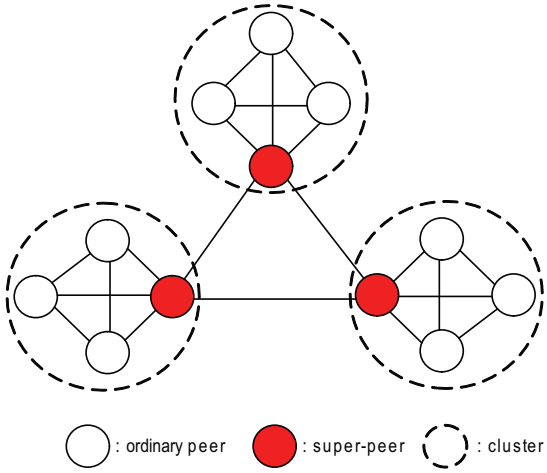


Figure 6.2: Two-Dimension Illustration of PALMS-SP that consists of super-peers and ordinary peers layers

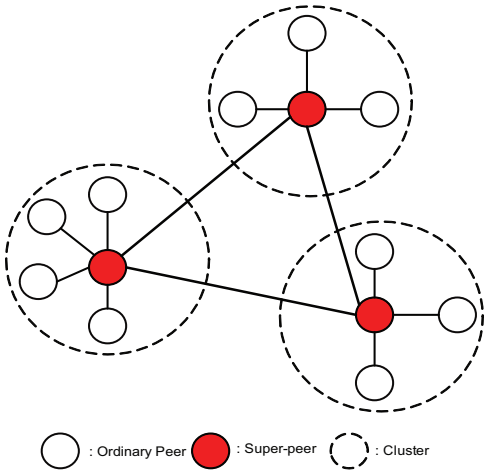


Figure 6.3: Illustration of a traditional super-peer network.

and its ordinary peers $OP_1, OP_2, ...OP_n (n \geq 1)$, and it is referred to as a *cluster* C_{SP}. A super-peer SP_i is connected with another super-peer SP_j at the super-peer layer. The PALMS-SP topology can be summarized as the following:

- each node is either super-peer or a ordinary peer;
- each ordinary peer OP is associated to exactly one super-peer SP;
- the number of ordinary nodes associated to a super-peer SP does not exceed $max(SP)$.

In traditional super-peer networks shown in Figure 6.3, ordinary peers in a cluster cannot directly communicate with each other. The ordinary peers have to communicate with each other through super-peer in the cluster. It takes at least two hops to delivery message from a ordinary peer to another ordinary peer. In this paper, we assume each ordinary peer can directly communicate with every neighbor peer in a cluster. Because of this assumption, the number of communication between a super-peer and its ordinary peers can be reduced and the super-peer has a lighter workload.

Each node has a member table that contains a list of neighbor nodes obtained from the super-peer. The information in member tables is encapsulated into a UDP packet and exchanged among neighbors periodically. Each node updates its member table in accordance with the member table sent by its neighbors. A super-peer SP holds all the information on service of every ordinary peer in a cluster C_{SP}. Each node sends a periodical *heartbeat* message to update its super-peer. If a node does not update its super-peer periodically, it will be removed from the member table. Once a node leaves, super-peer will broadcast a "leave message" to all its ordinary peers within its cluster. The nodes that receive this message will delete the respective node from its member table as well. Therefore, the failure of any neighbors can be detected by constantly monitoring periodical messages from super-peer.

In order to locate a better neighbor, which has higher uplink, a peer in PALMS-SP periodically replaces the neighbor with the least contribution by selecting nodes with higher scores (the ratio of uploaded packets over downloaded packets). This operation helps each node maintain a stable number of partners in the presence of node departures, and it also helps to discourage the existence of free riders within the network.

6.3 Super-Peer Management Mechanism

It is a feature of the super-peer based P2P system that the OP should select only one SP for sharing resources and can participate in the network only through the chosen SP. Compared with pure P2P systems, super-peer based P2P systems have to deal well with a large number of queries from OPs. The selected SP must handle queries efficiently and search for files requested by the OP. At the super-peer layer, a super-peer is connected with other super-peers in a flat unstructured overlay network. The ordinary peer and super-peer layers are composed of a set of ordinary peers and a set of super-peers respectively. One of the main obstacle for super-peers network is the super-peer selection.

6.3.1 Super-peer Selection Problem

The super-peer selection problem is highly challenging because in the peer-to-peer environment, a large number of super-peers must be selected from a huge and dynamically changing network in which neither the node characteristics nor the network topology is known priori. Often they use simple strategies such as random selection, when an OP chooses a SP. Although this technique is simple, it does not deal well with the heterogeneity of the participating peers both in terms of dynamic capabilities and a content similarity. Thus, simple strategies such as random selection don't work.

We broadly define the super-peer selection problem as that of selecting some subset of the peers in a large scale peer-to-peer overlay network to take a special role, with the designated super-peer providing service to the ordinary peers. Super-peer selection is more complex that classic *dominating set* and *p-centers* from graph theory, known as the NP-hard problems, because it must respond to the dynamicity of nodes join and leave (churn) and function in an environment that is highly heterogeneous. Since the original purpose of super-peer based P2P systems was to improve performance, we should consider search efficiency and network performance in the design of the system.

6.3.2 Super-peer Distribution Criteria

The super-peers must be distributed throughout the peer-to-peer overlay network in a topologically sensitive way to meet one or more of the distribution criteria listed below.

- *Access:* ordinary peers must have low latency access to one or more super-peers. Access can be measured in hop counts or delay.
- *Dispersal:* super-peers must be evenly distributed throughout the overlay network; they should not be clustered within only a few subregions of the overlay.
- *Proportion:* a pre-specified global ratio of super-peers to ordinary peers must be maintained to meet application-specific performance requirements.
- *Load balance:* super-peers should not serve more than k ordinary peers, where k can be configured locally based on the resource capability of each super-peer.

Note these criteria are inter-related in that specification of requirements for one may impact another, or a given application may require multiple criteria.

6.3.3 Dominating sets, p-centers, and leader election

A wealth of research in graph theory, location theory, and distributed computing provides a formal foundation for the super-peer selection problem. The basic *dominating set problem* is the problem of finding a minimal subset of the vertices in graph G, called the dominator set, such that every node is either a dominator or adjacent to a dominator. Dominating set problems and algorithms are described thoroughly in [HHS98]. Most versions are NP-hard.

Distance domination seeks to find a minimum size d-dominating set such that the distance from an arbitrary node to a dominator is $\leq$ d. *Multiple (c,d)-domination* requires that every peer be within distance d of c dominators. *Colored domination* presumes that each node in graph G has an associated color from the set $c_1, c_2, ...c_n$. A dominating set of color c_i is one in which the dominators are all of that color. Colored domination can be used to model heterogeneous networks in which only certain nodes are qualified to be dominators.

A *secure dominating set* is a subset of vertices S such that for any vertex v not in S there exists a neighbor u of v in S such that, if we add v to S and remove u from S, we get another secure dominating set S. A *global defensive alliance* is a variant of dominating set where the set S is such that every vertex v not in S has a neighbor in S and every vertex v in S has a majority of its neighbors in S. A *global offensive alliance* is such that every vertex not in S has a majority of its neighbors in S. A *k-defensive dominating set* is a set of vertices that can counter an attack on any k vertices where an attack on a vertex must be countered by itself or by a neighbor.

The p-center problem is applicable when placing a *fixed* number of super-peers in a network. Algorithms and variations on this NP-hard discrete location problem are found in [HM79]. The *p-center problem* is the problem of finding a subset of p vertices in a graph G, called centers, to minimize the maximum (or total) distance between a non-center node and its nearest center. *Colored p-centers* can be used in a colored graph for the problem of finding a subset of p_i vertices of color c_i to minimize the above distance criteria.

The classic leader election problem from distributed computing differs from super-peer selection in that the former assumes all nodes vote (directly or indirectly) on the choice of each super-peer. Leader election algorithms are not scalable because they require broadcasting or passing a token to all nodes. The best known leader election protocols electing a leader (typically the node with highest ID number) under various fault tolerant scenarios, such as Ring, Bully, etc.

Heuristic algorithms developed for these classic problems have been utilized in the field of networking, but their applicability is usually limited to smaller scale, static networks. For the most part, they involve centralized algorithms or high message passing overhead. These algorithms were not designed for large scale peer-to-peer networks that exhibit a high degree of churn and that are dynamically heterogeneous.

6.3.4 Gnutella : Super-peer selection

The best know example of super-peer selection in a peer-to-peer application is the *gnutella* [gnu] protocol for selection of ultrapeers - peers with sufficient bandwidth and processing power to serve as proxies for other peers. The use of ultrapeers reduces network traffic and speeds up content delivery. In gnutella, any peer can select itself as an ultrapeer if it meets the following requirements : it has been up for at least 5 minutes, has high bandwidth, sufficient processing power, able to handle a large number of simultaneous TCP connections, and if not behind any firewall or NAT. The ultrapeer selection protocol dynamically adjusts the number of super-peers as follows: if a leaf peer cannot find an ultrapeer with free slots, it can promote itself to be an ultrapeer. Ultrapeers also can downgrade themselves when they are no longer serve

as any leaf nodes, or through negotiation with nearby peers. In term of cluster size, there is a tradeoff between aggregate and individual load. It is good to choose a cluster size that is small enough to keep a reasonable individual load and provide reliability to the system, but large enough to avoid the knee in aggregate load when cluster size is small. For PALMS-SP, we employ a simple heuristic protocol for super-peer selection.

6.3.5 SOLE: Super-peer selection in structured overlay networks

SOLE is designed to select a group of super-peers in a structured overlay network, with a goal of keeping the super-peer to ordinary peer ratio stable as peers join and leave the overlay. SOLE also maintains low access from ordinary peers to super-peers and provides load balancing for each super-peer.

A DHT (Distributed Hash Table) built on a structured overlay network such as CAN [RFH+01], Chord [SMK+01], and Pastry [RD01] makes use of a symmetric, regular node label space, in which each physical node owns a virtual subspace in the overlay. In these structured overlay networks, a compact *node label expression* can encode a (large) collection of virtual nodes. SOLE exploits this notation and uses a node label expression to designate a subset of the virtual node label space as super-peers. The super-peer label expression is stored in the DHT for fast and easy lookup. The number of super-peers can be expanded simply by changing the node label expression. Because a structured overlay network maps physical nodes to virtual subspaces in a manner that is sensitive to both density and topology, the super-peers are evenly distributed among physical ordinary peer nodes and every ordinary peer has one or more nearby super-peers.

Initiation of super-peer selection: A node initiates the super-peer selection procedure for some service by hashing information about the : service into the DHT: the public key of the initiator and the super-peer selection policy. This policy contains the super-peer label expression, the minimum criteria for a node to be a super-peer, and maximum lifetime of a super-peer. An ordinary peer can discover the identity of super-peers for this service by accessing the DHT using the service related key to lookup the super-peer label expression.

super-peer takes charge of the service: The initiator can send a message towards the super-peer labels to inform the chosen super-peers. The notification is done via multicast in the overlay network. Each physical node that owns a node whose label matches the super-peer label expression will receive the message and become a super-peer. Alternatively, a super-peer takes charge upon receiving the first request from an ordinary peer.

Ordinary peer joins service: If an ordinary peer wants to join a service, it looks up the super-peer label expression in the DHT. The ordinary peer can then figure out the nearest super-peer in the virtual overlay according to the label-based routing protocol. The structured overlay network provides bounded virtual routing with path length proportional to the distance between labels in the label space. In CAN the ordinary peer uses the Cartesian distance between its own label and the super-peer label to estimate distance. In Pastry, the ordinary peer uses the bit-wise XOR operation to compute the label distance from which it estimates the physical distance.

Many peer-to-peer applications built on structured overlay networks can benefit from SOLE. For example, SOLE can be used to dynamically select super-peers to act as rendezvous points for applications such as cycle sharing, publish/subscribe, or storage sharing. Research in resource discovery in peer-to-peer cycle sharing systems has shown that if rendezvous points are used to collect resource information and to match queries from clients, the performance will be dramatically improved compared with probing-based or advertisement-based resource discovery methods.

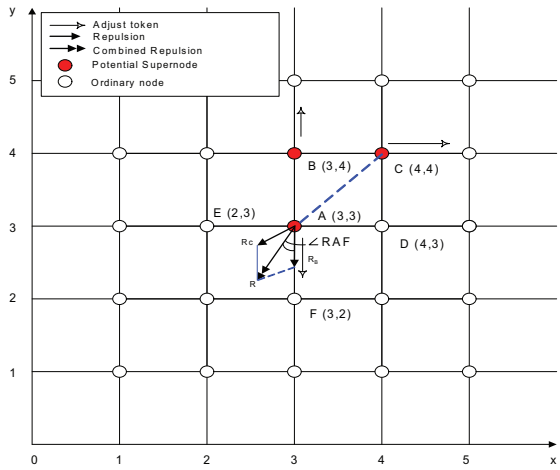


Figure 6.4: Illustration of PoPCorn repulsion protocol.

6.3.6 PoPCorn: Super-peer selection on a coordinate-based overlay network

PoPCorn assumes an n-dimensional Euclidean coordinate space using an Internet coordinate system such as GNP [NZ02] or Vivaldi [DCKM04]. PopCorn is suited for applications that wish to select a fixed set of k super-peers and distribute them evenly throughout the overlay, to perform a service such as security monitoring, protocol testing, or data repositories. PoPCorn's primary distribution criteria, *dispersal*, is achieved by maximizing the sum of inter-node distances between all pairs of super-peers. PoPCorn also achieves good *access* from ordinary peers to super-peer, and can be easily extended to address heterogeneity, adaptability, and fault tolerance.

The PoPCorn protocol selects k super-peers by dispersing k tokens through the overlay coordinate space using a repulsion model among the tokens, analogous to forces among charged particles. Each token represents one of the super-peers which moves through the overlay based on the forces exerted on it by other tokens. When equilibrium is reached, each node holding a token is selected as a super-peer.

Initial Token Placement: The Initiator sends out k tokens to random peers in the overlay. Each peer can receive at most one token. Any peer which receives a token becomes a potential super-peer. The initiator can distribute the tokens itself or it can ask its neighbors to help distribute tokens.

Token Adjustment: The repulsion model is used to adjust the location of the tokens. After a node receives a token and becomes a potential super-peer, it will start a scoped gossiping session with its neighbors to tell them the coordinates of the token it holds. Whenever a gossiping message arrives, a potential super-peer will recalculate the combined force vector of repulsions from nearby tokens. If the magnitude of the combined repulsions exceeds a threshold TR, this potential super-peer will pass its token to the neighbor whose position is closest to the direction of the combined repulsion vector.

Figure 6.4 illustrates the repulsion model with a simple example in which the nodes are evenly distributed in a 2-dimensional Euclidean space. There are three potential super-peers: A, B, and C. Node A receives two repulsions R_C and R_B, from nodes C and B, respectively. The combination of the two repulsions is vector R. Among A's neighbors, F has the smallest angle with R ($\angle\ RAF$ is the smallest). If the magnitude of R exceeds A's threshold, A will pass its token to node F, which will became a new potential super-peer.

PoPCorn was designed as part of the CCOF (Cluster Computing on the Fly) [ZL04] project

for peer-to-peer cycle sharing (harnessing idle cycles throughout the Internet). PoPCorn places tasks that collectively form a distributed point-of-presence (PoP) application into a cycle sharing overlay network. PoP applications typically have low CPU and moderate communication requirements. Examples include security monitors, Internet measurement monitors, and distributed protocol testing. Compared to volunteer systems, like NETI@home, PoPCorn can better satisfy the distribution criteria and place tasks evenly throughout the overlay.

6.3.7 H_2O : Super-peer selection in unstructured overlay networks

The H_20 (Hierarchical 2-level Overlay) protocol [LZL+05] for super-peer selection is a distributed negotiation protocol for unstructured overlay networks. It is essentially a scalable protocol for multiple (c, d) colored domination that addresses the following super-peer selection requirements: access, load balance, and fault tolerance in a dynamic and heterogeneous environment, and some security issues.

H_20 is currently used to create a 2-level hierarchy for communication among security monitors within the Sequoia collaborative security monitoring system. The super-peers form themselves into an overlay network which is utilized as a backbone for fast and secure information dissemination.

6.3.8 PALMS-SP : Super-peer selection Mechanism

In this section, we demonstrate that many peer-to-peer and networking applications are seeking solutions to variations on the same fundamental problem. We first define a general model for the super-peer selection problem, describing its key requirements and challenges. We show how the super-peer selection problem is related to dominating set and p-centers problems, and describe how the super-peer selection problem is instantiated in several well-known peer-to-peer applications.

Standard networking techniques such as random selection of peers, or flooding-based search for suitable SP have significant drawbacks. SPs selected by a random strategy may not be the best and flooding-based searches for a large number of SP neither scales nor adapts well to dynamic changes in network. Therefore, when the OP selects the SP, we must consider how well the SP can deal with queries to provide OP's requested files as accurately and quickly as possible.

We adopt the super-peer selection protocol which is similar to the H_2O (Hierarchical 2-level Overlay) [LZL+05] protocol for super-peer selection. The basic idea behind super-peers management mechanism for PALMS-SP is simple and intuitive. Ordinary peers with similar locality e.g., *IP addresses* are connected to the same super-peer. At the initial stage, all nodes start as ordinary peers. Nodes may switch role at will. The decision process is completely decentralized. An ordinary peer selects one super-peer to send queries and share resources. Since the ordinary peer depends on super-peer's capabilities, the ordinary peer should select the super-peer which can provide it with the best service. There are many metrics that may be used to select the best super-peer, such as average response time, bandwidth, processing capabilities, storage and so on. These metrics may have different weights depending on the objective. For PALMS-SP, we focus on response time, bandwidth and processing capabilities. In order to be selected as super-peer, ordinary peer must obtain reasonable scores for all the metrics. A super-peer can switch back to ordinary peer role only when a super-peer has lost all its clients due to nodes leaving or crashing. Super-peers exchange connected ordinary peers information at the super-peers layer. Information of connected ordinary peers is encapsulated into a UDP packet and exchanged among super-peers periodically.

Super-peer Advertisement: A node capable of serving as a super-peer advertises itself to its neighbors within a certain scope. The advertisement includes information about its qualifications to be a super-peer (such as trust level, uptime, bandwidth, and neighborhood size). Each advertisement is propagated a certain number of hops (set by each individual super-peer) and carries with it the route travelled, i.e. an ordered list of the overlay nodes visited. This information is used by ordinary peers as part of the selection criteria. A node that receives an advertisement message caches the advertisement about the potential super-peer in its local cache.

Super-peer Search. If a new node want to find super-peers, it first consults its local cache for super-peer candidates. If there are no suitable candidates in the cache, it queries its immediate neighbors. If a contacted neighbor is a super-peer, it replies to the requestor with its qualifications. If a contacted neighbor is not a super-peer, it replies with entries from its local cache and the requestor will cache these new responses. The requestor then chooses the best candidate(s) according to its own criteria, and applies to those candidate super-peers. The contacted super-peers can confirm or reject such requests. If the requestor does not hear from anyone within a given time interval or is not satisfied with the current super-peers, it has several choices: if it is qualified, it can declare itself a super-peer and begin the advertisement protocol; it can join the overlay at another node, or it wait for a random amount of time and try again.

The super-peers form themselves into an overlay network which is utilized as a backbone for fast and secure information dissemination. Only nodes that hold a security certificate (obtained from a central authority) are eligible to be super-peers. An ordinary peer can check the trust level information in the certificate presented by a super-peer. An ordinary peer can also choose a super-peer based on trust-based routing criteria by evaluating the trust level of nodes along the path to the potential super-peer.

6.4 PALMS-SP : Scheduling Algorithm

Given the buffer map of a node and that of its partners, a schedule is to be generated for fetching the expected segments from the partners through the pull and push mechanisms. For a homogenous and static network, a simple round-robin scheduler may work well, but for a dynamic and heterogeneous network, a more intelligent scheduler is necessary. Specifically, the scheduling algorithm strikes to meet two constraints: the playback deadline for each segment, and the heterogeneous streaming bandwidth from the partners. If the first constraint cannot be satisfied, then the number of segments missing deadlines should be kept minimum. This problem is a variation of the Parallel machine scheduling, which is known NP-hard. It is thus not easy to find an optimal solution, particularly considering that the algorithm must quickly adapt to the highly dynamic network conditions. Therefore, we resort to a simple heuristic of fast response time.

6.4.1 PALMS-SP : Pull Mechanism

The main algorithms used for peer selection for pull and push mechanisms are an altruistic algorithm. PALMS-SP's pull heuristic algorithm first calculates the number of potential suppliers for each packets (i.e., the partners containing in their buffers). Since a packet with less potential suppliers is more difficult to meet the deadline constraints, the algorithm determines the supplier of each packet starting from those with only one potential supplier, then those with two, and so forth. Among the multiple potential suppliers, the one with the highest bandwidth and enough available time is selected. The main purpose of the pull method is to

Input:
$bw[k]$: bandwidth from neighbor k
$num\_suppliers$: number of suppliers
$bm[i]$: buffer map of connected node i
$deadline[j]$: deadline of packet j
$expected\_set$: set of packets to be pulled
$set\_neighbors$: number of neighbors of the node

Scheduling :
for packet $j \in expected\_set$ **do**
Make $num\_suppliers = 0$
for $l \in \{1..set\_neighbors\}$
• Calculate T_j, Time for Transmitting packet j :
$T_j = deadline[j]$ - $current\_time$
$num\_suppliers = num\_suppliers + bm[l, i]$
end for
end for
if $num\_supplier$=1
• Potential supplier = 1
for $j \in$ {1..expected_set}
$supplier[j] \leftarrow \arg_r\{bm[r, i]=1\}$
end for j
else
• Potential Suppliers > 1
for $j \in$ {1..expected_set}
for $r \in$ {1..num_suppliers}
• Find r with the highest $bw[r]$ and enough
available time $t[r, j]$
$supplier[j] \leftarrow \arg_r\{$ bw[r] > bw[r'],
t[r',j] > t[r,j],$r, r' \in set\_suppliers\}$
end for
end for
end if
Output $supplier[j]$
Do Pull packets from $supplier[j]$
Do Update *Buffer Map*

Figure 6.5: PALMS-SP : Pull Method Heuristic Algorithm

Input:
$set\_clients$: number of connected client nodes
$bm[i]$: buffer map of connected client node i
$deadline[k]$: deadline of packet k
$expected\_set$: set of packets to be pushed

Scheduling :
for packet $k \in expected\_set$ **do**
for $l \in \{1..set\_clients\}$
• Find Packet with the highest time-stamp :
$T_k = deadline[k]$ - $current\_time$
end for
end for
for $receiver \in \{1..set\_clients\}$
• Roulette Wheel Selection for receiver
end for
Output $receiver[k]$
Do Push packet to $receiver[k]$

Figure 6.6: PALMS-SP : Push Method Algorithm

request the rarest packets among those that are locally available, and to distribute the request across different possible suppliers. The algorithm for pull methods is similar to the heuristic used in DONet [ZLLY05] and BitTorrent [Coh03]. The pull algorithm is shown in Figure 6.5.

Using the information gathered from the *buffer map* exchanged among neighbor sets, packets that are rarest across the neighborhood are requested with higher priority than more common ones. Packets with the same number of suppliers are randomly requested from one of the neighbors that can provide them. This is to limit the load on any single peer.

6.4.2 PALMS-SP : Push Mechanism

The push mechanism is the process of packet delivery by a super-peers to connected clients. Inspired by the work conducted by [BLBS06], the push mechanism for PALMS-SP employs two simple techniques too. In a nutshell, the push mechanism consists of a proactive component where data packets are pushed forward by super-peer to connected clients, and a reactive mechanism where packets are pushed forward based NACKs information received.

In order to increase delivery ratio, each super-peer at the super-peers layer, proactively send data packets to connected ordinary peers. The priority of data packets to be pushed is based on the *least frequently used (LFU) policy*. Moreover, due to the unreliability of the network link or a neighbor failure, some of the packets are lost during transmission. An overlay node can detect missing packet using gaps in the packet sequence numbers. This information is used to trigger NACK-based re-transmission through the next interval of push mechanism for the super-peer. Thus, with the help of the push mechanism, packets are pushed and received at the receiver nodes at a second time interval. A good selection strategy is required to distribute the packets. This is to ensure that each super-peer pushes packets that are not too close to the playout deadline and helps to reduce redundancy in push packets. Push packets also take into account the NACK requests from connected nodes. The push algorithm is shown in Figure 6.6.

For the push packet scheduling, each super-peer tries to allocate packets that are least frequently used (LFU) into the *Super-Peer Packet Map*, $SPPm$ to be pushed. A *Super-Peer Packet Map*, $SPPm$ consists of node id and packet sequence number. A simple roulette wheel

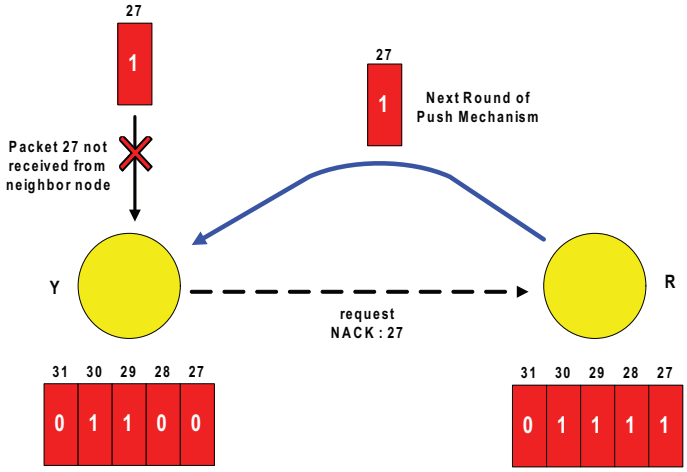


Figure 6.7: PALMS-SP : Delivery Ratio as function to Group Size.

selection scheme is applied for the next time interval for each super-peer to push the available segments. Packets with the highest time-stamp or *least sent* will be given higher priority to be allocated into the *Super-Peer Packet Map*, $SPPm$. Each super-peer keeps a counter of how many times each packet is sent. Packets with the least number of times sent will be chosen. In addition to that, packets that required re-transmission based on NACKs received will be allocated into the *Super-peer Packet Map*, $SPPm$.

6.5 Simulation Scenario

In this section, we perform extensive simulations to study the performance of PALMS-SP. Simulations on the algorithms' behavior test for under different user arrival/ departure patterns, different network sizes, bandwidth distributions, and streaming rates using network simulator ns-2 [ns2].

6.5.1 Simulation Parameters

In this section, we first describe the simulation setup and parameters for PALMS-SP in the Network Simulator, ns-2 environment.

1) Video Data: The length of the video is 120 minutes (a typical length for a movie).

2) Video Coding: We used a video stream that is Multiple Description Coding (MDC) encoded with 5 descriptions. For simplicity, we assume that all descriptions have the same constant bit rate of 100 Kbps. Therefore, the rate of the full quality version of the stream is 500 Kbps.

3) Peer Parameters: The incoming access link bandwidth for all peers are set to 500 Kbps. The incoming access links of all peers are set to 500 Kbps so that each peer can easily receive the full quality playback rate. The buffer length is set to 30 seconds. The playback starts 10 seconds after receiving the first packet. In all our experiments we use a *heartbeat* period of 5 seconds for all simulated protocols. The interval for the next round of push mechanism is set for every 5 seconds. Table 6.1 shows simulation parameters used in the evaluation of PALMS-SP and their values. Let N_P , N_{SP} , and N_{OP} mean the number of peers, super-peers, and ordinary peers in each overlay network, respectively. Since there are two roles, a super-peer and an ordinary peer, in each network topology, the number of peers in the network topology is equal to the number of super-peers and ordinary peers in the network topology ($N_P = N_{SP} + N_{OP}$). Each

Parameters	Assigned Values
Number of peers, N_P	A
Number of super-peers, N_{SP}	A/10
Number of ordinary peers, N_{OP}	A-A/10
Number of clusters, N_c	A/10
Cluster size C_{Size}	A/10

Table 6.1: Simulation Parameters for PALMS-SP

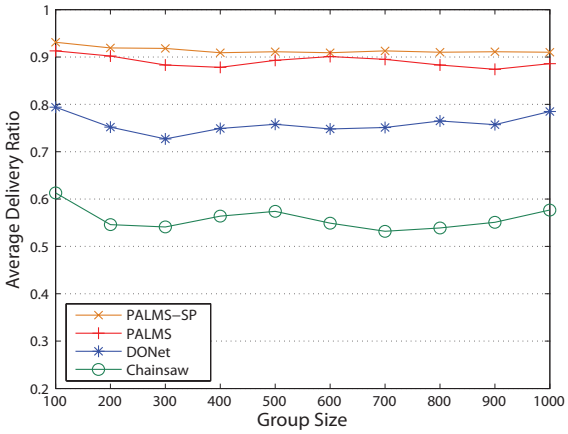


Figure 6.8: PALMS-SP : Delivery Ratio as function to Group Size.

cluster has only one super-peer, i.e. the number of clusters is equal to the number of super-peers ($N_C = N_SP$). The cluster size C_{Size} shows the number of a super-peer and ordinary peers in a cluster and is defined as $N_{SP}/N_C + N_{OP}/N_C = (N_{SP} + N_{OP})/N_C = N_P/N_C$. Here, the number of clusters multiplied by the cluster size shows the number of peers in the network topology ($N_P = N_C \times C_{Size}$). A link between a super-peer and an ordinary peer is symmetric and a link between super-peers may be asymmetric.

4) Network Topology: Topology is generated by using Georgia Tech Internetwork Topology Models (GT-ITM) generator [Zeg96]. The delay on the access links are randomly selected between 5 ms to 25 ms.

5) Performance Metrics: We use three basic Quality of Service (QoS) performance metrics, i.e., Average Delivery Ratio, Delivery Latency and Data Overheads.

6.6 Simulation Results

We have examined the impact of heterogenous bandwidths and different nodes arrival/departure patterns on the performance of PALMS-SP streaming. We also study the three metrics of interest: Delivery quality, Delivery latency and Data overheads. We compare the push-pull protocol performance of PALMS-SP with two existing streaming protocols : DONet [ZLLY05] and Chainsaw [PKT+05]. Both DONet and Chainsaw are quite successful in term of implementation of live streaming application and both streaming protocols employ pure pull mechanism. However, DONet employs a rarest-first strategy as the block scheduling method, and select suppliers with the most surplus bandwidth and enough available time first. Chainsaw on the other hand, uses a purely random strategy to decide what blocks to request from neighbors.

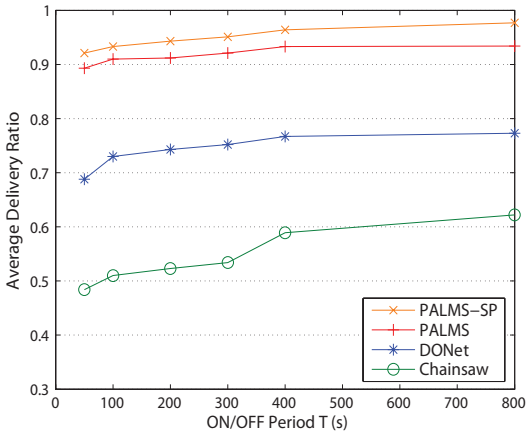


Figure 6.9: PALMS-SP : Average Delivery Ratio as a function to ON/OFF Period, T(s). Group size = 1000.

6.6.1 Comparison of Delivery Quality and Scalability for PALMS-SP

Content delivery among peers is performed using the combination of push scheduling coupled with pull requesting by child peers. Each peer receives content from all of its connected neighbors and provides content to all of its neighbors peers in the overlay. The requested packets from each neighbor are determined by a packet scheduling mechanism. Given a peer's playout time as well as the available content and available bandwidth among its supplier peers, this receiver-driven packet scheduling mechanism should select requested packets from each supplier peer in order to maximize its delivered quality, *i.e.,* accommodating in-time delivery of requested packets while effectively utilizing available bandwidth from all parents. In order to increase the content delivery, PALM-SP employs push mechanism at the super-peers layers. Moreover, due to the unreliability of the network link or a neighbor failure, some of the packets are lost during transmission. A node in the overlay network can detect any missing packets using gaps in the packet sequence numbers. This information is shared with connected neighbors and used to trigger NACK-based re-transmission through the next interval of push mechanism for the super-peer. Thus, with the help of the push mechanism, packets are pushed and received at the receiver nodes at a second time interval

Maintaining continuous playback is a primary objective for streaming applications. To evaluate continuity, we define ***delivery ratio*** to represent the number of packets that arrive at each node before playback deadline over the total of number of packets encoded. We set the streaming rate as 500kbps. Figure 6.8 shows the average delivery ratio for PALMS-SP in comparison to PALMS, DONet and Chainsaw. From the result, we can observe that the performances for PALMS-SP, PALMS and DONet remain almost the same when group size increases. This is an indication that the performance of swarming based protocols or data-driven protocols is not affected by group size. In other words, swarming protocols have a good scalability. However, Chainsaw method decreases more in comparison to PALMS-SP, PALMS and DONet. As shown in Fig. 6.8, PALMS-SP has 2% gains compared to PALMS, 20% gains compared to DONet and over 45% gains compared to Chainsaw.

Another key question is how PALMS-SP streaming mechanism scales with the number of participating nodes. Figure 6.8 clearly shows as the overlay network sizes grows, the delivery quality is almost independent of the overlay network size. As summary, PALMS-SP is scalable in terms of both overlay size and streaming rate.

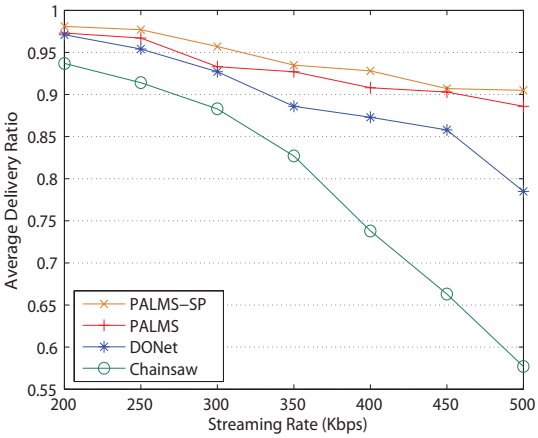


Figure 6.10: PALMS-SP : Average Delivery Ratio as a function to Streaming Rate. Group Size is 1000.

6.6.2 Performance under Dynamic Environment for PALMS-SP

The high connectivity of our network topology and the flexible choice of neighbors allows to build up and exchange buffer map information quickly. Several scenarios have been considered in which a large fraction of peers leaves simultaneously. It turns out that it is easy to maintain the topology and to recover even from such massive concurrent network changes.

We also tested the performances of PALMS-SP in comparison to PALMS, DONet and Chainsaw under dynamic network environment. We set all the nodes to join in an initialization period of around 1 minute, and then we set each node changes its status according the ON/OFF model. The node actively participates the overlay during the ON period, and leaves (or fails) during the OFF period. Both ON and OFF periods are exponentially distributed. Figure 6.9 shows that a shorter ON/OFF period leads to a lower delivery ratio. However, the overall delivery ratio for PALMS-SP is higher in comparison to PALMS, DONet and Chainsaw because the additional push mechanism employed at the super-peer layer is able to help to recover from a vast majority of losses. Note that Chainsaw displays the poorest performance in term of delivery ratio.

6.6.3 PALMS-SP : Comparison of Different Streaming Rate

We also examine the correlation between the delivered quality and the streaming rate. Figure 6.10 shows that as the streaming rate increases, the delivery ratio for PALM-SP remains at a relatively high delivery ratio. Even when the streaming rate reaches 500Kbps, its delivery ratio still remain above 80%. In summary, the result reveals that the network capacity of PALMS-SP is sufficient to support the streaming session with streaming rate of 250–500Kbps. As shown in Fig. 6.10, PALMS-SP has 2% gains of delivery ratio compared to PALMS, 5% gains of delivery ratio as compared to DONet and over 11% gains as compared to Chainsaw.

6.6.4 PALMS-SP : Comparison of Data Overheads

PALMS-SP manages the membership for all nodes in a fully distributed fashion. Each node keeps partial information about other nodes, and gossips these information to preferentially chosen neighbors. Group membership management protocols are crucial to multicasting. They provide applications with the dynamic membership information. Due to large overhead, tradi-

Group Size	Control Overheads (control traffics/video traffics)		Average Delivery Ratio	
	PALMS	PALMS-SP	PALMS	PALMS-SP
100	0.0173	0.0180	0.91	0.93
200	0.0175	0.0182	0.90	0.92
300	0.0183	0.0185	0.88	0.92
400	0.0184	0.0191	0.88	0.91
500	0.0196	0.0201	0.89	0.91
1000	0.0232	0.0243	0.89	0.92
2000	0.0244	0.0253	0.90	0.92
4000	0.0296	0.0312	0.90	0.92

Table 6.2: Comparison of Control Overheads/Delivery Ratio for PALMS-SP & PALMS

tional protocols are not suitable for large-scale P2P networks. As PALMS-SP employs employs a light-weight gossip protocol, most control messages in PALMS-SP are for exchanging data availability information between connected neighbors and super-peers. The number of nodes in a group thus becomes a key factor to the control overhead.

In this section, we compare the overheads of PALMS-SP to PALMS. Table 6.2 shows that PALMS-SP incurs very low additional data overheads in comparison to PALMS. Control overhead is defined as the ratio of control traffic over video traffic. The control overheads at different overlay nodes increase log-arithmically with the increase in group size. The control overheads for PALMS-SP are slightly higher due to the additional messages such as *Super-Peer Packet Map* messages and NACKs. However the amount of increase at each overlay node is essentially minor, less than 3% of the total overall traffic. We believe the data overheads for PALMS-SP can be further reduced by increasing the window size. It can be observed that the control overhead has little relationship with the group size because each node only communicates with its neighbors and super-peer, which demonstrates the good scalability of our proposed protocol.

6.7 Summary

In this chapter, we presented PALMS-SP, a two-layer super-peer based P2P system for live media streaming. Our system's innovative and simple features are designed with the usage of the combination push-pull protocol and the presence of two-layer super-peer based overlay network that leverages on the heterogeneity of connected nodes. In order to successfully deploy PALMS-SP streaming services, we proposed push-pull mechanism to address the issue of delivery quality and delivery latency. In this framework, the existence of super-peers improves delivered video quality by incorporating the proactive and the reactive push packets mechanism. We discussed the key design issues of PALMS-SP, and proposed an intelligent scheduling push-pull algorithm, which enables efficient streaming for medium-to-high-bandwidth contents with low control overhead.

We evaluated the performance of PALMS-SP in comparison to PALMS, DONet and Chainsaw. Based on simulation results conducted over Network Simulator, ns-2, the performance of PALMS-SP is quite acceptable for live media streaming. Its control overhead is reasonably low, which is around 3% of the video traffic, and this ratio remains unchanged with an increase of the overlay size. As compared to other existing streaming protocols, the playback continuity of PALMS-SP is much better, particularly under highly dynamic environments with nodes leaving and joining at a high rate.

Our results also show that PALMS-SP has an easy and fast startup time, due both to its intrinsic simplicity. PALMS-SP also delivers quite a good playback quality even under formidable

network conditions *i.e.*, heterogeneity of network bandwidths, different user arrival/departure patterns, different network sizes, and different streaming rates. □

CHAPTER 7

Conclusions

A conclusion is the place where you got tired of thinking.
Arthur McBride Bloch. Author of Murphy's Laws

7.1 Conclusions

The previous chapters presented several methods and models based on push-pull scheduling algorithms for live streaming system. The main goal for these methods is to create a large-scale, self-organizing overlay network with simple network construction and maintenance mechanisms, and the ability to deliver high-bandwidth data streams across a highly volatile and transient node population. We also hope that these efforts are able to reduce the constraints to be satisfied by any involved node, so to avoid the exclusion of significant subsets of users with asymmetric connections lacking a sufficient upstream bandwidth, and moreover we take into the consideration of the contribution of each peer to the system. A feature we wish to develop that is resilient towards dynamics of peers arrival and departure, or *churn* and ability to serve low-rate-contributing peers normally, as long as the system capacity is under-utilized, and to gently decrease their service level when scarcity appears.

Chapter 2 introduced the background information about the basic concepts and terminology of Peer-to-Peer (P2P) networking, as a preface for understanding how a P2P streaming system is able to provide better solution for streaming multimedia over the Internet. Next, we discuss the challenges of streaming service over P2P networks as they inherent instability and unreliability. Finally, we look at several streaming protocols and projects that have been proposed as P2P overlay streaming.

Based on the recent researches and developments on P2P streaming system, we focus our research on unstructured overlay network where PALMS (P2P Unstructured Live Media Streaming) was proposed. Chapter 3 presented the main design and implementation. By coupling

push and pull streaming scheduling, we noticed improvements in the delivery quality of live streaming, as shown by the experimental results. Furthermore, we evaluated PALMS by simulations conducted using Network Simulator, ns2. We examined the impact of heterogenous bandwidth and free-riders on the performance of PALMS streaming. We also study the three metrics of interest: Delivery Quality, Delivery Latency and Data Overheads.

However there are some disadvantages for PALMS. Mainly PALMS faces two main problems - high volume of traffic and lack of monitoring. This because PALMS is based on data-driven receiver-based P2P overlay network. Similar with swarm-like content delivery mechanism of BitTorrent-style networks where all participants share resources in an unstructured P2P networks. These approaches, although similar in nature, each have their own distinct disadvantages, especially when considered in relation to a scientific research community utilizing volunteer resources. However, the swarm-like content could leads to high volume of traffic. A client could send consecutive requests for packet lists to connected neighbors. A possible solution for this problem would be to have a group peers to monitor the requests made by connected nodes. Due to the distributed nature of unstructured overlay network of PALMS, lack of monitoring is one of the disadvantages of the PALMS approach. PALMS unstructured overlay networks have multiple peers that send multiple traffic to other peer. This may introduces extra data overhead for retransmits, communication and redundancy if no proper monitoring system is employed.

The extension proposed in chapter 6 is part of the solutions to address this issue. An extension to PALMS, termed as PALMS-SP - a super-peer based two-payer P2P overlay network that focuses on the monitoring aspect to improve the latency between peers and delivered streaming quality of live media streaming. PALMS-SP is based on simple features that are designed with the usage of the combination push-pull protocol and the presence of two-layer super-peer based overlay network that leverages on the heterogeneity of connected nodes.

7.2 Directions for Future Research

The work in this dissertation has opened up some interesting avenues for future research. Some of these ideas directly extend the PALMS architecture while others touch upon the application of our work to other research problems.

7.2.1 Planet Lab

PlanetLab is a group of computers available as a testbed for computer networking and distributed systems research. It was established in 2002 and as of October 2007 was composed of 825 nodes at 406 sites worldwide. Each research project has a "slice", or virtual machine access to a subset of the nodes. With PlanetLab, it is possible to test the performances of PALMS and PALMS-SP under actual distributed systems.

7.2.2 Network Coding

Network coding is a field of information theory and coding theory and is a method of attaining maximum information flow in a network. The core notion of network coding is to allow mixing of data at intermediate network nodes. A receiver sees these data packets and deduces from them the messages that were originally intended for that data sink. In order for PALMS to achieve better streaming quality, network coding could improve overall P2P network performance.

7.2.3 Simplified Streaming Scheduling Algorithm

As part of the enhancement for PALMS and PALMS-SP, we hope to evaluate other streaming scheduling algorithm in order to improve content delivery and reduce end-to-end delay. We plan to extend the idea of quality adaptation to other congestion control schemes and investigate the implications of the details of rate adaption on our mechanism. □

Credits for Illustrations

Figure 2.2: Reproduced from [YJC07]
Figure 2.4: Adapted from [YJC07]
Figure 2.5: Adapted from [THT04]
Figure 2.6: Adapted from [YJC07]
Figure 2.9: Adapted from [ZLLY05]

The figures not listed here were originally created for this work.

Publications

Journals

- Poo Kuan Hoong, Hiroshi Matsuo, A Two-Layer Super-Peer based P2P Live Media Streaming System, *Journal of Convergence Information Technology*, vol.2, no. 3, p.47–57, September 2007.

- Poo Kuan Hoong, Hiroshi Matsuo, Push-Pull Two-layer Super-Peer based P2P Live Media Streaming, *Journal of Applied Sciences*, 2008. vol.8, no. 4, p. 585–595, January 2008.

International Conferences

- Poo Kuan Hoong, Hiroshi Matsuo PALMS : Reliable P2P Live Media Streaming. In: *The 2007 IAENG International Conference on Communication Systems and Applications*, Hong Kong. Proceedings..., IAENG Publication, 2007, p.1230–1237.

- Poo Kuan Hoong, Hiroshi Matsuo A Super-Peer based P2P Live Media Streaming System. In: *International Conference on Internet and Multimedia Technologies 2007*, Berkeley, San Francisco, USA. Proceedings..., IAENG Publication, 2007, p.639–647.

Best Paper Award

- Poo Kuan Hoong, Hiroshi Matsuo A Super-Peer based P2P Live Media Streaming System. In: *International Conference on Internet and Multimedia Technologies 2007*, Berkeley, San Francisco, USA. Proceedings..., IAENG Publication, 2007, p.639–647.

Scholarship and Research Grant

- Monbukagakusho Scholarship from the Ministry of Education, Culture, Sports, Science and Technology, Government of Japan, from April 2004 to March 2008.

- Research grant from the Hori Information Science Promotion Foundation, from June 2006 to May 2007.

Bibliography

[18095] FIPS 180-1. Secure Hash Standard, US Dept. of Commerce NIST, National Technology Information Service. 1995.

[ABKM01] D. Andersen, H. Balakrishnan, M. Kaashoek, and R. Morris. Resilient overlay networks. *In Proc. of ACM Symposium on Operating Systems Principles (SOSP)*, 2001.

[aim] In re Aimster Copyright Litigation, 334 F.3d 643 (7th Cir. 2003).

[aka] Akamai technologies inc, "Delivery a better Internet," [online] available:. *http://www.akamai.com/*.

[ANS] American National Standard for Telecommunications, Telecom Glossary 2000 T1.523-2001," 2001.

[BBK02] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable application layer multicast. *Proceedings of ACM SIGCOMM 2002*, August 2002.

[BLBS06] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan. Resilient multicast using overlays. *IEEE/ACM Trans. Netw.*, 14(2):237–248, 2006.

[CDK+03] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. Splitstream : High-bandwidth multicast in cooperative environments. *SOSP '03: Proceedings of the 19th ACM symposium on Operating systems principles, Bolton Landing, NY. USA*, pages 298–313, October 2003.

[Coh03] B. Cohen. Incentives build robustness in bittorrent. Berkeley, CA, 2003.

[CRZ00] Y.-H. Chu, S. G. Rao, and H. Zhang. A case for end system multicast. In *SIGMETRICS '00: Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 1–12, New York, NY, USA, 2000. ACM Press.

[DBGM01] H. Deshpande, M. Bawa, and H. Garcia-Molina. Streaming live media over peers. *Tech. Rep. 2001-31, CS Dept, Stanford University*, 2001.

[DCKM04] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: a decentralized network coordinate system. *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 15–26, 2004.

[GAA03] E. Gurses, G. Bozdagi Akar, and N. Akar. Selective frame discarding for video streaming in TCP/IP networks. *Packet Video Workshop, Nantes, France*, April 2003.

[GKM01] A.J. Ganesh, A.-M. Kermarrec, and L. Massoulie. SCAMP: peer-to-peer lightweight membership service for large-scale group communication. *In Proc. 3rd Intnl. Wshop Networked Group Communication (NGC'01)*, LNCS 2233:44–55, 2001.

[GKM03] A. J. Ganesh, A.-M. Kermarrec, and Laurent Massoulié. Peer-to-Peer Membership Management for Gossip-Based Protocols. *IEEE Transactions on Computers*, 52(2):139–149, 2003.

[gnu] Gnutella. (2003) the annotated gnutella protocol specification v0.4. Online. available: http://rfc-gnutella.sourceforge.net/developer/stable/index.html.

[GST03] Y. Guo, K. Suh, and D. Towsley. A peer-to-peer on-demand streaming service and its performance evaluation. In *Proceedings of 2003 IEEE International Conference on Multimedia & Expo (ICME 2003)*, volume 2, pages 649–652, Baltimore, MD, July 2003.

[HC04] A. Habib and J. Chuang. Incentive mechanism for peer-to-peer media streaming. *IWQOS 2004: 12th International Workshop on Quality of Service*, pages 171–180, 2004.

[HHL06] Z. J. Haas, J. Y. Halpern, and L. Li. Gossip-based ad hoc routing. *IEEE/ACM Transactions on Networking*, 14(3):479–491, 2006.

[HHR+03] M. Hefeeda, A. Habib, R. Rotev, D. Xu, and B. Rhargava. PROMISE: peer-to-peer media streaming using collectcast. In *MULTIMEDIA '03: Proceedings of the eleventh ACM international conference on Multimedia*, pages 45–54, Berkeley, CA, USA, 2003. ACM Press.

[HHS98] T. W. Haynes, S. Hedetniemi, and P. Slater. *Fundamentals of Domination in Graphs*. CRC Press, January 1998.

[HM79] G. Handler and P. Mirchandani. *Location on networks: Theory and algorithms*. MIT Press, Cambridge, 1979.

[JDXB03] X. Jiang, Y. Dong, D. Xu, and B. Bhargava. Gnustream: a P2P media streaming system prototype. *ICME '03: Proceedings of the 2003 International Conference on Multimedia and Expo*, pages 325–328, 2003.

[JGJ+00] J. Jannotti, D. Gifford, K. Johnson, M. Kaashoek, and J. OŤoole. Overcast: Reliable multicasting with an overlay network. *Proc. the Fourth Symposium on Operating Systems Design and Implementation*, pages 197–212, 2000.

[kaz] KaZaa.com. kaZaa. http://www.kazaa.com/us/index.htm.

[KLL+97] D. Karger, E. Lehman, F. Leighton, M. Levine, D. Lewin, and R. Panigrahy. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web. *In Proceedings of the 29th Annual ACM Symposium on Theory of Computing, El Paso, TX*, pages 654–663, 1997.

[KLW01] B. Krasic, K. Li, and J. Walpole. The case for streaming multimedia with TCP. *Lecture Notes in Computer Science, LNCS 2158*, pages 213–218, January 2001.

[KRAV03] D. Kostic, A Rodrigues, J. Albrecht, and A. Vahdat. Bullet: high bandwidth data dissemination using an overlay mesh. *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 282–297, October 2003.

[KSG02] M. Kalman, E. Steinbach, and B. Giro. Adaptive playout for real-time media streaming. *in IEEE International Symposium on Circuits and Systems, ISCAS-2002*, 1:45–48, May 2002.

[KSGM03] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The Eigentrust algorithm for reputation management in P2P networks. *Proceedings of the 12th International World Wide Web Conference (WWW), ACM Press (Budapest, Hungary)*, (640–651), May 2003.

[LBCL99] P. Liu, S. Battista, F. Casalino, and C. Land. MPEG-4: a multimedia standard for the third millennium, part 1. *IEEE on Multimedia*, 6(4):74–83, October 1999.

[LZL+05] V. Lo, D. Zhou, Y. Liu, C. GauthierDickey, and Jun Li. Scalable Supernode Selection in Peer-to-Peer Overlay Networks. *HOT-P2P Ó5 : Proceedings of the Second International Workshop on Hot Topics in Peer-to-Peer Systems*, pages 18–27, 2005.

[mgm] MGM Studios, Inc. v. Grokster, ltd. 545 u.s. 913 (2005).

[MOV96] A. J. Menezes, P. C. V. Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, October 1996.

[nap] A & M Records. inc. v. Napster. inc. 114 f. supp. 2d 896, 921 (n. d. cal. 2000).

[ns2] Network simulator, ns-2. [online] available:. *http://www.isi.edu/nsnam/ns/*.

[NZ02] T.S.E. Ng and Hui Zhang. Predicting Internet network distance with coordinates-based approaches. *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 1:170–179, 2002.

[OWS+02] J. Ott, S. Wenger, N. Sato, C. Burmeister, and J. Rey. Extended RTP profile for RTCP-based feedback (RTP/AVPF). draft-ietf-avt-rtcp-feedback-04.txt, October 2002.

[PKT+05] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A. Mohr. Chainsaw: Eliminating trees from overlay multicast. *4th International Workshop on Peer-to-Peer Systems*, February 2005.

[PRR97] C. G. Plaxton, R. Rajaraman, and A. W. Richa. Accessing nearby copies of replicated objects in a distributed environment. *SPAA '97: Proceedings of the ninth annual ACM symposium on Parallel algorithms and architectures*, pages 311–320, 1997.

[PWC03] V. N. Padmanabhan, H. J. Wang, and P. A. Chou. Resilient peer-to-peer streaming. *Network Protocols*, pages 16–27, 2003.

[PWC04] V. N. Padmanabhan, H. J. Wang, and P. A. Chou. Supporting Heterogeneity and Congestion Control in Peer-to-Peer Multicast Streaming. *The 3rd International Workshop on Peer-to-Peer Systems (IPTPS 2004), San Diego*, pages 54–63, 2004.

[PWCS02] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai. Distributing streaming media content using cooperative networking. *NOSSDAV '02: Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video*, pages 177–186, 2002.

[RD01] A. I. T. Rowstron and P. Druschel. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. *Middleware '01: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*, pages 329–350, 2001.

[RFH+01] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 161–172, 2001.

[RHKS02] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-aware overlay construction and server selection. *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings*, 3:1190–1199, 2002.

[RLM+02] J. Rey, D. Leon, A. Miyazaki, V. Varsa, and R. Hakenberg. RTP retransmission payload format. draft-ietf-avt-rtp-retransmission-04.txt, December 2002.

[RO03] R. Rejaie and A. Ortega. PALS: peer-to-peer adaptive layered streaming. *NOSSDAV '03: Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video*, pages 153–161, 2003.

[SC96] H. Schulzrinne and S. Casner. RFC 1890: RTP profile for audio and video conferences with minimal control, January 1996.

[Sch95] B. Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons., 2nd edition, October 1995.

[sky] Skype - the global internet telephony company. http://www.skype.org/.

[SMK+01] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 149–160, 2001.

[THT04] Duc A. Tran, Kien A. Hua, , and Tai T. A peer-to-peer architecture for media streaming. *IEEE Journal on Selected Areas in Communications*, 22(1):121–133, January 2004.

[TZ99] Wai-Tian Tan and A. Zakhor. Real-time Internet video using error resilient scalable compressionand TCP-friendly transport protocol. *IEEE Transactions on Multimedia*, 1(2):172–186, June 1999.

[VChS03] A. Vetro, C. Christopoulos, and huifang Su. Video Transcoding Architectures and Techniques: An Overview. *IEEE Signal Processing Magazine*, 20(2):18 –29, March 2003.

[YGM03] B. Yang and H. Garcia-Molina. Designing a super-peer network. *IEEE International Conference on Data Engineering (ICDE'03), Bangalore, India*, pages 49–60, 2003.

[YJC07] W. P. K. Yiu, X. Jin, and S. H. Gary Chan. Challenges and Approaches in Large-Scale P2P Media Streaming. *IEEE MultiMedia*, 14(2):50–59, 2007.

[YM04] S. Ye and F. Makedon. Collaboration-aware peer-to-peer media streaming. *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, pages 412–415, 2004.

[Zeg96] E. W. Zegura. GT-ITM: georgia tech internetwork topology models (software). *http://www.cc.gatech.edu/project*, 1996.

[ZHJK04] B.Y. Zhao, L. Huang, A.D. Joseph, and J.D. Kubiaotowicz. Tapestry: a resilient global-scale overlay for service deployment. *EEE Journal on Selected Areas in Communications (JSAC)*, pages 41–53, January 2004.

[ZL04] D. Zhou and V. Lo. Cluster Computing on the Fly: resource discovery in a cycle sharing peer-to-peer system. *IEEE International Symposium on Cluster Computing and the Grid, 2004. CCGrid 2004*, pages 66–73, 2004.

[ZLLY05] X. Zhang, J. Liu, B. Li, and T.P. Yum. CoolStreaming/DONet : A data-driven overlay network for efficient live media streaming. *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, 3:2102–2111, 2005.

[ZZJ01] S. Q. Zhuang, B. Y. Zhao, and A. D. Joseph. Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. *NOSSDAV '01: Proceedings of the 11th international workshop on Network and operating systems support for digital audio and video*, pages 11–20, June 2001.